Н.Г. Хитрово

Начала

системного программирования

в среде MS-DOS7

2009

ББК 32.973.26-018.2
УДК 004.451.9dos

Н.Г.Хитрово

Начала системного программирования в среде MS-DOS7

Эта книга преимущественно адресована молодым людям, связывающим свои планы на будущее с освоением системных функций компьютеров. Чтобы стать понятной для неподготовленного читателя, книга начинается с изложения элементарных сведений и включает ряд простых примеров, но основное ее содержание составляют справочные данные профессионального уровня, часть которых опубликована на русском языке впервые. Они будут полезны преподавателям и студентам ВУЗов соответствующего профиля, специалистам по обслуживанию компьютеров, системным программистам, а также разработчикам оборудования на микропроцессорах с совместимой системой команд.

Содержание

Предисловие

В наше время "окна" Windows-2000/XP стали привычным элементом интерьера. В них можно работать и развлекаться. Создается впечатление, что сквозь них доступно все. Однако впечатление обманчиво: те глубинные процессы, которые порождают все компьютерные чудеса, из "окон" не видны.

Каждому "продвинутому" пользователю однажды становится ясно, что дальше корректирования параметров реестра операционная система Windows никого не пустит. Для тех, кто стремится понять компьютер глубже, все дальнейшие пути ведут за рамки "окон" операционной системы Windows. Но там, за "окнами" – темно, и кажется, будто не на что опереться. Если все же Вы намерены сделать следующий шаг, то эта книга – для Вас.

0.01 Прикладное и системное программирование.

Понятно, что все компьютерные "чудеса" сотворены разными программами, каждая из которых играет свою роль. Программы бывают прикладные и системные. Например, текстовый редактор WORD – типичная прикладная программа, потому что ей для работы необходима среда операционной системы Windows. А системными называют те программы, которые образуют, конфигурируют и обслуживают ту самую операционную среду.

Сейчас много внимания уделяют прикладному программированию. Фирма Microsoft – разработчик операционной системы Windows – пропагандирует свой программный пакет VISUAL STUDIO, включающий несколько языков программирования высокого уровня. Интерес фирмы прост: чем больше будет написано прикладных программ, нуждающихся в операционной среде Windows, тем больше экземпляров этой операционной системы удастся продать.

Отношение к системным сведениям совсем другое: фирмы не заинтересованы в том, чтобы кто-нибудь со стороны мог составить конкуренцию их продуктам. Право купить некоторые коммерческие тайны предоставляют только надежным партнерам при условии неразглашения, причем настолько дорого, что не каждая фирма может себе это позволить. Такая информационная политика, понятно, не способствует освоению системного программирования. Тем не менее было бы наивно думать, что можно стать специалистом по компьютерной технике без понимания роли системных функций и принципов их осуществления.

К системным программам относятся не только сами операционные системы, но и драйвера различных устройств, прошивки BIOS (базовой системы ввода-вывода), а также сервисные, диагностические и восстановительные программы, к помощи

которых приходится прибегать в случае отказа основной операционной системы компьютера. Все они построены и совершенствуются посредством системного программирования. Обучение решению таких задач всегда было важным элементом профессионального образования по компьютерным специальностям. Как правило, системное программирование преподают на базе низкоуровневого языка ассемблера (MASM или TASM) в среде документированных версий дисковой операционной системы фирмы Microsoft (MS-DOS).

К нашему времени все документированные версии MS-DOS устарели настолько, что не обеспечивают даже доступа к дискам большого объема, используемым в современных компьютерах. А современные операционные системы защищают себя и аппаратные средства компьютера, отвечая на каждую попытку прямого вызова системных функций сообщением: "Ваша программа совершила недопустимую операцию и будет закрыта". Тем не менее, выход есть: существуют операционные среды, которые способны обеспечить прямое исполнение системных операций и могут быть инсталлированы на современных компьютерах. Цель данной книги состоит как раз в том, чтобы научить Вас работе с такой операционной средой, а также решению несложных системных задач в этой среде, причем без премудростей языка ассемблеров MASM или TASM.

0.02 Режимы работы: защищенный и реальный.

Защита от исполнения недопустимых операций аппаратно обеспечивается защищенным режимом работы процессоров. Помимо собственно защиты, он предоставляет ряд важных преимуществ, и потому стал основным для всех современных операционных систем. При защищенном режиме все действия прикладных пользовательских программ и самих пользователей ограничены виртуальной оболочкой, не позволяющей нарушать жизненно важные функции операционной системы. Именно это явилось главным фактором в достижении той повышенной устойчивости функционирования, которая отличает операционные системы Windows-2000 и Windows-XP.

В противоположность тому реальный режим – это "беззащитный" режим эмуляции команд давно устаревшего процессора 8086. Казалось бы, пора уже отправить этот анахронизм на покой. Но проходят годы, сменяют друг друга поколения процессоров, а он по-прежнему жив. Поддержка реального режима сохраняется, потому что он, как оказывается, необходим, и притом не для архаичных, а для самых современных систем. Реальный режим необходим базовой системе ввода-вывода (BIOS), которая иначе не смогла бы "инвентаризировать" и тестировать аппаратные средства компьютера. Именно с этой целью центральный процессор автоматически устанавливается в реальный режим при включении электропитания. Операционным системам, даже самым современным, для

обретения "власти" над ресурсами компьютера также необходимо стартовать не в защищенном, а в "беззащитном" реальном режиме.

Получив управление в свои "руки", операционные системы серии Windows подготавливают переход в защищенный режим так, что после перехода высший (нулевой) уровень привилегий получает сама система Windows, а пользователям и прикладным программам достается низший (третий) уровень привилегий. Из-за этого изменить "расклад прав" пользователь уже никогда не сможет. По той же причине в защищенном режиме все возможности за пределами круга API-функций системы Windows становятся недоступны для прикладных программ.

Сейчас обычная практика пользователя вообще не предполагает знакомства с работой за пределами низшего уровня привилегий. Доминирует противоположная концепция разграничения пользовательских и системных сфер разумения. Считается, что избыточное любопытство пользователя априори опасно и должно быть пресечено. Современные операционные системы хорошо защищены защищенным режимом ... от Вас. Каковы же альтернативы?

0.03. Почему MS-DOS 7 ?

Если операционная система не ограничивает права пользователя низшим уровнем привилегий, то при прочих равных условиях она неизбежно проигрывает по стабильности и надежности. В этом одна из причин вытеснения старомодных операционных систем типа DOS более современными. Но ситуация радикально изменяется, когда дело доходит до системных задач, которые нельзя решить без предоставления специалисту неограниченных прав доступа. Тогда тот же недостаток DOS становится решающим преимуществом и делает ее незаменимой. Все загрузочные дискеты – диагностические, сервисные, восстановительные, которые требуют наличия операционной системы, используют именно DOS. DOS применяется также в загрузочных оптических дисках. Наконец, DOS намного проще других операционных систем, и потому освоение системных операций целесообразно начинать именно с нее.

Распространено мнение, что DOS – операционная система реального режима. Это верно лишь отчасти: DOS начинает работать в реальном режиме, но может быть переведена в защищенный режим либо драйвером (5.04-02), либо самим пользователем, причем тогда Вы сами вправе присвоить высший уровень привилегий себе. Только в таком случае процессор будет повиноваться любым Вашим командам, в том числе тем, которые исполняются лишь на высшем уровне привилегий. Ни одна другая операционная система свой уровень привилегий Вам не отдаст. Только среда DOS обеспечит Вам полную свободу действий как в реальном, так и в защищенном режиме.

Практическая потребность в операционной среде реального режима вынуждает заинтересованных поставщиков программного обеспечения продолжать совершенствование DOS. Независимо от ведущих фирм продвигается работа над проектом FreeDOS (http://www.freedos.org/). Недавно появилась новая платная версия ROM-DOS (http://www.datalight.com/), Находят своего потребителя также несколько менее "свежих" разновидностей DOS. Написано большое количество драйверов, придающих устаревшим официальным версиям DOS (MS-DOS 6.22, PC DOS 2000 и др.) новые способности, в том числе доступ к дискам с файловыми системами FAT-32 и NTFS, которые сейчас широко распространены. Но драйверы, не включенные в состав ядра операционной системы, не дают возможности устанавливать ее на такие диски.

Если бы Вы рискнули приобрести компьютер без покупки операционной системы, то обнаружили бы в нем типичную строку приглашения DOS и жесткий магнитный диск с файловой системой FAT-32. Но ни одна из упомянутых выше версий DOS, скорее всего, не имела бы к этому никакого отношения. Начальную инициализацию компьютеров обычно выполняют с помощью средств, поставляемых фирмой Microsoft на загрузочных дискетах к операционным системам Windows-95 OSR2 или Windows-98. Официально это называется "загрузкой в командную строку".

Несложно убедиться, однако, что "загрузка в командную строку" представляет собой типичную версию DOS. Внутри почти каждого файла на загрузочной дискете имеется строка: "MS-DOS Version 7...". Седьмая версия MS-DOS – это и есть та самая недокументированная операционная среда фирмы Microsoft, которая представляет для нас наибольший интерес. Если попытаться уточнить номер версии DOS с помощью функции INT 21\AH=30h (8.02-22), то она возвратит значение 07.0Ah, то есть в десятичной форме 7.10. Далее в этой книге при всех упоминаниях MS-DOS7 будет иметься ввиду именно эта версия.

MS-DOS7 – это не последняя из версий MS-DOS. В комплекте поставки операционной системы Windows-ME содержится восьмая версия MS-DOS. Она перекомпилирована под современные процессоры и стала более компактной, но при этом утратила совместимость с некоторыми вариантами 486-х процессоров. Кроме того, в процессе загрузки основной операционной системы Windows-ME она не играет активной роли и не позволяет строить сценарии загрузки. Однако во всех других отношениях MS-DOS7 и MS-DOS8 очень близки, так что почти все материалы данной книги в равной мере применимы к ним обоим. Везде, где свойства MS-DOS7 и MS-DOS8 не совпадают, их отличия специально оговорены.

0.04 О чем эта книга.

Системное программирование - обширная тема, которая не вмещается в разумные пределы объема книги. Потому эта книга не претендует на полноту: несколько больших разделов (например, сети) пришлось исключить. Ряд других вопросов изложен лишь настолько, насколько это достаточно для понимания.

Главы 1 - 4 этой книги знакомят читателя с клавиатурой, с композицией командных строк и со встроенными командами. Эти короткие главы адресованы тем, кому приходится осваивать MS-DOS7 "с нуля".

Глава 5 представляет наиболее важные драйверы для компьютерного оборудования, в том числе самые современные, разработанные в течение 2004 – 2008 гг. Глава 6 – это обзор полезных программ (утилит) для MS-DOS7. В нем много внимания уделено главному инструменту программирования в MS-DOS7 – отладчику DEBUG.EXE. Ассемблерным командам отладчика посвящена отдельная глава 7. Затем глава 8 описывает различные служебные функции, которые доступны в MS-DOS7 через прерывания.

В главе 9 представлены конкретные примеры системного программирования с использованием средств из стандартной поставки WINDOWS-95/98 для написания сервисных программ и составления конфигурационных файлов. Примеры помогут Вам создать аналогичные сервисные и конфигурационные файлы применительно к Вашим задачам, а также покажут разнообразие возможностей, которые MS-DOS7 открывает перед теми, кто умеет ее правильно попросить.

Завершают книгу 16 тематических приложений. Они включают много таблиц со справочными данными, относящимися как собственно к MS-DOS7, так и к аппаратуре AT-совместимых компьютеров. Последнее приложение – словарь с объяснениями всех англоязычных аббревиатур, встречающихся в этой книге.

0.05 Еще несколько замечаний.

Ведущие фирмы давно пытаются навязать пользователям операционные системы без лазеек для доступа к реальному режиму. Но ни OS/2 (IBM, 1989), ни Windows-NT (Microsoft, 1994) популярными не стали. Очередная попытка с Windows-2000 открыла возможность использования средств MS-DOS7 на дисках с файловой системой FAT-32. Как только успех Windows-2000 стал очевиден, тотчас последовало решение прикончить весь конкурирующий выводок версий Windows-95/98/ME. Однако это решение не устранило необходимости в реальном режиме и не предложило новых инструментов для работы в нем. Изменилось другое: теперь больше нет причин ожидать появления такой новой версии DOS, которая смогла бы обесценить Ваши усилия по освоению MS-DOS7.

В 2002 году поводом для новых ожиданий предстоящего обновления стала разработка компанией Intel процессора Itanium, не поддерживающего старомодный 16-битный машинный код. Все прежние программные средства реального режима, включая MS-DOS7, утратили бы свое значение с появлением новых компьютеров, имеющих 32-битный код BIOS, и тогда эту книгу не стоило бы писать. Но с тех пор прошло уже 7 лет, а ожидаемое чудо не случилось. Массовые персональные компьютеры с процессором Itanium так и не появились. Следует признать, что сохранение поддержки 16-битного кода во всех более новых процессорах имеет под собой весомые основания. И пока эти основания имеются, умение пользоваться DOS не перестанет быть актуальным и полезным для Вас.

Как запустить MS-DOS7? Если у Вас уже установлены Windows-95 OSR2 или Windows-98, то просто: придерживайте нажатой клавишу F8 в момент начала загрузки операционной системы, и тогда на экран будет выведено загрузочное меню. В нем надо выбрать вариант загрузки в командную строку ("command prompt only") – и готово. Если у Вас операционная система типа Windows-95/98 не установлена, то достаньте загрузочную дискету к ней, и загрузите компьютер с этой дискеты. Подходящие образы загрузочных дискет выложены на многих сайтах сети Интернет, например, на сайте http://www.bootdisk.com/ . После стандартной загрузки MS-DOS7 предстанет перед Вами в виде пустой и неуютной командной строки. Но в разделах 6.25 и 9.01 этой книги Вы найдете примеры того, как сделать экран DOS гораздо более информативным, удобным и дружественным. А в разделе 9.11 дано детальное описание разных вариантов запуска MS-DOS7, включая ее установку на диск с операционной системой Windows-XP.

Эта книга не основана на официальных материалах по MS-DOS7, так как они, насколько известно, не опубликованы до сих пор. Во всем, что здесь написано, повинен только автор. Несмотря на искреннее стремление проверить все-все-все, 100%-гарантии достоверности приводимых сведений автор дать не может: аппаратурная база проверок ограничена, располагаемые ресурсы не бесконечны. Кроме того, с течением времени многое изменяется – от компьютерного "железа" до адресов в Интернете. Из-за таких обстоятельств автор не может нести ответственность за последствия Ваших действий, даже если они инспирированы этой книгой. Любое дело, сопряженное с риском, требует осторожности и умения. Однако есть надежда на то, что благодаря этой книге Ваши действия смогут стать умелыми, безопасными и результативными.

Ваши замечания по содержанию книги просьба высылать по адресу For-H@yandex.ru. Все они будут с благодарностью приняты.

Глава 1. Знакомство с клавиатурой в DOS

Для всех версий DOS основное устройство ввода – клавиатура. Нажатие каждой ее клавиши вызывает исполнение по крайней мере одного заранее загруженного в память (резидентного) программного модуля. Иногда исполнение такого модуля вообще никак не проявляет себя, но чаще сводится к индикации на экране соответствующего клавише символа. Однако некоторые клавиши, называемые активными (или "горячими" = "hot" keys), инициируют сложные последовательности вложенных вызовов. Резидентные программные модули, вызываемые по нажатию клавиши, могут быть загружены в память

- базовой системой ввода-вывода (BIOS);
- загрузчиком DOS (действующим только в процессе загрузки);
- драйвером консоли (т.е. клавиатуры и экрана) в составе ядра DOS;
- командным интерпретатором (обычно это файл COMMAND.COM).

В этой главе не описывается действие других программных модулей, которые могут быть загружены устанавливаемыми драйверами, файл-менеджерами (Norton Commander, Volcov Commander и т.п.) или другими резидентными программами. Загружаясь позже, они могут перехватывать вызовы и тем самым заменять действия клавишей или подменять их частичной эмуляцией. Командная строка, которую представляют файл-менеджеры, уже не эквивалентна исходной командной строке DOS: значительная часть исходных функций клавиатуры перехвачена и дезактивирована (пример – в разделе 6.25).

Следующие разделы первой главы представляют функции наиболее распространенной сейчас "улучшенной" ("enhanced") клавиатуры, различные модели которой насчитывают от 101 до 108 клавишей. Эволюция функций описана в той последовательности, в которой клавиши активизируются в процессе загрузки MS-DOS7. Конечной стадией эволюции предполагается та, которая формируется в результате загрузки командного интерпретатора COMMAND.COM.

1.01 Клавиши вызова операций системы BIOS

В момент включения электропитания компьютер начинает работать под управлением базовой системы ввода-вывода (BIOS). С самого начала BIOS загружает обработчик прерывания INT 09 (8.01-09), воспринимающий вызовы контроллера клавиатуры по линии IRQ 1. Благодаря этому BIOS начинает чувствовать каждое нажатие любой клавиши, но только некоторые клавиши вызывают отклик. Многие версии BIOS начинают реагировать на нажатие следующих клавишей и их комбинаций:

Ctrl-Alt-Delete – инициирует перезагрузку компьютера.

Delete – запускает программу BIOS Setup, позволяющую указать параметры для системы BIOS (примечания 1 и 2).. Клавиша "Delete" автоматически дезактивируется примерно через 2 секунды, так что при необходимости ее надо держать нажатой с момента включения компьютера.

Pause/Break (или Ctrl-Break) – вызывает временный останов загрузки до нажатия какой-либо другой клавиши, позволяя тем самым прочитать выведенные на экран сообщения.

Shift-PrtScr– посылает содержимое буфера экрана на принтер, подключенный к порту LPT1. Принтер должен быть включен и должен быть способен к взаимодействию с системой BIOS. Принтеры с интерфейсом USB, а также разработанные под WINDOWS ("designed for WINDOWS") для этого не годятся.

Единого стандарта на клавишные функции систем BIOS не существует; в разных компьютерах их реализация может несколько отличаться. Например, BIOS версии 8.01 фирмы American Megatrends дополнительно по нажатию клавиши F8 выводит меню альтернатив загрузки. Тем не менее некоторые функции фактически стали стандартом для всех современных версий BIOS: в частности, функции клавиши Delete (примечание 2) и клавишной комбинации Ctrl-Alt-Delete.

Активные клавиши, установленные системой BIOS, могут быть дезактивированы, как любая программно реализованная функция. В частности, комбинация Shift-PrtScr нередко дезактивируется программой, "зашитой" в постоянное запоминающее устройство видеокарты. Другие активные клавиши, установленные системой BIOS, обычно не дезактивируются намеренно, но тем не менее могут пострадать в результате сбоев, приводящих к потере или искажениям данных в таблице прерываний (от 0000:0000h до 0000:0400h) или в области данных BIOS (A.01-1). Поэтому наиболее важная функция выхода в перезагрузку во многих компьютерах аппаратно дублирована кнопкой RESET на лицевой панели системного блока.

Примечание 1: в самой программе BIOS Setup могут быть активизированы другие клавиши – это зависит от версии BIOS. Ими приходится пользоваться, когда нужно изменять настройки программы BIOS Setup. Одна из этих настроек, в частности, позволяет запретить выведение логотипа BIOS, чтобы он не скрывал результаты стартовых тестов.

Примечание 2: в некоторых (преимущественно старых) компьютерах запуск программы BIOS Setup может осуществляться клавишами F1, F2, F10, ESC или клавишными комбинациями F3-F2, Ctrl-Alt-S, Ctrl-Alt-Ins, Ctrl-Alt-Esc.

Примечание 3: на многих "улучшенных" клавиатурах выпуска 1990-х годов имелась клавиша TURBO, причем комбинация TURBO-F11

включала и выключала блокировку клавиатуры, а комбинация TURBO-F12 включала и выключала блокировку звуковых сигналов. На современных клавиатурах клавиша TURBO обычно отсутствует.

Примечание 4: на многих моделях клавиатур имеются особые клавиши управления электропитанием компьютера: POWER, SLEEP и WAKE UP. Они также предназначены для обслуживания системой BIOS, но эти функции бывают "перехвачены", в частности, операционными системами Windows-XP и Windows Vista.

1.02 Клавиши вызова операций загрузчика DOS

В ходе загрузки наступает момент, когда на экране дисплея логотип BIOS сменяется логотипом операционной системы Windows-95/98. Это значит, что система BIOS завершила свою начальную миссию и передала управление загрузчику операционной системы. В частности, у операционных систем Windows-95/98 и MS-DOS7 первичный загрузчик один и тот же. Он входит в состав файла IO.SYS и начинает свою деятельность со считывания параметров загрузки DOS из файла MSDOS.SYS (5.01-01), а затем в соответствии со считанными параметрами временно активизирует еще несколько "горячих" клавишей и клавишных комбинаций:

F5 – изменяет режим последующей загрузки WINDOWS на безопасный (Safe Mode), при котором графическая оболочка (GUI) загружается с установками по умолчанию, а конфигурационные файлы (CONFIG.SYS and AUTOEXEC.BAT) полностью игнорируются.

SHIFT-F5 – загрузка в режим командной строки ("command prompt only" mode), т.е. процесс завершается по окончании нормальной загрузки MS-DOS7, далее система WINDOWS не загружается.

F6 – изменяет режим последующей загрузки WINDOWS на безопасный (Safe Mode), так же как и клавиша F5, но дополнительно устанавливает службы сетевой поддержки.

F8 – вызывает индикацию стандартного загрузочного меню системы WINDOWS и останавливает процесс загрузки до выбора пользователем пункта в этом меню. В версии MS-DOS8 действие клавиши F8 дублируется клавишей CTRL.

SHIFT-F8 – переводит исполнение командных строк в конфигурационных файлах в пошаговый режим; это позволяет избежать исполнения отдельных строк по выбору пользователя.

Перечисленные функции "горячих" клавишей в MSDOS7 и MSDOS8 отличаются от тех, которые использовались в предшествовавших версиях DOS. На количество активизируемых "горячих" клавишей и длительность поддержания их в

активном состоянии влияют параметры (BOOTDELAY, BOOTKEYS, BOOTMULTI), указываемые в файле MSDOS.SYS (5.01-01). Когда MS-DOS7 используется как отдельная операционная система, и пакет программ, составляющий собственно систему WINDOWS, вообще недоступен, тогда, конечно, загрузка графической оболочки системы WINDOWS не может быть выполнена, и нажатия соответствующих клавишей будут проигнорированы.

Если пользователь успеет вовремя нажать клавишу F8, то на экране появится стандартное загрузочное меню операционной системы Windows. Пока загрузочное меню остается на экране, загрузчик воспринимает нажатия клавишей управления курсором вверх-вниз, клавиши ENTER и числовых клавишей (0 – 9) в основной части клавиатуры. Если переключатель NUMLOCK (4.23) включен, клавиши числового набора в правой части клавиатуры позволяют выбирать пункт меню по номеру. Когда выбор сделан, все связанные с меню клавиши дезактивируются. Если выбранным пунктом меню не предусмотрено иное, то затем загрузчик переходит к исполнению команд из строк конфигурационного файла CONFIG.SYS (пример – в разделе 9.01-01).

При пошаговом исполнении команд из файла CONFIG.SYS загрузчик запрашивает подтверждения, на которые можно отвечать нажатием клавишей Y (= yes), N (= no), ENTER (= yes) или A (= yes для всех последующих строк). Нормальное (не-пошаговое) исполнение команд файла CONFIG.SYS по умолчанию не контролируется, потому что экран дисплея в это время продолжает воспроизводить логотип операционной системы Windows. Однако выведение логотипа можно запретить, указав параметр "Logo=0" в файле MSDOS.SYS (подробнее – в разделе 5.01-01). Тогда на экране будут видны быстро смещающиеся сообщения от загружаемых драйверов. Чтобы внимательно рассмотреть эти сообщения, процесс исполнения команд можно остановить нажатием клавиши PAUSE/BREAK или комбинации CTRL-S. Затем после нажатия любой клавиши процесс исполнения продолжится.

Закончив интерпретацию всех строк конфигурационного файла CONFIG.SYS, загрузчик DOS дезактивирует задействованные им "горячие" клавиши и передает управление командному интерпретатору COMMAND.COM.

Примечание 1: в версии 7.00 MS-DOS загрузчик DOS активизировал клавишу F4 для загрузки той DOS, "поверх" которой была установлена операционная система Windows-95. Но начиная с версии 7.10 возможность загрузки старой DOS не поддерживается.

1.03 Активные клавиши при исполнении командных файлов

Интерпретатор COMMAND.COM принимает управление на себя, когда процедура загрузки резидентных программ еще не завершена. Его первая миссия

состоит в интерпретации строк последнего конфигурационного файла AUTOEXEC.BAT (пример – в разделе 9.01-02). Считывая команды из файла, интерпретатор действует вполне самостоятельно, но при этом роль пользователя все же не сведена к нулю. Во-первых, отказавшись от воспроизведения на экране логотипа Windows (5.01-01), можно просматривать сообщения, выводимые по ходу исполнения команд из файла AUTOEXEC.BAT. Во-вторых, имеются активные клавиши, посредством которых можно временно остановить или вообще прекратить исполнение любого командного файла.

Клавиши и клавишные комбинации, продолжающие оставаться активными во время исполнения командных файлов, инициируют довольно сложную последовательность вызовов (подробнее – в разделе 8.01-95), в которую вовлечены резидентные модули ядра DOS и обработчики прерываний, установленные системой BIOS. Из-за этого в каждом конкретном компьютере перечень активных клавишей и характер их действия могут зависеть от версии системы BIOS. Тем не менее в большинстве AT-совместимых компьютеров задействуемые "горячие" клавиши одинаковы.

Клавиша BREAK (Pause) вызывает временный останов исполнения, позволяет прочитать индицируемые сообщения, но не дает возможности прекратить исполнение командного файла: после нажатия любой клавиши исполнение возобновляется.

Прервать исполнение командного файла можно с помощью "горячих" клавишных комбинаций CTRL-C, CTRL-BREAK и ALT-03, причем цифры 03 в комбинации ALT-03 должны быть набраны в группе цифровых клавишей в правой части клавиатуры. Действие этих клавишных комбинаций зависит от того, как был запущен на исполнение сам интерпретатор COMMAND.COM. Если интерпретатор был запущен с параметрами /K или /P (6.04), то перечисленные клавишные комбинации останавливают исполнение и предлагают выбор, прекратить исполнение или нет:

```
"Terminate batch job? Y/N"
```

Но когда интерпретатор COMMAND.COM запущен для исполнения одиночной задачи с параметром /C (6.04), как обычно это делают файл-менеджеры, то клавишные комбинации CTRL-C, CTRL-BREAK и ALT-03 прекращают исполнение сразу, не давая шанса на продолжение.

Комбинация CTRL-S вызывает временный останов, но всегда дает шанс продолжить исполнение после нажатия любой клавиши, кроме комбинаций CTRL-C, CTRL-BREAK, ALT-03 и CTRL-2. Последние действуют в зависимости от условий запуска интерпретатора COMMAND.COM, как было описано выше, но комбинация CTRL-2 выделяется тем, что прекращает исполнение только если оно

уже было приостановлено клавишной комбинацией CTRL-S. Нормальный процесс исполнения командных файлов не прерывается комбинацией CTRL-2.

Функции клавиатуры, прерывающие исполнение командных файлов, могут быть заблокированы командой CTTY NUL (3.07), указываемой в одной из строк исполняемого командного файла. Но пользование такой блокировкой бывает оправдано только в особых обстоятельствах (пример – в разделе 9.03-02).

Закончив исполнение файла AUTOEXEC.BAT, интерпретатор COMMAND.COM должен будет передать управление загрузчику "графической оболочки" операционной системы Windows – файлу WIN.COM. Но это не произойдет, если

– файл WIN.COM не будет найден (например, при загрузке с дискеты);
– в момент начала загрузки была нажата комбинация Shift-F5 (1.02);
– в загрузочном меню (1.02) выбран пункт "Command prompt only";
– в файле MSDOS.SYS (5.01-01) указан параметр "BootGUI=0";
– в команде DOS указан параметр SINGLE (примечание 1 к 4.08).

В любом из перечисленных пяти случаев вместо операционной системы Windows будет загружена MS-DOS7, интерпретатор COMMAND.COM перейдет в режим ввода командных строк с клавиатуры, покажет на экране свое "приглашение", и с этого момента начнет работать с клавиатурой совсем по-другому.

1.04 Ввод командных и текстовых строк

Когда интерпретатор показывает приглашение командной строки, он принимает ввод с клавиатуры через драйвер консоли (устройства CON). Последний позволяет вводить буквы, цифры и специальные символы в соответствии с таблицей раскладки клавиатуры, заранее загруженной в память компьютера. Символы можно вводить как с помощью соответствующих клавишей, так и по номеру в стандартном коде ASCII: для этого надо, придерживая клавишу ALT, набрать номер на группе цифровых клавишей в правой части клавиатуры. Каждый следующий символ добавляется к имеющимся и одновременно увеличивает на единицу указатели текущей позиции как командной строки, так и внутреннего буфера, в котором автоматически запоминается предыдущая командная строка.

Выборка символов, набираемых с помощью буквенных клавишей, зависит от положения управляющих (регистровых) клавишей SHIFT и CAPSLOCK. Нажатие клавиши CAPSLOCK переключает клавиатуру в верхний регистр заглавных букв и обратно. Клавиша SHIFT тоже переключает клавиатуру в верхний регистр, но только на то время, пока она не отпущена. DOS предоставляет ограниченные возможности редактирования содержания вводимых строк посредством функциональных клавишей, описываемых в разделе 1.05.

Все перечисленные особенности действительны также при вводе текстовых строк. Переход к вводу текстовых строк происходит в результате обращения к драйверу консоли (устройства CON), например, с помощью следующей команды:

```
COPY CON FILENAME.TXT
```

здесь FILENAME.TXT – произвольное имя файла, куда надлежит записать вводимые строки текста.

Команда COPY является не отдельной программой, а встроенной командой командного интерпретатора COMMAND.COM (3.06). Другие программы также могут вызывать режим ввода текстовых строк, если способны обращаться для ввода к драйверу консоли.

Различия режимов ввода командных и текстовых строк проявляются в том, что происходит с набранной строкой, когда пользователь подтверждает окончание набора нажатием клавиши ENTER или эквивалентной клавишной комбинации CTRL-M. При вводе текстовых строк происходит запись строки во внутренний буфер, замещение там предыдущей строки и открытие для ввода новой строки. Конец введенной строки отмечается двумя байтами 0Dh 0Ah, и в таком виде она присоединяется к области памяти, где записаны все предшествовавшие строки, образуя тем самым многострочный текст. Интерпретатор COMMAND.COM позволяет послать этот текст в файл или в указанный канал (подробнее – в разделе 3.06).

Возвращение обратно к режиму ввода командных строк производится нажатием комбинаций клавишей F6-ENTER, CTRL-Z-ENTER или ALT-26-ENTER (в последней комбинации цифры "26" должны быть набраны в группе цифровых клавишей в правой части клавиатуры).

При вводе командных строк нажатие клавиши ENTER также вызывает запись строки во внутренний буфер, но потом события разворачиваются иначе: происходит разбор содержимого буфера с целью выделения имени команды и выяснения, является ли она встроенной командой интерпретатора или именем отдельной программы (утилиты). Для отдельных программ (утилит) проводится их поиск (2.02-02), после чего программа считывается с носителя и подготавливается к исполнению. Потом управление передается подготовленной программе, а когда исполнение завершается, управление возвращается командному интерпретатору, который выводит на экран новое приглашение командной строки и начинает очередной цикл ожидания ввода команды пользователем.

Если строка содержит вызов на исполнение командного файла, то нажатие клавиши ENTER вызывает переход интерпретатора в режим построчного считывания команд из этого файла. Когда исполнение командного файла завершится, интерпретатор вернется к обычному вводу командных строк с клавиатуры.

Конкретное содержание командной строки, конечно, зависит от того, какому интерпретатору она адресована, и представляет собой тему для отдельного разговора, который начнется с главы 2 и будет продолжаться на протяжении всех последующих глав этой книги.

1.05 Клавиши редактирования вводимых строк

Самой "горячей" клавишей при вводе строк, конечно, является клавиша ENTER, роль которой была показана в предыдущем разделе 1.04. Помимо нее, однако, при вводе строк действуют еще несколько клавишей и клавишных комбинаций, наделенных особыми функциями редактирования строк и управления. Оба интерактивных командных интерпретатора в DOS (COMMAND.COM и DEBUG.EXE) наследуют эти функции, имеющие долгую историю в предшествующих поколениях компьютеров. Некоторые из них весьма архаичны, но некоторые до сих пор активно используются.

Клавиша BACKSPACE (левая стрелка) уменьшает на 1 значение указателя позиции как для текущей строки, так и для внутреннего буфера. Последний знак строки перестает воспроизводиться на экране, и это знакоместо готово принять следующий вводимый знак. Клавишные комбинации CTRL-H и ALT-08, левая стрелка среди клавишей управления курсором и левая стрелка в группе цифровых клавишей в правой части клавиатуры действуют так же.

Комбинации CTRL-2, CTRL-C, CTRL-BREAK и ALT-03 аннулируют текущую строку и открывают для набора новую пустую строку. При построчном вводе текста пропадает весь набранный к тому моменту текст.

Комбинации CTRL-J, CTRL-ENTER и ALT-10 сворачивают текущую строку и дают возможность набирать ее оставшуюся часть в следующем ряду знакомест. Это позволяет видеть набираемую строку полной длины (до 128 знаков) на экране, ширина которого обычно составляет 80 знакомест. Свернутая строка воспринимается и исполняется так же, как непрерывная.

Комбинации CTRL-G и ALT-07 вводят код 07h ("Звуковой сигнал"). В текстовых строках он себя не проявляет, но когда строка с этим кодом подается для воспроизведения на экране, то раздается короткий звуковой сигнал (beep).

Комбинации CTRL-P и ALT-16 переключают вывод данных с дисплея на принтер и обратно. Это бывает опасно, если принтер не готов к работе, не подсоединен или подключен не к порту LPT1. В каждом таком случае на экран выводится сообщение "Abort,

Retry?" ("Отменить, Повторить?"), но выбор альтернативы "Abort" ("Отменить") не восстанавливает исходное состояние. Чтобы вернуться в командную строку, нужно нажать CTRL-P еще раз, иначе вывод сообщения "Abort, Retry?" будет продолжаться.

Клавиша DEL (DELETE) при редактировании строк влияет только на внутренний буфер: увеличивает на 1 значение указателя позиции в нем. Это воспринимается как смещение предыдущей строки в буфере на одно знакоместо влево. Если после нажатия клавиши DEL (DELETE) производится копирование в текущую строку, то один знак из предыдущей строки оказывается пропущен.

Клавиша ESC, а также комбинации CTRL-ESC, CTRL-[и ALT-27 сбрасывают текущую строку без сохранения ее в буфере, закрывают ее изображение на экране знаком обратной косой черты (\ ), и открывают для набора новую пустую строку. Предыдущая строка в буфере сохраняется. Перечисленные действия не вызваны знаком обратной косой черты, его можно вводить в строку наравне с другими знаками.

Клавиша F1 дополняет текущую строку одним знаком, считываемым из той же позиции в предшествующей строке, которая хранится во внутреннем буфере. Правая стрелочка среди клавиш управления курсором и правая стрелочка в группе цифровых клавиш в правой части клавиатуры действуют так же.

Клавиша F2 вызывает паузу ожидания ввода одного знака. Следующий вводимый знак на экране не отображается. Если этот знак не содержится в оставшейся части предыдущей строки во внутреннем буфере, то нажатие клавиши F2 аннулируется без последствий; но если введенный знак там присутствует, то группа знаков, предшествующих введенному, копируется из внутреннего буфера в текущую строку.

Клавиша F3 копирует предыдущую строку из внутреннего буфера на свободное пространство текущей строки. Если текущая строка уже содержит несколько знаков, то они сохраняются, и знаки из соответствующих знакомест внутреннего буфера не выводятся.

Клавиша F4 вызывает паузу ожидания ввода одного знака. Следующий вводимый знак на экране не отображается. Если этот знак не содержится в оставшейся части предыдущей строки во внутреннем буфере, то нажатие клавиши F4 аннулируется без последствий; но если введенный знак там присутствует, то предшествующие ему знаки удаляются из буфера. Содержимое буфера сдвигается влево, так что введенный знак занимает то же положение в буфере, на котором находится курсор в текущей

строке. Это позволяет предотвратить копирование фрагмента предыдущей строки при нажатии клавиш F1 или F3.

Клавиша F5 копирует текущую строку во внутренний буфер, закрывает ее на экране символом "at" (@) и открывает для набора новую пустую строку. Копируемая строка не исполняется и не присоединяется к тексту. Перечисленные действия не вызваны символом "at" (@), его можно использовать в строке наравне с другими знаками.

Клавиша F7 и комбинация ALT-00 вводят код 00h, который маркирует конец интерпретируемой части командной строки. Последующие вводимые знаки при интерпретации игнорируются.

Клавиша INS (INSERT) изменяет на противоположное состояние бита по адресу 0040:0017 (A.02-3), который останавливает приращение значения указателя позиции во внутреннем буфере при вводе знаков в текущую строку. Если Вы скопировали часть предыдущей строки, затем остановили приращение указателя клавишей INS, ввели несколько знаков с клавиатуры, а потом копируете оставшуюся часть предыдущей строки, то введенные с клавиатуры знаки оказываются вставленными между знаками предыдущей строки. Для восстановления нормального приращения указателя нужно нажать клавишу INS еще один раз.

Клавиша TAB, а также комбинации CTRL-I или ALT-09 вводят код 09h горизонтальной табуляции. При выдаче этого кода для индикации на экране он автоматически заменяется на 8 пробелов. При записи текста в файл код 09h пробелами не заменяется. Некоторые программы редактирования текстов сами заменяют код 09h пробелами, и тогда количество пробелов на один код табуляции может быть не равно 8.

Действия перечисленных клавишей с функциями редактирования бывают часто перехвачены и подменены резидентными программами, которые загружаются позже командного интерпретатора. В частности, файл-менеджеры (Norton Commander, Volcov Commander, и др.) обычно перехватывают действия клавишей INS, DEL, F1 – F7 и некоторых других, дезактивируют их или наделяют иными функциями (6.25). Тем не менее, перечисленные исходные функции всегда остаются действующими при построчном вводе текста, а также при интерактивной работе с интерпретатором DEBUG.EXE.

1.06 Раскладки клавиатуры и индицируемые символы

По умолчанию MS-DOS7 использует американский (US) набор знаков, определяемый кодовой страницей 437, но дает возможность установить другую

кодовую страницу с каким-либо национальным набором знаков. Реализация этой возможности требует исполнения ряда операций по изменению настроек клавиатуры и знакогенератора, формирующего изображение для воспроизведения на экране. Фирма Microsoft предлагает следующую последовательность операций:

- изменить ограничения на выбор знаков для использования в именах, а также ряд других настроек путем загрузки данных из файла COUNTRY.SYS (5.02-01, пример – в разделе 9.01-01);
- подготовить место в памяти для одной или нескольких дополнительных кодовых страниц посредством загрузки драйвера DISPLAY.SYS (5.02-02, пример – в разделе 9.01-01);
- запустить программу MODE.COM, чтобы загрузить желаемые кодовые страницы и сделать одну из них активной (6.18, пример – в разделе 9.01-02);
- запустить программу KEYB.COM, чтобы загрузить избранную альтернативную раскладку клавиатуры (5.02-04, пример – в разделе 9.01-02);
- загрузить резидентный модуль программы NLSFUNC.EXE (5.02-03), чтобы обеспечить оперативное переключение между подготовленными кодовыми страницами.

В поставке операционной системы WINDOWS-95 национальные кодовые страницы упакованы в четыре файла данных: EGA.CPI, EGA2.CPI, EGA3.CPI и ISO.CPI. Каждая национальная кодовая страница содержит 256 знаков, в том числе два отдельных набора знаков: американский (знаки 32 – 127) и какой-либо другой (с номерами знаков от 128-го и выше). Поэтому переключение между знаками национального языка и американскими не требует смены кодовых страниц, оно выполняется в пределах одной кодовой страницы. Этого бывает вполне достаточно для того ограниченного круга задач, которые решают сейчас с помощью MS-DOS7. По той же причине в оперативном переключении кодовых страниц обычно необходимости нет.

Переключение между различными наборами знаков в пределах одной кодовой страницы выполняется с помощью "горячих" клавишей, активизируемых резидентным модулем драйвера KEYB.COM (5.02-04) или какого-либо другого (например, KEYRUS.COM, 5.02-05). В частности, KEYB.COM активизирует клавишную комбинацию CTRL-правыйSHIFT для переключения на национальный набор знаков и комбинацию CTRL-левыйSHIFT для переключения к американскому набору знаков. KEYRUS.COM позволяет в тех же целях активизировать различные клавишные комбинации, в том числе и упомянутые.

Фирма Microsoft поставляет таблицы раскладки клавиатуры упакованными в три файла: KEYBOARD.SYS, KEYBRD2.SYS, KEYBRD3.SYS. Среди этих файлов только KEYBOARD.SYS содержит раскладки, соответствующие клавиатурам пишущих машинок. Подробности выбора раскладки клавиатуры и кодовой

страницы для конкретной страны приведены в приложении A.02-2. Пример использования того варианта национальной адаптации, который предложила фирма Microsoft, показан в разделах 9.01-01 и 9.01-02.

Хотя раскладки клавиатуры и кодовые страницы, разработанные фирмой Microsoft, подходят для большей части стран мира, они не поддерживаются более и не сделаны открытыми для внесения изменений. В связи с этим к настоящему времени для DOS разработаны не менее пяти других драйверов клавиатуры. В этой книге внимание уделено только одной из альтернатив – драйверу KEYRUS.COM (5.02-05), который представляет собой открытую систему, поставляемую совместно со средствами для создания новых раскладок клавиатуры и для внесения изменений в кодовые страницы. К сожалению, предложенный фирмой Microsoft формат раскладок клавиатуры и кодовых страниц не подходит для KEYRUS.COM. Примеры национальной адаптации с помощью драйвера KEYRUS.COM показаны в разделах 9.04-01 и 9.09-01.

Глава 2 Командная строка

Командная строка в MS-DOS начинается с автоматически формируемого приглашения и затем должна быть заполнена знаками и словами, которые все вместе должны удовлетворять условиям машинной интерпретации в соответствии с определенными правилами. Завершающее нажатие клавиши ENTER запускает процедуру интерпретации посредством специальной программы – командного интерпретатора, причем синтаксические правила для разных интерпретаторов не одинаковы.

В MS-DOS7 имеются 3 интерпретатора: IO.SYS, COMMAND.COM, и DEBUG.EXE. Каждый из них использует свой набор команд, описываемый в главе 4 для IO.SYS, в главе 3 для COMMAND.COM и в разделе 6.05 для DEBUG.EXE. Поскольку загрузчик IO.SYS интерпретирует только строки конфигурационных файлов, перед пользователем первой открывается командная строка, представляемая основным резидентным интерпретатором COMMAND.COM. Этот интерпретатор называют резидентным, потому что он постоянно находится в памяти компьютера и всегда готов исполнить команды, которые пользователь введет с клавиатуры.

Посредством перенаправления ввода (подробнее – в разделе 2.04-02) можно заставить интерпретатор воспринимать команды не с клавиатуры, а из командных файлов, в которых каждая строка представляет собой отдельную командную строку. Пересылка командных файлов через перенаправление ввода – единственный способ автоматизировать исполнение сложных последовательностей операций интерпретатором DEBUG.EXE (примеры – в разделе 9.02).

Интерпретатор COMMAND.COM также принимает командные файлы через перенаправление ввода, но для него этот путь – не единственный и не лучший, потому что он способен принимать к исполнению особый вид командных файлов (batch-файлы) без перенаправления. Из batch-файлов интерпретатор COMMAND.COM исполняет несколько дополнительных операций, которые из обычных командных файлов и прямо с клавиатуры не исполняются. К числу таких операций относятся подмены имен переменных и формальных параметров их значениями (подробнее – в разделе 2.03-03), поиск меток, а также команды вызова, перехода и некоторые другие (3.02, 3.14, 3.21, 3.27). Здесь и далее в этой книге термин "batch-файлы" обозначает только такой особый вид командных файлов. Примеры batch-файлов показаны в разделе 9.03. Конфигурационный файл AUTOEXEC.BAT (9.01-02, 9.04-02, 9.09-02) – это тоже типичный batch-файл.

Далее во второй главе неформально описаны основные соглашения, определяющие композицию командных строк, как вводимых с клавиатуры, так и

строк командных файлов. В некоторой степени эти соглашения являются общими для всех трех интерпретаторов в MS-DOS7. У каждого из них, конечно, есть свои особенности, и они отмечены тоже. Но если не оговорено иное, то подразумеваются примеры командных строк для основного интерпретатора – COMMAND.COM.

2.01 Назначение имен

Каждая командная строка представляет один или несколько объектов. Объектами могут быть, в частности, встроенная команда или исполняемый файл, который должен осуществить желаемое действие. Принимаемые во внимание сведения об объекте должны быть достаточны для того, чтобы точно адресовать именно данный объект. С этой целью не допускается назначение одинаковых имен файлам (и подкаталогам), находящимся в одном каталоге. Точная адресация файлов и каталогов обеспечивается тем, что вместе с их именами учитывается путь ним, указываемый явно или принимаемый по умолчанию.

Существуют объекты – встроенные команды, порты и некоторые другие, адресация которых не может быть уточнена указанием пути. Таким объектам для обеспечения однозначной идентификации должны быть присвоены уникальные имена, выражаемые зарезервированными словами.

Помимо имени основного объекта и пути к нему, командная строка может содержать иные элементы, включая имена других объектов, параметры, ссылки, синтаксические знаки и т.п. Каждая строка в командных файлах, представленных в разделе 9.03, может быть взята в качестве примера командной строки.

Первое условие правильности композиции командной строки – это правильная адресация желаемых объектов по их именам. Поэтому с самого начала надо разобраться, какие слова могут быть использованы в качестве имен объектов в командных строках, а какие нет.

2.01-01 Зарезервированные слова

Зарезервированными словами являются имена встроенных команд интерпретатора, а также устройств, заявленных как имеющиеся в конкретном компьютере. Пользователь не вправе сам назначать и изменять имена таких объектов, но должен их знать хотя бы для того, чтобы не предпринимать попыток назначения этих имен другим объектам, которые он вправе именовать.

Встроенные команды – это те, которые командный интерпретатор выполняет сам. Все имена встроенных команд, приведенные в главе 3, будут считаться

зарезервированными словами, если исполнять данную командную строку будет интерпретатор COMMAND.COM. Например, файл нельзя назвать именем PROMPT, потому что интерпретатор COMMAND.COM, встретив такое имя, "поймет" его как вызов его собственной команды (3.22) и, конечно, попытается ее выполнить. Любые интерпретаторы аналогичным образом не позволяют присваивать имена своих встроенных команд другим объектам.

Зарезервированные имена устройств в компьютере относятся к источникам получения данных или адресатам для посылки данных. Наиболее известны следующие имена устройств:

AUX – 1-й последовательный порт
COM1 – 1-й последовательный порт (эквивалент имени AUX)
COM2 – 2-й последовательный порт
CON – консоль, т.е. клавиатура для ввода и дисплей для вывода
LPT1 – 1-й параллельный порт, обычно для подключения принтера
NUL – виртуальный порт вывода "в никуда"
PRN – 1-й параллельный порт (эквивалент имени LPT1)

Помимо перечисленных, зарезервированными именами устройств считаются CLOCK$, COM3, COM4, CONFIG$, LPT2, LPT3. Эти слова резервируются драйверами, входящими в состав ядра DOS, которые загружаются всегда, даже если обслуживаемое драйвером устройство в конкретном компьютере физически отсутствует. Полный список заявленных в Вашем компьютере имен логических устройств можно посмотреть с помощью программы MEM.EXE (6.17), если ее запустить с параметром /D. Будет выведена таблица, в 4-м столбце которой показаны имена, причем зарезервированные имена логических устройств показаны со смещением на 3 знакоместа вправо относительно других имен.

Некоторые устанавливаемые программные драйверы также могут быть адресованы по имени и резервируют слова с этой целью. Например, драйвер SETVER.EXE (5.01-02) резервирует имя SETVERXX, драйвер HIMEM.SYS (5.04-01) резервирует имя XMSXXXX0, драйвер EMM386.EXE (5.04-02) резервирует имя EMMXXXX0. Более того, когда Вы указываете произвольный идентификатор для оптического дисковода, например, MSCD001 (5.09-01, 5.09-02, 5.09-03), это имя регистрируется DOS и приобретает статус зарезервированного слова. Если потом Вы попытаетесь присвоить это имя файлу, DOS отвергнет Вашу попытку.

2.01-02 Имена и суффиксы

Операции назначения имен и переименования наиболее часто применяются по отношению к каталогам (3.19, 6.20) и к файлам (3.24, 3.25), как к обычным, так и к исполняемым (утилитам). Имена в DOS должны содержать не более 8 знаков и могут быть дополнены суффиксом (расширением), содержащим не более 3 знаков и отделяемым от имени знаком точки. Когда имя имеет суффикс, но он для конкретной операции не требуется, и потому в командной строке не указан, то разделяющий знак – точку – не следует указывать тоже.

В составе имен и суффиксов можно использовать буквы, цифры и некоторые знаки, не имеющие специальной синтаксической миссии: знак числа (#), знак доллара ($), амперсанд (&), дефис (–), восклицательный знак (!), знак подчеркивания (_) и несколько других (в частности, 2.04-06). Заглавные и строчные буквы рассматриваются в MS-DOS7 как эквивалентные почти во всех операциях, за исключением наименований сопоставляемых элементов в строках вызова команд IF (3.15-02), SEARCH (6.05-16) и программы FIND.EXE (6.14).

Поскольку точка служит для разделения имени и суффикса, постольку ее нельзя использовать в них как обыкновенный знак. В именах и суффиксах нельзя также использовать запятую (,), двоеточие (:), точку с запятой (;), знак равенства (=), вопросительный знак (?), плюс (+), левую стрелку (<), правую стрелку (>), звездочку (*), вертикальную черту (|), косую черту (/), обратную косую черту (\ ) и двойные кавычки ("). Знаки национальных алфавитов нельзя использовать, если ограничения на них не сняты командой COUNTRY (4.05, 5.02-01 и A.02-5).

Имена каталогов обычно суффиксов не имеют. Суффиксы в именах файлов служат для указания типа файла или его происхождения. Три суффикса - BAT, COM и EXE – имеют особый статус, поскольку интерпретатор COMMAND.COM считает файлы с такими суффиксами исполняемыми по умолчанию. Когда имя файла с одним из этих суффиксов оказывается первым в командной строке, то интерпретатор COMMAND.COM автоматически ставит его на исполнение. Если же в таком файле окажется не исполняемый код, а что-либо иное, то компьютер почти наверняка зависнет. Поэтому суффиксы BAT, COM и EXE не следует присваивать файлам, которые не являются специально предназначенными и заведомо пригодными для исполнения.

Присвоение других суффиксов не настолько критично, но тем не менее обычно выполняется в соответствии с рядом общепринятых правил. Файл-менеджеры способны связывать файлы с определенными суффиксами с теми программами, к которым эти файлы относятся. Например, файл с суффиксом BAS может быть автоматически послан на исполнение интерпретатору языка BASIC. Такая "привязка" файлов к соответствующим программам оказывается очень удобной (примеры в 6.25-03, 6.25-04). Знакомство с общепринятой практикой назначения

суффиксов также бывает полезно для визуального опознавания типов файлов по суффиксу. С этой целью ниже приведен краткий перечень ассоциаций, обычно приписываемых наиболее употребительным суффиксам в среде DOS.

BAK – архив или старая версия файла
BAT – batch-файл для интерпретатора COMMAND.COM
BIN – исполняемый файл, требующий фиксированного размещения
BMP – файл растрового изображения (Bit-Map Picture)
CAB – сжатый файл для поставок программного обеспечения
COM – исполняемый файл без заголовка
CPI – файл с несколькими таблицами шрифтов для DOS
DAT – файл с различными не-текстовыми данными
DOC – форматированный текстовый файл редактора WORD
DLL – динамически связываемая библиотека исполняемых кодов
EXE – исполняемый файл с управляющими параметрами в заголовке
EXT – файл спецификаций функционального расширения
GIF – файл графического изображения (Graphic Image File)
HTM – файл, написанный на языке HyperText Markup language
INI – файл с установочными параметрами для инициализации
JPG – файл изображения, сжатый согласно спецификации JPEG
RAR – сжатый архивный файл, созданный программой RAR.EXE
RTF – текстовый файл формата Rich Text Format
SCR – командный файл (обычно для интерпретатора DEBUG.EXE)
SYS – системный файл, текстовый или исполняемый
TMP – временный файл
TXT – неформатированный текстовый файл
ZIP – сжатый архивный файл, созданный программой PKZIP.EXE

Ассоциации для множества других суффиксов можно найти в сети интернет на сайте http://www.openwith.org/ .

2.01-03 Маски и знаки подстановки

Вопросительный знак (?) и звездочка (*) являются знаками подстановки, которые нельзя использовать в именах и суффиксах, но тем не менее можно включать в состав спецификаций имен и суффиксов в командных строках. Такие спецификации, в которых один или несколько знаков заменены знаками подстановки, называются маской файла и являются средством адресации нескольких файлов сразу.

При разборе командной строки, содержащей маску файла, интерпретатор COMMAND.COM может вызывать функцию раскрытия маски, которая осуществляет поиск файлов, имена которых соответствуют данной маске. Если такой файл удается найти, то его имя подставляется в командную строку вместо маски, и в таком виде командная строка подается на исполнение. Если таких файлов оказывается несколько, то аналогичная процедура подстановки и исполнения последовательно производится по отношению к каждому из найденных файлов.

Будут ли слова со знаками подстановки "раскрываться" – это зависит от исполняемой команды: некоторые команды вызывают упомянутую функцию раскрытия маски, а некоторые нет. В большинстве случаев "раскрытие" производится, но имеются следующие исключения:

– маски файлов не раскрываются встроенными командами ECHO, SET, TYPE и IF с условием равенства (описанными в главе 3);
– знаки подстановки среди параметров batch-файлов не раскрываются (при этом не важно, вызывается ли batch-файл командой CALL или исполняется непосредственно);
– в спецификации перенаправления ввода (2.04-02) знаки подстановки не раскрываются;
– команда FOR (3.13) раскрывает маски файлов только внутри скобок, а вне скобок знаки подстановки передаются исполняемой в цикле программе, и от нее зависит их дальнейшая судьба.

Вопросительный знак (?) представляет собой знак подстановки, который побуждает процедуры сопоставления выдавать положительное решение на любую одиночную букву или цифру. Если за вопросительным знаком не следует еще хотя бы один знак (кроме точки и вопросительного знака), то процедуры сопоставления также будут выдавать положительное решение на отсутствие какого-либо знака в этом месте. Например, в команде

```
DEL readme.??
```

маска будет соответствовать всем файлам с именем "readme" и суффиксом, содержащим не более двух знаков: readme.ru, readme.en, readme.f, и т.д. В результате исполнения такой команды DEL (3.09) все такие файлы из текущего каталога будут удалены.

Звездочка (*) представляет собой знак подстановки, который побуждает процедуры сопоставления давать положительное решение на любое количество последующих букв или цифр до конца слова, отмеченного точкой или пробелом. Например, маска файла в следующей команде

```
DEL C:\TEMP\*.*
```

будет соответствовать любому имени файла с любым суффиксом, так что команда DEL с такой маской удалит все файлы из указанного каталога.

Многочисленные примеры использования масок файлов приведены в разделах 2.02-03 и 9.09-02, а также в других batch-файлах в главе 9.

Примечание 1: при поиске соответствующих маске файлов могут быть приняты во внимание атрибуты файла, но это зависит от команды: например, команда ATTRIB (6.01) выводит по маске все файлы без исключения.

Примечание 2: интерпретаторы IO.SYS и DEBUG.EXE не раскрывают знаки подстановки. При интерпретации конфигурационного файла CONFIG.SYS загрузчиком IO.SYS вопросительный знак принимается не как знак подстановки, а как метка выдачи подсказки (4.06, 4.07, 4.15, 4.16, 4.25).

2.02 Спецификации путей

Точное расположение каждого файла на диске указано в каталоге. Каталогов на диске обычно много, и они организованы в иерархическую структуру: каждый каталог более высокого ранга может содержать не только данные о файлах, но и сведения о расположении каталогов более низкого ранга (подкаталогов). Чтобы получить доступ к файлу, необходимо указать диск, на котором он записан, и путь к нему – то есть каталог или целую цепочку подкаталогов, ведущую к тому подкаталогу, в котором имеются сведения о расположении интересующего Вас файла.

DOS предоставляет возможность (2.04-01, 3.03) выбрать любой конкретный путь в качестве пути, принимаемого по умолчанию. Этот путь будет записан во внутреннюю таблицу DOS (A.03-3). Каждый раз, когда Вы вводите командную строку, не содержащую спецификации пути, DOS будет руководствоваться тем самым записанным в таблицу путем. Тот диск и тот самый конечный каталог (или подкаталог), которые указаны в принимаемом по умолчанию пути, принято называть текущим диском и текущим каталогом соответственно. Приглашение DOS (3.22) обычно настраивают так, чтобы оно показывало текущий путь.

2.02-01 Типовая структура спецификации пути

По умолчанию командный интерпретатор ищет адресуемый объект в принимаемом по умолчанию (текущем) каталоге на принимаемом по умолчанию

(текущем) диске. Если объект следует искать где-то в другом месте, то перед именем объекта надо указать путь, например

```
C:\DOS\MS7\Edit.com
```

здесь:

Edit.com – адресуемый исполняемый файл;
C:\DOS\MS7\ – пример спецификации пути к этому файлу.

Показанная структура спецификации пути направляет процесс поиска: сначала надо перейти в корневой каталог диска C:, там найти каталог DOS и войти в него, затем в каталоге DOS найти подкаталог MS7, и уже в последнем искать подлежащий исполнению файл. Если на Вашем диске структура каталогов иная, то надо будет указать другие имена, но принцип останется тем же: сначала буква диска, за которой следует двоеточие, потом цепочка имен каталогов, отделяемых знаком обратной косой черты, и последним указывается имя адресуемого объекта.

Спецификации, заканчивающиеся знаком обратной косой черты, воспринимаются в MS-DOS как не имеющие адресуемого объекта и потому неполные. Такие спецификации либо игнорируются (2.04-01), либо считаются ошибкой, за исключением важного частного случая – пути, сокращенного до одного знака обратной косой черты. Одиночная обратная косая черта после буквы диска и двоеточия означает путь к корневому каталогу этого диска. Например,

```
A:\
```

обозначает путь к корневому каталогу диска A:.

Пути без указания буквы диска (например: \DOS\MS7\Edit.com) отсчитываются относительно корневого каталога текущего диска независимо от того, какой диск является текущим в данный момент. Такие спецификации пути позволяют писать командные файлы, которые одинаково исполняются на любом диске (примеры в разделах 9.01-01, 9.04-01, 9.09-01). Одиночная обратная косая черта воспринимается как путь к корневому каталогу текущего диска. Команда смены каталога (3.03) с указанием одной обратной косой черты вместо пути

```
CD \
```

выполняет переход в корневой каталог текущего диска.

Когда в начале спецификации пути нет ни буквы диска, ни обратной косой черты, указывающей на корневой каталог, то такие пути отсчитываются относительно текущего каталога. Например, интерпретация пути

```
DOS\VC4\Edit.com
```

будет исходить из предположения, что "DOS" является подкаталогом в текущем каталоге, и если это не так, то адресуемый файл не будет найден, и будет выдано сообщение об ошибке.

2.02-02 Переменная окружения PATH

Для упрощения пользования командной строкой в DOS имеется еще один механизм указания путей поиска – через переменную окружения PATH. Чтобы привести этот механизм в действие, нужно выполнить 4 условия:

– адресовать команду интерпретатору COMMAND.COM;
– заранее записать пути поиска в значение переменной PATH (3.20);
– не указывать путь в командной строке;
– первым объектом в командной строке указать имя файла, который не находится в текущем каталоге.

Если все эти условия выполнены, то интерпретатор обратится к переменной окружения PATH, прочитает пути поиска в значении этой переменной, и далее будет искать запрошенный файл последовательно по каждому из прочитанных путей. Примеры спецификации путей поиска в значении переменной окружения PATH показаны в разделах 3.20, 9.01-02, 9.04-02, 9.09-02.

Благодаря переменной окружения PATH запускать программы на исполнение оказывается так же просто, как будто они всегда находятся в любом каталоге. Это очень удобно, но только до тех пор, пока Вам не доведется встретить в текущем каталоге одноименную программу из другой версии DOS. Тогда первой будет найдена именно та программа в текущем каталоге; работать в MS-DOS7 такие программы обычно отказываются. Подходы с разных сторон к проблеме предотвращения подобных конфликтов обсуждаются в разделе 5.01-02 и во вводном разделе к главе 6. Наиболее универсальное решение состоит в вызове конфликтных программ через командные или конфигурационные файлы, содержащие полные спецификации путей. Примеры таких решений приведены в разделе 9.03.

2.02-03 Использование знака точки в спецификациях пути

Знак точки (.) в спецификациях пути интерпретируется как указание на текущий каталог. Обратите внимание, например, на завершающую точку, заменяющую путь к каталогу назначения в следующей команде:

```
Copy /B A:\MyDir\*.* .
```

Иногда командные файлы (3.02) должны быть написаны так, чтобы они смогли бы выполнить свою миссию в любом каталоге, который заранее неизвестен, и тогда знак точки оказывается единственной приемлемой заменой обязательной спецификации каталога назначения. К знаку точки в качестве обозначения текущего каталога приходится прибегать в тех случаях, когда по умолчанию был

бы принят иной путь, а также когда надо намеренно избежать поиска по путям, указанным в переменной окружения PATH (2.02-02).

Если точка является первым знаком в спецификации пути, то стартовой точкой отсчета такого пути будет принят текущий каталог:

```
.\VC4\Vc.exe
```

Эта форма спецификации пути эквивалентна пути без предшествующей обратной косой черты (см. 2.02-01), но тем не менее может быть полезна при написании batch-файлов с получением спецификаций пути от файл-менеджеров или других программ, потому что более простого способа избавиться от начального знака обратной косой черты MS-DOS не предоставляет. Нередко получаемые спецификации путей заканчиваются знаком обратной косой черты, например:

```
C:\DOS\MS7\
```

Для обращения к файлу такие спецификации достаточно лишь дополнить именем файла, но в обращениях к каталогам последней обратной косой черты не должно быть, и тогда самым простым выходом является добавление конечной точки:

```
C:\DOS\MS7\.
```

– такой путь к каталогу MS7 тоже считается правильным.

Сдвоенная точка, или точка-точка (`..`) используется в спецификациях путей так же, как одиночная точка, но замещает обозначение не текущего, а вышестоящего (родительского) каталога. Если, например, получив спецификацию C:\DOS\MS7\ , Вы хотите обратиться к вышестоящему каталогу C:\DOS , то достаточно добавить к полученной спецификации два знака точки:

```
C:\DOS\MS7\..
```

При разборе спецификаций путей, содержащих знак сдвоенной точки, DOS просто отбрасывает предшествующую стадию пути (в приведенном примере \MS7\ ), не проверяя, является ли указанный там объект каталогом или файлом. Это дает возможность адресовать новый файл в заранее неизвестном каталоге на основании получаемого во время исполнения полного пути с именем другого файла, находящегося в том же каталоге. Примеры такой адресации показаны в разделе 6.25-03.

Сдвоенная точка без предшествующего пути интерпретируется как указание на вышестоящий каталог относительно текущего, например, в команде смены текущего каталога (3.03):

```
cd ..
```

Чтобы подняться на два уровня вверх по древовидной структуре каталогов, нужно в той же команде указать знак сдвоенной точки дважды:

```
cd ..\..
```

Допустимы и более сложные комбинации знаков сдвоенной точки, их вполне можно использовать для исследования структур каталогов и для навигации по ним.

2.03 Синтаксис командных строк

2.03-01 Разделительные знаки

Слова в командных строках отделяются друг от друга разделительными знаками: пробелом ( ), запятой (,), знаком равенства (=) или точкой с запятой (;). Хотя пробел используется наиболее широко, любой из перечисленных знаков игнорируется в начале командной строки и играет роль разделительного знака в операциях разбора строк, включая разбор списка объектов в команде FOR (3.13). По той же причине перечисленные разделительные знаки не передаются из параметров командной строки во внутренние формальные параметры batch-файлов (2.03-03).

Имеется, однако, несколько исключений. В командах SET (3.26) и IF (3.15) знак равенства (=) играет особую роль и потому не является обычным разделителем. Если запятая, точка с запятой или знак равенства предшествуют команде ECHO (3.11), то имя команды – слово ECHO – включается в состав воспроизводимого на экране сообщения. В команде PATH (3.20) в качестве разделителя может быть использована только точка с запятой.

При обработке строк в файлах Config.sys и Msdos.sys (5.01-01) загрузчик IO.SYS воспринимает точку с запятой совсем по-другому: если она стоит первым знаком в строке, то сразу вызывает переход к интерпретации следующей строки. Строки, начинающиеся с точки с запятой, оказываются пропущенными. Этим можно пользоваться для введения комментариев и для временного предотвращения загрузки отдельных драйверов. Аналогичным образом воспринимает точку с запятой интерпретатор DEBUG.EXE при работе в режиме ассемблирования: он игнорирует саму точку с запятой и всю оставшуюся часть строки после нее, позволяя тем самым приписывать комментарии к строкам с ассемблерными командами в командных файлах (7.01-05).

2.03-02 Косая черта

Косая черта (/) в командных строках MS-DOS показывает, что следующий за ней знак или слово следует интерпретировать как параметр. Например, в команде

```
DEL C:\TEMP\*.* /P
```

косая черта (/) означает, что следующая за ней буква P – это параметр, в данном контексте вынуждающий команду DEL (3.09) выводить запрос на разрешение удаления каждого файла. Место и форма представления параметров индивидуальны для каждой команды и должны соответствовать ее спецификациям.

Иногда косая черта (/) используется в команде FOR как функциональный разделитель, вызывающий преобразование строчных букв в заглавные (подробнее об этом – в разделе 3.13).

2.03-03 Знак процента

Знак процента (%) означает замещение имени переменной или формального параметра batch-файла значением той же переменной или того же формального параметра. Такое замещение осуществляется до исполнения интерпретатором COMMAND.COM указываемой в строке команды и перенаправлений.

Формальные параметры batch-файлов обозначаются цифрами от 0 до 9. Значением 0-го формального параметра всегда является имя самого batch-файла. Другим формальным параметрам в качестве значения присваиваются в порядке их расположения отдельные слова (параметры), которые могут быть указаны вслед за именем batch-файла в командной строке, использованной для запуска этого batch-файла на исполнение. Например, обозначение %3 в любой строке batch-файла перед исполнением этой строки будет заменено словом, которое было указано третьим после имени batch-файла в той командной строке, которая запустила этот batch-файл на исполнение. Слитно указанные обозначения (например, %2%3) приводят к конкатенации слов, которыми эти обозначения замещаются. Примеры присвоения значений формальным параметрам показаны в разделах 2.03-04 и 9.03-01.

Если реальное количество слов, следующих за именем batch-файла, было меньше трех, то обозначение %3 просто будет устранено из обрабатываемой строки без сообщения об ошибке. Если количество слов, следующих за именем batch-файла, больше 9, то доступ к остальным возможен посредством сдвига нумерации формальных параметров командой SHIFT (3.27). Среди batch-файлов один является исключением: файл Autoexec.bat (например, 9.01-02), автоматически

запускаемый на исполнение при загрузке компьютера, может не иметь значений формальных параметров, так что вызов

```
CALL %0
```

в строке файла Autoexec.bat обычно не приводит к его рекурсивному исполнению (3.02).

Именами переменных окружения могут служить любые слова, начинающиеся с буквы (но не с цифры!). Значения переменных либо присваиваются с помощью команды SET (3.26) либо наследуются из родительского окружения, т.е. из окружения той программы, которая запустила на исполнение данную программу (6.04). Для осуществления подстановки знаки процента должны стоять с обеих сторон имени переменной (%VAR%, например).

Большое количество примеров строк с подстановками значений переменных окружения показано в разделе 9.03.

Примечание 1: чтобы знак процента не был бы интерпретирован как символ замещения, в строках batch-файла его следует указывать сдвоенным (%%). В результате интерпретации сдвоенный знак преобразуется в одиночный, который передается как параметр (т.е. без замещения) указанной в строке команде.

Примечание 2: команда FOR (3.13) использует свою собственную локальную переменную; ее имя, выражаемое одной буквой, вне batch-файлов должно указываться только с одним (предшествующим) знаком процента. Но чтобы передать этот знак процента в команду FOR из строки batch-файла, там имени этой переменной должны предшествовать два знака процента (например, %%Z, 3.13).

Примечание 3: другие интерпретаторы – IO.SYS и DEBUG.EXE – игнорируют знак процента, и подстановку значений переменных и формальных параметров не осуществляют.

2.03-04 Знак двойные кавычки (")

Когда интерпретатор встречает двойные кавычки в разбираемой строке, процесс разбора прерывается до того места той же строки, где будет найден еще один (замыкающий) знак двойные кавычки. Любая группа слов и знаков между открывающими и замыкающими двойными кавычками, возможно включающая знаки разделения и перенаправления, будет воспринята как единый элемент, причем и открывающие, и замыкающие двойные кавычки будут считаться принадлежащими этому элементу. Например, при исполнении строки

```
C:\>Batch.bat 1 " 2 3 " 4 ""
```

будет создан новый набор формальных параметров, в котором значением формального параметра %1 будет цифра 1, значением формального параметра %2 будет группа " 2 3 ", значением формального параметра %3 будет цифра 4, значением формального параметра %4 будет пустая пара двойных кавычек " ". Перечисленные значения формальных параметров будут сохраняться неизменными до завершения исполнения файла Batch.bat. Заключение группы слов в двойные кавычки позволяет сделать всю эту группу значением одного формального параметра. Этот прием нередко используется для неискаженного воспроизведения длинных имен файлов.

Пустая пара двойных кавычек (" ") рассматривается как специальный пустой символ, позволяющий не передавать никакого определенного значения и в то же время сохранить порядок присвоения значений формальным параметрам. Большая часть встроенных команд (кроме ECHO, IF и SET) игнорирует пустую пару двойных кавычек, но принимает то изменение порядка параметров, которое происходит в результате их введения. Например, следующая команда (3.03) перехода в другой каталог

```
C:\>cd ""
```

исполняется так, как будто никакого параметра ей не передано. Когда параметр заключен в двойные кавычки, команда исполняется так же, как если бы двойных кавычек вовсе не было:

```
C:\>cd "C:\dos"
```

Наличие замыкающей двойной кавычки при разборе строк обычно не проверяется, за исключением команд FIND и FOR. Разбор строк этих команд отличается тем, что сами двойные кавычки не считаются входящими в ту группу слов, которая в них заключена. Благодаря этому пустую пару двойных кавычек (" ") можно использовать в команде FIND (6.14) для подсчета полного числа строк в файлах. По той же причине команда FOR (3.13) позволяет избавиться от обрамляющих двойных кавычек, когда в них больше нет необходимости.

2.03-05 Квадратные скобки

Квадратным скобкам [ ] приписывается особое значение в файлах, которые должны быть интерпретированы отладчиком DEBUG.EXE или загрузчиком IO.SYS. Отладчик DEBUG.EXE интерпретирует данные в квадратных скобках как адреса тех ячеек памяти, где записаны операнды (подробнее – во вводной статье к главе 7).

В конфигурационных файлах MSDOS.SYS и CONFIG.SYS, которые должны быть интерпретированы загрузчиком IO.SYS, квадратные скобки имеют совсем

другое значение: заключенные в них слова отмечают начало отдельных конфигурационных блоков и в то же время являются именами этих конфигурационных блоков. Пример написания файла MSDOS.SYS приведен в разделе 5.01-01.

Для обозначения особых блоков в файле CONFIG.SYS существуют два зарезервированных слова: [menu] и [common]. Первое имя ([menu]) можно присваивать только блоку меню выбора вариантов конфигурации; если такой блок имеется, то он должен быть первым в файле CONFIG.SYS. Блок [menu] выделяется также особым составом команд: команды MENUCOLOR (4.19), MENUDEFAULT (4.20), MENUITEM (4.21) и SUBMENU (4.29) можно использовать только в блоках, объявленных как меню или субменю. Все другие конфигурационные команды, описанные в главе 4 (кроме NUMLOCK, 4.23), в блоке [menu] применять нельзя.

Команды, которые должны быть исполнены во всех конфигурациях, группируются в один или несколько блоков, названных одинаковым именем [common]. Вне заголовков имена конфигурационных блоков используются без ограничивающих квадратных скобок в качестве ссылок на соответствующие блоки (4.14). Примеры вариантов написания файла CONFIG.SYS с блоками [menu], [common] и рядом других приведены в разделах 9.04-01 и 9.09-01.

2.04 Знаки в роли команд

2.04-01 Двоеточие

Интерпретация двоеточия (:) зависит от его места в командной строке. Когда оно стоит первым в строке batch-файла, то следующее за ним слово считается меткой, обозначающей адресную точку для перехода. Если в той же строке имеются еще другие слова, то они не будут приняты во внимание. Сдвоенное двоеточие (::) в начале строки batch-файла иногда ставится для того, чтобы предотвратить исполнение всех указанных в строке операций, включая операции перенаправления (2.04-02 – 2.04-05).

Когда двоеточие (:) стоит в командной строке на втором месте, и ему предшествует буква, а остальная часть строки либо пуста, либо представляет собой одну обратную косую черту, либо заключена с обеих сторон знаками обратной косой черты, то в таких комбинациях начальная буква интерпретируется как буква диска, а вся комбинация воспринимается как команда перехода на этот диск, т.е. замены принимаемого по умолчанию (текущего) диска на указанный в этой команде. Например, чтобы перейти на диск A:, нужно набрать команду:

```
A:
```

или

```
A:\
```

или

```
A:\WINDOWS\
```

и потом нажать клавишу ENTER. Смена текущего диска не изменяет каталога, который считался принимаемым по умолчанию (текущим) на этом диске. Если, в частности, до перехода на диске A: принятым по умолчанию каталогом был A:\DOS, то он и останется в этой роли после исполнения любой из показанных выше команд перехода. В подобных командах любой путь, который может быть заключен между знаками обратной косой черты, не принимается во внимание и не проверяется. Фактически в качестве команды смены диска годится любой полный адрес, если справа к нему приписать знак обратной косой черты.

2.04-02 Левая стрелка

Левая стрелка (<) – это команда установить перенаправление ввода данных, исполняемая командным интерпретатором COMMAND.COM в ходе подготовки к передаче управления той программе, имя которой указано слева от левой стрелки в той же строке. По умолчанию стандартный канал ввода данных (STDIN) связан с консолью (CON), т.е. принимает ввод с клавиатуры. Фактически левая стрелка воспринимается как команда соединить стандартный канал ввода данных (STDIN) с тем источником, который указан справа от знака перенаправления ввода. Когда программа, указываемая слева от знака левой стрелки, запросит ввод данных через канал STDIN, она получит данные именно из этого источника. Естественно, воспользоваться перенаправлением ввода могут только те программы, которые запрашивают ввод через канал STDIN. Например, строка

```
MORE < C:\DOS\Filename.txt
```

означает, что программа MORE.COM (6.19) получит через канал STDIN данные, считываемые из указанного файла в каталоге C:\DOS\. Если путь к файлу не указан, то поиск его будет производиться только в текущем каталоге. Поиск по путям, указанным в переменной PATH, при перенаправлениях не производится, использование маски файла вместо имени не допускается, знаки подстановки (2.01-03) не раскрываются.

Помимо файлов, в качестве источника данных можно указывать порт (LPT1, LPT2, COM1, COM2, COM3, COM4). Когда устанавливаемые по умолчанию связи прерваны командой CTTY NUL, тогда для осуществления ввода с клавиатуры требуется явное перенаправление ввода с указанием консоли (CON) в качестве источника (пример – в разделе 3.07).

При любом пользовании перенаправлением ввода необходимо иметь основания для уверенности в том, что источник перенаправляемых данных сможет их предоставить. Ожидание ввода данных из пустого файла или из неработоспособного порта иногда кончается “зависанием” компьютера.

Примечание 1: перенаправления осуществляют путем подмены данных в таблице JFT (примечание 3 к A.07-1). Перенаправления, установленные командным интерпретатором для конкретной исполняемой программы, могут быть изменены или отменены самой исполняемой программой (пример – в разделе 9.07-02).

Примечание 2: интерпретаторы DEBUG.EXE и IO.SYS не выполняют перенаправлений и игнорируют все знаки перенаправления (2.04-02 – 2.04-05), в том числе левую стрелку. Тем не менее DEBUG.EXE способен воспринимать команды через перенаправление ввода, которое установит для него интерпретатор COMMAND.COM (примеры – в разделе 9.02).

2.04-03 Правая стрелка

Правая стрелка (>) – это команда установить перенаправление вывода данных, исполняемая командным интерпретатором COMMAND.COM в ходе подготовки к передаче управления той программе, имя которой указано слева от правой стрелки в той же строке. По умолчанию стандартный канал вывода данных (STDOUT) связан с консолью (CON), т.е. осуществляет вывод на экран дисплея. Перенаправление вывода позволяет связать канал STDOUT с другой целью назначения – той, которая указана справа от правой стрелки в командной строке. Например, команда DEL /? (3.09) выводит на экран справочные данные, но когда вслед за ней стоит правая стрелка, эти справочные данные будут выведены не на экран, а в указанный файл:

```
DEL /? > Filename.txt
```

Командный интерпретатор автоматически создаст файл с указанным именем, чтобы записать туда выводимые данные. Если файл с таким именем уже существовал, то он будет перезаписан без предупреждения, и его прежнее содержание будет потеряно.

Помимо файлов, допустимыми целями перенаправления вывода являются порты LPT1, LPT2, COM1, COM2, COM3, COM4, принтер PRN, который фактически эквивалентен порту LPT1, а также виртуальное устройство NUL, которое действует как "черная дыра": любые сообщения теряются там навсегда (примеры – в разделе 3.21). Перенаправление вывода в устройство NUL часто

используется именно для того, чтобы избежать воспроизведения на экране нежелательных сообщений.

Когда устанавливаемая по умолчанию связь канала STDOUT прервана командой CTTY NUL (3.07), тогда для осуществления вывода сообщений на экран необходимо явное перенаправление вывода с указанием консоли (CON) в качестве цели назначения (примеры – в разделе 9.03-02).

С помощью знака правой стрелки нельзя перенаправить те данные, которые посланы не по стандартному каналу вывода STDOUT, а по каким-либо другим каналам. В частности, сообщения, посылаемые через прерывание INT 29 (8.02-88), через прерывания BIOS (8.01-17, 8.01-21, 8.01-33) и через канал вывода сообщений об ошибках (STDERR, номерная ссылка 0002h), не перехватываются средствами перенаправления вывода.

Если перенаправления ввода и вывода приходится комбинировать в одной строке, то вслед за основной исполняемой командой надо сначала указать левую стрелку перенаправления ввода с последующей спецификацией источника, и лишь затем правую стрелку перенаправления вывода и новую цель назначения. Примеры комбинированных перенаправлений показаны в разделах 6.14, 6.25-03 и 9.03-02.

Всем знакам перенаправления (2.04-02 – 2.04-05) приписывается более высокий приоритет по сравнению с обычными операциями, за исключением знаков меток (2.04-01) и двойных кавычек (2.03-04). В частности, в командах ввода строк (ECHO, SET) все знаки перенаправления не включаются в состав вводимых строк, а инициируют исполнение перенаправления. По той же причине на исполнение перенаправлений не оказывают влияния условия, определяемые командой IF (3.15). Единственная возможность условного исполнения перенаправления состоит в том, чтобы обойти строку с перенаправлением посредством команды условного перехода "IF ... GOTO..." (3.15, 3.14).

Перенаправление осуществляется даже тогда, когда основная исполняемая команда никаких сообщений не выдает, или вовсе не исполняется из-за ошибок. Например, команда REM не выдает никаких сообщений, но тем не менее перенаправление ее пустого вывода – весьма распространенный способ создания новых файлов нулевой длины (3.24).

Чтобы перенаправить сообщения сразу от всех команд, указанных в строках batch-файла, нужно для исполнения этого файла запустить отдельный резидентный модуль командного интерпретатора COMMAND.COM с параметром /C (примеры – в 3.22 и 9.01-03). Без запуска отдельного резидентного модуля командного интерпретатора сообщения, выводимые при исполнении строк batch-файлов, можно перенаправлять только индивидуально, от каждой строки в отдельности.

Перенаправлением вывода следует пользоваться осторожно, потому что вместе с ожидаемыми сообщениями могут быть перенаправлены и не выведены на экран

предложения выполнить определенные действия и предупреждения, направляемые пользователю. Например, команда DIR /P останавливает вывод списка файлов после каждого очередного заполнения площади экрана и выдает сообщение, что вывод будет продолжен после нажатия любой клавиши. Но когда вывод сообщений перенаправлен, экран остается пустым, и кажется, будто компьютер завис.

2.04-04 Сдвоенная правая стрелка

Сдвоенная правая стрелка (>>), как и обычная правая стрелка (2.04-03), является командой перенаправления вывода, но действует по-другому в тех случаях, когда выводимое сообщение перенаправляется в уже существующий файл. При использовании знака сдвоенной правой стрелки содержимое этого файла не теряется; перенаправленное сообщение добавляется сзади к прежнему содержимому этого файла. Все прочие особенности исполнения перенаправления остаются такими же, как и при использовании знака одиночной правой стрелки (2.04-03).

2.04-05 Вертикальный штрих

Вертикальный штрих (|) является командой промежуточного перенаправления, т.е. осуществления передачи данных от одной программы к другой. Специально для этого командный интерпретатор COMMAND.COM заранее создает временный файл. Первой исполняется программа, имя которой указано слева от вертикального штриха. Сообщения, которые она направляет в канал STDOUT, перенаправляются на запись в тот самый временный файл. После окончания миссии первой программы на исполнение запускается вторая, имя которой указано справа от знака вертикального штриха. Когда эта вторая программа выдаст запрос на получение данных из стандартного канала ввода STDIN, эти данные будут считаны из того самого временного файла. По завершении исполнения обеих обслуживаемых программ временный файл автоматически уничтожается.

Например, следующая последовательность команд позволяет избежать прерывания исполнения при удалении всех файлов из указанного каталога:

```
ECHO Y | DEL C:\TEMP\*.*
```

Первым делом сообщение, выводимое командой ECHO (3.11), записывается во временный файл. В данном случае это сообщение состоит из одной буквы Y . Затем запускается на исполнение команда DEL (3.09). Обнаружив маску *.*, она выдает запрос к пользователю: действительно ли он хочет удалить все файлы из каталога C:\TEMP. Но так как в данном случае канал ввода тоже перенаправлен,

пользователю не придется отвлекаться: ответ – буква Y – будет сразу считан из подготовленного временного файла.

В одной строке знаками промежуточного перенаправления могут быть связаны более чем две команды. Примеры командных строк с неоднократным промежуточным перенаправлением приведены в разделах 3.08 и 3.28.

Если команда, указываемая справа от знака промежуточного перенаправления, не запрашивает сведения из созданного временного файла, то команда, указываемая слева от знака промежуточного перенаправления, не обязана выдавать сообщение в канал STDOUT. Поэтому вертикальный штрих потенциально может служить разделителем, позволяющим записывать несколько команд в одной строке. Тем не менее его использование в этой роли нельзя рекомендовать, потому что команда FOR (3.13) позволяет сделать то же самое гораздо быстрее и без обращений к записываемому диску для создания временных файлов.

Примечание 1: промежуточное перенаправление предполагает создание временного файла либо в каталоге, указанном в переменной окружения TEMP, либо в текущем каталоге. Обе эти попытки найти место для записи временного файла, однако, могут кончиться неудачно, если DOS загружен с оптического диска CD-ROM или с диска, защищенного от записи. В таких случаях выдается сообщение о том, что промежуточное перенаправление не может быть исполнено, и тогда команда, указанная справа от знака вертикального штриха, тоже не исполняется.

2.04-06 Знак "эт" (@)

Когда знак "эт" указан первым в строке batch-файла, он интерпретируется как команда запрещения отображения этой строки на экране. Поэтому первая строка почти каждого batch-файла начинается со знака "эт", за которым следует команда ECHO OFF. Иногда такое действие знака "эт" (@) оказывается полезным не только в первой строке batch-файла (примеры – в разделах 3.13, 6.25-02, 6.25-03).

Примечание 1: DOS не накладывает ограничений на использование знака "эт" в именах и суффиксах файлов, но неправильная интерпретация имен файлов, начинающихся с этого знака, может привести к серьезным ошибкам.

Глава 3. Команды интерпретатора COMMAND.COM

Встроенные команды – это те, которые исполняются интерпретатором, обслуживающим командную строку. В отличие от других программ, которые перед исполнением приходится искать и загружать с диска, резидентный командный интерпретатор постоянно находится в памяти, вследствие чего исполнение его собственных встроенных команд происходит гораздо быстрее. Раз встроенные команды не приходится искать, то и спецификации пути перед их именами быть не должно. Еще одно общее свойство всех встроенных команд состоит в том, что по завершении исполнения они не оставляют код ошибки (errorlevel).

Данная глава представляет набор встроенных команд основного в MS-DOS7 командного интерпретатора COMMAND.COM (размер файла 93812 байт, дата файла 12.06.1996). Национально-адаптированные версии этого интерпретатора обычно имеют ненамного больший размер файла и более позднюю дату, но тем не менее исполняют те же самые команды. Среди представленных здесь команд есть несколько таких, которые предназначены для ввода не с клавиатуры, а только из строк batch-файлов (3.02, 3.14, 3.21, 3.27).

Для большинства встроенных команд в составе COMMAND.COM имеется краткая справка; чтобы ее вывести на экран, надо набрать имя команды, пробел,

косую черту и знак вопроса /? , а потом нажать клавишу Enter. Практика показывает, однако, что одной краткой справки бывает недостаточно. В разделах данной главы содержится большое количество дополнений и пояснений, которые позволят Вам использовать встроенные команды гораздо более эффективно и избежать многих распространенных ошибок.

3.01 BREAK – управление перехватом доступа к дискам

Команда BREAK (= прервать) – эквивалент одноименной конфигурационной команды (подробнее – в разделе 4.02). Они обе позволяют воздействовать на один и тот же двоичный флаг, от которого зависит проведение проверок нажатий клавиш BREAK и CTRL-C при выполнении дисковых операций. Состояние этого двоичного флага не пропадает вместе с переменными локального окружения, когда вторичный резидентный модуль командного интерпретатора кончит свою работу. В отличие от загрузчика IO.SYS (4.02), интерпретатор COMMAND.COM отвечает на ввод команды BREAK без параметров индикацией текущего состояния упомянутого двоичного флага.

3.02 CALL – вызов batch-файла

CALL (= вызвать) – команда для вызова на исполнение одного (вторичного) batch-файла из строки другого (первичного) batch-файла. Batch-файлы – это неформатированные текстовые командные файлы с суффиксом *.BAT, из которых интерпретатор Command.com принимает расширенный набор команд. Команда CALL – это как раз одна из тех команд, которые предназначены для ввода не с клавиатуры, а только из строк batch-файлов.

Когда в строке batch-файла командный интерпретатор встречает имя исполняемой программы (утилиты), он передает управление этой программе, а по завершении ее работы снова берет управление на себя и переходит к интерпретации следующей строки batch-файла. Но когда в строке batch-файла встречается вызов другого (вторичного) batch-файла, командный интерпретатор переходит к интерпретации строк вторичного batch-файла и продолжает это делать до его конца, после чего процесс завершается, не возвращаясь к прерванной интерпретации строк первичного batch-файла. Чтобы прерванное исполнение первичного batch-файла было бы доведено до конца, вторичный batch-файл должен быть запущен на исполнение с помощью команды CALL, например:

```
CALL C:\DOS\VC4\HELP.BAT J 96
```

здесь:

C:\DOS\VC4\ – пример пути к файлу HELP.BAT; если путь не указан, DOS будет искать этот файл в текущем каталоге, а потом по всем путям, указанным в переменной окружения PATH.
HELP.BAT – пример имени вторичного batch-файла.
J 96 – группа параметров, которая должна быть передана файлу HELP.BAT (другим batch-файлам будут нужны другие параметры или могут быть не нужны вовсе).

Фактически команда CALL предотвращает закрытие первичного batch-файла, сохраняет выделенную ему память вместе с его формальными параметрами и указателем точки возврата, позволяет командному интерпретатору выполнить вторичный batch-файл, а потом восстанавливает доступ к сегменту первичного batch-файла и обеспечивает продолжение его исполнения со следующей операции.

Примечание 1: вторичный batch-файл наследует не копии переменных окружения, а доступ к той же области переменных окружения. Значения переменных, присвоенные во вторичном batch-файле, становятся доступны из первичного batch-файла после возобновления его исполнения.
Примечание 2: с помощью команд CALL можно организовать многократно вложенные вызовы batch-файлов.
Примечание 3: команда CALL позволяет осуществлять рекурсивные вызовы, т.е. batch-файл может быть вызван из содержащейся в нем же строки. При этом пользователь сам должен позаботиться об условиях, предотвращающих неограниченное нарастание глубины вложенности.
Примечание 4: если вторичный batch-файл не найден ни в текущем каталоге, и ни по одному из путей, объявленных в переменной PATH, то происходит переход к интерпретации следующей строки первичного batch-файла без какого-либо сообщения об ошибке.
Примечание 5: в строках с командой CALL не допускается наличие знаков перенаправления (2.04-02 – 2.04-05).
Примечание 6: не следует путать описанную здесь команду CALL, исполняемую интерпретатором Command.com, с одноименной ассемблерной командой (7.03-08), исполняемой отладчиком Debug.exe.

3.03 CD – смена текущего каталога

Для упрощения работы с командной строкой DOS способна принимать команды без спецификации конкретного диска или конкретного пути. Неопределенность разрешается посредством установок по умолчанию, для чего должен быть заранее задан принимаемый по умолчанию (текущий) диск, как показано в разделе 2.04-01, а также принимаемый по умолчанию (текущий) каталог

на каждом диске. Назначение конкретного каталога на роль принимаемого по умолчанию (текущего) выполняется с помощью команды CD.

Команда CD (Change Directory = сменить каталог) заменяет текущий каталог на любом логическом диске (но не сам текущий диск!) в соответствии с указанным путем. Например, команда

```
CD C:\DOS
```

сменит текущий каталог на \DOS, если текущим диском в данный момент является диск C:. Если же команда CD обращается не к тому диску, который в данный момент является текущим, то тогда указанный путь будет принят во внимание как предварительная установка, которая определит текущий каталог только после того, как текущим диском станет диск, указанный в команде CD.

Путь в команде CD может быть указан в любой из допустимых форм (2.02-01). Последним словом пути обязательно должны быть либо имя целевого каталога, либо комбинация замещающих его знаков обратной косой черты, точки и сдвоенной точки (2.02-03). Комбинации этих знаков позволяют осуществить переход в корневой каталог (`CD \`), в вышестоящий каталог (`CD ..`), подняться на два уровня вверх по структуре каталогов (`CD ..\..`) и т.п.

Если вместо пути в команде CD указать только диск, например

```
CD C:
```

то будет выведен на экран путь к каталогу, который предустановлен на роль текущего на указанном диске. Если не указывать и диск, то будет выведен путь к текущему каталогу на текущем диске.

Примечание 1: CHDIR – еще одно допустимое имя той же команды CD.

Примечание 2: предустановленный командой CD каталог невозможно удалить командой RD (3.23), причем не только на текущем диске, но и тогда, когда указанный диск не является в данный момент текущим.

Примечание 3: предустановленные пути к текущим каталогам для всех логических дисков DOS хранит в таблице CDS (A.03-3), причем при формировании этой таблицы туда автоматически записываются указатели на корневые каталоги всех дисков.

3.04 CHCP – смена кодовой страницы

Кодовая страница – это набор знаков для вывода сообщений на экран. Когда в командной строке вслед за командной CHCP (CHange CodePage = сменить кодовую страницу) не указан номер кодовой страницы, эта команда выводит сообщение о том, какая кодовая страница задействована в данный момент.

Для осуществления смены кодовых страниц командой CHCP необходимо заранее выполнить следующие условия:

– драйвер DISPLAY.SYS (5.02-02) должен подготовить буферные области в памяти для размещения не менее чем двух кодовых страниц;
– кодовые страницы должны быть загружены в подготовленные буферные области командой MODE.COM CON CP PREP (6.18-03);
– должен быть загружен резидентный модуль драйвера NLSFUNC.EXE (5.02-03), выполняющий смену кодовых страниц.

Обычно загружают только одну кодовую страницу, потому что каждая национальная кодовая страница, помимо набора национальных знаков, содержит также знаки американского алфавита. Переключение между этими двумя наборами знаков не требует переключения кодовых страниц: оно выполняется в пределах одной кодовой страницы.

Загружать несколько кодовых страниц приходится тогда, когда Вы намерены использовать не менее чем два набора национальных (не-американских!) знаков. Номера национальных кодовых страниц приведены в таблице A.02-2. Если, например, Вы подготовили норвежскую кодовую страницу 865 и русскую кодовую страницу 866, тогда командами CHCP 865 и CHCP 866 можно будет выполнять переключение этих кодовых страниц.

Примечание 1: смену кодовых страниц также можно осуществлять командой MODE.COM CON CP SEL (6.18-03), для которой не требуется загружать резидентный модуль драйвера NLSFUNC.EXE.

Примечание 2: команда CHCP оперирует с кодовыми страницами, поставляемыми фирмой Microsoft и загружаемыми с помощью драйверов фирмы Microsoft. Другие кодовые страницы, в том числе используемые драйвером KEYRUS.COM (5.02-05), не переключаются командой CHCP.

Примечание 3: если Вы пользуетесь национальной версией какой-либо нужной для Вас программы (например, файл-менеджера Norton Commander), то следует иметь ввиду, что смена кодовой страницы сделает все не-американские надписи полностью нечитаемыми. Смена кодовой страницы не повлияет только на американские знаки (с номерами 32 – 127), общие для всех кодовых страниц.

3.05 CLS – очистка экрана

Команда CLS (CLear Screen = очистить экран) не только стирает все содержимое текущей страницы в буфере видеопамяти, но также восстанавливает принимаемые по умолчанию параметры индикации: сбрасывает установки цвета и обеспечивает воспроизведение белых знаков на черном фоне.

3.06 COPY – копирование файла

Команда COPY (= копировать) позволяет выполнять копирование одного или нескольких файлов, переименование копии, а также конкатенацию (слияние) нескольких файлов. Вот пример использования команды COPY для копирования одного файла в другой каталог:

```
COPY /A C:\DOS\MS7\TRIAL.TXT A:\DOS /V /Y
```

здесь:

/A – необязательный параметр, вызывающий прерывание копирования на первой встреченной метке конца файла (1Ah). По умолчанию при копировании одного файла принимается альтернативный параметр /B, обеспечивающий копирование файла целиком, поскольку байт 1Ah в исполняемых и других бинарных файлах может играть совсем другую роль.

C:\DOS\MS7\ – пример пути к копируемому файлу. Другие допустимые формы спецификаций пути показаны в разделах 2.02-01 – 2.02-03. Если путь не указан, поиск файла будет производиться только в текущем каталоге. Пути, указываемые в переменной PATH, команда COPY игнорирует.

TRIAL.TXT – пример имени файла, подлежащего копированию. Имя нужно указывать полностью, с суффиксом, если он имеется. Допускается копировать файлы без атрибутов, а также с атрибутами "A" (подлежащий архивированию) и "R" (только для чтения). Файлы с атрибутами "H" (скрытый) и "S" (системный) командой COPY не копируются.

A:\DOS – пример пути, который будет интерпретирован как путь к каталогу назначения, где должна быть размещена копия, если такой каталог существует. Если же такого каталога нет, то последнее имя (DOS) будет интерпретировано как новое назначаемое копии имя, и копия с таким именем будет помещена в корневой каталог диска A:. Если Вы не намерены присваивать копии другое имя, то в качестве каталога назначения нельзя указывать тот, где размещен копируемый файл. Когда файл находится не в текущем каталоге, путь к каталогу назначения можно не указывать: по умолчанию копирование будет производиться в текущий каталог.

/V – необязательный параметр, вызывающий проверку соответствия копии исходному файлу. Проверка замедляет процесс копирования и фактически не нужна, если копирование производится на жесткий магнитный диск. Но если

производится копирование на дискету, то проверка может оказаться не лишней.

/Y – необязательный параметр, позволяющий выполнять без предупреждения перезапись одноименного файла, если такой файл будет обнаружен в каталоге назначения. Параметр /Y может быть предустановлен в переменной окружения COPYCMD (командой SET COPYCMD= /Y), и тогда указывать его в командной строке уже не нужно. Такая предустановка может быть преодолена указанием в командной строке параметра /-Y , если предупреждение о перезаписи необходимо.

Когда параметр /A указан последним в командной строке, его действие оказывается совсем другим: он не препятствует копированию файла целиком, но вызывает добавление метки конца файла (байта 1Ah) к копии, если этой метки не было в конце копируемого файла.

Первый путь, указываемый в команде COPY, всегда интерпретируется как путь к копируемому файлу, а последний путь – как путь к каталогу назначения, куда должна быть помещена копия. Указание более чем двух путей считается ошибкой, за исключением копирования с конкатенацией, когда посредством последовательного сцепления копий нескольких исходных файлов должна быть получена объединенная копия. При копировании с конкатенацией спецификациям остальных копируемых файлов (кроме первого) должен предшествовать знак плюс (+), например:

```
COPY /B T1.DAT + T2.DAT /A + REMARK.TXT C:\DOS
```

здесь:

/B – необязательный параметр, предшествующий первому копируемому файлу, распространяет свое действие на последующие копируемые файлы до тех пор, пока не будет встречен альтернативный параметр (в данном случае параметр /A). Последний распространит свое действие на остальные копируемые файлы. Поскольку конкатенация применяется преимущественно к текстовым файлам, постольку при копировании с конкатенацией по умолчанию принимается параметр /A, т.е. копирование каждого файла осуществляется до первой встреченной в нем метки конца файла.

T1.DAT, T2.DAT, REMARK.TXT – это примеры спецификации трех исходных файлов, копии которых подлежат конкатенации. Поскольку пути к исходным файлам не указаны, предполагается их наличие в текущем каталоге. Спецификации каждого исходного файла, кроме первого, в командной строке предшествует плюс – знак конкатенации.

C:\DOS – пример пути, который будет интерпретирован как путь к каталогу назначения, если такой каталог существует, и тогда в этот каталог будет помещен файл, полученный в результате конкатенации, который будет наследовать имя первого из копируемых файлов (T1.DAT). Если же каталог C:\DOS не существует, то последнее имя (DOS) будет интерпретировано как новое имя, назначаемое файлу, полученному в результате конкатенации, и файл с таким именем будет записан в корневой каталог диска C:.

Использование масок файлов (2.01-03) вместо имен копируемых файлов в команде COPY допускается, но требует осторожности. Для пояснения рассмотрим следующий пример:

```
COPY /B T*.DAT  C:\DOS\CONCAT.DAT
```

Команда COPY проверяет последнее слово в спецификации пути назначения (справа от последнего знака обратной косой черты) на предмет того, является ли это слово именем существующего каталога или нет. Если каталог CONCAT.DAT не существует, то копии всех файлов, соответствующих указанной маске, будут объединены (конкатенированы) в общий файл, который будет создан в каталоге C:\DOS и получит имя CONCAT.DAT. Если же каталог CONCAT.DAT существует, то копии всех файлов, соответствующих указанной маске, не будут объединены, а будут поодиночке записаны в этот каталог, сохраняя имена исходных файлов. Последний пример показывает, что одна и та же строка с командой COPY дает возможность получать совершенно разные результаты. Во избежание этого необходимо точно знать, что назначаемое имя копии не совпадает с именем существующего каталога.

При конкатенации копий нескольких файлов назначаемое имя объединенной копии может совпадать с именем первого копируемого файла. Имя любого из прочих копируемых файлов присваивать объединенной копии нельзя, иначе содержимое этого исходного файла будет утрачено до копирования. Если Вы уверены, что назначаемое имя копии не будет интерпретировано неверно и не вызовет конфликта имен, то тогда можно указывать один и тот же путь к исходным файлам и к каталогу назначения, или вовсе не указывать эти пути: по умолчанию копирование будет выполняться в текущий каталог.

Копирование файла "самого в себя" считается ошибкой, но форма копирования с конкатенацией все-таки позволяет указать один и тот же путь как к копируемому файлу, так и к каталогу назначения. При этом спецификации второго (добавляемого) файла вообще можно опустить, например:

```
COPY /B \DOS\FILE.EXT +,,\DOS
```

Такую форму псевдокопирования применяют, когда надо обновить дату файла, а также когда надо удалить файл, если он пуст (примечания 1 и 2).

Использование в команде COPY зарезервированного слова CON (консоль) вместо спецификации копируемого файла переключает командный интерпретатор в режим ввода текста:

```
COPY CON C:\DOS\REMARKS.TXT
```

Такая команда позволяет набирать текст с клавиатуры в указанный файл до тех пор, пока не будет нажата клавишная комбинация F6-ENTER для возврата обратно в режим командной строки (подробнее в разделе 1.04).

Вместо каталога назначения в команде COPY можно указывать зарезервированные слова PRN (принтер), LPT1 – LPT4 (параллельный порт), COM1 – COM4 (последовательный порт) и NUL (виртуальный порт вывода "в никуда"). В частности, команда

```
COPY CON PRN
```

превращает компьютер в пишущую машинку. Конечно, указываемое терминальное устройство должно быть подключено, правильно сконфигурировано и способно адекватно взаимодействовать с DOS. При копировании файлов на внешние устройства по умолчанию принимается копирование с параметром /A, то есть до первой метки конца файла. В отличие от реальных устройств вывода виртуальное устройство NUL всегда сконфигурировано правильно. Копирование в устройство NUL иногда используется для выяснения того, не пуст ли тестируемый файл и можно ли его прочитать.

Примечание 1: пустые файлы (т.е. имеющие нулевую длину) не копируются.
Примечание 2: если копируемый файл пуст, и в качестве имени копии указывается имя существующего пустого файла, то последний уничтожается.
Примечание 3: результат копирования нельзя перенаправить, перенаправляются только выводимые на экран сообщения.
Примечание 4: атрибуты копируемого файла не копируются.
Примечание 5: копиям файлов всегда присваивается атрибут "A" (подлежит архивированию).

3.07 CTTY – смена канала ввода-вывода

Команда CTTY позволяет изменить принимаемые по умолчанию установки сразу для всех трех основных каналов ввода и вывода: STDIN, STDOUT и STDERR. Начальные установки, действующие с момента включения компьютера, эквивалентны команде CTTY CON и связывают все три упомянутых канала с консолью, то есть с клавиатурой в качестве устройства ввода и с дисплеем в качестве устройства вывода. Вместо консоли (CON) в команде CTTY может быть

указан один из портов COM1 (AUX), COM2, COM3, COM4, LPT1 (PRN), LPT2 или виртуальное устройство NUL (для вывода "в никуда").

CTTY – это архаичная команда, расшифровка ее названия Change TeleTYpewriter (= сменить телетайп) напоминает о временах, когда еще не было дисплеев, а ввод-вывод происходил через порты с подключенными печатающими устройствами типа телетайпов.

В наше время есть два повода использовать команду CTTY в batch-файлах. Первый – для предотвращения нежелательного прерывания исполнения. Второй - для предотвращения вывода на экран сообщений из канала STDERR, которые нельзя перенаправить иначе. В обоих случаях проблему решает установка с направлением "в никуда", представляемым виртуальным устройством NUL. Команда CTTY NUL должна предшествовать защищаемой группе строк, а после команда CTTY CON должна восстановить нормальное взаимодействие с клавиатурой и дисплеем. В промежутке между CTTY NUL и CTTY CON недопустимы команды, которые могут вызвать неожиданный останов исполнения, потому что сообщение об этом не будет воспроизведено на экране, ввод с клавиатуры не будет воспринят, и компьютер фактически зависнет. Только перезагрузка через CTRL-ALT-DELETE обычно остается возможной.

Действие команды CTTY распространяется на принимаемые по умолчанию установки, но не распространяется на перенаправления, указываемые в строках batch-файла явно. Рассмотрим, например, следующий фрагмент batch-файла:

```
@ctty nul
copy /B trial.dat suit.dat
echo Press any key to exit > con
pause < con
ctty con
```

Здесь сообщение от команды COPY не будет показано на экране, даже если это будет сообщение об ошибке, а вот сообщение "Press any key to exit" будет воспроизведено, потому что оно явно перенаправлено на консоль (CON), и команда PAUSE тоже сработает правильно, потому что для нее явно указано перенаправление ввода с клавиатуры. Такое использование команды CTTY требует осторожности, но открывает возможности более радикально влиять на отображаемые сообщения. Пример batch-файла с подобным использованием команды CTTY приведен в разделе 9.03-02.

Примечание 1: сообщения, посылаемые через канал STDERR, не могут быть перенаправлены. Когда командой CTTY в качестве установки по умолчанию назначено виртуальное устройство NUL, эти сообщения безвозвратно теряются.

3.08 DATE – установка даты.

Для установки даты вслед за именем команды DATE (= дата) в командной строке должна быть указана дата, которую надлежит установить, например

```
DATE 11.07.2002
```

Когда дата в командной строке не указана, на экран выводятся текущая дата и приглашение ввести новую дату (обратите внимание на примечание 2). Если изменять дату Вы не намерены, достаточно ответить на приглашение нажатием клавиши ENTER.

Чтобы дополнить текстовый файл строкой с указанием текущей даты, команду DATE можно использовать, например, следующим образом:

```
ECHO= | DATE | FIND.EXE "Current" >> ANYFILE.TXT
```

В этой строке первое промежуточное перенаправление (ECHO= | DATE) автоматически отвечает на выводимое приглашение, второе промежуточное перенаправление (DATE | FIND.EXE) устраняет лишние строки выводимого сообщения, и третье перенаправление (>> ANYFILE.TXT) приписывает строку с текущей датой в конец указанного файла. При этом, конечно, должны быть удовлетворены все условия исполнения перенаправлений (2.04-05) и успешного обнаружения упоминаемых файлов (FIND.EXE и того, который надлежит дополнить).

Примечание 1: порядок указания даты, месяца и года неодинаков для разных стран и устанавливается командой COUNTRY (4.05).

Примечание 2: приглашение к вводу новой даты сопровождается подсказкой с обозначением года двухзначным числом. Это ошибка: в MS-DOS7 год обозначается четырехзначным числом.

3.09 DEL – удаление файла

Команда DEL (DELete = удалить) не стирает с диска удаляемый файл, а лишь делает недействительной запись о регистрации этого файла в каталоге. Занятые файлом кластеры становятся свободными и могут быть использованы для записи других файлов; но пока они не перезаписаны, удаленный файл можно восстановить (например, с помощью программы UNDELETE.EXE).

Вот пример командной строки для удаления одного файла командой DEL:

```
DEL D:\TEMP\FILENAME.EXT /P
```

здесь:

D:\TEMP\ – пример пути к файлу, который надлежит удалить; если путь не указан, то поиск будет производиться только в текущем каталоге.

FILENAME.EXT – пример имени удаляемого файла.
/P – необязательный параметр, вызывающий запрос разрешения на удаление каждого файла.

Если вместо имени удаляемого файла указана маска файла (2.01-03), то будут удалены все файлы, соответствующие данной маске. Но при попытке удалить все файлы в текущем каталоге посредством маски *.* исполнение будет остановлено запросом к пользователю, причем независимо от того, указан ли параметр /P в командной строке. На запрос надо ответить нажатием клавиши Y (да) или N (нет).

При написании batch-файлов бывает важно обеспечить безостановочное исполнение, не отвлекающее пользователя запросами. В частности, безостановочное удаление файлов по маске *.* обеспечивается, когда команда DEL исполняется в составе цикла FOR (3.13):

```
FOR %%Z in (*.*) do DEL %%Z
```

Исполнение такого цикла сопровождается индикацией списка удаляемых файлов. Чтобы избежать не только запросов, но и появления на экране каких-либо сообщений, можно командную строку скомпоновать, например, так:

```
ECHO Y | IF EXIST D:\TEMP\*.* DEL D:\TEMP\*.* > NUL
```

В этом примере ответ на запрос обеспечивает команда "ECHO Y", а условие "IF EXIST" введено только для того, чтобы избежать сообщения об ошибке "File not found" ("Файл не найден"), когда указанный каталог изначально оказывается пуст.

Примечание 1: ERASE – еще одно имя той же самой команды DEL.
Примечание 2: командой DEL нельзя удалить файлы с атрибутами R (только для чтения), H (скрытый), S (системный), а также каталоги. Удаление каталогов выполняет команда RD (3.23).
Примечание 3: команда DEL . (с точкой) тождественна команде DEL *.*.
Примечание 4: команда DEL \ удаляет все не защищенные атрибутами файлы в корневом каталоге текущего диска.

3.10 DIR – вывод содержания каталога

В MS-DOS команда DIR (DIRectory = каталог) – главный инструмент исследования содержимого каталогов. Вот пример командной строки с командой DIR:

```
DIR C:\DOS /P /A:HS /O:GN /S /L /V
```

здесь:

C:\DOS – пример пути к заданному каталогу. Если путь не указан, то по умолчанию предполагается текущий каталог.

/P – необязательный параметр, который при заполнении экрана останавливает вывод строк до тех пор, пока пользователь не разрешит продолжить вывод нажатием любой клавиши.

/A:HS – параметр, разрешающий показывать объекты с указанными атрибутами: H (скрытый), S (системный), A (подлежащий архивированию), R (только для чтения), D (каталоги). Можно использовать префикс "–", чтобы реверсировать отбор: –H (кроме скрытых), –D (кроме каталогов), и т.д.. Если указан только параметр /A (без перечня атрибутов), то будет показано все, что содержится в каталоге. Если параметр /A вовсе опущен, то по умолчанию скрытые и системные файлы не показываются.

/O:GN – параметр, задающий порядок сортировки: G – сначала каталоги, N – по именам, S – по возрастанию размера, E – по суффиксу, D – по возрастанию срока давности, A – по возрастанию срока с момента последнего доступа. Можно использовать префикс "–" для реверсирования порядка: –N – по именам в обратном алфавитном порядке, –S – по мере уменьшения размера, и т.д.. Если параметр /O не указан, то по умолчанию происходит сортировка по именам.

/S – необязательный параметр, вызывающий выведение содержания не только указанного каталога, но и всех вложенных в него подкаталогов.

/L – необязательный параметр, вызывающий выведение имен файлов строчными буквами. Если параметр /L не указан, то имена выводятся так, как они были первоначально набраны.

/V – необязательный параметр, вызывающий выведение дополнительных сведений: атрибутов, времени последнего доступа, занятого дискового пространства, полного дискового пространства и степени его использования. Выведение дополнительных сведений не производится, если заданы другие форматы индикации следующими параметрами:

- /W – показывать имена в 5 колонок,
- /B – показывать имена в одну колонку вообще без сводки данных о диске; исключение затрат времени на подготовку этих данных делает команду DIR /B гораздо более быстродействующей.

Команду DIR также можно использовать для выведения сведений об определенном файле или о нескольких файлах, селектируемых по заданной маске, например:

```
DIR *.txt /P /S /B
```

Сочетание в последнем примере параметра /S с отсутствием спецификации пути фактически означает проведение поиска файлов, удовлетворяющих указанной маске, в текущем каталоге и вдоль всей структуры нижележащих подкаталогов. Если текущим каталогом является корневой каталог диска, то поиск будет производиться по всему диску. Дополнительное указание параметра /B в последнем примере целесообразно тогда, когда надо сделать выводимый список файлов более простым для восприятия.

Параметры команды DIR могут быть записаны в переменную окружения DIRCMD, (например, командой SET DIRCMD= /P /S /B), и тогда для осуществления нужного действия не потребуется набирать эти параметры в командной строке. При необходимости установки, определенные в переменной DIRCMD, можно преодолеть указанием в командной строке тех же параметров с предшествующим знаком дефис "–" (например, /–P).

Когда команда DIR выполняется с параметром /A, тогда маска *.* (все файлы) включает и каталоги тоже. Если нужно выделить только файлы, то следует указать параметр /A:–D. Эти особенности команды DIR позволяют использовать ее для выяснения, пуст ли данный каталог или нет. В качестве примера рассмотрим следующий фрагмент batch-файла:

```
@echo off
set DIRCMD=/a /b
dir *.* > C:\Temp\Found.lst
copy C:\Temp\Found.lst NUL | Find.exe "0 file" > nul
if errorlevel 1     echo Current directory is NOT empty
if not errorlevel 1 echo Current directory is empty
```

Вторая строка приведенного фрагмента задает параметры так, что команда DIR не выдаст никаких сообщений, если текущий каталог абсолютно пуст. В третьей строке сообщения команды DIR перенаправлены в файл FOUND.LST. Команда COPY в 4-й строке не станет копировать файл, если он пуст, и выдаст сообщение "0 files copied" ("Скопировано 0 файлов"). Перехватив такое сообщение через перенаправление, программа FIND.EXE (6.14) установит возвращаемый код ошибки (errorlevel) в нуль. Последние две строки просто воспринимают установленный код ошибки, чтобы показать результат теста на экране.

Когда для команды DIR задан обычный формат индикации (без параметров /W или /B), имена файлов выводятся по-разному в зависимости от того ,в какой операционной среде команда исполняется. В окне DOS операционной системы WINDOWS имена файлов отображаются как обычно, а в MS-DOS7 имена файлов и их суффиксы воспроизводятся отдельно без точки между ними. Эту особенность можно использовать в качестве теста для определения операционной среды.

Примечание 1: команда DIR \ показывает все файлы корневого каталога.

Примечание 2: прокрутку строк, выводимых на экран командой DIR, можно временно остановить нажатием клавиш BREAK или CTRL-S.

3.11 ECHO – посылка сообщения в канал вывода

Слова, указанные после имени команды ECHO (= отзвук) в той же командной строке, передаются в стандартный канал вывода STDOUT. Если этот канал не перенаправлен, то по умолчанию он связан с консолью (CON), то есть с дисплеем в качестве устройства вывода. Примеры вывода текстовых строк с помощью команды ECHO приведены, в частности, в предыдущем разделе 3.10.

Длина выводимого на экран сообщения – до 123 знаков – ограничивается длиной строки или первым встреченным знаком перенаправления (2.04-02 – 2.04-05). Сообщение может содержать служебные коды ASCII, показанные в приложении A.02-8. Но сообщение не должно быть пустым, не должно начинаться со слов ON или OFF, а также с некоторых символов, изменяющих миссию команды ECHO. Перечень таких исключений включает следующее:

Echo ON – устанавливает (включает) ECHO-флаг, разрешающий индикацию следующих командных строк при исполнении batch-файлов; это же состояние устанавливается по умолчанию.

Echo OFF – сбрасывает (выключает) ECHO-флаг, запрещая тем самым индикацию следующих командных строк при исполнении batch-файлов. Вне batch-файлов сброс ECHO-флага вызывает пропадание привычного приглашения командной строки.

Echo – вообще без последующих знаков – показывает состояние ECHO-флага в данный момент.

Echo= – с приписанным вплотную знаком равенства – посылает в канал STDOUT байты 0Dh 0Ah, действующие эквивалентно нажатию клавиши ENTER (пример – в разделе 3.08). В частности, на экране или в файле это вызывает появление пустой строки.

Echo+ – тоже посылает в канал STDOUT байты 0Dh 0Ah, но, кроме того, посылает и слова, если они указаны правее знака плюс, в том числе слова ON и OFF. Так же действует команда ECHO с приписанной вплотную точкой или косой чертой.

ECHO-флаг имеет статус локальной двоичной переменной, состояние которой сохраняется до тех пор, пока исполняемые batch-файлы работают в общем пространстве переменных окружения. Но состояние ECHO-флага не наследуется и устанавливается по умолчанию каждый раз, когда командный интерпретатор COMMAND.COM создает производное (дочернее) пространство переменных окружения.

Исполняемые командные строки, показываемые на экране при включенном состоянии ECHO-флага, не являются точной копией исходных строк из batch-файлов: в показываемых на экране строках имена вызываемых переменных окружения и формальных параметров уже заменены их значениями. Это бывает очень полезно для поиска возможных ошибок. Но отлаженные batch-файлы обычно исполняются при выключенном состоянии ECHO-флага, и потому почти всегда начинаются с команды @ECHO OFF. Знак @ перед командой ECHO предотвращает воспроизведение самой этой команды на экране.

3.12 EXIT – выход из интерпретатора COMMAND.COM

Командный интерпретатор COMMAND.COM (6.04) – это резидентная программа, которая помогает запускать на исполнение другие программы. С другой стороны, командный интерпретатор сам может быть запущен на исполнение как обычная программа, например, для создания отдельного (локального) пространства окружения. В таких ситуациях в памяти компьютера могут оказаться одновременно несколько резидентных модулей командного интерпретатора, причем работающим оказывается тот, который загружен последним. Команда EXIT (= выйти) вызывает завершение работы действующего резидентного модуля командного интерпретатора, освобождает занятую им память и передает управление той программе, которая запустила его на исполнение.

Вместе с прекращением деятельности резидентного модуля теряется созданное им пространство окружения со всеми локальными переменными, но зато вновь становятся доступными сохраненные значения переменных окружения вызвавшей его программы. Исполнение этой программы автоматически возобновляется.

Первичная загрузка резидентного модуля командного интерпретатора происходит при его запуске командой SHELL из файла CONFIG.SYS (пример в 9.01-01). У этого модуля вызвавшей его программы ("предка") нет. Если бы было можно задействовать команду EXIT по отношению к первичному резидентному модулю, то передавать управление было бы некому, и компьютер бы завис. Чтобы предотвратить такой исход, командный интерпретатор COMMAND.COM в первый раз должен быть запущен на исполнение с параметром /P (6.04), который блокирует исполнение команды EXIT.

3.13 FOR – оператор цикла

Оператор цикла FOR (= для) организует многократное исполнение других команд. Предположим, например, что нам надо прочитать с экрана три коротких файла: FIRST.TXT, SECOND.TXT и THIRD.TXT. Вместо того, чтобы вызывать эти файлы отдельными командами, можно записать:

```
FOR %Z IN (FIRST SECOND THIRD) DO TYPE %Z.TXT
```

здесь:

%Z – пример имени локальной переменной цикла, которая последовательно принимает все указанные в скобках значения. Это имя не должно начинаться с цифры и обычно сводится к одной букве. При исполнении из командной строки перед этим именем должен быть указан знак процента (%), а при исполнении из batch-файла – сдвоенный знак процента (%%).

IN – (= в) обязательное зарезервированное слово, предшествующее следующему за ним в скобках перечню значений для локальной переменной цикла.

DO – (= делай) обязательное зарезервированное слово, предшествующее следующему за ним имени команды, которая должна быть исполнена в цикле столько раз, сколько значений локальной переменной цикла указано в скобках, причем каждый раз с подстановкой нового значения вместо указанного в команде имени переменной цикла.

В качестве перечисляемых в скобках значений годятся любые слова, включая подстановки значений переменных окружения (например, %TEMP%) и формальных параметров (2.03-03). Перечисляемые значения можно разделять пробелами, знаками точки с запятой (;) или запятыми (,). Заметьте, что пути в значении переменной %PATH% (3.20) отделены друг от друга знаками точки с запятой. Следовательно, такой список путей будет рассматриваться в цикле FOR не как одно целое, а как перечень из нескольких отдельных путей. Последнее обстоятельство позволяет использовать цикл FOR для выяснения того, будет ли заданный файл найден при автоматическом поиске, а также для нахождения конкретного пути к этому файлу. Пример поиска пути к конкретному файлу Fc.exe показан в следующем фрагменте batch-файла:

```
@echo off
set P=
FOR %%Y IN (. %PATH%) DO if exist %%Y\Fc.exe set P=%%Y\Fc.exe
if %P%"==" echo Requested file hasn't been found!
if not %P%"==" echo Path to the requested file is %P%
```

Здесь во второй строке вспомогательной переменной (P) присваивается пустое значение, а в третьей строке – путь к искомому файлу, если его удается найти. Обратите внимание, что локальной переменной цикла (%%Y) в третьей строке предшествуют два знака процента, как должно быть в batch-файлах. Последние две строки проверяют значение вспомогательной переменной (P) и по результату проверки выдают соответствующее сообщение.

Перечисляемые в скобках значения могут содержать знаки подстановки (2.01-03), но такие значения интерпретируются только как маски файлов, по которым производится поиск файлов в текущем каталоге или согласно указанному при маске пути. Вот пример цикла, в котором каждое из указанных в скобках значений является маской файла:

```
FOR %%X IN (A:\*.txt A:\*.doc) DO COPY /B %%X C:\DOS
```

Иногда желательно отобразить на экране каждую из выполняемых в цикле операций, не воспроизводя на экране строку, задающую сам цикл. Для этого приведенный выше пример нужно модифицировать следующим образом:

```
ECHO ON
@FOR %%X IN (A:\*.txt A:\*.doc) DO COPY /B %%X C:\DOS
@ECHO OFF
```

Если файл, указанный в скобках цикла FOR, не удается найти, то соответствующая операция просто пропускается без выведения сообщения об ошибке. Благодаря этой особенности цикл FOR иногда используют как средство устранения нежелательных сообщений об ожидаемых ошибках (пример в разделе 3.09). Еще один "побочный эффект" цикла FOR состоит в том, что он позволяет "разобрать" состоящее из нескольких слов значение переменной, устраняя при этом лишние пробелы перед или после отдельных слов в этом значении.

Группа слов, включающая разделительные знаки, в скобках цикла FOR может рассматриваться как единое значение, если она заключена с обеих сторон в двойные кавычки, причем сами двойные кавычки не считаются входящими в это значение. Отсутствие замыкающей двойной кавычки воспринимается как ошибка. Использование двойных кавычек позволяет записывать в одной строке несколько разных последовательно исполняемых операций, например:

```
FOR %%Z IN ("set E=%W%" "echo E is set" "goto L23") DO %%Z
```

Встречаются последовательности операций, которые нельзя выполнить в отдельных строках, но можно выполнить в составе цикла FOR. Примеры решения таких задач имеются в 46-й строке batch-файла, приведенного в разделе 9.03-02, и в 6-й строке batch-файла, приведенного в разделе 9.01-03.

Заключаемые в двойные кавычки группы слов могут содержать подстановки значений переменных (например, %W%), условные команды (IF), операторы перехода (goto) и знаки перенаправления (2.04-02 – 2.04-05). Если происходит переход из команды, указанной в цикле FOR, то следующие указанные в этом цикле команды не исполняются.

Если в составе группы слов, заключенной в двойные кавычки, имеется знак перенаправления, то он распространяет свое действие на следующие операции, исполняемые в пределах того же цикла FOR. В частности, в следующем цикле FOR

сообщение от второй операции не индицируется на экране, а попадает в файл вслед за сообщением от первой операции:

```
For %%Z in ("echo 1-st line >> Q.txt" "echo 2-nd line") do %%Z
```

Изменить перенаправление в последующих операциях цикла можно, но его нужно указать явно.

Способность цикла FOR не включать знаки двойных кавычек в значение переменной цикла полезна для обеспечения правильной индикации сообщений, состоящих из нескольких слов и предохраняемых от разделения в промежуточных операциях посредством заключения в двойные кавычки.

Вложенные циклы FOR не допускаются, но тем не менее возможны, если внутренний цикл FOR исполняется отдельным резидентным модулем командного интерпретатора COMMAND.COM, запускаемым из внешнего цикла FOR, указываемого в той же командной строке. Еще одна возможность состоит в том, чтобы из внешнего цикла FOR командой CALL запустить на исполнение batch-файл, в котором содержатся другие циклы FOR.

Примечание 1: имя переменной цикла должно быть выбрано так, чтобы исключить совпадения с другими именами используемых переменных.

Примечание 2: в пределах скобок цикла FOR знак косой черты (/) играет роль особого сепаратора: следующие за ним буквы преобразуются в заглавные, и вместе с самим знаком косой черты образуют отдельное слово, задающее одно из значений переменной цикла. В прежних версиях DOS действие знака косой черты было иным.

Примечание 3: попытки извне перенаправить сразу все сообщения, выводимые из цикла FOR, воздействуют только на первую операцию в этом цикле. Если для последующих операций перенаправления не указаны, то для них будут действовать установки, принимаемые по умолчанию.

Примечание 4: когда резидентный модуль командного интерпретатора COMMAND.COM запускается из командной строки специально для исполнения цикла FOR, то там имени переменной цикла должны предшествовать два знака процента, так же, как в batch-файлах.

3.14 GOTO – команда перехода на метку

Команда GOTO (GO TO = идти к) используется в batch-файлах для выполнения переходов на метку, которая должна быть записана в одной из строк того же batch-файла. Строка с меткой начинается с двоеточия (:), после которого следует произвольное имя метки. Для перехода на эту метку то же самое имя должно быть указано после команды GOTO. В длинных именах меток принимаются во внимание только первые 8 знаков. Наличие нескольких одноименных меток в

одном batch-файле не допускается. Например, если в одной из строк batch-файла имеется метка `:L36`, то команду перехода на эту метку следует записать так:

```
GOTO L36
```

Для исполнения условных переходов команде GOTO в той же строке может предшествовать оператор условия IF (3.15). Большое количество примеров условных и безусловных команд перехода имеется в batch-файлах, приведенных в разделах 9.03-02, 9.09-02.

Примечание 1: в строке с командой GOTO имя метки может быть получено посредством замещения имени переменной (например, %VAR%) ее значением (2.03-03).

Примечание 2: в строке с командой GOTO в качестве имени метки нельзя указывать подстановку значения формального параметра (%1, %2..), но ее можно использовать в составе имени метки, если подстановке предшествует хотя бы одна буква.

Примечание 3: переход по метке не изменяет код ошибки (errorlevel), так что проверки кода ошибки могут быть продолжены после перехода.

3.15 IF – оператор условия

Оператор условия IF (= если) позволяет проверять три вида условий: условия существования, условия равенства и условия превышения заданного значения кода ошибки (errorlevel).

В общем случае командная строка с условным исполнением какой-либо операции начинается с оператора условия IF, за которым следует идентификатор вида условия, формулировка условия, и затем полная спецификация той команды, которую надлежит выполнить в случае удовлетворения условия. В одной строке можно последовательно размещать несколько операторов условия, и в таком случае общий результат получается путем выполнения логической операции И (AND) над частными результатами проверки отдельных условий. Примеры комбинирования нескольких условий приведены в разделах 3.15-03, 9.03-01, 9.09-02. Ниже рассмотрены особенности композиции командных строк с оператором IF для осуществления проверки каждого из видов условий.

3.15-01 Оператор условия IF: проверка существования

Проверка существования применима к файлам, каталогам и логическим устройствам. Она выполняется командой IF EXIST (= если существует). Инверсия этого условия, то есть проверка отсутствия, обеспечивается командой IF NOT EXIST. Вот два примера применения проверки существования по отношению к файлу:

```
IF EXIST C:\DOS\Format.com C:\DOS\Format.com A: /S
IF NOT EXIST C:\DOS\Format.com ECHO Format.com hasn't been found!
```

здесь:

EXIST – зарезервированное слово, определяющее проверку условия существования и интерпретацию ближайшего следующего элемента как имени или маски искомого объекта.

C:\DOS\Format.com – пример имени искомого файла с предшествующим путем. Поскольку путь указан, поиск будет производиться только согласно этому пути. Если путь не указан, то поиск будет производиться только в текущем каталоге.

C:\DOS\Format.com A: /S – пример вызова программы, которая будет запущена на исполнение, если предшествующее условие выполнено. Заметьте, что после имени программы указаны все параметры, необходимые для ее исполнения. Если перед именем этой программы не указывать путь, то будет предпринят ее поиск сначала в текущем каталоге, а потом по всем путям, записанным в значение переменной %PATH%.

NOT – зарезервированное слово, вызывающее логическую инверсию результата, возвращаемого после проверки условия.

ECHO Format.com hasn't been found! – еще один пример команды, исполнение которой поставлено в зависимость от условия, на сей раз условия отсутствия указанного файла.

Приведенные примеры обеспечивают исполнение указываемой программы (FORMAT.COM), если ее удается найти в должном месте, или получение понятного сообщения, если она там не найдена.

Имена логических устройств считаются зарезервированными словами (2-01-01), назначаемыми ядром DOS или драйверами в процессе загрузки. Путем проверки существования имени логического устройства можно выяснить, загружен ли соответствующий драйвер. Полный список заявленных в Вашем компьютере имен логических устройств можно посмотреть с помощью программы MEM.EXE (6.17), если ее запустить с параметром /D. Например, драйвер EMM386.EXE (5.04-02) резервирует имя логического устройства EMMXXXX0. Поэтому проверить факт его загрузки можно посредством следующей командной строки:

```
IF NOT EXIST EMMXXXX0 ECHO The EMM driver isn't loaded!
```

Если в имени искомого объекта содержатся знаки подстановки (2.01-03), то оно интерпретируется как маска файла, и тогда проверка существования будет производиться только по соответствующим файлам, не включая каталоги и имена логических устройств. Условие существования с маской *.* (все файлы) удовлетворяется, если в каталоге существует хотя бы один файл (пример – в разделе

3.09). Чтобы проверить существование каталога, нужно вместо имени искомого файла использовать зарезервированное слово NUL:

```
IF EXIST C:\DOS\NUL ECHO The C:\DOS directory exists!
```

К сожалению, подобные проверки применительно к оптическим дискам CD-ROM могут дать неверный результат вследствие особенностей используемой там файловой системы ISO 9660.

При необходимости обеспечить безостановочное исполнение batch-файлов следует иметь ввиду, что проверка существования файлов и каталогов выполняется безостановочно только на доступных носителях. Файл или каталог могут не существовать, но диск, к которому происходит обращение, должен существовать и быть доступен, то есть должен быть вставлен в дисковод и отформатирован в соответствии с такой файловой системой, которая доступна для MS-DOS7 непосредственно (FAT12, FAT16 и FAT32) или хотя бы с помощью устанавливаемых драйверов. Если заранее неизвестно, доступен ли данный носитель, то безостановочное исполнение проверок существования файлов или каталогов на нем тоже возможно, но требует принятия мер по предотвращению вызова обработчика критических ошибок (8.02-84) и по блокированию выдачи неуместных сообщений (пример – в разделе 9.03-02).

3.15-02 Оператор условия IF: проверка равенства

Условие равенства идентифицируется по наличию двух слов, отделенных друг от друга сдвоенным знаком равенства (= =). Поскольку бессмысленно сравнивать априори известные слова, постольку условие равенства предполагает осуществление одной или нескольких подстановок либо вместо наименований формальных параметров, либо вместо имен переменных окружения (2.03-03). Вопросительный знак и звездочка в сопоставляемых словах не интерпретируются как знаки подстановки (2.01-03), но допускаются на правах обычных букв. Вопреки обычной практике MS-DOS, строчные и заглавные буквы в сопоставляемых словах не считаются равнозначными. Вот два примера использования проверок условия равенства batch-файлах:

```
IF %VAR%==%2 GOTO L23
IF NOT %VAR%==%2 GOTO HELP
```

здесь:

%VAR% – имя переменной VAR, окруженное знаками процента, означает, что в ходе исполнения оно должно быть заменено значением этой переменной.

%2 – цифра с предшествующим знаком процента интерпретируется как номер формального параметра, который в ходе исполнения должен быть заменен значением этого формального параметра.

GOTO L23 – пример команды, которую надлежит выполнить, если будет удовлетворено условие равенства значений указанной переменной (VAR) и указанного формального параметра.

NOT – зарезервированное слово, означающее логическую инверсию результата, возвращаемого проверкой условия.

GOTO HELP – пример команды, которую надлежит выполнить, если проверяемое условие не выполняется.

Конечно, любое из сопоставляемых слов может быть задано явно, без подстановок. Нередко по крайней мере одно из сопоставляемых слов представляет собой объединение явно заданной части с одной или несколькими подстановками. Но нельзя допускать, чтобы место сопоставляемого слова осталось пустым: это считается синтаксической ошибкой. Между тем любой batch-файл потенциально может быть запущен на исполнение без параметров, и тогда вместо замещаемого наименования формального параметра не будет подставлено ничего. Переменные окружения также могут иметь пустое значение. При таких обстоятельствах показанный выше пример условия равенства вполне может оказаться ошибочным. Простейший способ исправить ситуацию состоит в том, чтобы приписать какой-либо знак – например, точку – к обоим сопоставляемым словам:

```
IF %VAR%.==%2. GOTO L23
IF NOT %VAR%.==%2. GOTO HELP
```

Такие условия равенства эквивалентны приведенным выше, но не приводят к ошибкам, когда исполняемые подстановки не возвращают никакого значения. Тот же принцип используется для построения проверок наличия значения переменной окружения или формального параметра:

```
IF .==%CASH%. ECHO The CASH variable has an empty value
```

Особенно внимательно следует подходить к построению условий равенства, когда подставляемое значение переменной окружения может содержать пробелы. Пробелы в слове справа от сдвоенного знака равенства не допускаются. Если значение переменной CASH в приведенном выше примере содержало бы пробелы, то это было бы расценено как синтаксическая ошибка. Но слева от сдвоенного знака равенства допускается подстановка значений переменной, состоящих из нескольких разделенных пробелами слов. В таких случаях только первое, самое левое из этих слов будет принято во внимание при сопоставлении. Эти особенности иллюстрированы приводимыми ниже тремя примерами:

```
IF NOT A: B: C:.==. ECHO Compared items (A: and a dot) are not equal
IF .A: B: C:==.A: ECHO Compared items (.A: and .A:) are equal
IF . B: C:==. ECHO Compared items (dots) are equal too
```

В первой строке сопоставляются A: слева и знак точки справа, во второй - одинаковые комбинации .A: слева и справа, в третьей – знаки точки слева и справа. Во всех трех примерах наличие членов B: C: игнорируется.

3.15-03 Оператор условия IF: проверка кода ошибки

Программы (утилиты) в момент завершения своей работы обычно оставляют код ошибки (errorlevel), который информирует о том, успешно ли завершилась их миссия или какие обстоятельства этому помешали. Код ошибки представляет собой слово в области текущих данных DOS (смещение 14h в A.01-3), которое принято отображать десятичным числом от 0 до 255 без знака. Обычно код ошибки 0 свидетельствует об успешном завершении, а другие значения соответствуют различным причинам неудач, причем интерпретация значений индивидуальна для каждой программы.

Проверка кода ошибки с помощью оператора условия IF идентифицируется по наличию слова ERRORLEVEL. Она дает возможность определить, не превысил ли код ошибки заданную величину, например:

```
IF ERRORLEVEL 1 ECHO Execution has failed
IF NOT ERRORLEVEL 1 ECHO Execution has terminated successfully
```

Условие проверки в первой строке удовлетворяется для всех значений кода ошибки от 1 до 255, то есть для любого из вариантов неудачного завершения. Проверка во второй строке учитывает наличие зарезервированного слова NOT, вызывающего логическую инверсию результата, и потому условие удовлетворяется только для значений меньше единицы. А такое значение только одно – нуль. Следовательно, проверка во второй строке выявляет факт успешного завершения.

Иногда требуется выполнить определенную процедуру только в случае возврата одного конкретного значения кода ошибки. Такие задачи решают путем объединения двух проверок в одной строке. Для выполнения перехода только при одном значении кода ошибки, например, 15, нужно написать командную строку

```
IF NOT ERRORLEVEL 16 IF ERRORLEVEL 15 GOTO ERROR15
```

В этой строке первое условие удовлетворяется для всех значений кода ошибки от 0 до 15, а второе условие – для всех кодов ошибки от 15 до 255. Оба условия вместе удовлетворяются только при одном значении, равном 15, и только тогда будет выполнена команда GOTO ERROR15.

Примечание 1: у некоторых программ ненулевые значения возвращаемого кода ошибки не свидетельствуют о неудаче исполнения.

Примечание 2: все встроенные команды интерпретатора COMMAND.COM (3.01 – 3.34) не возвращают и не изменяют значение кода ошибки.

Примечание 3: комбинирование нескольких разнородных проверок в одной строке выполняется так же, как показано выше для проверок кода ошибки.

3.16 LFNFOR – установка формы представления имен

Команда LFNFOR (Long File Names FOR = для длинных имен) показывает состояние двоичной локальной переменной, а также позволяет установить ее в единицу (LFNFOR ON) и сбросить в нуль (LFNFOR OFF). По умолчанию эта переменная сброшена в нуль (OFF). В MS-DOS7 ее состояние игнорируется, но в "окне DOS" операционных систем Windows-95/98 установление переменной LFNFOR в единицу позволяет правильно воспроизводить длинные имена файлов в циклах FOR, например, таких:

```
FOR %%Z in (*.*) do echo %%Z
```

Когда переменная LFNFOR сброшена в нуль, выводимые в цикле FOR длинные имена файлов подвергаются "обрезанию" до 8 знаков, как это всегда происходит в MS-DOS7.

3.17 LH – команда загрузки в область выше 640 кбайт

Команда LH (Load High) служит для загрузки драйверов и резидентных программ за пределы обыкновенной памяти (выше 640 кбайт) на компьютерах с процессором не древнее 80386. Доступ за пределы обыкновенной памяти должен быть обеспечен заранее указанием команды DOS=UMB (4.08) в файле CONFIG.SYS, загрузкой драйвера HIMEM.SYS (5.04-01) и последующей загрузкой драйвера EMM386.EXE (5.04-02) или драйвера UMBPCI.SYS (5.04-04). В обоих случаях обращение к загруженным резидентным модулям будет происходить через адресное пространство области UMB (640 – 1024 кбайт). Когда в области UMB свободного адресного пространства нет, тогда команда LH не выдает сообщения об ошибке, а просто продолжает размещение загружаемых модулей в обыкновенной памяти, то есть ниже границы 640 кбайт.

Команда LH делает почти то же, что команда INSTALLHIGH (4.16); основная разница состоит в том, что команда INSTALLHIGH, выполняемая загрузчиком IO.SYS, не может принимать участие в оптимизации распределения памяти. Команда LH исполняется командным интерпретатором COMMAND.COM предпочтительно из файла AUTOEXEC.BAT, но может быть исполнена также из командной строки. Вот пример строки файла AUTOEXEC.BAT, в которой драйвер MSCDEX.EXE загружается командой LH:

```
LH /L:1,23680 \DOS\DRV\Mscdex.exe /D:CD1 /E /S /V /L:O /M:32
```

Имени загружаемого драйвера предшествует путь (\DOS\DRV\ ), который может быть задан в любой из допустимых форм (2.02-01). Если путь не указан, то поиск драйвера будет производиться в текущем каталоге, а затем по всем путям, записанным в значение переменной %PATH% (2.02-02). Любые слова, указываемые правее имени драйвера, не идентифицируются командой LH, а просто передаются загружаемому драйверу как параметры.

Между именем команды LH и спецификацией загружаемого драйвера может быть введен необязательный параметр /L, позволяющий указать, через какую часть области UMB следует адресовать загружаемый драйвер и насколько большой участок памяти нужно ему выделить (об этом также 4.07). В приведенном выше примере он представлен как /L:1,23680, где /L:1 означает адресацию через первую часть области UMB, а число 23680 – запрашиваемый размер участка памяти в байтах. Нумерацию частей области UMB можно узнать с помощью программы MEM.EXE (6.17) при ее исполнении с параметром /F.

Запрашиваемый размер участка памяти указывать не обязательно, но если он указан, то команда LH принимает еще один параметр /S, например

```
LH /L:1,2160 /S \DOS\COM\Escape.com
```

Параметр /S означает, что выделяемый блок UMB следует ограничить согласно указанному размеру. Благодаря тому использование области UMB становится более эффективным, но вместе с тем возрастает риск зависания компьютера, если размер окажется выбран неточно. Не рекомендуется указывать параметр /S и конкретный размер после параметра /L без предварительного проведения процедуры оптимизации распределения памяти, выполняемой программой MEMMAKER.EXE (5.04-03). В результате проведения такой процедуры в строки файла AUTOEXEC.BAT с командой LH будут автоматически вписаны параметры /S и /L с точной спецификацией номера части и размера запрашиваемого участка памяти.

Примечание 1: если доступ за пределы обыкновенной памяти открыт драйвером UMBPCI.SYS (5.04-04), то резидентные модули, загружаемые командой LH, будут размещены в области UMB (640 – 1024 кбайт). Но драйвер EMM386.EXE (5.04-02) действует по-другому: он перенастраивает таблицу преобразования адресов в процессоре так, чтобы она обеспечивала возможность обращения к участкам расширенной памяти (выше 1088 кбайт) через то же самое адресное пространство области UMB. Поэтому в случае использования драйвера EMM386.EXE команда LH, обращаясь к области UMB, физически будет загружать резидентные модули не в область UMB, а в участки памяти, расположенные выше границы 1088 кбайт.

3.18 LOCK – запрет конкурентного доступа к диску

MS-DOS7 обслуживает запросы программ на доступ к дискам для обеспечения эффективного управления буферизацией и очередностью доступа. Программам, которым требуется прямой доступ к диску, необходимо координировать свои действия с MS-DOS7 посредством прерывания INT 13\AH=45h (8.01-58). Но встречаются программы, которые этого не делают. К их числу относятся многие полезные программы из прежних версий DOS, например, программа восстановления удаленных файлов Undelete.exe из комплекта MS-DOS6.22. Для обеспечения возможности исполнения таких программ в MS-DOS7 предусмотрена команда LOCK (= запереть), предоставляющая исключительное право на пользование указанным логическим диском той программе, которая будет запущена из следующей командной строки.

Аргументами команды LOCK служат одна или несколько букв дисков, которые надлежит предоставить в исключительное пользование. Перед исполнением команды LOCK запрашивается подтверждение, на которое надо ответить нажатием клавиши Y (да) или N (нет). Чтобы обеспечить безостановочное исполнение команды LOCK в batch-файлах, на запрос нужно заранее заготовить ответ, например, так:

```
ECHO Y | LOCK C: D:
```

В результате исполнения такой командной строки запускаемая затем программа получит прямой доступ к логическим дискам C: и D:, не прерываемый запросами от других программ. Когда программа, требовавшая прямого доступа, завершит свою миссию, действие команды LOCK должно быть отменено командой UNLOCK (3.31). Поскольку операции прямого доступа могут быть вложенными, постольку предусмотрена возможность до 256 уровней вложенности команды LOCK. Разумеется, каждая участвующая в таких процессах программа должна быть обеспечена должной последовательностью команд LOCK и UNLOCK.

3.19 MD – создание каталога

Команда MD (Make Directory = создать каталог) позволяет создать каталог или подкаталог, например:

```
MD  C:\DOS\ARC
```

здесь:

C:\DOS\ – пример пути к существующему каталогу, в котором должен быть создан новый подкаталог.

ARC – пример имени для нового подкаталога. Имя не должно оканчиваться обратной косой чертой, но если она уже имеется, ее

можно сделать недействительной, добавив в конце точку (2.02-03).

Имя для нового подкаталога должно быть уникальным, то есть другого такого имени в том же каталоге не должно быть. Если предшествующий имени путь не указан, то новый подкаталог будет создан в текущем каталоге текущего диска.

Примечание 1: MKDIR – еще одно имя той же самой команды MD.

3.20 PATH – спецификация путей поиска

Команда PATH (= путь) задает перечень путей, по которым производится поиск вызываемой программы (утилиты) в том случае, если она не находится в текущем каталоге и если путь к ней не указан в командной строке. Задаваемый перечень путей становится значением одноименной переменной окружения PATH. Записать пути в значение переменной окружения можно также с помощью команды SET (3.26), но у команды PATH имеется важное отличие: она автоматически преобразует все строчные буквы в указываемых путях в заглавные. Это необходимо для правильной идентификации имен каталогов в процессе поиска. После имени команды PATH в командной строке должен следовать перечень из путей поиска, указываемых в любой из допустимых форм (2.02-01) и отделяемых друг от друга знаками точки с запятой, например:

```
PATH  C:\DOS\VC4;C:\DOS\MS7;C:\WINDOWS\COMMAND
```

С помощью команды PATH можно подвергать преобразованию буквы не только в спецификациях путей, но также в составе любых других слов.

Примечание 1: в перечне путей с обеих сторон от знаков точки с запятой не должно быть пробелов или других разделительных знаков.

Примечание 2: не допускается наличие знака точки с запятой в конце перечня путей.

Примечание 3: команда PATH без параметров показывает установленное значение переменной PATH.

Примечание 4: команда PATH ; со следующим за ней знаком точки с запятой удаляет установленное значение переменной PATH.

Примечание 5: в качестве разделительного знака между именем команды PATH и следующим за ним перечнем путей можно указывать не только пробел, но также знак равенства.

3.21 PAUSE – временная остановка исполнения

Команда PAUSE (= пауза) останавливает исполнение batch-файла и выводит на экран сообщение "Press any key to continue..." ("Для продолжения нажмите любую

клавишу"). Это сообщение не вполне верно, потому что клавишные комбинации CTRL-C, CTRL-BREAK и ALT-03 вызывают не продолжение, а прекращение исполнения batch-файла, причем действие этих клавишных комбинаций не зависит от команды BREAK. Если выводимое командой PAUSE сообщение нежелательно, то от него легко избавиться путем перенаправления:

```
PAUSE > NUL
```

После команды PAUSE в той же строке batch-файла может быть комментарий, как после команды REM. Если ECHO-флаг не установлен (3.11), то комментарий на экране не воспроизводится.

Когда принимаемые по умолчанию связи с консолью прерваны (например, командой CTTY NUL, 3.07), тогда для команды PAUSE необходимо явно указывать перенаправление ввода:

```
PAUSE < CON
```

Перенаправление команде PAUSE управляющего кода 03h (A.02-08), показываемого знаком ♥, вызывает немедленное прерывание исполнения batch-файла:

```
ECHO ♥| PAUSE > NUL
```

Управляющий код 03h вводится нажатием клавиш ALT-03, причем цифры 03 должны быть набраны числовыми клавишами в правой части клавиатуры.

В последнем примере важно обратить внимание на то, что команда PAUSE не получит перенаправляемый знак, если, например, перед ECHO поставить команду REM или если DOS не запишет данные промежуточного перенаправления во временный файл из-за отсутствия записываемого носителя. Чтобы компьютер не завис в бесконечном цикле ожидания, подобных ситуаций следует избегать.

3.22 PROMPT – установление вида приглашения

Команда PROMPT (= подсказка) переопределяет значение одноименной переменной окружения, которым командный интерпретатор руководствуется при формировании приглашения командной строки. Обычно команда PROMPT записывается в одной из строк файла AUTOEXEC.BAT, но может быть введена из обычной командной строки. Вслед за именем команды PROMPT указывается текст приглашения. В этом тексте особым образом интерпретируются и замещаются группы из двух знаков, начинающиеся со знака доллара. Ниже перечислены соответствия между замещаемыми группами знаков и тем, чем именно они замещаются:

$Q	знак равенства (=)
$$	знак доллара ($)
$T	текущее время
$D	текущая дата
$P	текущие диск и путь
$V	номер версии Windows
$N	текущий диск
$G	знак "больше", т.е. правая стрелка (>)
$L	знак "меньше", т.е. левая стрелка (<)
$B	вертикальный штрих (|)
$H	код 08h "Backspace" (A.02-08).
$_	коды 0Dh 0Ah "Возврат каретки" и "Перевод строки" (A.02-08).
$E	код 1Bh "Выйти" ("Escape", A.02-08).

Команда PROMPT без последующего текста удаляет значение переменной PROMPT, и тогда по умолчанию формируется приглашение командной строки, состоящее из буквы текущего диска и правой стрелки, то есть такое же, какое устанавливается командой PROMPT NG. В MS-DOS7 переменной PROMPT автоматически присваивается иное значение (PROMPT PG), которое определяет привычный вид приглашения, включающий полный путь к текущему каталогу и заканчивающийся знаком правой стрелки.

Данные, выводимые в приглашении командной строки, могут быть записаны в файл и затем присвоены как значение переменной окружения. В качестве примера рассмотрим запись в переменную окружения пути к текущему каталогу, выполняемую следующими строками batch-файла:

```
Prompt @echo off$_Set Ret$q$p
C:\Command.com /c Ret.bat > Ret.bat
Call Ret.bat
```

Первая строка задает сложный вид приглашения, а вторая строка записывает это приглашение в другой batch-файл RET.BAT. Заметьте: если файл RET.BAT уже существует, то его прежнее содержание будет полностью утрачено. Записываемое новое содержание файла RET.BAT будет выглядеть так:

```
@echo off
Set Ret=D:\BACKUP
```

Обратите внимание, что указываемый после команды PROMPT фрагмент текста "$_Set Ret$q$p" в приглашении оказался преобразован в отдельную строку "Set Ret=D:\BACKUP", где "D:\BACKUP" – это полный путь к тому каталогу, который являлся текущим на момент исполнения строк batch-файла. Когда в третьей строке показанного выше примера команда CALL запускает файл RET.BAT на исполнение, полный путь оказывается записанным в значение

переменной окружения RET. После этого файл RET.BAT можно удалить, а путь, записанный в переменную окружения RET, позволит в любой момент вернуться к тому же каталогу того же диска:

```
%Ret%\
CD %Ret%
```

Пример использования команды PROMPT для определения буквы текущего диска приведен в разделе 9.01-03. Время, дата, и версия операционной системы могут быть определены подобным же образом.

3.23 RD – удаление каталога

Чтобы с помощью команды RD (Remove Directory = удалить каталог) можно было бы удалить каталог, должны быть удовлетворены следующие условия:

– удаляемый каталог должен быть пуст;
– диск с удаляемым каталогом должен быть доступен для записи;
– удаляемый каталог не должен быть корневым;
– удаляемый каталог не должен быть текущим на своем диске, даже если этот диск не является текущим в данный момент.

Вот пример использования команды RD:

```
RD D:\TEMP\NOTES
```

здесь:

NOTES – пример имени удаляемого пустого каталога.
D:\TEMP\ – пример пути к удаляемому каталогу. Путь может быть указан в любой из допустимых форм (2.02-01, 2.02-03) или вовсе не указан; в последнем случае будет предположено, что удалению подлежит подкаталог текущего каталога.

Примечание 1: RMDIR – еще одно имя той же команды RD.

3.24 REM – вставка комментария

Командный интерпретатор воспринимает REM (REMark = замечание) как команду игнорировать все остальные слова в той же строке до первого встреченного знака перенаправления (2.04-02 – 2.04-05) или до конца командной строки. Основная миссия команды REM – обеспечить возможность введения строк комментариев в batch-файлах. Максимальная длина строки комментариев – 123 знака. С помощью команды REM вводят такие комментарии, которые не нужно воспроизводить на экране при нормальном исполнении отлаженного batch-файла, но которые полезны для ориентации в тексте файла или для понимания сути

записанных там команд. Эти комментарии воспроизводятся на экране только в процессе отладки, пока не сброшен ECHO-флаг (3.11).

Команду REM иногда вставляют в начало строки специально для того, чтобы временно предотвратить исполнение записанной в данной строке другой команды. Однако надо иметь ввиду, что команда REM не может предотвратить исполнение перенаправления, тогда как сдвоенное двоеточие может (2.04-01).

Еще одна миссия команды REM – это миссия "пустой" команды, которая формально исполняется, но тем не менее не делает ничего. Пример такого применения команды REM показан в файле VCEDIT.EXT в разделе 6.25-03. Поскольку команда REM не посылает ничего в канал STDOUT, постольку перенаправление ее пустого выходного сообщения в файл часто используют для создания пустых файлов нулевой длины:

```
REM > Anyfile.ext
```

Если перенаправление пустого выходного сообщения производится в существующий файл, то его прежнее содержание утрачивается. Помимо прочего, оказывается затерт адрес первого кластера файла в спецификации каталога, и потому восстановить утраченное содержание с помощью программы UNDELETE.EXE оказывается невозможно.

Примечание 1: опасно ставить команду REM в начале строки с промежуточным перенаправлением (2.04-05). Не будет исполнена только та команда, которая стоит слева от знака промежуточного перенаправления, создаваемый временный файл окажется пустым, и потом команда, стоящая справа от знака промежуточного перенаправления, начнет безуспешно ждать считывания из пустого файла. Кончается это, как правило, “зависанием” компьютера (пример – в разделе 3.21).

Примечание 2: в "окне DOS" операционных систем Windows перенаправление выходного сообщения команды REM не исполняется, и там создать пустой файл таким способом нельзя.

3.25 REN – переименование файлов

Команда REN (REName = переименовать) позволяет переименовать один файл или сразу несколько файлов, если их исходные имена соответствуют определенной маске. Вот пример использования команды REN для переименования одного файла:

```
REN C:\DOS\Notes.txt Notes.old
```

здесь:

C:\DOS\ – пример пути к файлу, который надлежит переименовать. Путь может быть указан в любой из допустимых форм (2.02-01,

2.02-03) или быть опущен, если переименовываемый файл находится в текущем каталоге.

Notes.txt – пример исходного имени переименовываемого файла.

Notes.old – пример назначаемого имени. Оно не должно совпадать с имеющимися именами объектов в том же каталоге и должно быть указано без предшествующего пути, даже если файл находится за пределами текущего каталога.

Пользование маской файла вместо одного исходного (старого) имени допускается, но часто ведет к ошибке: попытке создать несколько одноименных файлов в одном каталоге. Поэтому рекомендуется в назначаемом имени тоже "закрывать" знаками подстановки (см. 2.01-03) те позиции знаков, в которых целесообразно оставить без изменений знаки из старых (исходных) имен, чтобы в результате переименования разные файлы получили отличающиеся имена. Допустим, что в текущем каталоге имеется группа файлов PART_01.TXT – PART_12.TXT, которые надо переименовать в CHAP_01.TXT – CHAP_12.TXT. Такая задача решается в одну строку командой

```
REN PART_??.TXT CHAP_??.TXT
```

Примечание 1: файлы с атрибутом H (скрытые) команда REN не переименовывает.
Примечание 2: атрибуты переименовываемых файлов не изменяются.

3.26 SET – присвоение значения переменной окружения

Когда команда SET (= установить) используется без параметров, она показывает действующие в данный момент значения всех переменных окружения. Если же в строке после команды SET следует слово, то оно интерпретируется как имя переменной окружения, которой следует присвоить новое значение, например:

```
SET  TEMP=D:\Temp
```

здесь:

TEMP – пример имени переменной окружения.

D:\Temp – пример значения, которое следует присвоить переменной TEMP. В составе значения не должно быть знаков равенства и знаков перенаправления (2.04-02 - 2.04-05).

Примечание 1: справа от разделяющего знака равенства все пробелы, как предшествующие назначаемому значению, так и следующие после него до конца строки, будут включены в присваиваемое значение переменной окружения.

Примечание 2: если справа от разделяющего знака равенства значение не указано, то данная переменная удаляется и перестает существовать.

Примечание 3: команда SET способна расширить пространство окружения, если его оказывается недостаточно для размещения нового значения.

Примечание 4: в batch-файлах назначаемое значение переменной может включать подстановки значений переменных (например, %Var2%) и формальных параметров (например, %2 %3, 2.03-03). Все такие подстановки будут сначала исполнены, и только потом полученный результат будет присвоен указанной переменной в качестве нового значения.

Примечание 5: одноименная команда SET (4.25) в строках файла CONFIG.SYS интерпретируется загрузчиком IO.SYS, который не исполняет подстановки и перенаправления, но именно благодаря этому дает возможность включения знаков подстановки и перенаправления в состав значений переменных окружения.

3.27 SHIFT – смещение нумерации параметров

Команда SHIFT (= сдвинуть) уменьшает на единицу порядковые номера всех параметров batch-файла (2.03-03), так что прежний параметр %0 становится недоступен, прежний параметр %1 становится %0, прежний параметр %2 становится %1, и т.д. Важно, что параметром %9 становится тот, который был недоступен прежде. Это дает возможность указывать более 9 параметров в командной строке и последовательно обращаться к ним, смещая их нумерацию от цикла к циклу с помощью команды SHIFT. Примером такого цикла может служить подпрограмма, представленная в строках 29 – 38 файла DISK.BAT из раздела 9.03-02. Когда последовательность указанных в командной строке параметров исчерпывается, очередной параметр оказывается имеющим пустое значение, и это служит признаком выхода из подобных циклов.

3.28 TIME – установка времени

Для установки времени вслед за именем команды TIME (= время) в командной строке должно быть указано время, которое надлежит установить, например:

```
TIME 11:39:23,24
```

Числа за именем команды TIME последовательно обозначают часы, минуты, секунды и сотые доли секунды. В данном случае в качестве разделительных знаков в спецификации времени использованы двоеточия и запятая, но это зависит от установки национальных данных командой COUNTRY (4.05).

Когда время в командной строке не указано, на экран выводится текущее время вместе с приглашением ввести новое время. Если изменять время Вы не намерены, достаточно ответить на приглашение нажатием клавиши ENTER.

Чтобы дополнить текстовый файл строкой с указанием текущего времени, команду TIME можно использовать, например, следующим образом:

```
ECHO= | TIME | FIND.EXE "Current" >> ANYFILE.TXT
```

В представленной строке первое промежуточное перенаправление (ECHO= | TIME) обеспечивает безостановочное исполнение, второе промежуточное перенаправление (TIME | FIND.EXE) исключает нежелательные строки выходного сообщения, а третье перенаправление (>> ANYFILE.TXT) приписывает строку сообщения в конец указанного файла. При этом, конечно, должны быть удовлетворены все условия исполнения перенаправлений, а также условия успешного обнаружения задействуемых файлов.

3.29 TRUENAME – приведение пути в каноническую форму

Любая спецификация, следующая за именем команды TRUENAME (= истинное имя) в той же командной строке, интерпретируется как имя объекта – файла или каталога, которому может предшествовать путь. Команда TRUENAME не проверяет их соответствие физической реальности, а преобразует спецификацию в каноническую форму, которая обязательно начинается с буквы диска и включает полный путь. Если путь в спецификации неполный или его вообще нет, то вместо недостающих элементов приписываются путь к текущему каталогу и буква текущего диска. Если в спецификации пути обнаруживаются знаки "точка" и "точка-точка" (2.02-03), то они таким же образом заменяются конкретным путем. Помимо того, все буквы преобразуются в заглавные, знаки прямой косой черты заменяются знаками обратной косой черты, знаки подстановки "звездочка" (2.01--03) заменяются соответствующим числом вопросительных знаков, длинные имена урезаются до 8 знаков, длинные суффиксы – до трех знаков.

Если в исходной спецификации указан фиктивный путь, "сфабрикованный" программами ASSIGN.COM, JOIN.EXE или SUBST.EXE, то команда TRUENAME возвращает истинный путь. Когда команда TRUENAME исполняется вообще без параметров, она возвращает путь к текущему каталогу, так же как команда CD.

Примечание 1: команда TRUENAME не выдает сообщений об ошибках.
Примечание 2: команда TRUENAME не может правильно распознавать сетевые пути, если не установлен драйвер локальной сети.
Примечание 3: действие команды TRUENAME основано на прерывании INT 21\AH=60h (8.02-72).

3.30 TYPE – считывание файла

Команда TYPE (= напечатай) посылает содержимое считываемого файла в канал STDOUT, который по умолчанию выводит посланное сообщение на экран

дисплея. Прокрутку строк воспроизводимого текста можно временно остановить клавишами CTRL-S или BREAK до нажатия любой другой клавиши, или вообще прервать считывание нажатием клавиш CTRL-C или CTRL-BREAK. Вот пример вызова команды TYPE:

```
TYPE  C:\DOS\Notes.txt
```

здесь:

Notes.txt – пример имени считываемого файла. Замена имени маской файла в команде TYPE не допускается.

D:\DOS\ – пример пути к считываемому файлу. Путь может быть указан в любой разрешенной форме (2.02-01, 2.02-03). Если путь не указан, то поиск файла производится только в текущем каталоге.

Выводимое командой TYPE сообщение может быть перенаправлено на принтер, подключенный к параллельному порту LPT1:

```
TYPE A:\Config.sys > PRN
```

Совместно с командой TYPE часто используется программа просмотра MORE.COM (6.19), организующая постраничный вывод сообщений на экран.

3.31 UNLOCK – разрешение конкурентного доступа к диску

После завершения исполнения каждой программы, получившей прямой доступ к конкретному диску с помощью команды LOCK (3.18), исходное состояние должно быть восстановлено командой UNLOCK (= отпереть), например, так:

```
UNLOCK C:
```

Фактически команда UNLOCK не разрешает конкурентный доступ к диску, а лишь уменьшает на единицу уровень запрета на конкурентный доступ, чтобы обеспечить правильное обслуживание вложенных вызовов программ. Но если исходный уровень запрета был равен единице, команда UNLOCK восстановит должное функционирование очереди запросов на доступ к конкретному диску. Это особенно важно в многозадачной операционной среде, например, в "окне DOS" операционной системы Windows-95/98. Если данный диск уже открыт для конкурентного доступа, то никаких действий команда UNLOCK не производит, и сообщений об ошибке не выдает.

3.32 VER – версия операционной системы

Команда VER (VERsion = версия) в MS-DOS7 и в MS-DOS8 показывает номер версии операционной системы WINDOWS, в поставку которой входят эти версии DOS. Если команда VER исполняется с параметром /R

```
VER /R
```

то в дополнение к номеру версии операционной системы выводится краткое сообщение о том, загружено ли ядро DOS в область верхней памяти или нет.

3.33 VERIFY – сверка копий файлов с оригиналами

Команда VERIFY (= сверить) без параметров показывает состояние служебной двоичной переменной, от которой зависит проведение сверки копии файла с оригиналом после каждого акта копирования файла. По умолчанию эта переменная находится в сброшенном состоянии (OFF), при котором сверка не проводится. Изменять состояние этой переменной можно командами VERIFY ON и VERIFY OFF (еще об этом – в примечании 2 к 8.02-60).

Примечание 1: высокая надежность современных дисководов на жестких магнитных дисках делает сверку лишней тратой времени. Рекомендуется включать сверку параметром /V команды COPY только при копировании файлов на дискеты.

Примечание 2: двоичная переменная VERIFY не меняет своего состояния, когда вторичный модуль командного интерпретатора выгружается из памяти, и все локальные переменные окружения теряются.

3.34 VOL – метка и серийный номер диска

Метка диска – это слово длиной до 11 знаков, которое может быть задано пользователем при форматировании. Если пользователь не захочет задавать метку, то в качестве метки будет записано NO NAME (безымянный). Метку можно изменить позже с помощью программы LABEL.EXE или встроенными средствами файл-менеджеров (Norton Commander, Volcov Commander, и т.п.).

Серийный номер – это 8-значный шестнадцатеричный идентификатор, который автоматически присваивается диску при форматировании. Дискеты, форматируемые в процессе изготовления способом контактного копирования, серийного номера не имеют. При обычном копировании дискет и метка, и серийный номер дискеты-оригинала наследуются дискетой-копией.

Чтобы вывести на экран метку и серийный номер диска, нужно после команды VOL (VOLume = том) указать букву этого диска, например

```
VOL A:
```

Если букву диска опустить, то по умолчанию будут выведены метка и серийный номер того диска, который в данный момент является текущим.

Глава 4. Команды файла CONFIG.SYS

Конфигурация загрузки MS-DOS7 определяется параметрами и командами, записанными в три неформатированных текстовых файла, которые находятся в корневом каталоге загрузочного диска: MSDOS.SYS (5.01-01), CONFIG.SYS (9.01-01) и AUTOEXEC.BAT (9.01-02). Среди них файл CONFIG.SYS имеет наиболее долгую историю в предшествовавших версиях DOS. От него зависят важнейшие особенности конфигурации, в том числе состав загружаемых драйверов. Каждая строка в файле CONFIG.SYS – это команда загрузчику IO.SYS (5.01-01). Загрузчик интерпретирует команды совсем не так, как основной командный интерпретатор – COMMAND.COM (6.04). И большинство команд, и синтаксис в файле CONFIG.SYS другие. Хотя некоторые конфигурационные команды (4.02, 4.24, 4.25) называются так же, как команды интерпретатора COMMAND.COM, загрузчик IO.SYS обрабатывает их иначе. IO.SYS требует обязательного указания суффиксов файлов, не исполняет перенаправлений, не замещает имена переменных окружения их значениями. Порядок выполнения команд определяется не только порядком их расположения в файле CONFIG.SYS, но также их приоритетами (подробнее в 4.15 и 4.25). Существует группа команд (4.19, 4.20, 4.21, 4.29) которые можно указывать только в блоках меню и субменю, когда файл CONFIG.SYS имеет блочную структуру, причем другие команды (кроме 4.23) в этих блоках использовать нельзя.

В MS-DOS7 был введен ряд новых конфигурационных команд (4.01, 4.04, 4.11, 4.13, 4.18, 4.28), а некоторые команды (4.08, 4.30) изменены, дополнены новыми параметрами. Старые команды загрузки драйверов в MS-DOS7 исполняются иначе – по умолчанию осуществляют загрузку за пределы обыкновенной памяти: команда DEVICE исполняется как DEVICEHIGH, команда BUFFERS – как BUFFERSHIGH, и т.д. Если какой-либо драйвер нужно загружать в обыкновенную память, то принятые по умолчанию установки необходимо отменять указанием параметра NOAUTO в команде DOS (4.08). Эти и другие особенности исполнения команд из файла CONFIG.SYS рассмотрены детально в последующих разделах 4-й главы.

4.01 ACCDATE – регистрация даты последнего доступа

Команда ACCDATE (ACCess DATE = дата доступа) позволяет разрешить или запретить занесение даты последнего доступа к файлу в соответствующую этому файлу запись в каталоге (A.09-1). По умолчанию дата доступа записывается на жесткие магнитные диски, но не записывается на дискеты. После команды ACCDATE нужно указать перечень дисков со знаками плюс (разрешить) или минус (запретить), например:

```
ACCDATE C+ D- E- R-
```

Число дисков в перечне не ограничено. Если быстродействие дисковых операций имеет значение, то занесение даты доступа лучше запретить.

4.02 BREAK – управление перехватом доступа к дискам

Команда BREAK (= прервать) изменяет состояние двоичной переменной (флага), от которой зависит перехват обращений к дисководам. По умолчанию эта переменная сброшена (OFF), так что нажатия клавиш BREAK и CTRL-C проверяются только тогда, когда DOS обращается к драйверу консоли для выдачи сообщений на экран или для ввода данных с клавиатуры. Только в эти моменты времени нажатия упомянутых клавиш останавливают исполнение программ. Но посредством команды

```
BREAK ON
```

можно дополнительно включить перехват дисковых операций. Это дает возможность останавливать исполнение программ в моменты обращения к дисководам. Вместе с тем быстродействие дисковых операций немного снижается. Отменить перехват дисковых операций можно командой

```
BREAK OFF
```

После того как загрузчик IO.SYS завершит свою миссию, команду BREAK будет исполнять командный интерпретатор COMMAND.COM (3.01).

4.03 BUFFERS – выделение памяти под буферы

Команда BUFFERS (= буферы) резервирует память под буферы по 512 байт каждый, обслуживающие доступ к дискам. По умолчанию MS-DOS7 создает 30 первичных буферов и 0 вторичных. С помощью команды BUFFERS допускается устанавливать от 1 до 99 первичных буферов и от 0 до 8 вторичных буферов. Вторичные буфера нужны тогда, когда предполагается осуществлять двойное буферирование с помощью драйвера DBLBUFF.SYS (5.06-02). Например, команда

```
BUFFERS=12,6
```

обеспечит выделение 9 кбайт памяти под 12 первичных и 6 вторичных буферов. При количестве буферов менее 30 операции доступа к дискам несколько замедляются. Когда загружен драйвер SMARTDRV.EXE (5.06-01), количество буферов может быть уменьшено до 10.

Примечание 1: по умолчанию буферы размещаются за пределами обыкновенной памяти (выше 640 кбайт), но будут размещены ниже границы 640 кбайт, если в команде DOS (4.08) указан параметр NOAUTO, а также если адресное пространство в области UMB недостаточно или недоступно (условия доступности – в разделе 4.07). В любом случае сообщение об ошибке не выдается.

Примечание 2: встречаются компьютеры, у которых контроллер прямого доступа в память (DMA) не обслуживает область UMB или ее часть, даже когда драйвер UMBPCI.SYS (5.04-04) открыл для доступа всю область UMB. В таких компьютерах лучше размещать буферы в обыкновенной памяти или в области выше 1088 кбайт, открываемой драйвером EMM386.EXE (5.04-02). Однако иногда эту проблему может решить дополнительный драйвер LOWDMA.SYS, поставляемый вместе с UMBPCI.SYS.

4.04 BUFFERSHIGH – выделение памяти под буферы

Команда BUFFERSHIGH делает то же самое, что команда BUFFERS (4.03), но только размещает буферы за пределами обыкновенной памяти независимо от того, указан ли параметр NOAUTO в команде DOS (4.08). Все остальные сведения из раздела 4.03 в равной мере применимы к команде BUFFERSHIGH.

4.05 COUNTRY – обеспечение национальной адаптации

Команда COUNTRY (= страна) осуществляет выборочное считывание данных из файла COUNTRY.SYS (5.02-01) и изменяет национальные настройки DOS (A.02-4, A.02-5) в соответствии со считанными данными. В частности, это обеспечивает доступ к файлам, имена которых содержат знаки национальных алфавитов. Вот пример спецификации команды COUNTRY:

```
COUNTRY=007,866,C:\DOS\DRV\COUNTRY.SYS
```

здесь:

007 – код страны для России.
866 – кодовая страница со знаками алфавита для России.
C:\DOS\DRV\ – пример пути к файлу COUNTRY.SYS.

Примечание 1: номера кодовых страниц и коды других стран приведены в приложении A.02-2.

4.06 DEVICE – загрузка драйверов устройств

Команда DEVICE (= устройство) загружает в память драйверы устройств, снабженные заголовком специального формата (A.05-1). Такие драйверы загружаются до завершения формирования системных структур DOS и обычно имеют суффикс *.SYS. У драйверов с суффиксами *.COM и *.EXE наличие специального заголовка не обязательно, и если его нет, то загружать такой драйвер следует не командой DEVICE, а командой INSTALL (4.15).

Вот пример загрузки драйвера с суффиксом *.SYS командой DEVICE:

```
DEVICE=C:\DOS\DRV\HIMEM.SYS /EISA /V
```

здесь:

HIMEM.SYS – пример имени загружаемого драйвера.
C:\DOS\DRV\ – пример пути к загружаемому драйверу.
/EISA /V – пример группы параметров, передаваемых загружаемому драйверу. Эти параметры специфичны для каждого конкретного драйвера.

Вот еще пример загрузки другого драйвера с другой группой параметров:

```
DEVICE?=\DOS\DRV\EMM386.EXE RAM /V
```

В последнем примере важно отметить два различия. Во-первых, путь (\DOS\DRV\) без указания буквы диска дает возможность загружать DOS с заранее неизвестного диска. Во-вторых, после команды DEVICE поставлен знак вопроса. Он вызывает выведение данной строки на экран дисплея с вопросом, загружать ли указанный в ней драйвер или нет:

```
[Enter=Y, Esc=N]?
```

Затем загрузчик останавливает исполнение и ожидает ответа пользователя нажатием клавиши Enter (да) или нажатием клавиши Esc (нет).

Примечание 1: загружаемые драйверы по умолчанию размещаются за пределами обыкновенной памяти (выше 640 кбайт), но будут размещены ниже границы 640 кбайт, если в команде DOS (4.08) указан параметр NOAUTO, а также если адресное пространство в области UMB недостаточно или недоступно (условия доступности – в разделе 4.07). В любом случае сообщение об ошибке не выдается.

Примечание 2: команда DEVICE не дает возможности оптимизировать распределение адресного пространства в области UMB. Если это существенно, то вместо команды DEVICE следует применять команду DEVICEHIGH (4.07).

4.07 DEVICEHIGH – загрузка драйверов устройств

Команда DEVICEHIGH делает то же самое, что команда DEVICE (4.06), но только размещает загружаемый драйвер за пределами обыкновенной памяти всегда, когда доступно адресное пространство в области UMB, независимо от того, указан ли параметр NOAUTO в команде DOS (4.08). Чтобы открыть адресное пространство области UMB (640 – 1024 кбайт), нужно выполнить следующие условия:

– иметь процессор не более древний, чем 80386;
– указать в команде DOS (4.08) параметр UMB;
– первым загрузить драйвер HIMEM.SYS (5.04-01) командой DEVICE;
– потом загрузить командой DEVICE драйвер EMM386.EXE (5.04-02) или драйвер UMBPCI.SYS (5.04-04).

Простая строка с командой DEVICEHIGH выглядит, например, так:

```
DEVICEHIGH=C:\DOS\DRV\SETVER.EXE
```

Указанный драйвер (в частности, SETVER.EXE) будет загружен так, чтобы его можно было бы адресовать через ту часть адресного пространства области UMB, где больше свободного места, при условии, что этого свободного места достаточно для размещения резидентного модуля данного драйвера.

Так же как и к команде DEVICE (4.06), к команде DEVICEHIGH в конце можно приписать вопросительный знак: DEVICEHIGH?=... . Он вызывает запрос к пользователю о подтверждении загрузки указанного в данной строке драйвера.

Команда DEVICEHIGH дает возможность указать конкретную часть адресного пространства области UMB, через которую предстоит обращаться к данному драйверу, например:

```
DEVICEHIGH /L:1,15792 =C:\DOS\DRV\DISPLAY.SYS CON=(EGA,,1)
```

здесь:

/L:1 – пример номера части адресного пространства области UMB: его можно узнать из листинга, выводимого программой MEM.EXE (6.17) при ее запуске с параметром /F.

15792 – необязательная спецификация размера участка памяти, который следует выделить для данного драйвера, причем этот размер обычно не совпадает с размером файла.

Если конкретный драйвер допускает загрузку в несколько частей адресного пространства, то при параметре /L можно указать несколько номеров, как со спецификацией размеров, так и без нее, например:

```
/L:2;3
```

или

```
/L:2,12192;3,3600
```

Заметьте, что перед численной спецификацией размера стоит запятая, тогда как спецификации разных частей адресного пространства отделены друг от друга точкой с запятой.

Если численная спецификация размера указана, то команда DEVICEHIGH принимает дополнительный параметр /S, например:

```
DEVICEHIGH /L:1,35008 /S =C:\DOS\DRV\MOUSE.SYS
```

Параметр /S означает, что выделяемый блок UMB следует ограничить согласно указанному размеру. Благодаря тому использование адресного пространства становится более эффективным, но вместе с тем возрастает риск зависания компьютера, если размер будет выбран неточно. Не рекомендуется указывать параметр /S и конкретный размер после параметра /L без предварительного проведения процедуры оптимизации распределения памяти, выполняемой программой MEMMAKER.EXE (5.04-03). В результате проведения такой процедуры в строки файла CONFIG.SYS с командой DEVICEHIGH будут автоматически вписаны параметры /S и /L с точной спецификацией номера части адресного пространства и размера запрашиваемого участка памяти.

Примечание 1: когда адресное пространство в области UMB недостаточно или недоступно, команда DEVICEHIGH размещает загружаемый драйвер в обыкновенной памяти (ниже границы 640 кбайт). Сообщение об ошибке при этом не выдается.

4.08 Команда DOS – режимы загрузки DOS

По умолчанию ядро операционной системы MS-DOS7 загружается в обыкновенную память, то есть в пределах 640 кбайт. Но если драйвер HIMEM.SYS (5.04-01) уже установлен, то загрузка ядра DOS может производиться в область 1024 – 1088 кбайт, называемую областью верхней памяти. Для этого в файле CONFIG.SYS должна быть указана команда

```
DOS=HIGH
```

Последующая установка менеджера расширенной памяти EMM386.EXE (5.04-02) или драйвера UMBPCI.SYS (5.04-04) открывает возможность загружать служебные структуры DOS и драйверы устройств за пределами обыкновенной памяти (выше 640 кбайт). Чтобы MS-DOS7 воспользовалась этой возможностью, в файле CONFIG.SYS должна быть указана команда

```
DOS=UMB
```

В MS-DOS7 команде DOS добавлен еще один параметр – NOAUTO. Он отменяет ряд принимаемых по умолчанию установок, в частности, загрузку по умолчанию драйверов HIMEM.SYS, DBLBUFF.SYS, IFSHLP.SYS, DBLSPACE.SYS, а также исполнение загрузки за пределы обыкновенной памяти с помощью команд DEVICE, INSTALL и некоторых других (4.03, 4.06, 4.10, 4.12, 4.15, 4.17, 4.27). Параметр NOAUTO фактически позволяет использовать MS-DOS7 как отдельную, независимо конфигурируемую операционную систему.

Допускается указание всех параметров команды DOS в одной строке:

```
DOS=HIGH,UMB,NOAUTO
```

Примечание 1: команда DOS еще принимает недокументированный параметр SINGLE, позволяющий выйти в MS-DOS7 тогда, когда по умолчанию была бы загружена операционная система Windows. Однако этот путь сопровождается неуместными вопросами к пользователю с риском выпасть в перезагрузку. Потому для выхода в MS-DOS7 используют другие способы, перечисленные в разделе 1.03.

4.09 DRIVPARM – подмена параметров дисковода

Команда DRIVPARM (DRIVe PARaMeters = параметры дисковода) служит для обеспечения доступа к носителям записи в устройствах, которые система BIOS компьютера неспособна правильно идентифицировать. Фактически здесь имеются ввиду устройства, которые уже были "известны" для MS-DOS7 в момент ее выхода в свет в 1995-м году, но не были "известны" системам BIOS старых компьютеров выпуска середины 1980-х годов. Вот пример использования команды DRIVPARM

для обеспечения доступа к дисководу с дискетами диаметром 3,5 дюйма в старом компьютере, у которого система BIOS "знает" только дисководы для дискет диаметром 5,25 дюйма:

```
DRIVPARM /D:1 /c /f:7 /h:2 /i /s:18 /t:80
```

здесь:

/D:1 – номер физического дисковода, "1" означает флоппи-дисковод B:, "0" означает флоппи-дисковод A:, "2" – первый дисковод на жестких магнитных дисках:, и т.д.

/c – необязательный параметр, разрешающий поддержку обнаружения смены носителей записи (дискет). Для накопителей с несменным носителем вместо параметра /c следует указывать параметр /n .

/f:7 – задает тип дисковода, причем цифра здесь означает:

- 0 – дисковод 5,25-дюймовых дискет 160/180/320/360 кбайт;
- 1 – дисковод 5,25-дюймовых дискет на 1.2 Мбайт;
- 2 – дисковод для 3,5-дюймовых дискет на 720 кбайт;
- 5 – дисковод на жестких магнитных дисках;
- 6 – накопитель на магнитной ленте (стример);
- 7 – дисковод для 3,5-дюймовых дискет на 1.44 Мбайт;
- 8 – оптический дисковод;
- 9 – дисковод для 3,5-дюймовых дискет на 2.88 Мбайт.

/h:2 – задает число головок дисковода; по умолчанию принимаются две головки для двухсторонних дискет.

/i – обеспечивает поддержку 3.5-дюймовых дисководов в компьютерах, у которых BIOS эти дисководы не поддерживает.

/s:18 – задает число секторов на дорожке:

- 8 – для старых 5.25-дюймовых дискет на 160/320 кбайт;
- 9 – для дискет на 360 и 720 кбайт;
- 15 – для старых 5.25-дюймовых дискет на 1.2 Мбайт;
- 18 – для 3.5-дюймовых дискет на 1.44 Мбайт.

/t:80 – задает число дорожек на одной стороне дискеты:

- 40 – для старых дискет на 160/180/320/360 кбайт;
- 80 – для 3.5-дюймовых дискет на 720 кбайт и 1.44 Мбайт.

Примечание 1: установки по умолчанию для параметров /f: и /s: соответствуют дисководам для дискет диаметром 5.25 дюйма с 9 секторами на дорожке.

4.10 FCBS – число блоков управления файлами

Команда FCBS резервирует участки памяти под File Control BlocS (блоки управления файлами), по 80 байт на каждый блок. Допустимое количество таких блоков – от 1 до 255. Блоки управления файлами – это устаревшая форма спецификации на открываемые для доступа файлы, она позволяет адресовать только файлы текущего каталога и не годится для адресации на дисках с файловой системой FAT32. Современные программы осуществляют доступ к файлам не с помощью FCBS, а посредством номерных ссылок ("handles", 4.12). Тем не менее MS-DOS7 поддерживает FCBS ради обеспечения совместимости со старыми программами и сетевыми службами (INTERLNK.EXE, SHARE.EXE и т.д.).

В большинстве случаев принимаемое по умолчанию значение

```
FCBS=4
```

вполне достаточно.

Примечание 1: по умолчанию место для блоков FCB резервируется за пределами обыкновенной памяти (выше 640 кбайт), но будет резервировано ниже границы 640 кбайт, если в команде DOS (4.08) указан параметр NOAUTO, а также если адресное пространство в области UMB недостаточно или недоступно (условия доступности – в разделе 4.07). В любом случае сообщение об ошибке не выдается.

Примечание 2: задаваемое командой FCBS ограничение на количество открываемых блоков управления файлами не распространяется на "неоткрытые" блоки (A.09-5), входящие в состав PSP (A.07-1), а также используемые в ряде поисковых операций.

4.11 FCBSHIGH – число блоков управления файлами

Команда FCBSHIGH делает то же самое, что команда FCBS (4.10), но только размещает блоки управления файлами за пределами обыкновенной памяти независимо от того, указан ли параметр NOAUTO в команде DOS (4.08). Все остальные сведения из раздела 4.10 в равной мере применимы к команде FCBSHIGH.

4.12 FILES – резервирование места для таблицы файлов

Команда FILES (= файлы) резервирует место в памяти для заявляемого количества записей системной таблицы файлов SFT (A.01-4). Каждая запись определяет состояние одного из открытых для доступа объектов – файла или канала, а также его ассоциацию с присвоенной ему номерной ссылкой ("handle"),

через которую данный объект адресуется. По умолчанию принимается 60 записей в SFT. Для MS-DOS7 это количество избыточно. Обычно вполне достаточно указать

```
FILES=30
```

SFT с 30 записями займет примерно 1800 байт. Только для работы с базами данных количество записей в SFT целесообразно увеличивать до 40.

Примечание 1: по умолчанию системная таблица файлов размещается за пределами обыкновенной памяти (выше 640 кбайт), но будет размещена ниже границы 640 кбайт, если в команде DOS (4.08) указан параметр NOAUTO, а также если адресное пространство в области UMB недостаточно или недоступно (условия доступности – в разделе 4.07). В любом случае сообщение об ошибке не выдается.

4.13 FILESHIGH – резервирование места для таблицы файлов

Команда FILESHIGH делает то же самое, что команда FILES (4.12), но только размещает системную таблицу файлов за пределами обыкновенной памяти независимо от того, указан ли параметр NOAUTO в команде DOS (4.08). Все остальные сведения из раздела 4.12 в равной мере применимы к команде FILESHIGH.

4.14 INCLUDE – ввести блок конфигурационных команд

Команда INCLUDE (= содержать) включает именованный блок команд в состав исполняемой последовательности конфигурационных команд. Этот именованный блок команд в файле CONFIG.SYS должен быть указан отдельно и должен начинаться отдельной строкой с именем блока – то есть с одним словом или числом, заключенным в квадратные скобки, например [L055] (пример из раздела 9.09-01). Конец блока обозначается аналогичной строкой заголовка с именем следующего блока. Если больше нет блоков, которые должны быть исполнены обособленно, то в качестве последнего заголовка используется строка с зарезервированным именем [common]. Следующие далее команды, если они имеются, будут исполнены во всех конфигурациях загрузки DOS.

В частности, чтобы исполнить команды, входящие в состав блока [L055], в файле CONFIG.SYS в желаемой позиции должна быть указана строка

```
INCLUDE=L055
```

Обратите внимание, что имя блока команд указано справа от знака равенства без ограничивающих квадратных скобок.

Если один и тот же блок команд должен быть исполнен в каждой из нескольких конфигураций загрузки, то одинаковые строки с командой INCLUDE должны содержаться в спецификации каждой такой конфигурации. Вынесение групп повторяющихся команд в отдельные конфигурационные блоки способствует упрощению файла CONFIG.SYS и делает его структуру более понятной. Примеры такого построения файла CONFIG.SYS приведены в разделах 9.04-01, 9.09-01 и 9.11-03.

4.15 INSTALL – загрузить программу

Команда INSTALL (= установить) служит для того, чтобы загружать в память резидентные модули программ и исполняемые драйверы, которые не имеют специального драйверного заголовка (A.05-1) и потому не могут быть загружены командами DEVICE (4.06) и DEVICEHIGH (4.07). Обычно исполняемые драйверы имеют суффиксы *.COM или *.EXE. Запускаемые командой INSTALL программы и драйверы должны удовлетворять следующим условиям:

– они не должны нуждаться в пространстве окружения;
– они не должны вызывать запросы на отработку критических ошибок;
– они не должны содержать обращений к резидентному модулю командного интерпретатора COMMAND.COM, который в этот момент еще не загружен.

Перечисленным условиям удовлетворяют многие программы и драйверы, которые также могут быть загружены из файла AUTOEXEC.BAT и прямо из командной строки, но загрузка командой INSTALL требует меньше памяти и считается более надежной.

Строки с командой INSTALL интерпретируются после строк с командами DEVICE, DEVICEHIGH и SET, но перед строкой с командой SHELL, даже если в файле CONFIG.SYS последовательность строк иная. Во избежание путаницы желательно размещать команды в строках файла CONFIG.SYS в порядке их исполнения.

Вот пример загрузки резидентной программы командой INSTALL:

```
INSTALL=\DOS\DRV\Mkecdex.com /B /L:0
```

Здесь справа от знака равенства записаны путь к программе, ее имя и группа передаваемых ей параметров. Так же, как командам DEVICE и DEVICEHIGH, команде INSTALL можно приписать вопросительный знак (INSTALL?=...), что вызовет запрос к пользователю ([Enter=Y, Esc=N]?) для подтверждения целесообразности исполнения данной строки.

Команду INSTALL можно использовать для временной загрузки резидентных модулей, которые должны быть выгружены после исполнения своей миссии.

Особенность таких операций в том, что высвобождение памяти, занятой уже загруженным модулем, DOS осуществляет только в пределах обыкновенной памяти, то есть до границы 640 кбайт. Когда в команде DOS (4.08) указан параметр NOAUTO, загрузка командой INSTALL будет производиться в обыкновенную память, и тогда можно временно загрузить, например, командный интерпретатор, чтобы вызвать паузу и дать возможность прочитать выведенные на экран сообщения:

```
INSTALL=C:\Command.com /c pause
```

Еще один пример использования команды INSTALL для временной загрузки резидентного модуля приведен в разделе 9.09-01.

Примечание 1: команда INSTALL не принимает участия в процедуре оптимизации распределения памяти, организуемой программой MEMMAKER.EXE и не принимает дополнительные параметры (/L и /S, 4.07), влияющие на размещение загружаемых модулей.

Примечание 2: по умолчанию загружаемые командой INSTALL программы и драйверы размещаются за пределами обыкновенной памяти (выше 640 кбайт), но будут размещены ниже границы 640 кбайт, если в команде DOS (4.08) указан параметр NOAUTO, а также если адресное пространство в области UMB недостаточно или недоступно (условия доступности – в разделе 4.07). В любом случае сообщение об ошибке не выдается.

4.16 INSTALLHIGH – загрузить программу

Команда INSTALLHIGH делает то же самое, что команда INSTALL (4.15), но только загружает программы и драйверы за пределы обыкновенной памяти независимо от того, указан ли параметр NOAUTO в команде DOS (4.08).Команду INSTALLHIGH не следует использовать для загрузки резидентных модулей, которые должны быть выгружены после исполнения своей миссии. Все остальные сведения из раздела 4.15 в равной мере применимы к команде INSTALLHIGH.

4.17 LASTDRIVE – последний дисковод

Команда LASTDRIVE (= последний дисковод) определяет адресное пространство, выделяемое для системной таблицы CDS (A.03-03). Записи в таблице CDS хранят имена текущих каталогов для всех логических дисков – как реальных, так и виртуальных. В начальный момент загрузки MS-DOS7 создает в таблице CDS по одной записи на каждый логический диск, опознанный системой BIOS компьютера, и потом заполняет оставшееся пространство недействительными записями (резервациями) до тех пор, пока не будет зарезервирована запись для

последней буквы диска, указанной в команде LASTDRIVE. По умолчанию принимается

```
LASTDRIVE=Z
```

Для размещения такой таблицы CDS потребуется 2288 байт. Если Вы считаете этот объем таблицы CDS избыточным, то можно указать другую последнюю букву диска, но в любом случае места там должно быть достаточно для всех дисков, включая те, которые могут быть задействованы позже посредством установки дополнительных драйверов (для CD/DVD-ROM, сетевых служб и т.п.).

Примечание 1: по умолчанию MS-DOS7 размещает таблицу CDS за пределами обыкновенной памяти (выше 640 кбайт), но будет размещать ее ниже границы 640 кбайт, если в команде DOS (4.08) указан параметр NOAUTO, а также если адресное пространство в области UMB недостаточно или недоступно (условия доступности – в разделе 4.07). В любом случае сообщение об ошибке не выдается.

4.18 LASTDRIVEHIGH – последний дисковод

Команда LASTDRIVEHIGH делает то же самое, что команда LASTDRIVE (4.17), но только определяет создание таблицы CDS за пределами обыкновенной памяти независимо от того, указан ли параметр NOAUTO в команде DOS (4.08). Все остальные сведения из раздела 4.17 в равной мере применимы к команде LASTDRIVEHIGH.

4.19 MENUCOLOR – цвет меню

Команда MENUCOLOR может быть применена только в составе конфигурационных меню или субменю, то есть в таких блоках команд, которые либо имеют заголовок [menu], либо объявлены как субменю в вышестоящем меню или субменю. Каждая команда MENUCOLOR задает цвета текста и фона при отображении того меню (или субменю), в блок команд которого она включена, например, так:

```
MENUCOLOR=7,0
```

Здесь первая цифра означает код цвета текста (7 – белый), а следующая после запятой цифра означает код цвета фона (0 – черный). Именно такие значения кода цвета принимаются по умолчанию, то есть когда команда MENUCOLOR в блоке меню отсутствует. Другие допустимые значения кодов цвета приведены в приложении A.10-5.

4.20 MENUDEFAULT – выбираемый по умолчанию пункт меню

Команда MENUDEFAULT может быть применена только в составе конфигурационных меню или субменю, то есть в таких блоках команд, которые либо имеют заголовок [menu], либо объявлены как субменю в вышестоящем меню или субменю. Обычно команда MENUDEFAULT помещается после перечня пунктов меню и определяет тот пункт, который должен быть выбран тогда, когда пользователь не осуществил свой выбор в течение заданного интервала времени, например

```
MENUDEFAULT=L007,20
```

Здесь справа от знака равенства указано имя блока конфигурационных команд [L007], исполнение которого следует начать по умолчанию. Затем после запятой указан интервал времени – 20 секунд, в течение которого загрузчик IO.SYS должен ожидать выбора пункта меню пользователем, прежде чем запустить на исполнение вариант, заданный командой MENUDEFAULT. Допускаются задержки от 1 до 99 секунд. Примеры конфигурационных меню с использованием команды MENUDEFAULT приведены в разделах 9.04-01, 9.09-01 и 9.11-03.

4.21 MENUITEM – пункт меню

Команда MENUITEM может быть применена только в составе конфигурационных меню или субменю, то есть в таких блоках команд, которые либо имеют заголовок [menu], либо объявлены как субменю в вышестоящем меню или субменю. Для каждого пункта меню командой MENUITEM должно быть объявлено имя соответствующего блока конфигурационных команд, а также задан текст названия пункта, воспроизводимый на экране в составе меню, например

```
MENUITEM=L007, Relocate DOS to 5600 kb RAM-disk R:
```

Здесь "L007" – пример имени блока конфигурационных команд, который должен быть в файле CONFIG.SYS и должен начинаться с отдельной строки заголовка [L007] (2.03-05, пример в разделе 9.09-01). Далее в строке с командой MENUITEM, справа от запятой приведен текст названия данного пункта меню. Этот текст может содержать слова, разделяемые пробелами, но не должен содержать квадратных скобок [ ], знаков точки с запятой (;), а также знаков косой черты, как прямых (/), так и обратных (\ ).

Во время интерпретации выбора пункта меню загрузчик IO.SYS создает переменную окружения CONFIG и присваивает ей в качестве значения имя блока конфигурационных команд, соответствующего выбранному пункту меню (в приведенном выше примере это L007). Позже, во время интерпретации строк файла AUTOEXEC.BAT или любого другого batch-файла, значение переменной CONFIG

может быть использовано для адаптации исполнения в соответствии с выбранным вариантом конфигурации.

Примечание 1: суммарное количество команд MENUITEM и SUBMENU (4.29) в любом блоке меню не должно превышать 9.

4.22 MULTITRACK – многодорожечный доступ

Для доступа к любому диску необходимо указать начальный сектор и число секторов, которые надлежит прочитать или записать. Старые версии BIOS в компьютерах выпуска 1980-х годов обслуживали такие запросы на доступ только в пределах одной дорожки диска. Требовалось, чтобы сумма номера начального сектора и числа секторов не превосходила бы полного числа секторов на дорожке, иначе процесс записи или считывания "заворачивался" обратно на начало той же дорожки. На таких старых компьютерах нельзя пользоваться операциями многодорожечного доступа, и для того в файле CONFIG.SYS должна быть строка

```
MULTITRACK OFF
```

Все современные системы BIOS поддерживают операции многодорожечного доступа, то есть автоматически переводят процесс на следующую дорожку по завершении записи или считывания предыдущей дорожки. MS-DOS7 пользуется многодорожечным доступом, фактически принимая команду MULTITRACK ON по умолчанию. По этой причине сейчас команда MULTITRACK в файле CONFIG.SYS обычно не упоминается.

4.23 NUMLOCK – режим цифровой части клавиатуры

Команда NUMLOCK (NUMerical keypad LOCK = запирание клавиш цифровой группы) определяет состояние функционирования группы клавиш цифрового набора в правой части клавиатуры. Обычно в современных компьютерах принимается по умолчанию выключенное состояние, при котором эти клавиши дублируют роли клавиш основной части клавиатуры (стрелочек, PgUp – PgDn и др.). Если важно избежать иного состояния, то надо ввести команду

```
NUMLOCK OFF
```

Для перехода к набору цифр и знаков арифметических операций необходимо изменить состояние на противоположное командой

```
NUMLOCK ON
```

В обоих состояниях переключателя NUMLOCK клавишами цифровой группы можно выбирать пункт конфигурационного меню в ходе загрузки MS-DOS7, но при выключенном состоянии они действуют как стрелочки вверх и вниз, а при

включенном позволяют выбирать пункт по номеру. Если способ выбора пункта меню небезразличен, то команду NUMLOCK можно указать в составе блока [menu] файла CONFIG.SYS.

О влиянии переключателя NUMLOCK на коды клавиш, возвращаемые обработчиком прерывания INT 16\AX=10h, написано в примечании 6 к таблице A.02-1.

4.24 REM – вставка комментария

Загрузчик IO.SYS воспринимает REM (REMark = замечание) как команду игнорировать все остальные слова в той же строке до конца данной строки. Основная миссия команды REM – обеспечить возможность введения в файл CONFIG.SYS строк комментариев, которые не нужно воспроизводить на экране. Команду REM иногда вставляют в начало строки специально для того, чтобы временно предотвратить исполнение записанной в данной строке другой команды.

Примечание 1: загрузчик IO.SYS при интерпретации строк файла CONFIG.SYS не предоставляет тех дополнительных возможностей, которые обусловливают особенности исполнения одноименной команды REM (3.24) командным интерпретатором COMMAND.COM.

4.25 SET – присвоение значения переменной

Команда SET (= установить) в строке файла CONFIG.SYS позволяет присваивать и переопределять значение переменной окружения, например:

```
SET  Var_Name=New_Var_Value
```

здесь:

Var_name – пример имени переменной окружения. Оно может содержать цифры, но должно начинаться с буквы.

New_Var_Value – пример строкового значения, присваиваемого указанной переменной. Значение не должно содержать знаков равенства. Если в строке имеются пробелы, в том числе предшествующие или следующие до конца строки, то все эти пробелы будут включены в состав значения переменной окружения.

Команда SET, интерпретируемая загрузчиком IO.SYS в строках файла CONFIG.SYS, действует не совсем так, как одноименная команда SET, исполняемая командным интерпретатором COMMAND.COM (3.25). Особенности исполнения команды SET загрузчиком IO.SYS состоят в следующем:

– если в строке файла CONFIG.SYS после команды SET не указано имя переменной, то исполнение этой команды не вызывает воспроизведения на экране перечня действующих переменных окружения.
– приписывание вопросительного знака справа (SET?=...) вызывает останов исполнения с запросом [Enter=Y, Esc=N]?. Дальнейшее исполнение или неисполнение этой команды зависит от ответа пользователя.
– все команды SET в файле CONFIG.SYS исполняются после команд DEVICE и DEVICEHIGH, но перед исполнением команд INSTALL, NSTALLHIGH и SHELL. Именно такого порядка расположения команд желательно придерживаться при составлении файла CONFIG.SYS.
– подстановки значений (2.03-03) и перенаправления (2.04-02 – 2.04-05) в строках файла CONFIG.SYS не исполняются. Вместо того соответствующие знаки вводятся в значение переменной окружения. Иногда этим намеренно пользуются для введения в значение переменной знаков процента и перенаправлений.

4.26 SHELL – загрузка командного интерпретатора

Команда SHELL (= оболочка) служит для передачи управления исполняемому файлу, который уже не вернет управление обратно загрузчику IO.SYS. Поэтому команда SHELL исполняется последней в файле CONFIG.SYS и обычно записывается в его последней строке. Программы, которым передается управление посредством команды SHELL, могут быть загрузчиками других операционных систем (например, LOADLIN.EXE для системы LINUX) или командными интерпретаторами. Вот пример строки с передачей управления командному интерпретатору NDOS.COM:

```
SHELL=C:\DOS\NU\Ndos.com /f @C:\DOS\NU\Ndos.ini
```

здесь:

C:\DOS\NU\ – пример пути к командному интерпретатору;
/f @C:\DOS\NU\Ndos.ini – пример группы параметров, передаваемых загружаемому командному интерпретатору.

Когда строки с командой SHELL в файле CONFIG.SYS нет, загрузчик IO.SYS пытается найти собственный командный интерпретатор MS-DOS7 – файл COMMAND.COM – в корневом каталоге текущего диска и передать управление ему. В таком случае COMMAND.COM будет запущен на исполнение с параметрами, принимаемыми по умолчанию. Тем не менее лучше указывать параметры запуска явно, например, так:

```
SHELL=COMMAND.COM A:\ /e:1008 /p
```

В этом примере отсутствие пути перед именем файла COMMAND.COM означает, что его следует искать в текущем каталоге. Назначение параметров командного интерпретатора COMMAND.COM подробно рассмотрено в разделе 6-04. Другие примеры передачи управления командному интерпретатору с помощью команды SHELL приведены в разделах 9.01-01, 9.04-01 и 9.09-01.

4.27 STACKS – спецификация дополнительных стеков

Команда STACKS (= стеки) в файле CONFIG.SYS резервирует место в памяти для дополнительных стеков DOS, используемых при обработке вложенных прерываний. Параметры команды определяют количество дополнительных стеков и пространство, выделяемое каждому стеку. Принимаемые по умолчанию значения эквивалентны команде

```
STACKS=9,256
```

здесь:

9 – количество дополнительных стеков (допустимо от 8 до 64 и 0);
256 – длина каждого стека в байтах (допустимо от 32 до 512 и 0).

Переполнение стека – неприятное событие, оно влечет за собой необходимость перезагрузки компьютера и может быть причиной потери части данных. Чтобы наверняка избежать переполнения, длину стеков не рекомендуется устанавливать меньше принимаемой по умолчанию величины.

Примечание 1: по умолчанию MS-DOS7 размещает дополнительные стеки за пределами обыкновенной памяти (выше 640 кбайт), но разместит их ниже границы 640 кбайт, если в команде DOS (4.08) указан параметр NOAUTO, а также если адресное пространство в области UMB недостаточно или недоступно (условия доступности – в разделе 4.07). В любом случае сообщение об ошибке не выдается.

4.28 STACKSHIGH – спецификация дополнительных стеков

Команда STACKSHIGH делает то же самое, что команда STACKS (4.27), но только определяет создание стеков за пределами обыкновенной памяти независимо от того, указан ли параметр NOAUTO в команде DOS (4.08). Все остальные сведения из раздела 4.27 в равной мере применимы к команде STACKSHIGH.

4.29 SUBMENU – объявление субменю

Команда SUBMENU (= субменю) применяется наравне с командой MENUITEM (4.21) только в составе конфигурационных меню или субменю, то есть в таких блоках команд, которые либо имеют заголовок [menu], либо объявлены как субменю в вышестоящем меню или субменю. Команда SUBMENU объявляет имя блока конфигурационных команд, придает этому блоку статус меню, а также задает текст названия субменю, воспроизводимый на экране в составе меню, например:

```
SUBMENU=6000, Relocation to RAM-disk
```

Здесь "6000" – имя блока [6000] конфигурационных команд, которому будет присвоен статус субменю, а справа от запятой – текст названия субменю. На этот текст налагаются те же ограничения, что и в команде MENUITEM (4.21). Блок конфигурационных команд, которому присваивается статус субменю, должен иметь такую же структуру, что и основной блок меню. Он может включать до 9 позиций выбора, каждая из которых представлена отдельной строкой с командой MENUITEM или SUBMENU. Отличие блока субменю состоит только в том, что его произвольное уникальное имя не должно совпадать как с зарезервированными именами ([menu] и [common]), так и с именами других конфигурационных блоков.

4.30 SWITCHES – дополнительные параметры

Команда SWITCHES (= переключатели) позволяет задать четыре необязательные установки режима работы DOS:

```
SWITCHES= /K /N /F /E:64
```

здесь:

- /K – позволяет старым программам, разработанным в расчете на 86-клавишную клавиатуру, работать на компьютерах с усовершенствованной 101/108-клавишной клавиатурой.
- /N – блокирует клавиши F5 и F6 на время загрузки DOS, предотвращая тем самым возможность загрузки без исполнения команд в файлах CONFIG.SYS и AUTOEXEC.BAT.
- /F – исключает двухсекундную задержку после воспроизведения на экране сообщения "Starting WINDOWS...".
- /E:64 – выделяет 64 байта обыкновенной памяти (допускается от 48 до 1024 байт) для EBIOS – обработчика обращений к дискам с адресацией LBA (примечание 4 к A.13-6). Если цифра после параметра /E не указана, то весь код EBIOS будет размещен в обыкновенной памяти. Параметр /E нужен старым компьютерам, у которых система BIOS обеспечивает только адресацию CHS.

Глава 5 Избранные драйверы для MS-DOS7

Драйверы – это файлы, содержащие исполняемый резидентный машинный код. Резидентным называется код, написанный специально для того, чтобы его загрузили в оперативную память компьютера и оставили там ждать момента, когда он будет запрошен. Когда запрос поступает, код драйвера принимает управление на себя, исполняет свою миссию, и потом снова "ждет" очередного запроса. По тому же принципу действуют элементы ядра любой операционной системы. MS-DOS сочетает ограниченное число функций своего ядра с разнообразными функциональными расширениями от драйверов. Правильный выбор и своевременное обновление драйверов обеспечивают адаптацию и выживание DOS в условиях постоянного совершенствования аппаратной части компьютеров.

Драйверы могут представлять собой файлы со специальным драйверным заголовком (A.05-1), как правило, имеющие суффикс *.SYS, или обычные исполняемые файлы с резидентной частью, как правило, имеющие суффикс *.COM или *.EXE. Драйверы с суффиксом *.SYS необходимо загружать командой DEVICE (4.06) или DEVICEHIGH (4.07) из строк конфигурационного файла CONFIG.SYS. Примеры составления файла CONFIG.SYS приведены в разделах 9.01-01, 9.04-01, 9.09-01. Загрузка командами DEVICE и DEVICEHIGH дает больше возможностей повлиять на построение системных структур DOS, потому что процесс их построения к тому моменту еще не завершен.

Драйверы с суффиксами *.COM и *.EXE обычно загружаются позже либо из файла CONFIG.SYS командами INSTALL (4.15) или INSTALLHIGH (4.16), либо из файла AUTOEXEC.BAT (9.01-02, 9.04-02, 9.09-02), либо из командной строки – непосредственно или с помощью команды LH (3.17). Загрузка из файла CONFIG.SYS в меньшей степени подвержена влиянию других программ и потому считается более безопасной. С другой стороны, команды INSTALL и INSTALLHIGH не вовлекаются в процесс оптимизации распределения памяти, организуемый программой MEMMAKER.EXE. Если последнее существенно, то следует предпочесть загрузку командой LH (3.17) из файла AUTOEXEC.BAT.

Базовую группу драйверов для MS-DOS7 составляют те, которые содержатся в комплектах поставки операционных систем WINDOWS-95/98 и на загрузочных дискетах для них. При стандартной установке этих операционных систем драйвера для MS-DOS7 находятся в каталогах \WINDOWS и \WINDOWS\COMMAND. Но если MS-DOS7 используется как независимая операционная система, то относящиеся к ней драйверы лучше скопировать в отдельный каталог, например, в C:\DOS\DRV. Такой путь указан в большинстве приведенных в этой главе примеров. Когда дело дойдет до практического следования этим примерам, тогда

будет важно, чтобы указываемый путь соответствовал фактическому месту размещения каждого драйвера конкретно на диске Вашего компьютера.

Помимо "родных" драйверов фирмы Microsoft, имеется много драйверов для MS-DOS7, созданных после 1998 года заинтересованными разработчиками и поставщиками компьютерного оборудования. Из этого многообразия только немногие драйверы описаны в данной главе. Сюда не вошли некоторые драйверы, описанные в файле MSDOSDRV.TXT из комплектов поставки систем WINDOWS-95/98, а также драйверы для мало распространенного оборудования – магнитооптических дисководов, дисководов марок ZIP и LS120, стримеров и т.п. Предпочтение отдано тем драйверам, которые особенно нужны при конфигурировании загрузки MS-DOS7 для выполнения восстановительных и настроечных работ.

5.01 Системные службы DOS

5.01-01 Файл ядра IO.SYS и файл параметров MSDOS.SYS

В корневом каталоге диска, с которого производится загрузка систем MS-DOS7 или Windows-95/98, имеются два скрытых системных файла: IO.SYS и MSDOS.SYS. Они находятся там с момента установки операционной системы на данный диск, чтобы обеспечивать процесс ее загрузки. Файл IO.SYS содержит ядро MS-DOS7 в сочетании с интерпретатором и загрузчиком DOS, а файл MSDOS.SYS – это перечень параметров загрузки. Если у Вас этих файлов нет, то их можно считать из комплекта поставки Windows-95/98 на компакт-диске или скачать из сети, например, с сайта http://www.micosyen.com/msdos.php в составе архива dos7.zip.

Чтобы диск или дискета стали загрузочными, наличие упомянутых системных файлов в корневом каталоге необходимо, но не достаточно. Исполняемый код boot-сектора диска не будет "знать", кому передать управление продолжением процесса загрузки, если имя файла-загрузчика не "прописано" в boot-секторе. Поэтому копирование группы системных файлов в корневой каталог, как правило, выполняют согласованно с обновлением boot-сектора с помощью программы SYS.COM (6.24).

Получив управление в свои "руки", загрузчик DOS считывает из файла MSDOS.SYS параметры, определяющие конфигурацию загрузки. От них зависят предоставляемые пользователю альтернативы, о которых написано в разделе 1.02, а также то, какая операционная система будет загружена – MS-DOS7 или Windows-95/98. Если программой ATTRIB.EXE (6.01) снять с файла MSDOS.SYS атрибуты HRS (H = скрытый, S = системный, R = только для чтения), то его можно

корректировать как обычный неформатированный текстовый файл, например, с помощью текстового редактора EDIT.COM (6.09).

Некоторые или даже все параметры могут быть не представлены в файле MSDOS.SYS, и тогда им будут приписаны значения, принимаемые по умолчанию. В отличие от других системных файлов, файл MSDOS.SYS не копируется программой SYS.COM на новый носитель, а создается заново пустым, и это не мешает операционной системе WINDOWS-95 нормально загружаться. Однако по крайней мере некоторые из принимаемых по умолчанию значений не подойдут, если Вы захотите загружать MS-DOS7 как отдельную операционную систему. Подходящие значения всех параметров представлены в образце файла MSDOS.SYS, который приведен ниже.

```
[Paths]
WinDir=C:\WINDOWS
; базовый путь для переменных окружения TMP, TEMP и PATH
WinBootDir=C:\WINDOWS
; задание значения переменной окружения WINBOOTDIR
HostWinBootDrv=C
; указание на диск, с которого компьютер был загружен
[Options]
Logo=0
; скрыть загрузочные сообщения под картинкой (= 1) или нет (= 0)
BootMenu=0
; выводить загрузочное меню системы Windows (= 1) или нет (= 0)
BootMenuDelay=20
; задержка в секундах до выбора пункта меню по умолчанию
BootMenuDefault=1
; номер пункта меню, который должен быть выбран по умолчанию
BootKeys=1
; активизировать (= 1) или нет (= 0) при загрузке "горячие клавиши"
; F5, Shift-F5, F6, F8 и Shift-F8, описанные в разделе 1.02
BootDelay=2
; интервал ожидания нажатия "горячих клавиш" (в секундах)
BootMulti=0
; не активизировать клавишу F4 (1.02) загрузки прежней версии DOS
BootWin=1
; загружать MS-DOS7 и Windows (= 1) или прежнюю версию DOS (= 0)
BootSafe=0
; загружать Windows в обычном (= 0) или в безопасном режиме (= 1)
BootWarn=0
; не давать предупреждения о загрузке в безопасном режиме
BootGUI=0
; загружать графическую оболочку Windows (= 1) или MS-DOS7 (= 0)
```

```
LoadTop=0
; загружать Command.com и Dblspace.bin ниже границы 640 кбайт
AutoScan=0
; условия автоматического запуска программы Scandisk.exe:
;        - никогда не запускать автоматически (= 0),
;        - запускать после каждого сбойного завершения (= 1)
;        - запускать при каждой загрузке компьютера (= 2)
DBLSpace=0
DRVSpace=0
; не загружать драйверы записи на диск со сжатием "на лету", но если
;  загружать, то значение = 1 должно быть только для одного драйвера.
Network=0
; загружать (= 1) или не загружать (= 0) поддержку сетевых служб
DoubleBuffer=0
; не загружать по умолчанию драйвер Dblbuff.sys (5.06-02)
DisableLog=1
; не вести отчет об этапах загрузки WINDOWS в файле Bootlog.txt.
```

Строки файла MSDOS.SYS считываются интерпретатором, входящим в состав того же файла IO.SYS. Интерпретатор пропускает строки, начинающиеся со знака точки с запятой, и потому они служат для введения комментариев. Конечно, строки комментариев можно из файла исключить, но имеется один аргумент против: для сохранения совместимости с устаревшими антивирусными программами длина скрытых системных файлов не должна быть меньше 1024 байт. Для многих современных антивирусных программ это ограничение не существенно.

Все значения параметров в показанном образце файла MSDOS.SYS совместимы с вариантами конфигурационных файлов, приведенными в разделах 9.01, 9.04, 9.09 и 9.11. Установки в секции [PATHS] используются только операционной системой WINDOWS; для MS-DOS7 соответствующие значения переменных окружения все равно приходится переопределять при интерпретации файла AUTOEXEC.BAT. Часть параметров в секции [Options] тоже можно опустить. Однако приведенный полный перечень параметров поможет Вам принять свое решение, надо ли указывать каждый конкретный параметр и какое значение ему следует приписать. Подготовив свою версию файла MSDOS.SYS, не забудьте поместить ее в корневой каталог загрузочного диска и вернуть ей ее "родные" атрибуты HRS (Hidden, Read-only, System).

В соответствии с подготовленными параметрами из того же файла IO.SYS происходит загрузка ядра MS-DOS7, обслуживающего системные структуры DOS и вызовы базовых функций прерывания INT 21 (они описаны в разделе 8.02). Последняя задача загрузчика состоит в интерпретации команд конфигурационного файла CONFIG.SYS, согласно которым производится загрузка большинства

используемых драйверов. Варианты файла CONFIG.SYS показаны в разделах 9.01-01, 9.04-01 и 9.09-01. При дальнейшем функционировании MS-DOS7 никогда не вызывает файл IO.SYS на исполнение, но его наличие тем не менее требуется для копирования каждый раз, когда нужно сделать загрузочным еще один диск.

Примечание 1: в прежних версиях MS-DOS файл MSDOS.SYS не был текстовым: он содержал ядро DOS и загружался по умолчанию как драйвер.

Примечание 2: в 2001 году была обнаружена ошибка в ядре MS-DOS7: оно неверно реагировало на сбои магнитных дисков с LBA-адресацией. Фирма Microsoft исправила файл ядра IO.SYS и поставляет его в составе SFX-архива 311561usa8.exe, который можно загрузить с сайта http://support.microsoft.com/kb/311561/en-us?spid=6519&sid=global . Программа WINRAR версии 3.2 (и выше) распаковывает файл 311561usa8.exe как CAB-архив. В нем под прозвищами Winboot.98s и Winboot.98g спрятаны два варианта файла IO.SYS. Если команда VER (3.32) сообщает версию 4.10.2222, тогда файл IO.SYS нужно получить переименованием Winboot.98s. Если же сообщаемая версия – 4.10.1998, то надо будет переименовать Winboot.98g.

5.01-02 Подмена номера версии DOS: драйвер SETVER.EXE

Эволюция операционных систем осложняется проблемой обеспечения пользования программами, разработанными для их предыдущих версий. Фирма Microsoft решает эту проблему путем подмены реального номера версии MS-DOS требуемым старым номером версии в ответах, которые операционная система дает на запросы программ через прерывание INT 21\AH=30h (8.02-22). Причем такая подмена должна осуществляться избирательно, только для тех программ, совместимость которых точно установлена. SETVER.EXE и есть как раз тот самый драйвер, на который возложена миссия обманывать совместимые программы в отношении действительного номера версии MS-DOS.

При стандартной установке операционных систем Windows-95/98 драйвер SETVER.EXE находится в каталоге \WINDOWS. Если предполагается организовать альтернативную загрузку Windows-95/98 и MS-DOS7, то бывает удобнее в каталоге с драйверами для DOS иметь отдельную копию драйвера SETVER.EXE. Ее надо загружать из файла CONFIG.SYS командой DEVICE (4.06) или DEVICEHIGH (4.07):

```
DEVICEHIGH=C:\DOS\DRV\SETVER.EXE
```

здесь:

C:\DOS\DRV\ – пример пути к файлу SETVER.EXE в отдельном каталоге для драйверов DOS.

"Подставной" номер версии DOS будет сообщен в ответе на запрос программы только в том случае, если имя этой программы вместе с должным номером версии заранее внесены в таблицу, содержащуюся в загруженном резидентном модуле драйвера SETVER.EXE. Хотя никаких гарантий такая подстановка не дает, тем не менее большинство старых программ вполне способно работать в MS-DOS7.

Файл SETVER.EXE можно запустить на исполнение из командной строки как обычную программу, например, чтобы вывести на экран краткую справку:

```
Setver.exe /?
```

Если файл SETVER.EXE запустить на исполнение из командной строки без параметров, то он воспроизведет на экране таблицу, содержащуюся в его резидентном модуле. Изначально таблица не пуста: ее исходное содержание отражает рекомендации фирмы Microsoft. Чтобы дополнить эту таблицу именем еще одной программы, нужно в командной строке набрать, например:

```
Setver.exe QBASIC.EXE 6.22
```

здесь:

QBASIC.EXE – пример имени программы, которую нужно "обмануть". Имя программы должно иметь суффикс *.COM или *.EXE.

6.22 – пример номера версии DOS, который следует сообщить в ответе на запрос этой программы.

Команда на удаление имени той же программы из встроенной таблицы требует указания параметра /D и выглядит так:

```
Setver.exe QBASIC.EXE /D
```

По завершении операций с таблицей SETVER.EXE оставляет одно из следующих значений кода ошибки (3.15-03 и 9.07-03):

0 – успешное завершение операции
1 – ошибка в спецификации параметра
2 – ошибка в имени файла
3 – недостаточно памяти для исполнения команды
4 – неверный формат номера версии
5 – запрошенный файл в таблице не найден
8 – слишком много параметров в командной строке
9 – по крайней мере одного параметра в строке не хватает
10 – ошибка при считывании таблицы с диска
11 – таблица версий повреждена
13 – в таблице нет больше места для других файлов
14 – ошибка при записи обновленной таблицы на диск

Все операции по внесению изменений в таблицу версий заканчиваются записью на диск файла SETVER.EXE, содержащего скорректированную таблицу, но от

этого изменения не начинают действовать. Необходимо еще перенести измененную таблицу из файла в действующий резидентный модуль, загруженный в память компьютера командой DEVICE или DEVICEHIGH. Поэтому внесенные изменения вступают в силу только после перезагрузки компьютера.

5.02 Средства национальной адаптации MS-DOS7

5.02-01 Файл спецификаций COUNTRY.SYS

При стандартной установке операционных систем Windows-95/98 в каталоге \WINDOWS\COMMAND находится файл COUNTRY.SYS со спецификациями национальной адаптации MS-DOS7. Они фактически представляют собой набор таблиц, из которого данные, задаваемые кодом страны, загружаются специальной командой COUNTRY (4.05) в файле CONFIG.SYS:

```
COUNTRY=007,866,C:\DOS\DRV\COUNTRY.SYS
```

здесь:

007 – код страны (A.02-2);

866 – номер кодовой страницы (A.02-2), определяющей набор знаков;

C:\DOS\DRV\ – пример пути к файлу COUNTRY.SYS, скопированному в каталог для драйверов DOS.

Когда данные из файла COUNTRY.SYS загружены, они изменяют ряд внутренних настроек DOS, относящихся к специфичным для отдельных стран формам представления времени, дат, денежных единиц, знаков пунктуации, порядка сортировки и национальных ограничений на использование букв в именах (A.02-5). Последнее обстоятельство имеет особое значение, потому что иначе файлы, в именах которых содержатся знаки национальных алфавитов, могут оказаться недоступными.

5.02-02 Драйвер знакогенератора DISPLAY.SYS

Драйвер DISPLAY.SYS подготавливает буферные области памяти для размещения одной или нескольких национальных кодовых таблиц, которые определяют состав набора знаков и их внешний вид (A.02-2). При стандартной установке операционных систем Windows-95/98 драйвер DISPLAY.SYS находится в каталоге \WINDOWS\COMMAND. Его следует загружать из строки файла CONFIG.SYS командой DEVICE (4.06) или DEVICEHIGH (4.07):

```
DEVICE=C:\DOS\DRV\DISPLAY.SYS CON=(EGA,866,2,1)
```

здесь:

C:\DOS\DRV\ – пример пути к драйверу DISPLAY.SYS, скопированному в каталог для драйверов DOS.

CON – (console) – безальтернативная спецификация дисплея как устройства вывода.

EGA – означает, что компьютер оборудован видеоплатой, поддерживающей видеорежимы классов EGA, VGA или SVGA; альтернативы спецификации EGA таковы:

LCD– портативные компьютеры класса ноутбук с жидкокристаллическим дисплеем;

CGA – устаревшая видеоплата для цветной индикации без переключения кодовых страниц;

MONO – устаревшая видеоплата MDA для монохромной индикации, также без переключения кодовых страниц.

Если тип видеоплаты не задавать, то драйвер DISPLAY.SYS будет пытаться определить его сам, но риск ошибки при этом возрастет.

866 – номер основной кодовой страницы (A.02-2). Кодовые страницы содержатся в файлах EGA.CPI, EGA2.CPI, EGA3.CPI и ISO.CPI, по несколько кодовых страниц в каждом. Позднее программа MODE.COM (6.18) позволит выбрать нужную кодовую страницу и записать ее в буферный участок памяти, подготовленный драйвером DISPLAY.SYS.

2 – число буферных участков памяти, которые должны быть подготовлены в дополнение к тому, который выделен для указанной ранее основной кодовой страницы. Разрешено иметь до 3 дополнительных буферных участков памяти для EGA-совместимых видеоплат, до 6 – для VGA-совместимых видеоплат, только одну – для варианта LCD, и только 0 – для видеоплат CGA и MDA.

1 – число аппаратно поддерживаемых вариантов шрифта для каждой кодовой страницы. Это число можно не указывать вместе с предшествующей запятой. По умолчанию принимается 1 для варианта LCD и 2 шрифта для видеоплат EGA и VGA.

Примечание 1: параметры в скобках можно не указывать, оставив только сами скобки, тогда драйвер DISPLAY.SYS примет значения по умолчанию.

Примечание 2: резидентный модуль драйвера DISPLAY.SYS взаимодействует с другими программами через прерывания INT 2F\AX=AD00-AD03h (8.03-26, 8.03-27).

5.02-03 Переключатель кодовых страниц NLSFUNC.EXE

Драйвер NLSFUNC.EXE обеспечивает возможность оперативного переключения командой CHCP (3.04) кодовых страниц и других национальных настроек DOS. При стандартной установке операционных систем Windows-95/98 драйвер NLSFUNC.EXE находится в каталоге \WINDOWS\COMMAND. Его можно загружать непосредственно или командой LH (3.17) из командной строки, из файла AUTOEXEC.BAT, а также командами INSTALL (4.15) или INSTALLHIGH (4.16) из файла CONFIG.SYS, например:

```
INSTALLHIGH=C:\DOS\DRV\NLSFUNC.EXE C:\DOS\DRV\COUNTRY.SYS
```

здесь:

C:\DOS\DRV\ – пример пути к драйверу NLSFUNC.EXE, скопированному в каталог для драйверов DOS.

C:\DOS\DRV\COUNTRY.SYS – пример пути и имени файла, содержащего спецификации национальных настроек DOS (5.02-01).

Примечание 1: переключение между английской и другой национальной нотацией не требует переключения кодовых страниц: английская нотация имеется в каждой национальной кодовой странице (A.02-02).

Примечание 2: переключение между различными национальными кодовыми страницами также может быть выполнено программой MODE.COM (6.18-03), причем она не является резидентной программой и освобождает память по завершении исполнения.

5.02-04 Драйвер клавиатуры KEYB.COM

При стандартной установке операционных систем Windows-95/98 в каталоге \WINDOWS\COMMAND находится драйвер KEYB.COM, предназначенный для управления раскладками клавиатуры в MS-DOS7. Он может быть загружен из файла AUTOEXEC.BAT непосредственно или с помощью команды LH (3.17), а также из файла CONFIG.SYS командами INSTALL (4.15) или INSTALLHIGH (4.16), например

```
INSTALL=C:\DOS\DRV\KEYB.COM UK,850,C:\DOS\DRV\KEYBRD3.SYS /E /ID:168
```

здесь:

C:\DOS\DRV\ – пример пути к драйверу KEYB.COM, скопированному в каталог для драйверов DOS.

UK – пример двухбуквенного обозначения национальной раскладки клавиатуры (A.02-2).

850 – номер национальной кодовой страницы (A.02-2). Когда номер кодовой страницы указан именно здесь, он не будет изменяться автоматически при изменении кодовой страницы

знакогенератора командой CHCP. Если желательна синхронная смена кодовых страниц знакогенератора и клавиатуры, то здесь номер кодовой страницы указывать не нужно, но обе окружающие его запятые нужно оставить (`..UK,,C:\DOS\DRV\KEYBRD3.SYS...`).

C:\DOS\DRV\KEYBRD3.SYS – пример файла, содержащего раскладки клавиатуры, с предшествующим полным путем к нему. Каждый такой файл содержит раскладки клавиатуры для нескольких стран (A.02-2).

/E – этот параметр адаптирует раскладку к использованию "усовершенствованной" 101/108-клавишной клавиатуры.

/ID:168 – идентификатор варианта клавиатуры. Он нужен только для тех стран, в которых используются несколько вариантов раскладки клавиатуры (A.02-2). В большинстве стран используется только одна раскладка, и для них идентификатор /ID указывать не нужно.

Когда драйвер KEYB.COM загружен, он активизирует несколько дополнительных "горячих" клавишных комбинаций:

CTRL-RightSHIFT – переключение на набор знаков с номерами 128-255, специфичных для установленной национальной кодовой страницы;

CTRL-LeftSHIFT – переключение на набор знаков с номерами 032-127 – цифр, знаков пунктуации и английских букв, одинаковых на всех кодовых страницах;

CTRL-ALT-F1 – переключение на набор знаков исходной американской кодовой страницы 437;

CTRL-ALT-F2 – возврат от американской кодовой страницы 437 обратно к загруженной национальной кодовой странице;

CTRL-ALT-F7 – перевод клавиатуры в режим пишущей машинки, если такой режим поддерживается загруженной таблицей раскладки клавиатуры.

Все переключения сопровождаются коротким звуковым сигналом, избавиться от которого простыми средствами, к сожалению, невозможно.

Примечание 1: резидентный модуль драйвера KEYB.COM взаимодействует с другими программами через прерывания INT 2F\AX=AD80h-AD83h (8.03-28, 8.03-30).

5.02-05 Комбинированный драйвер KEYRUS.COM

Драйвер KEYRUS.COM, написанный Дмитрием Гуртяком из г. Донецка, – это комбинированный драйвер клавиатуры и знакогенератора. Он особенно популярен среди русских пользователей, потому что поставляется со встроенной 866-й (русской) кодовой страницей и с русской раскладкой клавиатуры. Однако в комплекте к нему приложены дополнительные программы, которые позволяют пользователю самому написать, установить и активизировать любую кодовую страницу и любую раскладку клавиатуры. Драйвер KEYRUS.COM свободно выложен на многих российских сайтах. Последняя версия 8.0b (1994 года) доступна с сайта автора http://www.gurtjak.skif.net/pages/programs.htm в составе файла архива keyrus8b.zip.

KEYRUS.COM можно загружать из файла CONFIG.SYS командами INSTALL (4.15) или INSTALLHIGH (4.16), а также из командной строки или из файла AUTOEXEC.BAT, непосредственно или командой LH (3.17), например

```
LH  C:\DOS\DRV\KEYRUS.COM
```

Если принимаемые по умолчанию установки необходимо изменить, то вслед за именем драйвера в командной строке нужно указать необязательные параметры. Таких параметров может быть много, и потому предусмотрена возможность считывания параметров из файла; произвольному имени этого файла в командной строке должен предшествовать знак "@" (эт):

```
LH  C:\DOS\DRV\KEYRUS.COM  @OPT_FILE.EXT
```

KEYRUS.COM не будет загружаться в память и не будет влиять на свой собственный уже загруженный резидентный модуль, если запускать KEYRUS.COM чтобы

– показать принимаемые по умолчанию установки: KEYRUS /?
– вывести в файлы встроенные шрифты и раскладки: KEYRUS /FILES
– изменить принимаемые по умолчанию установки.

Для изменения принимаемых по умолчанию установок после имени драйвера должна следовать группа параметров, и последним из них должен быть параметр /SAVE.

KEYRUS.COM состоит из модуля клавиатуры, знакогенераторного модуля и интерфейсного модуля. Каждый модуль принимает свою собственную группу параметров. Если не оговорено иное, во всех приведенных ниже примерах принятие драйвером KEYRUS.COM установки "ON" вместо установки "OFF" (и наоборот) ведет к инверсии получаемого результата. В частности, модуль клавиатуры принимает следующие необязательные параметры:

/KEYBOARD=Off – не загружать модуль клавиатуры, а использовать раскладку, задаваемую системой BIOS.

/BASE_KEYS – разрешить переназначение клавишей (по умолчанию оно запрещено).

/KEYS=filename.ext – загрузить раскладку клавиатуры из указанного файла. Такие файлы длиной от 212 до 318 байт каждый создаются программой KEYEDIT, поставляемой вместе с драйвером. Когда KEYRUS.COM запускается с дополнительным параметром /SAVE, раскладка клавиатуры не загружается, а вводится в состав файла KEYRUS.COM и затем становится встроенной раскладкой, принимаемой по умолчанию.

/BUFFER=ON – увеличивает длину буфера клавиатуры до 31 знака.

/FAST=ON,10,1 – задает темп работы клавиатуры (0 – самый быстрый, потом 1, 2, 4, 8, 10, 13, 16, 20, 31 – самый медленный) и задержку от 0 (0,25 с) до 3 (1 с).

/RUSALT=ON – обеспечивает набор знаков [ ] ; ' , . / при удержании нажатой клавиши ALT тогда, когда активизирована национальная раскладка клавиатуры.

/BEEP=OFF,rus– запрещает подтверждение нажатия клавиш звуковым сигналом, когда активизирована национальная раскладка клавиатуры (RUS). Вместо RUS можно указывать раскладки LAT (латинскую) или ALT (псевдографическую).

/CLICK=OFF,rus – запрещает подтверждение щелчком нажатия клавиш, когда активизирована национальная раскладка клавиатуры (RUS). Вместо RUS можно указывать раскладки LAT (латинскую) или ALT (псевдографическую).

/LAMP=ON,rus– включает световой индикатор ScrollLock, когда активизирована национальная раскладка клавиатуры (RUS). Вместо RUS можно указывать раскладки LAT (латинскую) или ALT (псевдографическую).

/COLOR=0,2 – показывает активную раскладку клавиатуры цветом рамки. Левая цифра (0 = черный) – это код цвета при активизации национальной раскладки клавиатуры (RUS), правая цифра (2 = темно-зеленый) – это код цвета при активизации псевдографической раскладки клавиатуры (ALT). Другие допустимые коды цвета приведены в приложении А.10-5.

/ALT=87,4 – установить "горячую клавишу" для переключения к псевдографической раскладке клавиатуры. Смещение 4 и скэн-код 87 соответствуют переключению по нажатию комбинации клавиш CTRL-F11 (примечания 3 и 4). /ALT=OFF означает запрещение доступа к набору знаков псевдографики.

/SCAN=54,4 – установить "горячую клавишу" для переключения к национальной раскладке клавиатуры. Смещение 4 и скэн-код 54 соответствуют переключению по нажатию комбинации клавиш CTRL-RightSHIFT (примечания 3 и 4).

/LAT=42,4 – установить "горячую клавишу" для переключения к латинской раскладке клавиатуры. Смещение 4 и скэн-код 42 соответствуют переключению по нажатию комбинации клавиш CTRL-LeftSHIFT (примечания 3 и 4). /LAT=OFF означает использование с этой целью той же "горячей клавиши", которая задействована для переключения к национальной раскладке клавиатуры.

/MODESHIFT=87,1 – установить "горячую клавишу" для временного переключения раскладок клавиатуры, пока эта клавиша удерживается нажатой. Смещение 1 и скэн-код 87 соответствуют комбинации клавиш RightSHIFT-F11 (примечания 3 и 4). Указание /MODESHIFT=OFF означает не задействовать функцию временного переключения.

/CLRSCAN=ON – вернуться к первоначальным принимаемым по умолчанию установкам "горячих клавиш".

Для реконфигурирования установок "горячих клавиш" нужно, чтобы был загружен модуль интерфейса драйвера KEYRUS.COM. Устанавливаемые "горячие клавиши" следует выбирать так, чтобы они не были бы перехвачены впоследствии при запуске файл-менеджера или других резидентных программ.

Знакогенераторный модуль драйвера KEYRUS.COM принимает следующие необязательные параметры:

/BLANK=ON,9,ON,ON – гасить экран после 9 минут бездействия, второе "ON" означает учитывать движение манипулятора "мышь", самое правое "ON" означает реагировать на выводимые на экран сообщения.

/SWITCH=22,6 – установить "горячую клавишу" для переключения модуля знакогенератора на исходную американскую кодовую страницу 437. Смещение 6 и скэн-код 22 соответствуют переключению по нажатию комбинации клавиш CTRL-LeftSHIFT-U (примечания 3 и 4). Режим модуля клавиатуры при этом не изменяется. /SWITCH=OFF означает запрет переключения модуля знакогенератора на кодовую страницу CP437.

/EGA – адаптировать знакогенераторный модуль к работе с EGA-совместимой видеоплатой. Вместо /EGA можно указать /VGA для адаптации к работе с VGA-совместимыми видеоплатами. Обе эта установки не запоминается при запуске KEYRUS.COM с параметром /SAVE.

/8x8=ON – загрузить шрифт 8x8 для текстовых видеорежимов форматов 80x43, 80x50 и т.п. Вместо ON может быть указано OFF или AUTO, последнее означает загрузку шрифта по запросам программ.

/8x14=ON – загрузить шрифт 8x14, используемый программой редактирования MS Word для DOS. Вместо ON может быть указано OFF или AUTO, как для шрифта /8x8.

/8x16=ON – загрузить шрифт 8x16 для основного видеорежима 80x25. Вместо ON может быть указано OFF или AUTO, как для шрифта /8x8.

/FULL – загрузить все 3 вида встроенных шрифтов.

/ROM – не загружать встроенные шрифты драйвера KEYRUS.COM, использовать собственный шрифт DOS.

/FONT=filename.ext – загрузить шрифт из отдельного файла. Если этот параметр используется совместно с параметром /SAVE, шрифт не загружается в память, а становится встроенным шрифтом драйвера KEYRUS.COM, загружаемым по умолчанию.

/DELETEFONT – удалить шрифт, который загружен последним.

/COMPRESS=OFF – не осуществлять сжатия шрифтов, которое разрешено только для текстовых видеорежимов DOS. Для работы в "окне DOS" операционной системы Windows и для графических видеорежимов сжатие необходимо отключать.

/ALL – загружать все знаки шрифта от 0 до 255, при этом предполагается, что сжатие не осуществляется.

/128 – загружать знаки шрифта от 128 до 255, при этом предполагается, что сжатие не осуществляется.

/RANGE=128-175,224-239 – пример указания диапазона номеров знаков шрифта, которые следует загружать. При этом предполагается, что сжатие не осуществляется.

/RUSSIAN – загружать знаки русского шрифта в указанных выше диапазонах номеров, и при этом по умолчанию применить сжатие. Если сжатие нежелательно, то следует указать параметр /COMPRESS=OFF.

Интерфейсный модуль драйвера KEYRUS.COM принимает следующие необязательные параметры:

/ANYWAY – допустить повторную загрузку драйвера KEYRUS.COM.

/DELAY_INIT – загрузить драйвер, но отложить его инициализацию до тех пор, пока он не будет запущен повторно.

/INTERFACE=OFF – дезактивировать интерфейсный модуль драйвера KEYRUS.COM. При этом KEYRUS.COM не способен к программному реконфигурированию, не может обнаружить

присутствие своих резидентных модулей в памяти, и не может выгрузить их из памяти.

/RELEASE – выгрузить KEYRUS.COM из памяти. Эта операция возможна только если интерфейсный модуль не отключен, то есть если принято /INTERFACE=ON.

Примечание 1: KEYRUS.COM несовместим с драйверами DISPLAY.SYS (5.02-02) и KEYB.COM (5.02-04), разработанными фирмой Microsoft, а также с таблицами раскладок клавиатуры KEYB*.SYS (A.02-2).. Если Вы намерены использовать драйвер KEYRUS.COM, то упомянутые драйверы фирмы Microsoft загружать не следует.

Примечание 2: KEYRUS.COM взаимодействует с другими программами через прерывание INT 2F\AX=4352h (8.03-24).

Примечание 3: смещение здесь – это сумма числа 1 для клавиши RightSHIFT, числа 2 для клавиши LeftSHIFT, числа 4 для клавиши CTRL, числа 8 для клавиши ALT, числа 16 для режима ScrollLock, числа 32 для режима NumLock, числа 64 для режима CapsLock и числа 128 для режима Insert. Эта сумма представляет собой байт, возвращаемый в регистре AL обработчиком прерывания INT 16\AH=12h (8.01-85). Например, смещение 12 означает удержание нажатыми клавиш CTRL и ALT в то время, когда Вы нажимаете основную "горячую клавишу", заданную ее скэн-кодом.

Примечание 4: KEYRUS.COM принимает из параметров десятичные скэн-коды клавиш, так что шестнадцатеричные значения из второй колонки таблицы A.02-1 необходимо сначала переводить в десятичную форму. Кроме того, KEYRUS.COM не может правильно интерпретировать скэн-коды, которые отличаются только префиксом E0h.

Примечание 5: драйвер KEYRUS.COM обслуживает "окно DOS" операционных систем Windows только в текстовых видеорежимах, то есть когда это "окно" раскрыто на весь экран. Переключение видеорежимов "окна" обычно осуществляется клавишной комбинацией ALT-ENTER. Для обслуживания "окна DOS" операционных систем Windows-2000/XP драйвер KEYRUS.COM следует загружать из файла CONFIG.NT или из файла AUTOEXEC.NT в каталоге \WINDOWS\SYSTEM32.

5.03 Драйверы манипуляторов "мышь"

Чтобы пользоваться какой-либо функцией "мыши" нужна, во-первых, такая "мышь", которая аппаратно реализует эту функцию. Во-вторых, исполнение желаемой функции должно быть поддержано драйвером "мыши". В-третьих,

функция должна быть востребована той программой, в которой Вы хотели бы данной функцией пользоваться. Наконец, в-четвертых, и драйвер, и программа должны быть работоспособны в той операционной среде, которая у Вас поставлена.

По отношению к манипуляторам типа "мышь" MS-DOS7 выглядит непоследовательной. С одной стороны, в комплекте поставки операционных систем Windows-95/98 "мышиный" драйвер для MS-DOS7 отсутствует. С другой стороны, функции "мыши" востребованы, например, текстовым редактором EDIT.COM (6.09), и "мышиные" драйверы из предыдущих версий DOS хорошо себя чувствуют в среде MS-DOS7. Все они взаимодействуют с программами одинаково – через прерывание INT 33 (8.03-31 – 8.03-55).

Как у любого периферийного прибора, у манипуляторов типа "мышь" соединение с компьютером определяется адресом порта и номером IRQ линии запроса прерывания (A.14-1). Всем "мышиным" драйверам, описываемым в этой части главы 5, подсказывать способ соединения не требуется: они способны сами искать подключенный манипулятор по обычно используемым портам и линиям IRQ. Но когда способ подключения заранее известен, его бывает полезно указать в командной строке, чтобы избежать поиска и сократить время загрузки.

С началом нового тысячелетия стали размножаться "мыши", подключаемые к шине USB. Обычно их обслуживают система BIOS компьютера через прерывания INT15\AX=C2xx и драйвер, запрашивающий эти прерывания (5.03-03). Но в устаревших компьютерах, где система BIOS на вызовы INT15\AX=C2xx не отвечает, необходимая поддержка может быть обеспечена подходящим драйвером контроллера USB (5.07-05), и тогда USB-"мышь" может быть обслужена любым из драйверов, описываемых ниже в этой части главы 5.

5.03-01 Драйвер "мыши" GMOUSE.COM

Драйвер GMOUSE.COM поставляется в комплектах программного обеспечения для распространенных манипуляторов "мышь" с торговой маркой Genius, производимых фирмой KYE System Corp. Этот драйвер также часто встречается на компакт-дисках, содержащих подборки драйверов. Версию 10.20 драйвера GMOUSE.COM (1996 года) можно свободно скачать из сети Интернет с сайта http://www.dosbootsector.narod.ru/drivers.htm в составе файла архива Gmouse.zip.

Драйвер GMOUSE.COM можно загружать из файла AUTOEXEC.BAT и из командной строки командой LH (3.17) или непосредственно, а также из файла CONFIG.SYS командами INSTALL (4.15) или INSTALLHIGH (4.16), например

```
INSTALLHIGH=C:\DOS\DRV\GMOUSE.COM
```

здесь:

C:\DOS\DRV\ – пример пути к драйверу GMOUSE.COM.

Обычно драйвер GMOUSE.COM в дополнительных спецификациях не нуждается. Тем не менее после имени драйвера в командной строке можно указать один из следующих взаимно исключающих параметров:

/1	– манипулятор подключен к порту COM1
/2	– манипулятор подключен к порту COM2
/3	– манипулятор подключен к порту COM3
/4	– манипулятор подключен к порту COM4
/P	– манипулятор подключен к порту PS/2
/P2	– 2-кнопочный манипулятор подключен к порту PS2
/P3	– 3-кнопочный манипулятор подключен к порту PS2
/U	– выгрузить драйвер GMOUSE.COM из памяти
/?	– показать встроенную справку.

Перечисленные здесь параметры подходят и для более поздних версий драйвера GMOUSE.COM, но они отличаются увеличенным объемом и наличием не очень нужных "наворотов", например, автоматическим определением языка выдачи сообщений в зависимости от установленной кодовой страницы. Драйвер версии 10.20 еще принимает дополнительные параметры из отдельного файла GMOUSE.INI, который должен находиться в том же каталоге.

Примечание 1: если во время поиска "мыши" по портам процесс будет остановлен нажатием клавиши "Break" (например, чтобы прочитать выведенные на экран сообщения), то драйвер GMOUSE.COM потом не может возобновить процесс поиска и вызывает "зависание" компьютера.

Примечание 2: драйвер GMOUSE.COM не обеспечивает совместимость с операционной системой WINDOWS, если манипулятор "мышь" подключен к портам COM3 или COM4.

5.03-02 Драйвер "мыши" MOUSE.COM

Распространенным именем MOUSE.COM названы по крайней мере несколько драйверов, написанных в разное время разными авторами. Здесь речь пойдет о популярном драйвере версии 8.20, разработанном фирмой Microsoft (длина файла 37681 байт, дата – 29.06.93). Он поставлялся в комплекте MS-DOS6, но вполне пригоден и для MS-DOS7. В сети Интернет он выложен отдельно, например, на адресе http://www.computerhope.com/download/hardware.htm#05 , и оттуда его можно свободно скачать в файле архива Mouse.zip.

Драйвер MOUSE.COM можно загрузить из файла CONFIG.SYS командами INSTALL (4.15) или INSTALLHIGH (4.16), а также из файла AUTOEXEC.BAT и из командной строки как с помощью команды LH (3.17), так и непосредственно, например:

```
C:\DOS\DRV\MOUSE.COM /C1 /R1 /S50 /P1 /N50 /Y
```

здесь:

C:\DOS\DRV\ – пример пути к драйверу MOUSE.COM

/C1 – манипулятор подключен к порту COM1. Вместо этого параметра можно указать:
- /C2 – манипулятор подключен к порту COM2
- /Z – манипулятор подключен к порту PS2
- /I1 – манипулятор подключен к порту LPT1
- /I2 – манипулятор подключен к порту LPT2
- /B – манипулятор соединен через плату расширения.

/R1 – частота опроса 30 Гц (принимается по умолчанию). Вместо этого параметра можно указать:
- /R2 – частота опроса 50 Гц,
- /R3 – частота опроса 100 Гц,
- /R4 – частота опроса 200 Гц.

/S50 – чувствительность: число после /S должно быть от 0 до 100, по умолчанию принимается 50. Вместо /S50 можно указать отдельно чувствительность по горизонтали /H50 и чувствительность по вертикали /V50.

/P1 – профиль ускорения:
- /P1 – неускоренный,
- /P2 – медленный,
- /P3 – средний,
- /P4 – быстрый.

/N50 – период корректировки положения курсора на экране, допускается в пределах от 0 до 255.

/Y – использовать аппаратные средства отображения курсора.

В большинстве случаев перечисленные здесь необязательные параметры указывать в командной строке не нужно, так как драйвер вполне удовлетворительно работает с установками параметров по умолчанию и способен последовательно искать манипулятор "мышь" на всех упомянутых выше портах.

Чтобы выгрузить драйвер MOUSE.COM из памяти, достаточно запустить его из командной строки с единственным параметром OFF:

```
C:\DOS\DRV\MOUSE.COM OFF
```

5.03-03 Драйвер "мыши" CTMOUSE.EXE

Драйвер CTMOUSE.EXE, совместимый с MS-DOS7, разрабатывается с 2002 года в рамках проекта Free DOS. Версия 2.1 этого драйвера (2007 года) выложена на сайте http://www.ibiblio.org/pub/micro/pc-stuff/freedos/files/dos/mouse/ в составе

архива Ctmous21.zip. Помимо того, драйвер CTMOUSE.EXE можно найти на сайте http://cutemouse.sourceforge.net/ .

По своей функциональности драйвер CTMOUSE.EXE близок к другим драйверам "мыши" для DOS, но отличается компактностью и поддержкой манипуляторов, снабженных колесом прокрутки. Еще одной особенностью этого драйвера является взаимодействие с BIOS через прерывания INT15\AX=C2xx, благодаря чему он сможет, вероятно, обслуживать "мышей" с интерфейсом USB. Принимаемые по умолчанию установки хорошо продуманы, так что обычно параметры командной строки оказываются драйверу CTMOUSE.EXE не нужны. Его можно загружать просто из командной строки, из файла AUTOEXEC.BAT командой LH (3.17), а также из файла CONFIG.SYS командами INSTALL (4.15) или INSTALLHIGH (4.16), например:

```
INSTALL=\DOS\DRV\CTMOUSE.EXE /S14 /3 /R33 /N
```

здесь:

\DOS\DRV\ – пример пути к драйверу CTMOUSE.EXE

/S14 – искать "мышь" только на последовательном порте, причем первая цифра обозначает номер порта (= COM1), а вторая – номер линии запроса прерывания (= IRQ 4). Если параметр /S не указывать, то драйвер будет искать "мышь" сначала на порте PS2, а потом поочередно на всех последовательных портах.

/3 – активизировать третью кнопку "мыши" (для "мышей" типа Mouse Sys tem, а также для "мышей" с колесом прокрутки вместо средней кнопки эту функцию драйвер CTMOUSE.EXE не обеспечивает).

/R33 – чувствительность: первая цифра обозначает градацию чувствительности по горизонтали, а вторая – по вертикали. Если указать только одну цифру, будет установлена одинаковая чувствительность по обеим координатам. Если параметр /R вообще не указывать, то по умолчанию будет установлено значение /R33.

/N – загрузить драйвер CTMOUSE.EXE независимо от того, был или не был загружен ранее какой-либо драйвер "мыши". Этот параметр позволяет строить batch-файлы так, чтобы после исполнения миссии драйвера CTMOUSE.EXE и его выгружения из памяти обеспечить возврат конфигурации системы к исходному состоянию.

Помимо перечисленных выше параметров, драйвер CTMOUSE.EXE может принимать следующие:

/? – показать краткую справку;

/P – искать "мышь" только на порте PS2;

/Y – не искать "мышь" типа Mouse System;
/V – сначала сканировать последовательные порты, и только потом искать "мышь" на порте PS2;
/O – не обнаруживать колесо прокрутки;
/L – адаптировать порядок кнопок для леворуких людей;
/B – не загружать CTMOUSE.EXE, если какой-либо драйвер "мыши" уже загружен;
/W – не загружать драйвер в блоки UMB;
/U – выгрузить драйвер из памяти (это возможно, только если поставленные драйвером функции не перехвачены).

5.04 Драйверы обслуживания памяти

Контроллер памяти в комплектах микросхем (чипсетах) AT-совместимых компьютеров аппаратно предопределяет использование области 640 – 1024 кбайт для размещения "теневой" памяти и "окна" доступа в видеопамять (подробнее – в примечаниях 2 и 3 к таблице A.12-1). Из-за 16-разрядной адресации и особого статуса области 640 – 1024 кбайт адресное пространство, доступное программам реального режима, оказывается существенно ограниченным. С 1980-х годов фирмы IBM и Microsoft вели разработки драйверов с целью преодоления проблемы ограниченности адресного пространства. Сейчас совершенствование драйверов памяти продолжают многие независимые поставщики системного программного обеспечения. Лучшие из найденных решений представлены ниже в разделе 5.04.

5.04-01 Драйвер расширенной памяти HIMEM.SYS

Драйвер HIMEM.SYS (версия 3.95, 1995) разработан фирмой Microsoft для обслуживания доступа к памяти за пределами границы 1024 кбайт в компьютерах с процессором не древнее 80386. Доступ обеспечивается с учетом особенностей отдельных типов процессоров в соответствии со спецификацией XMS доступа к расширенной памяти (XMS = eXtended Memory Specification). Поэтому память, обслуживаемую драйвером HIMEM.SYS, часто называют XMS-памятью. Начиная с версии 3.10 драйверу HIMEM.SYS доступно пространство до 65535 кбайт.

Помимо доступа к расширенной памяти, на драйвер HIMEM.SYS возложены еще две важных миссии: управление коммутацией линии A20 шины адреса и распределение участков расширенной памяти. Это необходимо для предотвращения конфликтов между программами как при доступе к участку верхней памяти (1024 – 1088 кбайт), так и при пользовании участками расширенной памяти. Благодаря драйверу HIMEM.SYS каждый участок расширенной памяти находится в исключительном распоряжении только той программы, которой он предоставлен.

Официальных разъяснений принципа действия драйвера HIMEM.SYS нет, но из анализа его исполняемого кода можно предположить, что он пользуется 32-разрядной линейной адресацией в реальном режиме (подробнее об этом – в разделе 9.10). Во всяком случае, реализуемый принцип действия не позволяет исполнять программный код непосредственно в XMS-памяти, а дает лишь возможность копировать код вместе с данными в XMS-память и обратно.

Драйвер HIMEM.SYS поставляется в составе WINDOWS-95/98 и при стандартной установке находится в каталоге \WINDOWS. Загружать драйвер HIMEM.SYS следует командой DEVICE (4.06) предпочтительно в первой же исполняемой строке файла CONFIG.SYS, например:

```
DEVICE=C:\DOS\DRV\HIMEM.SYS /V
```

здесь:

C:\DOS\DRV\ – пример пути к драйверу HIMEM.SYS;

/V – необязательный параметр, предписывающий показывать сведения о загрузке. Тот же результат можно получить без параметра /V, если во время загрузки драйвера держать нажатой клавишу ALT.

Значения параметров, принимаемые драйвером HIMEM.SYS по умолчанию, пригодны для весьма разнообразных вариантов сочетаний аппаратных средств в составе компьютера. Тем не менее в редких случаях приходится указывать иные значения параметров в командной строке. Принимаются следующие необязательные параметры:

/a20control:OFF – разрешает драйверу HIMEM.SYS брать на себя управление адресной линией A20 только тогда, когда эта линия не активна, предотвращая тем самым возможные сбои доступа к памяти. По умолчанию драйвер берет на себя управление линией A20 безотносительно к ее состоянию.

/cpuclock:ON – активизировать управление тактовой частотой центрального процессора, когда на эту частоту может повлиять режим доступа к памяти. Этот параметр замедляет работу драйвера HIMEM.SYS.

/eisa – разрешить доступ к расширенной памяти за пределами 16 Мбайт на компьютерах с шиной EISA.

/hmamin=40 – запретить выделение участков верхней памяти HMA (1024 – 1088 кбайт) программам, которые запрашивают менее 40 кбайт (можно указывать от 0 до 63 кбайт). По умолчанию участки верхней памяти HMA будут выделены той программе, которая их первой запросит, безотносительно к размеру запрашиваемого участка.

/int15=128 – зарезервировать 128 кбайт расширенной памяти для программ, запрашивающих доступ через прерывание INT 15\AH=87h (8.01-76). Можно указывать от 64 до 65535 кбайт, по умолчанию принимается 0.

/machine:AT – спецификация типов компьютеров, которые драйвер HIMEM.SYS не способен правильно идентифицировать. Спецификация задается либо числом от 1 до 17, либо буквенным идентификатором (таблица А.11-2). По умолчанию принимается число 1 или идентификатор AT, причем оба соответствуют компьютеру PC/AT фирмы IBM.

/noabove16 – не пользоваться прерыванием INT 15\AX=E801h для получения сведений о расширенной памяти. При этом драйвер сможет обслуживать не более 16 Мбайт.

/noeisa – запретить поиск плат расширения памяти на шине EISA.

/numhandles=32 – максимальное количество ссылок на участки расширенной памяти, которые можно одновременно держать открытыми для доступа. Допускается от 1 до 128 ссылок, по умолчанию принимается 32.

/shadowRAM:ON – не препятствовать пользованию копией кода BIOS в области памяти F000 – FFFFh. Когда драйвер HIMEM.SYS обнаруживает, что память компьютера составляет всего 2 Мбайт или меньше, он пытается высвободить больше памяти, в том числе за счет участка F000 – FFFFh.

/testmem:OFF – не проводить тестирование памяти.

/X – не пользоваться прерыванием INT 15\AX=E820h для получения сведений о расширенной памяти.

Примечание 1: драйвер HIMEM.SYS не может обеспечить доступ к памяти за пределами 16 Мбайт, если процедурой BIOS Setup установлен параметр "Memory hole 15 – 16 Mb".

Примечание 2: драйвер HIMEM.SYS принимает запросы на исполнение операций от программ посредством команд дальнего вызова (CALL FAR, 7.03-08), причем адрес вызова предоставляется через прерывание INT 2F\AX=4310h (8.03-23). Перечень операций, исполняемых по запросам программ, приведен в приложении А.12-3.

Примечание 3: известны несколько аналогов драйвера HIMEM.SYS. Дж.Р.Иллисом (J.R.Ellis) написан драйвер QHIMEM2.SYS, открывающий доступ к адресному пространству до 4 Гбайт. Его версия 3.1 (2005-го года) имеется в составе SFX-образа дискеты – файла USB18.EXE на сайте http://johnson.tmfc.net/dos/usbdrv.html . Последнюю разработку XMS-драйвера с пределом 4 Гб – XMGR.SYS – можно скачать со страницы http://johnson.tmfc.net/dos/driver.html . Еще один аналог –

HIMEM.EXE – создан в рамках проекта FreeDOS и поставляется вместе с EMM386.EXE (примечание 4 к 5.04-02). Упомянутые здесь аналоги принимают различные перечни параметров, описанные в сопровождающих файлах.

Примечание 4: в MS-DOS8 драйвер XMS-памяти включен в состав загрузчика IO.SYS, так что HIMEM.SYS для MS-DOS8 не нужен.

5.04-02 EMM386.EXE: драйвер отображаемой памяти и VCPI-сервер.

Драйверы EMM (= Expanded Memory Manager) первоначально были разработаны в 1982 – 1983 годах с целью управления доступом к дополнительной памяти на платах расширения для компьютеров IBM PC. Тогда роль драйверов EMM состояла в переключении банков памяти на платах расширения с тем, чтобы выборочно обращаться к любому из них через одно и то же "окно" адресного пространства (обычно E000 – EFFFh). Тогда же такой способ управления доступом к банкам памяти на платах расширения был регламентирован спецификацией LIM EMS. В ней, помимо прочего, оговорено разделение выделенного "окна" адресного пространства на несколько страниц по 16 килобайт, независимо направляемых на адресацию того или другого банка памяти.

Позже появились компьютеры со встроенной памятью более 1 Мегабайта, а платы расширения с дополнительными банками памяти вышли из моды. Но совместимость с программами, использующими старую адресацию LIM EMS, необходимо было сохранить, и потому принцип действия драйверов EMM был радикально изменен. Они стали управлять не платами расширения, а механизмом табличного преобразования адресов, реализованным во всех процессорах начиная с модели 80386. С тех пор такие драйверы известны под именем EMM386. Они дают возможность обращаться через те же страницы "окна" адресного пространства к участкам расширенной памяти за пределами первого мегабайта.

Уместно вспомнить, что расширенной памятью за пределами первого мегабайта "распоряжается" драйвер HIMEM.SYS (5.04-01). Именно он выделяет участки расширенной памяти в пользование программам по их запросам. Поэтому драйвер EMM386 выступает в роли посредника: получив запрос от программы на доступ к памяти согласно спецификации LIM EMS, он обращается к драйверу HIMEM.SYS и получает в пользование участок расширенной памяти, а потом изменяет адреса в таблице преобразования TLB так, чтобы обеспечить запрашивающей программе постраничный доступ к предоставленному участку расширенной памяти через "окно" адресного пространства. Когда в расширенной памяти нет свободного участка нужного размера, драйвер EMM386 способен удовлетворить запрос путем комбинирования адресации к нескольким участкам меньшего размера. Благодаря процессорному механизму преобразования адресов возможно исполнение программного кода непосредственно в расширенной памяти.

Миссию драйвера EMM386 дополнительно осложняет то, что механизм табличного преобразования адресов в процессорах действует только когда процессоры работают в защищенном режиме. Поэтому драйвер EMM386 переключает процессор в режим виртуального 8086 (V86), являющийся вариантом защищенного режима, и обеспечивает исполнение программ DOS в этом режиме на третьем (низшем) уровне привилегий. Конечно, компьютер должен иметь процессор не древнее модели 80386, способный реализовать режим V86. Программы DOS получают разрешение на проведение операций ввода-вывода на низшем уровне привилегий, так что условия доступа к дискам и к портам сохраняются такими же, как в реальном режиме. Возможности, предоставляемые режимом V86, драйвер EMM386 использует не только для отображения участков расширенной памяти в страницы "окна" адресного пространства, но также для загрузки драйверов через свободные участки области 640 – 1024 кбайт и для координации действий программ, организующих работу в защищенном режиме.

Помимо драйвера EMM386.EXE, взять управление в защищенном режиме "на себя" пытаются программы-экстендеры (например, DOS4GW.EXE), программы-серверы DPMI (например, CWSDPMI.EXE), а также многозадачные операционные системы (например, Windows). Все они не могли бы выполнить свою миссию в защищенном режиме V86 на третьем (низшем) уровне привилегий, то есть на правах обычных прикладных программ. Чтобы передача управления не вызывала конфликтов, в 1989 году был согласован протокол VCPI (Virtual Control Program Interface), разработанный фирмами Phar Lap Software и Quarterdeck Office Systems. Согласно протоколу VCPI программа, обслуживающая работу в режиме V86, должна взять на себя исполнение ряда дополнительных функций, включая переход по запросу из режима V86 в защищенный режим с нулевым (высшим) уровнем привилегий. Поскольку в MS-DOS7 роль такой программы играет драйвер EMM386.EXE, постольку его также называют VCPI-сервером. Некоторые из специфичных функций VCPI-серверов описаны в главе 8 (8.03-71 – 8.03-73).

Для загрузки драйверов в расширенную память возможность исполнения там программного кода необходима, но не достаточна. Дело в том, что обращения к драйверам производятся по фиксированным адресам, и исполняемый код драйверов не рассчитан на оперативное "перелистывание" страниц. Поэтому для загрузки драйверов и других резидентных программ драйвер EMM386 организует статичное отображение участков расширенной памяти на UMB-блоки, которые могут быть размещены во всех свободных промежутках адресного пространства от 640 до 1024 кбайт. Если драйвер EMM386 уже загружен, и затем еще раз запущен на исполнение из командной строки (без параметров), то он покажет на экране, как распределено доверенное ему адресное пространство.

В комплекте поставки операционной системы Windows-95 имеется драйвер EMM386.EXE версии 4.95 (6/12/1996, 125495 байт), который, однако, не предназначен для обслуживания автономной работы MS-DOS7. Драйвер версии

4.95 передает вызовы исключений на исполнение обработчикам защищенного режима, устанавливаемым операционной системой Windows. А когда MS-DOS7 действует отдельно от системы Windows, вызовы исключений не находят исполнителя и обычно заканчиваются "зависанием" компьютера. Поэтому для автономной работы MS-DOS7 нужны драйверы EMM386.EXE более ранних версий 4.49 или 4.50. Варианты загрузки и параметры у них такие же. Версия 4.49 (31/05/1993, 120926 байт) имеется на сервере ftp://ftp.vgt.ru/dos/ в составе образа дискеты Dos622_1.img. Его надо записать на дискету с помощью программы IMG.EXE (6.06), а там файл EMM386.EX_ можно будет распаковать с помощью программы EXPAND.EXE (6.10). Версию 4.50 драйвера EMM386.EXE (30/04/1998, 119390 байт) из комплекта PC DOS 2000 фирмы IBM можно скачать с сервера ftp://ftp.eesnet.ru/dos/ в составе SFX-архива Pcdos2k.exe.

Только после активизации драйвера EMM386.EXE становится возможной загрузка в расширенную память командами LH (3.17), DEVICEHIGH (4.07) и всеми другими с окончанием ...HIGH. Но доступ к расширенной памяти согласно спецификации XMS должен быть предоставлен заранее, до загрузки драйвера EMM386.EXE. Поэтому драйвер EMM386.EXE нужно загружать командой DEVICE (4.06) из той строки файла CONFIG.SYS, которая следует за строкой загрузки драйвера HIMEM.SYS (5.04-01) и предшествует всем строкам с командами, имеющими окончание ...HIGH. Строка загрузки драйвера EMM386.EXE может выглядеть, например, так:

```
DEVICE=C:\DOS\DRV\EMM386.EXE RAM V
```

здесь:

C:\DOS\DRV\ – пример пути к драйверу EMM386.EXE

RAM – разрешить создание EMS-страниц и UMB-блоков (другой пример использования параметра RAM показан ниже)

V – вывести сообщение о распределении EMS-страниц и UMB-блоков.

Помимо перечисленных выше параметров, в этой строке файла CONFIG.SYS также могут быть указаны другие необязательные параметры, в частности:

OFF – отложить активизацию драйвера EMM386 до выдачи команды "EMM386.EXE ON" из командной строки. Если вместо OFF указать параметр AUTO, то отложенная активизация драйвера EMM386 может быть произведена как из командной строки, так и по запросам программ. Пока драйвер EMM386 не активизирован, загрузку других драйверов через область UMB он не обслуживает.

8196 – пример спецификации размера EMS-памяти в килобайтах, допускается от 16 кбайт до фактического размера свободной XMS-памяти, но не свыше 32768 кбайт. По умолчанию

запрашивается вся доступная XMS-память, если только не указан параметр NOEMS, при котором устанавливаемый по умолчанию размер равен 0. В любом случае размер будет округлен до величины, кратной 16 кбайт.

min=256 – не устанавливать доступ к расширенной памяти согласно спецификации EMS, если размер этой памяти не превышает указанного минимального значения в пределах от 0 до запрашиваемой величины (только когда не указан параметр NOEMS).

W=ON – разрешить поддержку арифметического сопроцессора Weitek; по умолчанию его поддержка отключена (OFF). Когда драйвер EMM386 активизирован, он принимает из командной строки команды

EMM386 W=OFF
EMM386 W=ON.

M4 – пример спецификации начального сегментного адреса блока памяти (кадра) для последовательного размещения четырех EMS-страниц с номерами 0 – 3, по 16 кбайт каждая. Числа после буквы M – это коды (от 1 до 14), которым соответствуют адреса

1 = C000h; 2 = C400h; 3 = C800h; 4 = CC00h;
5 = D000h; 6 = D400h; 7 = D800h; 8 = DC00h;
9 = E000h; 10 = 8000h;
11 = 8400h; 12 = 8800h; 13 = 8C00h; 14 = 9000h

По умолчанию принимается начальный адрес E000h.

FRAME=CC00 – пример непосредственного задания того же самого начального сегментного адреса. Другие адреса, кроме 14 показанных выше, не допускаются. Для запрета постраничного доступа можно указать FRAME=NONE, это будет эквивалентно параметру NOEMS.

/PCC00 – еще один пример спецификации сегментного адреса кадра из EMS-страниц 0 – 3. После параметра /P может быть задан один из тех же самых 14 адресов.

P4=DC00– пример спецификации для добавления еще одной, пятой EMS-страницы P4 к кадру CC00h – DBFFh, содержащему EMS-страницы с номерами 0 – 3. Параметры "P" в строке можно указывать несколько раз с номерами EMS-страниц от 4 до 14. Указывать параметры "P" с номерами 0 – 3 тоже можно, но только если начальный сегментный адрес кадра не задан как-либо иначе, и если последовательное расположение EMS-страниц 0 – 3 сохранено.

X=F000-FFFF – пример спецификации, запрещающей использовать указанную область памяти в пределах диапазона адресов A000h

– FFFFh. Указанные сегментные адреса будут округлены вниз до величины, кратной 4-м килобайтам.

I=BC00-BFFF – пример спецификации, разрешающей использование указанного участка памяти для размещения UMB-блоков. Указанные сегментные адреса будут округлены вниз до величины, кратной 4-м килобайтам. Когда параметры "I" и "X" задают перекрывающиеся диапазоны адресов, преимущество имеет параметр "X".

B=4000 – сегментный адрес (в пределах 1000h – 4000h) допустимой нижней границы области размещения EMS-страниц. По умолчанию нижняя граница – 4000h.

L=256 – спецификация размера области памяти, резервируемой для доступа согласно спецификации XMS (в килобайтах, по умолчанию принимается 0).

A=7 – число банков регистров, создаваемых для обслуживания многозадачного доступа. Допускается от 0 до 254 банков, по умолчанию принимается 7. Каждый банк регистров занимает около 200 байт.

h=64 – количество ссылок на участки расширенной памяти, которые можно одновременно держать открытыми для доступа. Допускается от 2 до 255 ссылок, по умолчанию принимается 64.

d=32 – размер резервного буфера для обслуживания прямого доступа в память (DMA). Допускается от 16 до 256 кбайт, по умолчанию принимается 32.

RAM=C000-EFFF – пример спецификации границ области размещения UMB-блоков и EMS-страниц. Если за параметром RAM не следует спецификация сегментных адресов, то считается разрешенным все доступное пространство памяти.

WIN=E000-EFFF – пример резервирования области памяти для использования операционной системой WINDOWS.

ROM=F000-FFFF – загрузить копию кода BIOS в указанную область памяти, если копирование не производится самой системой BIOS. Копирование кода BIOS способствует повышению быстродействия компьютера.

NOEMS – не создавать EMS-страниц, исключить доступ к расширенной памяти согласно спецификации EMS, но использовать пространство 640 – 1024 кбайт для создания UMB-блоков и для загрузки драйверов.

NOHI – загрузить весь резидентный модуль драйвера EMM386 в обыкновенную память, то есть ниже границы 640 кбайт.

NOMOVEXBDA – не копировать код BIOS в память по умолчанию.

NOTR – не искать сетевую карту компьютера (только для EMM386 версий 4.45 – 4.95)

NOVCPI – не предоставлять службы VCPI (примечание 2) по запросам других программ, не давать им прав "распоряжаться" памятью по своему усмотрению. Параметр NOVCPI можно указывать только совместно с параметром NOEMS.

HIGHSCAN – запустить процедуру сканирования для поиска свободных участков памяти. Благодаря этой процедуре использование памяти становится более эффективным, но вместе с тем возрастает риск ошибок, из-за которых возможно зависание компьютера.

ALTBOOT – заменить обработчик перезагрузки компьютера. Этот параметр следует указывать только тогда, когда обычная комбинация CTRL-ALT-DEL перестает действовать после загрузки драйвера EMM386.EXE.

NOBACKFILL – предотвратить создание UMB-блоков в обыкновенной памяти, когда драйвер EMM386.EXE загружается на компьютерах, у которых вся память меньше 640 кбайт.

Когда драйвер EMM386.EXE уже загружен и активизирован, ему можно послать из командной строки команды "EMM386 OFF" и "EMM386 AUTO". Но эти команды не будут исполнены, если к тому моменту через область UMB уже загружены другие драйверы, которые должны оставаться доступными.

Примечание 1: драйвер EMM386.EXE принимает запросы на исполнение операций от программ посредством прерывания INT 67 (8.03-57 – 8.03-74).

Примечание 2: службы VCPI предоставляются программам через прерывания INT 67\AX=DE00h-DE0Ch. Они дают шанс исполнить свою миссию тем программам, которым необходимо использовать особенности защищенного режима. Благодаря службам VCPI, в частности, становится возможной последующая загрузка операционной системы Windows. Но в "окне DOS" операционной системы Windows службы VCPI недоступны, чтобы предотвратить операции, которые могли бы привести к дестабилизации.

Примечание 3: вариант драйвера EMM386.EXE, поставляемый в составе Windows-ME, не предназначен для обслуживания автономной работы MS-DOS8 и в такой роли с некоторыми модификациями процессоров вообще несовместим (вызывает "зависание" компьютера при загрузке).

Примечание 4: известны несколько аналогов описанного драйвера EMM386.EXE, пригодных для работы в среде MS-DOS7. Лучший из них – драйвер JEMM386.EXE, разработку которого курирует Том Ехлерт (Tom Ehlert). В декабре 2007 года архив JEMM568.ZIP с версией 5.68

этого драйвера был выложен на сайте http://japheth.de/ . Еще один аналог разрабатывался для проекта FreeDOS, его можно свободно скачать с сервера ftp://ftp.devoresoftware.com/downloads/ . Аналоги намного компактнее оригинала и отличаются от него перечнем принимаемых параметров.

5.04-03 Оптимизация памяти: драйвер CHKSTATE.SYS

Файл CHKSTATE.SYS – это служебная резидентная программа для обслуживания оптимизации распределения памяти в MS-DOS6.22, но оказавшаяся вполне пригодной и для MS-DOS7. CHKSTATE.SYS загружается как драйвер командой DEVICE из первой строки файла CONFIG.SYS, причем эта строка вписывается автоматически оптимизационной программой MEMMAKER.EXE, и так же автоматически удаляется по завершении процедуры оптимизации. В процессе пробной загрузки CHKSTATE.SYS формирует временный файл отчета со сведениями об участках памяти, выделяемых каждому загружаемому драйверу.

Примечание 1: для проведения процедуры оптимизации распределения памяти основная программа MEMMAKER.EXE должна находиться в одном каталоге с файлами CHKSTATE.SYS, EMM386.EXE, HIMEM.SYS, MEMMAKER.HLP, MEMMAKER.INF и SIZER.EXE. Все эти файлы входят в комплект поставки MS-DOS6.22, а также содержатся в файле архива OLDDOS.EXE, который можно свободно скачать из сети Интернет с сервера ftp://ftp.microsoft.com/softlib/mslfiles/ .

Примечание 2: программа MEMMAKER.EXE неспособна правильно занести скорректированные параметры (3.17, 4.07) в файлы CONFIG.SYS и AUTOEXEC.BAT при наличии меню с несколькими вариантами конфигураций загрузки. Каждый вариант загрузки следует оптимизировать отдельно, и только по полученным таким образом данным можно составлять конфигурационные файлы с несколькими оптимизированными вариантами загрузки.

5.04-04 UMBPCI.SYS – драйвер блоков UMB.

Чтобы открыть область UMB (C000h – EFFFh) для загрузки, драйвер EMM386.EXE (5.04-02) организует преобразования адресов через таблицы TLB и вынуждает платить за это переводом процессора в режим V86. Но когда необходимо оставаться в реальном режиме, доступ к области UMB может быть открыт путем перепрограммирования контроллера памяти, входящего в состав комплекта микросхем (чипсета) на материнской плате компьютера. Этот альтернативный вариант доступа реализован драйвером UMBPCI.SYS.

Разработка драйвера UMBPCI.SYS имеет долгую историю с участием многих программистов. Сейчас ее продолжает Уве Зибер (Uwe Sieber). Свежая версия драйвера UMBPCI.SYS бывает выложена на сайте http://www.uwe-sieber.de/ , и ее можно свободно скачать в составе файла архива UMBPCI.ZIP с текстами на немецком языке или в составе файла архива UMBPCI_E.ZIP с текстами на английском языке. Ниже описана версия 3.66 драйвера UMBPCI.SYS, датируемая мартом 2006 года.

Конечно, UMBPCI.SYS не заменяет драйвер EMM386.EXE в части осуществления спецификации EMS, он выполняет только функции, оговоренные спецификацией XMS, но именно те, которые обычно возложены не на HIMEM.SYS, а на EMM386.EXE (они перечислены в примечании 6 к A.12-3).

Для перепрограммирования контроллера памяти драйвер UMBPCI.SYS пользуется обработчиками прерываний BIOS, обслуживающими шину PCI (INT 1A\AH=B1h). Поэтому драйвер UMBPCI.SYS можно применять на компьютерах, которые имеют шину PCI и обеспечивают управление ею. Этим условиям удовлетворяют почти все AT-совместимые компьютеры, выпущенные после 1996 года. Исключение составляют компьютеры с процессорами AMD-K7: в них блоки UMB, организованные посредством UMBPCI.SYS, нельзя использовать для загрузки драйверов, обслуживающих платы расширения на шине PCI.

Физически UMBPCI.SYS задействует свободную часть области "теневой" памяти, которая предназначена для копирования кодов из более инерционных микросхем постоянной памяти BIOS. Не все комплекты микросхем (чипсеты) способны обеспечить прямой доступ (DMA) в эту область памяти, а он нужен контроллеру флоппи-дисков, драйверу SMARTDRV.EXE (5.06-01), и даже системе Windows, если потом надо будет ее загружать. Последствия ограничений на прямой доступ (DMA) к "теневой" памяти можно предотвратить с помощью дополнительных программ, поставляемых вместе с драйвером UMBPCI.SYS в тех же файлах архива. Там же даны рекомендации по преодолению проблем, специфичных для отдельных комплектов микросхем (чипсетов). Разумеется, критичные особенности состава оборудования компьютера должны быть учтены в конфигурации загрузки.

Загружать драйвер UMBPCI.SYS надо командой DEVICE (4.06) из строки файла CONFIG.SYS, следующей за строкой загрузки драйвера HIMEM.SYS, но предшествующей всем командам загрузки в область UMB (имеющим обычно окончание ...HIGH). Строка загрузки драйвера UMBPCI.SYS может выглядеть, например, так:

```
DEVICE=\DOS\DRV\UMBPCI.SYS  /I=D000-DFFF
```

Здесь

\DOS\DRV\ – пример пути к драйверу UMBPCI.SYS

/I=D000-DFFF – необязательный параметр, предписывающий открыть часть адресного пространства для размещения UMB-блоков. Границы выделенной части области UMB должны быть кратны 16 килобайтам (то есть C800, CC00, D000, D400 и так далее).

Если указан параметр /I=, то UMBPCI.SYS не будет искать незанятые участки памяти для размещения UMB-блоков. Допускается наличие нескольких параметров /I= в одной командной строке, и тогда каждая предписанная часть области UMB получит свой порядковый номер. Команды DEVICEHIGH (4.07) и LH (3.17) принимают порядковый номер части области UMB после параметра /L:, и благодаря тому можно загрузить любой конкретный драйвер в предназначенную для него часть адресного пространства. Последнее важно для драйверов, пользующиеся прямым доступом (DMA), потому что некоторые комплекты микросхем (чипсеты) – например, i430TX – обеспечивают DMA не во всей области UMB, а только в пределах сегментных адресов E000–EFFFh. Однако обычно параметр /I= указывать не приходится, так как наиболее распространенные комплекты микросхем (i815, i820, i845, i850, i855 и многие другие) не накладывают никаких ограничений на доступ к области UMB.

Примечание 1: в старых компьютерах, не имеющих шины PCI, доступ к области UMB может быть открыт с помощью драйвера HIRAM.EXE и сопровождающих его дополнительных программ. Весь этот комплект, написанный еще в 1993 году, можно скачать в составе одного файла архива http://www.uwe–sieber.de/files/hiram.zip .

5.05 Драйверы RAM-дисков

Драйверы RAM-дисков используют часть оперативной памяти компьютера (RAM) для создания виртуального записываемого диска. Это дает возможность пользоваться записью на диск при восстановительных и настроечных работах, когда физическое записываемое дисковое пространство может быть недоступно или когда оставлять какие-либо следы доступа на реальном диске нежелательно. Помимо прочего, виртуальные RAM-диски работают намного быстрее реальных дисководов. Поскольку содержимое RAM-дисков теряется при каждом выключении компьютера, постольку их целесообразно использовать для записи временных файлов и всего того, чем физические диски захламлять не хочется.

Общий недостаток многих драйверов RAM-дисков – их неспособность обеспечить присвоение заданного буквенного обозначения создаваемому виртуальному диску. DOS всегда назначает создаваемому диску первую свободную букву, которая заранее неизвестна. Одно решение, пример которого приведен в разделе 9.04-02, заключается в том, чтобы потом искать букву, назначенную именно RAM-диску. Другой выход – это заранее обмануть DOS в отношении

действительного количества имеющихся дисков с помощью драйвера фантомных дисков. Пример такого драйвера приведен в разделе 9.08.

5.05-01 Драйвер RAM-диска RAMDRIVE.SYS

Драйвер RAMDRIVE.SYS разработан фирмой Microsoft и поставляется в составе операционных систем WINDOWS-95/98/ME. При их стандартной установке этот драйвер находится в каталоге \WINDOWS.

Драйвер RAMDRIVE.SYS предназначен для загрузки командой DEVICE (4.06) или DEVICEHIGH (4.07) из строки файла CONFIG.SYS:

```
DEVICE=C:\DOS\DRV\RAMDRIVE.SYS 16000 512 256 /E
```

здесь:

C:\DOS\DRV\ – пример пути к драйверу RAMDRIVE.SYS.

16000 – пример спецификации размера создаваемого RAM-диска в килобайтах, допускается от 4 до 32767 кбайт, по умолчанию принимается 64 кбайт.

512 – пример спецификации размера сектора в байтах; допускаются значения 128, 256 и 512 байт, по умолчанию принимается 512 байт. Если размер сектора указан, то размер RAM-диска должен быть указан тоже.

256 – пример спецификации максимального суммарного числа файлов и каталогов в корневом каталоге RAM-диска, допускается от 2 до 1024, по умолчанию принимается 64. Если объем корневого каталога указан, то размер сектора RAM-диска должен быть указан тоже.

/E – необязательный параметр, определяющий предпочтительное размещение RAM-диска в XMS-памяти, обеспечиваемой драйвером HIMEM.SYS. Вместо параметра /E можно указать параметр /A, задающий предпочтение использованию EMS-памяти, обеспечиваемой драйвером EMM386.EXE.

Чтобы создать несколько RAM-дисков, нужно добавить в файл CONFIG.SYS по одной строке загрузки драйвера RAMDRIVE.SYS на каждый диск.

Примечание 1: RAM-диски объемом до 2 Гб может создать драйвер RDISK.COM, выложенный на сайте http://johnson.tmfc.net/dos/driver.html . Для этого драйвера необходим менеджер памяти, предоставляющий доступ к XMS-пространству до 4 Гб (примечание 3 к 5.04-01).

5.05-02 Реконфигурирующие драйверы TDSK.EXE и BITDISK.EXE

Свободно распространяемые драйверы TDSK.EXE и BITDISK.EXE дают возможность задавать размер создаваемого RAM-диска не только в момент загрузки, как это делает драйвер RAMDRIVE.SYS, но также и потом, и даже неоднократно. Данное свойство особенно важно при загрузке на незнакомый компьютер: оно позволяет сначала исследовать объем имеющейся памяти и только затем принимать решения о создании RAM-диска и выборе его емкости.

Драйверы TDSK.EXE и BITDISK.EXE написаны Гарсиа де Селисом (Garcia de Celis) в 1992 – 1995 годах, и тогда последней была версия 2.3. Позже эти драйверы были модернизированы и включены в пакет программ FreeDOS. Теперь их можно скачать с сайта http://www.ibiblio.org/pub/micro/pc-stuff/freedos/files/dos/ramdisk/ . Там оригинальные драйверы Гарсиа де Селиса содержатся в файле архива TDSK23.ZIP, а модернизированная версия 2.42 драйвера TDSK.EXE – в файле архива TDSK242B.ZIP.

BITDISK.EXE – это упрощенный и укороченный вариант драйвера TDSK.EXE. В то время как последний способен размещать RAM-диск в EMS-памяти, XMS-памяти или в обыкновенной памяти, BITDISK работает только с XMS-памятью и потому всегда требует, чтобы заранее был загружен драйвер XMS-памяти HIMEM.SYS. TDSK.EXE принимает все параметры драйвера BITDISK.EXE и, помимо того, ряд своих собственных параметров.

Драйверы RAM-дисков версии 2.3 загружают командой DEVICE (4.06) или DEVICEHIGH (4.07) из строки файла CONFIG.SYS. Драйвер TDSK.EXE версии 2.42 следует загружать в обыкновенную память, т. е. только командой DEVICE (4.06) при обязательном наличии в файле CONFIG.SYS строки DOS=NOAUTO (4.08). Командная строка загрузки драйверов может включать все параметры, определяющие создание RAM-диска, и тогда он будет создан сразу. Но если создание RAM-диска надо отложить, то конкретные значения параметров в строке файла CONFIG.SYS не указывают, и она может выглядеть, например, так:

```
DEVICEHIGH=C:\DOS\DRV\BITDISK.EXE 0
```

или так:

```
DEVICE=C:\DOS\DRV\TDSK.EXE 0
```

здесь:

C:\DOS\DRV\ – пример пути к драйверу;
0 – нулевой размер означает, что создавать RAM-диск не нужно.

В результате исполнения такой командной строки память под RAM-диск не выделяется, но происходит загрузка и инициализация резидентного модуля, занимающего около 600 байт. На этой же стадии DOS регистрирует новый RAM-диск и присваивает ему буквенное обозначение. Если файл CONFIG.SYS

содержит несколько таких строк, то произойдет инициализация нескольких RAM-дисков.

Чтобы сделать RAM-диск реально доступным, нужно потом тот же драйвер запустить на исполнение из строки файла AUTOEXEC.BAT или непосредственно из командной строки, например:

```
C:\DOS\DRV\BITDISK.EXE R: 5600 512 384 4 /F:2
```

или

```
C:\DOS\DRV\TDSK.EXE R: 5600 512 384 4 /F:2 /E /M /I=1
```

здесь:

R: – пример буквы, назначенной адресуемому RAM-диску, если создано несколько RAM-дисков. Для единственного RAM-диска букву можно не указывать.

5600 – пример обязательной спецификации размера создаваемого RAM-диска в килобайтах. Минимальный размер 4 кбайт, максимальный – 32766 кбайт для BITDISK.EXE и 65534 кбайт для TDSK.EXE. Если указать размер 0, то выделенная RAM-диску память будет освобождена, но резидентный модуль драйвера останется в памяти и будет готов принять запрос на воссоздание RAM-диска с новыми параметрами.

512 – пример необязательной спецификации размера сектора в байтах; допустимы значения 64, 128, 256 и 512 байт, по умолчанию принимается 512 байт.

384 – пример необязательной спецификации максимального суммарного числа файлов и каталогов в корневом каталоге RAM-диска, допускается от 1 до величины размера диска, по умолчанию принимается 384. Если объем корневого каталога указан, то размер сектора RAM-диска должен быть указан тоже.

4 – необязательное число секторов в кластере, в MS-DOS7 оно должно выражаться степенью числа 2. По умолчанию оно принимается минимально возможным для заданного размера RAM-диска. Если число секторов в кластере не опущено, то объем корневого каталога должен быть указан тоже.

/F:2 – этот необязательный параметр задает создание RAM-диска с двумя таблицами FAT, как у реальных физических дисков. По умолчанию создается только одна таблица FAT. В редких случаях встречаются программы, которые не могут правильно работать с дисками, имеющими только одну таблицу FAT.

Командная строка, адресуемая драйверу TDSK.EXE, содержит несколько дополнительных необязательных параметров. В частности, драйвер TDSK.EXE версии 2.42 дополнительно принимает следующие параметры:

/E – разместить RAM-диск в XMS-памяти при условии, что драйвер HIMEM.SYS уже успешно загружен. Именно этот вариант размещения принимается по умолчанию.

/C – разместить RAM-диск в обыкновенной памяти.

/X – (и также /A) – разместить RAM-диск в EMS-памяти, при условии, что уже успешно загружены оба драйвера памяти – EMM386.EXE и HIMEM.SYS.

/B – не загружать драйвер TDSK.EXE, если в компьютере имеется хотя бы один накопитель на жестких магнитных дисках (этот параметр не принимается версией 2.3).

/M – выводить сообщения на экран в черно-белом виде (по умолчанию сообщения выводятся в цвете).

/I=1 – выводить сообщения на английском языке (английский язык используется по умолчанию). Помимо того, можно выводить сообщения на испанском (/I=34) и немецком (/I=49) языках.

Дополнительные необязательные параметры /E, /C, /X, /A являются взаимно исключающими. Их обычно не указывают, потому что драйвер TDSK.EXE способен самостоятельно находить оптимальный вариант размещения RAM-диска.

Показанная выше композиция командной строки может быть использована для изменения размера уже существующего диска, но при каждом таком переопределении RAM-диск создается заново, и все его прежнее содержание полностью теряется. Только в двух случаях запуска на исполнение из командной строки содержимое RAM-диска не изменяется:

– если вызов производится без параметров, чтобы показать статус RAM-диска;

– если вызов производится с единственным параметром /?, чтобы вывести на экран краткую справку.

Чтобы снять проблему определения буквенного обозначения, назначенного RAM-диску, драйвер TDSK.EXE версии 2.42 вписывает эту букву в значение переменной окружения, если сможет найти в пространстве окружения переменную с именами TURBODSK или RAMDRIVE. Переменную, имеющую одно из указанных имен, следует создать в пространстве окружения заранее с помощью команды SET (3.26), причем в качестве значения этой переменной должны быть указаны знак вопроса и двоеточие:

```
SET RAMDRIVE=?:
```

Если переменная с искомым именем в момент вызова драйвера имеет другое значение, то оно будет сохранено без изменения. Но если значение именно такое, и если к данному моменту RAM-диск уже создан, то после вызова драйвера на исполнение знак вопроса в значении переменной будет заменен буквой диска. Далее уже бывает несложно составить файл AUTOEXEC.BAT так, чтобы буква

RAM-диска была автоматически учтена в конфигурации загрузки компьютера (пример – в разделе 9.09-02).

После запуска на исполнение из командной строки драйвер TDSK.EXE оставляет код ошибки (errorlevel, 3.15-03 и 9.07-03); коды от 1 до 128 означают номерную ссылку (handle), присвоенную выделенной области XMS-памяти или EMS-памяти. Другие значения кода ошибки означают следующее:

0 – диск не создан потому что не определен
252 – синтаксическая ошибка
253 – попытка создать RAM-диск в многозадачной среде (например, в Windows)
254 – неверное указание буквы диска
255 – резидентный модуль драйвера не загружен.

5.06 Драйверы обслуживания дисковых накопителей

5.06-01 Драйвер кэш-буфера SMARTDRV.EXE

Драйвер SMARTDRV.EXE организует в памяти кэш-буфер с обслуживанием операций считывания и записи данных через контроллер прямой записи в память (DMA). Благодаря тому снижается загрузка центрального процессора, быстрее выполняются пересылки данных как накопителями на магнитных дисках, так и оптическими дисководами. Эффект особенно существенен при пересылке больших объемов данных, например, в процессе установки операционных систем Windows-98/ME/2000/XP из среды DOS.

Драйвер SMARTDRV.EXE входит в поставку операционных систем Windows-95/98 и при их стандартной установке находится в каталоге \WINDOWS. Однако сами операционные системы Windows-95/98 осуществляют буферирование пересылок средствами защищенного режима и не пользуются драйвером SMARTDRV.EXE. Он бывает нужен только при работе в среде DOS, в том числе в MS-DOS7.

Поскольку кэш-буфер целесообразно создавать за пределами обыкновенной памяти (выше 640 кбайт), постольку доступ туда надо открыть заранее, до загрузки SMARTDRV.EXE, с помощью драйверов памяти (5.04-01, 5.04-02, 5.04-04). Если для доступа к оптическим дискам Вы намерены использовать программу MSCDEX.EXE (5.08-03), то ее тоже надо загрузить заранее. Обычно драйвер SMARTDRV.EXE загружают либо командой INSTALL (4.15) из файла CONFIG.SYS, либо из строки файла AUTOEXEC.BAT, например:

```
C:\DOS\DRV\SMARTDRV.EXE /X A- B- C+ /U /N /L /V 128 /E:2048 /B:4096
```

здесь:

C:\DOS\DRV\ – пример пути к драйверу SMARTDRV.EXE

/X – необязательный параметр, исключающий кэширование записи на всех дисках, кроме тех, которые перечислены после этого параметра со знаком плюс (в частности, C+). Если параметр /X опущен, то по умолчанию кэширование записи осуществляется только по отношению к жестким магнитным дискам. На дисках, которые указаны после параметра /X со знаком минус (в частности, A– B–) исключается также кэширование операций чтения.

/U – необязательный параметр, исключающий загрузку модуля обслуживания оптических дисководов (CD-ROM).

/N – необязательный параметр, разрешающий обслуживание следующей команды тогда, когда еще не закончен процесс записи на диск данных, принятых в буфер от предыдущей команды. Параметр /N затрагивает операции только на тех дисках, на которых разрешено кэширование записи. Если параметр /N не указывать, то приглашение командной строки не появится, пока запись не будет завершена.

/L – необязательный параметр, предписывающий разместить кэш-буфер в обыкновенной памяти, то есть в области ниже 640 кбайт. Это бывает нужно, когда контроллер DMA не обслуживает область UMB (пример – в 5.04-04).

/V – необязательный параметр, вызывающий вывод на экран сведений о статусе драйвера SMARTDRV.EXE (по умолчанию никакие сообщения не выводятся). Вместо параметра /V можно указать параметр /S, вызывающий вывод на экран сведений о статусе вместе со статистическими данными.

128 – размер кэш-буфера в килобайтах. Его следует указывать, когда значительную часть XMS-памяти надо резервировать в других целях, например, для создания RAM-диска.

/E:2048 – размер передаваемых блоков данных. Допускаются значения 1024, 2048, 4096 и 8192 байт, по умолчанию принимается 8192 байт.

/B:4096 – размер в байтах буфера для упреждающего считывания, он должен быть кратен размеру передаваемых блоков данных, по умолчанию принимается 16384 байт.

Когда драйвер SMARTDRV.EXE загружен и уже работает, его можно снова вызвать на исполнение из командной строки, но с другим комплектом необязательных параметров для однократного исполнения команды или для реконфигурации, например:

```
C:\DOS\DRV\SMARTDRV.EXE /X C- D+ /R /F /Q
```

здесь:

/X – отменить кэширование операций записи на всех дисках, если оно не было отменено раньше, или изменить статус кэширования операций записи для перечисляемых далее дисков. В данном примере C– D+ означает запретить кэширование операций записи для диска C: и разрешить кэширование операций записи для диска D:.

/R – очистить кэш-буфер и перезапустить драйвер SMARTDRV.EXE. Вместо параметра /R можно указать другую однократную операцию, задав параметр /C – записать на диск все данные, находящиеся в настоящий момент в кэш-буфере.

/F – отменить действие параметра /N, если он действует в настоящий момент, и вернуться к исходному состоянию, когда приглашение командной строки появляется только после окончания записи данных из кэш-буфера. Если же нужно изменить состояние в противоположную сторону, то вместо параметра /F нужно указать параметр /N.

/Q – не выводить на экран сообщение о статусе, которое выводится по умолчанию. Вместо параметра /Q можно указать параметр /S – показать статус вместе со статистикой кэш-буфера.

Примечание 1: при выполнении операции сброса данных из кэш-буфера на диск, задаваемой параметром /C, никакие сообщения на экран не выводятся, а параметры /V и /S игнорируются.

Примечание 2: драйвер SMARTDRV.EXE не обслуживает пересылки данных, организуемые программой SHSUCDX.COM (5.08-04).

Примечание 3: для работы с интерфейсом SATA и для выполнения пересылок в режиме UltraDMA следует использовать кэш-драйвер UIDE.SYS, выложенный на сайте http://johnson.tmfc.net/dos/driver.html .

5.06-02 Драйвер двойного буферирования DBLBUFF.SYS

Драйвер DBLBUFF.SYS обеспечивает двойное буферирование для достижения совместимости с некоторыми контроллерами, которые иначе не могут работать с EMS-памятью и в системе Windows. В частности, двойное буферирование требуется многим контроллерам, обслуживающим жесткие магнитные диски с интерфейсом SCSI. Драйвер DBLBUFF.SYS входит в комплект поставки операционных систем Windows-95/98 и при их стандартной установке находится в каталоге \WINDOWS.

В конфигурационном файле MSDOS.SYS (5.01-01) имеется строка с командой DoubleBuffer. Если там указано DoubleBuffer=1, то MS-DOS7 будет пытаться загрузить драйвер DBLBUFF.SYS по умолчанию. В противном случае драйвер

DBLBUFF.SYS нужно загружать явно командой DEVICE (4.06) из строки файла CONFIG.SYS:

```
DEVICE=C:\DOS\DRV\DBLBUFF.SYS /D+
```

здесь:

C:\DOS\DRV – пример пути к драйверу DBLBUFF.SYS.

/D+ – необязательный параметр, вызывающий постоянную установку двойного буферирования для всех дисков. По умолчанию подлежат двойному буферированию только операции ввода-вывода с использованием UMB-блоков (5.04-02), причем драйвер DBLBUFF.SYS не загрузится, если двойное буферирование не будет сочтено нужным.

Примечание 1: сообщение о статусе, выводимое на экран драйвером SMARTDRV.EXE (5.06-01) содержит колонку "buffering" (буферирование). Наличие в этой колонке хотя бы одного подтверждения "yes" свидетельствует о том, что драйвер DBLBUFF.SYS должен быть загружен.

Примечание 2: если двойное буферирование необходимо, то в файле CONFIG.SYS командой BUFFERS (4.03) или командой BUFFERSHIGH (4.04) должно быть зарезервировано ненулевое количество вторичных буферов.

5.06-03 Сжатие логических дисков: драйвер DRVSPACE.SYS

Драйвер DRVSPACE.SYS – это загрузчик резидентной программы DRVSPACE.BIN, осуществляющей сжатие логических дисков "на лету", то есть выполняющей сжатие и восстановление данных в процессе доступа к дискам. Поток сжатых данных формируется так, чтобы заполнять кластеры на 100%, благодаря чему свободное место не теряется в частично заполненных кластерах. Все это существенно повышает эффективность использования дискового пространства.

Файлы DRVSPACE.SYS и DRVSPACE.BIN входят в комплект поставки операционной системы Windows-95 и должны находиться в корневом каталоге вне сжимаемой части дискового пространства. Когда MS-DOS7 обнаруживает наличие сжатой области на загрузочном диске, она по умолчанию запускает на исполнение загрузчик DRVSPACE.SYS. Загрузку по умолчанию можно отменить введением строки "DRVSPACE=0" в файл MSDOS.SYS (5.01-01) или командой "DOS=NOAUTO" в файле CONFIG.SYS (4.08, 9.01-01).

Файл DRVSPACE.SYS загружается как драйвер командой DEVICE (4.06) из строки файла CONFIG.SYS, причем эта строка должна предшествовать командам загрузки драйверов памяти. Вот пример строки загрузки файла DRVSPACE.SYS:

```
DEVICE=C:\DRVSPACE.SYS /MOVE /NOHMA /LOW
```

здесь:

C:\ – пример пути к файлу DRVSPACE.SYS.

/MOVE – необязательный параметр, вызывающий перемещение кода DRVSPACE.BIN из верхней части обыкновенной памяти, куда он загружается сначала. Перемещение происходит после исполнения всех команд DEVICE и DEVICEHIGH в файле CONFIG.SYS с целью предотвращения конфликтов с другими программами. Обычно перемещение производится в области 640 – 1024 или 1024 – 1088 кбайт, но если драйверы расширенной памяти не загружены, то используется нижняя часть обыкновенной памяти.

/NOHMA– не перемещать код DRVSPACE.BIN в область верхней памяти 1024 – 1088 кбайт (HMA).

/LOW – переместить код DRVSPACE.BIN в нижнюю часть обыкновенной памяти, даже если расширенная память доступна.

Если код резидентной программы DRVSPACE.BIN загружен, то слово DRVSPACE становится зарезервированным названием команды, исполнение которой открывает диалоговое взаимодействие с пользователем. В ходе этого взаимодействия пользователю предоставляется возможность преобразовывать диски и дискеты в сжатый формат и обратно, создавать новые сжатые диски, подвергать сжатые диски тестированию и дефрагментации, устанавливать режим защиты и т.д.

Сжатие логических дисков программой DRVSPACE.BIN не получило широкого распространения по нескольким причинам. Во-первых, это несовместимость сжатых дисков с другими версиями DOS и другими операционными системами (среди которых WINDOWS-95 – исключение). Вторая причина состоит в том, что сжатые диски более подвержены влиянию возможных ошибок и в меньшей степени поддаются действию процедур восстановления данных. Третья причина в том, что в наше время проблема нехватки дискового пространства потеряла ту остроту, которую она имела в начале 1990-х годов. Для современных быстродействующих дисков большого объема сжатие "на лету" не окупает того снижения быстродействия, которое вызвано необходимостью проведения операций сжатия и восстановления данных. По тем же причинам загрузка со сжатием "на лету" не включена в примеры конфигурационных файлов в главе 9.

Примечание 1: программа DRVSPACE.BIN обслуживает только логические диски с файловой системой FAT-16. Диски с файловой системой FAT-32 сжатию "на лету" не подлежат.

5.07 Драйверы контроллеров интерфейса

5.07-01 ATAPIMGR.SYS: драйвер интерфейса ATAPI

В наше время наибольшее распространение получили дисководы со встроенными блоками электроники (IDE = Integrated Drive Electronics). Впервые такие дисководы были изготовлены фирмой Western Digital и применены в компьютерах PC-AT фирмы IBM в 1984 году. Протокол взаимодействия таких дисководов с контроллером получил название ATA (= Advanced technology attachment) и с 1994 года стал стандартом ANSI X3.221-1994. В связи с распространением оптических дисководов и дисководов на сменных магнитных дисках повышенной емкости протокол ATA пришлось дополнить в 1998 году новыми командами, в том числе командами пакетной передачи данных. После этого он стал называться пакетным интерфейсом ATA (ATA Packet Interface), или ATAPI.

В компьютерах, выпущенных после 2003 года, взаимодействие по протоколу ATAPI обычно обеспечивается системой BIOS материнской платы. Характерным признаком поддержки протокола ATAPI системой BIOS является возможность загрузки компьютера с оптических дисков DVD. Но если Ваш компьютер на это не способен, то Вам для пользования функциями протокола ATAPI надо будет загрузить драйвер интерфейса ATAPIMGR.SYS, разработанный фирмой Matsushita (торговая марка Panasonic). Благодаря ему стандартный IDE-контроллер сможет полноценно взаимодействовать с драйверами дисководов на сменных магнитных дисках, а также с драйверами оптических дисководов DVD-ROM, CD/DVD-R/RW и т.д. Версию 2.04 драйвера ATAPIMGR.SYS можно свободно скачать с сайта http://panasonic.co.jp/pcc/products/drive/internal/support/info_dd2.html в составе самораспаковывающегося архива 85x_dos.exe.

Драйвер ATAPIMGR.SYS нужно загружать из строки файла CONFIG.SYS командой DEVICE (4.06) или DEVICEHIGH (4.07), причем раньше всех тех драйверов, которые будут пользоваться его услугами. Строка загрузки может выглядеть, например, так:

```
DEVICE=\DOS\DRV\ATAPIMGR.SYS /P:170,15 /W:2 /NDR /NRS /C:2 /T:5 /LUN
```

здесь:

\DOS\DRV\ – пример пути к драйверу ATAPIMGR.SYS.

/P:170,15 – шестнадцатеричный адрес порта и номер IRQ линии запроса прерывания. Допустимы адреса 1F0, 170, 1E8, 168 (A.14-1). Допустимые номера IRQ 10, 11, 12, 14 и 15. Если эти данные не указаны, драйвер будет проводить поиск дисководов по всем принимаемым по умолчанию адресам.

/W:2 – циклы ожидания, от 0 до 99, для операций ввода-вывода данных в старых компьютерах, не поддерживающих команду подтверждения готовности к обмену (IOCHRDY). Количество циклов ожидания зависит от тактовой частоты центрального процессора: 50 МГц – /W:2, 75 МГц – /W:6, 100 МГц – /W:9, 166 МГц – /W:19, 240 МГц – /W:30. В более современных компьютерах параметр /W указывать не нужно.

/NDR – не устанавливать дисководы CD/DVD-ROM в начальное состояние. Этот параметр нужен, когда компьютер загружается с оптического диска, иначе процесс загрузки будет прерван.

/NRS – не отвечать "запрос принят" (RequestSense), когда дисковод посылает команду "проверить условия" (CheckCondition).

/C:2 – переустановить режим ввода-вывода (PIO) для указанного дисковода :

- 0 – ведущий дисковод 1-го контроллера (Primary Master),
- 1 – ведомый дисковод 1-го контроллера (Primary Slave),
- 2 – ведущий дисковод 2-го контроллера (Secondary Master),
- 3 – ведомый дисковод 2-го контроллера (Secondary Slave).

/T:5 – установить 5-секундный предел интервала ожидания отклика от дисковода; по умолчанию принимается 30 секунд.

/LUN – избирательно поддерживать обращение к устройству с локальным номером 0 (LUN = Local Unit Number). Номера LUN позволяют рассматривать многофункциональные дисководы как разные устройства, в частности, дисковод с диском DVD-RAM – как сменный магнитный диск, а тот же дисковод с обычным компакт-диском – как дисковод CD-ROM. По умолчанию обращения по номерам LUN не поддерживаются.

Если "поверх" MS-DOS7 с загруженным драйвером ATAPIMGR.SYS устанавливается операционная система WINDOWS-95/98, то надо побеспокоиться о том, чтобы она не осталась работать в режиме совместимости с MS-DOS. Для этого в файл \Windows\IOS.INI надо добавить строку:

```
ATAPIMGR.SYS  ; MKE ATAPI Manager
```

Если драйвер ATAPIMGR.SYS обнаружит, что функции обслуживания интерфейса ATAPI уже предоставлены системой BIOS компьютера, то он загружаться не станет. Когда драйвер ATAPIMGR.SYS не загружен, некоторые драйверы дисководов (в частности, SR_ASPI.SYS) отказываются загружаться тоже.

В сочетании с драйвером ATAPIMGR.SYS лучше использовать такой драйвер дисковода, который задействует функции обслуживания интерфейса ATAPI, не проверяя, кем они предоставлены. Примерами таких драйверов являются OAKCDROM.SYS (5.09-01) и VIDE-CDD.SYS (5.09-02). Они обеспечат доступ к дискам DVD независимо от того, "захочет" ли драйвер ATAPIMGR.SYS загружаться или нет.

5.07-02 Драйверы интерфейса PCMCIA

С начала 1990-х годов портативные компьютеры класса "ноутбук" оснащали специальным 68-контактным разъемом для карт расширения памяти. Интерфейс этих карт был стандартизован в 1990-м году Международной Ассоциацией производителей карт расширения памяти для компьютеров (PC Memory Card International Association = PCMCIA). В 1995-м году он был переименован в "PC card"- интерфейс, потому что в то время его уже использовали для подключения разного периферийного оборудования: внешних дисководов, модемов и т.п. Последняя 8-я версия стандарта интерфейса "PC card" была принята в 2001-м году. Позднее интерфейс "PC card" был вытеснен интерфейсом USB 2.0 (5.07-05).

В первой половине 1990-х годов применяли несколько взаимно несовместимых типов контроллеров PCMCIA, и тогда каждое внешнее устройство в интерфейсом PCMCIA приходилось снабжать несколькими драйверами для DOS. Так, в комплекте программ оптического дисковода Panasonic KXL-DN720A (1995 года) для разных контроллеров PCMCIA имеются 3 драйвера , позволяющие обращаться к подключенным приборам посредством унифицированного набора команд ASPI, то есть так же, как к приборам с интерфейсами SCSI (5.07-03) и USB (5.07-05). Этот комплект можно свободно скачать из сети Интернет с сервера ftp://ftp.panasonic.com/pub/Panasonic/Drivers/CDROM/ в самораспаковывающемся файле архива 720PCM32.EXE.

Позже преимущественное распространение получили контроллеры типа i82365 и их аналоги, а портативные компьютеры стали продавать с предустановленной операционной системой Windows без драйверов для DOS. Чтобы сохранить возможность пользования портативными накопителями в нештатных ситуациях, их теперь снабжают драйверами, которые обращаются из среды DOS прямо к портам контроллера i82365. Такие драйверы предоставляет, в частности, фирма Novac. Версия 4.0 (1999 года) драйвера NVICDF.EXE для оптических дисководов свободно выложена на сайте http://www.driver.novac.co.jp/driver/sta_PCMCIA/pcm_drv.html в составе файла архива Fdos.zip. Версия 4.0 (2000 года) драйвера NVIHD.EXE для не-оптических накопителей также свободно выложена на сайте http://www.driver.novac.co.jp/driver/hd150p/hd150p_drv.html в составе файла архива compact_PCMCIA.zip

Каждый из упомянутых драйверов фирмы Novac загружают без параметров из строк файла CONFIG.SYS командами DEVICE (4.06) или DEVICEHIGH (4.07). Потом, когда в слот PCMCIA будет вставлена карта с дисководом, надо повторно вызвать тот же драйвер из командной строки:

```
NVICDF.EXE /E
```

или

```
NVIHD.EXE  /E  /I
```

где параметры означают:

/E – инициализировать карту PCMCIA
/I – запустить мотор дисковода

Привод оптического дисковода запускается автоматически, когда в нем имеется диск, и потому драйвер NVICDF.EXE не нуждается в параметре /I. К сожалению, автор не имел достаточной базы для тестирования драйверов PCMCIA, так что к приведенным здесь сведениям нужно относиться осмотрительно.

Примечание 1: комплект поставки WINDOWS-ME содержит DOS-драйвер CARDDRV.EXE для внешних накопителей на жестких магнитных дисках с интерфейсом PCMCIA. Однако точных сведений об этом драйвере получить не удалось.

5.07-03 Драйверы интерфейса SCSI

Системный интерфейс для малых компьютеров (Small Computer's System Interface = SCSI) был разработан в 1982-м году фирмой Shugart и стал широко известен благодаря использованию в популярных тогда компьютерах фирмы Apple. В 1986-м году интерфейс SCSI получил статус стандарта ANSI X3.131-1986. С тех пор он неоднократно подвергался радикальным усовершенствованиям. В AT-совместимых компьютерах интерфейс SCSI применяют редко, причем преимущественно в серверах. По этой причине здесь кратко изложены только основные принципы конфигурирования загрузки DOS на компьютеры с интерфейсом SCSI.

Доступ к приборам, подключенным к шине SCSI, обеспечивается либо системой BIOS, либо устанавливаемыми драйверами. Серверные материнские платы обычно содержат встроенный SCSI-контроллер и расширения BIOS, обеспечивающие доступ через шину SCSI такой же, как через шину IDE, то есть без загрузки дополнительных драйверов. Однако многие версии расширений BIOS не будут загружаться, если на шине SCSI в момент включения компьютера не имеется хотя бы одно активное устройство, причем дисководы на сменных дисках без вставленного в них диска активными устройствами не считаются. Если к конкретному дисководу на сменных дисках требуется доступ средствами BIOS,

например, чтобы отформатировать сменный диск как жесткий магнитный диск, то надо успеть вставить этот сменный диск в дисковод до того, как BIOS начнет проводить свои стартовые тесты. Когда BIOS обнаруживает в дисководе загрузочный диск, с него можно загружать операционную систему, но для этого данный дисковод должен быть заранее отмечен как загрузочный в настройках программы BIOS Setup. Те дисководы на шине SCSI, которые в начальный момент зарегистрированы системой BIOS как неактивные, не будут доступны через расширения BIOS, даже если SCSI-расширения BIOS загружены и обеспечивают доступ к другим дисководам.

Некоторые SCSI-контроллеры на платах расширения также содержат встроенную память с записью расширений BIOS, и тогда доступ к устройствам на шине SCSI обеспечивается так же, как посредством встроенного контроллера. Наличие SCSI-расширений BIOS легко определить по перечню настроек, предлагаемых программой BIOS Setup. Тем не менее большинство дешевых SCSI-контроллеров на платах расширения не содержат встроенного программного обеспечения. Загружать компьютер с таких плат нельзя. Доступ к устройствам, подключаемым к таким контроллерам, обеспечивается с помощью загружаемых драйверов. Для дисководов на сменных дисках такая форма доступа удобнее, потому что позволяет регистрировать структуру разделов заново при каждой смене диска, тогда как большинство версий BIOS регистрирует структуру разделов диска только один раз при включении компьютера.

Комплекты драйверов для устройств, подключаемых к шине SCSI, разработаны несколькими фирмами; наиболее широко известны Adaptec, DTC, Mylex, Tekram. Каждый комплект включает драйвер SCSI-контроллера (ASPI8U2.SYS от Adaptec, AS80DOS.SYS от DTC, и т.д.), драйвер для накопителей на жестких магнитных дисках (ASPIDISK.SYS от Adaptec, DISKDOS.SYS от DTC, и т.д.) и драйвер дисковода CD-ROM (ASPICD.SYS от Adaptec, CDDOS.SYS от DTC, и т.д.). Все эти драйверы надо загружать командами DEVICE или DEVICEHIGH из строк файла CONFIG.SYS, причем загрузка драйвера SCSI-контроллера должна предшествовать загрузке драйверов для всех других устройств, подключаемых к шине SCSI.

Драйверы для накопителей на жестких магнитных дисках и для дисководов CD-ROM, предлагаемые производителями SCSI-контроллеров, в значительной степени унифицированы, потому что они пользуются стандартизованным набором команд ASPI. Возможность обращаться к разным устройствам посредством одного набора команд обеспечивают драйверы SCSI-контроллеров, каждый из которых, как правило, обслуживает серии из нескольких типов контроллеров. В частности, загрузочные дискеты для операционных систем Windows-95/98 содержат два комплекта SCSI-драйверов от фирм Adaptec и Mylex. Считают, что их достаточно для почти любого компьютера с шиной SCSI. Конфигурационные файлы на этих дискетах могут быть взяты за образец загрузки драйверов для обслуживания шины SCSI.

5.07-04 Драйверы интерфейса IEEE1394 (FireWire)

Интерфейс FireWire разработан фирмой Apple в 1987 году для подключения приборов с большими скоростями передачи данных (до 393 Мегабит/с). Такие потоки данных характерны для видеокамер, устройств видеозаписи, внешних дисковых накопителей. С 1995 года интерфейс FireWire стал стандартом IEEE1394. Для AT-совместимых компьютеров выпускают контроллеры интерфейса IEEE1394 на платах расширения, но массового распространения они не получили.

К настоящему времени известны два драйвера, обслуживающих контроллеры интерфейса IEEE1394 в среде DOS. Версию 1.01 (2002 года) драйвера ASPI1394.SYS фирмы Iomega можно свободно скачать из сети Интернет с сайта http://www.stefan2000.com/darkehorse/PC/DOS/Drivers/USB/ в составе файла архива iomega_usb_firewire_dos_driver_boot_disk.zip. Там же приведен пример строки загрузки этого драйвера, но без объяснения назначения параметров. Версия 1.02 другого драйвера – SBP2ASPI.SYS фирмы Medialogic – содержится в самораспаковывающемся файле архива DAT.EXE, который свободно выложен на сайте http://www.datoptic.com/fw25fr.html .

Поскольку оба упомянутых драйвера дают возможность обращаться к подключенным приборам посредством унифицированного набора команд ASPI, постольку для обслуживания накопителей и дисководов на шине IEEE1394 потенциально пригодны те же драйверы, которые обращаются посредством команд ASPI к таким же накопителям и дисководам с интерфейсами SCSI (5.07-03) и USB (5.07-05). Но надо иметь ввиду, что все сведения о драйверах шины IEEE1394 приведены здесь без экспериментальных проверок, только для уведомления.

5.07-05 Драйверы интерфейса USB

Универсальная последовательная шина USB (= Universal Serial Bus) разработана совместно фирмами Compaq, Intel, Microsoft и NEC. Спецификация USB 1.0 была принята ими в 1996 году. С тех пор встроенный USB-контроллер стал привычным элементом на материнских платах всех массовых AT-совместимых компьютеров. Практическое значение интерфейса USB еще более возросло с принятием в 2002 году спецификации USB 2.0, расширившей диапазон скоростей передачи данных до 480 Мегабит/с. Сейчас подавляющее большинство выпускаемых периферийных приборов рассчитано на подключение посредством интерфейса USB.

Контроллеры шины USB реализуют три разных варианта взаимодействия с аппаратными средствами компьютера:

"Открытый" – Open Host Controller Interface (OHCI),
"Унифицированный" – Universal Host Controller Interface (UHCI),
"Усовершенствованный" – Enhanced Host Controller Interface (EHCI).

Все современные USB-контроллеры соответствуют спецификации USB 2.0 и реализуют вариант EHCI, но способны эмулировать более старые контроллеры, соответствующие спецификации USB 1.1, которые поддерживают либо вариант OHCI, либо вариант UHCI. Разница состоит в том, что в варианте UHCI обращения происходят как обычно – через порт, а в варианте OHCI – через выделенную зону памяти. Вариант OHCI реализован комплектами микросхем фирм SIS и ALi, а вариант UHCI – комплектами микросхем фирм Intel, VIA и ряда других.

Доступ из среды DOS к периферийным приборам, подключаемым к шине USB, возможен либо посредством системы BIOS компьютера, либо посредством загружаемых драйверов. Доступ посредством BIOS необходим, когда предстоит загружать операционную систему с внешнего накопителя. Более детально этот вопрос рассмотрен в разделе 9.11-01. Однако системы BIOS не обязательно готовы обслуживать любой внешний прибор: некоторые воспринимают только определенные виды накопителей, например, флоппи-дисководы с интерфейсом USB. Кроме того, системы BIOS фиксируют свойства конкретного внешнего накопителя в момент включения компьютера и не позволяют их менять. Если, например, первоначально вставленную в адаптер флэш-карту заменить на другую, то доступа к этой другой карте не будет. Возможность оперативной смены внешних накопителей обеспечивается с помощью загружаемых драйверов.

Чтобы избежать вероятного конфликта драйверов с системой BIOS, в настройках программы BIOS Setup на страничке "Advanced" параметр "Legacy USB Support" должен быть поставлен в положение "Disabled" (= выключено). Если в настройках программы BIOS Setup Вашего компьютера подобный параметр отсутствует, то, скорее всего, данная система BIOS не обслуживает внешние USB-накопители. В таких компьютерах доступ к приборам на шине USB возможен только посредством загружаемых драйверов. В общем случае нужно сначала загрузить драйвер USB-контроллера, а затем – драйверы всех тех периферийных приборов, которыми предстоит пользоваться.

В 1998 – 2001 годах фирмой SoftConnex были разработаны первые два драйвера для USB-контроллеров – UHCI.EXE и OHCI.EXE. Их версии 2.3 используются на загрузочных дискетах, формируемых известным пакетом программ Norton Ghost (до 8-й версии включительно). Эти же драйверы версии 2.5 выложены на сайте http://www.stefan2000.com/darkehorse/PC/DOS/Drivers/USB/ . Их нужно загружать последовательно, один за другим, но фактически загрузится только один из них – тот, который соответствует варианту взаимодействия, реализуемому USB-контроллером. Возможна загрузка прямо из командной строки, из строк файла AUTOEXEC.BAT командой LH (3.17), или из строк файла CONFIG.SYS командами DEVICE (4.06) или DEVICEHIGH (4.07), например:

```
DEVICEHIGH = \DOS\DRV\UHCI.EXE
DEVICEHIGH = \DOS\DRV\OHCI.EXE
```

Драйверы фирмы SoftConnex предназначены для обслуживания USB-клавиатур, а также манипуляторов типа "мышь" и внешних портов, подключаемых к шине USB. Разумеется, затем необходимо загрузить драйвер обслуживаемого прибора. В частности, для манипуляторов типа "мышь" сгодится любой из драйверов, описанных в разделе 5.03. Но нужно иметь ввиду, что обслуживаемый прибор должен быть подключен обязательно к первому USB-контроллеру, потому что всех остальных USB-контроллеров, которыми бывает оборудован современный компьютер, драйверы фирмы SoftConnex "не видят". Кроме того, драйверы фирмы SoftConnex не предоставляют услуг, которые обеспечили бы доступ к картам памяти, к магнитным или оптическим дискам.

Проблема доступа из DOS к дисковым и другим накопителям, подключаемым к шине USB, сейчас особенно актуальна. На эту тему в сети Интернет имеется много разнородных сведений. Чтобы составить собственное мнение, автору пришлось предпринять ряд экспериментов, в которых роль подопытного дисковода играл USB-адаптер типа ImageMate-2 для карт Compact Flash.

Лучшим драйвером для USB-контроллеров оказался USBASPI.SYS фирмы Matsushita (торговая марка Panasonic). Версия 2.27 этого драйвера (2008 года) содержится в самораспаковывающемся файле архива F2H_USB.EXE, который выложен на сайте http://panasonic.co.jp/pcc/products/drive/other/f2h_usb.html . Драйвер USBASPI.SYS обслуживает все типы USB-контроллеров, причем по умолчанию сканирует шину каждого USB-контроллера, если их в компьютере несколько, и регистрирует все действующие номера LUN (примечание 1 к A.03-2) у каждого подключенного устройства. В отличие от большинства других драйверов USB, версия 2.27 этого драйвера не конфликтует с системой BIOS даже когда параметр "Legacy USB support" активизирован. Полной спецификации параметров драйвера USBASPI.SYS фирма Matsushita не предоставляет. Тем не менее установлено назначение следующих необязательных параметров:

/e – активизировать USB-контроллеры варианта EHCI.
/o – активизировать USB-контроллеры варианта OHCI.
/u – активизировать USB-контроллеры варианта UHCI.
/nocbc – не искать USB-адаптеры в слотах PCMCIA (PcCard).
/w – напомнить пользователю, чтобы он подключил устройства, которые должны быть зарегистрированы на шине USB.
/slow – снизить темп опроса подключенных устройств, чтобы медленные считыватели карт успели ответить.
/v – показать сведения об обнаруженных USB-контроллерах и устройствах, подключенных к шинам USB.
/r – не прерывать загрузку драйвера из-за ошибок, а также когда USB-контроллером управляет система BIOS.
/norst – не посылать USB-устройствам сигнал RESET, чтобы не прерывать загрузку компьютера с одного из них.

Параметры /e, /o, /u позволяют сократить время поиска, если заранее известно, каким USB-контроллером предстоит пользоваться в конкретном компьютере. Последние два параметра (/r и /norst) необходимы при осуществлении загрузки компьютера с одного из USB-устройств, так как иначе либо процесс загрузки будет прерван, либо другие USB-устройства останутся недоступными после завершения загрузки. Кроме того, при пользовании драйвером USBASPI.SYS нельзя прежде него загружать драйверы оптических дисководов, даже когда они не имеют отношения к шине USB.

Лучшим драйвером для не-оптических накопителей продолжает оставаться ASPIDISK.SYS фирмы Adaptec, первоначально разработанный для интерфейса SCSI. Но прямого отношения к типу интерфейса он не имеет, потому что обращается к накопителям опосредованно, используя стандартизованный набор команд ASPI. Поскольку именно этот набор команд реализован драйвером USBASPI.SYS на базе интерфейса USB, постольку драйвер ASPIDISK.SYS обслуживает накопители на картах памяти и на жестких магнитных дисках, подключаемые к шине USB.

Версию 4.01b драйвера ASPIDISK.SYS (длина 15060 байтов, дата 02.12.1998) можно скачать из сети интернет в составе файла архива DOSDRVR.EXE с сайта фирмы Adaptec http://www.adaptec.com/en-US/speed/scsi/dos/dosdrvr_exe.htm . Этот драйвер обеспечивает доступ к накопителям с файловыми системами FAT-12, FAT-16, FAT-32 и "Big Floppy". Если сменный носитель в накопитель не вставлен, то по умолчанию ему будет выделено буквенное обозначение одного логического диска. Но носители, отформатированные как жесткий магнитный диск с несколькими разделами FAT-16, представляют несколько логических дисков. Чтобы на таких носителях получить доступ к остальным логическим дискам, помимо первого, необходимое количество дополнительных буквенных обозначений следует заблаговременно резервировать посредством указания параметра /r в командной строке. Полный набор принимаемых необязательных параметров включает следующее:

/id=2:0+1 – пример списка номеров устройств, которые драйверу следует взять под свое управление: устройство номер 2 на шине 1-го USB-контроллера и устройства номер 0 и 1 на шине 2-го USB-контроллера.

/nospin – не выдавать в момент инициализации команду на включение мотора привода дисков.

/d – показать сведения о накопителях, которые драйвер взял под свое управление.

/pause – сделать паузу, чтобы дать возможность прочитать выведенное сообщение.

/r4 – пример резервирования четырех буквенных обозначений для дополнительных логических дисков. Допускается резервировать от 1 до 24 буквенных обозначений.

Если список номеров не указан, то драйвер ASPIDISK.SYS будет разбираться с каждым USB-устройством и постарается взять под свое управление все подходящие устройства, причем не требуя наличия сменного носителя в них в момент загрузки. Наличие списка номеров устройств предотвратит потери времени на разборки с неподходящими устройствами. В приведенном примере указание устройства номер 2 на шине 1-го USB-контроллера означает, что драйверу предписано проигнорировать устройства с номерами 0 и 1, которые могут быть, например, сканером и принтером.

Из драйверов оптических дисководов CD/DVD-ROM, видимо, подойдет NJUSBCDA.SYS фирмы Workbit Corp. Версия 3.9 этого драйвера (2000 года) выложена на сайте http://www.driver.novac.co.jp/driver/sta_black/bst_drv.html в составе файла архива BST_DOS.ZIP. Как многие другие драйверы оптических дисководов, NJUSBCDA.SYS принимает в командной строке параметр /D: с последующим произвольным идентификатором длиной не более 8 знаков (например, /d:USBCD001) для опознания программой MSCDEX.EXE (5.08-03) или программой SHSUCDEX.EXE (5.08-04), одна из которых потом тоже должна быть запущена. В строку ее запуска надо будет добавить точно такой же параметр с точно тем же идентификатором.

Описанные драйверы USB-устройств следует загружать из строк файла CONFIG.SYS командами DEVICE (4.06) или DEVICEHIGH (4.07). Если предположить, что все драйверы USB-устройств находятся в каталоге \DOS\DRV того диска, с которого производится загрузка компьютера, то строки их загрузки могут выглядеть, например, так:

```
devicehigh=\DOS\DRV\Usbaspi.sys /slow /v
devicehigh=\DOS\DRV\Aspidisk.sys /nospin /d /pause
devicehigh=\DOS\DRV\Njusbcda.sys /d:USBCD001
```

Параметры в приведенном примере подобраны в расчете на загрузку компьютера с устройства, не имеющего отношения к шине USB, причем когда заранее нет сведений о числе имеющихся USB-контроллеров и других подключенных устройств. Если же загрузка выполняется не первый раз в компьютере с заранее известным составом оборудования, то параметры /slow и /pause бывает целесообразно убрать. Кроме того, для сокращения времени поиска полезно указать, какого типа USB-контроллер следует искать и какие именно номера устройств на его шине нужно регистрировать.

Упоминаемую выше версию 2.27 драйвера USBASPI.SYS фирмы Matsushita (длина файла 39729 байт) часто путают с одноименным драйвером версии 1.07

фирмы Novac (длина файла 43528 байт), который очень напоминает самые ранние версии того же драйвера фирмы Matsushita. Он обслуживает только первый USB-контроллер, только согласно спецификации USB 1.1, игнорирует устройства с ненулевым номером LUN (примечание 1 к A.03-2), и принимает из командной строки другой набор необязательных параметров:

/w – напомнить пользователю, чтобы он подключил устройства, которые должны быть зарегистрированы на шине USB.

/v – показать сведения об обнаруженных устройствах, подключенных к шине USB 1-го контроллера.

/r – не выгружать резидентный модуль драйвера, даже если USB-контроллер находится под управлением BIOS.

/m=D0 – пример спецификации зоны памяти для USB-контроллера варианта OHCI. D0 означает зону D0000h – DFFFFh.

/p=A400 – пример адреса порта USB-контроллера варианта UHCI.

Драйвер USBASPI.SYS фирмы Novac можно свободно скачать из сети Интернет с сайта http://www.driver.novac.co.jp/driver/hd352u/hd352u_drv.html в составе файла архива HD352u_dos.zip. В том же файле архива, помимо USBASPI.SYS, содержится еще драйвер фирмы Novac Di1000dd.sys версии 2.00 для доступа к не-оптическим накопителям. Он принимает из командной строки параметр

/dS – пример буквы ("S"), которую следует назначить диску, доступному посредством данного драйвера.

Однако назначение заданной буквы выполняется "грязно", без дезактивирования фантомных дисков. Драйвер Di1000dd.sys "не понимает" номера LUN и файловую систему FAT-32, регистрирует накопители только на шине первого USB-контроллера, причем при условии, что они в момент загрузки содержат носитель записи. Ограниченные способности комплекта драйверов фирмы Novac все же могут быть достаточны для выполнения основных операций на компьютерах с известным составом оборудования.

Еще один путь доступа к USB-устройствам из DOS открывает комбинированный драйвер DUSE.EXE, разработанный фирмой Cypress Semiconductor. Версию 4.9 этого драйвера (2003 года) можно свободно скачать из сети Интернет с сайта http://www.pocketec.net/support.taf в подкаталоге "downloads" в составе файла архива Duse_4_9.zip. Файл DUSE.EXE объединяет в себе драйверы USB-контроллера, оптических дисководов, а также накопителей на жестких магнитных дисках и на картах памяти.

Возможность получить "все в одном" выглядит очень привлекательно, но сопровождается неприятными неожиданностями. По непонятной причине DUSE.EXE требует, чтобы DOS работала в реальном режиме, то есть без драйвера EMM386.EXE (5.04-02). Но тогда теряется доступ к области UMB, и DOS вынуждена загружать все остальные драйверы в обыкновенную память. Хуже того,

при загрузке с установками по умолчанию (без параметров) резидентный модуль DUSE.EXE занял еще 233 кбайт обыкновенной памяти, так что там вообще не осталось достаточного пространства для пользовательских программ.

Минимально-приемлемую конфигурацию удалось получить, отказавшись от загрузки модуля поддержки оптических дисководов и обеспечив загрузку большинства драйверов в область UMB в реальном режиме с помощью драйвера UMBPCI.SYS (5.04-04). Объем резидентного модуля DUSE.EXE уменьшился до 153 кбайт, но все попытки разместить его в области UMB кончились неудачно. Для сравнения: резидентные модули драйверов USBASPI.SYS и ASPIDISK.SYS предоставляют примерно те же возможности, но вместе занимают всего 45 кбайт и размещаются в области UMB без каких-либо требований к режиму работы DOS. Если все-таки возникнет необходимость воспользоваться драйвером DUSE.EXE, то рекомендуемая строка его загрузки в файле CONFIG.SYS может выглядеть, например, так:

```
device=\DOS\DRV\Duse.exe NOC EMU XFER=8
```

Показанные здесь необязательные параметры означают:

NOC – исключить поддержку дисководов CD-ROM.

EMU – эмулировать вызовы по линиям IRQ для предотвращения взаимной несовместимости драйверов.

XFER=8 – уменьшить объем буфера до 8 кбайт. По умолчанию принимается 64, допускается от 1 до 64 кбайт.

Полный перечень параметров драйвера DUSE.EXE приведен в файле DUSEUsersGuide.pdf, содержащемся в том же архиве Duse_4_9.zip. Но надо отметить, что упоминаемый там параметр INT в экспериментах автора не обеспечивал возможность проводить разметку дисков на шине USB с помощью программы FDISK.EXE (6.13). Более того, разметка посредством команд ASPI (примечание 5 к 6.13) была невозможна независимо от наличия параметра INT, тогда как при доступе с помощью драйвера USBASPI.SYS проблемы с такой разметкой не возникали.

В последнее время стали известны две персональные попытки написать комплекты USB-драйверов для DOS. В декабре 2006 года Г.Потхаст выложил версию 1.0 своего комплекта на сайте http://www.georgpotthast.de/usb/ . Позднее, в августе 2009 года, на сайте http://bretjohnson.us/ появилась версия 0.08 другого комплекта драйверов, написанного Б.Джонсоном.

Комплект Г.Потхаста содержит драйвера для USB-контроллера, для некоторых типов принтеров, для не-оптических USB-накопителей, а также ряд служебных утилит, позволяющих исправлять ошибки конфигурирования USB-интерфейсов. К сожалению, эти драйвера не могут сами определять число обслуживаемых дисков, требуют наличия носителя в накопителе в момент загрузки, обслуживают только

первый раздел одного диска только через первый USB-контроллер и имеют другие существенные ограничения.

Комплект Б.Джонсона, помимо аналогичных основных драйверов и служебных утилит, содержит драйвера USB-клавиатуры и USB-мыши. Кроме того, он предоставляет доступ к дискам с файловой системой FAT-32. Однако его нельзя использовать с контроллерами OHCI-типа, и скорость передачи выше 12 Мб/с не обеспечивается. Возможно, не все недостатки выявлены, так как этот комплект стал доступен в последний момент, и его полное тестирование не проводилось.

Несмотря на все недостатки ранних версий, комплекты драйверов Г. Потхаста и Б.Джонсона заслуживают серьезного внимания. Они оба находятся в процессе разработки, и их будущие версии могут оказаться намного лучше. Но самое ценное достоинство этих комплектов драйверов – частично открытый программный код. В нем найдется много полезного для тех, кому предстоит более детально разбираться в особенностях интерфейса USB.

5.08 Службы устанавливаемых файловых систем

5.08-01 Вспомогательный драйвер IFSHLP.SYS

Вспомогательный драйвер IFSHLP.SYS предоставляет сервисные функции для обслуживания устанавливаемых файловых систем IFS (= Installable File Systems) и обеспечения доступа к ним. Файловые системы IFS позволяют скрыть от пользователя реальные структуры данных и реальные пути доступа, чтобы не показывать технические подробности и осуществлять доступ избирательно, в соответствии с данными пользователю правами.

Обработчики прерывания INT 13, поставляемые системой BIOS компьютера, исполняют 16-битовые обращения к дисководам в реальном режиме и не способны действовать опосредованно, например, через сеть. При работе в защищенном режиме перед каждым вызовом INT 13 приходится переходить в реальный режим, а потом возвращаться обратно. Эти переходы делают медленный 16-битовый доступ еще более медленным. Сервисные функции, предоставляемые драйвером IFSHLP.SYS, обеспечивают гораздо более быстрый 32-битный доступ как к локальным, так и к сетевым накопителям без лишних переходов при работе в защищенном режиме. Если же предстоит работать только в реальном режиме и без сети, то, скорее всего, драйвер IFSHLP.SYS не потребуется.

Драйвер IFSHLP.SYS входит в пакет программ Microsoft Network Client, а также в комплекты поставки операционных систем Windows-3.11/95/98/ME. MS-DOS7 ищет файл IFSHLP.SYS в каталоге \Windows и загружает его по умолчанию, если

только загрузка по умолчанию не отменена командой DOS=NOAUTO (4.08) в файле CONFIG.SYS.

5.08-02 Драйвер NTFSDOS.EXE для доступа к дискам NTFS

Возможность доступа из среды DOS к дискам с файловой системой NTFS предоставляет драйвер NTFSDOS.EXE, написанный Марком Руссиновичем (Mark Russinovich) и Брюсом Когсвеллом (Bruce Cogswell). Полнофункциональные версии этого драйвера (4-я и 5-я) были не бесплатны. Ниже будет описана свободно распространяемая версия 3.02 от 2001 года, которая обеспечивает только считывание файлов с дисков NTFS, включая считывание программ в память для исполнения. К сожалению, драйвер NTFSDOS и ряд полезных программ исчезли с собственного сайта авторов в конце 2006-го года, когда авторы были приняты в команду разработчиков фирмы Microsoft. Сейчас версию 3.02 драйвера NTFSDOS.EXE можно скачать из сети Интернет в составе архива Ntfs30r.zip, например, с сервера ftp://ftp.uni-koeln.de/pc/msdos/diskutils/ или с архивного сайта http://web.archive.org/web/20020123013310/www.sysinternals.com/new.shtml .

Здесь уместно напомнить, что программа установки (SETUP) операционных систем Windows-2000/XP, запускаемая с компакт-диска, в качестве одной из альтернатив позволяет открыть так называемую восстановительную консоль (Recovery Console), которая обеспечит запись на диски NTFS, но по умолчанию не даст считывать файлы на дискету. Драйвер NTFSDOS.EXE версии 3.02 поможет сделать именно то, что запрещено делать в среде Recovery Console.

Драйвер NTFSDOS.EXE пользуется XMS-памятью и потому требует, чтобы заранее был загружен драйвер HIMEM.SYS (5.04-01). Помимо того, драйвер NTFSDOS.EXE занимает весьма много места в обыкновенной памяти. В частности, для обслуживания диска NTFS объемом 10 Гбайт драйвер "забирает" до 285 кбайт обыкновенной памяти. При таких запросах постоянно держать этот драйвер в памяти нецелесообразно. Поэтому его загружают из командной строки непосредственно перед моментом доступа к диску NTFS, например, так:

```
C:\DOS\DRV\NTFSDOS.EXE /L:K /C:1024 /N /X /U /V
```

здесь:

C:\DOS\DRV\ – пример пути к файлу NTFSDOS.EXE

/L:K – необязательный параметр, предписывающий присвоить диску NTFS буквенное обозначение "K". Следующим дискам NTFS, если они имеются, будут назначены следующие по порядку буквенные обозначения. Если параметр /L: не указан, то

обозначения будут назначены начиная с первой свободной буквы.

/C:1024 – необязательный параметр, предписывающий создать в XMS-памяти кэш-буфер объемом 1024 кбайт. Если параметр /C: не указан, то в XMS-памяти по умолчанию будет создан кэш-буфер объемом 500 кбайт.

/N – необязательный параметр, предписывающий не загружать модуль восстановления сжатых фрагментов файловой системы NTFS (он не нужен, если сжатие не применено). Отказ от доступа к сжатым фрагментам дает некоторое уменьшение объема занимаемой памяти.

/X – необязательный параметр, предписывающий не пользоваться расширенными функциями прерывания INT 13 (8.01-55). Параметр /X полезен при работе на компьютерах, выпущенных до 1996 года и снабженных дисководом емкостью менее 8.4 Гбайт.

/U – необязательный параметр, обеспечивающий трансляцию имен файлов и каталогов, написанных универсальным кодом (unicode, по два байта на знак).

/V – необязательный параметр, предписывающий при загрузке выводить на экран не только сведения о приписанных дискам буквенных обозначениях, но также сведения об используемой драйвером памяти.

Примечание 1: драйвер NTFSDOS.EXE загружает в память обработчики прерываний, обеспечивающие просмотр "длинных" имен файлов на дисках NTFS. Ими могут воспользоваться все программы, которые способны к ним обращаться, в том числе файл-менеджер Volcov Commander (6.25). Тем не менее при копировании "длинные" имена урезаются.

Примечание 2: попытки доступа к дискам с поврежденной файловой системой NTFS могут вызвать "зависание" компьютера. Рекомендуется заранее проверить и при необходимости исправить файловую систему с помощью программы CHKDSK в среде Recovery Console.

5.08-03 Программа MSCDEX.EXE для доступа к оптическим дискам

MSCDEX.EXE – это резидентная программа, расширяющая функции ядра DOS в части взаимодействия с драйверами оптических дисководов, создания соответствующих им логических дисков и обеспечения доступа к ним. Фактически MSCDEX.EXE играет роль переводчика применяемых в оптических дисках

файловых систем "High Sierra" и ISO 9660, поскольку они отличаются от тех файловых систем, которые "понятны" ядру DOS.

Программа MSCDEX.EXE входит в комплект поставки операционных систем Windows-95/98 и при их стандартной установке находится в каталоге \WINDOWS\COMMAND. Загружать программу MSCDEX.EXE нужно после всех драйверов имеющихся оптических дисководов, но раньше драйвера SMARTDRV.EXE (5.06-01), если Вы намерены его использовать. Обычно программу MSCDEX.EXE загружают из файла CONFIG.SYS командами INSTALL (4.15) или INSTALLHIGH (4.16), но ее также можно запускать из файла AUTOEXEC.BAT командой LH (3.17) или прямо из командной строки:

```
C:\DOS\DRV\MSCDEX.EXE /D:MSCD001 /e /k /s /v /L:N /M:12
```

здесь:

C:\DOS\DRV\ – пример пути к файлу MSCDEX.EXE.

/D:MSCD001 – пример объявления произвольно задаваемого идентификатора "MSCD001" для опознания соответствующего драйвера оптического дисковода. Этот драйвер к данному моменту должен быть загружен с тем же идентификатором. Если, помимо того, загружены другие драйверы оптических дисководов, то они должны иметь другие идентификаторы, и каждый из них представляется отдельным параметром /D: в той же строке после имени программы MSCDEX.EXE.

/e – необязательный параметр, определяющий предпочтительное размещение буферов за пределами обыкновенной памяти, при условии, что доступ туда уже обеспечен драйвером EMM386.EXE (5.04-02).

/k – необязательный параметр, определяющий предпочтительное использование дополнительного дескриптора тома, написанного японской двухбайтовой нотацией (Kanji), если этот дополнительный дескриптор будет найден. При отсутствии параметра /k поиск дополнительного дескриптора не производится, используется только первичный дескриптор.

/s – необязательный параметр, подготавливающий резидентный модуль к последуюшей загрузке сетевого программного обеспечения, чтобы потом избежать конфликтов при назначении букв дисков и обеспечить возможность сетевого доступа к оптическим дискам.

/v – необязательный параметр, вызывающий выведение на экран сообщения о статусе оптических дисководов.

/L:N – необязательный параметр, задающий назначение буквы (в данном примере N:) первому из дисководов, обслуживаемых драйвером с идентификатором, указанным после первого

параметра /D: (в данном примере MSCD001). Следующим дисководам, если такие имеются, будут назначены следующие буквы O: , P: и т.д. Если же параметр /L не указан, то буквы назначаются по порядку, начиная с первой свободной буквы. Однако MSCDEX.EXE не может превысить предел, определенный командой LASTDRIVE (4.17) в файле CONFIG.SYS.

/M:12 – пример резервирования 24 кбайт памяти для создания 12 буферов по 2048 байт каждый с целью увеличения скорости доступа. Допускается задавать от 4 до 64 буферов, по умолчанию принимается 12 буферов.

Примечание 1: резидентный модуль программы MSCDEX.EXE взаимодействует с другими программами через прерывания INT 2F\AX=1500-150Fh (8.03-13 – 8.03-19).

5.08-04 Программа SHSUCDX.COM для доступа к оптическим дискам

В последние годы приняты несколько модификаций файловых систем, допускающие использование "длинных" имен файлов и каталогов на оптических дисках. Однако программа MSCDEX.EXE не справляется с модифицированными вариантами файловых систем: она показывает файлы с "обрезанными" длинными именами, но не обеспечивает доступа к ним. Этим недостатком не обладает аналогичная по назначению свободно распространяемая программа SHSUCDX.COM, разработка которой была начата Джоном Маккоем (John H. McCoy) и продолжена Джейсоном Худом (Jason Hood). Версию 3.02 программы SHSUCDX.COM (2005 года) можно свободно скачать из сети Интернет с сайта http://www.shsucdx.adoxa.cjb.net/ в составе файла архива shcdx302.zip.

Программу SHSUCDX.COM можно запускать из файла AUTOEXEC.BAT командой LH (3.17) или прямо из командной строки, но обычно ее загружают из файла CONFIG.SYS командами INSTALL (4.15) или INSTALLHIGH (4.16), например:

```
installhigh=\DOS\DRV\Shsucdx.com /D:?CD001,N,0,2 /~+ /R /Q
```

здесь

\DOS\DRV\ – пример пути к файлу SHSUCDX.COM

/D:?CD001 – пример объявления произвольно задаваемого идентификатора "CD001" для опознания драйвера оптического дисковода. Тот же идентификатор должен быть указан в строке загрузки драйвера. Если возможна загрузка нескольких драйверов оптических дисководов, то остальным должны быть даны другие

идентификаторы, представленные отдельными параметрами /D: после идентификатора первого драйвера. Знак вопроса перед идентификатором отмечает драйвера, которые могут быть не задействованы, например, при загрузке на компьютер с не известным заранее составом оборудования.

,N,0,2 – группа необязательных спецификаций для обслуживания драйвера, к которому относится данный параметр /D:. Первая буква (в данном случае "N") – это буквенное обозначение, которое следует назначить дисководу. Вторая цифра (по умолчанию "0") – номер дисковода, которому будет назначено буквенное обозначение, в списке дисководов, представляемых данным драйвером. Третья цифра ("2") – число дисководов, которое будет принято на обслуживание от данного драйвера. По умолчанию будут приняты все дисководы, причем остальным будут назначены буквы, следующие за указанной. Если букву не задавать, то буквенные обозначения будут назначены по порядку, начиная с первой свободной буквы.

/~+ – необязательный параметр, предписывающий использовать знак тильда (~) в именах, длина которых сокращена до принятых в DOS 8 знаков. Действие этого параметра может быть потом отменено посредством запуска программы SHSUCDX.COM на исполнение из командной строки с обратным параметром /~– .

/R – необязательный параметр, предписывающий сбрасывать атрибут "Только для чтения" с файлов, копируемых с незаписываемых оптических дисков.

/Q – необязательный параметр, предписывающий не выводить на экран ничего, кроме назначенных дискам буквенных обозначений. Чтобы вообще запретить выведение сообщений, нужно указать параметр /QQ.

Помимо перечисленных выше параметров, программа SHSUCDX.COM допускает указывать следующие:

/L:N – альтернативное предписание буквенного обозначения ("N") для первого из принимаемых на обслуживание оптических дисководов (остальным будут назначены следующие по порядку буквы). Параметр /L: тождественен одноименному параметру программы MSCDEX.EXE (5.08-03), причем его можно использовать только тогда, когда дополнительные спецификации к параметру /D: не указаны.

/C – загружать программу SHSUCDX.COM в обыкновенную память (по умолчанию она оставляет свой резидентный модуль в блоках UMB, если они доступны).

/V – вывести на экран дополнительные сведения. С этим параметром программу SHSUCDX.COM можно запускать отдельно из командной строки после загрузки. Очевидно, параметр /V несовместим с параметрами /Q и /QQ.

/U – выгрузить из памяти резидентный модуль программы SHSUCDX.COM. Естественно, запускать программу SHSUCDX.COM с параметром /U следует только из командной строки при условии, что ее резидентный модуль был заранее загружен. После выгрузки занимаемая память будет освобождена, а диски, которые обслуживал резидентный модуль, станут недействительными.

Примечание 1: пересылки данных с оптических дисководов, организованные программой SHSUCDX.COM, не обслуживаются кэш-драйвером SMARTDRV.EXE (5.06-01), но обслуживаются кэш-драйвером UIDE.SYS (примечание 3 к разделу 5.06-01).

5.09 Драйверы оптических дисководов

Оптические дисководы различаются по типу интерфейса, посредством которого они подключаются к компьютеру. Если в одном компьютере имеются оптические дисководы, подключенные через разные интерфейсы, то для их обслуживания, как правило, необходимо загружать отдельные драйверы. Здесь, в разделе 5.09, представлены драйверы для оптических дисководов с интерфейсом IDE (ATA), получивших наибольшее распространение в современных AT-совместимых компьютерах.

На материнских платах компьютеров обычно имеются два IDE-контроллера, причем они обслуживают не только оптические дисководы, но и накопители на жестких магнитных дисках. Чтобы не снижать скорость пересылки данных, желательно подключать оптический дисковод к отдельному IDE-контроллеру. Разумеется, этот контроллер должен быть активизирован ("Enabled") установлением соответствующего флажка в программе BIOS Setup.

Программы BIOS Setup современных компьютеров позволяют установить несколько режимов работы IDE-контроллеров, в том числе с опросом только последовательной шины ATA (S-ATA). Поскольку оптические дисководы обычно подключаются посредством параллельной шины ATA (P-ATA), постольку для обслуживания оптического дисковода контроллером должен быть установлен совместимый режим ("Compatible Mode"). Если программа BIOS Setup позволяет изменять режим ускоренного прямого доступа к памяти (UltraDMA), то следует установить обычный IDE-режим ("Legacy IDE mode"). Перечисленные режимы

обеспечат правильное функционирование описываемых ниже драйверов оптических дисководов в среде DOS.

5.09-01 Драйвер OAKCDROM.SYS для дисководов CD-ROM

Драйвер OAKCDROM.SYS разработан фирмой Oak Technology Inc. и предназначен для обслуживания оптических дисководов, подключаемых к стандартным IDE-контроллерам. Файл OAKCDROM.SYS входит в состав поставки операционных систем WINDOWS-95/98. В загрузочных дискетах к операционной системе WINDOWS-98 используется версия OAKCDROM.SYS, датированная 1997-м годом. Эту же версию драйвера можно скачать из сети Интернет с сайта http://www.computerhope.com/download/hardware.htm#02 .

На современных компьютерах, снабженных DVD-дисководом, драйвер OAKCDROM.SYS обеспечивает просмотр содержания и считывание файлов с дисков CD и DVD. Но в большинстве компьютеров, выпущенных до 2005 года, система BIOS не поддерживает взаимодействие IDE-контроллеров с DVD-дисководами, и потому обеспечивается доступ только к дискам CD. В таких компьютерах доступ к дискам DVD тоже возможен, но для этого необходимо заранее, до драйвера OAKCDROM.SYS, загрузить драйвер ATAPIMGR.SYS (5.07-01).

Загружать драйвер OAKCDROM.SYS нужно командами DEVICE (4.06) или DEVICEHIGH (4.07) из строки файла CONFIG.SYS, например:

```
DEVICEHIGH=\DOS\DRV\OAKCDROM.SYS /D:MSCD001 /V
```

здесь:

\DOS\DRV\ – пример пути к драйверу OAKCDROM.SYS

/D:MSCD001 – параметр /D: объявляет произвольный идентификатор длиной до 8 знаков для опознания драйвера программой MSCDEX.EXE (5.08-03) или программой SHSUCDX.COM (5.08-04). В строке запуска этих программ должен быть указан такой же параметр /D: с точно тем же идентификатором.

/V – необязательный параметр, вызывающий вывод на экран сообщения о загрузке драйвера.

Драйверу OAKCDROM.SYS не нужно указывать, куда конкретно подключен оптический дисковод, он осуществляет поиск оптических дисководов по IDE-контроллерам, имеющим типовые значения адреса порта и номера линии IRQ запроса прерывания (во всяком случае, по портам 1F0h и 170h). Если в компьютере имеются несколько оптических дисководов, то драйвер OAKCDROM.SYS будет обслуживать все оптические дисководы, какие он сможет найти.

5.09-02 Драйвер VIDE-CDD.SYS для дисководов CD-ROM

Драйвер VIDE-CDD.SYS версии 2.14, разработанный фирмой Acer Co в 1998 году, функционально близок к драйверу OAKCDROM.SYS (5.09-01), но существенно более компактен (всего 11,8 кбайт) и позволяет указывать в командной строке номера задействуемых портов и линий IRQ: это важно, когда надо избежать поиска, а также когда используются нестандартные спецификации подключения. Самораспаковывающийся архив Apicd214.exe, содержащий драйвер VIDE-CDD.SYS, можно скачать из сети интернет, например, с сервера ftp://ftp.benq.co.uk/cd-rom/drivers/apicd214.exe .

Если система BIOS компьютера не поддерживает DVD-дисководы, то для обеспечения доступа к дискам DVD посредством драйвера VIDE-CDD.SYS необходимо заранее, до него, загрузить драйвер ATAPIMGR.SYS (5.07-01). Драйвер VIDE-CDD.SYS следует загружать командами DEVICE (4.06) или DEVICEHIGH (4.07) из строки файла CONFIG.SYS, например:

```
DEVICEHIGH=\DOS\DRV\Vide-cdd.sys /D:MSCD001 /P:170,15
```

здесь:

\DOS\DRV\ – пример пути к драйверу VIDE-CDD.SYS.

/D:MSCD001 – параметр /D: объявляет произвольный идентификатор длиной до 8 знаков для опознания драйвера программой MSCDEX.EXE (5.08-03) или программой SHSUCDX.COM (5.08-04). В строке запуска этих программ должен быть указан такой же параметр /D: с точно тем же идентификатором.

/P:170,15 – необязательная спецификация базового адреса порта и линии IRQ запроса прерывания, используемых IDE-контроллером, к которому подключен оптический дисковод.

В одной строке можно указывать несколько параметров /P: с различными спецификациями для нескольких IDE-контроллеров. Если хотя бы один параметр /P: указан, то обращаться по не указанным адресам драйвер не будет. Если ни одного параметра /P: в строке нет, то драйвер будет производить поиск по всем типовым спецификациям портов, используемых IDE-контроллерами: /P:1F0,14 , /P:170,15 , /P:1E8,12 , /P:168,10 (A.14-1). Драйвер VIDE-CDD.SYS возьмет под свое управление все оптические дисководы, какие он сможет найти.

5.09-03 Драйвер QCDROM.SYS для дисководов CD/DVD-ROM.

Драйвер оптических дисководов QCDROM.SYS версии 4.2 разработан в 2007 году Джеком Иллисом (Jack R. Ellis) на основе своего же более раннего драйвера XCDROM.SYS. В отличие от последнего, драйвер QCDROM.SYS обеспечивает

доступ не только к дискам CD, но и к дискам DVD, причем не нуждаясь в предварительной загрузке драйвера ATAPIMGR.SYS. Драйвер QCDROM.SYS способен обслужить до трех оптических дисководов, подключенных не более чем к двум стандартным IDE-контроллерам, которые пользуются стандартными адресами портов и линиями запроса прерывания (1F0h с IRQ 14 и/или 170h с IRQ 15).

Драйвер QCDROM.SYS выложен в сети Интернет, и его можно свободно скачать с сайта http://cyberia.dnsalias.com/Cyb.05.Htm в составе файла архива QCDROM42.ZIP. Загружать драйвер следует командами DEVICE (4.06) или DEVICEHIGH (4.07) из строки файла CONFIG.SYS, например:

```
DEVICEHIGH=\DOS\DRV\Qcdrom.sys /D:MSCD001 /L
```

Здесь:

\DOS\DRV\ – пример пути к драйверу QCDROM.SYS.

/D:MSCD001 – параметр /D: объявляет произвольный идентификатор длиной до 8 знаков для опознания драйвера программой MSCDEX.EXE (5.08-03) или программой SHSUCDX.COM (5.08-04). В строке запуска этих программ должен быть указан такой же параметр /D: с точно тем же идентификатором. Если параметр /D: не задан, то по умолчанию драйвер присвоит идентификатору значение QCDROM1.

/L – необязательный параметр, предписывающий не использовать прямой доступ к памяти (DMA) за пределами обыкновенной памяти (выше 640 кбайт). Это необходимо, если контроллер памяти не поддерживает DMA в области UMB, тогда как она открыта для обычного доступа, например, с помощью драйвера UMBPCI.SYS (5 04-04). Если параметр /L указан, то обмен данными происходит через буфер в XMS-памяти, для чего предварительно должен быть загружен драйвер HIMEM.SYS (5.04-01).

Помимо перечисленных параметров, в командной строке могут быть указаны следующие:

/A – заставляет драйвер обращаться к IDE-контроллеру по резервным адресам 01E8h-01EFh для первого канала и 0168h-016Fh для второго канала. Это может потребоваться, в частности, для некоторых версий BIOS, обслуживающих интерфейс SATA.

/I – заставляет драйвер QCDROM.SYS сформировать свой собственный буфер в XMS памяти. Это позволит избежать возможных конфликтов даже тогда, когда система BIOS и контроллер DMA не вполне соответствуют принятым сейчас стандартам.

/UF – использовать ускоренный прямой доступ в память (UltraDMA). Осуществимость UltraDMA установлена для многих, но не для всех комплектов микросхем (чипсетов) материнских плат. В каждом конкретном случае рекомендовано проверять, допустимо ли указывать параметр /UF.

/UX – запретить ускоренный прямой доступ в память (UltraDMA), даже если дисковод и материнская плата компьютера способны его обеспечить (это бывает нужно для диагностики и тестирования). Когда ускоренный прямой доступ не используется, драйвер QCDROM.SYS не нуждается в XMS-памяти.

/PM – (= Primary Master): не проводить поиск дисководов по портам, а проверить наличие ведущего оптического дисковода у первого IDE-контроллера.

/PS – (= Primary Slave): не проводить поиск дисководов по портам, а проверить наличие ведомого оптического дисковода у первого IDE-контроллера.

/SM – (= Secondary Master): не проводить поиск дисководов по портам, а проверить наличие ведущего оптического дисковода у второго IDE-контроллера.

/SS – (= Secondary Slave): не проводить поиск дисководов по портам, а проверить наличие ведомого оптического дисковода у второго IDE-контроллера,

В одной командной строке могут быть сразу несколько параметров, исключающих проведение поиска (но не более трех). Нумерация оптических дисководов производится начиная с нуля в том порядке, в каком соответствующие параметры перечислены в командной строке. Если ни в одной из указанных позиций оптические дисководы не будут обнаружены, то драйвер QCDROM.SYS не загрузится, причем наличие оптических дисководов в других позициях будет проигнорировано.

5.09-04 Драйвер DVS.SYS для дисководов CD/DVD-ROM.

Драйвер оптических дисководов DVS.SYS версии 1.1 разработан в 1999 году корпорацией Digital Video Systems Corp. Он обеспечивает доступ к дискам CD и к дискам DVD, не нуждаясь в предварительной загрузке драйвера ATAPIMGR.SYS. На сайте фирмы DVS драйвера DVS.SYS нет, но его можно скачать из архива http://web.archive.org/web/20030212210152/www.dr-tech.com/drivers/cdroms.html в составе самораспаковывающегося архивного файла DRDVDWD.EXE.

Загружать драйвер DVS.SYS следует командами DEVICE (4.06) или DEVICEHIGH (4.07) из строки файла CONFIG.SYS, например:

```
DEVICEHIGH=\DOS\DRV\Dvs.sys /D:MSCD001
```

здесь:

\DOS\DRV\ – пример пути к драйверу DVS.SYS.

/D:MSCD001 – параметр /D: объявляет произвольный идентификатор длиной до 8 знаков для опознания драйвера программой MSCDEX.EXE (5.08-03) или программой SHSUCDX.COM (5.08-04). В строке запуска этих программ должен быть указан такой же параметр /D: с точно тем же идентификатором.

Драйвер DVS.SYS выполняет поиск оптических дисководов по портам, причем делает это довольно медленно. Экспериментально установлено, что он способен обслужить по крайней мере два оптических дисковода, подключенных к любому из двух стандартных IDE-контроллеров с обычными спецификациями портов и линий вызова прерываний (1F0h с IRQ 14 и/или 170h с IRQ 15).

Глава 6 Избранные программы для MS-DOS7

Расширение и совершенствование состава операций в MS-DOS7 обеспечиваются привлечением исполняемых программных файлов (утилит). К операционным системам WINDOWS-95/98 фирма Microsoft прилагает комплект программ для DOS. Если источник программы не указан, то, значит, она входит в прилагаемый комплект и при стандартной установке находится в каталоге \WINDOWS\COMMAND. Именно эти программы преимущественно представлены в данной главе. Помимо того, в MS-DOS7 успешно работают многие программы из предыдущих версий DOS, а также весьма многочисленные программы, созданные другими фирмами и частными лицами.

Большинство программ, разработанных для MS-DOS, содержит встроенную краткую справку. Чтобы вывести ее на экран, обычно достаточно запустить программу на исполнение со знаком вопроса в качестве единственного параметра: /?. Реже встречаются программы, выводящие справку при запуске с параметром – h или вообще без параметров.

Некоторые программы запрашивают номер версии MS-DOS и выдают сообщение об ошибке, если возвращаемый номер отличен от ожидаемого. Это не обязательно обусловлено несовместимостью: почти все такие программы успешно работают в MS-DOS7, когда возвращаемый номер версии подменен данными из таблицы драйвера SETVER.EXE (5.01-02).

Несоответствие номеров версий DOS иногда вызывает конфликты. Наличие в текущем каталоге программы, относящейся к другой версии DOS, блокирует

возможность адресовать действующую программу с тем же именем через переменную окружения PATH (2.02-02), потому что MS-DOS первой находит и пытается запустить на исполнение неподходящую программу из текущего каталога. В комплекте поставки MS-DOS7 подобным конфликтам потенциально подвержены программы ATTRIB.EXE, CHKDSK.EXE, COMMAND.COM, DEBUG.EXE, DISKCOPY.COM, DOSKEY.COM, FC.EXE, FIND.EXE, FORMAT.COM, LABEL.EXE, MEM.EXE, MODE.COM, SORT.EXE, SUBST.EXE, XCOPY.EXE. Для пользования этими программами желательно заранее заготавливать пункты меню файл-менеджера или batch-файлы, содержащие строки вызова с полной спецификацией пути (примеры – в 9.03).

Приводимые в данной главе примеры применения упомянутых и других программ, как правило, не содержат спецификации пути к вызываемой программе. Предполагается, что драйвер SETVER.EXE, переменная окружения PATH и распределение файлов по каталогам уже подготовлены к моменту начала пользования программами так, чтобы эти программы можно было бы вызывать по именам.

6.01 ATTRIB.EXE – изменение атрибутов файлов

В каталогах каждому содержащемуся в них файлу соответствует запись (A.09-1), в которой байт атрибутов со смещением 0Bh определяет статус файла и права доступа к нему (A.09-2). Изменить атрибуты файлов можно с помощью программы ATTRIB.EXE, например, так:

```
ATTRIB.EXE +R -A C:\DOS\COM\*.txt /S
```

здесь:

+R –A – установить атрибут R (только для чтения) и снять атрибут A (подлежит архивированию). Всего может быть указано до четырех атрибутов: A, H (скрытый файл), R и S (системный файл). Для установки атрибута ему должен предшествовать знак плюс, для снятия – знак минус. Не указанные атрибуты не изменяются.

C:\DOS\COM*.txt – пример пути и маски для файлов, атрибуты которых должны быть изменены: в данном случае это – все текстовые файлы в указанном каталоге. Вместо маски может быть указано конкретное имя файла. Если путь не указан, то по умолчанию принимается текущий каталог.

/S – необязательный параметр, вызывающий продолжение поиска подлежащих преобразованию файлов в подкаталогах указанного каталога.

Примечание 1: если в строке вызова атрибуты не указаны, то программа ATTRIB.EXE выводит на экран список файлов, соответствующих приведенной маске, с перечнем их атрибутов и других характеристик.

6.02 CHKDSK.EXE – программа проверки дисков

Программа CHKDSK.EXE анализирует и исправляет FAT (File Allocation Tables = таблицы размещения файлов) на дискетах и жестких магнитных дисках с файловыми системами FAT-12, FAT-16 и FAT-32. Путем сопоставления данных в первой и второй таблицах FAT с данными из каталогов CHKDSK.EXE выявляет перекрестные ссылки и потерянные кластеры. Последние преобразуются в файлы с суффиксом *.CHK, помещаемые в корневой каталог того же диска. Закончив проверку, CHKDSK.EXE выводит на экран сводку полученных результатов.

Когда CHKDSK.EXE запускается на исполнение без параметров, он начинает проверять текущий диск с установками по умолчанию. Допускается в командной строке указывать следующие необязательные параметры:

```
CHKDSK.EXE C: /F /V
```

здесь:

C: – пример спецификации диска, подлежащего проверке;
/F – разрешение исправлять ошибки по мере их обнаружения;
/V – выводить на экран имена всех файлов с указанием путей.

Примечание 1: программа SCANDISK.EXE (6.21) выполняет все проверки, которые делает CHKDSK.EXE, и ряд проверок сверх того. Поэтому CHKDSK.EXE целесообразен в основном как информационная программа, показывающая сведения об использовании диска.

Примечание 2: с помощью программы CHKDSK.EXE нельзя проверять сетевые диски, оптические диски CD-ROM, а также виртуальные диски, создаваемые программами ASSIGN.COM, SUBST.EXE и JOIN.EXE.

Примечание 3: программа CHKDSK.EXE не выясняет, поврежден или нет каждый проверяемый файл и можно ли его прочитать.

Примечание 4: не следует запускать программу CHKDSK.EXE с параметром /F для проверки дисков, подозреваемых в том, что они заражены вирусом. Сначала надо запустить антивирусную программу.

Примечание 5: не следует проверять программой CHKDSK.EXE диски, имеющие от 4085 до 4087 кластеров, потому что на таких дисках CHKDSK.EXE может показывать несуществующие ошибки, попытки исправления которых приводят к потере данных.

6.03 CHOICE.COM – ввод выбора пользователя

Программа CHOICE.COM предназначена для создания интерактивных меню в ходе исполнения batch-файлов. Она принимает знак, задаваемый нажатием клавиши на клавиатуре или посланный посредством перенаправления, и устанавливает код ошибки (errorlevel, 3.15-03, 9.07-03) в соответствии с номером принятого знака в заранее заданной последовательности знаков, выражающей множество допустимых альтернатив.

Вот пример использования программы CHOICE.COM по ее прямому назначению:

```
CHOICE.COM /C:YNC /T:C,10 Yes, No or Continue
```

здесь:

/C:YNC – необязательный параметр /C: представляет перечень принимаемых знаков. Возвращаемый код ошибки соответствует порядковому номеру принятого знака в этом перечне: Y – 1, N – 2, C – 3. Когда параметр /C: не указан, по умолчанию принимается перечень из двух букв: YN (Y – 1, N – 2).

/T:C,10 – необязательный параметр /T: устанавливает интервал 10 секунд ожидания ответа пользователя (допускается от 0 до 99 секунд) и определяет одну из альтернатив (в данном примере C, код ошибки 3), которую следует выбрать по умолчанию, если ответ пользователя в течение указанного интервала времени не поступит. Если интервал ожидания не указан, CHOICE.COM будет ждать сколь угодно долго. При нулевом интервале ожидания CHOICE.COM просто устанавливает желаемый код ошибки сразу.

Yes, No or Continue – пример необязательной строки подсказки, выводимой на экран перед началом интервала ожидания ответа пользователя. В роли подсказки годится любая группа слов, которой не предшествует знак прямой косой черты.

Перед строкой подсказки CHOICE.COM позволяет указать еще два необязательных параметра:

/S – воспринимать заглавные и строчные буквы как разные.

/N – не выводить на экран после строки подсказки перечень альтернативных знаков ответа и знак вопроса.

Возвращаемый программой CHOICE.COM код ошибки используется для того, чтобы изменять дальнейший ход исполнения команд batch-файла. Код ошибки сохраняется при исполнении встроенных команд командного интерпретатора, пока не будет вызвана на исполнение еще какая-либо программа. Значения кода ошибки можно проверять в нескольких последующих строках посредством исполнения

команд по условию "If errorlevel" (3.15-03) или в цикле FOR (3.13), позволяющем присвоить возвращаемый код ошибки в качестве значения переменной окружения, например, так:

```
FOR %%Z in (1 2 3) DO if errorlevel %%Z set Err=%%Z
```

Аналогичным образом цикл FOR с проверкой кода ошибки можно использовать для исполнения условного перехода (командой goto L%%Z), но только перечисляемые в скобках значения кода ошибки в этом случае должны следовать не в порядке возрастания, а в порядке убывания.

Существенно иная область применения программы CHOICE.COM – это разбор слов по буквам. Предположим, что для анализа слов написан batch-файл CHECK.BAT, которому каждое анализируемое слово должно быть представлено раздельно буква за буквой. В этой задаче программа CHOICE.COM может помочь следующим образом:

```
ECHO ; | CHOICE /S /C:;Anyword; Call CHECK.BAT > Temp.bat
Call Temp.bat
```

Здесь подлежащее разбору слово (Anyword) окружено знаками точки с запятой, которые необходимы для правильного выделения первой и последней букв. Кроме того, добавление знаков точки с запятой к перечню допустимых альтернатив позволяет перенаправить командой ECHO заведомо имеющийся знак и гарантирует тем самым безостановочное исполнение данной командной строки. Направляемая в канал STDOUT подсказка представлена группой слов "Call CHECK.BAT". К ней программа CHOICE.COM добавляет перечень альтернатив, разделенных запятыми, который заключен в квадратные скобки и заканчивается знаком вопроса. Все это сообщение на экран не попадает, оно перенаправлено во временный файл TEMP.BAT. В результате в файл TEMP.BAT оказывается записана строка:

```
Call CHECK.BAT [;,A,n,y,w,o,r,d,;]?
```

Вызов на исполнение файла TEMP.BAT приводит к исполнению данной командной строки, в ходе которого анализирующий batch-файл CHECK.BAT получает буквы разбираемого слова в виде значений формальных параметров от %2 до %8, так что их можно анализировать отдельно одну за другой.

Примечание 1: если пользователь прервет интервал ожидания ответа нажатием клавишных комбинаций CTRL-BREAK или CTRL-C, то программа CHOICE.COM оставит код ошибки 0.

Примечание 2: когда пользователь нажимает какую-нибудь клавишу, не входящую в перечень альтернатив, программа CHOICE.COM посылает в канал STDOUT управляющий символ 07h, вызывающий звуковой сигнал.

Примечание 3: когда программа CHOICE.COM обнаруживает в командной строке какую-либо ошибку, она возвращает код ошибки 255.

6.04 COMMAND.COM – командный интерпретатор

Командный интерпретатор COMMAND.COM – это резидентная программа, представляющая приглашение командной строки, обеспечивающая исполнение команд из командной строки, а также автоматическое считывание и исполнение команд из строк batch-файлов. В файле COMMAND.COM содержится исполняемый код всех встроенных команд интерпретатора, описанных в главе 3.

Управление компьютером передается командному интерпретатору COMMAND.COM в процессе загрузки, когда загрузчик IO.SYS выполняет команду SHELL (4.26) в файле CONFIG.SYS. В результате этого первого исполнения командного интерпретатора загружается его резидентный модуль и создается первичное окружение, содержащее глобальные переменные. После этого каждое следующее исполнение командного интерпретатора приводит к повторной загрузке резидентного модуля и созданию нового (дочернего) окружения, наследующего копии всех переменных из первичного (родительского) окружения.

В отличие от обычных программ, командный интерпретатор COMMAND.COM не оставляет в своем PSP (= префиксе сегмента программы, A.07-1) ссылки на исходный (родительский) PSP, в результате чего родительское окружение сохраняется скрытно без легального пути доступа. Повторная загрузка командного интерпретатора COMMAND.COM позволяет изменить локальные переменные окружения и условия исполнения программ, например, обеспечить пошаговое исполнение batch-файла. Когда повторно загруженный резидентный модуль выполнит свою миссию, его можно закрыть командой EXIT (3.12), при этом дочернее окружение со всеми его переменными теряется, управление снова берет на себя загруженный ранее родительский резидентный модуль командного интерпретатора, и снова становятся доступными сохраненные прежние значения переменных из родительского окружения.

Вот пример загрузки командного интерпретатора, в частности, для пошагового исполнения одного batch-файла:

```
COMMAND.COM C:\dos\ CON /E:1008 /L:512 /U:255 /Y /C R:\Trial.bat
```

здесь:

C:\dos\ – пример пути к файлу COMMAND.COM; этот путь должен оканчиваться знаком обратной косой черты и служит для переопределения значения переменной %COMSPEC%. Если имеется родительское окружение, то путь можно не указывать, и тогда переменная %COMSPEC% унаследует свое значение из

родительского окружения. Но если путь указан, то он должен быть первым параметром после имени командного интерпретатора.

CON – пример спецификации устройства, через которое должен производиться ввод и вывод данных. CON – это консоль, то есть дисплей для вывода данных и клавиатура для ввода. Именно так осуществляется ввод-вывод по умолчанию, и потому спецификация устройства ввода-вывода в командной строке обычно не указывается. Однако здесь потенциально могут быть указаны и другие устройства ввода-вывода (3.07).

/E:1008 – необязательный параметр, вызывающий резервирование 1008 байт памяти для размещения переменных окружения. Допускается резервировать от 256 до 32768 байт. По умолчанию выделяется пространство для копирования всех переменных родительского окружения, но не менее 160 байт.

/L:512 – необязательный параметр, устанавливающий размер внутреннего буфера 512 байт. Допускается от 128 до 1024 байт, по умолчанию принимается 256 байт. Размер внутреннего буфера должен быть достаточно большим, чтобы в нем уместить командную строку после выполнения всех подстановок.

/U:255 – необязательный параметр, устанавливающий размер входного буфера 255 байт. Допускается от 128 до 255 байт, по умолчанию принимается 128 байт. Этот размер определяет максимальную длину вводимой командной строки.

/Y – необязательный параметр, задающий пошаговый режим интерпретации строк batch-файлов. Параметр /Y несовместим с параметром /P, и когда они указаны вместе, параметр /Y игнорируется.

/C – этот параметр объявляет, что вслед за ним указано имя команды, которую интерпретатору надлежит исполнить и потом выгрузить свой резидентный модуль из памяти. Если вместо /C указать параметр /K, то после исполнения команды резидентный модуль интерпретатора останется в памяти, пока не поступит команда EXIT (3.12). Вместо /C также можно указать параметр /P, который не объявляет команду, а означает постоянную загрузку интерпретатора с блокировкой выхода по команде EXIT. Параметр /P должен быть последним в строке при загрузке интерпретатора командой SHELL (4.26) из файла CONFIG.SYS.

R:\Trial.bat – пример имени файла, передаваемого на исполнение интерпретатору COMMAND.COM. Имя исполняемого файла можно указывать после параметров /C или /K, которые должны быть последними из параметров командного интерпретатора в

командной строке. Вслед за именем исполняемого файла допускается указывать только те параметры, которые должны быть переданы этому исполняемому файлу.

Помимо перечисленных выше, командный интерпретатор принимает также следующие необязательные параметры:

/MSG – заранее загрузить в память сообщения о возможных ошибках. При сбоях, препятствующих считыванию текста сообщения из файла, этот параметр обеспечивает воспроизведение на экране адекватного сообщения об ошибке.

/LOW – загрузить резидентный модуль командного интерпретатора в обыкновенную память, то есть ниже границы 640 кбайт. Параметр /LOW применяется преимущественно при первичной загрузке совместно с параметром /P.

/F – этот параметр обеспечивает безостановочное исполнение с игнорированием ошибок, как после ответа пользователя "Fail". Параметр /F действует только совместно с параметром /C на время исполнения одной команды. Но команда FOR (3.13) распространяет действие параметра /F на все операции цикла, а команда CALL (3.02) – на все операции batch-файла.

/Z – показывать код ошибки каждый раз после исполнения любой программы, возвращающей значение кода ошибки.

Примечание 1: от параметра /C зависит то, как происходит прерывание исполнения batch-файлов (1.03).

Примечание 2: когда интерпретатор COMMAND.COM загружается впервые с параметром /P, то он по умолчанию начинает исполнение файла AUTOEXEC.BAT, который должен находиться в корневом каталоге текущего диска.

Примечание 3: если файл AUTOEXEC.BAT исполняется по умолчанию при запуске интерпретатора командой SHELL, то формальный параметр %0 не вызывает рекурсивного исполнения файла AUTOEXEC.BAT.

Примечание 4: когда командный интерпретатор COMMAND.COM запущен с параметром /K, он способен принимать на исполнение команды от других процессов или из командных файлов через перенаправление ввода (еще об этом – в разделах 2.04-02 и 2.04-05, а также в примечании 1 при вводной статье к разделу 6.05).

Примечание 5: исполняемые командным интерпретатором batch-файлы (имеющие суффикс *.BAT) непосредственно пользуются переменными окружения, принадлежащими интерпретатору. В отличие от этого обычные исполняемые файлы (с суффиксами *.COM и *.EXE) получают в свое распоряжение только копии переменных окружения.

6.05 DEBUG.EXE – отладчик и мини-ассемблер

Отладчик DEBUG.EXE - это специализированный командный интерпретатор, написанный Тимом Паттерсоном (Tim Patterson) как инструмент для создания той операционной системы, которая позже была куплена фирмой Microsoft и стала называться первой версией MS-DOS. Отладчик DEBUG.EXE помогает искать и исправлять ошибки как в настройке компьютера, так и в кодах программ. Возможности отладчика DEBUG.EXE не ограничены его встроенными командами, потому что эти команды позволяют ассемблировать и исполнить любой машинный код. Конечно, по части написания сложных программ отладчик - не конкурент языкам высокого уровня. Однако обстоятельства диктуют другие критерии, когда способности известных компиляторов оказываются недостаточны, когда нужно что-либо выяснять или проверять. Если Вы делаете это в реальном режиме с помощью отладчика DEBUG.EXE, то все AT-совместимые компьютеры покорно предоставят себя в Ваше распоряжение. Вы получите прямой доступ к дискам, к портам, к памяти, к данным и к исполняемому коду в файлах.

При вызове отладчика DEBUG.EXE из командной строки после его имени может быть указано имя файла, который сразу будет загружен в память и подготовлен к отлаживанию, например:

```
DEBUG.EXE  Trial.com /B /S
```

здесь:

Trial.com – пример имени файла, подлежащего отлаживанию. Указанные после имени параметры (/B /S) – это параметры не отладчика, а файла Trial.com. Они будут записаны в PSP файла Trial.com так же, как это делает командный интерпретатор COMMAND.COM, когда подготавливает файл к исполнению. Размещение загружаемого файла в памяти зависит от его суффикса (подробнее - в разделе 6.05-10).

Если подлежащий исполнению файл в командной строке не указан, то он может быть задан и загружен позднее встроенными командами отладчика "N" и "L" (6.05-10, 6.05-12). Отложенная загрузка командами отладчика дает возможность размещать загружаемый код начиная с произвольного стартового адреса, что важно, в частности, при отлаживании драйверов (подробнее - в разделе 6.05-18).

Как и командный интерпретатор COMMAND.COM, отладчик DEBUG.EXE принимает команды из командной строки и активизирует клавиши редактирования командных строк (1.05). Если в строке вызова указано перенаправление ввода (2.04-02), то вместо команд с клавиатуры DEBUG.EXE будет принимать перенаправляемые команды. Это свойство позволяет записывать последовательности команд в текстовые командные файлы и затем посылать их отладчику для автоматического исполнения:

```
Debug.exe < Cmnd_txt.scr
```

Все спецификации загружаемых и ассемблируемых кодов могут быть включены в такие командные файлы (примеры в разделах 9.02, 9.06, 9.08 и 9.10).

Примечание 1: когда исполняющая программа принимает команды из файла посредством перенаправления, она теряет контакт с клавиатурой через канал STDIN, и тогда ввести данные с клавиатуры посредством прерываний INT 21\AH=01h,06h,07h,08h,0Ah (8.02-02, 8.02-04, 8.02-06) невозможно. По той же причине исполняющая программа "зависает", если в последних строках командного файла не приняты меры для восстановления взаимодействия с клавиатурой через канал STDIN. В простейшем случае эту роль играет команда завершения сеанса ("Q"). Другой способ восстановления взаимодействия с клавиатурой без завершения сеанса показан в разделе 9.07-02.

Примечание 2: Пауль Войта (Paul Vojta) разрабатывает усовершенствованный вариант отладчика DEBUG.EXE, который отличается от описываемого здесь отладчика фирмы Microsoft тем, что способен "понимать" команды более современных процессоров. Сейчас, в 2008 году, на сайте http://www.japheth.de/dwnload4.html свободно выложен файл архива Debug113.zip, содержащий версию 1.13 отладчика Пауля Войта. Если специально не оговорено иное, то материал раздела 6.05 в равной мере относится к этому отладчику.

6.05-01 Отладчик DEBUG.EXE: команды и адреса

Когда отладчик DEBUG.EXE запущен, он представляет приглашение командной строки в виде знака дефис (–). Это означает, что он готов принять к исполнению команду от пользователя.

Команды отладчика DEBUG.EXE имеют однобуквенные или двухбуквенные имена, за которыми могут следовать параметры, отделенные друг от друга запятыми или пробелами. Разделительные знаки можно не указывать, если их отсутствие в конкретной позиции не вызывает неопределенности. DEBUG.EXE воспринимает все цифры в командах как шестнадцатеричные, а строчные и заглавные буквы как эквивалентные; исключением является только сопоставление данных при поиске командой SEARCH (6.05-16).

Первый параметр после имени команды обычно представляет адрес стартовой точки в памяти. Он может быть задан в полной форме – как сегментный адрес и смещение (например, 1FA5:0100), или через ссылку на сегментный регистр

(например, CS:0100), или в краткой форме – как только смещение (например, 0100). В последнем случае сегментный адрес будет взят по умолчанию: из регистра CS (Code Segment = кодовый сегмент) – для команд A (Assemble = собрать), G (Go = исполнить), L (Load = загрузить), P (Proceed = продолжить), T (Trace = отследить), U (Unassemble = разобрать) и W (Write = записать), а для всех других команд – из регистра DS (Data Segment = сегмент данных). В спецификации смещения допускается не указывать начальные нули: например, смещение 100 всегда будет интерпретироваться как 0100h.

Начальные установки всех сегментных регистров одинаковые и определяются тем сегментом, который выделила DOS по запросу отладчика DEBUG.EXE для отлаживаемой программы. Начальная установка смещения по умолчанию принимается 0100h (т.е. в десятичной форме 256), она записывается в регистр IP (Instruction Pointer = указатель команд) и выражает смещение первого байта исполняемого кода отлаживаемой программы относительно начала выделенного ей сегмента. Резервируемый таким образом 256-байтный участок адресного пространства известен как PSP (Program Segment Prefix = префикс сегмента программы, показанный в приложении A.07-1).

Когда действие команды отладчика DEBUG.EXE должно быть направлено на группу байт, эта группа задается своим начальным адресом в любой форме и еще вторым параметром – либо длиной, либо смещением конца группы. Параметр, задающий длину, должен начинаться с буквы "L" – например, параметр L20 определяет группу длиной в 20h байт. Сумма начального адреса и длины не должна превосходить FFFFh. Если шестнадцатеричное число в позиции второго параметра спецификации группы байт не начинается с буквы "L", то оно интерпретируется как смещение конца группы, причем указывать сегментный адрес перед этим смещением нельзя. Разумеется, смещение для конца группы должно быть больше, чем смещение для ее первого байта в составе начального адреса.

Закончив набор командной строки, пользователь посылает ее на исполнение нажатием клавиши ENTER (на некоторых клавиатурах она обозначена как CR).

Наиболее простые команды отладчика DEBUG.EXE представлены всего лишь одним знаком и не нуждаются ни в параметрах, ни в особых комментариях:

Q – ("Quit") – закончить сеанс работы с отладчиком DEBUG.EXE
? – показать краткий список команд отладчика DEBUG.EXE.

Показываемый отладчиком список команд, однако, слишком краток, чтобы служить руководством к действию. Необходимые пояснения по пользованию всеми другими командами приведены в следующих статьях раздела 6.05.

Примечание 1: если Вы ошиблись при наборе команды, отладчик DEBUG.EXE выводит в следующей строке сообщение "Error" (= ошибка) и указатель в виде стрелочки вверх ^ (caret), показывающий на то

место в предыдущей строке, которое он не способен "понять". Наиболее вероятно ошибка находится именно в этом месте, но тем не менее нередко ошибка находится раньше, в предшествующей части строки.

Примечание 2: абсолютный адрес в памяти рассчитывается как сумма смещения с сегментным адресом, умноженным на 16. Если сегментный адрес, например, 1FA5h, а смещение 0100h, то абсолютный адрес будет

(1FA5h x 10h) + 0100h = 1FB50h.

6.05-02 Отладчик DEBUG.EXE: команда "Assemble"

Команда "Assemble" (= собирать) – переключает отладчик DEBUG.EXE в режим преобразования ассемблерных инструкций в машинный код. Этот код не исполняется сразу, а записывается в память, образуя последовательность машинных команд, которая после выхода из режима ассемблирования может быть исполнена или сохранена в файле. В командной строке отладчика вызов команды "Assemble" задается буквой "A":

```
A 0100
```

После имени команды "A" указан начальный адрес для размещения получаемого машинного кода. Здесь адрес представлен только смещением, но допустимы все формы адреса, перечисленные в разделе 6.05-01. Если сегмент в адресе не указан, то по умолчанию он будет взят из регистра CS:. Можно вообще адрес не указывать, тогда он будет взят из регистров CS:IP.

После получения команды "A" отладчик DEBUG.EXE показывает полный стартовый адрес, выделенный для размещения машинного кода команды, и мигающим курсором предлагает ввести ассемблерную команду, которую предстоит перевести в машинный код. В режиме ассемблирования отладчик DEBUG.EXE не принимает команды, описываемые в разделе 6.05, а принимает только те инструкции и ассемблерные команды, которые детально описаны в главе 7. Окончание набора каждой ассемблерной команды пользователь подтверждает нажатием клавиши "ENTER". Отладчик DEBUG.EXE сразу переводит команду в машинный код, записывает его в память и показывает следующий полный адрес с приглашением ввести следующую ассемблерную команду.

Чтобы выйти из режима ассемблирования нужно, не вводя очередной ассемблерной команды, просто нажать клавишу "ENTER" при пустой командной строке. Тогда отладчик выйдет в интерактивный режим, выведет на экран свое приглашение в виде знака дефис и снова будет готов принимать команды, описываемые в разделе 6.05. Когда отладчик DEBUG.EXE получает команды из

файла через перенаправление, для выхода из режима ассемблирования нужно оставить в командном файле пустую строку (7.01-04).

6.05-03 Отладчик DEBUG.EXE: команда "Compare"

Команда "Compare" (= сравнить) показывает несовпадающие байты в двух группах последовательно расположенных ячеек памяти. Одинаковые байты при этом пропускаются. В командной строке отладчика вызов команды "Compare" задается буквой "C":

```
C 113 L8 153
```

В приведенном примере первый и третий параметры после буквы "C" – это начальные адреса сопоставляемых последовательностей, в данном случае представленные смещениями 0113h и 0153h. Однако допустимы все формы адресов, перечисленные в разделе 6.05-01. Если сегмент в адресе не указан, то по умолчанию он будет взят из регистра DS:. Второй параметр задает либо смещение конечного байта первой группы (если числу не предшествует буква "L"), либо длину интервала сопоставления; в данном случае заявлена длина 8h байт. Все три параметра в этой команде обязательные.

6.05-04 Отладчик DEBUG.EXE: команда "Dump"

Команда "Dump" (= вывалить) показывает в шестнадцатеричной байтовой форме состояние ячеек памяти, и одновременно в правой части экрана показывает содержание, записанное в те же ячейки памяти, знаками кода ASCII.

В командной строке отладчика DEBUG.EXE вызов команды "Dump" задается буквой "D":

```
D 19A9:02E0 L10
```

Здесь сразу после имени команды указан пример полного адреса начальной ячейки той группы ячеек памяти, которые надлежит показать. Адрес можно указывать в любой допустимой форме (6.05-01). Если сегмент в адресе не указан, то по умолчанию он будет взят из регистра DS:. Второй параметр задает либо смещение конечной ячейки той же группы (если ему не предшествует буква "L"), либо длину показываемой группы ячеек. В данном случае задана длина 10h (= 16) байт. Если второй параметр в команде "Dump" не задавать, то по умолчанию будет принята длина L80, то есть 128 байт. Если не задавать и адрес тоже, то будут показаны 80h байт, начиная с текущего смещения относительно сегментного регистра DS:. При каждом исполнении команды "Dump" значение текущего смещения увеличивается на длину показанной группы ячеек, так что при каждом

следующем исполнении без указания начального адреса выводится на экран не та же, а следующая группа ячеек памяти.

Примеры пользования командой "Dump" показаны на рис. 8 – 12 (в главе "A").

6.05-05 Отладчик DEBUG.EXE: команда "Enter"

Команда "Enter" (= ввести) записывает новые данные в ячейки памяти начиная с указанного адреса. В командной строке отладчика DEBUG.EXE вызов команды "Enter" задается буквой "E":

```
E 0211
```

Здесь после имени команды указан пример адреса той ячейки памяти, начиная с которой следует записывать вводимые данные. Адрес может быть указан в любой допустимой форме (6.05-01), но он должен быть указан обязательно. Если сегмент в адресе не указан, то по умолчанию он будет взят из регистра DS:. Исполняя такую команду, отладчик DEBUG.EXE покажет полный адрес запрошенной ячейки памяти, содержащийся там байт данных и финальную точку. Эту точку надо понимать как приглашение ввести байт данных, который должен заместить прежнее содержимое этой ячейки. При вводе с клавиатуры в ответ на приглашение команды "Enter" данные в знаках кода ASCII не принимаются, их следует вводить в байтовой форме, по две шестнадцатеричных цифры на байт. По нажатию клавиши "Пробел" ("Spacebar") новый байт, если он введен, будет записан в ячейку, а если строка ввода оставлена пустой, то прежнее содержимое ячейки памяти сохранится. В любом случае потом выводится байт из следующей ячейки, и Вам точно так же предлагается его заменить. Если вместо клавиши "Пробел" нажать клавишу "–" (минус или дефис), то Вы получите возможность вернуться к изменению содержимого предыдущей ячейки. Чтобы закончить процесс ввода данных, достаточно нажать клавишу ENTER.

Команда "Enter" исполняется иначе, без обращения к вводу с клавиатуры, если после адреса запрашиваемой ячейки в той же командной строке указаны подлежащие вводу данные, например:

```
E 03E0 'Data error' 0D 0A
```

Вводимые данные в командной строке могут быть представлены либо побайтно, по две шестнадцатеричных цифры на байт, либо знаками кода ASCII, заключенными с обеих сторон в кавычки или двойные кавычки, причем допускается чередование обеих форм представления данных в произвольном порядке. Перед записью заключенные в кавычки знаки кода ASCII буква за буквой переводятся в шестнадцатеричную форму, но сами кавычки не переводятся и не записываются. Побайтно вводимые данные допускается представлять одной шестнадцатеричной цифрой на байт, которая будет интерпретирована как младшая

цифра байта, например, цифра "A" будет интерпретирована как 0Ah. В любом случае запись данных в последовательно расположенные ячейки памяти производится в том порядке, в каком данные указаны в командной строке.

6.05-06 Отладчик DEBUG.EXE: команда "Fill"

Команда "Fill" (= заполнить) заполняет группу ячеек памяти повторяющейся записью одного и того же байта или одной и той же последовательности байтов. В командной строке отладчика DEBUG.EXE вызов команды "Fill" задается буквой "F":

```
F 03E0 L2E 0D 0A 'Reserved' 90 90
```

В приведенном примере первый параметр – это адрес первой ячейки из заполняемой группы ячеек, он представлен смещением 03E0h, но здесь допустимы все формы адресов, перечисленные в разделе 6.05-01. Если сегмент в адресе не указан, то по умолчанию он будет взят из регистра DS:. Второй параметр задает либо смещение для конечной ячейки группы (если числу не предшествует буква "L"), либо длину группы ячеек; в данном случае заявлена длина 2Eh байт. Третий и все последующие параметры в приведенной командной строке представляют ту последовательность данных, которой надлежит заполнять ячейки памяти. Так или иначе длина, начальный адрес и хотя бы один байт данных в команде "F" должны быть заданы обязательно.

Данные в командной строке могут быть представлены либо побайтно, по две шестнадцатеричных цифры на байт, либо знаками кода ASCII, заключенными с обеих сторон в кавычки или двойные кавычки, причем допускается чередование обеих форм представления данных в произвольном порядке. Заключенные в кавычки знаки кода ASCII буква за буквой переводятся в шестнадцатеричную форму, но сами кавычки не переводятся и не записываются. Если после записи всех данных из командной строки не все ячейки в группе оказались заполнены, то процесс повторяется: следующие ячейки заполняются теми же данными. Напротив, если группа ячеек не вмещает записываемую последовательность данных целиком, то процесс заполнения прекращается на последней ячейке без выдачи сообщения об ошибке.

6.05-07 Отладчик DEBUG.EXE: команда "Go"

Команда "Go" (= пошел!) запускает на исполнение весь заранее подготовленный код отлаживаемой программы или его часть. В командной строке отладчика DEBUG.EXE вызов команды "Go" задается буквой "G":

```
G =100 143
```

Необязательные параметры после имени команды "G" – это адреса. Первый адрес, которому предшествует знак равенства, указывает на первый байт машинной команды, с которой следует начать исполнение. Этот адрес может быть задан в любой из допустимых форм (6.05-01). Если сегмент в адресе не указан, то по умолчанию он будет взят из регистра CS:. Если среди приведенных адресов не будет ни одного с предшествующим знаком равенства, то стартовый адрес будет взят из регистров CS:IP.

Адреса без предшествующего знака равенства – это точки остановки (breakpoints), их можно указывать до 10 штук, чтобы хватило на каждую ветвь отлаживаемой программы. В точках остановки отладчик DEBUG.EXE замещает байты кода программы кодом "CCh" прерывания INT 03 (8.01-04). При исполнении прерывание INT 03 возвращает управление отладчику, и тогда он восстанавливает во всех точках остановки байты исходного кода отлаживаемой программы, а также сохраняет состояния регистров и флагов для продолжения отладки. Чтобы процесс завершился безошибочно, должны быть выполнены два условия. Во-первых, адреса точек остановки должны соответствовать первым байтам кода машинных команд, иначе код "CCh" будет воспринят не как прерывание INT 03, а как составная часть предыдущей машинной команды. Во-вторых, прекращение исполнения не должно произойти как-либо помимо точек остановки, потому что тогда отладчик не сохранит состояния регистров и флагов, не восстановит замещенные байты кода отлаживаемой программы, и ее придется перезагружать (командой "Load", 6.05-10).

Нужно иметь ввиду, что возможны еще два варианта завершения исполнения. Если в ходе исполнения произойдет вызов INT 21\AH=4Ch (8.02-55), то сеанс работы с отладчиком DEBUG.EXE закончится, управление будет возвращено к DOS. Если в ходе исполнения произойдет вызов INT 20 (8.02-01) или если при исходном состоянии стека будет выполнена команда RET (7.03-73), то управление будет передано от отлаживаемой программы к отладчику DEBUG.EXE, и он станет ожидать поступления следующих команд с клавиатуры или с перенаправления – смотря по тому, как он был первоначально запущен. Когда последний вариант завершения (командой RET или вызовом INT 20) предусмотрен намеренно, тогда точки остановки после команды "G" указывать не нужно, и код отлаживаемой программы перезагружать обычно не приходится, однако состояния регистров и флагов не будут сохранены.

6.05-08 Отладчик DEBUG.EXE: команда "Hexadecimal"

Команда "Hexadecimal" (= шестнадцатеричный) рассчитывает и показывает на экране сумму и разность двух четырехзначных шестнадцатеричных чисел. В командной строке отладчика DEBUG.EXE вызов команды "Hexadecimal" задается буквой "H":

```
H 12BA 00AE
```

В показанном примере два параметра представляют те числа, сумму и разность которых надлежит рассчитать. Если любое из них состоит менее чем из четырех цифр, то оно будет интерпретировано как содержащее нули в старших разрядах. Наличие обоих параметров в командной строке обязательно.

6.05-09 Отладчик DEBUG.EXE: команда "Input".

Команда "Input" (= ввод) считывает байт из указанного порта и показывает его на экране. Считанный байт никуда не записывается. В командной строке отладчика DEBUG.EXE вызов команды "Input" задается буквой "I":

```
I 03f8
```

Единственный обязательный параметр при команде "Input" – адрес порта, задаваемый четырехзначным шестнадцатеричным числом, в котором нули в старших разрядах можно опустить. Сегментный адрес перед адресом порта не имеет смысла и не допускается. Адреса некоторых портов приведены в приложении A.14-1.

6.05-10 Отладчик DEBUG.EXE: команда "Load"

Команда "Load" (= загрузить) считывает код с диска и загружает его в память компьютера, начиная с указанного адреса. Допускаются две формы задания того, откуда именно нужно считывать код: по имени файла или по номеру сектора на логическом диске. В командной строке отладчика DEBUG.EXE вызов команды "Load" задается буквой "L". Вот пример вызова этой команды для считывания файла:

```
L 0100
```

Здесь единственный параметр – адрес ячейки памяти, начиная с которой будет размещен в памяти код считываемого файла. Адрес может быть задан в любой из допустимых форм (6.05-01). Если указано только смещение, то по умолчанию сегментный адрес будет взят из регистра CS:.

Предполагается, что имя файла, который надлежит загрузить, заранее записано в PSP (A.07-1) командой "Name" (6.05-12) или автоматически перенесено туда из командной строки запуска отладчика DEBUG.EXE. Кроме того, предполагается, что сегментный адрес в регистре DS:, относительно которого адресуются данные в PSP, не изменился с момента внесения имени файла в PSP.

Если адрес после имени команды "L" не указан, то по умолчанию загрузка производится начиная с адреса CS:0100, за исключением загрузки файлов с суффиксами *.EXE и *.HEX, спецификации загрузки которых содержатся в их

заголовке. Для файлов с суффиксом *.HEX адрес из заголовка суммируется с тем, который приведен в команде "L". Для файлов с суффиксом *.EXE адрес в команде "L" игнорируется, и заголовок файла не загружается. Чтобы увидеть "как есть" файлы, загружаемые без заголовка или вовсе не загружаемые, их суффикс надо заранее заменить (предпочтительно на *.BIN). В любом случае длина файла записывается в регистр CX. Если длина файла превышает 64 килобайта, то старшие разряды длины записываются в регистр BX.

Для загрузки кода из секторов логического диска в командной строке вызова команды "Load" должны быть указаны четыре обязательных параметра, например:

```
L 0100 2 0 1
```

Первый параметр, как и в предыдущем примере, – адрес ячейки памяти, начиная с которой будет размещен в памяти считываемый код. Второй параметр интерпретируется как номер логического диска: 0 – диск A:, 1 – диск B:, 2 – диск C: и т.д. Третий параметр – номер сектора, четвертый – количество считываемых секторов (допускается до 80h секторов). В частности, показанный пример команды задает считывание одного нулевого (загрузочного) сектора с логического диска C: и запись его содержания в ячейки памяти начиная с адреса CS:0100. В отличие от операции загрузки файлов, при считывании данных из секторов длина массива в регистры BX:CX не записывается. Считывание секторов за пределами логических дисков командой "Load" не обеспечивается.

6.05-11 Отладчик DEBUG.EXE: команда "Move"

Команда "Move" (= сдвинуть) копирует блок данных из одного места памяти в другое. В командной строке отладчика DEBUG.EXE вызов команды "Move" задается буквой "M":

```
M 0100 L20 0180
```

Здесь первый параметр – начальный адрес исходного (копируемого) блока данных. Этот адрес может быть задан в любой из допустимых форм (6.05-01). Если вместо полного адреса задано только смещение, то сегментный адрес по умолчанию будет взят из регистра DS:.

Второй параметр в командной строке – конечное смещение того же исходного блока данных (если ему не предшествует буква "L") или его длина; в данном примере – длина 20h байт. Третий параметр – стартовый адрес размещения копии. Он может быть дан в полной форме, указывающей на иной сегмент. Все три параметра в этой команде обязательны.

Если участки размещения исходного блока данных и копии не перекрываются, то данные в исходном блоке не изменяются. Если участок размещения копии перекрывает блок исходных данных, то они перезаписываются без

предупреждения, но в любом случае порядок исполнения процедуры копирования автоматически выбирается так, чтобы копирование из перекрывающейся группы ячеек произошло прежде, чем туда будут записаны копируемые данные.

6.05-12 Отладчик DEBUG.EXE: команда "Name"

Команда "Name" (= имя) объявляет имя файла, который затем должен быть загружен командой "Load" (6.05-10) или записан командой "Write" (6.05-19). В командной строке отладчика DEBUG.EXE вызов команды "Name" задается буквой "N":

```
N Trial.com /S /D
```

После буквы "N" должно быть указано объявляемое имя файла. Если у имени имеется суффикс, то он должен быть указан обязательно. Имени может предшествовать путь, после имени может следовать группа передаваемых файлу параметров (как "/S /D" в данном примере).

Все данные, объявляемые командой "Name", записываются в область PSP (A.07-1), созданную отладчиком для отлаживаемой программы, начиная с адреса DS:0081. Кроме того, длина строки данных записывается в ячейку DS:0080, имя файла без суффикса записывается начиная с адреса DS:005D, а суффикс – начиная с адреса DS:0065. Поскольку все эти адреса отсчитываются относительно сегментного регистра DS:, постольку записанные данные станут недоступны командам "Load" и "Write", как только сегментный адрес в регистре DS: изменится. Когда команда "Name" исполняется без параметров, она затирает данные в ячейках DS:005C – DS:0080 и записывает код 0Dh в ячейку DS:0081.

Примечание 1: по умолчанию начальные значения сегментных адресов в регистрах CS: и DS: устанавливаются одинаковыми. Поэтому при размещении исполняемого кода отлаживаемой программы в области ниже принимаемого по умолчанию адреса CS:0100h, он может оказаться записанным поверх данных PSP (A.07-1) и, помимо того, сам может быть перезаписан данными, вводимыми командой "Name".

6.05-13 Отладчик DEBUG.EXE: команда "Output"

Команда "Output" (= выдать) посылает байт данных в указанный порт. В командной строке отладчика DEBUG.EXE вызов команды "Output" задается буквой "O":

```
O 0378 00
```

После буквы "O" в командной строке должны быть указаны два обязательных параметра. Первый параметр представляет четырехзначный шестнадцатеричный адрес порта, куда следует послать байт данных (00h), определяемый вторым

параметром. Приведенный в показанном примере адрес 0378h обычно принадлежит порту LPT1. Сегментный адрес перед адресом порта не имеет смысла и не допускается. Нули в старших разрядах адреса порта можно не указывать. Адреса некоторых портов приведены в приложении A.14-1.

Примечание 1: неосторожное пользование командой "Output" может привести к нарушению режимов работы аппаратных средств компьютера.

6.05-14 Отладчик DEBUG.EXE: команда "Proceed"

Команда "Proceed" (= продвигаться) выполняет заданное число машинных команд, причем выполняет циклы (LOOP, LOOPZ, LOOPNZ), вызовы (CALL), повторения (REPZ, REPNZ) и прерывания как одну машинную команду. В этом главное отличие команды "Proceed" от команды "Trace" (6.05-17).

В командной строке отладчика DEBUG.EXE вызов команды "Proceed" задается буквой "P":

```
P =0100 5
```

В приведенном примере после буквы "P" следуют два параметра. Первый параметр – это адрес, с которого нужно начать исполнение команд, перед этим адресом должен стоять знак равенства. Адрес можно указывать в любой допустимой форме (6.05-01). Если вместо адреса указано только смещение, то сегментный адрес по умолчанию будет взят из регистра CS:. Второй параметр – количество подлежащих исполнению команд. Если количество не указано, по умолчанию будет исполнена одна команда. Если вообще ни один параметр не указан, то будет исполнена одна машинная команда по адресу CS:IP.

После команды "Proceed" на экран дисплея выводятся текущие состояния регистров и флагов, а также результат дизассемблирования кода следующей машинной команды.

Примечание 1: команду "Proceed" нельзя применять для исполнения машинных кодов, считываемых из постоянной памяти (ROM), для этого годится только команда "Trace" (6.05-17).

6.05-15 Отладчик DEBUG.EXE: команда "Register"

Команда "Register" (= регистр), заданная в командной строке одной буквой "R" без параметров, выводит на экран состояния регистров и флагов процессора. Типичные состояния регистров и флагов, устанавливаемые в момент запуска отладчика DEBUG.EXE на исполнение, показаны на рис. 1.

Рис. 1

Обратите внимание, что в сегментные регистры (DS, ES, SS, CS) записано одно и то же число: это сегментный адрес участка памяти, выделенного операционной системой для отлаживаемой программы. Другие регистры и флаги в начальный момент приобретают фиксированные исходные состояния. Двухбуквенные обозначения начальных состояний флагов расшифровывают следующим образом:

NV	No oVerflow = нет переполнения,
UP	count UP = счет индексов на увеличение,
EI	Enable Interrupts = прерывания разрешены,
PL	Positive sign = положительное число,
NZ	No Zero = ненулевое значение или неравенство,
NA	No Auxiliary carry = нет переноса в 4-м разряде,
PO	Parity Odd = нечетная сумма битов младшего байта,
NC	No Carry = нет переноса в старшем разряде.

В последней строке выводимого на экран сообщения показаны код и дизассемблированное представление машинной команды, которая находится по адресу CS:IP. Именно эта машинная команда будет исполнена, если отладчик получит команду "Trace" или "Proceed" с параметрами по умолчанию.

Чтобы получить возможность изменять состояние флагов процессора, в командной строке после буквы "R" должен быть указан параметр "F":

```
R F
```

В ответ отладчик DEBUG.EXE показывает двухбуквенные обозначения текущих состояний флагов и приглашение в виде знака дефис указать таким же двухбуквенным кодом их новое состояние. Состояниям, противоположным показанным на рис. 1, соответствуют следующие двухбуквенные обозначения:

OV	OVerflow = переполнение,
DN	count DowN = счет индексов на уменьшение,
DI	Disable Interrupts = прерывания запрещены,
NG	NeGative sign = отрицательное число,
ZR	ZeRo = нулевое значение или равенство,
AC	Auxiliary Carry = перенос в 4-м разряде,
PE	Parity Even = четная сумма битов младшего байта,
CY	CarrY = перенос в старшем разряде.

Порядок указания новых состояний флагов безразличен. Если для каких-либо флагов новое состояние не будет указано, то эти флаги сохранят свои прежние состояния.

Чтобы получить возможность изменить состояние регистра, в командной строке после буквы "R" должно быть указано обозначение желаемого регистра, например:

```
R AX
```

Приведенный здесь пример команды откроет регистр AX процессора для записи в него нового значения. Для записи командой "Register" также доступны регистры BX, CX, DX, BP, SP, DI, SI, CS, DS, ES, SS ,IP (и PC – это еще одно имя регистра IP). При исполнении каждой такой команды отладчик DEBUG.EXE показывает записанное в регистр значение и двоеточие, приглашая тем самым ввести новое четырехзначное шестнадцатеричное значение в этот регистр. Если Вы не введете ничего и просто нажмете клавишу ENTER, то прежнее значение в данном регистре сохранится. Команда "R" не предоставляет раздельного доступа к младшему и старшему байтам (например, AH и AL) 16-разрядных регистров. Если нужно изменить состояние только одного из них, то в другой байт нужно повторно записать прежнее значение.

Примечание 1: упоминаемые здесь 8 флагов – это лишь часть флагов центрального процессора, перечень которых приведен в разделе A.11-4.

6.05-16 Отладчик DEBUG.EXE: команда "Search"

Команда "Search" (= поиск) выполняет поиск представленной последовательности данных в указанной области адресного пространства. В командной строке отладчика DEBUG.EXE вызов команды "Search" задается буквой "S":

```
S 0100 L200 20 'st'
```

В приведенном примере первый параметр – это адрес первой ячейки из той области, в пределах которой надо проводить поиск. Здесь допустимы все формы адресов, перечисленные в разделе 6.05-01. Если сегмент в адресе не указан, то по умолчанию он будет взят из регистра DS:. Второй параметр задает либо смещение для конечной ячейки области поиска (если числу не предшествует буква "L"), либо длину области поиска; в данном случае заявлена длина 200h байт. Третий и все последующие параметры в приведенной командной строке представляют ту последовательность данных, которую надлежит искать. Так или иначе длина и начальный адрес области поиска, а также хотя бы один байт искомых данных в команде "S" должны быть заданы обязательно.

Данные искомой последовательности в командной строке могут быть представлены либо побайтно, по две шестнадцатеричных цифры на байт, либо

знаками кода ASCII, заключенными с обеих сторон в кавычки или двойные кавычки, причем допускается чередование обеих форм представления данных в произвольном порядке. Перед поиском заключенные в кавычки знаки кода ASCII (кроме самих кавычек) буква за буквой переводятся в шестнадцатеричную форму, причем в ходе поиска строчные и заглавные буквы не будут считаться одинаковыми.

После нажатия клавиши ENTER отладчик DEBUG.EXE покажет все адреса в пределах области поиска, где искомая последовательность будет найдена. Следует иметь ввиду, что DEBUG.EXE не анализирует, какую миссию выполняют найденные фрагменты. Одни и те же последовательности байтов могут представлять собой данные, а могут входить в состав исполняемых команд. Все неопределенности такого рода предоставляется разрешать пользователю.

6.05-17 Отладчик DEBUG.EXE: команда "Trace"

Команда "Trace" (= проследить) выполняет заданное число машинных команд, начиная с указанного адреса. В отличие от команды "Proceed" (6.05-14), команда "Trace" прослеживает исполнение машинных кодов в составе циклов, вызовов подпрограмм и т.п. В командной строке отладчика DEBUG.EXE вызов команды "Trace" задается буквой "T":

```
T =0100 5
```

В приведенном примере после буквы "T" следуют два параметра. Первый параметр – это адрес, с которого нужно начать исполнение команд, перед этим адресом должен стоять знак равенства. Адрес можно указывать в любой допустимой форме (6.05-01). Если вместо адреса указано только смещение, то сегментный адрес по умолчанию будет взят из регистра CS:. Второй параметр – количество подлежащих исполнению команд. Если количество не указано, по умолчанию будет исполнена одна команда. Если вообще ни один параметр не указан, то будет исполнена одна машинная команда по адресу CS:IP.

После команды "Trace" на экран дисплея выводятся текущие состояния регистров и флагов, а также результат дизассемблирования кода следующей машинной команды.

Примечание1: команду "Trace" не следует применять для исполнения прерываний, вызовов подпрограмм и других машинных команд, которые вовлекают в процесс исполнения фрагменты уже отлаженного машинного кода. Для этого лучше использовать команду "Proceed" (6.05-14).

6.05-18 Отладчик DEBUG.EXE: команда "Unassemble"

Команда "Unassemble" (= дизассемблировать) переводит машинные коды команд в форму ассемблерных команд и показывает их на экране. В командной строке отладчика DEBUG.EXE вызов команды "Unassemble" задается буквой "U":

```
U 014B L10
```

Первый параметр команды "Unassemble" – стартовый адрес того фрагмента машинного кода, который надлежит перевести. Адрес можно указывать в любой допустимой форме (6.05-01). Если указано только смещение, то сегментный адрес по умолчанию будет взят из регистра CS:. Второй параметр означает либо конечное смещение того же фрагмента (если ему не предшествует буква "L"), либо его длину, в данном примере – длину 10h (= 16) байт. Когда параметры при команде "Unassemble" вообще не указаны, тогда переводятся 20h байт, начиная с текущего смещения относительно регистра CS:. При каждом исполнении текущее смещение увеличивается на длину дизассемблированного фрагмента, так что каждое следующее исполнение команды "Unassemble" без параметров показывает результат дизассемблирования следующего фрагмента машинного кода.

Важно иметь ввиду, что команда "Unassemble" не отличает исполняемый машинный код от других данных, и не способна находить первый байт в многобайтовых машинных командах. Если дизассемблированию подвергаются данные, или если заданный пользователем стартовый адрес не соответствует первому байту машинной команды, то в результате дизассемблирования получается ерунда. Два примера дизассемблирования при правильном и при неправильном указании стартового адреса показаны на рис. 2.

```
R:\DOS\MS7>debug.exe  fit.com
-u181 L15
197C:0181 8E06FC00      MOV     ES,[00FC]
197C:0185 BA5F03        MOV     DX,035F
197C:0188 8CC0          MOV     AX,ES
197C:018A 26            ES:
197C:018B 3B061600      CMP     AX,[0016]
197C:018F 7419          JZ      01AA
197C:0191 26            ES:
197C:0192 8E061600      MOV     ES,[0016]
-u180 L16
197C:0180 028E06FC      ADD     CL,[BP+FC06]
197C:0184 00BA5F03      ADD     [BP+SI+035F],BH
197C:0188 8CC0          MOV     AX,ES
197C:018A 26            ES:
197C:018B 3B061600      CMP     AX,[0016]
197C:018F 7419          JZ      01AA
197C:0191 26            ES:
197C:0192 8E061600      MOV     ES,[0016]
-
```

Рис. 2

Обратите внимание, что при неправильном указании стартового адреса (0180h) две первых машинных команды (0180h и 0184h) дизассемблируются неверно, но начиная со смещения 0188h "фаза" дизассемблирования все-таки устанавливается правильно. Стохастический процесс установления занимает до 10h байт; далее его влияние на дизассемблирование команд обычно не распространяется.

Правильность дизассемблирования адресов в командах коротких переходов зависит от начального адреса размещения кода анализируемой программы, потому что машинные коды этих команд содержат не адрес перехода, а лишь приращение смещения относительно регистра IP. В ходе дизассемблирования эти приращения преобразуются в адреса путем суммирования с текущим смещением, которое зависит от начального размещения анализируемой программы, тогда как значения смещения в других командах не преобразуются. Во избежание вызываемой этим путаницы любой исполняемый код при дизассемблировании всегда должен быть размещен в памяти точно так же, как его следует размещать для исполнения, то есть начиная с адреса CS:0000h для драйверов и с адреса CS:0100h для исполняемого кода программ с суффиксами *.COM и *.EXE.

Интерпретация команд при дизассемблировании, конечно, должна быть идентична их интерпретации процессором. Но процессор может интерпретировать одни и те же машинные коды по-разному в зависимости от состояния бита разрядности в дескрипторе сегмента кода (примечание 5 к A.12-2). Кроме того, некоторые коды интерпретируются особым образом только 64-разрядными процессорами. Должное состояние того процессора, для которого предназначен анализируемый машинный код, отладчик DEBUG.EXE выяснять "не умеет". Он дизассемблирует только коды, предназначенные для 16-разрядного исполнения. Иные машинные коды предоставлять отладчику бессмысленно.

При дизассемблировании иногда встречаются коды команд, которые отладчик DEBUG.EXE "не знает". В таких случаях лучше обратиться к более "толковому" варианту DEBUG.EXE, упоминаемому в примечании 2 при вводной статье к разделу 6.05. Еще более респектабельные дизассемблеры вроде CV.EXE из комплекта MASM требуют подавать им готовые файлы и не позволяют нахально "влезать" на диск или прямо в память компьютера. Для тех, кому поневоле придется пользоваться старым вариантом DEBUG.EXE, некоторые неизвестные ему коды показаны в приведенной ниже таблице. Ее первый столбец содержит первый байт команды, который отладчик представляет как аргумент инструкции DB (7.01-01) в отдельной строке. Следующие байты, как правило, дизассемблируются неверно, но по коду первого и второго байтов (в первом и втором столбцах таблицы) смысл операции удается понять. Часто этого бывает достаточно. При необходимости по ссылке или по названию команды, указанным в четвертом столбце, можно найти более детальные сведения о ней.

1-байт	2-й байт	Байты данных	Операция, команда
0F	00	1–3	загрузка регистра задачи (LTR)
0F	01	1–3	операции регистров GDTR, IDTR, MSWR
0F	02	1–3	загрузка прав доступа (LAR)
0F	03	1–3	загрузка размера сегмента (LSL)
0F	05		загрузка системных регистров (LOADALL)
0F	2(0–3)		MOV CR, MOV DR (7.03-58, примечание 1)
0F	4(0–F)	1–3	копирование по условию (CMOV)
0F	8(0–F)	2	условные переходы в пределах 64 кбайт
0F	9(0–F)		установление бита по условию (SET)
0F	A(0,8)		PUSH FS, PUSH GS (7.03-69)
0F	A(1,9)		POP FS, POP GS (7.03-67)
0F	A2		идентификация процессора (CPUID)
0F	A(4–7)	1–2	выдвижение битов (SHLD, SHRD)
0F	B(2,4,5)	0–2	загрузка сегментов (LSS, LFS, LGS)
60			копирование AX – DI в стек (PUSHA)
61			выталкивание из стека в AX – DI (POPA)
62		3	проверка границ массива (BOUND)
63		2	приведение прав доступа (ARPL)
6(4,5)			префиксы сегментов FS:, GS: (7.02-01)
66			префикс разрядности операнда (7.02-06)
67			префикс разрядности адреса (7.02-07)
6(8,A)		1–2	загрузка числа в стек (PUSH, 7.03-69)
6(9,B)		1–3	умножение (IMUL, примечание к 7.03-25)
6(C,D)			групповой вывод из порта (INSB, INSW)
6(E,F)			групповая запись в порт (OUTSB, OUTSW)
8(C,E)	E(0–F)		MOV FS, MOV GS (7.03-58, примечание 2)
C(0,1)	E(0–7)	1	SHL bl,0f, SHL bx,0f (7.03-82)
C(0,1)	E(8–F)	1	SHR bl,0f, SHR bx,0f (7.03-83)

6.05-19 Отладчик DEBUG.EXE: команда "Write"

Команда "Write" (= записать) записывает код из памяти компьютера на логический диск либо в виде файла, либо просто в заданные сектора диска. В командной строке отладчика DEBUG.EXE вызов команды "Write" задается буквой "W". Вот пример вызова команды "Write" для записи в файл:

```
W 0100
```

Здесь первый и единственный параметр – адрес, начиная с которого будут считаны данные из памяти. Адрес может быть задан в любой допустимой форме

(6.05-01). Если вместо полного адреса задано только смещение, то сегментный адрес будет взят из регистра CS:. Если адрес вообще не указан, то по умолчанию данные считываются из памяти начиная с адреса CS:0100. Длина записываемого фрагмента, отсчитываемая от начального адреса, должна быть заранее загружена в регистры CX (младшие два байта длины файла) и BX (старшие два байта).

Предполагается, что имя файла уже определено предшествовавшей процедурой загрузки или командой "Name" (6.05-12), и что с тех пор сегментный адрес в регистре DS: не изменился. Если имя файла задано без указания пути, то файл будет создан в текущем каталоге текущего диска. Если файл с таким же именем в каталоге назначения уже существует, то он будет перезаписан без предупреждения. Отладчик DEBUG.EXE отказывается записывать файлы с суффиксами *.HEX и *.EXE, потому что такие файлы должны иметь заголовок, который DEBUG.EXE автоматически формировать не может. Впрочем, если Вы уверены в себе, то файл несложно переименовать потом.

Для осуществления записи в сектора логического диска в строке вызова команды "Write" должны быть указаны четыре обязательных параметра, например:

```
W 0100 0 0 1
```

Здесь первый параметр указывает стартовый адрес считывания кода, как и при записи в файл. Второй параметр задает номер логического диска: 0 – диск A:, 1 – диск B:, 2 – диск C: и т.д. Третий параметр – это шестнадцатеричный номер сектора, с которого следует начинать запись, четвертый параметр – шестнадцатеричное количество записываемых секторов. Длина записываемого фрагмента данных зависит только от количества секторов, содержимое регистров BX:CX не учитывается. Запись в физические сектора за пределами логических дисков командой "W" осуществить нельзя.

Примечание 1: если возникает необходимость вывести исполняемый код в файл из области PSP (A.07-1), то сначала следует либо переместить код в другую область памяти, либо перенаправить запись имени в безопасное место путем изменения сегментного адреса в регистре DS:. Только после этого можно объявлять имя файла командой "Name" и выполнять запись командой "Write" (пример – в разделе 9.08).

6.05-20 Отладчик DEBUG.EXE: команда "Allocate"

Команда "Allocate" (= выделить) обращается к драйверу отображаемой памяти EMM386.EXE (5.04-02, 8.03-59) с запросом об образовании в расширенной памяти, за границей 1088 кбайт, выделенной области адресного пространства и присвоении ей номерной ссылки (handle) – двухбайтового шестнадцатеричного числа,

посредством которого на эту область можно ссылаться. Чтобы такую операцию провести, нужно, во-первых, иметь компьютер с объемом памяти более одного мегабайта и, во-вторых, заранее загрузить драйвер EMM386.EXE. В командной строке отладчика DEBUG.EXE вызов команды "Allocate" задается буквами "XA", например:

```
XA 1A
```

В строке вызова команды "Allocate" должен быть один обязательный параметр: запрашиваемый размер выделяемой области, выраженный в количестве логических страниц, по 16 кбайт каждая. В показанном примере запрошен размер области 1Ah, то есть 26 логических страниц. Отладчик DEBUG.EXE в ответ сообщает, например: "Handle created 0006" (= для ссылок выделен номер 0006h). Логические страницы памяти нумеруются последовательно, так что показанная команда "XA 1A" создает область, содержащую логические страницы с номерами от 00h до 19h. Чтобы получить доступ к этим страницам памяти, нужно еще отобразить логические страницы на физическое адресуемое пространство с помощью команды "Map" (6.05-22).

6.05-21 Отладчик DEBUG.EXE: команда "Deallocate"

Команда "Deallocate" (= расформировать) обращается к драйверу отображаемой памяти EMM386.EXE (5.04-02, 8.03-61) с запросом о расформировании ранее выделенной области расширенной памяти. Такой запрос имеет смысл только на компьютерах с объемом памяти свыше одного мегабайта, когда драйвер EMM386.EXE заранее установлен, и когда области памяти, которую надлежит расформировать, уже присвоена номерная ссылка (handle). В командной строке отладчика DEBUG.EXE вызов команды "Deallocate" задается буквами "XD", например:

```
XD 0006
```

В командной строке вызова команды "Deallocate" должен быть один обязательный параметр: номерная ссылка (handle), присвоенная той выделенной области, которую надлежит расформировать. В данном примере будет расформирована выделенная область расширенной памяти, которой была присвоена номерная ссылка 0006h. Отладчик DEBUG.EXE ответит: "Handle 0006 deallocated" (= номерная ссылка 0006h аннулирована). Занятое выделенной областью адресное пространство будет считаться свободным. Доступ к логическим страницам памяти в расформированной области утрачивается.

6.05-22 Отладчик DEBUG.EXE: команда "Map"

Команда "Map" (= отобразить) обращается к драйверу отображаемой памяти EMM386.EXE (5.04-02, 8.03-60) с запросом об отображении 16-килобайтной логической страницы в заранее выделенной области расширенной памяти на "физическую" страницу памяти такого же размера. Здесь под термином "отображение" имеется ввиду такая настройка механизма преобразования адресов в центральном процессоре, которая обеспечит преобразование обычных 16-разрядных адресов, относящихся к адресному пространству "физической" страницы, в 32-разрядные адреса реальных ячеек памяти, находящихся за границей обыкновенной памяти на том участке, который поставлен в соответствие запрошенной логической странице. Естественно, запрос об отображении логической страницы имеет смысл только на компьютерах с 32-разрядным процессором и с объемом памяти свыше одного мегабайта, когда драйвер EMM386.EXE заранее установлен, и когда выделенной области расширенной памяти, в которой находится запрашиваемая логическая страница, уже присвоена номерная ссылка (handle). В командной строке отладчика DEBUG.EXE вызов команды "Map" задается буквами "XM", например:

```
XM 0B 03 0006
```

В командной строке вызова команды "Map" должны быть указаны три обязательных параметра. Первый параметр – в данном примере 0Bh – номер запрашиваемой логической страницы в выделенной области расширенной памяти. Второй параметр – в данном примере 03h – номер физической страницы. Третий параметр – в данном примере 0006h – номерная ссылка, присвоенная той выделенной области расширенной памяти, в которой находится запрашиваемая логическая страница.

Номера физических страниц можно выбирать в пределах 00h – 1Bh, но нужно помнить, что физические страницы с номерами от 04h и выше располагаются в обыкновенной памяти ниже 640 кбайт. Потому наиболее активно используются физические страницы 00h – 03h, располагаемые по умолчанию в сегментных адресах E000h, E400h, E800h и EC00h соответственно. Однако запросы системы BIOS компьютера и параметры установки драйвера EMM386.EXE могут повлиять на размещение физических страниц в адресном пространстве, так что в каждом конкретном случае его следует проверять командой "Show" (6.05-23) или посредством вызова обработчика прерывания INT 67\AX=5800h (8.03-70).

6.05-23 Отладчик DEBUG.EXE: команда "Show"

Команда "Show" (= показать) обращается к драйверу отображаемой памяти EMM386.EXE (5.04-02, 8.03-70) с запросом предоставить сведения об

использовании расширенной памяти и о размещении физических страниц. Для пользования командой "Show" необходимо, чтобы компьютер располагал памятью свыше одного мегабайта и чтобы драйвер EMM386.EXE был заранее установлен. В командной строке отладчика DEBUG.EXE вызов команды "Show" задается буквами "XS" без параметров:

```
XS
```

В ответ отладчик DEBUG.EXE выводит таблицу, в начале которой приведены сведения о том, сколько логических страниц ассоциировано с каждой номерной ссылкой. Затем показаны сегментные адреса, назначенные каждой физической странице. Последние две строки выводимого на экран сообщения показывают статистику: общее число логических страниц в расширенной памяти, число свободных логических страниц, максимальное число номерных ссылок (handles) и сколько номерных ссылок уже выделено.

Однако таблица получится слишком длинной и выйдет за рамку экрана, если в нее будут включены сведения о всех 28 страницах, предусмотренных спецификацией EMS 4.0. Чтобы просмотреть таблицу полностью, нужно либо перенаправить выводимое сообщение в текстовый файл (2.04-03), либо установить видеорежим 108h (A.10-1), или уменьшить число физических страниц посредством изменения параметров драйвера EMM386.EXE (5.04-02).

6.06 DISKCOPY.COM – копирование дискет

Программа DISKCOPY.COM копирует содержание одной дискеты на другую, при условии, что обе эти дискеты одного типа. Содержание копируется полностью, включая метку тома и серийный номер. Вот пример пользования программой DISKCOPY.COM:

```
DISKCOPY.COM  A: B: /V
```

здесь:

A: – пример спецификации первого дисковода, который содержит копируемую дискету.

B: – пример спецификации второго дисковода с дискетой, на которую должно производиться копирование.

/V – необязательный параметр, вызывающий выверку изготовленной копии относительно оригинала.

Помимо показанных параметров, в командной строке могут быть следующие:

/1 – необязательный параметр, вызывающий копирование только одной стороны дискеты (для устаревших односторонних дискет)

/M – необязательный параметр, вызывающий многозаходное копирование по частям с сохранением частей в памяти, без создания буферного файла на жестком магнитном диске.

Допускается указывать один и тот же дисковод как в качестве источника, так и в качестве дисковода-приемника. В этом случае программа DISKCOPY.COM создает временный буферный файл в каталоге, указанном в переменной окружения %TEMP%, и после его создания предлагает заменить копируемую дискету той, на которую должно производиться копирование. По завершении копирования временный буферный файл автоматически уничтожается. Но когда переменная %TEMP% не определена, или если указан параметр /M, то копирование происходит по частям согласно имеющемуся свободному пространству памяти, так что пользователю приходится по нескольку раз менять дискеты в одном и том же дисководе.

Формирование постоянного файла-образа дискеты программой DISKCOPY.COM не предусмотрено. Кроме того, программа DISKCOPY.COM "не умеет" разумно обращаться с дискетами, содержащими поврежденные секторы. Поэтому сейчас предпочтение отдают другим программам копирования, не обладающим перечисленными недостатками. Можно рекомендовать, например, программу IMG.EXE, которая свободно выложена в сети Интернет на сервере ftp://ftp.elf.stuba.sk/pub/pc/utildisk/ в составе файла архива IMG.ARJ.

Примечание 1: программу DISKCOPY.COM и ее аналоги нельзя применять для копирования жестких магнитных дисков, оптических и сетевых дисков, а также виртуальных дисков, создаваемых программами ASSIGN.COM, SUBST.EXE и JOIN.EXE.

6.07 DOSKEY.COM – менеджер командной строки

Резидентная программа DOSKEY.COM дополняет функции командного интерпретатора по обслуживанию командной строки. Программа DOSKEY.COM расширяет возможности редактирования командной строки, повторного вызова команд, позволяет создавать и исполнять макрокоманды, содержащие простые последовательности операций. Определяемые макрокомандами процедуры начинают действовать после загрузки их спецификаций в специальный буфер, организуемый в памяти программой DOSKEY.COM. Помимо этого буфера, резидентный модуль программы DOSKEY.COM занимает в памяти около 4 кбайт.

Программа DOSKEY.COM в MS-DOS7 усовершенствована: она получила способность изменять размер буфера клавиатуры, буфера командной строки, а также вводить список определений макрокоманд из текстового файла.

Программу DOSKEY.COM можно загружать из командной строки или из строки файла AUTOEXEC.BAT, непосредственно или с помощью команды LH, например

```
LH DOSKEY.COM /bufsize:1024 /insert /file:C:\DOS\MS7\Macro.scr
```

здесь:

/bufsize:1024 – необязательный параметр, определяющий размер буфера макрокоманд. Минимальный размер буфера 256 байт, по умолчанию принимается 512. Этот параметр следует использовать только при начальной загрузке и перезагрузке. Свободная часть данного буфера используется для хранения предыдущих командных строк ("истории").

/insert – необязательный параметр, переводящий командную строку в режим вставки знаков, позволяющий вставлять новые знаки между теми, которые набраны ранее. По умолчанию устанавливается режим повторного набора (overstrike), при котором новые знаки набираются поверх прежних.

/file:C:\DOS\MS7\Macro.scr – пример спецификации для загрузки определений макрокоманд из файла (пример показан далее). Путь к этому файлу можно не указывать, если он находится в текущем каталоге, но суффикс, если он имеется, должен быть указан обязательно.

При первичной загрузке и перезагрузке можно дополнительно указывать следующие необязательные параметры:

/echo:off – не выводить на экран командную строку исполняемых макрокоманд.

/keysize:31 – увеличить размер буфера клавиатуры до 31 знака, тогда как по умолчанию он составляет 15 знаков.

/line:256 – установить размер буфера для редактирования командной строки. По умолчанию его размер – 128 знаков.

/reinstall – повторно установить резидентный модуль программы DOSKEY.COM. Каждая повторная установка занимает дополнительно 4 кбайт памяти (прежний резидентный модуль при этом не выгружается). Повторная установка позволяет без перезагрузки DOS изменять размер буферов и восстанавливать работоспособность DOSKEY.COM, нарушенную из-за перехвата функций клавиатуры другими программами.

Когда DOSKEY.COM уже загружен, его можно вызывать с параметрами /H и /M, чтобы показать предыдущие командные строки (историю):

```
DOSKEY /H
```

или чтобы показать действующие макрокоманды:

```
DOSKEY /M
```

Кроме того, программу DOSKEY.COM можно вызывать для загрузки определения еще одной макрокоманды из командной строки, например, так:

```
DOSKEY count=C:\DOS\COM\Find.exe /v /c "" $1
```

Слово "count" здесь будет интерпретировано как имя загружаемой макрокоманды. Если потом это имя набрать сразу же после приглашения командной строки, то произойдет исполнение команды, указанной в определении макрокоманды справа от знака равенства. В данном случае это определение включает одну команду, выполняющую подсчет числа строк любого текстового файла, который здесь представлен формальным параметром $1. При исполнении макрокоманды формальный параметр будет замещен именем реального файла, указанным после имени макрокоманды в командной строке.

Команды в составе определений макрокоманд могут содержать знаки равенства и подстановки переменных окружения (например, %Temp%). Специальная роль в определениях макрокоманд отведена знаку "$": он интерпретируется совместно со следующей буквой или цифрой, и эта пара знаков вместе замещается:

$G	– знаком правой стрелки ">" (перенаправления выхода);
$L	– знаком левой стрелки "<" (перенаправления ввода);
$B	– знаком вертикального штриха "|" (промежуточного перенаправления);
$T	– разделителем, позволяющим указывать несколько команд в одной строке макрокоманды;
$1 - $9	– формальными параметрами (как %1 - %9 в batch-файлах);
$*	– сразу всеми словами, указанными после имени макрокоманды в командной строке;
$$	– одним знаком доллара "$".

Командный файл для загрузки определений макрокоманд должен содержать по одному определению в каждой строке и может выглядеть, например, так:

```
count=C:\DOS\COM\Find.exe /v /c "" $1
newbat=echo @echo off$G %Temp%\Trial.bat $T Edit.com %Temp%\Trial.bat
```

Уже после загрузки макрокоманд в буфер памяти любую из них можно из буфера удалить, указав после имени макрокоманды только знак равенства, например:

```
DOSKEY count=
```

Когда DOSKEY.COM загружен, он активизирует следующие "горячие клавиши":

Левая и правая стрелки	– сдвигают курсор влево и вправо
CTRL-левая стрелка	– сдвигает курсор на одно слово влево
CTRL-правая стрелка	– сдвигает курсор на одно слово вправо
HOME	– сдвигает курсор к началу командной строки
END	– сдвигает курсор к концу командной строки
INS	– переключает режимы вставки и повторного набора
ESC	– удаляет все из командной строки
ALT-F10	– удаляет все макрокоманды из буфера памяти
F7	– показывает предыдущие команды (историю)
PageUp	– показывает старейшую из предыдущих команд
PageDown	– показывает последнюю из предыдущих команд
стрелки вверх и вниз	– выбирают одну из предыдущих команд
F9	– выбирает одну из прошлых команд по номеру
Alt+F7	– удаляет все из списка предыдущих команд
F8	– вызывает команду из списка предыдущих команд по совпадению ее начальных знаков с теми, которые набраны в командной строке.

Примечание 1: макрокоманда не будет исполнена, если ее имени в командной строке предшествует хотя бы один пробел.

Примечание 2: если имя макрокоманды совпадает с именем какой-либо команды, то исполняться будет именно макрокоманда.

Примечание 3: программа DOSKEY.COM несовместима с файл-менеджерами Norton Commander, Volcov Commander, и т. п., причем предпочтение обычно отдается файл-менеджерам.

6.08 DELTREE.EXE – удаление каталогов

Программой DELTREE.EXE следует пользоваться очень осторожно, потому что она позволяет удалять каталоги вместе со всеми содержащимися в них файлами и подкаталогами, причем не обращая внимания на атрибуты файлов. Только корневые каталоги дисков удалить программой DELTREE.EXE нельзя. Вот пример вызова этой программы:

```
DELTREE /Y C:\TEMP\TDIR1
```

здесь:

/Y – необязательный параметр, разрешающий исполнение удалений без предупреждения

C:\TEMP\TDIR1 – пример спецификации подлежащего удалению каталога. В одной строке можно указывать несколько таких

спецификаций, но использовать в них знаки подстановки не разрешается.

6.09 EDIT.COM – текстовый редактор

EDIT.COM – популярная программа редактирования преимущественно текстовых файлов, предоставляющая также ограниченные возможности редактирования файлов, содержащих данные и исполняемый код. Редактор EDIT.COM позволяет работать с одним или с двумя "окнами" редактирования, и при этом держать открытыми до десяти редактируемых файлов. На рис. 3 показан внешний вид двух "окон" редактора EDIT.COM с открытыми в них текстовыми файлами.

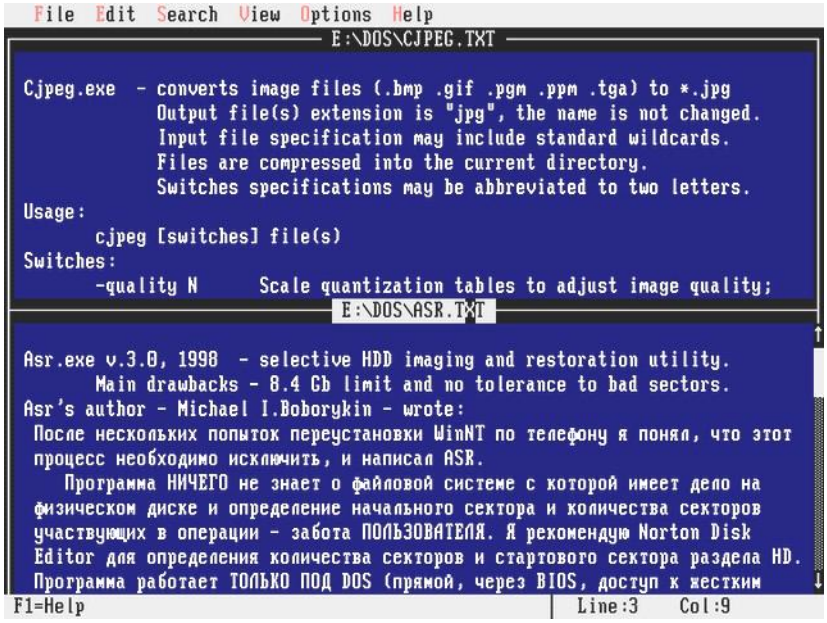


Рис. 3

Размеры редактируемых файлов ограничены только имеющейся в компьютере памятью. Редактор EDIT.COM создает буфер обмена, позволяющий пересылать фрагменты текста из любого редактируемого файла в любой другой редактируемый файл. Помимо прочего, редактор EDIT.COM успешно взаимодействует с драйверами манипуляторов типа "мышь", благодаря чему процесс редактирования становится намного проще и быстрее.

Редактор EDIT.COM можно вызывать из командной строки без параметров, и тогда подлежащие редактированию файлы нужно будет открывать через меню "File". Но можно указать подлежащие редактированию файлы сразу в командной строке. Вот пример вызова редактора EDIT.COM для редактирования двух файлов:

```
EDIT.COM  C:\DOS\Addons.txt Part6.txt
```

Имени первого файла в этом примере предшествует путь; без указания пути редактор EDIT.COM открывает только файлы, находящиеся в текущем каталоге. Так, второй файл, перед именем которого путь не указан, должен находиться в текущем каталоге. По умолчанию оба файла открываются в текстовом режиме редактирования, причем только один первый файл раскрывается в показываемом на экране "окне". Пользователю предоставляется самому открыть второе "окно" нажатием клавишной комбинации CTRL-F6 и перейти к редактированию другого файла по нажатию клавиши F8.

Установки параметров редактора EDIT.COM и пояснительные тексты хранятся в отдельных файлах EDIT.INI и EDIT.HLP, которые должны находиться в том же каталоге. Однако редактор EDIT.COM может обходиться и без упомянутых файлов, используя установки, принимаемые по умолчанию. Когда файл EDIT.INI отсутствует, пользователь может установить параметры посредством меню OPTIONS, и тогда редактор EDIT.COM автоматически воссоздаст файл EDIT.INI.

Немногие клавиши в редакторе EDIT.COM действуют так же, как в командной строке DOS. Так же действует клавиша Backspace, так же после нажатия клавиши ALT можно ввести код ASCII цифровыми клавишами в правой части клавиатуры, и это вызовет появление на экране соответствующего знака. Но большинство клавиш действует по-другому. Нажатие клавиши ALT активизирует буквенные "горячие клавиши" для выбора пунктов меню в верхней части экрана. При нажатой клавише ALT нажатие цифровых клавиш 0 – 9 в основной части клавиатуры вызывает переключение показываемых в "окне" открытых файлов по их номеру, если, конечно, для редактирования открыто несколько файлов. Стрелочки перемещения курсора позволяют свободно перемещать его по всему окну редактирования. Если при перемещении курсора держать нажатой клавишу SHIFT, то произойдет выделение фрагмента текста для его последующего удаления или копирования в буфер обмена. Выделять фрагмент текста удобно также с помощью "мыши", придерживая нажатой ее левую клавишу.

Помимо перечисленных выше функций, редактор EDIT.COM задействует следующие "горячие" клавиши и клавишные комбинации:

CTRL-C	– копирует выделенный фрагмент текста в буфер обмена; клавишная комбинация CTRL-INS действует так же.
CTRL-F4	– закрывает активное окно (если открыты оба окна)
CTRL-F6	– разделяет окно редактирования на два (если первоначально было открыто только одно окно)
CTRL-F8	– позволяет изменить размеры окон редактирования (если открыты оба окна)
CTRL-Home	– смещает курсор на начало файла
CTRL-End	– смещает курсор на конец файла

CTRL-P	– запись специальных знаков последующим нажатием клавиши CTRL и вводом цифрового кода знака
CTRL-PageDown	– смещает окно просмотра на размер строки вправо
CTRL-PageUp	– смещает окно просмотра на размер строки влево
CTRL-Q	– выводит внизу экрана подсказку, позволяющую затем выполнять операции поиска, замены и удаления
CTRL-Space	– удаляет выделенный фрагмент текста без занесения его в буфер обмена
CTRL-V	– вставляет фрагмент из буфера обмена в редактируемый текст согласно текущей позиции курсора
CTRL-W	– смещает показываемый в окне текст на одну строку вниз (комбинация CTRL-стрелка вверх действует так же)
CTRL-X	– удаляет выделенный фрагмент текста в буфер обмена
CTRL-Y	– удаляет указываемую курсором строку, даже если она не выделена
CTRL-Z	– смещает текст в окне текст на одну строку вверх (комбинация CTRL-стрелка вниз действует так же)
Delete	– удаляет выделенный фрагмент текста без занесения его в буфер обмена. Если выделенных фрагментов нет, то удаляется один знак справа от курсора.
F3	– вызывает поиск следующего элемента по спецификации, уже введенной клавишной комбинацией CTRL-Q-F.
F6	– переключает открытые окна: активное и пассивное.
F8	– переключает активное окно на следующий по порядку открытый файл (когда открыты несколько файлов).
PageUp	– смещает курсор на одну страницу текста вверх
PageDown	– смещает курсор на одну страницу текста вниз

Некоторые из перечисленных клавишных комбинаций указаны в пунктах раскрывающихся меню, названия которых приведены на верхней строке экрана. Сами эти меню, а также их отдельные пункты можно выбирать не только с клавиатуры, но и нажатием левой клавиши манипулятора "мышь".

Вот еще один пример вызова редактора EDIT.COM для редактирования не-текстовых файлов:

```
EDIT.COM /78 Sc_sct.dat
```

здесь

/78 – параметр, вызывающий переход в режим редактирования не-текстовых файлов с ограничением длины строки до 78 знаков. Вместо 78 можно указывать любое число до 255, но 78

соответствует фактической ширине окна редактирования в наиболее популярном текстовом режиме дисплея 80x25.

Допускается также вызывать редактор EDIT.COM с указанием в командной строке следующих параметров:

/B – отключить цвет, перейти к индикации в черно-белом виде
/H – перевести дисплей в режим с наибольшим числом строк
/R – открыть файл только для чтения, запретить редактирование
/S – воспринимать укороченные версии длинных имен файлов
/? – показать краткую справку.

6.10 EXPAND.EXE – распаковщик сжатых файлов

Программа EXPAND.EXE распаковывает файлы, сжатые программой COMPRESS.COM фирмы Microsoft. Они выделяются тем, что в их имени последняя буква суффикса заменена знаком подчеркивания. Каждый такой сжатый файл содержит только один исходный файл. Файлы, сжатые программой COMPRESS.COM, входили в поставку предыдущих версий MS-DOS и ряда других программных пакетов, причем обычно вместе с программой распаковки EXPAND.EXE. Вот пример пользования программой EXPAND.EXE:

```
EXPAND.EXE E:\DOS\MSDOS622\Country.tx_ C:\DOS\MS6\Country.txt
```

Первое имя (Country.tx_) принадлежит сжатому файлу, второе имя (Country.txt) будет присвоено распакованному файлу. Оба имени указаны с предшествующим путем. Если путь не указывать, то поиск сжатого файла будет производиться только в текущем каталоге, и результат распаковки тоже будет помещен в текущий каталог, но это в данном случае допустимо, так как сжатый и распакованный файлы имеют разные имена.

Использовать знаки подстановки (2.01-03) в строке вызова программы EXPAND.EXE нельзя, но можно указывать подряд несколько сжатых файлов. При этом последним в строке должно быть указано имя каталога (без конечной обратной косой черты), куда будет происходить распаковка. Если распаковка производится в текущий каталог, он может быть представлен знаком точки.

Файлы, сжатые старыми версиями программы COMPRESS.COM, не содержат сведений о прежнем имени подвергнутого сжатию файла. Это не вызывает проблем, пока такие файлы распаковываются поодиночке с явным указанием нового имени для каждого распакованного файла. Но если группа таких файлов распаковывается в общий каталог, то имена распакованным файлам сменены не будут. В таком случае нельзя выполнять распаковку в тот же каталог, где находятся сжатые файлы, во избежание возникновения конфликта имен.

Примечание 1: у сжатых файлов из комплектов поставки операционных систем Windows-2000\XP последняя буква суффикса также заменена на знак подчеркивания, но они сформированы посредством иного алгоритма сжатия. Для их распаковки в среде MS-DOS7 следует использовать программу EXTRACT.EXE (6.11).

6.11 EXTRACT.EXE – распаковщик сжатых файлов

Программа EXTRACT.EXE распаковывает файлы, сжатые программой MAKECAB.EXE. Фирма Microsoft использует этот алгоритм для сжатия одиночных файлов в поставках операционных систем Windows-2000/XP, а также для компоновки больших многотомных сжатых файлов с суффиксом *.CAB в поставках операционных систем Windows-95/98/ME. Помимо собственно распаковки, программа EXTRACT.EXE позволяет просматривать перечень содержащихся в поставке файлов, а также извлекать отдельные файлы из сжатых файлов с суффиксом *.CAB. Вот пример использования программы EXTRACT.EXE для просмотра содержания многотомной поставки:

```
EXTRACT.EXE /A /D E:\Win95\OSR2.PE\Win95_21.cab > C:\Temp\List.txt
```

здесь:

E:\Win95\OSR2.PE\ – пример пути к одному из CAB-файлов из поставки операционной системы Windows-95. Если путь не указывать, то поиск файла будет производиться только в текущем каталоге.

/D – необязательный параметр, вызывающий выведение таблицы содержания CAB-файла без его распаковки.

/A – необязательный параметр, вызывающий продолжение исполнения той же операции по отношению ко всем последующим CAB-файлам из той же поставки. В данном случае это означает, что будет выведено содержание CAB-файлов с 21-го по 26-й.

Как правило, в CAB-файлах содержится в сжатом виде множество исходных файлов, список содержания получается весьма длинным, внимательно просмотреть его в процессе вывода на экран фактически невозможно. Поэтому в приведенном выше примере выводимый список перенаправлен в файл List.txt в каталоге C:\Temp. Потом файл List.txt можно спокойно посмотреть с помощью программы просмотра (например, MORE.COM, 6.19) или текстового редактора.

Вот другой пример использования программы EXTRACT.EXE для извлечения нужного файла (в данном случае – Msvcrt40.dll) из сжатого CAB-файла, входящего в комплект поставки операционной системы, причем когда заранее неизвестно, в каком именно CAB-файле содержится подлежащий извлечению файл:

```
EXTRACT.EXE /A /Y /L C:\Windows\System Win95_02.cab Msvcrt40.dll
```

здесь:

/Y – необязательный параметр, позволяющий перезаписать без предупреждения одноименный файл в каталоге назначения при размещении там распаковываемого файла.

/L – необязательный параметр, указывающий на то, что приводимый далее путь (C:\Windows\System) следует считать путем к каталогу назначения, куда надо поместить распаковываемый файл. Когда параметр /L не указан, распаковываемый файл по умолчанию помещается в текущий каталог.

После спецификации пути к каталогу назначения в подобных командных строках может быть указано несколько имен файлов, причем первое из них относится к CAB-файлу, начиная с которого должен производиться поиск, а все последующие интерпретируются как имена файлов, которые надлежит извлечь и распаковать. При исполнении показанной выше командной строки начнется процедура поиска подлежащего извлечению файла в сжатых CAB-файлах с 02-го по 26-й. Искомый файл будет обнаружен в 13-м томе поставки, распакован и помещен в каталог C:\WINDOWS\SYSTEM, затерев при этом имевшуюся там прежнюю версию файла. Такие процедуры распаковки занимают гораздо меньше времени, чем полная распаковка поставки операционной системы, и часто используются для замещения поврежденных системных файлов.

Программа EXTRACT.EXE не позволяет использовать в именах файлов знаки подстановки (2.01-03), но принимает еще один необязательный параметр /E , который вызывает распаковку всех файлов, содержащихся в указанном сжатом файле. При этом конкретное имя извлекаемого файла указывать уже не нужно. Параметр /E целесообразно применять по отношению к одиночным сжатым файлам из поставки Windows-2000/XP (у которых последняя буква суффикса заменена знаком подчеркивания). Но полная распаковка больших CAB-файлов в среде MS-DOS выполняется медленно. Особенно осторожно следует относиться к совместному использованию параметров /A и /E, потому что это может повлечь за собой распаковку большой группы сжатых CAB-файлов, требующую много времени и значительного объема дискового пространства.

Примечание 1: некоторые поставщики программного обеспечения используют суффикс *.CAB в именах файлов, сжатых посредством других алгоритмов. Кроме того, существует несколько взаимно несовместимых версий программы EXTRACT.EXE. В любом случае для распаковки CAB-файлов из поставки программного обеспечения с наибольшей вероятностью подойдет именно та программа EXTRACT.EXE, которая может содержаться в составе той же поставки.

6.12 FC.EXE – сопоставление файлов

Программа FC.EXE позволяет сопоставлять файлы в бинарном и в текстовом режимах. Бинарный режим сопоставления целесообразно использовать для нахождения отдельных несовпадающих байтов в почти одинаковых исполняемых файлах. FC.EXE показывает каждую несовпадающую пару байтов из сопоставляемых файлов вместе с их смещением относительно начала файла. Вот пример вызова программы FC.EXE для сопоставления файлов в бинарном режиме:

```
FC.EXE /B Trial2.com D:\Temp\Trial1.com
```

здесь:

- /B – параметр, задающий сопоставление в бинарном режиме
- Trial2.com – пример спецификации первого сопоставляемого файла. Поскольку перед его именем не указан путь, предполагается наличие этого файла в текущем каталоге.
- D:\Temp\Trial1.com – пример спецификации второго сопоставляемого файла с указанием полного пути.

Бинарный режим сопоставления не предполагает поиска соответствий во взаимно смещенных последовательностях байтов.

В текстовом режиме сопоставления FC.EXE сравнивает файлы строка за строкой и показывает рядом несовпадающие строки. Когда порядок соответствия между последовательностями строк нарушается, FC.EXE способен его восстановить посредством поиска групп совпадающих строк. Условия сопоставления и поиска соответствий определяются необязательными параметрами, задаваемыми в командной строке, например:

```
FC.EXE /A /C /L /LB9 /N /T /W /1 A:\Config.sys C:\Config.sys
```

здесь:

- /A – показывать только две строки (первую и последнюю) от каждой группы несовпадающих строк
- /C – считать строчные и заглавные буквы одинаковыми (только в первой половине таблицы кода ASCII, до знака 127).
- /L – сопоставлять файлы в текстовом режиме, как текст в коде ASCII.
- /LB9 – пример задания ограничения (9 последовательных строк) на протяженность зоны поиска совпадений. По умолчанию протяженность этой зоны составляет 100 строк.
- /N – показывать номера строк
- /T – не преобразовывать код табуляции 09h в последовательность пробелов.

/W – сжать пустое пространство строки, выражаемое знаками табуляции и пробелами, и не учитывать их количество при сопоставлении.

/1 – пример задания числа строк, которые должны совпасть, чтобы можно было бы считать, что совпадение снова найдено.

Примечание 1: бинарный режим сопоставления устанавливается по умолчанию для файлов, имена которых имеют суффиксы BIN, COM, EXE, LIB, OBJ, SYS. Все остальные файлы по умолчанию сопоставляются в текстовом режиме.

Примечание 2: в специальных случаях допускается замена одного или обоих имен файлов зарезервированными именами устройств: NUL, CON и т.д. (2.01-01). В частности, указание устройства CON (консоли) вместо имени одного из файлов приводит к исполнению сопоставления с текстом, вводимым с клавиатуры (1.04).

6.13 FDISK.EXE – разметка разделов диска

Диски, знакомые нам по буквенным обозначениям, принято называть логическими дисками. В отличие от них, физические накопители на жестких магнитных дисках буквами не обозначают. Там физическое записываемое пространство может быть представлено состоящим из нескольких разделов, каждому из которых поставлен в соответствие отдельный логический диск. Для разметки структуры разделов на жестких магнитных дисках в составе MS-DOS7 имеется программа FDISK.EXE.

Когда программу FDISK.EXE запускают на исполнение без параметров, она прежде всего запрашивает пользователя, нужно ли вводить поддержку дисков большого объема или нет. Отказ будет означать, что новые разделы объемом от 512 до 2048 Мбайт будут маркированы для последующего форматирования согласно файловой системе FAT-16. Принятие предложения о поддержке дисков большого объема означает, что предпочтение будет отдано файловой системе FAT-32. Для разделов, размер которых выходит за указанные выше границы, маркировка разделов выполняется по умолчанию: FAT-32 принимается для разделов свыше 2048 Мбайт, FAT-16 – для разделов от 16 до 512 Мбайт, FAT-12 – для разделов, емкость которых не превышает 16 Мбайт.

Тем, кто в большей степени полагается на себя, целесообразно сразу указывать в командной строке параметры, снимающие ненужные вопросы и ограничения:

```
FDISK.EXE /fprmt /actok
```

Параметр /fprmt предоставляет пользователю самому выбирать тип файловой системы, устраняет разговоры о поддержке дисков большого объема и сразу

выводит на экран меню выбора операции: показать, создать, удалить раздел или сделать его загрузочным (активным). Параметр /actok позволяет создать загрузочный раздел на любом физическом диске (а не только на первом). Пользователь получает возможность составлять структуру разделов диска, но эта структура не записывается на диск сразу. Программа FDISK.EXE дает шанс вернуться к меню для корректирования построенной структуры. Если с размечаемого жесткого магнитного диска предстоит загружать компьютер, то важно не забыть, что один из разделов должен быть сделан загрузочным. После этого можно завершить сеанс работы с программой FDISK.EXE.

Если в ходе сеанса структура разделов была изменена, то программа FDISK.EXE записывает изменения на диск и выводит компьютер в перезагрузку, чтобы внесенные измерения были бы зарегистрированы системой BIOS компьютера. При регистрации разделов системой BIOS им присваиваются буквенные обозначения логических дисков, после чего эти логические диски можно форматировать программой FORMAT.COM (6.15). Только после форматирования новые логические диски становятся доступны для использования.

Помимо описанного интерактивного составления структуры разделов, программа FDISK.EXE способна выполнять отдельные операции, задаваемые параметрами в командной строке:

`FDISK.EXE /?` – показать краткую справку.

`FDISK.EXE /status` – показать имеющиеся дисководы на жестких магнитных дисках и распределение по ним логических дисков.

`FDISK.EXE /mbr` – записать или перезаписать MBR (master boot record = главную загрузочную запись) на физическом дисководе номер 1, оставляя структуру разделов без изменений (примечание 1). Подтверждающее сообщение о перезаписи не выдается.

`FDISK.EXE /cmbr 2` – записать или перезаписать MBR на физическом дисководе, номер которого указан после параметра /cmbr .В данном случае указан номер 2, допускаются 1, 2, 3 и т.д., если, конечно, эти дисководы в компьютере имеются. Подтверждающее сообщение о перезаписи не выдается.

При любом варианте использования программы FDISK.EXE в командной строке может быть дополнительно указан параметр /X – не применять функции расширенной поддержки доступа к жестким магнитным дискам. Параметр /X следует указывать тогда, когда обычные попытки разметки диска заканчиваются неудачно: диск либо не опознается, либо оказывается недоступен, либо попытка доступа ведет к зависанию компьютера с сообщением "stack overflow" (=

переполнение стека). Такое бывает из-за отказов аппаратуры, из-за неправильного конфигурирования, из-за заражения служебных областей диска компьютерным вирусом. Не всегда, но в некоторых подобных ситуациях указание параметра /X может помочь.

Программа FDISK.EXE способна также осуществлять разметку дисководов на жестких магнитных дисках в автоматическом режиме. Это удобно, когда требуется подготовить к работе несколько компьютеров с одинаковыми новыми дисководами. Вот пример командной строки для выполнения автоматической разметки:

```
Fdisk.exe 1 /PRI:2000 /EXT:8000 /LOG:8000 /Q
```

здесь:

1 – номер адресуемого физического дисковода на жестких магнитных дисках (1, 2, ...).

/PRI:2000 – разметить первичный раздел, в частности, объемом 2000 Мбайт. Если требуется маркировать этот раздел под файловую систему FAT-16, то вместо параметра /PRI: следует указывать /PRIO:.

/EXT:8000 – разметить расширенный раздел, в частности, объемом 8000 Мбайт.

/LOG:8000 – разметить логический диск объемом, в частности, 8000 Мбайт, внутри расширенного раздела. Когда размер логического диска не превышает 2000 Мбайт, и требуется маркировать этот логический диск под файловую систему FAT-16, то вместо параметра /LOG: следует указывать параметр /LOGO:.

/Q – "quiet" – необязательный параметр, исключающий выведение на экран сообщений при проведении разметки диска в автоматическом режиме.

В случае успешного завершении своей миссии программа FDISK.EXE, как обычно, выводит компьютер в перезагрузку.

Примечание 1: перезапись MBR устраняет возможные повреждения, в том числе вызываемые компьютерными вирусами, но вместе с тем следует иметь ввиду, что специфичные варианты MBR иногда используются намеренно. Так действуют, например, система DDO (Dynamic Disk Overlay) фирмы OnTrack, а также многие менеджеры загрузки операционных систем. Если такие средства установлены, то вместо программы FDISK.EXE надо использовать процедуры восстановления нестандартных записей MBR (пример показан в разделе 9.02-03).

Примечание 2: программе FDISK.EXE фирмы Microsoft присущ ряд существенных ограничений. Она не позволяет резервировать дисковое пространство посредством не-последовательного расположения

разделов. Она неправильно размечает разделы, переходящие через границу 8.4 Гбайт от начала дискового пространства. Кроме того, она не всегда правильно опознает разделы, созданные другими операционными системами: иногда не позволяет их удалить, а иногда располагает новые разделы "поверх" неверно опознанных. В таких ситуациях положения границ формируемых разделов следует внимательно проверять.

Примечание 3: наиболее серьезные недостатки оригинальной программы FDISK.EXE фирмы Microsoft исправлены в новой неофициальной версии (2006-го года), которую можно скачать с сайта http://radified.com/Files/FDISK.EXE . Кроме того, еще одна одноименная программа была заново написана Б.Е.Райфснайдером (Brian E. Reifsnyder). Версию 1.30 этой программы (2003-го года) можно скачать с сервера ftp://ftp.uni-koeln.de/pc/msdos/diskutils/ в составе архива FDISK130.ZIP.

Примечание 4: при любом изменении структуры разделов диска посредством программы FDISK.EXE или ее аналогов нельзя избежать полной потери данных в тех разделах, которые это изменение затрагивает. Переразметка разделов диска с сохранением данных возможна, но для этого требуются более мощные средства, например, программа Partition Magic фирмы PowerQuest.

Примечание 5: с помощью программы FDISK.EXE или ее аналогов нельзя выполнять разметку дисков, доступ к которым осуществляется по сети или посредством устанавливаемых драйверов. По этой причине нередко не поддаются разметке, в частности, сменные диски с интерфейсом SCSI и карты памяти с интерфейсом USB. В таких случаях могут оказаться полезными другие программы разметки дисков, например, BTFDISK.EXE фирмы Buslogic или TFDISK.EXE фирмы Tekram. Обе эти программы можно скачать с сайта http://www.neuron.alt.ru/drivers/Driver/Controllers/ из подкаталога BUSLOGIC в составе самораспаковывающегося файла архива DOSASPI.EXE и из подкаталога TEKRAM в составе файла архива DC390FBW.ZIP.

6.14 FIND.EXE – поиск слов в файлах

Программа FIND.EXE действует как фильтр для текстовых файлов: она получает поток строк из файла или с перенаправления, отбирает из этого потока строки по критерию наличия или отсутствия в них определенных слов, и направляет отобранные строки в стандартный канал вывода STDOUT, через который они по умолчанию поступают к драйверу консоли для воспроизведения на

экране. Вот типичный пример использования программы FIND.EXE по ее прямому назначению:

```
FIND /N /I " INT 13 " C:\DOS\SRV\Drives.txt
```

здесь:

/N – необязательный параметр, вызывающий выведение порядковых номеров отображаемых строк.

/I – необязательный параметр, заставляющий при сопоставлении игнорировать различие строчных и заглавных букв.

" INT 13 " – пример искомой группы слов, заключаемой с обеих сторон в двойные кавычки. Обратите внимание на пробелы между кавычками и словами искомой группы: благодаря этим пробелам гарантируется выделение только тех строк, в которых искомые слова употреблены отдельно, а не входят в состав других слов.

C:\DOS\SRV\Drives.txt – пример файла, строки которого надлежит анализировать. Имени файла предшествует полный путь. Если путь не указан, то поиск файла будет производиться только в текущем каталоге. Допускается указывать последовательно несколько имен анализируемых файлов или заменять имя маской файла с использованием знаков подстановки (2.01-03).

В результате исполнения показанной командной строки на экран будут выведены все строки анализируемого файла, которые содержат слова INT 13, вместе с кратким напоминанием о том, из какого файла эти строки выделены. Перед строками будет указан их номер, который поможет их находить в исходном файле. Если вы ожидаете, что число выводимых строк будет велико, то удобнее выводить строки в файл посредством перенаправления вывода (2.04-03) или воспользоваться утилитой просмотра MORE.COM (6.19).

Закончив поиск, программа FIND.EXE оставляет код ошибки (errorlevel) 0, если ей удалось найти хотя бы одну строку, содержащую искомую группу слов, или код ошибки 1, если ни одной такой строки найти не удалось. Определить результативность поиска по коду ошибки можно так же, как показано на примерах в разделах 3.15-03 и 6.03.

Другой менее типичный пример использования программы FIND.EXE – это подсчет полного числа строк в тексте:

```
FIND.EXE  /V /C "" < Draft.txt
```

здесь:

/V – необязательный параметр, вызывающий выведение тех строк, которые не содержат указанную в кавычках группу слов.

/C – необязательный параметр, вызывающий выведение результата подсчета числа найденных строк, но не самих строк.

"" – пустая спецификация группы слов, которые надлежит искать.

< Draft.txt – пример посылки анализируемого текстового файла программе FIND.EXE через перенаправление ввода (2.04-02). Так как имени файла не предшествует путь, предполагается наличие этого файла в текущем каталоге. При получении потока считываемых текстовых строк через перенаправление ввода программа FIND.EXE не добавляет упоминание об анализируемом файле к выводимому на экран результату.

Пустая спецификация искомой группы слов в показанном выше втором примере рассматривается программой FIND.EXE как специальный несуществующий объект. Поэтому программа FIND.EXE будет просто считать все строки, включая пустые строки. После такого подсчета числа строк программа FIND.EXE всегда возвращает код ошибки 0.

Третий пример применения программы FIND.EXE представляет собой часть batch-файла. Предположим, что в ходе исполнения получен от пользователя и записан в переменную окружения %P% путь к каталогу назначения, но неизвестно, заканчивается ли этот путь знаком обратной косой черты или нет. Если его нет, то его необходимо добавить, но если он уже имеется, то добавлять нельзя. Данную задачку можно решить следующим образом:

```
echo %P%\\ | Find.exe "\\\" > nul
if errorlevel 1 set P=%P%\
```

В первой строке программа FIND.EXE получает через промежуточное перенаправление значение переменной %P%, дополненное еще двумя знаками обратной косой черты. Искомый объект, представленный в кавычках тремя знаками обратной косой черты подряд, будет найден только в том случае, если один знак обратной косой черты в конце значения переменной уже был указан. Сообщение, выводимое программой FIND.EXE, не представляет интереса и потому перенаправляется в NUL (т.е. "в никуда"). Результат становится известен через оставляемый программой FIND.EXE код ошибки (errorlevel), который анализируется во второй строке показанного выше примера. Код ошибки 1 означает, что программа FIND.EXE не нашла в анализируемом потоке данных искомой последовательности знаков, и только тогда команда SET добавит к концу спецификации пути недостающий знак обратной косой черты.

Последний (четвертый) пример пользования программой FIND.EXE также представляет собой часть batch-файла. Предположим, что активизирована процедура подготовки списка файлов, которые надлежит сохранить в упакованном архиве, и при этом требуется избежать повторной упаковки, когда в подготовленном списке нет ничего, кроме упакованных архивных файлов того же вида (например, с суффиксом RAR). Проверить подготовленный список файлов %Temp%\Files.lst можно следующим образом:

```
FIND /C /I /V ".rar" < %Temp%\Files.lst | FIND ": 0" > nul
if not errorlevel 1 echo Chosen file(s) - already RAR-archive(s)
if not errorlevel 1 goto NO_PACK
```

В первой строке программа FIND.EXE вызывается дважды. В первый раз она получает через перенаправление ввода строки подготовленного списка и считает только те строки, в которых нет суффикса ".RAR". Результат подсчета строк посредством промежуточного перенаправления записывается во временный файл. Затем программа FIND.EXE вызывается второй раз и запрашивает данные для анализа из перенаправления, то есть из уже подготовленного временного файла, куда записан результат подсчета числа строк. Если результат подсчета нулевой, то обнаруживается цифра 0, и возвращаемый код ошибки устанавливается в 0. Последние две строки регистрируют нулевое значение кода ошибки, выводят на экран соответствующее сообщение и выполняют переход, исключающий повторное исполнение упаковки.

6.15 FORMAT.COM – форматирование дисков

Программа FORMAT.COM формирует логические диски на носителях записи, в том числе на дискетах и в разделах жестких магнитных дисков. Форматирование включает проверку пригодности секторов диска для записи, формирование служебных заголовков секторов и загрузочного сектора (boot-сектора), создание таблицы размещения файлов (FAT) и корневого каталога. Кластеры, в которых обнаруживаются непригодные для записи секторы, маркируются в таблице FAT как плохие (BAD), и благодаря тому потом не используются. На дисках объемом от 16 Мбайт и ниже формируется файловая система FAT-12. Для разделов жестких магнитных дисков программа FORMAT.COM выбирает файловую систему (FAT-16 или FAT-32) смотря по тому, какой идентификатор файловой системы (A.13-6) приписан данному разделу в MBR программой FDISK.EXE (6.13). Размер кластеров по умолчанию устанавливается минимально допустимым для выбранного типа файловой системы при имеющемся фактическом размере раздела.

Форматирование гибких магнитных дисков (дискет) включает низкоуровневую рекалибровку дорожек, так что фактическая емкость дискеты в результате форматирования может быть изменена. Вот пример пользования программой FORMAT.COM для форматирования дискеты:

```
FORMAT A: /V:Archives /Q /F:1.44 /B
```

здесь:

A: – пример обязательной спецификации буквенного обозначения дисковода или логического диска, подлежащего форматированию. Это обозначение должно быть назначено системой BIOS при включении компьютера. Буквенные

обозначения, которые назначены или переопределены позже без участия системы BIOS, программа FORMAT.COM считает недействительными.

/V:Archives – необязательный параметр, представляющий образец метки диска. Метка должна быть словом или группой слов суммарной длиной не более 11 знаков. Если метку диска не объявлять в командной строке, то программа FORMAT.COM попросит ввести ее по окончании процедуры форматирования. Пользователь вправе отказаться, просто нажав в ответ клавишу ENTER, и тогда диск по умолчанию получит метку NO_NAME (безымянный).

/Q – "quick format" – необязательный параметр, задающий режим быстрого форматирования, при котором не производятся тестирование секторов и запись секторных заголовков. Быстрому форматированию можно подвергать дискеты, которые уже были форматированы прежде, чтобы полностью стереть их прежнее содержание. Если нет уверенности в хорошем состоянии рабочей поверхности дискеты, то использовать режим быстрого форматирования не рекомендуется.

/F:1.44 – необязательная спецификация типа формата дискеты, допустимо указывать 160, 180, 320, 360, 720 (килобайт), а также 1.2, 1.44, 2.88 (Мегабайт). Вместо параметра /F можно указывать другие, реже используемые спецификации формата, показанные в примечании. Но если ни одна из альтернативных спецификаций не указана, FORMAT.COM установит тип формата по данным из CMOS-памяти системы BIOS с учетом сигналов сенсоров дисковода.

/B – необязательный параметр, вызывающий резервирование места на форматируемой дискете для последующего размещения системных файлов, чтобы сделать дискету загрузочной. Если вместо параметра /B указан параметр /S, то программа FORMAT.COM не только выделит место, но и выполнит копирование системных файлов (IO.SYS и COMMAND.COM) на форматируемый диск из корневого каталога того диска, с которого компьютер был загружен. При этом файл MSDOS.SYS не копируется, а создается заново пустым, чтобы параметры загрузки устанавливались по умолчанию.

В современных жестких магнитных дисках структура дорожек однократно формируется в процессе изготовления, и при форматировании изменять ее нельзя. Поэтому для разделов на жестких магнитных дисках спецификации формата не

задаются. Вот пример вызова программы FORMAT.COM для форматирования раздела жесткого магнитного диска:

```
FORMAT D: /s /c /z:64
```

здесь:

/c – необязательный параметр, вызывающий повторное тестирование тех кластеров, которые при предыдущем форматировании того же диска были помечены как негодные (BAD). Иногда это помогает восстановить диски и дискеты, у которых неуверенно читается какой-либо сектор на первой дорожке. Но повторное тестирование большого количества негодных кластеров может существенно увеличить затраты времени на форматирование.

/z:64 – необязательный параметр, задающий формирование кластеров размером 32 килобайта, содержащих по 64 сектора каждый. Это бывает нужно, если Вы в дальнейшем намерены увеличить размер данного раздела. Параметр /z:1 позволяет формировать файловую систему FAT-16 на малых дисках объемом от 4 до 16 Мбайт. Но в большинстве случаев принимаемый по умолчанию размер кластера изменять не приходится, и потому параметр /Z: обычно не указывают.

Примечание 1: вместо формата дискеты допускается указывать число дорожек на одной стороне дискеты и число секторов на каждой дорожке, например, так:

/T:80 /N:18.

Еще одной альтернативной спецификацией формата является указание числовых параметров из следующей группы:

/1 – форматируемая дискета – односторонняя;

/4 – дискета на 360 кбайт в дисководе на 1.2 Мбайт;

/8 – размечать дорожки на 8 секторов.

Примечание 2: оригинальная программа FORMAT.COM фирмы Microsoft неспособна правильно форматировать разделы, расположенные далее 64 Гб от начала диска. Исправленную новую неофициальную версию программы FORMAT.COM (2006-го года) можно скачать с сайта http://www.mdgx.com/files/FDSKFRMT.EXE в составе SFX архива FDSKFRMT.EXE.

6.16 LABEL.EXE – запись метки диска

Метка представляет собой слово или группу слов суммарной длиной не более 11 знаков, используемую для идентификации диска. Метка вписывается в загрузочный сектор диска (A.03-4) и, кроме того, в одну из скрытых записей

корневого каталога. Нередко метку назначают в ходе форматирования, но ее можно назначить или изменить позже с помощью программы LABEL.EXE, например:

```
LABEL.EXE  R:RAMDRIVE
```

здесь:

R: – пример спецификации буквенного обозначения для диска, которому надлежит назначить метку. Если букву диска не указывать, то по умолчанию будет принят текущий диск.

RAMDRIVE – пример метки, которая должна быть назначена.

Когда в командной строке метка не указана, программа LABEL.EXE показывает имеющуюся метку адресуемого диска и предлагает пользователю ввести новую метку, причем метка может быть введена не только напрямую с клавиатуры, но также посредством перенаправления ввода (2.04-02, 2.04-05). Если пользователь отклонит предложение ввести новую метку, то ему будет предоставлена возможность удалить или сохранить имеющуюся метку. По завершении своей миссии программа LABEL.EXE оставляет следующие значения кода ошибки (errorlevel):

082 – попытка записи метки диска не удалась.

015 – неверно задана буква диска, такого диска нет.

002 – сменный диск в указанный дисковод не вставлен.

000 – диск либо доступен, либо не форматирован, но попытка записи метки не предпринималась.

Выяснить значение оставленного кода ошибки можно так же, как показано в разделе 6.03. Благодаря информативности оставляемого кода ошибки программа LABEL.EXE иногда используется для тестирования доступности дисков (пример – в разделе 9.03-02).

6.17 MEM.EXE – сводка распределения памяти

Программа MEM.EXE выводит на экран сведения о том, сколько памяти имеется в компьютере, как она используется, какие модули загружены и т.п. Вот пример вызова программы MEM.EXE:

```
MEM.EXE /A /C /P
```

здесь:

/A – дополнить сводку сведениями о наличии свободного места в области верхней памяти (HMA).

/C – показать таблицу, левый столбец которой содержит перечень названий загруженных резидентных модулей, а в остальных столбцах – количество выделенной этим модулям памяти. Далее

выводятся сводные данные об объеме использованной и свободной памяти. Вместо параметра /C можно указать:

/D – показать статус резидентных модулей и драйверов.

/F – показать только количество свободной памяти.

/M:HIMEM – показать данные о месте размещения резидентного модуля, название которого (HIMEM) приведено после двоеточия. Подобным образом могут быть запрошены сведения о любых других резидентных модулях, названия которых выводятся в левом столбце таблицы сводки.

/P – приостанавливать дальнейший вывод данных каждый раз после заполнения всех строк экрана.

6.18 MODE.COM – настройка оборудования

Программа MODE.COM служит для настройки режимов работы оборудования компьютера, в частности, портов, видеокарты и знакогенератора. Сводку данных об установленных режимах работы "подведомственных" устройств программа MODE.COM дает, когда ее вызывают с параметром /STATUS:

```
MODE.COM  /STATUS
```

6.18-01 MODE.COM: операции настройки портов

Вот пример вызова программы MODE.COM для настройки режима работы последовательного порта COM1 (вместо COM1 можно указать порт COM2, COM3 или COM4):

```
MODE.COM  COM1:11,E,8,2,B
```

здесь:

11 – установить скорость 110 бод; допускаются значения 11, 15, 30, 60, 12, 24, 48, 96, 19, соответствующие скоростям передачи 110, 150, 300, 600, 1200, 2400, 4800, 9600 и 19200 бод.

E – выполнять проверку по четности; вместо E можно указать:

O – выполнять проверку по нечетности;

N – не выполнять проверку данных.

8 – передавать 8 бит в кодовом слове; допускается 7 или 8.

2 – передавать 2 стоповых бита; допускается 1 или 2.

B – означает нормальную реакцию на ошибки устройства; вместо "B" допускается указывать:

E – сообщать об ошибке (ERROR), когда устройство занято;

N – не обращаться к устройству повторно после ошибки;

P – продолжать вызовы устройства до нажатия Ctrl-Break;
R – посылать устройству на исполнение новое задание, даже если выполнение предыдущего не завершено.

При использовании программы MODE.COM для настройки режимов работы параллельных портов предполагается, что подключенным устройством является принтер. После имени параллельного порта (LPT1, LPT2 или LPT3) программа MODE.COM принимает другую группу параметров:

```
MODE.COM  LPT1:80,6,B
```

здесь:

80 – число печатаемых знаков в строке; допустимо 80 или 132.
6 – число строк на дюйм; допускается 6 или 8.
B – означает нормальную реакцию на ошибки принтера; вместо "B" допускается указывать те же альтернативные параметры, что и для последовательных портов.

Программа MODE.COM способна осуществить переадресацию сообщений с параллельного порта (LPT1 – LPT3) на последовательный (COM1 – COM4), позволяя тем самым работать с принтером, подключаемым к какому-либо последовательному порту:

```
MODE LPT1:=COM2
```

Для возврата из состояния переадресации к нормальной адресации необходимо вызвать программу MODE.COM повторно, на сей раз без указания каких-либо параметров после имени порта:

```
MODE.COM LPT1:
```

6.18-02 MODE.COM: переключение текстовых видеорежимов

Программа MODE.COM допускает две формы обращения для переключения текстовых режимов видеокарты (A.10-1). Согласно первой, устаревшей форме параметры можно указывать так:

```
MODE.COM  co80
```

здесь:

co80 – означает цветной видеорежим с 80 знаками в строке; вместо параметра "co80" допускается указывать:
bw40 – монохромный видеорежим, 40 знаков в строке
bw80 – монохромный видеорежим, 80 знаков в строке
co40 – цветной видеорежим, 40 знаков в строке.

Другая форма обращения позволяет устанавливать большее число текстовых строк:

```
MODE.COM  80,25
```

здесь:

80 – число знаков в строке; допускается 40 или 80.
25 – число строк текста; допускается 25, 43 или 50.

Примечание 1: при изменении видеорежимов программой MODE.COM происходит смена шрифтов. Если до изменения видеорежима шрифты были установлены не программой MODE.COM, то возможна смена кодовой страницы на 437-ю, принимаемую по умолчанию.

6.18-03 MODE.COM: установка кодовых страниц

Команды на выполнение программой MODE.COM операций по подготовке и выбору кодовых страниц знакогенератора обычно записываются в файл AUTOEXEC.BAT (9.01-02). Для выполнения этих операций требуется, чтобы драйвер DISPLAY.SYS (5.02-02) уже был загружен. Первой операцией необходимо подготовить подлежащие установке кодовые страницы:

```
MODE.COM CON CP PREP=((437,850,866) EGA3.CPI)
```

здесь:

CON – спецификация устройства, к которому обращена данная операция. Вместо CON (т.е. клавиатуры и дисплея) допускается указывать LPT1, LPT2 или PRN (т.е. принтер).
CP PREP – краткое наименование операции (CodePage PREPare = подготовить кодовую страницу).
(437,850,866) – номера подготавливаемых кодовых страниц (A.02-2). Только эти кодовые страницы будут доступны для переключения командой CHCP (3.04).
EGA3.CPI – пример имени файла, содержащего все те кодовые страницы, которые должны быть подготовлены.

Затем одна из подготовленных кодовых страниц должна быть активизирована следующей операцией выбора кодовой страницы:

```
MODE.COM CON CP SEL=866
```

здесь:

CP SEL – краткое наименование операции (CodePage SELect = выбрать кодовую страницу).
866 – пример номера выбираемой кодовой страницы.

Кодовая страница, активизированная операцией выбора для устройства CON (консоли), определяет вид букв и других знаков, воспроизводимых на экране дисплея. Чтобы выяснить, какая именно кодовая страница действует сейчас на данном устройстве (CON, PRN, LPT1 или LPT2), следует вызвать программу MODE.COM с именем операции CP (CodePage) и с параметром /STATUS, например, так:

```
MODE.COM CON CP /STATUS
```

Бывает, что уже загруженная кодовая страница оказывается повреждена в результате сбоя или неадекватных действий других программ. В таких случаях программа MODE.COM тоже может помочь, повторно загрузив ту же самую кодовую страницу:

```
MODE.COM CON CP REF
```

здесь:

CP REF – краткое наименование операции (CodePage REFresh = обновить кодовую страницу).

6.19 MORE.COM – постраничный просмотрщик

Выводимые на экран многостраничные сообщения иногда "проскакивают" так быстро, что ничего не удается прочесть. Для обеспечения последовательного постраничного вывода сообщений на экран в составе MS-DOS7 имеется программа MORE.COM. Каждый раз, когда экран заполняется строками сообщения, программа MORE.COM останавливает вывод строк на экран, пока пользователь не разрешит продолжить вывод строк нажатием любой клавиши. Программа MORE.COM получает выводимые на экран сообщения из канала STDOUT либо посредством перенаправления ввода (2.04-02):

```
MORE.COM < D:\MyDocs\Part2.txt
```

либо посредством промежуточного перенаправления (2.04-05):

```
Type D:\MyDocs\Part2.txt | MORE.COM
```

здесь:

D:\MyDocs\Part2.txt – пример спецификации файла, который надо прочитать. Перед именем файла указан полный путь. Если путь опустить, то поиск файла будет производиться только в текущем каталоге.

Type D:\MyDocs\Part2.txt – пример команды (3.30), которая считывает указанный файл в стандартный канал вывода STDOUT. С таким же успехом здесь могла бы быть указана любая другая команда,

которая пользуется каналом STDOUT для вывода своих сообщений на экран.

Примечание 1: сообщения могут быть выведены на экран также через прерывания системы BIOS и через канал вывода сообщений об ошибках STDERR. Такие сообщения не перехватываются программой MORE.COM.

Примечание 2: при пользовании промежуточным перенаправлением (2.04-05) DOS записывает сообщение во временный файл, для чего требуется доступ к записываемому носителю. Если текущий диск защищен от записи, и путь к записываемому носителю не указан в переменной окружения %TEMP%, то программа MORE.COM не сможет выполнить свою миссию. Поэтому всегда, когда это возможно, лучше использовать перенаправление ввода.

6.20 MOVE.EXE – перевод файлов в другой каталог

Программа MOVE.EXE переименовывает каталоги и переписывает файлы из одного каталога в другой.

Когда каталог-источник и каталог назначения находятся на разных дисках, то при переводе файла фактически выполняется копирование, после чего исходный файл удаляется. Но когда оба каталога находятся на одном диске, то в копировании нет необходимости: гораздо проще и быстрее перевести из одного каталога в другой только запись, которая регистрирует наличие файла в каталоге и указывает на первый кластер файла. Сам файл при этом никуда не считывается и не записывается. Программа MOVE.EXE сама определяет, как действовать в каждом конкретном случае.

Вместе с переводом файла программа MOVE.EXE позволяет выполнить его переименование. Помимо прочего, программа MOVE.EXE позволяет переименовывать каталоги: эта операция тоже выполняется путем исправления записи в каталоге, который является родительским по отношению к переименовываемому.

Вот пример пользования программой MOVE.EXE для перевода файлов из одного каталога в другой:

```
MOVE.EXE /Y  D:\MyDocs\Part*.txt C:\Dos\Chap*.txt
```

здесь:

/Y – необязательный параметр, разрешающий перезаписывать одноименные файлы в каталоге назначения без предупреждения. Если этот параметр отсутствует в командной строке, то программа ищет его в переменной окружения COPYCMD.

Напротив, если перезапись без предупреждения нежелательна, а в переменной COPYCMD параметр /Y имеется, то его действие можно отменить указанием параметра /-Y в командной строке.

D:\MyDocs\Part*.txt – пример спецификации подлежащих перемещению файлов с предшествующим путем к каталогу-источнику. Если путь не указан, то каталогом-источником будет служить текущий каталог. В одной строке можно указывать несколько спецификаций подлежащих перемещению файлов.

C:\Dos\Chap*.txt – пример спецификации каталога назначения с маской новых имен для файлов. Из всех подобных спецификаций в командной строке самая последняя интерпретируется как спецификация каталога назначения. Если последнее имя в этой спецификации не является именем существующего объекта, то оно используется для переименования перемещаемого файла. Но если последнее имя в спецификации каталога назначения – это имя существующего каталога, то файлы переводятся в этот каталог без переименования.

Чтобы с помощью программы MOVE.EXE переименовать каталог, первая из указанных в командной строке спецификаций должна кончаться не именем файла и не маской, а именем существующего каталога. После нее отдельно должно быть указано новое имя, назначаемое переименовываемому каталогу. Новое имя следует указывать без предшествующего пути, даже если переименовываемый каталог не находится в текущем каталоге.

6.21 SCANDISK.EXE – программа проверки дисков

Программа SCANDISK.EXE служит для проверки и исправления дискет с файловой системой FAT-12, а также разделов жестких магнитных дисков с файловыми системами FAT-16 или FAT-32. Программа SCANDISK.EXE проверяет загрузочный сектор (boot-сектор), выявляет наличие потерянных кластеров, исправляет перекрестные ссылки в таблице FAT, тестирует кластеры диска на пригодность для записи. Если обнаруживаются непригодные кластеры, то они помечаются в таблице FAT так, что их дальнейшее использование исключается. Когда плохой кластер уже использован для записи файла, программа SCANDISK.EXE старается по мере возможности считать данные из него и перезаписать их в свободный хороший кластер, сохранив целостность файла. Ход процесса тестирования и расположение непригодных кластеров наглядно отображаются на экране, как показано ниже на рис. 4.

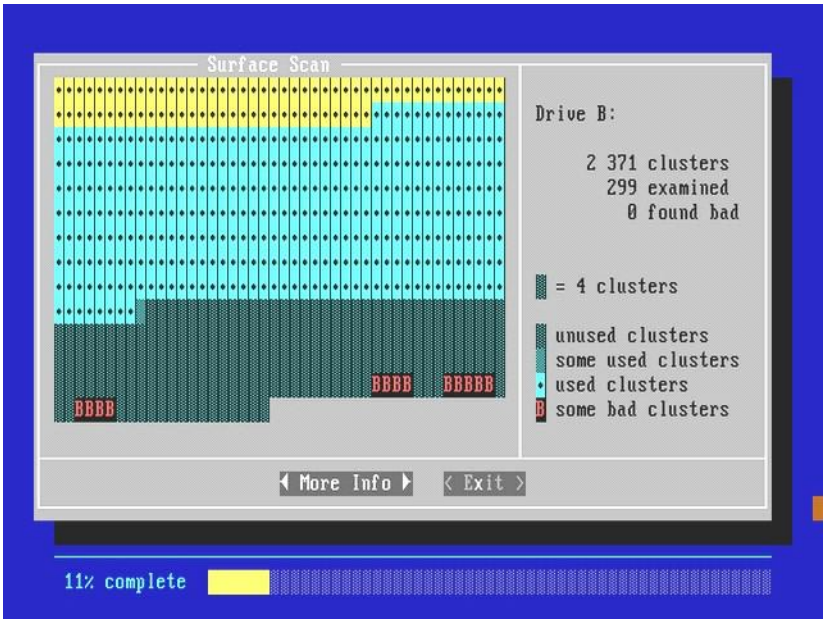


Рис. 4

Установки режимов проверок считываются программой SCANDISK.EXE из отдельного файла SCANDISK.INI, который должен находиться в одном каталоге с основным исполняемым файлом программы. Файл SCANDISK.INI – обычный текстовый файл, он содержит исчерпывающие комментарии, и установки режимов в нем пользователь может исправлять сам. Если нужно проверить диск с установками режимов из файла SCANDISK.INI, то программу SCANDISK.EXE следует запускать на исполнение с параметром /CUSTOM в командной строке. Когда параметр /CUSTOM в командной строке отсутствует, тогда учитывается только содержание секции [ENVIRONMENT] файла SCANDISK.INI.

Другие параметры в командной строке запуска программы SCANDISK.EXE имеют приоритет над теми, которые записаны в файл SCANDISK.INI. Чтобы в ходе проведения проверок программа SCANDISK.EXE автоматически исправляла найденные дефекты, ее нужно запустить с параметрами командной строки, например, так:

```
SCANDISK.EXE C: /AUTOFIX /NOSAVE /NOSUMMARY /SURFACE
```

здесь:

C: – пример спецификации диска, который нужно проверить. Если нужно проверить все доступные диски, то вместо буквы диска следует указать параметр /ALL. При проверке сжатых логических дисков вместе с буквой диска требуется указать имя тома, например, так: C:\DRVSPACE.000

/AUTOFIX – необязательный параметр, разрешающий исправление дефектов без предупреждения. Напротив, если исправление без

предупреждения нежелательно, но уже записано в файл SCANDISK.INI, то оно может быть отменено указанием в командной строке параметра /CHECKONLY.

/NOSAVE – необязательный параметр, разрешающий перемечать потерянные кластеры в свободные без сохранения данных. Этот параметр можно указывать только совместно с параметром /AUTOFIX. По умолчанию потерянные кластеры преобразуются в файлы с суффиксом *.CHK и помещаются в корневой каталог того же диска.

/NOSUMMARY – необязательный параметр, разрешающий проведение процедур без остановок на показ сводных данных. Этот параметр может быть указан только совместно с параметром /CHECKONLY или с параметром /AUTOFIX.

/SURFACE– необязательный параметр, разрешающий проведение тестирования поверхности диска. У сжатых логических дисков тестирование поверхности проводить нельзя.

Когда параметр /AUTOFIX в командной строке не указан, программа SCANDISK.EXE перед внесением каких-либо исправлений запрашивает разрешение на запись исходного состояния на восстановительные дискеты (UNDO-дискеты). Наличие UNDO-дискет позволит восстановить исходное состояние проверяемого диска, если потом выяснится, что внесенные исправления чему-нибудь повредили. Принимая решение о записи исходного состояния на UNDO-дискеты, нужно иметь ввиду, что объем записываемых данных зависит от количества вносимых исправлений. При исправлении ошибок на современных больших жестких магнитных дисках могут потребоваться сотни UNDO-дискет, и их запись займет очень много времени. По этой причине от записи исходного состояния на UNDO-дискеты обычно приходится отказываться.

Если все же копирование исходного состояния диска на UNDO-дискеты выполнено, и потом установлена необходимость восстановления исходного состояния, то процедура восстановления должна быть проведена сразу, до записи на подлежащий восстановлению диск каких-либо файлов:

```
SCANDISK.EXE /UNDO B:
```

здесь:

/UNDO – параметр, запускающий процедуру восстановления.

B: – пример спецификации дисковода, используемого для считывания данных с UNDO-дискет.

Программа SCANDISK.EXE также может быть использована для выяснения того, фрагментирован ли какой-либо конкретный файл или нет:

```
SCANDISK.EXE /FRAGMENT C:\WINDOWS\SYSTEM\KERNEL32.DLL
```

здесь:

/FRAGMENT – параметр, вызывающий процедуру выяснения фрагментированности файла.

C:\WINDOWS\SYSTEM\KERNEL32.DLL – пример спецификации файла, степень фрагментированности которого надлежит выяснить. Если путь к файлу не указан, то его поиск будет производиться только в текущем каталоге.

Помимо прочего, программа SCANDISK.EXE принимает в командной строке еще параметр /MONO, вызывающий монохромное выведение сообщений на экран (по умолчанию сообщения выводятся в цвете).

Примечание 1: с помощью программы SCANDISK.EXE нельзя тестировать сетевые диски, оптические диски CD-ROM, виртуальные диски, создаваемые программами ASSIGN.COM, SUBST.EXE и JOIN.EXE, а также диски с такими повреждениями служебных областей, которые делают невозможным доступ к диску. Некоторые нечитаемые диски с повреждениями boot-сектора тем не менее можно восстановить с помощью программ NDD.EXE и DISKEDIT.EXE фирмы SYMANTEC.

Примечание 2: программа SCANDISK.EXE выполняет тестирование в режиме прямого доступа к диску и рассчитана на использование только в однозадачной операционной среде. В многозадачной операционной среде, в частности, в "окне DOS" операционной системы WINDOWS, вместо SCANDISK.EXE автоматически вызывается другая программа – SCANDSKW.EXE из каталога C:\WINDOWS.

Примечание 3: не следует применять программу SCANDISK.EXE с параметром /AUTOFIX по отношению к дискам, которые могут быть заражены компьютерным вирусом. Это может привести к необратимой потере данных. Такие диски сначала следует проверить с помощью антивирусной сканирующей программы (например, DRWEB.EXE).

Примечание 4: каждое длинное имя, созданное операционной системой Windows, программа SCANDISK.EXE считает ошибочным. Попытки исправить все такие ошибки приводят к катастрофическим последствиям. Чтобы этого избежать, секция [ENVIRONMENT] файла SCANDISK.INI должна содержать строки:

```
LfnCheck      = Off
SpaceCheck    = Off
```

В поставке операционной системы Windows-ME имеется частично исправленная версия программы SCANDISK.EXE, которая менее радикально относится к длинным именам и не проверяет версию DOS, так что ее можно использовать в MS-DOS7.

Примечание 5: для проведения операции перезаписи данных из плохих кластеров в хорошие необходимо, чтобы на диске имелись свободные хорошие кластеры. Несложно высвободить место, переписав любой файл на другой носитель, однако имеются программы (например, версия 2.50 архиватора PKZIP.EXE), которые способны заполнить всю дискету одним файлом архива, не оставляя ни единого свободного сектора. Такой файл нельзя прочесть, пока он не исправлен, а исправить нельзя, потому что свободного места нет. Заполнения всего носителя одним файлом желательно заранее избегать.

6.22 SORT.EXE – сортировка строк

Программа SORT.EXE получает строки с перенаправления или из файла и посылает их в отсортированном виде в стандартный канал вывода STDOUT, обычно для воспроизведения на экране или для записи в файл. Сортировка строк происходит в порядке, задаваемом расположением знаков в таблицах кода ASCII.

Вот пример использования программы SORT.EXE для вывода строк файла на экран в измененном порядке:

```
SORT.EXE  /R /+12 D:\MyDocs\Unsort.txt
```

здесь:

/R – необязательный параметр, вызывающий изменение порядка сортировки на обратный, то есть от Z до A и затем от 9 до 0.

/+12 – пример спецификации номера знака в строке, по которому выполняется сортировка: в данном случае по знаку в колонке номер 12. Если этот параметр не указывать, то сортировка будет производиться по знаку в первой колонке.

D:\MyDocs\Unsort.txt – пример спецификации исходного текстового файла, строки которого надлежит сортировать. Если имени файла не предшествует путь, то поиск файла будет производиться только в текущем каталоге.

Вот еще один пример пользования программой SORT.EXE с получением строк посредством перенаправления ввода и записью отсортированных строк в файл:

```
SORT.EXE /+9 < D:\MyDocs\Unsort.txt > D:\MyDocs\Sort.txt
```

здесь:

D:\MyDocs\Sort.txt – пример спецификации файла, в который будут записаны строки после сортировки. Если такой файл уже существует, то он будет перезаписан без предупреждения. Указывать один и тот же файл в качестве источника и в качестве файла назначения нельзя: данные будут потеряны.

Последний пример показывает получение исходных строк через промежуточное перенаправление от другой команды и посылку отсортированных строк через перенаправление вывода на принтер, подключенный к порту LPT1:

```
Type D:\MyDocs\Unsort.txt | SORT /+3 > PRN
```

Примечание 1: перенаправлением вывода на принтер не следует пользоваться, если Вы не уверены, что он подключен именно к порту LPT1, готов к работе и способен работать в операционной среде MS-DOS.

Примечание 2: перенаправление вывода в файл и промежуточное перенаправление требуют доступа к записываемому носителю и не будут выполнены, если такой носитель недоступен (2.04-03 – 2.04-05).

6.23 SUBST.EXE – создание виртуальных дисков

Программа SUBST.EXE позволяет создавать и удалять виртуальные логические диски, которые подменяют собой какой-либо конкретный путь в структуре каталогов. Первоначально программа SUBST.EXE была создана для того, чтобы обеспечить возможность применения старых программ, разработанных для DOS первой и второй версии. В ранних версиях DOS не было иерархической структуры каталогов, и все файлы на диске помещались в корневой каталог. Программа SUBST.EXE позволяет обращаться к файлам так, как будто они находятся в корневом каталоге виртуального диска. В настоящее время программой SUBST.EXE пользуются редко, в основном для упрощения написания длинных спецификаций путей, сокращения длины командных строк и значения переменной %PATH%.

Вот пример использования программы SUBST.EXE для создания виртуального диска:

```
SUBST.EXE  V: D:\DATA386\For_K\MyDocs
```

здесь:

V: – пример буквенного обозначения, присваиваемого создаваемому виртуальному диску. Назначаемая буква должна быть свободна, то есть не должна принадлежать уже какому-либо диску. Кроме того, назначаемая буква не должна выходить за предел, устанавливаемый командой LASTDRIVE (4.17, 4.18) в файле CONFIG.SYS.

D:\DATA386\For_K\MyDocs – пример пути в структуре каталогов реального диска, подменяемого обозначением виртуального диска. Если путь не указан, то по умолчанию будет принят путь к текущему каталогу на текущем диске.

Чтобы потом удалить виртуальный диск, командная строка вызова программы SUBST.EXE должна быть составлена так:

```
SUBST V: /D
```

здесь:

V: – буквенное обозначение удаляемого виртуального диска.
/D – параметр, вызывающий операцию удаления виртуального диска.

Когда программа SUBST.EXE исполняется без параметров, она просто показывает перечень установленных виртуальных дисков.

Примечание 1: на виртуальные диски нельзя направлять действие следующих программ: Assign.com, Backup.exe, Chkdsk.exe, Defrag.exe, Diskcomp.com, Diskcopy.com, Fdisk.exe, Format.com, Label.exe, Mirror.exe, Recover.exe, Restore.exe, Scandisk.exe, Sys.com, Undelete.exe, Unformat.com.

Примечание 2: любой реальный путь в структуре каталогов останется доступным как прежде и после того, как ему будет поставлено в соответствие буквенное обозначение виртуального диска.

Примечание 3: виртуальные диски, создаваемые программой SUBST.EXE, наследуются операционной системой Windows-95, но не могут быть созданы после того, как операционная система Windows-95 уже загружена. Виртуальные диски следует создавать пока действует операционная среда MS-DOS7, предпочтительно посредством запуска программы SUBST.EXE из файла AUTOEXEC.BAT.

6.24 SYS.COM – подготовка загрузочных дисков

В процессе загрузки компьютера управление передается исполняемому коду, считанному из boot-сектора загрузочного носителя записи. Там должно быть указано имя файла-загрузчика, которому код boot-сектора, в свою очередь, передаст управление для обеспечения загрузки операционной системы. Чтобы успешно загрузить MS-DOS7 с дискеты или из раздела жесткого магнитного диска, их необходимо заранее подготовить: во-первых, вписать в boot-сектор соответствующий исполняемый код, во-вторых, указать имя файла IO.SYS, являющегося загрузчиком MS-DOS7, и, в-третьих, поместить файл IO.SYS вместе с комплектом основных системных файлов в корневой каталог. Для выполнения перечисленных подготовительных действий служит программа SYS.COM. Вот пример вызова программы SYS.COM:

```
SYS.COM  C:\  A:
```

здесь:

C:\ – пример пути к каталогу расположения системных файлов (IO.SYS и COMMAND.COM), которые надлежит копировать. Если этот путь не указан, то программа SYS.COM будет искать системные файлы в корневом каталоге того диска, с которого компьютер был загружен.

A: – пример буквенного обозначения того диска, который надлежит сделать загрузочным. Он должен быть опознан системой BIOS компьютера и должен быть заранее форматирован в операционной среде той же версии DOS.

Примечание 1: программу SYS.COM нельзя применять по отношению к сетевым дискам, к оптическим дискам CD/DVD-ROM и к виртуальным дискам, созданным как драйверами RAM-дисков, так и программами Assign.com, Subst.exe и Join.exe.

Примечание 2: чтобы сделать загрузочным жесткий магнитный диск, необходимо заранее присвоить статус загрузочного одному из его разделов, скорректировав соответствующим образом главную загрузочную запись (MBR) с помощью программы FDISK.EXE (6.13). Именно из этого раздела будет считан код boot-сектора при загрузке.

Примечание 3: в отличие от файлов IO.SYS и COMMAND.COM, системный файл MSDOS.SYS (5.01-01) не копируется, а создается заново пустым. Загрузка с нового диска с параметрами, принимаемыми по умолчанию, сочтена более безопасной, чем копирование параметров, которые могут оказаться неподходящими.

Примечание 4: в MS-DOS8 программа SYS.COM изменена: она не принимает путь к каталогу размещения системных файлов из командной строки, а ищет их только в корневом каталоге того диска, с которого компьютер был загружен.

6.25 VC.COM – файл-менеджер

6.25-01 Основные свойства файл-менеджера Volcov Commander.

Файл-менеджер Volcov Commander (VC.COM), написанный В.В.Волковым (г. Киев), напоминает известный файл-менеджер Norton Commander, но VC.COM компактнее и открывает больше возможностей подстройки к запросам пользователя. Volcov Commander представляет собой незавершенный проект, который не во всех отношениях удовлетворителен, но тем не менее многие признают его лучшим для обслуживания восстановительных и настроечных работ в операционной среде MS-DOS. Описываемую ниже версию VC.COM 4.99.07 от 1998 года можно скачать с сайта http://www.fdd5-25.net/shells.php в виде архива

vc499.zip. Самая последняя альфа-версия VC.COM 4.99.08 от 2000 года выложена на сайте http://vvv.kiev.ua/download/ в виде архива vc49908a.zip.

Собственно VC.COM – это стартовый файл. Поставка файл-менеджера также включает основной файл VC.OVL, ряд кодовых таблиц *.TBL, и еще несколько конфигурационных файлов:

VC.INI	– инициализационный файл с уставками параметров
VC.MNU	– файл меню, вызываемого нажатием клавиши F2
VC.EXT	– спецификации служб, вызываемых по клавише ENTER
VCARCH.EXT	– спецификации служб архивирования
VCEDIT.EXT	– спецификации служб, вызываемых по клавише F4 (EDIT)
VCVIEW.EXT	– спецификации служб, вызываемых по клавише F3 (VIEW)

Кроме VC.INI, все остальные конфигурационные файлы – текстовые, их можно редактировать с помощью программы EDIT.COM или любого другого редактора неформатированных текстовых файлов. Примеры составления некоторых конфигурационных файлов файл-менеджера VC.COM приведены в разделах 6.25-02 – 6.25-04.

Весь комплект файлов файл-менеджера VC.COM надо поместить в один каталог, и путь к этому каталогу записать в переменную окружения VC. С этой целью в файл AUTOEXEC.BAT следует ввести, например, такую строку:

```
set VC=C:\DOS\VC4
```

Запускать файл-менеджер можно из командной строки как обыкновенную программу, но чаще его запускают автоматически из последней строки файла AUTOEXEC.BAT, например, так:

```
C:\DOS\VC4\VC.COM /TSR /no2E /noswap
```

здесь:

/TSR – задействовать супервизор, который будет контролировать загрузку резидентных модулей других программ и обеспечит их автоматическое выгружение из памяти при завершении работы файл-менеджера VC.COM. Это помогает высвобождать память и иногда позволяет избежать перезагрузки компьютера. Напротив, если применение супервизора нежелательно, то в командной строке следует указать параметр /noTSR.

/no2E – не использовать прерывание INT 2E (8.02-89) для исполнения программ, а использовать основную функцию INT 21\AX=4B00h (8.02-53). Прерывание INT 2E действует быстрее, но оно не реентерабельно и не дает доступа к локальным переменным вызывающей программы. Напротив, если предпочтение отдано

прерыванию INT 2E, то в командной строке надо указать параметр /2E.

/noswap – не осуществлять выгрузку данных (swapping) из оперативной памяти во временный файл подкачки на диске (т.к. используемая адресация CHS для современных накопителей непригодна).

Помимо перечисленных, в командной строке запуска файл-менеджера VC.COM можно указывать следующие необязательные параметры:

/BW – выводить на экран монохромное изображение панелей файл-менеджера. Вместо /BW можно указать /LCD - использовать специальную палитру для жидкокристаллических индикаторных панелей. По умолчанию изображение панелей файл-менеджера выводится на экран в 16-цветном текстовом видеорежиме 03h.

/std – загрузить файл-менеджер VC.COM в обыкновенную память. По умолчанию предпочтительны иные варианты, если они доступны. Но если VC.COM все же загружен в обыкновенную память, то в командной строке можно дополнительно указать параметры:

- /big – загрузить в память весь резидентный модуль файл-менеджера VC.COM;
- /small – загрузить только часть резидентного модуля, требующую периодического пополнения с диска.

/XMS – загрузить файл-менеджер VC.COM в XMS-память, если драйвер XMS-памяти (HIMEM.SYS, 5.04-01) уже действует. Другие допустимые альтернативы по использованию памяти таковы:

- /noXMS – не использовать XMS-память.
- /EMS – загружаться в EMS-память, если она подготовлена уже драйвером EMM386.EXE (5.04-02).
- /noEMS – не использовать EMS-память.

/ini:Alter.ini– использовать другой инициализационный файл (в данном примере – Alter.ini) вместо принимаемого по умолчанию файла VC.INI. Если файл VC.INI переименовать, а потом нажатием клавиш Shift-F9 создать новый файл VC.INI с другими установками, то в одном каталоге окажутся два инициализационных файла. Аналогичным образом можно подготовить несколько инициализационных файлов и с помощью параметра /ini: выбирать желаемую конфигурацию.

/nozoom – не масштабировать "окна" вывода сообщений.

/? – показать краткую справку.

После всех параметров в командной строке запуска VC.COM допускается указывать имя программы, которая должна быть запущена на исполнение сразу после загрузки файл-менеджера. Эту возможность иногда целесообразно

использовать, например, для сохранения таблицы прерываний с помощью программы ESCAPE.COM.

Когда файл-менеджер VC.COM уже запущен, экран дисплея может выглядеть по-разному в зависимости от начальных установок в инициализационном файле (VC.INI). Но обычно на экран выводятся одна или две панели с перечислением файлов из соответствующих каталогов, как показано на рис. 5.

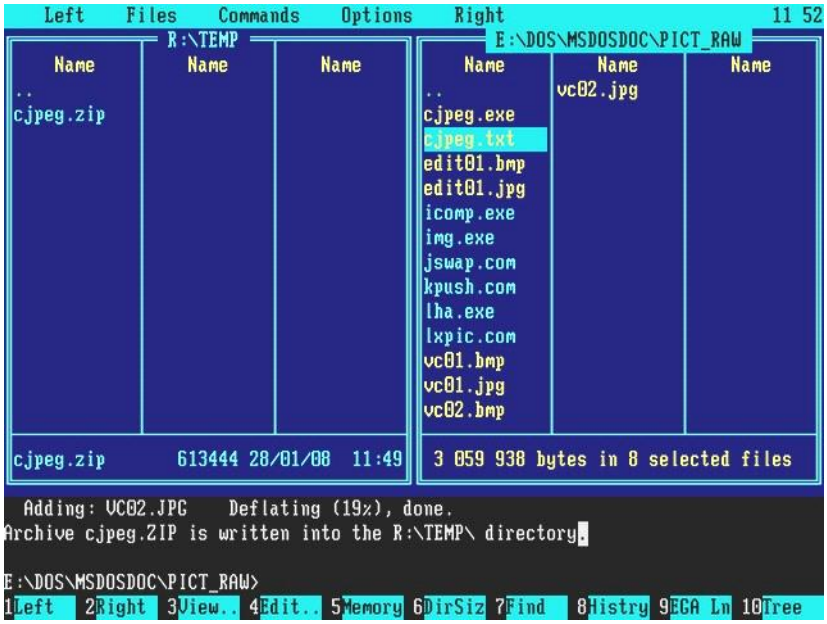


Рис. 5

Все основные функции файл-менеджера VC.COM доступны как через "горячие клавиши" клавиатуры, так и посредством манипулятора типа "мышь". Активизируемые файл-менеджером "горячие клавиши" перечислены ниже в порядке уменьшения их значимости для неопытного пользователя:

Ctrl-B	– включает/выключает меню функций внизу экрана
F9	– активизирует перечень меню вверху экрана
Ctrl-F1	– включает/выключает левую панель
Ctrl-F2	– включает/выключает правую панель
Tab	– активизирует попеременно левую и правую панели
Alt-F1	– выбор диска в левой панели
Alt-F2	– выбор диска в правой панели
Ctrl-O	– включает/выключает обе панели сразу
Ctrl-Q	– включает/выключает окно просмотра в неактивной панели
Ctrl-L	– включает/выключает окно сведений в неактивной панели
Ctrl-[	– вписывает в командную строку путь из левой панели
Ctrl-]	– вписывает в командную строку путь из правой панели
Ctrl-i	– вписывает в командную строку имя выбранного файла

Ctrl-P – включает/выключает активную панель
Ctrl-U – меняет панели местами
Стрелки – смещают выбор пункта меню или файла в активной панели
Home – смещает выбор на начало каталога в активной панели
End – смещает выбор на конец каталога в активной панели
F2 – выводит на экран меню пользователя
F3 – открывает избранный файл в окне просмотрщика
F4 – открывает избранный файл в программе редактирования
F5 – копирует избранные файлы в противоположную панель
Ctrl-F5 – копирует как F5, но только один файл, а не группу
F6 – перемещает или переименовывает избранные файлы
Ctrl-F6 – действует как F6, но только на один файл, не на группу
F7 – позволяет создать каталог
F8 – удаляет избранный файл или группу файлов
Ctrl-F8 – удаляет как F8, но только один файл, а не группу
Ins – добавляет выделенный файл в группу
Shift-F9 – записывает текущие установки параметров в файл VC.INI
Ctrl-A – показывает атрибуты файла и позволяет их изменить
Ctrl-C – сопоставление каталогов с выделением непарных файлов
Ctrl-E – повторяет последнюю из списка предыдущих командных строк
Ctrl-F – позволяет задать суффикс для выделения файлов по маске
Ctrl-N – меняет вид представления имен файлов в панелях
Ctrl-R – повторно считывает текущий каталог текущего диска
Ctrl-F4 – позволяет изменить метку текущего диска
Ctrl-F9 – позволяет изменить видеорежим (и Alt-F9 тоже)
Ctrl-\ – переход в корневой каталог диска в активной панели
Alt буква – поиск файла по буквам имени в текущем каталоге
Alt-F6 – размер выбранного каталога (в "full"-режиме панели)
Alt-F8 – показ списка предшествовавших командных строк
Alt-F10 – показ "дерева" каталогов, с переходом в любой каталог
Alt-F11 – позволяет уменьшить длину панелей клавишей "стрелка вверх"
Alt-F12 – позволяет увеличить длину панелей клавишей "стрелка вниз"
F10 – завершает сеанс работы с файл-менеджером VC.COM.

Действие некоторых "горячих клавиш" дублируется. Так, Ctrl-J и Ctrl-Enter действуют так же, как Ctrl-i: вписывают имя выделенного файла в командную строку. Одинаково показывают "дерево" каталогов Ctrl-Z и ALT-F10. Нуль "0" в группе цифровых клавиш в правой части клавиатуры действует как клавиша INSERT: добавляет избранный элемент к группе элементов, подготавливаемой для выполнения групповой операции. В панелях элементы такой группы выделяются цветом. Несколько клавиш в цифровой (правой) части клавиатуры наделены дополнительными функциями для подготовки групповых операций:

Ctrl + – включить в состав группы все файлы текущего каталога
\+ – включить в группу файлы, выбираемые по маске
Ctrl – – распустить подготовленную группу файлов
– – удалить из группы файлы, соответствующие заданной маске
Ctrl * – инвертировать выбор файлов в текущем каталоге
* – создать маску для выбора файлов, если группа еще не сформирована, или для удаления, если группа уже имеется.

Когда в каталоге группа файлов уже сформирована, тогда операции копирования (клавишей F5), перемещения (клавишей F6), удаления (клавишей F8) выполняются над всеми элементами – файлами и подкаталогами – входящими в состав группы. Список состава группы находится во временном файле VCLIST.000, который помещен в каталог для временных файлов, заданный значением переменной окружения %Temp%. Из конфигурационных файлов с суффиксами *.MNU, *.EXT список состава группы доступен посредством специальных макрокоманд "!~@" (для активной панели) и "%~@" (для пассивной панели). Благодаря этому пользователь получает возможность формулировать свои собственные групповые операции (примеры – в разделе 6.25-02).

Когда обе панели выключены, файл-менеджер VC.COM предоставляет дополнительные функции обслуживания командной строки: левая и правая курсорные стрелки позволяют перемещаться вдоль командной строки, вставлять знаки в любое место командной строки, а курсорные стрелки вверх и вниз обеспечивают доступ к списку предшествовавших командных строк. Если флаг NUMLOCK выключен, то перечисленные выше функции дублируются клавишами стрелочек в цифровой группе клавиш в правой части клавиатуры.

Доработка описываемой версии 4.99.07 файл-менеджера VC.COM не доведена до конца. Некоторые функции, определяемые пользователем посредством конфигурационных файлов, могут исполняться неправильно, если в текущем каталоге находятся другие файлы с суффиксами *.EXT и *.MNU, а также если в неактивной панели оставлен открытым архивный файл. Ряд заявленных "горячих клавиш" не действует или действует не так, в частности:

Alt-F4 – редактирование: не действует, пользуйтесь клавишей F4
Alt-F5 – сводка памяти: не действует
Alt-F7 – поиск файлов: действует только с версии 4.99.08
Ctrl-M – воссоздание группы: не действует
Ctrl-N – длинные имена: редко, но может вызывать зависание
Ctrl-Z – дерево каталогов: может идти без остановки до нажатия ESC
F1 – встроенная помощь: недоступна
F6 – перевод, переименование: при перемещении каталогов нарушает работу манипулятора "мышь", вынуждает перезагружать драйвер "мыши".

6.25-02 Меню файл-менеджера Volcov Commander.

Широкие возможности адаптации файл-менеджера Volcov Commander достигнуты благодаря обслуживанию файлов меню (*.MNU) и файлов расширений (*.EXT) разнообразным набором макрокоманд. Вызов макрокоманды происходит при опознании в командной строке определенной группы знаков, которая затем замещается возвращаемым макрокомандой результатом - именем, путем, строкой и т.п. Вот перечень групп знаков и соответствующих им макрокоманд, которые предоставляет файл-менеджер Volcov Commander версии 4.99.07.

`![prompt]` – приглашение к вводу строки знаков. Слова, указанные в квадратных скобках, выводятся как подсказка перед этим приглашением. Допускается ввод до 10 независимых строк.

`!0...!9` – вставка копии введенной с клавиатуры строки с запрошенным номером, от 0-й до 9-й.

`!:` – вставка буквы диска, относящейся к активной панели

`%:` – вставка буквы диска, относящейся к пассивной панели,

`!~\` – вставка пути, относящегося к активной панели. Путь кончается знаком обратной косой черты.

`%~\` – вставка пути, относящегося к пассивной панели. Путь кончается знаком обратной косой черты.

`!~` – вставка краткого имени файла, выделенного нажатием левой кнопки "мыши" в активной панели.

`%~` – вставка краткого имени файла, выделенного нажатием левой кнопки "мыши" в пассивной панели.

`!~.!` – вставка краткого имени и суффикса файла, выделенного нажатием левой кнопки "мыши" в активной панели.

`%~.%` – вставка краткого имени и суффикса файла, выделенного нажатием левой кнопки"мыши" в пассивной панели.

`!~@` – вставка имени временного файла, содержащего список кратких имен, выделенных в активной панели правой кнопкой "мыши".

`%~@` – вставка имени временного файла, содержащего список кратких имен, выделенных в пассивной панели правой кнопкой "мыши".

`!!` – замещение одним восклицательным знаком.

`%%` – замещение одним знаком процента.

Комбинации знаков, включающие знак тильда (~) "обрезают" длинные имена до 8 знаков, а суффиксы – до 3 знаков. В таких комбинациях знак тильда можно не указывать, и тогда в "окне DOS" операционных систем Windows-95/98/ME те же макрокоманды возвращают полные, не "обрезанные" длинные имена.

Пользователю предоставлена возможность самостоятельно вводить знаки вызова макрокоманд в строки вызова программ при компоновке меню с помощью любой программы редактирования неформатированных текстов. Каждый пункт

меню представлен в файле VC.MNU строкой заголовка и не менее чем одной исполняемой строкой. Исполняемыми строками меню считаются те непустые строки, которые начинаются со знака пробела. Строки заголовков пунктов меню напротив, не должны начинаться со знака пробела, причем начальная группа знаков (до первого знака двоеточия) интерпретируется как спецификация "горячей" клавиши для вызова данного пункта меню. Группа слов в той же строке правее знака двоеточия интерпретируется как наименование пункта меню.

После выбора пункта меню файл-менеджер замещает знаки вызова макрокоманд возвращаемыми ими данными во всех исполняемых строках, следующих за заголовком выбранного пункта меню, а потом посылает эти строки по одной командному интерпретатору COMMAND.COM, причем вызывает его заново для каждой строки. По этой причине, кстати, значения переменных окружения от одной строки к другой не передаются. При составлении исполняемых строк файла VC.MNU обычно предполагают, что удовлетворены все основные условия исполнения batch-файлов, перечисленные во вводной статье к главе 9.

Ниже приведен пример файла VC.MNU. Упоминаемые в нем программы Edit.com (6.09), Fc.exe (6.12) и Scandisk.exe (6.21) входят в комплект поставки Windows-95/98, а файлы Arc.bat и Turn_off.com – это те, которые описаны в разделах 9.03-01 и 9.05-02. Все они должны находиться в каталогах, перечисленных в значении переменной %PATH% (2.02-02). Файла справки у файл-менеджера Volcov Commander нет, поэтому текстовый файл Help.txt, упоминаемый в первом пункте меню, будет содержать то, что Вы напишете сами. Но указанный перед ним путь должен соответствовать действительности, причем обязательно на том же диске, с которого работает интерпретатор COMMAND.COM.

```
F1:  Help
     @for %%Z in (%%comspec%%) do %%Z\
     @Edit.com /R \DOS\VC4\Help.txt
     @!:\
F4:  Search for a file or filemask in the current subtree
     @rem ![Type filename or mask to search for & press ENTER:]
     @echo.
     @if not !0"==" Dir !0 /P /A:-D /S /B
F5:  Compare a file with synonymous file in non-active panel
     @if !:!~\==%:%~\ echo Other panel must show other folder!!
     @if not !:!~\==%:%~\ Fc /L /N !~.! %:%~\!~.! > %%Temp%%\Y.t
     @if not !:!~\==%:%~\ Edit.com %%Temp%%\Y.t
     @if not !:!~\==%:%~\ del %%Temp%%\Y.t
F6:  Pack file(s) marked in the active panel into a ZIP-archive
     @Arc.bat ZIP !: !~\ !~ !~@ %: %~\
F7:  Pack file(s) marked in the active panel into a RAR-archive
     @Arc.bat RAR !: !~\ !~ !~@ %: %~\
```

```
F8:  Repair a disk with Scandisk
     @rem ![Type diskletter of the disk to repair & press Enter]
     @if not !0"==" Scandisk.exe !0: /custom
F10: Switch the PC off
     cls
     @Turn_off.com
```

Каждая группа исполняемых строк, относящаяся к одному пункту меню, является почти batch-файлом. В первом пункте меню задача состоит лишь в выводе на экран заранее заготовленного текста. Однако при сценариях с перебазированием MS-DOS7 на другой диск (9.04, 9.09) этот текст тоже будет перенесен, так что букву диска заранее предсказать невозможно. Поэтому здесь задача решена иначе: первая исполняемая строка делает текущим диском тот, где находится командный интерпретатор, а во второй строке путь к заготовленному тексту дан без указания буквы диска, то есть от корневого каталога любого диска, который в данный момент принимается по умолчанию. Последняя исполняемая строка первого пункта меню снова сделает текущим тот диск, который был установлен прежде.

Второй пункт меню, вызываемый клавишей F4, представляет собой простой запрос данных от пользователя, а также пример исполнения команды по условию поступления непустого ответа. То же самое можно сказать и о шестом пункте меню, вызываемом клавишей F8. Вызов программы Scandisk.exe через меню полезен потому, что при вызове из командной строки часто забывают указывать параметр /custom, а от него существенно зависит характер выполняемой проверки диска.

Третий пункт меню, вызываемый клавишей F5, реализует очень полезную процедуру выявления различий в одноименных неформатированных текстовых файлах, представленных в левой и правой панелях файл-менеджера. Первая исполняемая строка проверяет, действительно ли в разных панелях файл-менеджера открыты разные каталоги. Если да, то во второй исполняемой строке программа Fc.exe сравнит представленные файлы, а отчет будет направлен во временный файл. В третьей исполняемой строке программа Edit.com покажет отчет на экране, а в четвертой строке временный файл отчета будет удален.

К сожалению, в описанной процедуре выявления различий реализованы далеко не все проверки, какие следовало бы провести. Дело в том, что файл-менеджер Volcov Commander выполняет подстановки возвращаемых макрокомандами данных в собственном буфере длиной 128 байт. При возврате макрокомандами длинных путей и имен возрастает риск "обрезания" конца строки до того, как она будет передана для исполнения командному интерпретатору COMMAND.COM. Эта опасность вполне реальна, в частности, для второй исполняемой строки с вызовом программы Fc.exe. Кроме того, вызов посредством переменной %PATH% (2.02-02) менее надежен по причинам, изложенным во вводной статье к главе 6. Наконец,

интерпретация длинных последовательностей командных строк из файла VC.MNU происходит довольно медленно, особенно когда DOS работает с дискеты.

Для осуществления процедур с большим числом проверок лучше запускать из файла меню заранее подготовленные batch-файлы. Примером такого подхода является batch-файл Arc.bat (9.03-01), вызываемый из исполняемых строк 4-го и 5-го пунктов меню. Batch-файлы дают возможность активно пользоваться переменными окружения, условными переходами и точной беспоисковой адресацией, благодаря чему удается снять упомянутые выше ограничения, повысить надежность и выполнять процедуры гораздо быстрее.

Примечание 1: в исполняемой строке пункта меню вместо вызова программы файл-менеджер Volcov Commander допускает указывать произвольно назначенное имя другого файла меню (субменю), которое должно, однако, иметь суффикс ".MNU". Потенциально это дает возможность строить тематические иерархии меню.

Примечание 2: в файлах меню (*.MNU) все строки, начинающиеся со знака одиночной кавычки ('), не исполняются. Это позволяет вводить в файлы меню строки комментариев.

6.25-03 Файлы расширения VC.EXT и VCEDIT.EXT.

Когда в панели файл-менеджера Volcov Commander выделен какой-либо файл, совершаемые с ним действия могут быть разными в зависимости от суффикса этого файла. Зависимые от суффикса действия файл-менеджера в ответ на двойной щелчок левой клавиши "мыши" (а также на нажатие клавиши ENTER) задаются текстовым файлом VC.EXT. Аналогичным образом зависимые от суффикса действия в ответ на нажатие клавиши F3 (VIEW) задаются файлом VCVIEW.EXT, а действия в ответ на нажатие клавиши F4 (EDIT) – файлом VCEDIT.EXT.

Определения суффиксов во всех упомянутых файлах расширения должны быть указаны вплотную к левой кромке строки и должны заканчиваться двоеточием. Справа от двоеточия записывается командная строка, которая должна быть исполнена в случае обращения к файлу с данным суффиксом. Если нажатие клавиши должно вызывать исполнение более чем одной команды, то следующие командные строки, располагаемые под первой командной строкой, должны начинаться с не менее чем одного знака пробела. Если вслед за определением суффикса командная строка не прописана, то для файлов с таким суффиксом будет исполнена ближайшая следующая командная строка. В определениях суффиксов допускается указывать знаки подстановки "?" и "*" (2.01-03).

Какие конкретно вызовы программ следует включать в файлы расширения – это каждый пользователь должен определить для себя сам. В качестве примера ниже приведен вариант текста файла VC.EXT.

```
bas: Qbasic.exe /run !~.!
bmp:
gif:
jpg:
pcx: Lxpic.com !~.! /A
1st:
diz:
doc:
lsm:
me:
rus:
txt:
?!!: @echo _______ > !@\..\t.txt
      @copy !@\..\t.txt /B + !~.! !@\..\.
      @Emagic.exe -q -n3 !@\..\t.txt
      @Svtxt.com !@\..\t.txt
htm: @echo _______ > !@\..\t.txt
      @Html2txt.com < !~.! >> !@\..\t.txt
      @Emagic.exe -q -n3 !@\..\t.txt
      @Svtxt.com !@\..\t.txt
scr: @rem ![Press ENTER to get listing or ESC to quit]
      @echo Processing goes on. Wait...
      @Debug.exe < !~.! > !@\..\Listing.txt
      @Edit.com !@\..\Listing.txt !~.!
ima:
wbt: Diskimg.mnu
```

Представленный вариант файла VC.EXT, как и показанные ниже другие файлы расширения, не имеют целью продемонстрировать все разнообразие операций, которое можно изобрести. Начинается файл с самого простого: командные файлы языка Qbasic направляются на исполнение к своему интерпретатору. Его можно взять из комплекта поставки MS-DOS6.22 или из файла архива OLDDOS.EXE, который выложен в сети Интернет на сервере ftp://ftp.microsoft.com/softlib/mslfiles/ . Напомним: для запуска программы QBASIC.EXE в среде MS-DOS7 должна быть внесена соответствующая запись в таблицу драйвера SETVER.EXE (5.01-02).

Следующая группа строк в файле VC.EXT направляет файлы изображений (*.BMP, *.GIF, *.JPG) к программе просмотра LXPIC.COM, написанной Стефаном Пейхлем (Stefan Peichl). Версия 7.3 этой программы (2002 года) выложена в сети Интернет на сайте http://hplx.pgdn.de/ в составе файла архива LXPIC.ZIP.

Далее в файле VC.EXT перечислена большая группа суффиксов (*.1ST, *.DIZ и др.), обычно относящихся к текстовым файлам. Проблема чтения текстов на русском языке состоит в множестве используемых кодировок. Но ситуацию спасает программа EMAGIC.EXE, написанная Сергеем Гернштейном. Она анализирует

текст и по полученным данным автоматически переводит его в кодировку CP866. Эта программа выложена на сервере ftp://ftp.botik.ru/pub/msdos/convert/ в архиве EMAGIC.ZIP. Тем, кто не пользуется кодировкой CP866, программа EMAGIC.EXE не потребуется, и тогда строку с ее вызовом следует удалить.

Просмотр текстовых файлов на экране обеспечивает программа SVTXT.COM, написанная Ло Ханг Че (Lo Hung Che) для проекта FreeDos. Она содержится в файле архива PG116.ZIP, который свободно выложен в сети Интернет на сервере ftp://sunsite.unc.edu/pub/micro/pc-stuff/freedos/files/util/file/pg/ . Эта программа показывает длинные строки сразу в "сложенном" виде и справляется с первыми 64 килобайтами длинных файлов. Препарировать фрагменты просматриваемых файлов программа SVTXT.COM не позволяет, но это можно сделать потом с помощью текстового редактора EDIT.COM (6.09), так как до следующего вызова той же процедуры копия последнего просмотренного файла в кодировке CP866 будет сохранена под именем T.TXT в каталоге для временных файлов.

Просмотр файлов с суффиксом *.HTM на русском языке сопряжен с теми же проблемами и потому обеспечивается так же, только с предварительным устранением знаков гипертекстовой разметки. Для устранения знаков разметки применена программа Шин Чана (Shin Chan) HTML2TXT.COM, скачанная с сайта http://www.vector.co.jp/download/file/dos/net/fh050307.html в составе файла архива HTML2T08.LZH. Тем, кого проблема национальных кодировок не волнует, лучше сразу направлять файлы *.HTM к программе просмотра VH.EXE, выложенной в архиве Viewht25.zip на ftp-сервере ftp://ftp.bu.edu/pub/mirrors/simtelnet/msdos/html/ .

Суффиксом *.SCR часто отмечают командные файлы отладчика DEBUG.EXE, но встречаются и исключения. Кроме того, исполнение командных файлов отладчика при неподходящих обстоятельствах может быть небезопасно. Поэтому для таких файлов предусмотрено выведение запроса. Если пользователь ответит на запрос утвердительно, то файл будет направлен на исполнение отладчику, а выведенный им листинг вместе с исходным командным файлом будет представлен на экране текстовым редактором EDIT.COM (6.09). Эта процедура очень удобна для исправления выявляемых по листингу ошибок ассемблирования (9.07-01).

Файлы с суффиксом "*.IMA" – это образы дискет. По отношению к ним могут быть предприняты две разные операции: запись на дискету или распаковка. Поэтому здесь пришлось прибегнуть к вызову из файла VC.EXT "самодельного" субменю DISKIMG.MNU, в котором запись на диск производится с помощью программы IMG.EXE, а распаковка – с помощью программы DDI2HDD.EXE. Обе они выложены в сети Интернет на сервере ftp://ftp.elf.stuba.sk/pub/pc/utildisk/ в составе файлов архива IMG.ARJ и DDI2HDD.ZIP соответственно. Пример текста субменю DISKIMG.MNU показан ниже; его в виде неформатированного тестового файла надо поместить в тот же каталог, где находятся все другие файлы файл-менеджера Volcov Commander. Напомним, что названия "горячих" клавишей

в файлах меню должны быть указаны вплотную к левому краю строки, без предшествующих пробелов.

```
F4:  Write the image onto diskette in drive A:
     @Img.exe !~.! A:
F8:  Unpack files from the diskette image
     @if not %~\"==" Ddi2hdd.exe !~.! %:%~\. /d /s
     @if %~\"==" echo Archive in passive panel must be closed
     @if not %~\"==" echo Unpacked files are in %:%~\!~ folder
```

При написании своих вариантов файлов расширения важно принимать во внимание, что помимо явно задаваемых действий файл-менеджер Volcov Commander может выполнять ряд зависящих от суффикса действий по умолчанию. В частности, после интерпретации строк файла VC.EXT файлы с суффиксами *.BAT, *.COM и *.EXE по умолчанию направляются на исполнение к командному интерпретатору COMMAND.COM, а архивы дополнительно обрабатываются в соответствии со спецификациями в файле VCARCH.EXT (6.25-04). Если суффиксы таких файлов указать в файле VC.EXT, то действия по умолчанию окажутся перехваченными и исполняться не будут. По этой причине в файле VC.EXT суффиксы архивных и исполняемых файлов, как правило, не указывают.

Аналогичным образом при нажатии на клавишу F3 (VIEW) файл-менеджер Volcov Commander передает своему встроенному просмотрщику только те файлы, которые не перехвачены спецификациями суффиксов в файле VCVIEW.EXT. Встроенный просмотрщик может показывать файлы и в текстовой, и в бинарной форме. Возможность просмотра любого файла "как он есть" ценна сама по себе, и потому я считаю, что файл VCVIEW.EXT не нужен. Но если у Вас мнение иное, то Вы можете по представленному образцу файла VC.EXT сформировать свой файл VCVIEW.EXT. Синтаксис записей в этих файлах абсолютно идентичен.

Когда файл для просмотра или редактирования выбран в панели файл-менеджера, показывающей содержимое архива, тогда нажатия клавишей F3 и F4 инициируют распаковку выбранного файла из архива во временный каталог %TEMP%\VC.000. При этом в строках файлов VCVIEW.EXT и VCEDIT.EXT соответствующая макрокоманда замещает группу знаков !~.! именем распакованного файла с предшествующим путем в тот самый временный каталог. Следовательно, нажатие клавиши F3 позволяет просматривать находящиеся в архиве файлы без их распаковки в явной форме. Аналогичным образом нажатие клавиши F4 позволит совершить над выделенным в архиве файлом любую операцию, определенную в строках файла VCEDIT.EXT.

Однако операция редактирования небезопасна: она может повредить файл, в том числе и такой, который не является распакованной копией. Композиция файла VCEDIT.EXT должна защищать важные не-текстовые файлы от случайных

повреждений. Поэтому исполняемые файлы и архивы в показанном ниже примере перехвачены пустой операцией (REM), устраняющей риск их повреждения.

```
bas: Qbasic.exe !~.!
bin:
cab:
com:
exe:
rar:
zip: @rem
bmp:
gif:
jpg:
pcx: Lxpic.com !~.! /A
*: Edit.com !~.!
```

Файлы изображений защищены другой перехватывающей операцией, которая позволяет просматривать изображения, находящиеся в архивах. Операция для всех остальных файлов определена в последней строке благодаря указанию там знака подстановки "звездочка" (вместо суффикса). Поскольку файл-менеджер Volcov Commander версии 4.99.07 своего текстового редактора не имеет, постольку в последней строке предлагаемого файла VCEDIT.EXT роль редактора исполняет программа EDIT.COM (6.09) из комплекта поставки Windows-95/98.

Примечание 1: все файлы расширения (*.EXT) считываются файл-менеджером Volcov Commander один раз, когда он впервые вызван на исполнение. Изменения, вносимые в файлы расширения, не вступят в силу до тех пор, пока сеанс работы с файл-менеджером не будет закрыт и начат снова.

Примечание 2: в файлах расширения (*.EXT) все строки, начинающиеся со знака одиночной кавычки ('), не исполняются. Это позволяет вводить в файлы расширения строки комментариев.

6.25-04 Файл VCARCH.EXT: спецификации доступа к архивам.

В панелях файл-менеджера Volcov Commander версии 4.99.07 архивы могут быть представлены как каталоги. Это очень удобное свойство реализовано путем перехвата тех сообщений, которые программы-архиваторы посылают в канал STDOUT, и разбора этих сообщений по словам в предопределенном порядке. Когда строка сообщения соответствует заданному порядку, тогда избранные слова из этой строки оказывается возможным идентифицировать как имя архивированного файла, его длину, дату и т.п. Затем на основании этих данных в панели файл-менеджера формируется изображение содержания архива, точно так же, как формируется изображение содержания обычного каталога.

Поскольку разные программы-архиваторы выводят данные в различных форматах, файл-менеджер должен адаптировать порядок разбора строк применительно к каждой программе архивирования в зависимости от суффикса архивного файла. Модели порядков разбора строк задаются файлом VCARCH.EXT. Более того, для каждого типа архива допускается указывать несколько моделей разбора строк. Строка будет принята, если она соответствует хотя бы одной из этих моделей.

Файл VCARCH.EXT состоит из отдельных записей для архивов разного типа. Каждая запись содержит до 5 строк. Первая строка в каждой записи определяет суффикс, соответствующий архивам данного типа, причем суффикс должен быть указан вплотную к левому краю строки. За суффиксом следует знак двоеточия, правее которого указывается модель разбора строк для данного типа архива. Спецификации моделей построены из разделительных знаков и заглавных букв, которым приписаны следующие значения:

A	– атрибуты архивированного файла
C	– размер архивированного файла в сжатом виде
D	– день или дата последнего изменения файла
L	– переход к продолжению разбора в следующей строке
M	– месяц последнего изменения файла
N	– имя архивированного файла
P	– путь в архивированной структуре каталогов
S	– размер архивированного файла
T	– время последнего изменения файла
W	– любое слово, которое следует игнорировать
Y	– год последнего изменения файла
;	– разделитель между разными моделями порядков разбора
' или "	– кавычки, заключающие опознаваемые знаки или слова
\	– обратная косая черта в позиции первого знака модели разбора означает, что эта модель относится к каталогу.

Следующие строки в каждой записи начинаются с не менее чем одного пробела и служат для определения форм запросов к соответствующей программе архивирования для выполнения определенных операций:

2-я строка: выведение содержания архива в канал STDOUT. Эта операция вызывается автоматически при каждой "перерисовке" той панели, в которой открыт архив.

3-я строка: распаковка выделенных файлов из архива. Распаковка вызывается нажатием клавиши F5 (копирование), если архив открыт в активной панели.

4-я строка: добавление выделенных файлов в архив. Добавление вызывается нажатием клавиши F5 (копирование), если архив открыт в пассивной панели.

5-я строка: удаление файлов из архива. Эта операция вызывается нажатием клавиши F8 (удаление).

Операция распаковки, указанная в третьей строке записи, вызывается также нажатием клавиш F3 (просмотр) и F4 (редактирование), но в таких случаях результат распаковки помещается во временный файл и сразу передается той программе, которая определена в установках файлов VCVIEW.EXT и VCEDIT.EXT. Для редко встречающихся архивов, активно пользоваться которыми заведомо не придется, бывает достаточно операций просмотра содержания и извлечения запрашиваемых файлов. В таких случаях длина соответствующей записи в файле VCARCH.EXT может быть сокращена до трех строк.

Во время интерпретации строк в записях файла VCARCH.EXT файл-менеджер не предоставляет доступа к переменным окружения, но вместо того предоставляет возможность пользоваться несколькими особыми макрокомандами, отличающимися от тех, которые доступны при интерпретации строк в файлах расширения. Вот список макрокоманд, которыми можно пользоваться в файле VCARCH.EXT:

!A	– вставка имени файла с предшествующим путем
!T	– вставка пути к каталогу для временных файлов
!F	– вставка имени файла, выделенного левой кнопкой "мыши"
!M	– вставка группы имен файлов, выделенных правой кнопкой "мыши"
!@	– вставка имени временного файла, содержащего список имен файлов, выделенных правой кнопкой "мыши".

В любом случае все программы, упоминаемые в строках файла VCARCH.EXT, должны быть доступны по путям, записанным в значение переменной PATH.

Ниже приведены предлагаемые примеры записей для файла VCARCH.EXT.

```
ARC: N S W C W D T W
     Pkxarc.exe -v !A
     Pkxarc.exe -e !A @!@
BSA:
BSN: W S C W W D "-" M "-" Y T A L N; W S C W W D "-" M "-1" Y T A L N
     Bsa.exe v !A
     Bsa.exe x -S !A @!@
CAB: M "-" D "-" Y T A S N; N ":" W W W W W W
     Extract.exe /d !A
     Extract.exe /e !A !M
```

```
RAR:
R0?:
R1?:
R2?: "*" N L S C W D T A W W W; N L S C W D T A W W W
     Unrar.exe v -r -w!T !A
     Unrar.exe x -r -w!T !A @!@
     Rar.exe a -std -ds -r -rr -ems- -w!T !A @!@
     Rar.exe d -std -r -rr -ems- -w!T !A @!@
TAR: A W S N L
     Tar.exe -tvf !A
     Tar.exe -xnf !A !M !F
UHA: N S D "-" M "-" Y T A
     Uharcd.exe  l -y+ !A
     Uharcd.exe  x !A !M !F
ZIP:
??!:
?$:
??$: S W C W D T W A N
     Pkunzip.exe -v !A
     Pkunzip.exe -d !A @!@
     Pkzip.com -b!T -P -wHS !A @!@
     Pkzip.com -b!T -d !A @!@
1:
2:
3:
4:
Z:   M "-" D "-" Y T S A C N
     Command.com /c Icomp.exe -l -h !A
     Command.com /c Icomp.exe -d -i -h !A !F
```

В поставке файл-менеджера Volcov Commander версии 4.99.07 имеется образец файла VCARCH.EXT, содержащего записи для довольно большого числа разных архивов. Предложенные здесь записи не следует рассматривать как законченный файл, скорее это просто совокупность примеров, которые отличаются от записей в оригинальном файле и могут быть введены туда по мере надобности. Упоминаемые здесь и многие другие программы архивирования, отсутствующие в поставке Windows-95/98, имеются в обширной коллекции программ на сервере ftp://ftp.elf.stuba.sk/pub/pc/pack/ .

6.26 XCOPY.EXE – копирование файлов и каталогов

Программа XCOPY.EXE копирует файлы и каталоги со всеми подкаталогами. Ей необходимо, чтобы в одном каталоге с ней находился вспомогательный файл XCOPY32.EXE. Вот пример пользования программой XCOPY.EXE:

```
XCOPY.EXE D:\TEMP\*.* C:\DOS /A /D /P /S /V /W
```

здесь:

D:\TEMP*.* – пример пути и маски для копируемых файлов.
C:\DOS – пример пути к каталогу назначения.
/A – копировать только файлы с атрибутом "A". Параметр /M означает то же, но со снятием атрибута "A" при копировании.
/D – не копировать файлы, копии которых с той же или более поздней датой обновления уже имеются в каталоге назначения. Если после параметра /D указана дата (например, /D:18/12/2003), то будут копироваться только файлы, имеющие более позднюю дату обновления (примечание 2).
/P – запрашивать разрешение на каждый акт копирования.
/S – копировать каталоги и подкаталоги, за исключением пустых. Если совместно указать параметры /S /E, то пустые каталоги будут копироваться тоже.
/V – сверять каждую копию файла с оригиналом.
/W – задержка исполнения копирования до получения сигнала с клавиатуры: нажатия любой клавиши (для обеспечения возможности смены дискет).

Примечание 1: в операционной среде MS-DOS программа XCOPY.EXE не копирует файлы, имеющие атрибуты H (скрытый) и S (системный). Эту и ряд других функций программа XCOPY.EXE выполняет только в "окне DOS" операционной системы Windows.

Примечание 2: формат даты после параметра /D, зависит от установок, заданных командой COUNTRY (4.05). В любом случае формат даты должен быть такой же, какой выдает команда DATE (3.08).

Примечание 3: в MS-DOS8 файл XCOPY32.EXE переименован в XCOPY32.MOD.

Примечание 4: в рамках проекта FreeDOS в 2003 году Rene Ableidinger написал одноименную программу XCOPY.EXE, которая выложена на сервере ftp://sunsite.unc.edu/pub/micro/pc-stuff/freedos/files/dos/ в составе файла архива RXCOPY2.ZIP. Эта программа, в отличие от описанной выше программы фирмы Microsoft, не нуждается во вспомогательных файлах и способна в среде MS-DOS7 копировать все, включая файлы с атрибутами HS (скрытые и системные).

Глава 7 Ассемблерные команды отладчика Debug.exe

Стоит ли сейчас тратить время на ассемблерные команды архаичного отладчика DEBUG.EXE, когда существуют более совершенные ассемблеры – например, MASM ? Этот вопрос заставляет вспомнить историю. Еще в 1950-х годах были созданы вычислительные машины, способные исполнять сразу несколько прикладных программ, запускаемых с разных терминалов. Каждой программе надо было предоставить не всю оперативную память, а лишь ее часть. В связи с этим были стандартизованы форматы исполняемых файлов с заголовком, в котором каждая программа заявляла необходимое ей пространство оперативной памяти. Тогда же ассемблеры и трансляторы, не обеспечивающие автоматического формирования заголовков исполняемых файлов, стали считать устаревшими. Именно такое отношение к себе унаследовал отладчик DEBUG.EXE.

Более совершенные ассемблеры автоматически формируют заголовки и адреса переходов. Это большое удобство, но у него есть обратная сторона: потеря свободы манипулирования адресным пространством и сегментными регистрами. Такие ограничения несущественны для прикладных программ, но могут препятствовать решению исследовательских и системных задач. По этой причине, кстати, ассемблер MASM откажется компилировать примеры программ из разделов 9.06, 9.08 и 9.10. Подобные не допускаемые ассемблером фрагменты были обнаружены в прошивках BIOS и даже в загрузочных модулях операционной системы Windows-XP. Однако то, что не позволяет сделать ассемблер MASM, можно сделать с помощью отладчика DEBUG.EXE.

При подборе материалов в этой книжке не менее важно то, что для начального знакомства с программированием в учебных целях предпочтительны инструменты более простые и наглядные, чем ассемблеры MASM или TASM. Отладчик DEBUG.EXE выделяется уникальной комбинацией качеств, которая делает его незаменимым для начального обучения системному программированию:

– во-первых, он дает наглядное представление о машинном коде;
– во-вторых, он сочетает в себе средства ассемблирования и отладки;
– в-третьих, он предоставляет доступ к системным областям памяти;
– в-четвертых, он интерактивно взаимодействует с пользователем.

Хотя отладчик DEBUG.EXE может подать на исполнение процессору любой машинный код, тем не менее полного описания всех машинных команд в этой книжке нет. Материал отобран так, чтобы вложиться в разумные рамки объёма и притом по мере возможности упростить Ваше начальное знакомство с ним. Потому основу содержания 7-й главы составляют ассемблерные команды, "понимаемые" версией отладчика DEBUG.EXE из комплекта поставки операционных систем Windows-95/98. Они представляют лишь часть команд современных процессоров

платформы x86, но эта часть играет особую роль. Во-первых, она включает почти все наиболее активно используемые команды. Во-вторых, число компьютеров, которым "понятны" эти команды, сейчас достигло 96% всего парка действующих компьютеров. Совместимость машинных кодов у подавляющего большинства современных компьютеров "держится" именно на том подмножестве команд, которое представлено здесь.

Более "свежие" команды у процессоров разных фирм совпадают не всегда. Тем не менее некоторые из таких команд получили широкое признание и необходимы для настройки современных компьютеров. В порядке исключения несколько таких команд и префиксов также приведены здесь, причем они представлены машинными кодами, чтобы их можно было бы вводить как данные независимо от того, каким вариантом отладчика Вам предстоит пользоваться. В любом случае это не будет препятствовать отладке программ, содержащих такие коды.

Для использования в качестве ассемблера отладчик DEBUG.EXE должен быть переведен в режим ассемблирования (6.05-02). Ассемблерный диалект отладчика имеет особенности, но основные принципы композиции ассемблерных команд везде одинаковы. Командная строка начинается с имени команды, за которым обычно следуют операнды. Между именем команды и операндами иногда требуется вводить маркер типа операнда. Если указаны несколько операндов, то они разделяются знаком запятой. Команды, которые выдают численный результат, замещают им свой первый (левый) операнд, и тогда прежнее содержимое того регистра или той ячейки памяти теряется. Примеры командных файлов с переводом отладчика DEBUG.EXE в режим ассемблирования и с множеством конкретных ассемблерных команд даны в разделах 9.05, 9.06, 9.08, 9.10.

Поскольку альтернативные спецификации ассемблерных команд весьма разнообразны, некоторым переменным и операндам в главе 7 даны легко опознаваемые обозначения, одни и те же во всех последующих описаниях ассемблерных команд. Допустимые подстановки для этих обозначений вместе с пояснениями приведены в следующей таблице.

Таблица 7.00

Обозначения	Пояснения и допустимые подстановки
`bl`	Любой из следующих однобайтовых регистров: AL ,CL, DL ,BL, AH, CH, DH, BH .
`bx`	Любой из следующих двухбайтовых регистров: AX , CX , DX , BX, SP, BP, SI , DI .
`ss`	Любой из сегментных регистров: CS, DS, ES, SS.
`ST`	Регистр ST(0) стека сопроцессора (примечание 1)
`ST(0-7)`	Любой регистр стека сопроцессора от ST(0) до ST(7)
`far`	Маркер 4-байтового адреса (сегмент + смещение).

Продолжение таблицы 7.00

`byte ptr`	Маркер однобайтового операнда (ptr = "pointer")
`word ptr`	Маркер 2-байтового операнда (слова)
`dword ptr`	Маркер 4-байтового операнда
`qword ptr`	Маркер 8-байтового операнда
`tbyte ptr`	Маркер 10-байтового операнда
`f`	Любая одиночная шестнадцатеричная цифра
`±7f`	Любое шестнадцатеричное число от –7Fh до +7Fh
`ff`	Любое шестнадцатеричное число от 00h до FFh
`ffff`	Любое шестнадцатеричное число от 0000h до FFFFh
`aaaa`	Адрес для "коротких" переходов (примечание 2)
`[bp+si+ffff]`	Выражение в квадратных скобках означает адресацию операнда, находящегося в ячейке памяти. Адрес этой ячейки (смещение) определяется путем вычисления выражения. Две группы форм таких выражений, используемых при 16-разрядной адресации, показаны ниже в примечаниях 3 и 4.

Примечание 1: верхний регистр стека арифметического сопроцессора обычно обозначается ST(0), но в некоторых позициях, где вместо него нельзя использовать какой-либо другой регистр, отладчик DEBUG.EXE принимает сокращенное обозначение ST.

Примечание 2: "aaaa" – число, содержащее не более четырех шестнадцатеричных цифр и обозначающее адрес перехода (смещение) в командах "ближней" передачи управления. Это смещение должно быть в пределах окрестности ±7fh от следующей машинной команды, иначе отладчик DEBUG.EXE выдаст сообщение об ошибке.

Примечание 3: выражения первой группы дают смещение, отсчитываемое от сегментного адреса в регистре DS:. Эти выражения представляют собой

либо просто смещение: [ffff]

либо ссылку на число в регистре: [BX], [DI], [SI]

либо сумму чисел из двух регистров: [BX+DI], [BX+SI]

либо сумму со смещением:

[BX±7f], [BX+ffff], [BX+DI±7f], [BX+DI+ffff], [BX+SI±7f], [BX+SI+ffff], [DI±7f], [DI+ffff], [SI±7f], [SI+ffff].

Примечание 4: выражения второй группы отличаются тем, что ссылаются на регистр BP и дают смещение, отсчитываемое относительно сегментного регистра SS:

[BP+DI], [BP+SI], [BP±7f], [BP+ffff], [BP+DI±7f], [BP+DI+ffff], [BP+SI±7f], [BP+SI+ffff].

Примечание 5: в качестве разделительного знака перед именами регистров в выражениях можно использовать знак минус, и результат от этого не изменяется: та же сумма вычисляется в любом случае.

Примечание 6: принимаемый по умолчанию сегментный регистр может быть заменен другим посредством введения префикса (7.02-01).

Примечание 7: если ассемблерная команда не содержит маркера типа операнда ("byte ptr", "word ptr"...), то отладчик DEBUG.EXE определяет тип операнда по регистру, содержащему другой операнд. Обращение к двухбайтовому регистру (AX, BX...) определяет двухбайтовый операнд типа "word", обращение к однобайтовому регистру (AL, AH, BL...) определяет однобайтовый операнд типа "byte". Но когда обращений к регистру в команде нет, наличие маркера типа операнда становится обязательным. В любом случае маркеры типа операнда можно сокращать до двух букв: "by", "wo", и т.д.

7.01 Управляющие инструкции отладчика

При работе в режиме ассемблирования отладчик DEBUG.EXE воспринимает не только ассемблерные команды, но также управляющие инструкции, которые в машинный код не транслируются, но влияют на процесс трансляции и потому могут играть очень важную роль.

7.01-01 DB – инструкция ввода байтов

Инструкция DB ("Data Byte" = байт данных) сообщает отладчику DEBUG.EXE, что данные, следующие за ней в той же строке, не нужно интерпретировать как ассемблерную команду, а нужно просто по байтам ввести их в машинный код. Каждый байт данных представляется двумя шестнадцатеричными цифрами без добавления буквы "h". Помимо того, строка может включать последовательности знаков кода ASCII, заключенные в кавычки или в двойные кавычки. Обе формы представления данных можно чередовать в любом порядке. Последовательности знаков кода ASCII, за исключением окружающих их кавычек, переводятся в шестнадцатеричные цифры байт за байтом. В строках с инструкцией DB комментарии не допускаются. Вот пример пользования инструкцией DB:

```
DB 71 6C 65 'data array'
```

Примечание 1: при дизассемблировании машинного кода могут встретиться байты, которые отладчик DEBUG.EXE не идентифицирует как "известные" ему машинные команды. Тогда "DB" выводится в качестве префикса перед каждым таким байтом.

7.01-02 DW – инструкция ввода слов

Инструкция DW ("data word" = слово данных) сообщает отладчику DEBUG.EXE, что данные, следующие за ней в той же строке, не нужно интерпретировать как ассемблерную команду, а нужно ввести их в машинный код как двухбайтовые слова. Каждое слово должно включать не более четырех шестнадцатеричных цифр. Если в слове менее четырех цифр, оно автоматически будет дополнено нулями в старших разрядах. В составе машинного кода две младших цифры каждого слова образуют первый байт, а две старших цифры того же слова – следующий байт. Строка с инструкцией DW может содержать группы знаков кода ASCII, заключенные в кавычки или двойные кавычки. Эти знаки переводятся в шестнадцатеричные цифры по одному байту на знак, то есть точно так же, как после инструкции DB (7.01-01). В строках с инструкцией DW комментарии не допускаются. Вот пример пользования инструкцией DW:

```
DW 71A0 F01 06D5 "other key" 0FFF
```

7.01-03 ORG – инструкция смены адреса

Инструкция ORG (ORGanize = организовать) сообщает отладчику DEBUG.EXE, что машинные коды ассемблируемых команд, начиная со следующей, надлежит записывать в другой адрес памяти – в тот, который указан в строке вслед за инструкцией ORG. После этого адреса в той же строке может следовать комментарий (7.01-05). На процесс исполнения ассемблированных машинных кодов инструкция ORG не влияет.

При интерактивной работе в режиме ассемблирования инструкция ORG позволяет перемещаться по адресному пространству, чтобы корректировать ошибки и те отсылки вперед, которые нельзя правильно написать заранее.

В пробных командных файлах отладчика инструкция ORG позволяет фиксировать положения точек рестарта и точек назначения переходов (пример – в разделе 9.02-02). Резервирование свободного адресного пространства перед фиксированными позициями точек назначения помогает избежать пересчета адресов, который иначе был бы необходим при каждом внесении исправлений в предыдущие части того же пробного командного файла.

Примеры	Действие
`ORG ffff:ffff`	установить заданные сегментный адрес и смещение
`ORG fff`	задать смещение 0fffh, сегментный адрес не изменять
`ORG ss:ffff`	установить сегментный адрес из ss:, смещение ffffh
`ORG ss:`	установить сегментный адрес из ss:, смещение 0000h

7.01-04 Инструкция "пустая строка"

Абсолютно пустая строка, без команд, без инструкций и без комментариев, сама по себе воспринимается отладчиком DEBUG.EXE как инструкция выйти из режима ассемблирования в обычный режим отладки, при котором DEBUG.EXE не будет воспринимать команды, описываемые в главе 7, но начнет воспринимать команды, описываемые в разделах 6.05-01 – 6.05-23. При вводе ассемблерных команд с клавиатуры для этого достаточно оставить последнюю строку пустой и просто нажать клавишу ENTER. При вводе ассемблерных команд через перенаправление переход в режим отладки будет вызван первой встреченной пустой строкой в ассемблерном тексте. Поэтому перед посылкой файла на исполнение необходимо убедиться, что внутри блока ассемблерных команд пустых строк нет, и что конец блока ассемблерных команд отмечен обязательным наличием пустой строки.

7.01-05 Точка с запятой – инструкция ввода комментариев

Знак точки с запятой " ; ", встреченный отладчиком DEBUG.EXE в любом месте ассемблерной строки, воспринимается как инструкция не переводить в машинный код сам знак точки с запятой и игнорировать все последующие знаки до конца данной строки. Благодаря такому действию знак точки с запятой используется в ассемблерных текстах для введения комментариев к командам.

Примечание 1: если строка содержит только комментарий, начинающийся со знака точки с запятой, то такая строка не считается пустой и не вызывает перехода в режим отладки (7.01-04). Это позволяет вводить в ассемблерные тексты заголовки и многострочные комментарии.

Примечание 2: сообщение отладчика " ^ Error" (= ошибка), указывающее на знак точки с запятой в предыдущей строке, означает, что ошибка имеется в предшествующей части строки. Наиболее вероятно, отладчик DEBUG.EXE считает спецификацию команды незавершенной из-за отсутствия какого-либо обязательного параметра.

Примечание 3: знак точки с запятой нельзя использовать для введения комментариев в строках с управляющими инструкциями DB и DW, а также при работе отладчика DEBUG.EXE вне режима ассемблирования.

7.02 Префиксы

В машинном коде AT-совместимых компьютеров имеются несколько конкретных байтов, которые наделены особым статусом префиксов. Каждый из них не является отдельной машинной командой. Однако байт префикса, встреченный процессором в позиции, предшествующей первому байту кода машинной команды,

заставляет изменить характер ее интерпретации или ее исполнения. Далее одной команды действие префиксов не распространяется.

Отладчик DEBUG.EXE допускает указание префикса отдельной строкой перед той ассемблерной командой, на которую должен действовать префикс, например:

```
CS:
ADD   byte ptr [BX],0F
```

Однако в равной мере допустимо указание префикса перед ассемблерной командой в одной строке:

```
CS: ADD   byte ptr [BX],0F
```

Машинной команде могут предшествовать до четырех префиксов, если их действие не противоречит друг другу. Все префиксы при одной команде должны быть разные, повторение любого префикса не допускается.

7.02-01 Префиксы смены сегмента

В большинстве ассемблерных команд сегментный адрес не указывают. Абсолютные адреса (примечание 2 к 6.05-01) для таких команд процессор вычисляет на основе сегментного адреса из принимаемого по умолчанию сегментного регистра (примечания 3 и 4 к таблице 7.00). Префиксы смены сегмента позволяют назначить желаемый сегментный регистр для вычисления адреса в ходе исполнения машинной команды вместо того сегментного регистра, который был бы взят по умолчанию. Естественно, указывать префиксы смены сегмента допускается только перед теми командами, которым приходится вычислять адрес для обращения к памяти.

Диалект отладчика DEBUG.EXE включает четыре префикса смены сегмента. Они обозначаются по именам соответствующих сегментных регистров, как показано во второй колонке приведенной ниже таблицы. Пример пользования одним из префиксов смены сегмента приведен в разделе 7.02. Многочисленные подобные примеры имеются в ассемблерных текстах в разделах 9.06 и 9.08.

В современных процессорах сегментных регистров не четыре, а шесть. Префиксы для адресации относительно дополнительных сегментных регистров FS и GS "неизвестны" отладчику DEBUG.EXE, но он позволяет вводить эти префиксы посредством инструкции DB и не препятствует отладке программ, содержащих такие префиксы. Естественно, для отладки и исполнения таких программ необходим компьютер с процессором не древнее 80386.

Код	Обозначение	Примечания
2E	CS:	
3E	DS:	
26	ES:	
36	SS:	
64	DB 64	относительно регистра FS:
65	DB 65	относительно регистра GS:

Примечание 1: посредством указания префикса нельзя изменить принимаемую по умолчанию адресацию относительно регистра ES:, в частности, в строковых командах CMPSB, CMPSW, INSB, INSW, MOVSB, MOVSW, SCASB, SCASW, STOSB, STOSW.

7.02-02 LOCK – префикс захвата системной шины,

Префикс LOCK вводит в машинный код префиксный байт F0h, который заставляет процессор выдавать сигнал занятости системной шины и удерживать его до завершения исполнения той машинной команды, перед которой префикс LOCK поставлен. Это бывает необходимо в многопроцессорных вычислительных машинах для предотвращения нескоординированного доступа к совместно используемой памяти. При исполнении кода на обычных однопроцессорных компьютерах префикс LOCK не требуется.

Указывать префикс LOCK допускается только перед выполнением записи в память, в частности, посредством следующих команд: ADC, ADD, AND, DEC, INC, NEG, NOT, OR, SBB, SUB, XCHG, XOR. Но когда эти же команды выполняют операции считывания и регистровые операции, тогда указывать перед ними префикс LOCK не следует, потому что в таких случаях процессор будет реагировать на наличие префикса LOCK вызовом прерывания INT 06 (8.01-07).

Код	Обозначение
F0	LOCK

Примечание 1: процессоры фирмы Intel не допускают сочетаний префикса захвата системной шины с любым из префиксов повторения (7.02-03, 7.02-04) в одной команде.

Примечание 2: современные процессоры, имеющие более 8 управляющих регистров, допускают использование префикса F0h перед операциями обращения к управляющим регистрам (примечание 1 к 7.03-58), однако при этом интерпретируют его не как префикс захвата системной шины, а как префикс обращения к системным регистрам CR8 – CR15.

7.02-03 Префикс повторения REPNZ

Обозначение REPNZ расшифровывается как "REPeat while Not Zero", то есть повторять, пока нет нуля. Префикс REPNZ вызывает циклическое повторение исполнения команды, перед которой он поставлен, причем цикл повторения прекратится, как только будет выполнено хотя бы одно из следующих двух условий:

- исполняемая машинная команда установит флаг ZF в состояние ZR (6.05-15) в результате обнаружения равных операндов.
- число в регистре CX станет равно нулю в результате исчерпания предустановленного там количества повторений.

Префикс REPNE, расшифровываемый как "REPeat while Not Equal" (= повторять, пока нет равенства) принимается отладчиком DEBUG.EXE как эквивалент REPNZ. Оба они вводят в машинный код один и тот же префиксный байт F2h, который заставляет процессор выполнять следующий цикл операций:

- во-первых, проверить условие CX = 0, если оно выполняется, то выйти из цикла повторения, а если не выполняется, то вычесть единицу из числа в регистре CX;
- во-вторых, установить флаг ZF в состояние NZ;
- в-третьих, выполнить машинную команду, перед которой префикс повторения поставлен;
- в-четвертых, проверить, не находится ли флаг ZF в состоянии ZR, если да, то выйти из цикла повторения, а если нет, то вернуться к началу цикла – к проверке состояния регистра CX.

Пока оба проверяемых условия не выполняются, процессор продолжает исполнение цикла снова и снова. Как только любое из условий окажется выполненным, процессор выйдет из цикла повторения и перейдет к исполнению той операции, которая следует за исполнявшейся в цикле.

Префиксы повторения ставят перед строковыми командами, которые при каждом исполнении автоматически увеличивают или уменьшают число в каком-либо индексном регистре, изменяя таким образом от цикла к циклу адрес операнда, к которому они обращаются. Строковые команды CMPSB, CMPSW, SCASB и SCASW влияют не только на индекс, но также на состояние флага ZF. Поэтому указание префикса повторения перед этими командами позволяет осуществить поиск заданных байтов или слов. Когда префикс повторения поставлен перед строковыми командами, не влияющими на состояние флага ZF (INSB, INSW, MOVSB, MOVSW, OUTSB, OUTSW, STOSB, STOSW), тогда эти команды просто будут исполнены заданное число раз, которое следует заранее записать в регистр CX.

Код	Обозначения
F2	REPNE
F2	REPNZ

Примечание 1: если перед командой, кроме префикса повторения, указан еще какой-либо префикс, то процессоры древнее 80386 не всегда возобновляют исполнение незавершенного цикла повторения после прерываний. Когда такого сочетания префиксов нельзя избежать, нужно либо перепроверять условия выхода из цикла, либо на время его исполнения запрещать прерывания (командой CLI, 7.03-12).

Примечание 2: префиксы повторения нельзя применять по отношению к любым не-строковым командам, потому что в таком случае современные процессоры будут интерпретировать их как префиксы расширения состава команд, в частности, для ввода инструкций SSE.

Примечание 3: когда префиксы повторения используются в сочетании с префиксом смены разрядности операндов (7.02-06), тогда заданное число повторений считывается не из регистра CX, а из 32-разрядного регистра ECX. В таких случаях важно не забыть о записи должного значения в старшие разряды 31 – 16 регистра ECX.

7.02-04 Префикс повторения REPZ

Обозначение REPZ расшифровывается как "REPeat while Zero", то есть повторять, пока есть нуль. Префикс REPZ вызывает циклическое повторение исполнения команды, перед которой он поставлен, причем цикл повторения прекратится, как только будет выполнено хотя бы одно из следующих двух условий:

- исполняемая машинная команда установит флаг ZF в состояние NZ (6.05-15) в результате обнаружения отличающихся операндов.
- число в регистре CX станет равно нулю в результате исчерпания предустановленного там количества повторений.

Префиксы REP (REPeat = повторять), REPE (REPeat while Equal = повторять пока равно) и REPZ эквивалентны: все они вставляют один и тот же байт F3h в ассемблируемый машинный код. Встретив байт F3h, процессор выполняет ту же последовательность операций, как и для префикса REPNZ (7.02-03), за исключением того, что начальная установка флага ZF изменена на обратную (ZR) а проверяемое состояние флага ZF заменено на противоположное (NZ). Все другие особенности исполнения команд с префиксом REPNZ, изложенные в разделе 7.02-03 и в примечаниях к нему, в равной мере присущи исполнению команд с префиксами REP, REPE, REPZ.

Код	Обозначения
F3	REP
F3	REPE
F3	REPZ

Примечание 1: префикс REPZ часто применяют совместно с командами CMPSB или CMPSW для сопоставления двух последовательностей знаков – слов, сигнатур или строк. По окончании таких циклов результат выражается оставляемым состоянием флага ZF: установленное состояние (ZR) свидетельствует о совпадении, сброшенное состояние (NZ) констатирует выявление различия.

7.02-05 Префиксы ожидания WAIT и FWAIT

Префиксы WAIT и FWAIT соответствуют одному и тому же префиксному байту 9Bh, который ставится перед командами, осуществляющими пересылку кодов между центральным процессором и асинхронно работающим сопроцессором. Байт 9Bh заставляет центральный процессор ждать поступления сигнала готовности сопроцессора на вход BUSY. Префиксный байт 9Bh должен предшествовать, в частности, каждой команде ESC (7.03-22), а также командам арифметического сопроцессора (7.04), если предполагается исполнять ассемблируемый машинный код на процессоре, не имеющем встроенного арифметического сопроцессора.

Современные центральные процессоры содержат встроенный арифметический сопроцессор с аппаратными средствами синхронизации, так что им прежняя роль префикса WAIT не нужна. Чтобы его игнорировать, нужно сбросить в нуль бит 01h "синхронизация сопроцессора" в управляющем регистре CR0 (A.11-4). По умолчанию бит 01h установлен, и тогда префикс WAIT может повлечь вызов прерывания INT 07, если одновременно установлен флаг переключения задач (бит 03h в регистре CR0). При переключении задач бывает нужно проверить и обработать зарегистрированные сопроцессором исключения. Эта миссия может быть возложена на обработчика прерывания INT 07, и ее исполнение будет обеспечиваться уместным применением префикса WAIT.

Код	Обозначения
9B	FWAIT
9B	WAIT

7.02-06 Префикс смены разрядности операнда.

При работе современных процессоров в реальном режиме они эмулируют действия с 16-разрядными операндами так, как их осуществлял устаревший процессор 8086. Однако фактически регистры общего назначения в современных процессорах 32-разрядные. Иногда требуется, не меняя реального режима работы, получить доступ к полному 32-разрядному операнду. Для этого в числе команд всех 32-разрядных процессоров платформы x86 имеется префикс 66h.

Поскольку отладчик DEBUG.EXE "не знает" префикса 66h, его следует вводить посредством инструкции DB (7.01-01),например:

```
DB   66
SHR  AX,CL
```

В приведенном примере наличие префикса 66h изменит действие команды SHR (7.03-83) так, что она будет выполнять сдвиг во всем 32-разрядном регистре. Если, в частности, предварительно записанное в регистр CL число сдвигов было равно 10h, то содержимое старших разрядов 31 – 16 будет перемещено в младшие разряды 15 – 0 и станет доступно как обычный 16-разрядный операнд.

По отношению к операциям PUSH, PUSHF, POP и POPF действие префикса 66h затрагивает также стек, обеспечивая "заталкивание" или "выталкивание" четырех байтов сразу, причем старшие разряды "заталкиваются" в стек первыми. Выведение старших разрядов через стек можно осуществить, например, так:

```
DB   66
PUSH AX
POP  BX
POP  BX
```

В приведенном примере команда PUSH под действием префикса 66h копирует в стек весь 32-разрядный регистр EAX. Затем первая команда POP "выталкивает" из стека данные, считанные из младших разрядов 15 – 0 регистра EAX, но они не нужны, так как доступны из AX непосредственно. Вторая команда POP, "затирая" эти данные, выводит в регистр BX искомое содержимое старших разрядов 31 – 16 регистра EAX.

Префикс 66h заставляет команду CMP (7.03-14) сравнивать четырехбайтовые операнды, в том числе содержимое 32-разрядных регистров. Если префикс 66h поставлен перед любой командой с операндом в следующих за ней байтах или в адресуемых ячейках памяти, то этот операнд должен быть четырехбайтовым (типа Dword). Отладчик DEBUG.EXE не позволяет вводить четырехбайтовые операнды в ассемблируемый код команд для центрального процессора. При необходимости дополнительные байты данных могут быть введены с помощью инструкции DB (7.01-01).

Примечание 1: программы с использованием префикса 66h нельзя запускать на компьютерах с 16-разрядными процессорами.

Примечание 2: не допускается указывать префикс 66h перед командами с однобайтовыми операндами, в том числе находящимися в любом из однобайтовых регистров (AH, AL, BH и т.д.), а также перед однобайтовыми строковыми командами (CMPSB, INSB, LODSB, MOVSB, OUTSB, SCASB, STOSB). Современные процессоры интерпретируют такие сочетания кодов как инструкции SSE.

Примечание 3: не допускается указание префикса 66h перед операциями обращения к операндам в сегментных регистрах, потому что в 32-разрядных процессорах сегментные регистры остались 16-разрядными. Однако это не касается операций, использующих содержимое сегментных регистров для доступа к ячейкам памяти.

Примечание 4: при отлаживании программ с помощью команд "Proceed" (6.05-14) или "Trace" (6.05-17) отладчик DEBUG.EXE не показывает в качестве очередной исполняемой команды ту, которая следует за префиксом 66h. Тем не менее 32-разрядные процессоры воспринимают префикс вместе со следующей за ним командой и исполняют их сразу, за один шаг.

Примечание 5: когда битом 6 в байте 06h дескриптора сегмента кода (примечание 5 к A.12-2) задан 32-разрядный размер операндов, тогда префикс 66h действует наоборот и указывает на 16-разрядный операнд. Здесь и далее действие префикса 66h рассматривается только при исходном 16-разрядном размере операндов, задаваемом по умолчанию "теневыми" регистрами процессора при его работе в реальном режиме.

7.02-07 Префикс смены разрядности адреса

При тестировании памяти на предмет сбоев и в ряде других задач необходимо получить доступ ко всему адресному пространству без тех ограничений, которые связаны с 32-разрядной адресацией в защищенном режиме. Чтобы в реальном режиме обеспечить доступ ко всему адресному пространству, надо установить соответствующий размер сегмента (пример – в разделе 9.10-01) и разрешить 32-разрядную адресацию. Префикс 67h разрешает 32-разрядную адресацию одной следующей за ним команде.

Префикс 67h "неизвестен" отладчику DEBUG.EXE, и потому его приходится вводить как данные с помощью инструкции DB (7.01-01). Когда перед командой нужно ставить несколько префиксов, префикс 67h обычно занимает место после префикса смены сегмента (7.02-01), но перед префиксом 66h смены разрядности

операнда (7.02-06). Естественно, префикс 67h не имеет смысла перед командами, которые не оперируют адресами ячеек памяти.

Префикс 67h влияет на длину кода многих команд и, помимо того, изменяет интерпретацию всех адресных выражений, перечисленных в примечаниях 3 и 4 к таблице 7.00. Не подвержены этим влияниям только команды с неявной косвенной адресацией: CMPSB, CMPSW, LODSB, LODSW, MOVSB, MOVSW, SCASB, SCASW, STOSB, STOSW. Для ввода всех других команд с префиксом 67h ассемблерные "способности" отладчика DEBUG.EXE непригодны.

В приведенной ниже таблице показано, как 32-разрядные процессоры будут интерпретировать одни и те же машинные коды адресных выражений при наличии префикса 67h и при его отсутствии. Для размещения в таблице отобраны только такие адресные выражения, взаимная замена которых не влияет на длину кода и на вид операции любых команд с косвенной адресацией. Пользуясь показанными здесь соответствиями, отладчика DEBUG.EXE несложно "обмануть", указав ему при ассемблировании то адресное выражение из левой колонки таблицы, которое будет интерпретировано процессором именно так, как Вам нужно.

Без префикса 67h	С префиксом 67h
[BP+DI]	[EBX]
[BX]	[EDI]
[BP+DI±7f]	[EBX±7f]
[BX±7f]	[EDI±7f]
[BP±7f]	[ESI±7f]
[DI±7f]	[EBP±7f]

Примечание 1: программы с использованием префикса 67h нельзя запускать на компьютерах с 16-разрядными процессорами.

Примечание 2: при отлаживании программ с помощью команд "Proceed" (6.05-14) или "Trace" (6.05-17) отладчик DEBUG.EXE не показывает в качестве очередной исполняемой команды ту, которая следует за префиксом 67h. Тем не менее 32-разрядные процессоры воспринимают префикс вместе со следующей за ним командой и исполняют их сразу, за один шаг.

Примечание 3: когда 32-разрядная адресация задана битом 6 в байте 06h дескриптора сегмента кода (примечание 5 к A.12-2), тогда префикс 67h действует наоборот и определяет 16-разрядную адресацию для следующей за ним команды. Здесь и далее действие префикса 67h рассматривается только при исходной 16-разрядной адресации, задаваемой по умолчанию "теневыми" регистрами процессора при его работе в реальном режиме.

7.02-08 Префиксы расширения состава команд.

Набор команд процессоров платформы x86 сложился в результате длительной, более чем полувековой эволюции. На каждом этапе эволюции в набор надо было встраивать новые команды. Потому пришлось использовать отдельные байты в качестве префиксов расширения состава команд. Эти префиксы изменяют дешифрацию последующего кода команды в дешифраторе процессора. У них нет какого-либо характерного признака их общности, за исключением того, что каждый из них представляет отдельную группу разнообразных машинных команд.

Давно, еще в до-микропроцессорные времена, роль префикса расширения состава команд стал играть байт FFh. С него начинаются машинные коды модификаций многих разных команд, описываемых в разделе 7.03. Позже байты D8h – DFh были назначены на роль префиксов для команд сопроцессора, описываемых в разделе 7.04. Еще позже новые команды процессора Pentium были введены посредством префикса 0Fh; расшифровка кодов некоторых из этих команд дана в таблице 6.05-18.

В наше время свободных префиксных байтов просто не осталось. Для потоковых команд SSE современных процессоров была введена альтернативная интерпретация тех комбинаций командных кодов с префиксами 66h (7.02-06), F2h (7.02-03), F3h (7.02-04), которые прежде считались недействительными. А ради 64-разрядных процессоров приходится переводить в класс префиксов байты 40h – 4Fh, которые другими процессорами воспринимаются как команды DEC (7.03-20) и INC (7.03-27). Такие изменения уже нельзя игнорировать, даже если задачи этой книги ограничены знакомством с 16-разрядным программированием. Необходимые меры по обеспечению совместимости машинных кодов некоторых команд с современными процессорами рассмотрены далее в статьях раздела 7.03.

7.03 Команды, исполняемые центральным процессором

7.03-01 AAA – корректировка после суммирования

Команда AAA ("Adjust After Addition") преобразует в регистре AX двоичную сумму, полученную после суммирования двух неупакованных десятичных цифр, в правильное неупакованное десятичное слово, содержащее по одной десятичной цифре на байт (о сложении в упакованном десятичном формате - в 7.03-18).

Команда AAA проверяет, не произошло ли в регистре AX десятичное переполнение, выражающееся либо в установке флага AF в состояние AC, либо в том, что число в четырех младших битах регистра AL превышает 9. Если переполнение не случилось, то команда AAA просто сбрасывает флаг CF в

состояние NC. Если переполнение произошло, то выполняется суммирование AL = (AL+6), AH = (AH+1), а флаги AF и CF устанавливаются в состояния AC и CY соответственно. В любом случае старшие четыре бита в регистре AL сбрасываются в нуль. Состояния флагов OF, SF, ZF и PF не сохраняются, команда AAA оставляет их в неопределенном состоянии.

Код	Пример
37	AAA

7.03-02 AAD – подготовка перед делением

Команда AAD (Adjust AX before Dividing) преобразует в регистре AX неупакованное десятичное слово, содержащее по одной десятичной цифре на байт, в форму двоичного числа с тем, чтобы его можно было бы подвергнуть операции двоичного деления (7.03-21).

Команда AAD вычисляет AL = AL+(10•AH), и затем обнуляет регистр AH. Флаги SF, ZF и PF принимают состояния, соответствующие результату, оставляемому в регистре AL. Состояния флагов OF, AF и CF не сохраняются, команда AAD оставляет их в неопределенном состоянии.

Код	Пример
D5 0A	AAD

Примечание 1: коды "D5 (1-F)(0-9,B-F)" ошибочно дизассемблируются отладчиком DEBUG.EXE как команда "AAD ff".

Примечание 2: числа в упакованном десятичном формате нельзя подготавливать к делению командой AAD, их нужно сначала распаковать.

7.03-03 AAM – корректировка после умножения

Команда AAM (Adjust After Multiplication) преобразует в регистре AX двоичное произведение, полученное в результате умножения двух десятичных неупакованных цифр, в правильное неупакованное десятичное слово, содержащее по одной десятичной цифре на байт. При этом предполагается, что перед умножением оба сомножителя имели байтовый формат, и четыре старших бита в обоих этих байтах содержали только нули.

Команда AAM делит двоичное произведение в регистре AX на 10, записывает частное в регистр AH, а остаток – в регистр AL. Подобным образом команда AAM может преобразовывать двоичные числа до 63h в десятичный эквивалент. Флаги SF, ZF и PF принимают состояния, соответствующие получаемому результату.

Состояния флагов OF, AF и CF не сохраняются, команда AAM оставляет их в неопределенном состоянии.

Код	Пример
D4 0A	AAM

Примечание 1: коды "D4 (1-F)(0-9,B-F)" ошибочно дизассемблируются отладчиком DEBUG.EXE как команда "AAM ff".

Примечание 2: десятичные числа в упакованном формате подвергать умножению нельзя, их нужно сначала распаковать.

7.03-04 AAS – корректировка после вычитания

Команда AAS (Adjust After Subtraction) преобразует в регистре AX двоичную разность, полученную после вычитания десятичных неупакованных цифр, в правильное десятичное неупакованное слово, содержащее по одной десятичной цифре на байт (о вычитании упакованных десятичных чисел – в разделе 7.03-19).

Команда AAS проверяет, не произошло ли в регистре AX десятичное переполнение, выражающееся либо в установке флага AF в состояние AC, либо в том, что число в четырех младших битах регистра AL превышает 9. Если переполнение не случилось, то команда AAS просто сбрасывает флаг CF в состояние NC. Если переполнение произошло, то выполняется вычитание AL = (AL–6), AH = (AH–1), а флаги AF и CF устанавливаются в состояния AC и CY соответственно. В любом случае старшие четыре бита в регистре AL сбрасываются в нуль. Состояния флагов OF, SF, ZF и PF не сохраняются, команда AAS оставляет их в неопределенном состоянии.

Код	Пример
3F	AAS

7.03-05 ADC – сложение с переносом

Команда ADC (ADd with Carry) выполняет сложение двух указанных операндов, принимая во внимание перенос в младшем разряде. Перенос выражается состоянием флага переноса CF, который должен быть сохранен без изменения с момента исполнения предыдущей операции. Состояния флагов OF, SF, ZF, AF, PF и CF приводятся в соответствие с получаемой суммой, которая замещает собою первый операнд.

Команда ADC – двоичная операция, но имеются два исключения. Если первый операнд находится в регистре AX, то команда ADC может быть использована для

сложения неупакованных десятичных цифр, после чего полученная двоичная сумма должна быть преобразована в неупакованное десятичное слово командой AAA (7.03-01). Если первый операнд находится в регистре AL, то команда ADC может быть применена для сложения упакованных десятичных байтов; затем получаемую двоичную сумму упакованных десятичных байтов надлежит преобразовать в правильный упакованный десятичный байт командой DAA (7.03-18).

1-й байт	2-й байт	Байты данных	Примеры
10	(0-B)(0-F)	0-2	`ADC [bp+si+ffff],bl`
10	(C-F)(0-F)		`ADC bl,bl`
11	(0-B)(0-F)	0-2	`ADC [bp+si+ffff],bx`
11	(C-F)(0-F)		`ADC bx,bx`
12	(0-B)(0-F)	0-2	`ADC bl,[bp+si+ffff]`
13	(0-B)(0-F)	0-2	`ADC bx,[bp+si+ffff]`
14		1	`ADC AL,ff`
15		2	`ADC AX,ffff`
80	(1,5,9)(0-7)	1-3	`ADC byte ptr [bp+si+ffff],ff`
80	D(1-7)	1	`ADC bl,ff`
81	(1,5,9)(0-7)	2-4	`ADC word ptr [bp+si+ffff],ffff`
81	D(1-7)	2	`ADC bx,ffff`
83	(1,5,9)(0-7)	1-3	`ADC word ptr [bp+si+ffff],±7f`
83	D(1-7)	1	`ADC bx,±7f`

Примечание 1: отладчик DEBUG.EXE дизассемблирует как команду ADC коды "1(2,3) (C-F)(0-F)" и "82 (1,5,9,D)(0-7)".

7.03-06 ADD – суммирование

Команда ADD выполняет суммирование операндов, игнорируя перенос в младшем разряде. Исходное состояние флага CF не принимается во внимание. Состояния флагов OF, SF, ZF, AF, PF и CF приводятся в соответствие с получаемой суммой, которая замещает собою первый операнд.

Команда ADD – двоичная операция, но имеются два исключения. Если первый операнд находится в регистре AX, то команда ADD может быть использована для сложения неупакованных десятичных цифр, после чего полученная двоичная сумма должна быть преобразована в неупакованное десятичное слово командой AAA (7.03-01). Если первый операнд находится в регистре AL, то команда ADD может быть применена для сложения упакованных десятичных байтов; затем получаемую двоичную сумму упакованных десятичных байтов надлежит преобразовать в правильный упакованный десятичный байт командой DAA (7.03-18).

1-й байт	2-й байт	Байты данных	Примеры
00	(0-B)(0-F)	0-2	`ADD [bp+si+ffff],bl`
00	(C-F)(0-F)		`ADD bl,bl`
01	(0-B)(0-F)	0-2	`ADD [bp+si+ffff],bx`
01	(C-F)(0-F)		`ADD bx,bx`
02	(0-B)(0-F)	0-2	`ADD bl,[bp+si+ffff]`
03	(0-B)(0-F)	0-2	`ADD bx,[bp+si+ffff]`
04		1	`ADD AL,ff`
05		2	`ADD AX,ffff`
80	(0,4,8)(0-7)	1-3	`ADD byte ptr [bp+si+ffff],ff`
80	C(1-7)	1	`ADD bl,ff`
81	(0,4,8)(0-7)	2-4	`ADD word ptr [bp+si+ffff],ffff`
81	C(1-7)	2	`ADD bx,ffff`
83	(0,4,8)(0-7)	1-3	`ADD word ptr [bp+si+ffff],±7f`
83	C(1-7)	1	`ADD bx,±7f`

Примечание 1: отладчик DEBUG.EXE дизассемблирует как команду ADD коды "0(2,3) (C-F)(0-F)" и "82 (0,4,8,C)(0-7)".

7.03-07 AND – логическая операция "И"

Команда AND поразрядно сопоставляет биты операндов и сбрасывает в нуль бит результата, если в соответствующем разряде хотя бы у одного из операндов бит сброшен в нуль. Результат замещает собою первый операнд. Состояния флагов SF, ZF, PF приводятся в соответствие с полученным результатом. Флаги CF и OF сбрасываются в состояния NC (No Carry) и NV (No oVerflow) соответственно.

1-й байт	2-й байт	Байты данных	Примеры
20	(0-B)(0-F)	0-2	`AND [bp+si+ffff],bl`
20	(C-F)(0-F)		`AND bl,bl`
21	(0-B)(0-F)	0-2	`AND [bp+si+ffff],bx`
21	(C-F)(0-F)		`AND bx,bx`
22	(0-B)(0-F)	0-2	`AND bl,[bp+si+ffff]`
23	(0-B)(0-F)	0-2	`AND bx,[bp+si+ffff]`
24		1	`AND AL,ff`
25		2	`AND AX,ffff`
80	(2,6,A)(0-7)	1-3	`AND byte ptr [bp+si+ffff],ff`
80	E(1-7)	1	`AND bl,ff`
81	(2,6,A)(0-7)	2-4	`AND word ptr [bp+si+ffff],ffff`
81	E(1-7)	2	`AND bx,ffff`

Продолжение таблицы 7.03-07

83	(2,6,A)(0-7)	1-3	`AND word ptr [bp+si+ffff],±7f`
83	E(1-7)	1	`AND bx,±7f`

Примечание 1: отладчик DEBUG.EXE дизассемблирует как команду AND коды "2(2-3) (C-F)(0-F)" и "82 (2,6,A,E)(0-7)".

7.03-08 CALL – вызов процедуры

Команда CALL записывает в стек адрес возврата, и затем выполняет переход к подпрограмме по указанному адресу. Состояния флагов при этом не изменяются.

Различают несколько форм команды CALL. Вызов подпрограммы, находящейся вне сегмента кода вызывающей программы, выполняется командой CALL FAR с четырехбайтовым адресом перехода (сегмент : смещение). Вызов подпрограммы, находящейся внутри сегмента кода вызывающей программы, выполняется командой ближнего вызова ("near" CALL), которая не изменяет сегментный адрес и оперирует только с двухбайтовым смещением. Существуют две разные формы команды ближнего вызова с машинными кодами операций FFh и E8h.

Команда CALL ближнего вызова с кодом FFh записывает в стек 2 байта, выражающие состояние регистра IP, и затем замещает прежнее состояние IP величиной смещения, взятого либо из указанных ячеек памяти, либо из указанного регистра общего назначения.

Команда CALL ближнего вызова с кодом E8h, за которым следует слово (2 байта) данных, действует иначе: после сохранения прежнего состояния регистра IP в стеке она суммирует свое слово данных с имеющимся смещением в регистре IP. Когда при ассемблировании отладчик DEBUG.EXE встречает двухбайтовый адрес перехода в ассемблерной команде, он автоматически вычисляет разность между заданным адресом и смещением для следующей команды. Получаемая разность представляет как раз то слово данных, которое записывается после байта E8h в ассемблируемый машинный код.

Поскольку обе формы команды CALL ближнего вызова осуществляют переходы только в пределах текущего сегмента, постольку возврат обратно к вызывающей программе из подпрограмм, вызванных командой CALL ближнего вызова, должен выполняться командой RET (7.03-73), которая восстанавливает из стека только содержимое регистра IP.

Команда дальнего вызова CALL FAR записывает в стек 2 слова (4 байта) из регистров CS:IP, а затем сегмент и смещение из адреса назначения замещают прежние значения и в регистре CS, и в регистре IP. Потому происходит переход в другой сегмент. По той же причине возврат обратно к продолжению вызывающей программы из подпрограмм, вызванных командой CALL FAR, следует выполнять

посредством команды RETF (7.03-74), которая восстанавливает из стека содержимое сразу двух регистров – CS: и IP.

1-й байт	2-й байт	Байты данных	Примеры	Примечания
9A		4	`CALL FAR ffff:ffff`	*1
E8		2	`CALL ffff`	
FF	(1,5,9)(0-7)	0-2	`CALL [bp+si+ffff]`	*2
FF	(1,5,9)(8-F)	0-2	`CALL FAR [bp+si+ffff]`	*2
FF	D(0-7)		`CALL bx`	*3

Примечание 1: в приведенном примере первое число – это сегментный адрес перехода, второе число – смещение. Указывать маркер "FAR" в такой строке вызова не обязательно: в любом случае будет выполнен дальний переход.

Примечание 2: когда команда CALL получает адрес перехода по ссылке, характер операции зависит от наличия в строке вызова маркера "FAR": при его наличии из памяти считываются 4 байта данных, и выполняется дальний переход, а при отсутствии маркера "FAR" из памяти считывается только смещение (два байта данных), и тогда выполняется ближний переход.

Примечание 3: если команда CALL получает адрес перехода из регистра, то в этот регистр заранее должно быть записано смещение. Обращение CALL к 16-битовому регистру всегда вызывает ближний переход.

Примечание 4: отладчик DEBUG.EXE дизассемблирует код "FF D(8-F)" как команду "CALL FAR bx".

Примечание 5: перед командой CALL нельзя ставить префиксы повторения F2h и F3h (7.02-03, 7.02-04).

7.03-09 CBW – преобразование байта в слово

Команда CBW (Convert Byte into Word) выполняет преобразование байтового числа со знаком в однобайтовом регистре AL в двухбайтовое число со знаком (слово) в регистре AX путем заполнения старшего однобайтового регистра AH состоянием знакового бита исходного числа из младшего однобайтового регистра AL. Состояния флагов команда CBW не изменяет.

Код	Пример
98	CBW

7.03-10 CLC – сброс флага переноса

Команда CLC (CLear Carry) сбрасывает флаг переноса CF в состояние NC.

Код	Пример
F8	CLC

7.03-11 CLD – сброс флага направления

Команда CLD (CLear Direction flag) сбрасывает флаг направления DF в состояние UP, при котором смещения в индексных регистрах DI и/или SI в ходе исполнения строковых операций (CMPSB, LODSB, MOVSB, SCASB, STOSB, и т.д.) будут изменяться в сторону увеличения.

Код	Пример
FC	CLD

7.03-12 CLI – сброс флага прерывания

Команда CLI (CLear Interrupt flag) сбрасывает флаг прерывания IF в состояние DI (Disable Interrupts = запретить прерывания). При этом процессор игнорирует все внешние прерывания, кроме немаскируемого прерывания INT 02 (8.01-03).

Код	Пример
FA	CLI

Примечание 1: программные прерывания исполняются командой INT (7.03-28) всегда, безотносительно к состоянию флага IF.

Примечание 2: команда CLI не выполняется из программ, уровень привилегий которых ниже уровня, установленного для операций ввода-вывода в битах 0Ch и 0Dh регистра флагов (A.11-4).

7.03-13 CMC – реверсирование флага переноса

Команда CMC (Complementary Carry) изменяет любое текущее состояние флага переноса CF на противоположное: NC (No Carry) на CY (CarrY) или наоборот.

Код	Пример
F5	CMC

7.03-14 CMP – сравнение операндов

Команда CMP (CoMPare) приводит флаги OF, SF, ZF, AF, PF и CF в соответствие с разностью между первым операндом (уменьшаемым) и вторым операндом (вычитаемым). При этом сама разность не никуда не записывается, оба операнда сохраняют свои значения.

Интерпретация состояний флагов, оставляемых командой CMP, зависит от того, были ли операнды числами со знаком или без знака. После сопоставления чисел без знака следует пользоваться командами условных переходов JA, JB, JBE, JNB, а после сопоставления чисел со знаком – командами JG, JGE, JL, JLE. Полные имена всех команд условного перехода отражают статус первого (левого) операнда по отношению ко второму (правому) операнду, например, JA (Jump if Above = перейти, если выше) означает, что для исполнения перехода левый операнд команды CMP должен быть больше, чем ее правый операнд.

1-й байт	2-й байт	Байты данных	Примеры
38	(0-B)(0-F)	0-2	`CMP [bp+si+ffff],bl`
38	(C-F)(0-F)		`CMP bl,bl`
39	(0-B)(0-F)	0-2	`CMP [bp+si+ffff],bx`
39	(C-F)(0-F)		`CMP bx,bx`
3A	(0-B)(0-F)	0-2	`CMP bl,[bp+si+ffff]`
3B	(0-B)(0-F)	0-2	`CMP bx,[bp+si+ffff]`
3C		1	`CMP AL,ff`
3D		2	`CMP AX,ffff`
80	(3,7,B)(8-F)	1-3	`CMP byte ptr [bp+si+ffff],ff`
80	F(9-F)	1	`CMP bl,ff`
81	(3,7,B)(8-F)	2-4	`CMP word ptr [bp+si+ffff],ffff`
81	F(9-F)	2	`CMP bx,ffff`
83	(3,7,B)(8-F)	1-3	`CMP word ptr [bp+si+ffff],±7f`
83	F(9-F)	1	`CMP bx,±7f`

Примечание 1: отладчик DEBUG.EXE дизассемблирует как команду CMP коды "3(A,B) (C-F)(0-F)" и "82 (3,7,B,F)(8-F)".

7.03-15 CMPSB – сопоставление байтов

Команда CMPSB (CoMPare Strings of Bytes) выполняет сравнение пары байтов. Адреса сопоставляемых байтов должны быть заранее загружены в пары регистров DS:SI и ES:DI. Если сравниваемые байты равны, флаг переноса CF сбрасывается в состояние NC (No Carry), а флаг нуля ZF устанавливается в состояние ZR (ZeRo). Если сравниваемые байты неодинаковы, то флаг нуля ZF сбрасывается в состояние

NZ (No Zero), а флаг переноса CF устанавливается в состояние CY (CarrY). Устанавливаемые состояния флагов OF, SF, AF, PF соответствуют результату вычитания сопоставляемых байтов, хотя сам этот результат нигде не сохраняется и не отображается.

После сравнения пары байтов оба смещения – в регистре SI и в регистре DI – увеличиваются на единицу или уменьшаются на единицу: это зависит от состояния флага направления DF, которое следует заранее задать командой CLD (счет вверх, 7.03-11) или командой STD (счет вниз, 7.03-85). Так или иначе, после завершения сравнения одной пары байтов командой CMPSB все значения смещений в регистрах оказываются подготовленными к сравнению следующей пары байтов.

Команде CMPSB часто предшествуют префиксы повторения F2h (REPNZ, 7.02-03) или F3h (REPZ, 7.02-04), которые позволяют исполнять команду CMPSB несколько раз подряд и тем самым сопоставлять группы байтов. Перед командой CMPSB также можно использовать один из префиксов смены сегмента (2Eh, 26h или 36h, 7.02-01); он позволит осуществить адресацию относительно другого сегментного регистра (CS:, ES: или SS:) вместо принимаемого по умолчанию сегментного регистра DS. Сегментный адрес ES другого сравниваемого байта посредством префикса смены сегмента изменить нельзя.

Код	Пример
A6	CMPSB

Примечание 1: при наличии перед командой CMPSB префикса повторения F2h или F3h порядок циклического исполнения операций включает сначала установление флагов по результату сравнения байтов, затем изменение смещений в индексных регистрах DI и SI, и только потом проверку условия выхода из цикла по состоянию флагов. Поэтому смещения в индексных регистрах DI и SI в момент выхода из цикла указывают не на ту пару байтов, которая удовлетворила условию выхода, а на следующую за ней пару байтов.

7.03-16 CMPSW – сопоставление слов

Команда CMPSW (CoMPare Strings of Words) сравнивает два двухбайтовых слова и затем изменяет смещения в индексных регистрах DI и SI на 2, подготавливая тем самым адреса к сравнению следующей пары слов. Под действием префикса 66h смены разрядности операнда (7.02-06) команда CMPSW сравнивает пары четырехбайтовых слов (типа Dword) и изменяет смещения в индексных регистрах на 4. Все остальные особенности команды CMPSW такие же, как у команды CMPSB (7.03-15).

Код	Пример
A7	CMPSW

7.03-17 CWD – формирование четырехбайтового числа.

Команда CWD (Convert Word into Double word) преобразует двухбайтовое число со знаком в регистре AX в четырехбайтовое число со знаком. Для размещения старших разрядов этого числа выделяется регистр DX, причем собственно преобразование выполняется посредством заполнения регистра DX знаковым разрядом из регистра AX. Состояния флагов команда CWD не изменяет.

Код	Пример
99	CWD

7.03-18 DAA – коррекция после сложения

Команда DAA (Decimal Adjustment after Addition) преобразует в однобайтовом регистре AL двоичный результат сложения двух упакованных десятичных байтов в правильное упакованное десятичное число, содержащее две десятичные цифры на байт (о сложении неупакованных десятичных чисел - в разделе 7.03-01).

Двоичная сумма упакованных десятичных байтов может вызывать десятичное переполнение отдельно как в четырех младших, так и в четырех старших битах однобайтового регистра AL. Сначала проверяются младшие 4 бита: если число в них превышает 9 или если флаг AF установлен в состояние AC, то команда DAA выполняет сложение AL=(AL+6). Затем аналогичная проверка выполняется для старших четырех бит однобайтового регистра AL: если число в AL превышает 9Fh или если флаг CF установлен в состояние CY, то команда DAA выполняет сложение AL =(AL+60h). Флаги AF, CF, SF, ZF и PF приводятся в соответствие с получаемым результатом. Состояние флага OF не сохраняется.

Код	Пример
27	DAA

7.03-19 DAS – коррекция после вычитания

Команда DAS (Decimal Adjustment after Subtraction) преобразует в однобайтовом регистре AL двоичный результат вычитания двух упакованных однобайтовых операндов в правильное упакованное десятичное число, содержащее две

десятичные цифры на байт (о вычитании неупакованных десятичных чисел - в разделе 7.03-04).

Двоичная разность упакованных десятичных байтов может вызывать десятичное переполнение отдельно как в четырех младших, так и в четырех старших битах однобайтового регистра AL. Сначала проверяются младшие 4 бита: если число в них превышает 9 или если флаг AF установлен в состояние AC, то команда DAS выполняет вычитание AL=(AL–6). Затем аналогичная проверка выполняется для старших четырех бит однобайтового регистра AL: если число в AL превышает 9Fh или если флаг CF установлен в состояние CY, то команда DAS выполняет вычитание AL=(AL–60h). Флаги AF, CF, SF, ZF и PF приводятся в соответствие полученному результату. Состояние флага OF не сохраняется.

Код	Пример
2F	DAS

7.03-20 DEC – уменьшение на единицу

Команда DEC (DECrement) уменьшает указанный операнд на единицу и приводит флаги OF, SF, ZF, AF и PF в соответствие с полученным результатом. Флаг CF при исполнении команды DEC сохраняет свое прежнее состояние.

1-й байт	2-й байт	Байты данных	Примеры
4(8-F)			`DEC bx`
FE	(0,4,8)(8-E)	0-2	`DEC byte ptr [bp+si+ffff]`
FE	C(8-F)		`DEC bl`
FF	(0,4,8)(8-E)	0-2	`DEC word ptr [bp+si+ffff]`

Примечание 1: байты 4(0-F) могут быть интерпретированы 64-разрядными процессорами как префиксы (7.02-08). Поэтому вместо однобайтового кода 4(8-F) команды "DEC bx" предпочтительно применять двухбайтовый код "FF C(8-F)". Он воспринимается как команда "DEC bx" всеми процессорами платформы x86 и правильно дизассемблируется отладчиком DEBUG.EXE, но при ассемблировании его приходится вводить как данные инструкцией DB (7.01-01).

7.03-21 DIV – деление чисел

Команда DIV (DIVide) выполняет деление чисел без знака (для чисел со знаком надо использовать команду IDIV, 7.03-24). Задаваемый в команде DIV операнд –

это делитель. Если делитель представляет собой байт, то заранее предполагается наличие делимого в регистре AX; после деления частное помещается в AL, а остаток – в AH. Если делитель представляет собой двухбайтовое слово, то заранее предполагается наличие старших разрядов делимого в регистре DX, младших разрядов делимого – в регистре AX; после деления частное помещается в регистр AX, а остаток – в регистр DX. Состояния флагов OF, SF, ZF, AF, PF, CF не сохраняются, после деления они остаются в неопределенном состоянии.

Хотя команда DIV является двоичной операцией, тем не менее неупакованные десятичные двухбайтовые слова могут быть подвергнуты операции двоичного деления, если они заранее преобразованы в приемлемую квази-двоичную форму командой AAD (7.03-02).

1-й байт	2-й байт	Байты данных	Примеры
F6	(3,7,B)(0-7)	0-2	`DIV byte ptr [bp+si+ffff]`
F6	F(0-7)		`DIV bl`
F7	(3,7,B)(0-7)	0-2	`DIV word ptr [bp+si+ffff]`
F7	F(0-7)		`DIV bx`

Примечание 1: если операция деления вызывает переполнение в том регистре, куда должно быть помещено частное, то процессор автоматически вызывает обработчика прерывания INT 00 (8.01-01). Дальнейший ход событий зависит от этого обработчика.

7.03-22 ESC – передача кодов асинхронному сопроцессору.

Команда ESC (= ESCape) использовалась для передачи данных и команд от центрального процессора к асинхронно работающему внешнему сопроцессору, подключенному к системной шине. Каждой команде ESC должен был предшествовать префикс ожидания WAIT (7.02-05), заставляющий центральный процессор ожидать поступления сигнала готовности от адресуемого устройства по шине BUSY. Позднее командам арифметических сопроцессоров были присвоены отдельные наименования (7.04), но оставшиеся машинные коды, начинающиеся с байтов D9h - DFh, отладчик DEBUG.EXE по-прежнему дизассемблирует как команду ESC:

D9 (0,4,8)(8-F), D9 D(1-7), DA (C-F)(0-F), DB (0,4,8,C,D,F)(8-F),
DB (2,3,6,7,A-D,F)(0-7), DB E(4 – F), DD (0,2,6)(8-F),
DD (E,F)(0-F), DE D(8,A-F), DF (0,4,8)(8-F), DF (E,F)(0-F).

Многие из перечисленных кодов уже задействованы в расширенном наборе команд современных арифметических сопроцессоров. Команда ESC, как и все другие сопроцессорные команды (7.04), начинающиеся с байтов D8h - DFh,

исполняется центральным процессором только тогда, когда сброшен бит 02h ("эмуляция сопроцессора") в управляющем регистре CR0 (A.11-4). Если же бит 02h установлен, то в ответ на поступление каждой такой команды центральный процессор генерирует прерывание INT 07 (8.01-08), позволяющее осуществить программную эмуляцию функций сопроцессора или других устройств.

Команда ESC потенциально может быть применена для передачи данных и команд к асинхронно работающим устройствам, однако для современных процессоров такая возможность не документирована. Первый операнд команды ESC – шестнадцатеричный номер, второй операнд считывается из указанного регистра. Интерпретация операндов определяется адресуемым устройством. Состояния флагов команда ESC не изменяет.

1-й байт	2-й байт	Примеры
DA	C(0-7)	`ESC 10,b1`
DA	C(8-F)	`ESC 11,b1`
DA	D(0-7)	`ESC 12,b1`
DA	D(8-F)	`ESC 13,b1`
DA	E(0-7)	`ESC 14,b1`
DA	E(8-F)	`ESC 15,b1`
DA	F(0-7)	`ESC 16,b1`
DA	F(8-F)	`ESC 17,b1`

7.03-23 HLT – останов процессора

Команда HLT вызывает останов процессора, при котором сохраняются адреса в регистрах CS:IP и флаги, обеспечивающие возможность правильной активизации процессора. Вывести процессор из состояния останова можно посредством перезагрузки или по сигналу внешнего прерывания, поступившему либо на вход NMI (8.01-03), либо через контроллер прерываний (8.01-09).

Код	Пример
F4	HLT

Примечание 1: команда HLT исполняется только в программах с высшим (нулевым) уровнем привилегий.

Примечание 2: когда небезразлично, каким именно внешним прерыванием процессор выведен из состояния останова, тогда выяснить номер прерывания можно так, как описано в разделе 8.01-09.

7.03-24 IDIV – деление чисел со знаком

Команда IDIV (Integer DIVide) осуществляет деление целых чисел со знаком (для чисел без знака используют команду DIV, 7.03-21). Задаваемый в команде IDIV операнд – это делитель. Если делитель представляет собой байт, то заранее предполагается наличие делимого в регистре AX; после деления частное помещается в AL, а остаток – в AH. Если делитель представляет собой двухбайтовое слово, то заранее предполагается наличие старших разрядов делимого в регистре DX, младших разрядов делимого – в регистре AX; после деления частное помещается в регистр AX, а остаток – в регистр DX. Знак остатка в регистре DX всегда совпадает со знаком делимого. Состояния флагов OF, SF, ZF, AF, PF, CF не сохраняются, после деления они остаются в неопределенном состоянии.

Хотя команда IDIV является двоичной операцией, тем не менее неупакованные десятичные двухбайтовые слова могут быть подвергнуты операции двоичного деления, если они заранее преобразованы в приемлемую квази-двоичную форму командой AAD (7.03-02).

1-й байт	2-й байт	Байты данных	Примеры
F6	(3,7,B)(8-F)	0-2	`IDIV byte ptr [bp+si+ffff]`
F6	F(8-F)		`IDIV bl`
F7	(3,7,B)(8-F)	0-2	`IDIV word ptr [bp+si+ffff]`
F7	F(8-F)		`IDIV bx`

Примечание 1: если операция деления вызывает переполнение в том регистре, куда должно быть помещено частное, то процессор автоматически вызывает обработчика прерывания INT 00 (8.01-01). Дальнейший ход событий зависит от этого обработчика.

7.03-25 IMUL – умножение чисел со знаком

Команда IMUL (Integer MULtiplcation) осуществляет умножение целых чисел со знаком (для чисел без знака используют команду MUL, 7.03-61). Операнд команды IMUL представляет одного из сомножителей. Если этот операнд – байт, то второй сомножитель должен быть заранее помещен в регистр AL, а результат умножения будет оставлен в регистре AX. Если задаваемый в команде операнд является двухбайтовым словом, то второй сомножитель должен быть заранее помещен в регистр AX, старшие два байта произведения будут оставлены в регистре DX, а младшие два байта – в регистре AX.

При исполнении умножения установление флагов OF и CF в состояния OV и CY свидетельствует о том, что старшая часть произведения в регистре AH или в регистре DX содержит значимые цифры. Напротив, сброс флагов OF и CF в состояния NV и NC означает, что старшая часть произведения заполнена только знаковым разрядом. Состояния флагов SF, ZF, AF и PF не сохраняются, они остаются в неопределенном состоянии.

Умножать командой IMUL можно двоичные и неупакованные десятичные числа. Упакованные десятичные числа необходимо сначала распаковать. В результате умножения неупакованных десятичных чисел получается двоичное произведение, которое следует преобразовать в правильное неупакованное десятичное число командой AAM (7.03-03).

1-й байт	2-й байт	Байты данных	Примеры
F6	(2,6,A)(8-F)	0-2	`IMUL byte ptr [bp+si+ffff]`
F6	E(8-F)		`IMUL bl`
F7	(2,6,A)(8-F)	0-2	`IMUL word ptr [bp+si+ffff]`
F7	E(8-F)		`IMUL bx`

Примечание 1: варианты команды IMUL с указанием двух и трех операндов (коды 69h и 6Bh) не поддерживаются отладчиком DEBUG.EXE.

7.03-26 IN – ввод данных из порта

Команда IN считывает данные из порта. В течение времени ее исполнения процессор выдает сигнал, по которому происходит переключение шин процессора с памяти на устройства ввода-вывода, и обеспечивается асинхронный прием данных. Первый операнд команды IN задает регистр для размещения считанных данных в соответствии с их форматом: регистр AL, если должен быть считан байт, или регистр AX, если должно быть считано двухбайтовое слово. Номер порта определяется вторым операндом, либо находящимся в регистре DX, либо указываемым в команде двухзначным шестнадцатеричным числом. На состояния флагов команда IN не влияет.

1-й байт	2-й байт	Байты данных	Примеры
E4		1	`IN AL,ff`
E5		1	`IN AX,ff`
EC			`IN AL,DX`
ED			`IN AX,DX`

Примечание 1: номера некоторых портов приведены в приложении A.14-1. Запросы к портам с номерами свыше FFh выполняются только через регистр DX.

Примечание 2: команда IN не выполняется из программ, уровень привилегий которых ниже уровня, установленного для операций ввода-вывода в битах 0Ch и 0Dh регистра флагов (A.11-4).

7.03-27 INC – увеличение на единицу

Команда INC (INCrement) увеличивает указанный операнд на единицу и приводит флаги OF, SF, ZF, AF и PF в соответствие с получаемым результатом. Флаг CF при исполнении команды INC сохраняет свое прежнее состояние.

1-й байт	2-й байт	Байты данных	Примеры
4(0-7)			`INC bx`
FE	(0,4,8)(0-7)	0-2	`INC byte ptr [bp+si+ffff]`
FE	C(0-7)		`INC bl`
FF	(0,4,8)(0-7)	0-2	`INC word ptr [bp+si+ffff]`

Примечание 1: байты 4(0-F) интерпретируются 64-разрядными процессорами как префиксы (7.02-08). Поэтому вместо однобайтового кода 4(0-7) команды "INC bx" предпочтительно применять двухбайтовый код "FF C(0-7)". Он воспринимается как команда "INC bx" всеми процессорами платформы x86 и правильно дизассемблируется отладчиком DEBUG.EXE, но при ассемблировании его приходится вводить как данные инструкцией DB (7.01-01).

7.03-28 INT – вызов обработчика прерывания

Команда INT (INTerrupt) передает управление обработчику того прерывания, номер которого определен ее операндом, предварительно обеспечив возможность возврата к исполнению текущей программы. Для этого команда INT выполняет следующую последовательность действий:

– сохраняет в стеке состояние регистра флагов;
– сохраняет в стеке состояние регистра CS;
– рассчитывает и сохраняет в стеке адрес (смещение) следующей команды для восстановления состояния регистра IP;
– сбрасывает флаг IF в состояние DI, перекрывая тем самым поступление запросов через контроллер прерываний;
– сбрасывает очередь команд, которую процессоры формируют на основании опережающей выборки;

– путем умножения номера прерывания на 4 вычисляет смещение той ячейки памяти, где хранится адрес обработчика прерывания;
– считывает адрес обработчика и передает ему управление путем записи его адреса (сегмента и смещения) в регистры CS:IP.

Состав и порядок расположения сохраняемых в стеке данных обеспечивают возврат к текущей программе посредством команды IRET (7.03-30), которая должна быть последней в коде любого обработчика прерывания. После возврата исполнение программы продолжится с команды, следующей за командой INT.

Почти каждому обработчику прерывания для исполнения его миссии должны быть предоставлены определенные условия и исходные данные. Об этом нужно побеспокоиться заранее, до вызова прерывания командой INT. Сведения о функциях и условиях вызова некоторых важных обработчиков прерываний приведены в главе 8.

1-й байт	2-й байт	Байты данных	Примеры
CC CD		 1	`INT 3` `INT ff`

Примечание 1: исходное состояние флага прерывания IF на исполнение команды INT не влияет. Флаг IF влияет только на исполнение внешних запросов, поступающих через контроллер прерываний.

Примечание 2: команда INT 3 (CCh), в отличие от всех остальных программных прерываний, не зависима от уровня привилегий, то есть при любом режиме процессора исполняется так же, как в реальном режиме.

Примечание 3: смещение следующей команды записывается в стек только при исполнении команд INT или INTO. Все остальные не-внешние прерывания записывают в стек текущее состояние регистра IP.

Примечание 4: чтобы обработчик прерывания смог пользоваться данными, сохраняемыми в стеке командой INT, нужно сразу после передачи управления зафиксировать состояние указателя стека (SP) в регистре BP, и тогда адрес [BP+00] укажет на смещение команды, которая будет исполнена после возврата в программу, вызвавшую прерывание. Сегментный адрес этой программы будет в ячейке [BP+02], а состояние флагов – в ячейке [BP+04].

7.03-29 INTO – прерывание при переполнении

Немедленная реакция на переполнение посредством вызова прерывания INT 00 (8.01-01) применима не всегда. Для обеспечения возможности более гибкой и отложенной обработки подобных ошибок введена команда INTO (= INTerrupt if Overflow). Она проверяет флаг переполнения OF, и если этот флаг установлен в

состояние OV (OVerflow), то вызывает на исполнение обработчик прерывания INT 04 (8.01-05) так же, как выполняет прерывания команда INT (7.03-28).

Код	Пример
CE	INTO

Примечание 1: загружаемый по умолчанию обработчик прерывания INT 04 просто возвращает управление в вызвавшую его программу. Если нужна какая-либо иная реакция на переполнение, то следует заранее побеспокоиться о том, чтобы подготовить другой обработчик и вписать его адрес в таблицу прерываний (8.02-18).

7.03-30 IRET – возврат из обработчика прерывания

Команда IRET (Interrupt RETurn) восстанавливает по сохраненным в стеке данным прежние состояния регистров IP, CS и флагов, чем обеспечивается возврат к продолжению исполнения программы, вызвавшей прерывание. Исполнение кода любого обработчика прерывания должно кончаться командой IRET.

Код	Пример
CF	IRET

Примечание 1: установление состояния регистра флагов командой IRET не подвержено тем ограничениям, которые наложены на команду POPF (7.03-68), и дает шанс их обойти.

Примечание 2: команда IRET сбрасывает очередь команд, которую процессоры формируют на основании опережающей выборки, чтобы после возврата следующие команды были дешифрированы по правилам, установленным для вызывающей программы.

7.03-31 JA – переход, если больше

Команда JA (Jump if Above) суммирует байт данных с содержимым регистра IP, если флаги CF и ZF сброшены в состояния NC (No Carry) и NZ (No Zero) соответственно. В результате происходит "короткий" переход к исполнению машинной команды, находящейся в пределах ±7Fh от прежнего значения смещения в регистре IP.

Команда JA используется после выполнения операций над числами без знака, в частности, после команд CMP, SBB, SUB (для чисел со знаком условие "больше" проверяет команда JG, 7.03-35).

Отладчик DEBUG.EXE принимает команду JA также под именем JNBE (Jump if Not Below or Equal), но дизассемблирует код 77h всегда как команду JA.

1-й байт	2-й байт	Байты данных	Пример
77		1	JA aaaa

7.03-32 JB – переход, если меньше

Команда JB (Jump if Below) суммирует байт данных с содержимым регистра IP, если флаг CF установлен в состояние CY (CarrY). В результате происходит "короткий" переход к исполнению машинной команды, находящейся в пределах ±7Fh от прежнего значения смещения в регистре IP.

Команда JB используется для переходов по условию неудачного завершения прерываний, отмечающих неудачу установлением флага CF в состояние CY, а также для переходов по результатам выполнения операций над числами без знака, в частности, после команд CMP, SBB, SUB (для чисел со знаком условие "меньше" проверяет команда JL, 7.03-37).

Отладчик DEBUG.EXE принимает команду JB также под именами JNAE (Jump if Not Above or Equal) и JC (Jump if Carry), но тем не менее дизассемблирует код 72h всегда как команду JB.

1-й байт	2-й байт	Байты данных	Пример
72		1	JB aaaa

7.03-33 JBE – переход, если меньше или равно

Команда JBE (Jump if Below or Equal) суммирует байт данных с содержимым регистра IP, если флаг CF установлен в состояние CY (CarrY) или если флаг ZF установлен в состояние ZR (ZeRo). В результате происходит "короткий" переход к исполнению машинной команды, находящейся в пределах ±7Fh от прежнего значения смещения в регистре IP.

Команда JBE используется после выполнения операций над числами без знака, в частности, после команд CMP, SBB, SUB (для чисел со знаком условие "меньше или равно" проверяет команда JLE, 7.03-38)..

Отладчик DEBUG.EXE принимает команду JBE также под именем JNA (Jump if Not Above), но тем не менее дизассемблирует код 76h всегда как команду JBE.

1-й байт	2-й байт	Байты данных	Пример
76		1	JBE aaaa

7.03-34 JCXZ – переход по нулю в регистре CX

Команда JCXZ (Jump if CX is Zero) суммирует байт данных с содержимым регистра IP, если число в регистре CX равно нулю. В результате происходит "короткий" переход к исполнению машинной команды, находящейся в пределах ±7Fh от прежнего значения смещения в регистре IP.

Поскольку регистр CX обычно играет роль счетчика циклов, постольку команда JCXZ позволяет заранее проверить условие исполнения цикла и обойти его, когда число в регистре CX оказывается равным нулю до входа в цикл.

1-й байт	2-й байт	Байты данных	Пример
E3		1	JCXZ aaaa

7.03-35 JG – переход, если больше

Команда JG (Jump if Greater) суммирует байт данных с содержимым регистра IP, если флаг ZF сброшен в состоянии NZ (No Zero) и если флаги SF и OF находятся в одном состоянии, то есть оба установлены или оба сброшены. В результате происходит "короткий" переход к исполнению машинной команды, расположенной в пределах ±7Fh от прежнего значения смещения в регистре IP.

Команда JG используется после выполнения операций над числами со знаком, в частности, после команд CMP, SBB, SUB (для чисел без знака условие "больше" проверяет команда JA, 7.03-31).

Отладчик DEBUG.EXE принимает команду JG также под именем JNLE (Jump if Not Lower or Equal), но дизассемблирует код 7Fh всегда как команду JG.

1-й байт	2-й байт	Байты данных	Пример
7F		1	JG aaaa

7.03-36 JGE – переход, если больше или равно

Команда JGE (Jump if Greater or Equal) суммирует байт данных с содержимым регистра IP, если флаг знака SF и флаг переполнения OF находятся в одном состоянии, то есть оба установлены или оба сброшены. В результате

происходит"короткий" переход к исполнению машинной команды, расположенной в пределах ±7Fh от прежнего значения смещения в регистре IP.

Команда JGE используется после выполнения операций над числами со знаком, в частности, после команд CMP, SBB, SUB (для чисел без знака условие "больше или равно" проверяет команда JNB, 7.03-40).

Отладчик DEBUG.EXE принимает команду JGE также под именем JNL (Jump if Not Lower), но тем не менее дизассемблирует код 7Dh всегда как команду JGE.

1-й байт	2-й байт	Байты данных	Пример
7D		1	`JGE aaaa`

7.03-37 JL – переход, если меньше

Команда JL (Jump if Lower) суммирует байт данных с содержимым регистра IP, если флаг знака SF и флаг переполнения OF находятся в разных состояниях, то есть если один из них установлен, тогда как другой сброшен. В результате происходит "короткий" переход к исполнению машинной команды, расположенной в пределах ±7Fh от прежнего значения смещения в регистре IP.

Команда JL используется после выполнения операций над числами со знаком, в частности, после команд CMP, SBB, SUB (для чисел без знака условие "меньше" проверяет команда JB, 7.03-32).

Отладчик DEBUG.EXE принимает команду JL также под именем JNGE (Jump if Not Greater or Equal), но дизассемблирует код 7Ch всегда как команду JL.

1-й байт	2-й байт	Байты данных	Пример
7C		1	`JL aaaa`

7.03-38 JLE – переход, если меньше или равно

Команда JLE (Jump if Lower or Equal) суммирует байт данных с содержимым регистра IP, если флаг нуля ZF установлен в состояние ZR (ZeRo) или если флаги SF и OF находятся в разных состояниях, то есть один из них установлен, тогда как другой сброшен. В результате происходит "короткий" переход к исполнению машинной команды, расположенной в пределах ±7Fh от прежнего значения смещения в регистре IP.

Команда JLE используется после выполнения операций над числами со знаком ,в частности, после команд CMP, SBB, SUB (для чисел без знака условие "меньше или равно" проверяет команда JBE, 7.03-33).

Отладчик DEBUG.EXE принимает команду JLE также под именем JNG (Jump if Not Greater), но тем не менее дизассемблирует код 7Eh всегда как команду JLE.

1-й байт	2-й байт	Байты данных	Пример
7E		1	`JLE aaaa`

7.03-39 JMP – безусловный переход

Команда JMP осуществляет переход к исполнению другой машинной команды путем изменения содержания регистра IP отдельно или вместе с изменением содержания регистра CS. Когда команда JMP исполняется с операндом длиной два двухбайтовых слова, то она выполняет "дальний" переход (JMP FAR), при котором первое слово замещает прежний сегментный адрес в регистре CS, а второе слово замещает прежнее значение смещения в регистре IP. Команда JMP с операндом длиной в одно двухбайтовое слово осуществляет "ближний" переход, при котором замещается только смещение в регистре IP. Команда JMP с операндом длиной в один байт выполняет "короткий" переход иначе: байт данных суммируется с прежним значением смещения в регистре IP.

Пока процессор работает в реальном режиме, команда JMP состояния флагов не изменяет.

1-й байт	2-й байт	Байты данных	Примеры	Примечания
E9		2	`JMP ffff`	*1
EA		4	`JMP ffff:ffff`	*2
EB		1	`JMP aaaa`	*1
FF	(2,6,A)(0-7)	0-2	`JMP [bp+si+ffff]`	*3
FF	(2,6,A)(8-F)	0-2	`JMP FAR [bp+si+ffff]`	*3
FF	E(0-7)		`JMP bx`	*4

Примечание 1: когда в ассемблерной команде JMP в качестве адреса перехода указано только смещение, отладчик DEBUG.EXE автоматически вычисляет разность между этим смещением и смещением следующей машинной команды. Если полученная разность находится в пределах ±7Fh, то команда JMP транслируется в машинный код EBh "короткого" перехода, а иначе она транслируется в машинный код E9h "ближнего" перехода.

Примечание 2: в приведенном примере первое число – это сегментный адрес перехода, второе число – смещение. Указывать маркер "FAR" в такой строке вызова не обязательно: в любом случае будет выполнен дальний переход.

Примечание 3: когда команда JMP получает адрес перехода по ссылке, характер операции зависит от наличия в строке вызова маркера "FAR". Когда он имеется, из памяти считывается полный четырехбайтовый адрес, и выполняется дальний переход. А при отсутствии маркера "FAR" из памяти считывается двухбайтовое слово. Оно интерпретируется как смещение, и тогда выполняется ближний переход.

Примечание 4: если команда JMP получает адрес перехода из регистра, то в этот регистр заранее должно быть записано смещение. Обращение JMP к 16-битовому регистру всегда вызывает ближний переход.

Примечание 5: после каждого переключения процессора из реального режима в защищенный и обратно, как правило, сразу же выполняют дальний переход (JMP FAR) на адрес следующей команды в том же сегменте кода. Эта команда JMP FAR нужна не для осуществления перехода, а ради двух других действий: во-первых, она приводит статус числа в регистре CS: (сегментный адрес или селектор) в соответствие с устанавливаемым режимом, и, во-вторых, она сбрасывает очередь команд, которую процессоры формируют на основании опережающей выборки, чтобы не оставлять там команды, дешифрированные по правилам прежнего режима.

Примечание 6: отладчик DEBUG.EXE дизассемблирует коды "FF E(8-F)" как команду "JMP FAR bx".

7.03-40 JNB – переход, если больше или равно

Команда JNB (Jump if Not Below) суммирует байт данных с содержимым регистра IP, если флаг CF сброшен в состояние NC (No Carry). В результате происходит "короткий" переход к исполнению машинной команды, находящейся в пределах ±7Fh от прежнего значения смещения в регистре IP.

Команда JNB используется для исполнения переходов по условию успешного завершения прерываний, которые отмечают успех сбросом флага CF в состояние NC, а также по результатам операций над числами без знака, в частности, после команд CMP, SBB, SUB (для чисел со знаком условие "больше или равно" проверяет команда JGE, 7.03-36).

Отладчик DEBUG.EXE принимает команду JNB также под именем JNC (Jump if Not Carry), но тем не менее дизассемблирует код 73h всегда как команду JNB.

1-й байт	2-й байт	Байты данных	Пример
73		1	`JNB aaaa`

7.03-41 JNO – переход, если нет переполнения

Команда JNO (Jump if No Overflow) суммирует байт данных с содержимым регистра IP, если флаг OF сброшен в состояние NV (No oVerflow). В результате происходит "короткий" переход к исполнению машинной команды, расположенной в пределах ±7Fh от прежнего значения смещения в регистре IP.

1-й байт	2-й байт	Байты данных	Пример
71		1	`JNO aaaa`

7.03-42 JNS – переход по положительному знаку числа

Команда JNS (Jump if not Sign) суммирует байт данных с содержимым регистра IP, если флаг SF сброшен в состояние PL, что соответствует положительному знаку числа. В результате происходит "короткий" переход к исполнению машинной команды, расположенной в пределах ±7Fh от прежнего значения смещения в регистре IP.

1-й байт	2-й байт	Байты данных	Пример
79		1	`JNS aaaa`

7.03-43 JNZ – переход по ненулевому результату или по условию неравенства

Команда JNZ (Jump if Not Zero) суммирует байт данных с содержимым регистра IP, если флаг ZF сброшен в состояние NZ (No Zero). В результате происходит "короткий" переход к исполнению машинной команды, расположенной в пределах ±7Fh от прежнего значения смещения в регистре IP.

Отладчик DEBUG.EXE принимает команду JNZ также под именем JNE (Jump if Not Equal), но тем не менее дизассемблирует код 75h всегда как команду JNZ .

1-й байт	2-й байт	Байты данных	Пример
75		1	`JNZ aaaa`

7.03-44 JO – переход при переполнении

Команда JO (Jump if Overflow) суммирует байт данных с содержимым регистра IP, если флаг OF установлен в состояние OV (OVerflow). В результате происходит "короткий" переход к исполнению машинной команды, расположенной в пределах ±7Fh от прежнего значения смещения в регистре IP.

1-й байт	2-й байт	Байты данных	Пример
70		1	JO aaaa

7.03-45 JPE – переход по четному результату

Команда JPE (Jump if Parity Even) суммирует байт данных с содержимым регистра IP, если флаг четности PF установлен в состояние PE, что соответствует четной сумме битов в младшем байте результата предшествовавшей операции (остальные байты не учитываются). При этом условии происходит "короткий" переход к исполнению машинной команды, расположенной в пределах ±7Fh от прежнего значения смещения в регистре IP.

Отладчик DEBUG.EXE принимает команду JPE также под именем JP (Jump if Parity), но тем не менее дизассемблирует код 7Ah всегда как команду JPE.

1-й байт	2-й байт	Байты данных	Пример
7A		1	JPE aaaa

7.03-46 JPO – переход по нечетному результату

Команда JPO (Jump if Parity Odd) суммирует байт данных с содержимым регистра IP, если флаг четности PF сброшен в состояние PO, что соответствует нечетной сумме битов в младшем байте результата предшествовавшей операции (остальные байты не учитываются). При этом условии происходит "короткий" переход к исполнению машинной команды, расположенной в пределах ±7Fh от прежнего значения смещения в регистре IP.

Отладчик DEBUG.EXE принимает команду JPO также под именем JNP (Jump if Not Parity), но тем не менее дизассемблирует код 7Bh всегда как команду JPO.

1-й байт	2-й байт	Байты данных	Пример
7B		1	JPO aaaa

7.03-47 JS – переход по отрицательному знаку числа

Команда JS (Jump if Sign) суммирует байт данных с содержимым регистра IP, если флаг SF установлен в состояние NG, что соответствует отрицательному знаку числа. В результате происходит "короткий" переход к исполнению машинной команды, расположенной в пределах ±7Fh от прежнего значения смещения в регистре IP.

1-й байт	2-й байт	Байты данных	Пример
78		1	JS aaaa

7.03-48 JZ – переход по нулевому результату или по условию равенства

Команда JZ (Jump if Zero) суммирует байт данных с содержимым регистра IP, если флаг ZF установлен в состояние ZR (ZeRo). В результате происходит "короткий" переход к исполнению машинной команды, расположенной в пределах ±7Fh от прежнего значения смещения в регистре IP.

Отладчик DEBUG.EXE принимает команду JZ также под именем JE (Jump if Equal), но тем не менее дизассемблирует код 74h всегда как команду JZ .

1-й байт	2-й байт	Байты данных	Пример
74		1	JZ aaaa

7.03-49 LAHF – копирование флагов в регистр

Команда LAHF (Load AH with Flags) копирует младший байт из регистра флагов в однобайтовый регистр AH, так что состояние знакового флага SF отображается битом 7 в AH, состояние флага нуля ZF отображается битом 6, состояние дополнительного флага переноса AF отображается битом 4, состояние флага четности PF отображается битом 2, и состояние флага переноса CF отображается битом 0. Биты 5, 3 и 1 однобайтового регистра AH каким-либо флагам не соответствуют. Бит 1 всегда устанавливается в состояние логической единицы, биты 5 и 3 сбрасываются в нуль.

Код	Пример
9F	LAHF

7.03-50 LDS – загрузка регистра DS

Команда LDS (Load address into DS) выполняет загрузку полного адреса из памяти, причем сегментный адрес загружается в регистр DS, а смещение – в другой регистр, указываемый в качестве первого операнда. Второй операнд команды LDS задает смещение, указывающее на ячейку памяти, откуда должен быть считан первый байт загружаемого полного адреса. Байты 2 и 1 полного адреса определяют смещение, байты 4 и 3 – сегментный адрес. Состояния флагов команда LDS не изменяет.

1-й байт	2-й байт	Байты данных	Пример
C5	(1,5,9)(8-F)	0-2	`LDS bx,[bp+si+ffff]`

Примечание 1: отладчик DEBUG.EXE ошибочно дизассемблирует как команду LDS коды "C5 (C-F)(0-F)".

Примечание 2 по умолчанию пара регистров DS:SI считается определяющей адрес источника данных, и потому в качестве первого операнда команды LDS (вместо bx) наиболее часто указывают регистр SI.

Примечание 3: и сегментный адрес в регистре DS:, и смещение могут быть использованы для адресации и перезагружены в одной операции; например, вполне допустима команда DS: LDS SI,[SI].

7.03-51 LEA – вычисление смещения

Команда LEA (Load Effective Address) вычисляет выражение в квадратных скобках, которое задано в качестве второго операнда и определяет некоторое смещение. Результат вычисления загружается в регистр, указанный в качестве первого операнда. Состояния флагов команда LEA не изменяет.

1й байт	2-й байт	Байты данных	Пример
8D	(0-B)(0-F)	0-2	`LEA bx,[bp+si+ffff]`

7.03-52 LES – загрузка регистра ES

Команда LES (Load address into ES) выполняет загрузку полного адреса из памяти, причем сегментный адрес загружается в регистр ES, а смещение – в другой регистр, указываемый в качестве первого операнда. Второй операнд команды LES задает смещение, указывающее на ячейку памяти, откуда должен быть считан первый байт загружаемого полного адреса. Байты 2 и 1 полного адреса определяют смещение, байты 4 и 3 – сегментный адрес. Состояния флагов команда LES не изменяет.

1-й байт	2-й байт	Байты данных	Пример
C4	(1,5,9)(8-F)	0-2	`LES bx,[bp+si+ffff]`

Примечание 1: отладчик DEBUG.EXE ошибочно дизассемблирует как команду LES коды "C4 (C-F)(0-F)".

Примечание 2: по умолчанию пара регистров ES:DI считается определяющей адрес назначения, и потому в качестве первого операнда команды LES (вместо bx) наиболее часто указывают регистр DI.

Примечание 3: и сегментный адрес в регистре ES:, и смещение могут быть использованы для адресации и перезагружены в одной операции; например, вполне допустима команда ES: LES DI,[DI].

7.03-53 LODSB – считывание байта

Команда LODSB (LOaD String of Bytes) считывает из памяти в однобайтовый регистр AL один байт, на который указывает полный адрес, заранее записанный в пару регистров DS:SI. После считывания смещение в регистре SI увеличивается или уменьшается на единицу: это зависит от состояния флага направления DF, которое следует заранее задать командой CLD (счет вверх, 7.03-11) или командой STD (счет вниз, 7.03-85). Состояния флагов команда LODSB не изменяет.

Команде LODSB может предшествовать префикс смены сегмента (7.02-01); он позволит адресовать считываемый байт через другой сегментный регистр вместо принимаемого по умолчанию сегментного регистра DS.

Код	Пример
AC	LODSB

7.03-54 LODSW – считывание слова

Команда LODSW (LOaD String of Words) считывает в регистр AX одно двухбайтовое слово и изменяет смещение в регистре SI на 2, подготавливая тем самым адрес для считывания следующего слова. Под действием префикса 66h смены разрядности операнда (7.02-06) команда LODSW считывает в регистр EAX 4-байтовое слово (типа Dword) и изменяет смещение в регистре SI на 4. Остальные особенности команды LODSW такие же, как у команды LODSB (7.03-53).

Код	Пример
AD	LODSW

7.03-55 LOOP – организация цикла

Команда LOOP сначала уменьшает число в регистре CX на единицу, а потом проверяет, не равен ли остаток нулю. Пока он не равен нулю, будет выполняться "короткий" переход в пределах ±7Fh путем прибавления байта данных к смещению в регистре IP. А когда станет CX = 0, процессор перейдет к исполнению следующей команды. Состояния флагов команда LOOP не изменяет.

Счет циклов в регистре CX исходит из предположения, что команда LOOP следует за телом цикла. В этом случае тело цикла будет исполнено один раз, прежде чем условие вхождения в цикл будет проверено командой LOOP. Чтобы предотвратить исполнение тела цикла до проверки, можно перед телом цикла поставить команду JCXZ (7.03-34). Другое решение состоит в вынесении тела цикла из основной последовательности команд. В последнем случае число, заранее записываемое в регистр CX, должно быть на единицу больше требуемого числа актов исполнения цикла.

1-й байт	2-й байт	Байты данных	Пример
E2		1	`LOOP aaaa`

7.03-56 LOOPNZ – возврат в цикл, если не нуль

Команда LOOPNZ (LOOP if Not Zero) сначала уменьшает число в регистре CX на единицу, а затем проверяет два условия: отличается ли остаток в регистре CX от нуля и сброшен ли флаг нуля ZF в состояние NZ. Когда оба условия выполнены, происходит "короткий" переход в пределах ±7Fh путем прибавления байта данных к смещению в регистре IP. Если же хотя бы одно условие не удовлетворяется, то процессор просто переходит к исполнению следующей команды. Особенности организации циклов командой LOOPNZ такие же, как у команды LOOP (7.03-55). Состояния флагов команда LOOPNZ не изменяет.

Отладчик DEBUG.EXE принимает команду LOOPNZ также под именем LOOPNE (LOOP, if Not Equal = цикл, если не равно), но тем не менее дизассемблирует код E0h всегда как команду LOOPNZ.

1-й байт	2-й байт	Байты данных	Пример
E0		1	`LOOPNZ aaaa`

7.03-57 LOOPZ – возврат в цикл, если нуль

Команда LOOPZ (LOOP if Zero) сначала уменьшает число в регистре CX на единицу, а затем проверяет два условия: отличается ли остаток в регистре CX от нуля и установлен ли флаг нуля ZF в состояние ZR. Когда оба условия выполнены, происходит "короткий" переход в пределах ±7Fh путем прибавления байта данных к смещению в регистре IP. Если же хотя бы одно условие не удовлетворяется, то процессор просто перейдет к исполнению следующей команды. Особенности организации циклов командой LOOPZ такие же, как у команды LOOP (7.03-55). Состояния флагов команда LOOPZ не изменяет.

Отладчик DEBUG.EXE принимает команду LOOPZ также под именем LOOPE (LOOP, if Equal = цикл, если равно), но тем не менее дизассемблирует код E1h всегда как команду LOOPZ.

1-й байт	2-й байт	Байты данных	Пример
E1		1	`LOOPZ aaaa`

7.03-58 MOV – копирование данных

Команда MOV копирует байт или слово данных, определяемых прямо или косвенно вторым операндом, в регистр или в ячейку памяти, задаваемые первым операндом. Явное указание размера копируемых данных – байт или слово – требуется только когда в качестве операнда не указан регистр процессора. Обычные формы команды MOV состояния флагов не изменяют; исключение составляют только обращения к управляющим, отладочным и тестовым регистрам, показанные ниже в примечании 1 (после них состояния флагов OF, SF, ZF, AF, PF, CF могут не сохраняться).

1-й байт	2-й байт	Байты данных	Примеры
88	(0-B)(0-5, 7-F)	0-2	`MOV [bp+si+ffff],bl`
88	(C-F)(0-F)		`MOV bl,bl`
89	(0-B)(0-5, 7-F)	0-2	`MOV [bp+si+ffff],bx`
89	(C-F)(0-F)		`MOV bx,bx`
8A	(0-B)(0-5, 7-F)	0-2	`MOV bl,[bp+si+ffff]`
8B	(0-B)(0-5, 7-F)	0-2	`MOV bx,[bp+si+ffff]`
8C	(0,1,4,5,8,9)(0-F)	0-2	`MOV [bp+si+ffff],ss`
8C	(C,D,E)(0-F)		`MOV bx,ss`
8E	(0,1,4,5,8,9)(0-F)	0-2	`MOV ss,[bp+si+ffff]`
8E	(C,D,E)(0-F)		`MOV ss,bx`
A0		2	`MOV AL,[ffff]`
A1		2	`MOV AX,[ffff]`
A2		2	`MOV [ffff],AL`
A3		2	`MOV [ffff],AX`
B(0-7)		1	`MOV bl,ff`
B(8-F)		2	`MOV bx,ffff`
C6	(0,4,8)(0-7)	1-3	`MOV byte ptr [bp+si+ffff],ff`
C7	(0,4,8)(0-7)	2-4	`MOV word ptr [bp+si+ffff],ffff`

Примечание 1: отладчик DEBUG.EXE "не знает" команд обращения к управляющим и отладочным регистрам 32-разрядных процессоров, но коды этих команд можно вводить как данные с помощью

инструкции DB (7.01-01). Такие команды копирования – трехбайтовые, начинаются с байта 0Fh; второй байт определяет исполнение копирования:

20h – из управляющего регистра (CR0, CR2 – CR4)
21h – из отладочного регистра (DR0 – DR3, DR6, DR7)
22h – в управляющий регистр (CR0, CR2 – CR4)
23h – в отладочный регистр (DR0 – DR3, DR6, DR7)

Третий байт в таких командах указывает на конкретные регистры:

C0h – на CR0 или DR0, например, 0F 20 C0 = MOV EAX,CR0
C8h – на DR1, например, 0F 23 C8 = MOV DR1,EAX
D0h – на CR2 или DR2, например, 0F 20 D0 = MOV EAX,CR2
D8h – на CR3 или DR3, например, 0F 20 D8 = MOV EAX,CR3
E0h – на CR4, например, 0F 22 E0 = MOV CR4,EAX
F0h – на DR6, например, 0F 21 F0 = MOV EAX,DR6
F8h – на DR7, например, 0F 23 F8 = MOV DR7,EAX

Чтобы вместо EAX задействовать какой-либо иной регистр, нужно к 3-му байту прибавить номер этого регистра (от 00h до 07h) в следующем перечне: EAX, ECX, EDX, EBX, ESP, EBP, ESI, EDI, например:

0F 20 C3 = MOV EBX,CR0

Отладчик DEBUG.EXE неспособен дизассемблировать эти коды, но не препятствует отладке программ, которые такие коды содержат, при условии работы на компьютере с 32-разрядным процессором. Префикс смены разрядности операнда перед такими командами не требуется.

Примечание 2: отладчик DEBUG.EXE "не знает" команд обращения к сегментным регистрам GS: и FS:, имеющимся в 32-разрядных процессорах, но коды этих команд можно вводить как данные с помощью инструкции DB (7.01-01). Такие команды копирования – двухбайтовые:

8C E0 = MOV AX,FS
8C E8 = MOV AX,GS
8E E0 = MOV FS,AX
8E E8 = MOV GS,AX

Чтобы вместо AX задействовать какой-либо иной регистр, нужно ко 2-му байту прибавить номер этого регистра (от 00h до 07h) в перечне, приведенном во 2-й строке таблицы 7.00, например:

8E E3 = MOV FS,BX

Отладчик DEBUG.EXE ошибочно дизассемблирует такие коды как относящиеся к регистрам CS: и ES:, но не препятствует правильной отладке программ, которые такие коды содержат, при условии работы на компьютере с 32-разрядным процессором.

Примечание 3: команду MOV нельзя использовать для записи данных в регистр CS:, это можно делать только посредством команд перехода или команд передачи управления (CALL, JMP, RETF и т.п.).

Примечание 4: команда копирования слова данных в регистр SS вызывает аппаратное блокирование внешних прерываний на время исполнения одной следующей команды, которой должна быть команда обновления указателя в регистре SP. Только такой порядок их исполнения исключает сбои из-за возникновения прерываний в моменты переходов на другой стек.

Примечание 5: отладчик DEBUG.EXE дизассемблирует как команду MOV коды "8(A,B) (C-F)(0-F)", "8(C,E) (2,3,6,7,A,B,F)(0-F)" и "C(6,7) (C-F)(0-F)".

7.03-59 MOVSB – копирование байта

Команда MOVSB (MOVe a String of Bytes) копирует один байт из адреса источника в адрес назначения. Оба адреса должны быть подготовлены заранее: адрес источника – в паре регистров DS:SI, адрес назначения – в паре регистров ES:DI. После копирования оба смещения – смещение источника в регистре SI и смещение назначения в регистре DI – увеличиваются на единицу или уменьшаются на единицу: это зависит от состояния флага направления DF, которое следует заранее задать командой CLD (счет вверх, 7.03-11) или командой STD (счет вниз, 7.03-85). Автоматическое изменение смещений в индексных регистрах подготавливает условия для копирования следующего байта в следующую ячейку памяти. Состояния флагов команда MOVSB не изменяет.

Перед командой MOVSB часто ставится префикс повторения REPNZ (7.02-03) или REPZ (7.02-04), что позволяет исполнять команду MOVSB несколько раз и таким образом копировать сразу строку байтов. Также перед командой MOVSB можно ставить один из префиксов смены сегмента (7.02-01); это позволяет отсчитывать адрес источника относительно другого сегментного регистра вместо принимаемого по умолчанию сегментного регистра DS:. Сегментный регистр адреса назначения (ES:) посредством префикса смены сегмента изменить нельзя.

Код	Пример
A4	MOVSB

7.03-60 MOVSW – копирование слова

Команда MOVSW (MOVe a String of Words) копирует двухбайтовое слово и затем изменяет смещение в адресах источника и назначения на 2, подготавливая их

тем самым к копированию следующего двухбайтового слова. Под действием префикса 66h смены разрядности операнда (7.02-06) команда MOVSW копирует по 4 байта сразу и изменяет смещения в индексных регистрах на 4. Все остальные особенности команды MOVSW такие же, как у команды MOVSB (7.03-59).

Код	Пример
A5	MOVSW

7.03-61 MUL – умножение чисел без знака

Команда MUL (MULtiplcation) осуществляет умножение двух целых чисел без знака (для чисел со знаком используют команду IMUL, 7.03-25). Операнд команды MUL представляет одного из сомножителей. Если этот операнд – байт, то второй сомножитель должен быть заранее помещен в регистр AL, а результат умножения будет оставлен в регистре AX. Если задаваемый в команде операнд является двухбайтовым словом, то второй сомножитель должен быть заранее помещен в регистр AX, старшие два байта произведения будут оставлены в регистре DX, а младшие два байта – в регистре AX.

При исполнении умножения установление флагов OF и CF в состояния OV и CY свидетельствует о том, что старшая часть произведения в регистре AH или в регистре DX содержит значимые цифры. Напротив, сброс флагов OF и CF в состояния NV и NC означает, что старшая часть произведения заполнена только нулями. Состояния флагов SF, ZF, AF и PF не сохраняются, эти флаги остаются в неопределенном состоянии.

Умножать командой MUL можно двоичные и неупакованные десятичные числа. Упакованные десятичные числа необходимо сначала распаковать. В результате умножения неупакованных десятичных чисел получается двоичное произведение, которое следует преобразовать в правильное неупакованное десятичное число командой AAM (7.03-03).

1-й байт	2-й байт	Байты данных	Примеры
F6	(2,6,A)(0-7)	0-2	`MUL byte ptr [bp+si+ffff]`
F6	E(0-7)		`MUL bl`
F7	(2,6,A)(0-7)	0-2	`MUL word ptr [bp+si+ffff]`
F7	E(0-7)		`MUL bx`

7.03-62 NEG – изменение знака числа

Команда NEG вычитает операнд из нуля, изменяя тем самым его знак. При этом нулевой операнд не изменяется. Состояния флагов OF, SF, ZF, AF, PF, CF приводятся в соответствие с полученным результатом.

1-й байт	2-й байт	Байты данных	Примеры

F6
F6
F7
F7
	(1,5,9)(8-F)
	D(8-F)
	(1,5,9)(8-F)
	D(8-F)
		0-2
		
		0-2
		
			`NEG byte ptr [bp+si+ffff]`
			`NEG bl`
			`NEG word ptr [bp+si+ffff]`
			`NEG bx`

7.03-63 NOP – команда без операции

Команда NOP (No OPeration) – "пустая" команда, которая не делает ничего, кроме увеличения смещения в регистре IP на единицу.

Код	Пример
90	NOP

7.03-64 NOT – инверсия битов операнда

Команда NOT выполняет логическую операцию НЕ путем инвертирования состояния всех битов операнда. Состояния флагов при этом не изменяются.

1-й байт	2-й байт	Байты данных	Примеры

F6
F6
F7
F7
	(1,5,9)(0-7)
	D(0-7)
	(1,5,9)(0-7)
	D(0-7)
		0-2
		
		0-2
		
			`NOT byte ptr [bp+si+ffff]`
			`NOT bl`
			`NOT word ptr [bp+si+ffff]`
			`NOT bx`

7.03-65 OR – логическая операция "ИЛИ"

Команда OR поразрядно сопоставляет биты операндов и устанавливает бит результата в единицу, если в соответствующем разряде хотя бы у одного из операндов бит установлен в единицу. Результат замещает собою первый операнд. Состояния флагов SF, ZF, PF приводятся в соответствие с полученным результатом. Флаги CF и OF сбрасываются в состояния NC (No Carry) и NV (No oVerflow) соответственно. Состояние флага AF не сохраняется.

1-й байт	2-й байт	Байты данных	Примеры
08	(0-B)(0-F)	0-2	`OR [bp+si+ffff],bl`
08	(C-F)(0-F)		`OR bl,bl`
09	(0-B)(0-F)	0-2	`OR [bp+si+ffff],bx`
09	(C-F)(0-F)		`OR bx,bx`
0A	(0-B)(0-F)	0-2	`OR bl,[bp+si+ffff]`
0B	(0-B)(0-F)	0-2	`OR bx,[bp+si+ffff]`
0C		1	`OR AL,ff`
0D		2	`OR AX,ffff`
80	(0,4,8)(8-F)	1-3	`OR byte ptr [bp+si+ffff],ff`
80	C(9-F)	1	`OR bl,ff`
81	(0,4,8)(8-F)	2-4	`OR word ptr [bp+si+ffff],ffff`
81	C(9-F)	2	`OR bx,ffff`
83	(0,4,8)(8-F)	1-3	`OR word ptr [bp+si+ffff],±7f`
83	C(9-F)	1	`OR bx,±7f`

Примечание 1: отладчик DEBUG.EXE ошибочно дизассемблирует как команду OR коды "0(A,B) (C-F)(0-F)" и "82 (0,4,8,C)(8-F)".

Примечание 2: когда оба операнда одинаковы, команда OR их не изменяет, но тем не менее нередко используется только для того, чтобы привести в соответствие состояния флагов.

7.03-66 OUT – отсылка данных в порт

Команда OUT выводит байт или слово данных в порт. При ее исполнении процессор выдает сигнал, по которому происходит переключение шин с памяти на устройства ввода-вывода. Номер адресуемого порта задается первым операндом либо косвенно – через регистр DX, либо непосредственно двухзначным шестнадцатеричным числом. Второй операнд команды OUT определяет регистр, содержащий выводимые данные, и вместе с тем их формат: однобайтовый регистр AL, если должен быть выведен байт, или двухбайтовый регистр AX, если должно быть выведено двухбайтовое слово. На состояния флагов команда OUT не влияет.

1-й байт	2-й байт	Байты данных	Примеры
E6		1	`OUT ff,AL`
E7		1	`OUT ff,AX`
EE			`OUT DX,AL`
EF			`OUT DX,AX`

Примечание 1: номера некоторых портов приведены в приложении A.14-1. Адресация портов с номерами свыше FFh выполняется только через регистр DX.

Примечание 2: команда OUT не выполняется из программ, уровень привилегий которых ниже уровня, установленного для операций ввода-вывода в битах 0Ch и 0Dh регистра флагов (A.11-4).

7.03-67 POP – выталкивание данных из стека

Команда POP копирует двухбайтовое слово данных из вершины стека в указанный регистр или в ячейки памяти, а затем смещает положение вершины стека путем увеличения на 2 смещения в регистре SP – указателе стека. На состояния флагов команда POP не влияет.

1-й байт	2-й байт	Байты данных	Примеры	Примечания
07			`POP ES`	
0F	A1		`DB 0F A1`	`= POP FS`
0F	A9		`DB 0F A9`	`= POP GS`
17			`POP SS`	
1F			`POP DS`	
5(8-F)			`POP bx`	
8F	(0,8)(0-7)	0-2	`POP [bp+si+ffff]`	

Примечание 1: операции "выталкивания" данных из стека в регистры FS и GS "неизвестны" отладчику DEBUG.EXE, и потому их приходится вводить посредством инструкции DB (7.01-01). По той же причине отладчик не может правильно дизассемблировать такие команды, однако вместе с тем он не препятствует их отладке при условии работы на компьютере с 32-разрядным процессором.

Примечание 2: отладчик DEBUG.EXE ошибочно дизассемблирует код "8F (C-F)(0-F)" как команду "POP bx".

7.03-68 POPF – восстановление флагов из стека

Команда POPF выталкивает из стека двухбайтовое слово данных в регистр флагов. При этом содержимое указателя стека SP увеличивается на 2.

Код	Пример
9D	POPF

Примечание 1: команда POPF может изменить состояние поля уровня привилегий для операций ввода-вывода – битов 0Ch и 0Dh в регистре флагов (A.11-4), только если программа исполняется на нулевом (высшем) уровне привилегий.

Примечание 2: команда POPF может изменить состояние флага IF только когда она выполняется из программ, уровень привилегий которых не ниже уровня, установленного для операций ввода-вывода в битах 0Ch и 0Dh регистра флагов (A.11-4).

Примечание 3: когда команде POPF предшествует префикс 66h (7.02-06), она "выталкивает" из стека 4 байта в расширенный регистр флагов. Однако при этом установление флага режима V86 аппаратно блокировано (подробнее – в примечаниях 4 и 5 к A.11-4).

7.03-69 PUSH – загрузка слова в стек.

Команда PUSH уменьшает смещение в регистре SP (указателе стека) на 2, увеличивая тем самым длину стека на две ячейки для новых данных. В эти ячейки копируется двухбайтовое слово из того источника, который определен операндом команды PUSH. На состояния флагов команда PUSH не влияет.

1-й байт	2-й байт	Байты данных	Примеры	Примечания
06			`PUSH ES`	
0E			`PUSH CS`	
0F	A0		`DB 0F A0`	`= PUSH FS`
0F	A8		`DB 0F A8`	`= PUSH GS`
16			`PUSH SS`	
1E			`PUSH DS`	
5(0-7)			`PUSH bx`	
68		2	`DB 68 ff ff`	`= PUSH ffff`
6A		1	`DB 6A ff`	`= PUSH 00ff`
FF	(3,7,B)(0-7)	0-2	`PUSH [bp+si+ffff]`	

Примечание 1: команды копирования в стек регистров FS и GS, а также команды "заталкивания" непосредственных операндов отладчику DEBUG.EXE "неизвестны", их приходится вводить посредством инструкции DB (7.01-01). По той же причине отладчик не может их правильно дизассемблировать, однако вместе с тем он не препятствует их отладке при условии работы на компьютере с 32-разрядным процессором.

Примечание 2: рекомендуется избегать применения операции PUSH SP, потому что разные процессоры исполняют ее по-разному: древние процессоры

сначала смещают указатель стека, а потом записывают его значение, а современные процессоры "вталкивают" в стек значение SP, имевшееся на момент выдачи команды. Программы, использующие операцию PUSH SP, на компьютерах с другими процессорами могут вести себя непредсказуемо.

Примечание 3: отладчик DEBUG.EXE ошибочно дизассемблирует код "FF F(0-7)" как команду "PUSH bx".

7.03-70 PUSHF – загрузка флагов в стек

Команда PUSHF (PUSH Flags) загружает стек двухбайтовым словом данных, копируемым из регистра флагов. Смещение в указателе стека SP при исполнении команды PUSHF уменьшается на 2. На состояния флагов команда PUSHF не влияет.

Код	Пример
9C	PUSHF

Примечание 1: когда команде PUSHF предшествует префикс 66h (7.02-06), она копирует в стек четырехбайтовый расширенный регистр флагов (примечание 4 к A.11-4). Однако при этом копирование флага режима V86 аппаратно блокировано.

7.03-71 RCL – сдвиг влево через перенос

Команда RCL (Rotate through Carry to the Left) выполняет кольцевые сдвиги своего первого операнда влево. При каждом сдвиге все биты смещаются на одну позицию в сторону старших разрядов, в младший бит операнда записывается состояние флага переноса, а старший бит операнда "выталкивается" и становится новым состоянием флага переноса CF. Вместе с ним может измениться и состояние флага переполнения OV. Состояния всех других флагов сохраняются.

Второй операнд (1 или CL) задает количество выполняемых сдвигов влево. Если это количество считывается из регистра CL, то принимаются во внимание только 5 младших бит, следовательно, максимальное количество сдвигов равно 31. Число в регистре CL при исполнении команды RCL не изменяется.

1-й байт	2-й байт	Байты данных	Примеры
D0	(1,5,9)(0-7)	0-2	`RCL byte ptr [bp+si+ffff],1`
D0	D(0-7)		`RCL bl,1`
D1	(1,5,9)(0-7)	0-2	`RCL word ptr [bp+si+ffff],1`

Продолжение таблицы 7.03-71

D1	D(0-7)		`RCL bx,1`
D2	(1,5,9)(0-7)	0-2	`RCL byte ptr [bp+si+ffff],CL`
D2	D(0-7)		`RCL bl,CL`
D3	(1,5,9)(0-7)	0-2	`RCL word ptr [bp+si+ffff],CL`
D3	D(0-7)		`RCL bx,CL`

7.03-72 RCR – сдвиг вправо через перенос

Команда RCR (Rotate through Carry to the Right) выполняет кольцевые сдвиги своего первого операнда вправо. При каждом сдвиге все биты смещаются на одну позицию в сторону младших разрядов, в старший бит операнда записывается состояние флага переноса, а младший бит операнда "выталкивается" и становится новым состоянием флага переноса CF. Вместе с ним может измениться и состояние флага переполнения OV. Состояния всех других флагов сохраняются.

Второй операнд (1 или CL) задает количество выполняемых сдвигов вправо. Если это количество считывается из регистра CL, то принимаются во внимание только 5 младших бит, следовательно, максимальное количество сдвигов равно 31. Число в регистре CL при исполнении команды RCR не изменяется.

1-й байт	2-й байт	Байты данных	Примеры
D0	(1,5,9)(8-F)	0-2	`RCR byte ptr [bp+si+ffff],1`
D0	D(8-F)		`RCR bl,1`
D1	(1,5,9)(8-F)	0-2	`RCR word ptr [bp+si+ffff],1`
D1	D(8-F)		`RCR bx,1`
D2	(1,5,9)(8-F)	0-2	`RCR byte ptr [bp+si+ffff],CL`
D2	D(8-F)		`RCR bl,CL`
D3	(1,5,9)(8-F)	0-2	`RCR word ptr [bp+si+ffff],CL`
D3	D(8-F)		`RCR bx,CL`

7.03-73 RET – возврат из подпрограммы

Команда RET (RETurn) используется для возврата в вызывающую программу из подпрограмм, находящихся в том же сегменте кода и запускаемых командой CALL с адресом длиной 2 байта (из подпрограмм, запускаемых командой CALL FAR с 4-байтовым адресом возврат надо осуществлять командой RETF, 7.03-74).

Перед исполнением команды RET в вершине стека должен находиться адрес возврата, то есть смещение следующей команды в вызывающей программе. Операнд после команды RET указывать не нужно, если завершаемая подпрограмма не оставляет после себя в стеке ничего лишнего. Но в момент завершения

подпрограмм, получающих параметры через стек, эти параметры должны быть из стека удалены, и тогда операнд команды RET обозначает число байт, которые нужно "выбросить" из стека. Команда RET "выталкивает" адрес возврата из стека в регистр IP (указатель команд), а потом прибавляет свой операнд к содержимому регистра SP (указателя вершины стека). Тем самым обеспечивается возврат в вызывающую программу в пределах одного сегмента и восстановление исходного положения вершины стека. Состояния флагов команда RET не изменяет.

1-й байт	2-й байт	Байты данных	Примеры
C2 C3		2	`RET ffff` `RET`

Примечание 1: после исполнения возврата командой RET FFFE в верхнем регистре стека сохраняется адрес возврата.

Примечание 2: когда код, ассемблируемый отладчиком DEBUG.EXE, не предполагается исполнять вне отладчика как отдельный файл, тогда команду RET можно использовать для прекращения исполнения этого кода в среде отладчика (пример – в разделе 9.02-03).

7.03-74 RETF – "дальний" возврат

Команда RETF (RETurn Far) делает все то же самое, что делает команда RET (7.03-73), но дополнительно восстанавливает из стека сегментный адрес в регистре CS:. Так происходит возврат в вызывающую программу из других сегментов.

Команду RETF следует использовать для возврата из подпрограмм и драйверов, вызываемых из других сегментов кода командой CALL FAR (7.03-08) с полным четырехбайтовым адресом, которая сохраняет в стеке сегментный адрес вызывающей программы.

1-й байт	2-й байт	Байты данных	Примеры
CA CB		2	`RETF ffff` `RETF`

7.03-75 ROL – кольцевой сдвиг влево

Команда ROL (ROtate to the Left) выполняет кольцевые сдвиги своего первого операнда влево. При каждом сдвиге все биты смещаются на одну позицию в сторону старших разрядов, а в младший разряд записывается состояние "выталкиваемого" самого старшего разряда. Соответственно тому на каждом шаге

могут изменяться состояния флага переноса CF и флага переполнения OV. Состояния всех других флагов сохраняются.

Второй операнд (1 или CL) задает количество выполняемых сдвигов влево. Если это количество считывается из регистра CL, то принимаются во внимание только 5 младших бит, следовательно, максимальное количество сдвигов равно 31. Число в регистре CL при исполнении команды ROL не изменяется.

1-й байт	2-й байт	Байты данных	Примеры
D0	(0,4,8)(0-7)	0-2	`ROL byte ptr [bp+si+ffff],1`
D0	C(0-7)		`ROL bl,1`
D1	(0,4,8)(0-7)	0-2	`ROL word ptr [bp+si+ffff],1`
D1	C(0-7)		`ROL bx,1`
D2	(0,4,8)(0-7)	0-2	`ROL byte ptr [bp+si+ffff],CL`
D2	C(0-7)		`ROL bl,CL`
D3	(0,4,8)(0-7)	0-2	`ROL word ptr [bp+si+ffff],CL`
D3	C(0-7)		`ROL bx,CL`

7.03-76 ROR – кольцевой сдвиг вправо

Команда ROR (ROtate to the Right) выполняет кольцевые сдвиги своего первого операнда вправо. При каждом сдвиге все биты смещаются на одну позицию в сторону младших разрядов, а в старший разряд записывается состояние "выталкиваемого" самого младшего разряда. Соответственно тому на каждом шаге могут изменяться состояния флага переноса CF и флага переполнения OV. Состояния всех других флагов сохраняются.

Второй операнд (1 или CL) задает количество выполняемых сдвигов вправо. Если это количество считывается из регистра CL, то принимаются во внимание только 5 младших бит, следовательно, максимальное количество сдвигов равно 31. Число в регистре CL при исполнении команды ROL не изменяется.

1-й байт	2-й байт	Байты данных	Примеры
D0	(0,4,8)(8-F)	0-2	`ROR byte ptr [bp+si+ffff],1`
D0	C(8-F)		`ROR bl,1`
D1	(0,4,8)(8-F)	0-2	`ROR word ptr [bp+si+ffff],1`
D1	C(8-F)		`ROR bx,1`
D2	(0,4,8)(8-F)	0-2	`ROR byte ptr [bp+si+ffff],CL`
D2	C(8-F)		`ROR bl,CL`
D3	(0,4,8)(8-F)	0-2	`ROR word ptr [bp+si+ffff],CL`
D3	C(8-F)		`ROR bx,CL`

7.03-77 SAHF – восстановление состояния флагов

Команда SAHF (Store AH in Flags) копирует байт из однобайтового регистра AH в младшие 8 бит регистра флагов, так что бит 7 становится состоянием знакового флага SF, бит 6 – состоянием флага нуля ZF, бит 4 – состоянием дополнительного флага переноса AF, бит 2 – состоянием флага четности PF, бит 0 – состоянием флага переноса CF. При этом может измениться состояние флага OF, хотя он и не входит в число флагов, устанавливаемых командой SAHF. Биты 5, 3 и 1 в однобайтовом регистре AH не соответствуют каким-либо флагам, и их состояния не принимаются во внимание.

Код	Пример
9E	SAHF

7.03-78 SAR – знаковый сдвиг вправо

Команда SAR (Shift Arithmetic to the Right) применяется по отношению к числу со знаком и выполняет его поразрядные сдвиги вправо. При каждом сдвиге все биты смещаются на одну позицию в сторону младших разрядов, состояние самого младшего разряда теряется, но состояние старшего знакового разряда сохраняется и распространяется на расположенный правее разряд. Соответственно тому изменяются состояния флагов ZF, PF и CF. Прежнее состояние флагов OF и AF утрачивается.

Второй операнд (1 или CL) задает количество выполняемых сдвигов вправо. Если это количество считывается из регистра CL, то принимаются во внимание только 5 младших бит, следовательно, максимальное количество сдвигов равно 31. Число в регистре CL при исполнении команды SAR не изменяется.

1-й байт	2-й байт	Байты данных	Примеры
D0	(3,7,B)(8-F)	0-2	`SAR byte ptr [bp+si+ffff],1`
D0	F(8-F)		`SAR bl,1`
D1	(3,7,B)(8-F)	0-2	`SAR word ptr [bp+si+ffff],1`
D1	F(8-F)		`SAR bx,1`
D2	(3,7,B)(8-F)	0-2	`SAR byte ptr [bp+si+ffff],CL`
D2	F(8-F)		`SAR bl,CL`
D3	(3,7,B)(8-F)	0-2	`SAR word ptr [bp+si+ffff],CL`
D3	F(8-F)		`SAR bx,CL`

7.03-79 SBB – вычитание с заёмом

Команда SBB (SuBtract with Borrow) вычитает свой второй операнд (вычитаемое) из первого операнда (уменьшаемого), принимая во внимание заем от предшествовавшей операции, выражаемый состоянием флага CF. Затем состояния флагов OF, SF, ZF, AF, PF и CF приводятся в соответствие с получаемой разностью, которая замещает собою первый операнд.

Интерпретация состояний флагов, оставляемых командой SBB, зависит от того, были ли операнды числами со знаком или без знака. После вычитания чисел без знака следует пользоваться командами условных переходов JA, JB, JBE, JNB, а после вычитания чисел со знаком – командами JG, JGE, JL, JLE. Полные имена всех команд условного перехода отражают соотношение между первым операндом (уменьшаемым) и вторым операндом (вычитаемым) команды SBB. Например, JA – переход, если больше – означает, что первый операнд должен быть больше второго.

Команда SBB – двоичная операция, но имеются два исключения. Если первый операнд находится в регистре AX, то команда SBB может быть использована для вычитания неупакованных десятичных слов, после чего полученная двоичная разность должна быть преобразована в неупакованное десятичное слово командой AAS (7.03-04). Если первый операнд находится в регистре AL, то команда SBB может быть применена для вычитания упакованных десятичных байтов; затем получаемую двоичную разность упакованных десятичных байтов надлежит преобразовать в правильный упакованный десятичный байт командой DAS (7.03-19).

1-й байт	2-й байт	Байты данных	Примеры
18	(0-B)(0-F)	0-2	`SBB [bp+si+ffff],bl`
18	(C-F)(0-F)		`SBB bl,bl`
19	(0-B)(0-F)	0-2	`SBB [bp+si+ffff],bx`
19	(C-F)(0-F)		`SBB bx,bx`
1A	(0-B)(0-F)	0-2	`SBB bl,[bp+si+ffff]`
1B	(0-B)(0-F)	0-2	`SBB bx,[bp+si+ffff]`
1C		1	`SBB AL,ff`
1D		2	`SBB AX,ffff`
80	(1,5,9)(8-F)	1-3	`SBB byte ptr [bp+si+ffff],ff`
80	D(9-F)	1	`SBB bl,ff`
81	(1,5,9)(8-F)	2-4	`SBB word ptr [bp+si+ffff],ffff`
81	D(9-F)	2	`SBB bx,ffff`
83	(1,5,9)(8-F)	1-3	`SBB word ptr [bp+si+ffff],±7f`
83	D(9-F)	1	`SBB bx,±7f`

Примечание 1: отладчик DEBUG.EXE ошибочно дизассемблирует как команду SBB коды "1(A,B) (C-F)(0-F)" и "82 (1,5,9,D)(8-F)".

7.03-80 SCASB – поиск байта

Команда SCASB (SCAn a String of Bytes) сравнивает байт в однобайтовом регистре AL с другим байтом, находящимся в памяти по адресу, который заранее загружен в пару регистров ES:DI. Если байты совпадают, флаг нуля ZF устанавливается в состояние ZR, а иначе он сбрасывается в состояние NZ. После сопоставления смещение в регистре DI увеличивается на единицу или уменьшается на единицу: это зависит от состояния ("UP" или "DN") флага направления DF, которое следует заранее задать командой CLD (счет вверх, 7.03-11) или командой STD (счет вниз, 7.03-85). Устанавливаемые состояния флагов OF, SF, AF, PF и CF соответствуют результату вычитания сопоставляемых байтов, хотя сам этот результат нигде не сохраняется и не отображается.

Перед командой SCASB часто ставится префикс повторения REPNZ (7.02-03) или REPZ (7.02-04), что позволяет исполнять команду SCASB несколько раз и таким образом осуществлять поиск заданного байта в строке байтов. Сегментный регистр ES: посредством префикса смены сегмента заменить нельзя.

Код	Пример
AE	SCASB

Примечание 1: при наличии перед командой SCASB префикса повторения порядок исполнения операций включает сначала установление флага по результату сравнения байтов, затем изменение смещения в индексном регистре DI, и только потом проверку условия выхода из цикла по состоянию флага. Поэтому смещение в индексном регистре DI в момент выхода из цикла указывает не на тот байт, который удовлетворил условию выхода, а на следующий за ним байт.

7.03-81 SCASW – поиск слова

Команда SCASW (SCAn a String of Words) сравнивает двухбайтовое слово данных в регистре AX с другим словом, находящимся в памяти, и затем изменяет смещение в регистре DI на 2, подготавливая тем самым адрес для сравнения следующего слова. Под действием префикса 66h смены разрядности операнда (7.02-06) команда SCASW сравнивает содержимое регистра EAX с находящимся в памяти четырехбайтовым словом (типа Dword) и изменяет смещение в регистре DI

на 4. Все остальные особенности команды SCASW такие же, как у команды SCASB (7.03-80).

Код	Пример
AF	SCASW

7.03-82 SHL – сдвиг влево

Команда SHL (SHift to the Left) выполняет сдвиги своего первого операнда влево. При каждом сдвиге все биты смещаются на одну позицию в сторону старших разрядов, самый старший разряд "выталкивается" во флаг переноса CF, а в самый младший разряд записываются нули. Состояния флагов SF, ZF и PF изменяются соответственно результату. Команда SHL не сохраняет прежние состояния флагов OF и AF, оставляя их в неопределенном состоянии.

Второй операнд (1 или CL) задает количество выполняемых сдвигов влево. Если это количество считывается из регистра CL, то принимаются во внимание только 5 младших бит, следовательно, максимальное количество сдвигов равно 31. Число в регистре CL при исполнении команды SHL не изменяется.

1-й байт	2-й байт	Байты данных	Примеры
D0	(2,6,A)(0-7)	0-2	`SHL byte ptr [bp+si+ffff],1`
D0	E(0-7)		`SHL bl,1`
D1	(2,6,A)(0-7)	0-2	`SHL word ptr [bp+si+ffff],1`
D1	E(0-7)		`SHL bx,1`
D2	(2,6,A)(0-7)	0-2	`SHL byte ptr [bp+si+ffff],CL`
D2	E(0-7)		`SHL bl,CL`
D3	(2,6,A)(0-7)	0-2	`SHL word ptr [bp+si+ffff],CL`
D3	E(0-7)		`SHL bx,CL`
C0	E(0-7)	1	примечание 2
C1	E(0-7)	1	примечание 2

Примечание 1: команда SHL полностью эквивалентна команде SAL, но отладчик DEBUG.EXE принимает только название SHL.

Примечание 2: часто бывает удобна команда SHL с непосредственным указанием числа сдвигов, введенная начиная с процессора 80286. Отладчик DEBUG.EXE ее "не знает", но позволяет ее вводить как данные с помощью инструкции DB (7.01-01). Например, коды команд сдвига влево на 4 битовых позиции выглядят так:

```
C0 E0 04        = SHL AL,4
C1 E0 04        = SHL AX,4
```

Последний байт выражает число сдвигов. Для осуществления сдвигов в ином регистре нужно ко второму байту (E0h) прибавить порядковый номер (00h – 07h) желаемого регистра в перечнях, приведенных в 1-й и 2-й строках таблицы 7.00.

7.03-83 SHR – сдвиг вправо

Команда SHR (SHift to the Right) выполняет сдвиги своего первого операнда вправо. При каждом сдвиге все биты смещаются на одну позицию в сторону младших разрядов, самый младший разряд "выталкивается" во флаг переноса CF, а в самый старший разряд записываются нули. Состояния флагов SF, ZF и PF изменяются соответственно результату. Команда SHR не сохраняет прежние состояния флагов OF и AF, оставляя их в неопределенном состоянии.

Второй операнд (1 или CL) задает количество выполняемых сдвигов вправо. Если это количество считывается из регистра CL, то принимаются во внимание только 5 младших бит, следовательно, максимальное количество сдвигов равно 31. Число в регистре CL при исполнении команды SHR не изменяется.

1-й байт	2-й байт	Байты данных	Примеры
D0	(2,6,A)(8-F)	0-2	`SHR byte ptr [bp+si+ffff],1`
D0	E(8-F)		`SHR bl,1`
D1	(2,6,A)(8-F)	0-2	`SHR word ptr [bp+si+ffff],1`
D1	E(8-F)		`SHR bx,1`
D2	(2,6,A)(8-F)	0-2	`SHR byte ptr [bp+si+ffff],CL`
D2	E(8-F)		`SHR bl,CL`
D3	(2,6,A)(8-F)	0-2	`SHR word ptr [bp+si+ffff],CL`
D3	E(8-F)		`SHR bx,CL`
C0	E(8-F)	1	`примечание 1`
C1	E(8-F)	1	`примечание 1`

Примечание 1: часто бывает удобна команда SHR с непосредственным указанием числа сдвигов, введенная начиная с процессора 80286. Отладчик DEBUG.EXE ее "не знает", но позволяет ее вводить как данные с помощью инструкции DB (7.01-01). Например, коды команд сдвига вправо на 4 битовых позиции выглядят так:

C0 E8 04 = SHR AL,4
C1 E8 04 = SHR AX,4

Последний байт выражает число сдвигов. Для осуществления сдвигов в ином регистре нужно ко второму байту (E8h) прибавить порядковый номер (00h – 07h) желаемого регистра в перечнях, приведенных в 1-й и 2-й строках таблицы 7.00.

7.03-84 STC – установление флага переноса

Команда STC (SeT Carry flag) устанавливает флаг переноса CF в состояние CY.

Код	Пример
F9	STC

7.03-85 STD – установление флага направления

Команда STD (SeT Direction flag) устанавливает флаг направления DF в состояние DN, при котором смещения в индексных регистрах DI и/или SI в ходе исполнения строковых операций (CMPSB, LODSB, MOVSB, SCASB, STOSB, и т.д.) будут изменяться в сторону уменьшения.

Код	Пример
FD	STD

7.03-86 STI – установление флага прерывания

Команда STI (SeT Interrupt flag) устанавливает флаг прерывания IF в принимаемое по умолчанию состояние EI (Enable Interrupts), при котором процессор принимает запросы на прерывания через контроллер прерываний.

Код	Пример
FB	STI

Примечание 1: команда STI не выполняется из программ, уровень привилегий которых ниже уровня, установленного для операций ввода-вывода в битах 0Ch и 0Dh регистра флагов (A.11-4).

7.03-87 STOSB – запись байта

Команда STOSB (STOre String of Bytes) копирует байт из однобайтового регистра AL в память по адресу назначения, который заранее подготовлен в регистрах ES:DI. После копирования смещение в регистре DI увеличивается на единицу или уменьшается на единицу: это зависит от состояния ("UP" или "DN") флага направления DF, которое следует заранее задать командой CLD (счет вверх, 7.03-11) или командой STD (счет вниз, 7.03-85). При исполнении команды STOSB состояния всех флагов сохраняются.

Перед командой STOSB часто ставится префикс повторения REPNZ (7.02-03) или REPZ (7.02-04), что позволяет исполнять команду STOSB несколько раз и таким образом заполнять заданным байтом строку байтов. Сегментный регистр ES посредством префикса смены сегмента заменить нельзя.

Код	Пример
AA	STOSB

7.03-88 STOSW – запись слова

Команда STOSW (STOre String of Words) копирует двухбайтовое слово из регистра AX в память и затем изменяет смещение в регистре DI на 2, подготавливая тем самым адрес для следующей операции копирования слова. Под действием префикса 66h смены разрядности операнда (7.02-06) команда STOSW копирует в память четырехбайтовое слово (типа Dword) из регистра EAX и изменяет смещение в регистре DI на 4. Все остальные особенности команды STOSW такие же, как у команды STOSB (7.03-87).

Код	Пример
AB	STOSW

7.03-89 SUB – вычитание без заема

Команда SUB (SUBtract) вычитает второй операнд (вычитаемое) из первого операнда (уменьшаемого), игнорируя состояние флага переноса CF. Затем состояния флагов OF, SF, ZF, AF, PF и CF приводятся в соответствие с получаемой разностью, которая замещает собою первый операнд.

Интерпретация состояний флагов, оставляемых командой SUB, зависит от того, были ли операнды числами со знаком или без знака. После вычитания чисел без знака следует пользоваться командами условных переходов JA, JB, JBE, JNB, а после вычитания чисел со знаком – командами JG, JGE, JL, JLE. Полные имена всех команд условного перехода отражают соотношение между первым операндом (уменьшаемым) и вторым операндом (вычитаемым) команды SUB. Например, JA – переход, если больше – означает, что первый операнд должен быть больше второго.

Команда SUB – двоичная операция, но имеются два исключения. Если первый операнд находится в регистре AX, то команда SUB может быть использована для вычитания неупакованных десятичных слов, после чего полученная двоичная разность должна быть преобразована в неупакованное десятичное слово командой AAS (7.03-04). Если первый операнд находится в регистре AL, то команда SUB

может быть применена для вычитания упакованных десятичных байтов; затем получаемую двоичную разность упакованных десятичных байтов надлежит преобразовать в правильный упакованный десятичный байт командой DAS (7.03-19).

1-й байт	2-й байт	Байты данных	Примеры
28	(0-B)(0-F)	0-2	`SUB [bp+si+ffff],bl`
28	(C-F)(0-F)		`SUB bl,bl`
29	(0-B)(0-F)	0-2	`SUB [bp+si+ffff],bx`
29	(C-F)(0-F)		`SUB bx,bx`
2A	(0-B)(0-F)	0-2	`SUB bl,[bp+si+ffff]`
2B	(0-B)(0-F)	0-2	`SUB bx,[bp+si+ffff]`
2C		1	`SUB AL,ff`
2D		2	`SUB AX,ffff`
80	(2,6,A)(8-F)	1-3	`SUB byte ptr [bp+si+ffff],ff`
80	E(9-F)	1	`SUB bl,ff`
81	(2,6,A)(8-F)	2-4	`SUB word ptr [bp+si+ffff],ffff`
81	E(9-F)	2	`SUB bx,ffff`
83	(2,6,A)(8-F)	1-3	`SUB word ptr [bp+si+ffff],±7f`
83	E(9-F)	1	`SUB bx,±7f`

Примечание 1: отладчик DEBUG.EXE ошибочно дизассемблирует как команду SUB коды "2(A,B) (C-F)(0-F)" и "82 (2,6,A,E)(8-F)".

7.03-90 TEST – побитовая проверка

Команда TEST приводит флаги SF, ZF и PF в соответствие с результатом логической операции "И" (AND) над своими операндами. Однако при исполнении команды TEST результат никуда не записывается, и оба операнда сохраняют свои значения. Флаг переноса CF и флаг переполнения OF всегда сбрасываются командой TEST в состояния NC (No Carry) и NV (No oVerflow) соответственно. Состояние флага AF не сохраняется.

1-й байт	2-й байт	Байты данных	Примеры
84	(0-B)(0-F)	0-2	`TEST [bp+si+ffff],bl`
84	(C-F)(0-F)		`TEST bl,bl`
85	(0-B)(0-F)	0-2	`TEST [bp+si+ffff],bx`
85	(C-F)(0-F)		`TEST bx,bx`
A8		1	`TEST AL,ff`
A9		2	`TEST AX,ffff`

Продолжение таблицы 7.03-90

F6	(0,4,8)(0-7)	1-3	`TEST byte ptr [bp+si+ffff],ff`
F6	C(1-7)	1	`TEST bl,ff`
F7	(0,4,8)(0-7)	2-4	`TEST word ptr [bp+si+ffff],ffff`
F7	C(1-7)	2	`TEST bx,ffff`

7.03-91 XCHG – обмен операндов

Команда XCHG (eXCHanGe) меняет местами содержимое указанных регистров или регистра и ячейки памяти. Состояния флагов при этом не изменяются.

1-й байт	2-й байт	Байты данных	Примеры
86	(0-B)(0-F)	0-2	`XCHG [bp+si+ffff],bl`
86	(C-F)(1-7,9-F)		`XCHG bl,bl`
87	(0-B)(0-F)	0-2	`XCHG [bp+si+ffff],bx`
87	(C-F)(1-7,9-F)		`XCHG bx,bx`
9(1-7)			`XCHG bx,AX`

7.03-92 XLAT – табличный перевод

Команда XLAT (transLATe) вычисляет сумму (AL + BX) и затем копирует байт из ячейки памяти по адресу DS:(AL + BX) в однобайтовый регистр AL, замещая его прежнее содержимое. Состояния флагов и содержимое регистра BX сохраняются без изменений.

Команда XLAT используется для перевода кодов по кодовой таблице длиной до 256 байт, которая должна быть заранее загружена начиная с адреса DS:BX. Принимаемый по умолчанию сегментный регистр DS: может быть заменен другим посредством указания префикса смены сегмента (7.02-01).

Код	Пример
D7	XLAT

7.03-93 XOR – исключающее ИЛИ

Команда XOR (eXclusive OR) поразрядно сопоставляет биты операндов, сбрасывая бит результата в нуль, если биты в соответствующих разрядах сопоставляемых операндов одинаковы, и устанавливая бит результата в единицу, если биты в соответствующих разрядах сопоставляемых операндов разные. Результат замещает собою первый операнд. Состояния флагов SF, ZF, PF приводятся в соответствие с полученным результатом. Флаги CF и OF

сбрасываются в состояния NC (No Carry) и NV (No oVerflow) соответственно. Состояние флага AF не сохраняется.

1-й байт	2-й байт	Байты данных	Примеры
30	(0-B)(0-F)	0-2	`XOR [bp+si+ffff],bl`
30	(C-F)(0-F)		`XOR bl,bl`
31	(0-B)(0-F)	0-2	`XOR [bp+si+ffff],bx`
31	(C-F)(0-F)		`XOR bx,bx`
32	(0-B)(0-F)	0-2	`XOR bl,[bp+si+ffff]`
33	(0-B)(0-F)	0-2	`XOR bx,[bp+si+ffff]`
34		1	`XOR AL,ff`
35		2	`XOR AX,ffff`
80	(3,7,B)(0-7)	1-3	`XOR byte ptr [bp+si+ffff],ff`
80	F(1-7)	1	`XOR bl,ff`
81	(3,7,B)(0-7)	2-4	`XOR word ptr [bp+si+ffff],ffff`
81	F(1-7)	2	`XOR bx,ffff`
83	(3,7,B)(0-7)	1-3	`XOR word ptr [bp+si+ffff],±7f`
83	F(1-7)	1	`XOR bx,±7f`

Примечание 1: команда XOR с указанием одного и того же регистра в качестве обоих операндов часто используется для обнуления регистров.

Примечание 2: отладчик DEBUG.EXE ошибочно дизассемблирует как команду XOR коды "3(2,3) (C-F)(0-F)" и "82 (3,7,B,F)(0-7)".

7.04 Команды, пересылаемые арифметическому сопроцессору

Ассемблерные команды, название которых начинается с буквы "F" (= Float), – это команды, которые центральный процессор пересылает для исполнения арифметическому сопроцессору. Все современные компьютеры исполняют такие команды, потому что имеют встроенный арифметический сопроцессор в составе центрального процессора.

В компьютерах со старыми процессорами, включая некоторые варианты 80486-x, арифметический сопроцессор нередко отсутствует. В таких компьютерах исполнение команд сопроцессора возможно посредством эмуляции, но для этого надо выполнить два условия:

– во-первых, обеспечить генерацию прерывания INT 07 (8.01-08) в ответ на каждую сопроцессорную команду посредством установки в единицу бита 02h в управляющем регистре CR0 (A.11-4).

– во-вторых, установить такой обработчик прерывания INT 07, который способен осуществить эмуляцию команд сопроцессора.

В некоторых старых компьютерах удовлетворение обоих перечисленных условий автоматически обеспечивается системой BIOS, в других об этом предоставлено побеспокоиться пользователю. В любом случае наличие сопроцессора в компьютере можно определить с помощью прерывания INT 11 (8.01-36, A.11-1).

Если не исключена вероятность исполнения Вашей программы на старых процессорах с отдельной микросхемой сопроцессора, то перед каждой командой сопроцессора надо ставить префикс WAIT (7.02-05), обеспечивающий синхронизацию передачи команд от процессора к сопроцессору. В современных процессорах со встроенным сопроцессором синхронизация осуществляется аппаратными средствами, им префикс WAIT не нужен, его наличие допускается, но обычно игнорируется.

7.04-01 F2XM1 – аппроксимация степенной функции

Команда F2XM1 возвращает сумму ряда, используемого для аппроксимации степенной функции числа 2 в интервале значений показателя степени от –1 до +1. Показатель степени должен быть в ST(0) – верхнем регистре стека сопроцессора, туда же после завершения операции помещается вычисленная сумма ряда. Конечный результат может быть представлен формулой

ST(0) = –1+2^ST(0).

Код	Пример
D9F0	F2XM1

7.04-02 FABS – абсолютное значение

Команда FABS сбрасывает в нуль знаковый разряд числа в верхнем регистре ST(0) стека сопроцессора, делая тем самым это число положительным.

Код	Пример
D9E1	FABS

7.04-03 FADD – сложение действительных чисел

Команда FADD суммирует число, считываемое из памяти или регистра сопроцессора, если он указан в качестве второго операнда, с другим числом, находящимся в верхнем регистре ST(0) стека сопроцессора или в другом регистре стека, если он указан первым из двух операндов. Получаемая сумма замещает

собою прежнее значение в регистре ST(0) или в другом регистре стека сопроцессора, если такой регистр ST(1-7) указан первым из двух операндов.

1-й байт	2-й байт	Байты данных	Примеры
D8	(0,4,8)(0-7)	0-2	`FADD dword ptr [bp+si+ffff]`
D8	C(0-7)		`FADD ST,ST(0-7)`
DC	(0,4,8)(0-7)	0-2	`FADD qword ptr [bp+si+ffff]`
DC	C(0-7)		`FADD ST(1-7),ST`

7.04-04 FADDP – сложение и выталкивание

Команда FADDP (ADD and Pop) суммирует число, находящееся в верхнем регистре ST(0) стека сопроцессора, с числом, находящимся в одном из других регистров ST(1-7) того же стека. Туда же помещается получаемая сумма, а затем указатель вершины стека увеличивается на единицу, так что доступ к прежнему числу в регистре ST(0) утрачивается, а номера регистров ST(1-7), включая тот, куда помещена полученная сумма, становятся на единицу меньше.

Код	Пример
DE C(0-7)	`FADDP ST(1-7),ST`

7.04-05 FBLD – загрузка с преобразованием

Команда FBLD (Binary LoaD) считывает из памяти, начиная с указанного адреса, целое 10-байтовое упакованное десятичное число, содержащее по две десятичные цифры на байт, преобразует его в формат двоичного действительного числа, и в таком виде загружает в регистр ST(7), который должен быть пуст, а затем уменьшает указатель вершины стека на единицу, вследствие чего регистры ST(0-6) переименовываются в ST(1-7), а регистр ST(7) – в ST(0), так что загруженное число оказывается в верхнем регистре ST(0).

1-й байт	2-й байт	Байты данных	Пример
DF	(2,6,A)(0-7)	0-2	`FBLD tbyte ptr [bp+si+ffff]`

Примечание 1: команда FBLD не проверяет, содержит ли считываемое из памяти число только десятичные цифры или нет. Если нет, то результат преобразования будет ошибочным.

Примечание 2: 10-байтовый формат двоичного действительного числа, с которым по умолчанию работает арифметический сопроцессор, включает

мантиссу (биты 0 – 63), показатель степени (биты 64 – 78) и знаковый бит 79. Путем изменения содержания поля PC (7.04-35, примечание 2) возможно переключение в 8-байтовый формат двойной точности (52 бита – мантисса, 11 бит – степень) и в 4-байтовый формат одинарной точности (23 байта мантисса, 8 бит – степень).

7.04-06 FBSTP – выгрузка с преобразованием

Команда FBSTP (Binary STore and Pop) преобразует двоичное действительное число в верхнем регистре ST(0) стека сопроцессора в целое 10-байтовое упакованное десятичное число, содержащее по две десятичных цифры на байт. Преобразование включает округление дробной части числа. В таком виде команда FBSTP копирует число в память, начиная с указанного адреса, а затем увеличивает указатель вершины стека на единицу, так что доступ к прежнему числу в регистре ST(0) утрачивается, а регистры ST(1-7) переименовываются в ST(0-6).

1-й байт	2-й байт	Байты данных	Пример
DF	(3,7,B)(0-7)	0-2	FBSTP tbyte ptr [bp+si+ffff]

7.04-07 FCHS – изменить знак

Команда FCHS (CHange Sign) изменяет на противоположный знак числа в верхнем регистре ST(0) стека арифметического сопроцессора.

Код	Пример
D9 E0	FCHS

7.04-08 FCLEX – сброс флагов ошибок

Команда FCLEX (CLear EXeptions) сбрасывает в служебном регистре SWR (Status Word Register) арифметического сопроцессора следующие флаги:

бит 0 – флаг недействительной операции
бит 1 – флаг денормализованного операнда
бит 2 – флаг деления на нуль
бит 3 – флаг переполнения
бит 4 – флаг антипереполнения (потери результата)
бит 5 – флаг потери точности
бит 7 – флаг запроса прерывания
бит 15 – флаг занятости сопроцессора

Код	Пример
DB E2	FCLEX

7.04-09 FCOM – сравнение чисел

Команда FCOM (COMpare) сравнивает действительное число в верхнем регистре ST(0) стека сопроцессора с содержимым указанного регистра или ячейки памяти. Флаги C0, C2 и C3 регистра SWR приводятся в соответствие с результатом сравнения. Сначала надо проверять C2: если C2 = 1, то операнды несопоставимы, и дальше проверять нечего. C3 = 1 отмечает равенство, а C0 = 1 означает, что число в регистре ST(0) меньше, чем другой операнд. О том, как проверять состояния флагов C0 – C3 регистра SWR, написано в разделе 7.04-64.

1-й байт	2-й байт	Байты данных	Примеры
D8	(1,5,9)(0-7)	0-2	`FCOM dword ptr [bp+si+ffff]`
D8	D(0-7)		`FCOM ST(0-7)`
DC	(1,5,9)(0-7)	0-2	`FCOM qword ptr [bp+si+ffff]`

Примечание 1: отладчик DEBUG.EXE ошибочно дизассемблирует код "DC D(0-7)" как команду FCOM ST(0-7).

7.04-10 FCOMP – сравнение и выталкивание

Команда FCOMP (COMpare and Pop) выполняет сравнение чисел точно так же, как это делает команда FCOM (7.04-09), но потом увеличивает указатель вершины стека на единицу, так что доступ к прежнему содержимому верхнего регистра ST(0) утрачивается, а регистры стека ST(1-7) переименовываются в ST(0-6).

1-й байт	2-й байт	Байты данных	Примеры
D8	(1,5,9)(8-F)	0-2	`FCOMP dword ptr [bp+si+ffff]`
D8	D(8-F)		`FCOMP ST(0-7)`
DC	(1,5,9)(8-F)	0-2	`FCOMP qword ptr [bp+si+ffff]`

Примечание 1: отладчик DEBUG.EXE ошибочно дизассемблирует как команду FCOMP коды "DC D(8-F)" и "DE D(0-7)".

7.04-11 FCOMPP – сравнение и 2 выталкивания

Команда FCOMPP (COMpare and Pop and Pop) сравнивает операнды в регистрах стека сопроцессора ST(0) и ST(1), а затем увеличивает указатель вершины стека на 2, так что оба операнда оказываются утрачены, а регистры стека ST(2-7) переименовываются в ST(0-5).

Результат сравнения выражается состояниями флагов C0, C2 и C3 служебного регистра SWR точно так же, как после исполнения команды FCOM (7.04-09).

Код	Пример
DE D9	FCOMPP

Примечание 1: отладчик DEBUG.EXE дизассемблирует код "DE D9" как команду "FCOMPP ST(1)", но при ассемблировании не принимает "ST(1)".

7.04-12 FDECSTP – уменьшить указатель вершины стека

Уменьшение указателя вершины стека на единицу командой FDECSTP (DECrement Stack Top Pointer) изменяет нумерацию регистров стека сопроцессора: регистр ST(7) становится регистром ST(0), а номера регистров ST(0-6) увеличиваются на единицу и становятся ST(1-7). Содержимое регистров стека сохраняется.

Код	Пример
D9 F6	FDECSTP

Примечание 1: указатель стека представляет собой трехразрядный реверсивный счетчик на базе битов 11, 12 и 13 служебного регистра SWR.

Примечание 2: команды записи в регистр ST(0) с "проталкиванием" стека вниз исполняются путем записи в регистр ST(7) и переименования его в ST(0) уменьшением указателя вершины стека. Такие команды не исполняются, если регистр ST(7) не пуст.

7.04-13 FDISI – запретить прерывания

Команда FDISI (DISable Interrupts) – команда запрета прерываний для старого арифметического сопроцессора 8087. Начиная с модели 80287 сопроцессоры не нуждаются в этой команде и игнорируют ее.

Код	Пример
DB E1	FDISI

7.04-14 FDIV – деление

Команда FDIV (DIVide) делит действительное число в верхнем регистре ST(0) стека сопроцессора или в другом регистре ST(1-7) стека, если он указан в качестве первого операнда, на действительное число в ячейке памяти или в регистре стека сопроцессора, если он указан в качестве второго операнда. Результат замещает делимое в регистре ST(0) или в каком-либо из регистров ST(1-7), если он указан в качестве первого операнда.

1-й байт	2-й байт	Байты данных	Пример
D8	(3,7,B)(0-7)	0-2	`FDIV dword ptr [bp+si+ffff]`
D8	F(0-7)		`FDIV ST,ST(0-7)`
DC	(3,7,B)(0-7)	0-2	`FDIV qword ptr [bp+si+ffff]`
DC	F(8-F)		`FDIV ST(1-7),ST`

7.04-15 FDIVP – деление и выталкивание.

Команда FDIVP (DIVide and Pop) делит число, находящееся в не-верхнем регистре ST(1-7) стека сопроцессора, на делитель в верхнем регистре стека ST(0), замещает делимое результатом в регистре ST(1-7), а затем увеличивает указатель вершины стека на единицу, так что доступ к делителю в верхнем регистре ST(0) утрачивается, а номера регистров ST(1-7), включая тот, в который помещено полученное частное, уменьшаются на единицу.

Код	Пример
DE F(8-F)	`FDIVP ST(1-7),ST`

7.04-16 FDIVR – деление в обратном порядке

Команда FDIVR (DIVide in Reverse order) делит действительное число, считываемое из ячейки памяти или из регистра стека сопроцессора, если он указан в качестве второго операнда, на делитель, находящийся в верхнем регистре ST(0) стека сопроцессора или в другом регистре стека ST(1-7), если он указан в качестве первого операнда. Получаемое частное замещает собою делитель в регистре ST(0) или в другом регистре стека ST(1-7), если он указан в качестве первого операнда.

1-й байт	2-й байт	Байты данных	Примеры
D8	(3,7,B)(8-F)	0-2	`FDIVR dword ptr [bp+si+ffff]`
D8	F(8-F)		`FDIVR ST,ST(0-7)`

Продолжение таблицы 7.04-16

DC DC	(3,7,B)(8-F) F(0-7)	0-2	`FDIVR qword ptr [bp+si+ffff]` `FDIVR ST(1-7),ST`

7.04-17 FDIVRP – обратное деление и выталкивание

Команда FDIVRP (DIVide in Reverse order and Pop) делит действительное число, находящееся в верхнем регистре ST(0) стека сопроцессора, на делитель, находящийся в другом регистре ST(1-7) стека, замещает делитель в регистре ST(1-7) полученным результатом, а затем увеличивает указатель вершины стека на единицу, так что доступ к прежнему числу в верхнем регистре ST(0) утрачивается, а номера регистров ST(1-7), включая тот, в который помещено полученное частное, уменьшаются на единицу. Например, результат операции FDIVRP ST(6),ST оказывается в регистре ST(5).

Код	Пример
DE F(0-7)	`FDIVRP ST(1-7),ST`

7.04-18 FENI – разрешить прерывания

Команда FENI (ENable Interrupts) – команда разрешения прерываний для старого арифметического сопроцессора 8087. Начиная с модели 80287 сопроцессоры не нуждаются в этой команде и игнорируют ее.

Код	Пример
DB E0	FENI

7.04-19 FFREE – освободить регистр

Команда FFREE помечает указанный регистр стека сопроцессора как пустой посредством занесения двоичной метки "11" в тег, соответствующий данному регистру, в регистре тегов TWR.

Код	Пример
DD C(0-7)	`FFREE ST(0-7)`

Примечание1: отладчик DEBUG.EXE ошибочно дизассемблирует как команду FFREE код "DF C(0-7)".

7.04-20 FIADD – сложение с целым числом

Команда FIADD (Integer ADDition) выполняет сложение целого числа, считываемого из ячейки памяти, с действительным числом, находящимся в верхнем регистре ST(0) стека сопроцессора. Получаемая сумма замещает второе слагаемое в регистре ST(0).

1-й байт	2-й байт	Байты данных	Примеры
DA DE	(0,4,8)(0-7) (0,4,8)(0-7)	0-2 0-2	`FIADD dword ptr [bp+si+ffff]` `FIADD word ptr [bp+si+ffff]`

7.04-21 FICOM – сравнение с целым числом

Команда FICOM (Integer COMpare) считывает целое число из ячейки памяти, преобразует его в действительное число и сопоставляет с действительным числом, находящимся в верхнем регистре ST(0) стека сопроцессора. Собственно сопоставление выполняется так же, как это делает команда FCOM (7.04-09).

1-й байт	2-й байт	Байты данных	Примеры
DA DE	(1,5,9)(0-7) (1,5,9)(0-7)	0-2 0-2	`FICOM dword ptr [bp+si+ffff]` `FICOM word ptr [bp+si+ffff]`

7.04-22 FICOMP – сравнение с целым и выталкивание

Команда FICOMP (Integer COMpare and Pop) считывает целое число из ячейки памяти, преобразует его в действительное число и сопоставляет с действительным числом, находящимся в верхнем регистре ST(0) стека сопроцессора. Собственно сопоставление выполняется так же, как это делает команда FCOM (7.04-09), но затем указатель вершины стека увеличивается на единицу, так что доступ к прежнему операнду в регистре ST(0) оказывается потерян, а регистры ST(1-7) переименовываются в ST(0-6).

1-й байт	2-й байт	Байты данных	Примеры
DA DE	(1,5,9)(8-F) (1,5,9)(8-F)	0-2 0-2	`FICOMP dword ptr [bp+si+ffff]` `FICOMP word ptr [bp+si+ffff]`

7.04-23 FIDIV – деление на целое число

Команда FIDIV (Integer DIVide) делит действительное число в верхнем регистре ST(0) стека сопроцессора на целое число, считываемое из указанной ячейки памяти. Результат замещает делимое в регистре ST(0).

1-й байт	2-й байт	Байты данных	Примеры
DA DE	(3,7,B)(0-7) (3,7,B)(0-7)	0-2 0-2	`FIDIV dword ptr [bp+si+ffff]` `FIDIV word ptr [bp+si+ffff]`

7.04-24 FIDIVR – обратное деление на целое

Команда FIDIVR (Integer DIVide in Reverse order) выполняет деление числа, считываемого из ячейки памяти, на делитель, находящийся в верхнем регистре ST(0) стека сопроцессора. Получаемое частное замещает делитель в регистре ST(0).

1-й байт	2-й байт	Байты данных	Примеры
DA DE	(3,7,B)(8-F) (3,7,B)(8-F)	0-2 0-2	`FIDIVR dword ptr [bp+si+ffff]` `FIDIVR word ptr [bp+si+ffff]`

7.04-25 FILD – загрузка целого числа

Команда FILD (Integer LoaD) загружает содержимое указанных ячеек памяти в регистр ST(7) стека сопроцессора, который должен быть пуст, одновременно преобразуя это содержимое в формат действительного числа, а затем уменьшает указатель вершины стека на единицу, вследствие чего регистры ST(0-6) переименовываются в ST(1-7), а регистр ST(7) – в ST(0), так что загруженное число оказывается в верхнем регистре ST(0).

1-й байт	2-й байт	Байты данных	Примеры

DB
DF
DF
	(0,4,8)(0-7)
	(0,4,8)(0-7)
	(2,6,A)(8-F)
		0-2
		0-2
		0-2
			`FILD dword ptr [bp+si+ffff]`
			`FILD word ptr [bp+si+ffff]`
			`FILD qword ptr [bp+si+ffff]`

7.04-26 FIMUL – умножение на целое число

Команда FIMUL (Integer MULtiply) умножает действительное число в верхнем регистре ST(0) стека сопроцессора на целое число, считываемое из указанных ячеек памяти. Получаемое произведение замещает первый сомножитель в регистре ST(0).

1-й байт	2-й байт	Байты данных	Примеры
DA DE	(0,4,8)(8-F) (0,4,8)(8-F)	0-2 0-2	`FIMUL dword ptr [bp+si+ffff]` `FIMUL word ptr [bp+si+ffff]`

7.04-27 FINCSTP – увеличить указатель вершины стека

Увеличение указателя вершины стека на единицу командой FINCSTP (INCrement Stack Top Pointer) изменяет нумерацию регистров стека сопроцессора: регистр ST(0) становится регистром ST(7), а номера регистров ST(1-7) уменьшаются на единицу и становятся ST(0-6). Содержимое регистров стека сохраняется.

Код	Пример
D9 F7	FINCSTP

Примечание 1: указатель стека представляет собой трехразрядный реверсивный счетчик на базе битов 11, 12 и 13 служебного регистра SWR.

Примечание 2: при исполнении операции увеличения указателя вершины стека в составе других команд переименование регистра ST(0) в ST(7) сопровождается приданием ему статуса пустого регистра (тега 11). Потому доступ к прежнему содержанию регистра ST(0) утрачивается. Подобные операции часто интерпретируют как "выталкивание" из стека прежнего содержимого регистра ST(0).

7.04-28 FINIT – установ начального состояния

Команда FINIT (INITialize) устанавливает служебные регистры арифметического сопроцессора (CWR, SWR, TWR, IPR, DPR) в начальное состояние. В регистр CWR записывается состояние 037Fh, означающее 80-битовый формат вычислений, маскирование всех исключений и выполнение округления до ближайшего целого числа. В регистр тегов TWR записывается состояние FFFFh, означающее, что все регистры стека сопроцессора пусты. В остальные регистры (SWR, IPR, DPR) записывается исходное состояние 0000h.

Код	Пример
DB E3	FINIT

7.04-29 FIST – запись целого числа

Команда FIST (Integer STore) считывает действительное число из верхнего регистра ST(0) стека сопроцессора, преобразует его в целое число, округляя согласно установленному в регистре CWR формату, и записывает в ячейки памяти по указанному адресу.

1-й байт	2-й байт	Байты данных	Примеры
DB DF	(1,5,9)(0-7) (1,5,9)(0-7)	0-2 0-2	`FIST dword ptr [bp+si+ffff]` `FIST word ptr [bp+si+ffff]`

7.04-30 FISTP – запись целого числа и выталкивание

Команда FISTP (Integer STore and Pop) делает то же самое, что команда FIST (7.04-29), но, кроме того, увеличивает указатель вершины стека на единицу, так что доступ к прежнему содержимому верхнего регистра стека ST(0) утрачивается, а регистры ST(1-7) переименовываются в ST(0-6).

1-й байт	2-й байт	Байты данных	Примеры

DB
DF
DF
	(1,5,9)(8-F)
	(1,5,9)(8-F)
	(3,7,B)(8-F)
		0-2
		
		0-2
			`FISTP dword ptr [bp+si+ffff]`
			`FISTP word ptr [bp+si+ffff]`
			`FISTP qword ptr [bp+si+ffff]`

7.04-31 FISUB – вычитание целого числа

Команда FISUB (Integer SUBtract) вычитает целое число, считываемое из ячейки памяти, из действительного числа в верхнем регистре ST(0) стека сопроцессора. Получаемая разность замещает уменьшаемое в регистре ST(0).

1-й байт	2-й байт	Байты данных	Примеры
DA DE	(2,6,A)(0-7) (2,6,A)(0-7)	0-2 0-2	`FISUB dword ptr [bp+si+ffff]` `FISUB word ptr [bp+si+ffff]`

7.04-32 FISUBR – обратное вычитание целого

Команда FISUBR (Integer SUBtract in Reverse order) выполняет вычитание действительного числа, находящегося в верхнем регистре ST(0) стека сопроцессора, из целого числа – уменьшаемого, считываемого из ячейки памяти. Получаемая разность замещает вычитаемое в регистре ST(0).

1-й байт	2-й байт	Байты данных	Примеры
DA DE	(2,6,A)(8-F) (2,6,A)(8-F)	0-2 0-2	`FISUBR dword ptr [bp+si+ffff]` `FISUBR word ptr [bp+si+ffff]`

7.04-33 FLD – загрузка действительного числа

Команда FLD (LoaDing) загружает действительное число, считываемое из ячейки памяти или из регистра стека сопроцессора, в регистр ST(7), который должен быть пуст, а затем уменьшает на единицу указатель вершины стека, вследствие чего регистры ST(0-6) переименовываются в ST(1-7), а регистр ST(7) – в ST(0), так что загруженное число оказывается в верхнем регистре ST(0).

1-й байт	2-й байт	Байты данных	Примеры

D9
D9
DB
DD
	(0,4,8)(0-7)
	C(0-7)
	(2,6,A)(8-F)
	(0,4,8)(0-7)
		0-2
		
		0-2
		0-2
			`FLD dword ptr [bp+si+ffff]`
			`FLD ST(0-7)`
			`FLD tbyte ptr [bp+si+ffff]`
			`FLD qword ptr [bp+si+ffff]`

7.04-34 FLD1 – загрузка единичной константы

Команда FLD1 (LoaD 1) загружает константу 1.0 в регистр стека ST(7), который должен быть пуст, а затем уменьшает на единицу указатель вершины стека, вследствие чего регистры ST(0-6) переименовываются в ST(1-7), а регистр ST(7) – в ST(0), так что загруженная константа оказывается в верхнем регистре ST(0).

Код	Пример
D9 E8	FLD1

7.04-35 FLDCW – загрузка регистра CWR

Команда FLDCW (LoaD Control Word) копирует слово данных, сохраненное командой FSTCW (7.04-55), из указанного адреса памяти в служебный регистр

CWR (Control Word Register). Если слово было изменено, то после копирования командой FLDCW внесенные изменения вступят в силу.

1-й байт	2-й байт	Байты данных	Пример
D9	(2,6,A)(8-F)	0-2	`FLDCW word ptr [bp+si+ffff]`

Примечание 1: биты 0 – 5 регистра CWR играют роль масок для флагов ошибок, представленных соответствующими битами 0 – 5 регистра SWR (7.04-08). Начальное состояние масок – установленное, но если маску сбросить, то возникновение ошибки повлечет запрос IRQ 13 на вызов обработчика прерывания INT 75 (8.03-75), который должен быть способен справиться с возникшей проблемой.

Примечание 2: биты 9 и 8 регистра CWR – это поле задания точности (Precision Control Field). Начальное состояние 11 означает 10-байтовый формат чисел, состояние 10 – округление до 8-байтового формата, состояние 00 – округление до 4-байтового формата (7.04-05). Однако понижение точности не ускоряет исполнение операций.

7.04-36 FLDENV – загрузка служебных регистров

Команда FLDENV (LoaD ENVironment) восстанавливает состояния всех служебных регистров арифметического сопроцессора (CWR, SWR, TWR, IPR, DPR) по записи, которую считывает из памяти, начиная с указанного адреса. Эта запись должна быть заранее сформирована командой FSTENV (7.04-56).

1-й байт	2-й байт	Байты данных	Пример
D9	(2,6,A)(0-7)	0-2	`FLDENV word ptr [bp+si+ffff]`

7.04-37 FLDL2E – загрузка константы log e

Команда FLDL2E (LoaD Log2E) загружает константу log e = 1.44269..., то есть логарифм числа e = 2.71828... по основанию 2, в регистр стека ST(7), который должен быть пуст, а затем уменьшает на единицу указатель вершины стека, вследствие чего регистры ST(0-6) переименовываются в ST(1-7), а регистр ST(7) – в ST(0), так что загруженная константа оказывается в верхнем регистре ST(0).

Код	Пример
D9 EA	FLDL2E

7.04-38 FLDL2T – загрузка константы log10

Команда FLDL2T (LoaD Log2 of Ten) загружает константу log10 = 3.32192..., то есть логарифм числа 10 по основанию 2, в регистр стека ST(7), который должен быть пуст, а затем уменьшает на единицу указатель вершины стека, вследствие чего регистры ST(0-6) переименовываются в ST(1-7), а регистр ST(7) – в ST(0), так что загруженная константа оказывается в верхнем регистре ST(0).

Код	Пример
D9 E9	FLDL2T

7.04-39 FLDLG2 – загрузка константы lg2

Команда FLDLG2 (LoaD Lg2) загружает константу lg2 = 0.301029..., то есть логарифм числа 2 по основанию 10, в регистр стека ST(7), который должен быть пуст, а затем уменьшает на единицу указатель вершины стека, вследствие чего регистры ST(0-6) переименовываются в ST(1-7), а регистр ST(7) – в ST(0), так что загруженная константа оказывается в верхнем регистре ST(0).

Код	Пример
D9 EC	FLDLG2

7.04-40 FLDLN2 – загрузка константы ln2

Команда FLDLN2 (LoaD LN2) загружает константу ln2 = 0.693147..., то есть логарифм числа 2 по основанию e = 2.71828..., в регистр стека ST(7), который должен быть пуст, а затем уменьшает на единицу указатель вершины стека, вследствие чего регистры ST(0-6) переименовываются в ST(1-7), а регистр ST(7) – в ST(0), так что загруженная константа оказывается в верхнем регистре ST(0).

Код	Пример
D9 ED	FLDLN2

7.04-41 FLDPI – загрузка константы "PI"

Команда FLDPI (LoaD PI) загружает константу PI = 3.14159… в регистр стека ST(7), который должен быть пуст, а затем уменьшает на единицу указатель вершины стека, вследствие чего регистры ST(0-6) переименовываются в ST(1-7), а регистр ST(7) – в ST(0), так что загруженная константа оказывается в верхнем регистре ST(0).

Код	Пример
D9 EB	FLDPI

7.04-42 FLDZ – загрузка константы нуль

Команда FLDZ (LoaD Zero) загружает константу 0 в регистр стека ST(7), который должен быть пуст, а затем уменьшает на единицу указатель вершины стека, вследствие чего регистры ST(0-6) переименовываются в ST(1-7), а регистр ST(7) – в ST(0), так что загруженная константа оказывается в верхнем регистре ST(0).

Код	Пример
D9 EE	FLDZ

7.04-43 FMUL – умножение

Команда FMUL (MULtiply) умножает действительное число в верхнем регистре ST(0) стека сопроцессора или в не-верхнем регистре стека, если он указан в качестве первого операнда, на другое действительное число – множитель, считываемое из ячейки памяти или из регистра стека ST(0-7), если он указан в качестве второго операнда. Получаемое произведение замещает собою прежнее содержимое регистра ST(0) или другого регистра ST(1-7), если он указан в качестве первого операнда.

1-й байт	2-й байт	Байты данных	Примеры
D8	(0,4,8)(8-F)	0-2	`FMUL dword ptr [bp+si+ffff]`
D8	C(8-F)		`FMUL ST,ST(0-7)`
DC	(0,4,8)(8-F)	0-2	`FMUL qword ptr [bp+si+ffff]`
DC	C(8-F)		`FMUL ST(1-7),ST`

7.04-44 FMULP – умножение и выталкивание

Команда FMULP (MULtiply and Pop) умножает действительное число в не-верхнем регистре стека сопроцессора на другое действительное число, находящееся в верхнем регистре ST(0), записывает результат в регистр ST(1-7) вместо первого сомножителя, а затем увеличивает указатель вершины стека на единицу, так что доступ к прежнему числу в регистре ST(0) утрачивается, а номера

всех других регистров стека, включая тот, в который записано полученное произведение, уменьшаются на единицу.

Код	Пример
DE C(8-F)	FMULP ST(1-7),ST

7.04-45 FNOP – "пустая" операция

Команда FNOP (No Operation) никаких действий не выполняет, лишь увеличивает число в регистре IP – указателе команд – на 2 (потому что код самой команды FNOP занимает 2 байта).

Код	Пример
D9 D0	FNOP

7.04-46 FPATAN – частичный арктангенс

Команда FPATAN (Partial ArcTANgent) делит положительное действительное число в верхнем регистре стека ST(0) на не меньшее положительное действительное число в регистре стека ST(1), вычисляет функцию Arctg(ST(0)/ST(1)) в радианах, записывает результат в регистр ST(1), а затем увеличивает указатель вершины стека на единицу, так что регистр ST(1), куда был записан результат, становится регистром ST(0), а доступ к прежнему содержимому регистра ST(0) утрачивается.

Код	Пример
D9 F3	FPATAN

7.04-47 FPREM – вычисление частичного остатка

Основное назначение команда FPREM (Partial REMainder) – приведение аргумента периодических тригонометрических функций в рамки начального интервала. Команда FPREM выполняет деление числа в верхнем регистре стека сопроцессора ST(0) на другое число – делитель – в регистре ST(1). Получаемый остаток замещает делимое в регистре ST(0). Если этот остаток оказывается больше, чем делитель в регистре ST(1), то результат считается частичным остатком и отмечается установлением бита C2=1 в регистре SWR. В таком случае команду FPREM следует исполнять повторно до тех пор, пока сброс в нуль бита C2 в регистре SWR не засвидетельствует получение конечного остатка.

Когда конечный остаток получен, состояния битов C3, C1 и C0 в регистре SWR указывают на тот из 8 секторов круга, в котором находится конечное значение аргумента периодической тригонометрической функции. О том, как проверять состояния битов C3 – C0, написано в 7.04-64.

Код	Пример
D9 F8	FPREM

7.04-48 FPTAN – частичный тангенс

Команда FPTAN (Partial TANgent) принимает в верхнем регистре стека ST(0) значение углового аргумента в радианах для вычисления тангенса Tg(ST(0)). Сначала указатель вершины стека уменьшается на единицу, так что заданный аргумент оказывается в регистре ST(1), затем производится вычисление тангенса, результат замещает прежнее значение аргумента в регистре ST(1), а в регистр ST(0) загружается константа 1.0. Успешное завершение отмечается сбросом в нуль бита C2 в регистре SWR.

Код	Пример
D9 F2	FPTAN

Примечание 1: указатель вершины стека будет уменьшен только если регистр ST(7) пуст, иначе команда FPTAN не будет исполнена.

Примечание 2: при использовании старых моделей арифметических сопроцессоров (до 80287 включительно) аргумент команды FPTAN должен быть в пределах от 0 до PI/4.

7.04-49 FRNDINT – округление числа

Команда FRNDINT (RouND to INTeger) преобразует действительное значение в верхнем регистре стека сопроцессора ST(0) в целое число посредством округления. Характер исполнения округления определяется двухбитовым полем RC (Rounding Control) в регистре CWR: при RC=00 производится округление до ближайшего целого числа, при RC=01 – округление вниз до ближайшего меньшего целого числа, при RC=10 – округление вверх до ближайшего большего целого числа, а при RC=11 – округление путем отбрасывания дробной части числа.

Код	Пример
D9 FC	FRNDINT

Примечание 1: поле RC – это биты 11 и 10 регистра CWR, их состояние можно записать в память командой FSTCW (7.04-55), изменить, а потом ввести в обратно в регистр CWR командой FLDCW (7.04-35).

7.04-50 FRSTOR – восстановить состояние сопроцессора

Команда FRSTOR (ReSTORe) восстанавливает полное состояние арифметического сопроцессора, включая состояния регистров стека и всех управляющих регистров, по записи длиной 96 или 108 байт, заранее заготовленной при помощи команды FSAVE (7.04-51). Операндом команды FRSTOR является адрес первого байта этой записи, причем длина записи определяется автоматически по режиму работы компьютера: реальному или защищенному. Поэтому важно, чтобы восстановление происходило в том же режиме, в котором производилась запись.

1-й байт	2-й байт	Байты данных	Пример
DD	(2,6,A)(0-7)	0-2	`FRSTOR [bp+si+ffff]`

7.04-51 FSAVE – сохранить состояние сопроцессора

Команда FSAVE записывает в память компьютера, начиная с указанного адреса, состояния регистров стека и всех управляющих регистров арифметического сопроцессора, а затем приводит управляющие регистры сопроцессора в исходное состояние так же, как это делает команда FINIT (7.04-28). Формируемая командой FSAVE запись длиной 96 или 108 байт позволяет впоследствии восстановить состояние сопроцессора командой FRSTOR (7.04-50).

1-й байт	2-й байт	Байты данных	Пример
DD	(3,7,B)(0-7)	0-2	`FSAVE [bp+si+ffff]`

7.04-52 FSCALE – умножение на степень числа 2

Команда FSCALE умножает действительное число в верхнем регистре сопроцессора ST(0) на число 2, возведенное в степень с целочисленным показателем, положительным или отрицательным. Если показатель степени, считываемый из регистра стека ST(1), не является целым числом, то перед возведением в степень производится его округление вниз до ближайшего меньшего целого числа. Результат умножения замещает первый сомножитель в регистре ST(0).

Код	Пример
D9 FD	FSCALE

7.04-53 FSQRT – квадратный корень

Команда FSQRT (SQuare RooT) вычисляет квадратный корень из положительного действительного числа в верхнем регистре ST(0) стека сопроцессора. Получаемый результат замещает собою операнд в регистре ST(0).

Код	Пример
D9 FA	FSQRT

7.04-54 FST – сохранить число

Команда FST (STore) копирует действительное число из верхнего регистра ST(0) стека сопроцессора в какой-либо другой регистр стека ST(1-7) или в ячейку памяти согласно указанному формату и адресу. Число, записываемое в память, подвергается округлению, если указанный в команде формат меньше, чем 10-байтовый формат регистра ST(0).

1-й байт	2-й байт	Байты данных	Примеры

D9
DD
DD
	(1,5,9)(0-7)
	(1,5,9)(0-7)
	D(0-7)
		0-2
		0-2
			`FST dword ptr [bp+si+ffff]`
			`FST qword ptr [bp+si+ffff]`
			`FST ST(1-7)`

Примечание 1: отладчик DEBUG.EXE ошибочно дизассемблирует как команду FST код "DF D(0-7)".

Примечание 2: команда FST позволяет осуществлять копирование в занятый регистр стека с замещением прежнего содержимого этого регистра.

7.04-55 FSTCW – сохранение регистра CWR

Команда FSTCW (STore Control Word) записывает в память по указанному адресу слово данных, копирующее все биты управляющего регистра CWR.

1-й байт	2-й байт	Байты данных	Пример
D9	(3,7,B)(8-F)	0-2	`FSTCW [bp+si+ffff]`

Примечание 1: восстановление состояния регистра CWR по сохраненному в памяти слову данных осуществляется командой FLDCW (7.04-35).

7.04-56 FSTENV – сохранение служебных регистров

Команда FSTENV записывает в память, начиная с указанного адреса, состояния всех служебных регистров сопроцессора: CWR (Control Word Register), SWR (Status Word Register), TWR (Tags Word Register), IPR (Instruction Pointer Register), DPR (Data Pointer Register). В отличие от команды FSAVE, команда FSTENV не приводит регистры сопроцессора в начальное состояние и не сохраняет содержимое регистров стека.

Данные из записи, сформированной командой FSTENV, могут быть загружены обратно в служебные регистры сопроцессора командой FLDENV (7.04-36).

1-й байт	2-й байт	Байты данных	Пример
D9	(3,7,B)(0-7)	0-2	`FSTENV [bp+si+ffff]`

7.04-57 FSTP – сохранение числа и выталкивание

Команда FSTP (STore and Pop) копирует действительное число из верхнего регистра ST(0) стека сопроцессора в память по указанному адресу или в любой другой регистр стека ST(1-7), а затем увеличивает указатель вершины стека на единицу, так что доступ к прежнему числу в регистре ST(0) утрачивается, а номера всех других регистров стека становятся на единицу меньше. Допустима запись в непустой регистр стека, прежнее содержимое которого утрачивается. Число, записываемое в память, подвергается округлению, если указанный в команде формат меньше, чем 10-байтовый формат регистра ST(0).

1-й байт	2-й байт	Байты данных	Примеры
D9	(1,5,9)(8-F)	0-2	`FSTP dword ptr [bp+si+ffff]`
DB	(3,7,B)(8-F)	0-2	`FSTP tbyte ptr [bp+si+ffff]`
DD	(1,5,9)(8-F)	0-2	`FSTP qword ptr [bp+si+ffff]`
DD	D(8-F)		`FSTP ST(1-7)`

Примечание 1: отладчик DEBUG.EXE ошибочно дизассемблирует как команду FSTP коды "D9 D(8-F)" и "DF D(8-F)".

7.04-58 FSTSW – копирование регистра SWR

Команда FSTSW копирует в указанный адрес состояние служебного регистра SWR (Status Word Register) арифметического сопроцессора, преимущественно в целях анализа результатов после исполнения сравнений. О расположении флагов в регистре SWR написано в 7.04-08 и 7.04-64.

1-й байт	2-й байт	Байты данных	Примеры	Примечание
DD DF	(3,7,B)(8-F) E0	0-2	`FSTSW [bp+si+ffff]` `ESC 3C,AL`	 *1

Примечание 1: процессоры не древнее 80486 исполняют команду FSTSW AX, которая копирует регистр SWR в регистр AX, но отладчик DEBUG.EXE "знает" только ее первоначальное название ESC 3C,AL.

7.04-59 FSUB – вычитание действительных чисел

Команда FSUB (SUBtract) вычитает действительное число, считываемое из ячейки памяти или из любого регистра сопроцессора, если он указан в качестве второго операнда, из другого действительного числа, находящегося в верхнем регистре ST(0) стека сопроцессора или в другом регистре стека ST(1-7), если он указан в качестве первого операнда. Результат замещает собою уменьшаемое в регистре ST(0) или в другом регистре стека ST(1-7), если он указан в качестве первого операнда.

1-й байт	2-й байт	Байты данных	Примеры

D8
D8
DC
DC
	(2,6,A)(0-7)
	E(8-F)
	(2,6,A)(0-7)
	E(8-F)
		0-2
		
		0-2
		
			`FSUB dword ptr [bp+si+ffff]`
			`FSUB ST,ST(0-7)`
			`FSUB qword ptr [bp+si+ffff]`
			`FSUB ST(1-7),ST`

7.04-60 FSUBP – вычитание и выталкивание

Команда FSUBP (SUBtract and Pop) вычитает действительное число, находящееся в верхнем регистре ST(0) стека сопроцессора, из другого действительного числа в другом регистре стека ST(1-7), замещает там уменьшаемое полученной разностью, а затем увеличивает указатель вершины стека на единицу, так что доступ к прежнему числу в регистре ST(0) утрачивается, а номера всех

других регистров стека, включая тот, куда помещен результат, становятся на единицу меньше.

Код	Пример
DE E(8-F)	`FSUBP ST(1-7),ST`

7.04-61 FSUBR – вычитание в обратном порядке

Команда FSUBR (SUBtract in Reverse order) вычитает действительное число, находящееся в верхнем регистре ST(0) стека сопроцессора или в другом регистре стека ST(1-7), если он указан в качестве первого операнда, из другого действительного числа, считываемого из ячейки памяти или из какого-либо регистра стека ST(0-7), если он указан в качестве второго операнда. Получаемая разность замещает вычитаемое в регистре ST(0) или в другом регистре стека ST(1-7), если он указан в качестве первого операнда.

1-й байт	2-й байт	Байты данных	Примеры
D8	(2,6,A)(8-F)	0-2	`FSUBR dword ptr [bp+si+ffff]`
D8	E(0-7)		`FSUBR ST,ST(0-7)`
DC	(2,6,A)(8-F)	0-2	`FSUBR qword ptr [bp+si+ffff]`
DC	E(0-7)		`FSUBR ST(1-7),ST`

7.04-62 FSUBRP – обратное вычитание и выталкивание

Команда FSUBRP (SUBtract in Reverse order and Pop) вычитает действительное число, находящееся в не-верхнем регистре ST(1-7) стека сопроцессора, из другого действительного числа – уменьшаемого – в верхнем регистре ST(0), замещает вычитаемое в регистре ST(1-7) полученной разностью, а затем увеличивает указатель вершины стека на единицу, так что доступ к прежнему числу в регистре ST(0) утрачивается, а номера всех других регистров стека, включая тот, куда помещен результат, становятся на единицу меньше.

Код	Пример
DE E(0-7)	`FSUBRP ST(1-7),ST`

7.04-63 FTST – сравнение с нулем

Команда FTST (TeST) сравнивает с нулем число, находящееся в верхнем регистре ST(0) стека сопроцессора. Состояния флагов C3 C2 C0 в служебном

регистре SWR приводятся в соответствие с результатом сравнения. Сначала следует проверить флаг C2: если C2 = 1, то операнды несопоставимы, и проверять дальше нечего. C3 = 1 отмечает равенство, то есть нулевое значение в регистре ST(0), а C0 = 1 соответствует отрицательному числу. Если число в регистре ST(0) положительное, то все три флага C0 C2 C3 должны быть сброшены в нуль. О том, как проверять состояния флагов C0 – C3, написано в 7.04-64.

Код	Пример
D9 E4	FTST

7.04-64 FXAM – определение типа операнда

Команда FXAM (eXAMine) служит для определения типа операнда, находящегося в верхнем регистре ST(0) стека сопроцессора. Результат отображается состоянием флагов C3 C2 C1 C0 в служебном регистре SWR. Флаг C1 показывает знак операнда, а состояния флагов C3 C2 C0 нужно интерпретировать согласно следующей таблице:

C3	C2	C0	Тип операнда
0	0	0	- неизвестный формат
0	0	1	- любой не-числовой формат
0	1	0	- правильное действительное число
0	1	1	- бесконечность (тег = 10)
1	0	0	- нуль (тег = 01)
1	0	1	- регистр ST(0) пуст (тег = 11).
1	1	0	- ненормализованное число

Чтобы проанализировать полученный результат, нужно скопировать слово статуса из регистра SWR командой FSTSW (7.04-58) предпочтительно в регистр AX, а далее либо проверять его командой TEST (7.03-90), либо загрузить во флаги центрального процессора командой SAHF (7.03-77). Флаги C0 C1 C2 C3 соответствуют в слове статуса (и в регистре AX) битам 08, 09, 10, 14, поэтому в однобайтовом регистре AH они окажутся в позициях 0, 1, 2, 6. При загрузке из AH во флаги центрального процессора флаг C3 будет отображен флагом нуля ZF, флаг C2 – флагом четности PF, флаг C0 – флагом переноса CF.

Код	Пример
D9 E5	FXAM

7.04-65 FXCH – обмен содержимого регистров

Команда FXCH (eXCHange) меняет местами содержимое указанного не-верхнего регистра ST(1-7) стека сопроцессора с содержимым верхнего регистра ST(0).

Код	Пример
D9 C(8-F)	FXCH ST(1-7)

Примечание 1: отладчик DEBUG.EXE ошибочно дизассемблирует как команду FXCH коды "DD C(8-F)" и "DF C(8-F)".

7.04-66 FXTRACT – разделение мантиссы и степени

Команда FXTRACT (eXTRACT exponent and significand) выполняет декомпозицию действительного числа в регистре ST(0) на мантиссу и показатель степени. Мантисса записывается в регистр ST(7), который в этот момент должен быть пуст. Затем команда FXTRACT уменьшает указатель вершины стека сопроцессора на единицу, так что регистры ST(0–6) оказываются переименованы в ST(1–7), а ST(7) – в ST(0). После переименования мантисса оказывается в верхнем регистре стека сопроцессора ST(0), а показатель степени – в регистре ST(1).

Код	Пример
D9 F4	FXTRACT

7.04-67 FYL2X – логарифм по произвольному основанию

Команда FYL2X вычисляет логарифм по основанию 2 от положительного числа, находящегося в верхнем регистре ST(0) стека сопроцессора, и умножает этот логарифм на число в регистре ST(1). Эта операция умножения позволяет преобразовать логарифм по основанию 2 в логарифм с любым другим основанием. Затем команда FYL2X увеличивает указатель вершины стека на единицу, так что регистр ST(1), куда помещено произведение, становится верхним регистром стека ST(0), а доступ к прежнему содержимому регистра ST(0) утрачивается. Конечный результат может быть представлен формулой ST(0)=ST(1)•log(ST(0)).

Код	Пример
D9 F1	FYL2X

7.04-68 FYL2XP1 – логарифм суммы ряда

Команда FYL2XP1 вычисляет логарифм по произвольному основанию так же, как это делает команда FYL2X (7.04-67), но принимает в качестве своего аргумента в регистре ST(0) сумму ряда, вычисляемую командой F2XM1 (7.04-01). Точность вычислений обеспечивается только в интервале значений этого аргумента от (–1+1/SQRT2) до (–1+SQRT2), что соответствует значениям логарифма по основанию 2 от –1/2 до +1/2. Умножение логарифма по основанию 2 на множитель в регистре ST(1) и увеличение указателя стека выполняются так же, как командой FYL2X (7.04-67). Вычисленный логарифм остается в верхнем регистре стека ST(0). Конечный результат исполнения команды FYL2XP1 выражается формулой ST(0)=ST(1)•log(1+ST(0))

Код	Пример
D9 F9	FYL2XP1

Глава 8 Вызовы обработчиков прерываний

Случаются ситуации, когда нормальную последовательность операций процессора приходится прерывать для исполнения неотложного запроса. При неожиданном появлении ошибок такие запросы может инициировать сам центральный процессор. Другие устройства посылают запросы через шины IRQ 00 – IRQ 15 контроллера прерываний. Исполняемые программы тоже могут вызывать прерывания с помощью команды INT (7.03-28). В любом случае запрос прерывания повлечет исполнение специальной подпрограммы – обработчика запрошенного прерывания, которая выполнит требуемое действие.

При нормальной работе компьютера в его памяти постоянно находится большое количество обработчиков прерываний. Их можно рассматривать как библиотеку стандартных подпрограмм. От умения пользоваться этой библиотекой в большой степени зависит результативность Вашего "общения" с компьютером.

Адреса вызова всех обработчиков прерываний сведены в таблицу прерываний. При защищенном и при реальном режимах работы процессора используются разные таблицы прерываний.

Таблицу прерываний защищенного режима заполняют 8-байтовыми дескрипторами адресов вызова обработчиков прерываний и служб API. Состав комплекта обработчиков, порядок размещения дескрипторов и место расположения самой таблицы прерываний в памяти компьютера не регламентированы, все это устанавливается "по усмотрению" той программы (или операционной системы), которая организует работу компьютера в защищенном режиме.

Таблица прерываний реального режима существенно более регламентирована. Прежде всего, ей выделено строго определенное место – первый килобайт оперативной памяти компьютера (от 0000:0000h до 0000:03FFh). Туда записаны не дескрипторы, а просто 4-байтовые адреса обработчиков прерываний, причем унифицированное расположение адресов вызова базовых функций BIOS в таблице прерываний реального режима служит основой совместимости программного обеспечения во всем классе AT-совместимых компьютеров. Последнее обстоятельство определяет важную роль таблицы прерываний реального режима и наше особое внимание к ней.

После перехода в защищенный режим таблица прерываний реального режима не пропадает, а иногда продолжает работать. Операционные системы переключают процессор обратно в реальный режим на время исполнения функций BIOS для обслуживания специфических аппаратных средств материнской платы, а также функций DOS, вызываемых программами из "окна DOS". Эти манипуляции

скрыты от пользователя, но именно из-за них программы в "окне DOS" исполняются заметно медленнее, чем в реальном режиме.

Смещение для считывания адреса любого обработчика из таблицы прерываний реального режима вычисляется автоматически умножением номера прерывания на 4. Каждый адрес состоит из двух двухбайтовых слов. Их надо интерпретировать в обратном порядке: четвертый и третий байты образуют сегментный адрес, а второй и первый байты образуют смещение. К примеру, содержащиеся в таблице данные 59 F8 00 F0 соответствуют адресу вызова F000:F859h.

Любой вызов прерывания по номеру исходит из предположения, что извлекаемый из таблицы адрес указывает на исполняемый код обработчика прерывания. Однако имеются исключения: смещения 0074h, 0078h, 007Ch, 0104h, 010Ch, 0118h в таблице прерываний реального режима содержат указатели на не-исполняемые таблицы данных, перечисленные в приложении A.12-1. Соответствующие числа 1Dh, 1Eh, 1Fh, 41h, 43h, 46h нельзя использовать в качестве номеров вызываемых прерываний.

Часть таблицы прерываний реального режима, обычно до номера 1Ch, заполняет адресами система BIOS. Ядро MS-DOS7 загружает в память свои обработчики прерываний, обычно с номерами от 20h до 2Eh, и добавляет в таблицу их адреса. Потом адреса добавляет почти каждый загружаемый драйвер. На каждом этапе ранее записанные адреса могут быть замещены, некоторые даже бывают замещены более чем дважды. Тем самым вызов прерывания оказывается "перехвачен" другим обработчиком. Наиболее часто это делается для обеспечения условного исполнения дополнительных функций, а если условие не удовлетворяется, то вызов переадресуется прежнему обработчику прерывания. В результате бывает трудно предвидеть, чей именно обработчик выполнит конкретный запрос: это будет зависеть от конфигурации загрузки Вашего компьютера. По той же причине строго распределить материалы главы 8 по разделам оказывается невозможно.

Поскольку обработка прерываний зависит от многих факторов, постольку ответственность за получаемые результаты нельзя возлагать только на MS-DOS7. Достоверность результатов приходится проверять после почти каждого вызова. Тем не менее существует большое число функций, которые стали базой для сохранения совместимости программных кодов, и которые почти наверняка можно вызвать в любом компьютере, работающем под управлением MS-DOS7. Избранные программные прерывания для вызова таких функций описаны ниже в этой главе.

8.01 Обработчики от BIOS (INT 00 – INT 1C)

8.01-01 INT 00 – отработка ошибки деления на нуль

Если при исполнении операций деления DIV (7.03-21) или IDIV (7.03-24) возникает переполнение в том регистре, куда должно быть помещено частное, то процессор инициирует прерывание INT 00. Устанавливаемый по умолчанию обработчик этого прерывания прекращает исполнение программы, вызвавшей ошибку, выводит на экран сообщение "Your program caused a divide overflow error..." (= Ваша программа вызвала ошибку переполнения при делении...), и затем передает управление DOS.

Примечание 1: когда вызов обработчика INT 00 происходит "по инициативе" процессора, тогда процессор оставляет в стеке в составе адреса возврата не смещение следующей команды, а смещение той самой команды деления, которая вызвала переполнение.

8.01-02 INT 01 – прерывание пошагового исполнения

Если в регистре флагов процессора установлен флаг прерывания TF (trap flag), то процессор вызывает обработчика прерывания INT 01 после исполнения каждой команды. Это позволяет осуществлять пошаговое отлаживание программ. Помимо того, все современные процессоры, начиная с модели 80386, содержат отладочные регистры (A.11-5), позволяющие вызывать обработчик прерывания INT 01 при любом обращении к заранее определенным областям памяти или к портам. Вызов обработчика прерывания INT 01 происходит также при исполнении недокументированного кода F1h.

Система BIOS по умолчанию устанавливает вместо обработчика прерывания INT 01 отсылку на команду IRET (7.03-30), которая просто возвращает управление следующей команде. Отлаживающие программы должны сами записывать в таблицу прерываний адрес своего обработчика прерывания INT 01. Тогда каждый вызов INT 01 будет останавливать исполнение отлаживаемой программы перед следующей командой и передавать управление тому процессу-отладчику, который вызвал исполнение отлаживаемой программы.

Примечание 1: флаг TF сбрасывается в нуль после исполнения каждой команды, но его исходное состояние (до сброса) записывается в стек вместе с адресом возврата при вызове прерывания INT 01. Поэтому обработчик прерывания всегда работает при сброшенном состоянии флага TF, но когда он кончает свою работу, завершающая команда IRET восстанавливает из стека исходное состояние флага TF.

Примечание 2: в качестве адреса возврата из прерывания INT 01 в стек обычно помещается адрес следующей команды, но при установке отладочных точек может быть передан адрес текущей команды. Адреса следует различать по состоянию флагов в регистре DR6 (A.11-5).

8.01-03 INT 02 – немаскируемое прерывание

Вызов обработчика прерывания INT 02 производится при поступлении на процессор сигнала по отдельной шине NMI (Non-Maskable Interrupt). В отличие от других аппаратно вызываемых прерываний, вызов по шине NMI нельзя заблокировать командой CLI (7.03-12) или битом маски в контроллере прерываний. У обработчика прерывания INT 02 особая задача: реагировать на возникновение аварийных ситуаций, например, при обнаружении сбоев памяти. Неисправимые аварийные ситуации вызывают индикацию соответствующего сообщения и кончаются остановом компьютера. Во многих компьютерах вызов по шине NMI поступает при появлении первых признаков пропадания электропитания для выполнения самых неотложных операций по предотвращению потери данных. В таких случаях обработчик прерывания INT 02, завершив свою миссию, обычно передает управление той программе, исполнение которой было прервано: ей дается шанс продолжить исполнение, если тревога окажется ложной, и сбой электропитания все-таки не произойдет.

Примечание 1: вызов по шине NMI можно заблокировать, если командой OUT (7.03-66) послать байт с установленным битом 7 в порт 70h CMOS-памяти BIOS, причем за этим должна последовать операция обращения к порту 71h (примечание 1 к A.14-1). Блокировка вызова NMI на момент обращения к CMOS-памяти BIOS предотвращает возможность искажения данных в ней.

8.01-04 INT 03 – точка останова

Процессор отвечает вызовом обработчика прерывания INT 03 на код CCh (7.03-28), встреченный в позиции первого байта исполняемой машинной команды. Код CCh обычно вставляют программы-отладчики для того, чтобы остановить исполнение отлаживаемой программы в заданном месте. Так действует, в частности, отладчик DEBUG.EXE.

Система BIOS по умолчанию устанавливает вместо обработчика прерывания INT 03 отсылку на команду IRET (7.03-30), которая просто возвращает управление следующей команде. Отлаживающие программы должны сами заранее внести в

таблицу прерываний адрес того обработчика, которому предстоит обслуживать вызовы прерывания INT 03.

8.01-05 INT 04 – отработка ошибки переполнения

В отличие от прерывания INT 00, которое заставляет программу реагировать на ошибки математических операций сразу, прерывание INT 04 обслуживает отложенную обработку ошибок по команде INTO (7.03-29). Встретив в позиции первого байта машинной команды код CEh команды INTO, процессор проверяет флаг переполнения OF (Overflow Flag), и если он не сброшен, то вызывает обработчика прерывания INT 04. Устанавливаемый по умолчанию обработчик не останавливает исполнение, а просто возвращает управление следующей команде в той программе, из которой он вызван. Каждой программе предоставлена возможность самой установить такой обработчик прерывания INT 04, который мог бы адекватно реагировать на возникновение ошибки переполнения.

8.01-06 INT 05 – распечатка экрана

Из-за несогласованности действий фирм IBM и Intel на обработчик прерывания INT 05 оказались возложены две взаимно не связанные задачи.

В IBM-совместимых компьютерах обработчик прерывания INT 05, устанавливаемый системой BIOS, выполняет распечатку воспроизводимого на экране изображения. Процедуру распечатки запускает пользователь клавишной комбинацией Shift – PrtSc. Распознав эту комбинацию, обработчик прерывания INT 09 вызывает на исполнение обработчика прерывания INT 05. Последний сначала проверяет байт статуса принтера по адресу 0000:0500h в области служебных данных (A.12-1). Состояния байта статуса означают следующее:

00h – принтер подключен к порту LPT1 и готов к работе;
01h – принтер занят выполнением предыдущего задания;
FFh – предыдущий запрос к принтеру завершился неудачно.

Если в байт статуса записан код 00h, то текущая экранная страница выводится через порт LPT1 на принтер и распечатывается.

Процессоры фирмы Intel (начиная с 80286) способны сами вызывать обработчик прерывания INT 05 в случае ошибок выхода индекса за границы массива при исполнении машинной команды BOUND. Очевидно, что в таком случае миссия обработчика прерывания INT 05 должна быть совсем иной.

Команда BOUND (62h) – одна из относительно новых команд, которые "неизвестны" отладчику DEBUG.EXE и потому не описаны в главе 7. Во избежание конфликтов каждая программа реального режима, которая использует команду

BOUND, должна устанавливать свои обработчики прерываний, предотвращающие неадекватную реакцию на вызовы INT 05. У программ защищенного режима подобных проблем не бывает, потому что операционные системы формируют таблицу прерываний защищенного режима заново и не связывают прерывание INT 05 с функцией распечатки изображений.

Примечание 1: вызов обработчика прерывания INT 05 часто подменяется вызовом процедуры из постоянного запоминающего устройства видеокарты по нажатию клавиши PrtSc (8.01-66).

Примечание 2: при вызове INT 05 командой BOUND в качестве адреса возврата в стек помещается адрес не следующей команды, а самой команды BOUND. Такой вызов нельзя перенаправлять команде IRET, так как это приведет к "зависанию" компьютера в бесконечном цикле возвратов и вызовов.

8.01-07 INT 06 – обнаружение неизвестного кода

Центральный процессор отвечает вызовом обработчика прерывания INT 06 на попытки исполнения ошибочных команд, в частности, в случаях обнаружения

- вызова команд защищенного режима при работе в реальном режиме;
- префикса LOCK перед командами, которые не обращаются к памяти;
- регистрового операнда у команд, которые работают только с памятью;
- команд, которые процессор не способен распознать.

Посредством обработчиков прерывания INT 06 можно эмулировать исполнение команд современных процессоров на компьютере с устаревшим процессором. Однако устанавливаемый по умолчанию обработчик прерывания INT 06 просто возвращает управление следующей команде в вызывающей программе.

8.01-08 INT 07 – обслуживание функций сопроцессора

Центральный процессор вызывает обработчика прерывания INT 07 в ответ на попытки исполнения команды ESC (7.03-22) или любой команды арифметического сопроцессора (7.04), если установлен бит 02h в управляющем регистре CR0 (A.11-4). Бит 02h может быть установлен намеренно с целью вызова программы эмуляции функций сопроцессора. Некоторые системы BIOS делают это автоматически, если арифметический сопроцессор в компьютере отсутствует.

В современных компьютерах арифметический сопроцессор всегда имеется, так что эмуляция его функций не нужна. Поэтому на прерывание INT 07 может быть возложена другая миссия: принятие должных мер при наличии в сопроцессоре немаскированных исключений в момент переключения задач. С этой целью предусмотрен вызов INT 07 префиксом WAIT (7.02-05), когда одновременно

установлены биты 01h и 03h в управляющем регистре CR0 (A.11-4). Чтобы запретить вызов INT 07 префиксом WAIT, нужно сбросить бит 01h.

Для реализации любой миссии прерывания INT 07 необходимо загрузить соответствующий обработчик. Обычно устанавливаемый по умолчанию обработчик прерывания INT 07 просто возвращает управление следующей команде в вызывающей программе.

8.01-09 INT 08 – INT 0F: обслуживание запросов IRQ 0 – IRQ 7

В реальном режиме работы компьютера эта группа обработчиков прерываний обслуживает запросы, поступающие по линиям IRQ 0 – IRQ 7 от разных устройств к первому контроллеру прерываний. Некоторые линии запроса предназначены для определенных устройств, указанных в четвертой колонке таблицы, но остальные готовы принять запрос от любого устройства, если оно сконфигурировано для посылки запроса именно по этой линии и поддерживается драйвером, загружающим обработчик соответствующего прерывания.

Помимо того, современные процессоры фирмы Intel вызывают некоторые прерывания из этой же группы для обслуживания особых ситуаций (исключений), не связанных с запросами внешних устройств. Те особые ситуации, которые могут приводить к вызову прерываний INT 08 – INT 0F при работе процессоров в реальном режиме, перечислены в примечаниях к приведенной ниже таблице. Контроллер прерываний не регистрирует вызовы, генерируемые процессором. Если командой OUT (7.03-66) послать байт 0Ah в порт 20h контроллера прерываний, то из этого же порта командой IN (7.03-26) можно будет считать байт, в котором поступление запроса по каждой линии отмечено установлением в единицу соответствующего бита, показанного в колонке 3 приведенной ниже таблицы. По этому признаку внешние вызовы можно отличить от вызовов "по инициативе" процессора.

Генерируемые процессором вызовы передают через стек в качестве адреса возврата адрес не следующей команды, а той, из-за которой исключение возникло. Это дает возможность повторить операцию после устранения причины исключения, но, с другой стороны, такие вызовы нельзя отсылать команде IRET, потому что компьютер зависнет в бесконечном цикле возвратов и вызовов. Если обработчик прерывания не может устранить причину исключения, то повторение необходимо предотвратить либо прекращением исполнения программы, либо корректировкой находящегося в стеке адреса возврата. Кроме того, на время обслуживания каждого генерируемого процессором вызова надо блокировать поступление конкурирующего вызова через контроллер прерываний путем посылки командой OUT (7.03-66) в порт 21h байта маски с установленным в единицу битом блокируемой линии, показанным в колонке 3 приведенной ниже таблицы.

Вызов	Линия	Маска	Источник запросов	Примечания
INT 08	IRQ 0	бит 0	Системный таймер	*1
INT 09	IRQ 1	бит 1	Контроллер клавиатуры	*2
INT 0A	IRQ 2	бит 2	2-й контроллер прерываний	*3
INT 0B	IRQ 3	бит 3		*4
INT 0C	IRQ 4	бит 4	Порт COM-1	*5
INT 0D	IRQ 5	бит 5		*6
INT 0E	IRQ 6	бит 6	Контроллер флоппи-диска	*7
INT 0F	IRQ 7	бит 7	Параллельный порт LPT-1	

Примечание 1: вызовы обработчика прерывания INT 08 следуют регулярно 18.2 раза в секунду для обеспечения счета времени. Для обслуживания пользовательских программ обработчик прерывания INT 08, в свою очередь, вызывает INT 1C (8.01-96). Процессор может также вызывать INT 08 в ситуации "двойного отказа", которая обычно должна выводить компьютер в перезагрузку.

Примечание 2: обработчик прерывания INT 09 реагирует на каждую клавишу, обслуживает буфер клавиатуры и готовит коды, приведенные в приложении A.02-1, которые доступны программам через прерывание INT 16. Однако очевидные действия следуют только в ответ на некоторые клавишные комбинации, перечисленные в разделе 1.01.

Примечание 3: второй контроллер прерываний обслуживает линии запроса IRQ 8 – IRQ 15 и вызывает прерывания INT 70 – INT 77 (8.03-75).

Примечание 4: предпочтительным источником запросов по линии IRQ 3 является последовательный порт COM-2, если он имеется в компьютере.

Примечание 5: процессоры могут вызывать прерывание INT 0C при выходе стека за границу выделенного ему сегмента.

Примечание 6: процессоры могут вызывать прерывание INT 0D при попытках адресации за пределы границ сегментов кода или данных.

Примечание 7: процессоры могут вызывать прерывание INT 0E при попытках обращения к "закрытым" страницам памяти (адреса которых в буфере TLB отсутствуют).

Примечание 8: путем перепрограммирования контроллера прерываний отображение внешних вызовов IRQ 0 – IRQ 7 на INT 08 – INT 0F может быть заменено иным, чтобы они не поступали к тем же обработчикам, которые обслуживают генерируемые процессором исключения. Такое перепрограммирование обычно выполняют при подготовке к переключению процессора в защищенный режим согласованно с формированием таблицы прерываний для защищенного режима.

8.01-10 INT 10\AH=00h – переключение видеорежима

При вызове:

AH = 00h
AL – код задаваемого видеорежима из приложения A.10-1

При возврате:

содержимое AL может быть изменено

Примечание 1: эта операция переводит любые видеокарты, в том числе новые, в видеорежимы, совместимые со старыми видеокартами VGA. Для перевода в видеорежимы SVGA нужно вызывать INT 10\AX=4F02h.

Примечание 2: код установленного видеорежима записывается в область данных BIOS, показанную в приложении A.10-6, по адресу 0040:0049h.

Примечание 3: координированное переключение видеорежима вместе со сменой параметров управления манипулятором типа "мышь" обеспечивается вызовом драйвера "мыши" через прерывание INT 33\AX=0028h (8.03-52).

Примечание 4: переключение видеорежимов сопровождается гашением экрана дисплея на 1 – 2 секунды, дискомфортным для зрительного восприятия. Потому лишних переключений желательно избегать. По той же причине для очистки экрана лучше пользоваться вызовом INT 10\AH=06h (8.01-15).

8.01-11 INT 10\AH=01h – размер курсора в текстовых видеорежимах

При вызове:

AH = 01h
CH – биты 0 – 4: - номер верхней граничной строки курсора
CL – биты 0 – 4: – номер нижней граничной строки курсора

Примечание 1: строки считают в пределах знакоместа сверху вниз. Установленные номера верхней и нижней строк курсора записаны в область данных BIOS, показанную в приложении A.10-6, по адресу 0040:0060h. Размер знакоместа записан там же по адресу 0040:0085h, но его также можно получить с помощью INT 10\AX=1130h.

Примечание 2: если установить в единицу бит 5 в регистре CH, то можно сделать курсор невидимым.

8.01-12 INT 10\AH=02h – изменение положения курсора

При вызове:

AH = 02h
BH – номер экранной страницы (примечания 1 и 2 к 8.01-14)
DH – номер ряда (счет от верхнего ряда с номера 00h)

DL – номер колонки (счет от левой колонки с номера 00h)

Примечание 1: на каждой экранной странице положение курсора задается отдельно.

Примечание 2: до 8 пар координат курсора для разных экранных страниц записываются в область данных BIOS, показанную в приложении A.10-6, начиная с адреса 0040:0050h.

8.01-13 INT 10\AH=03h – определение размера и положения курсора

При вызове:

AH = 03h

BH – номер экранной страницы (примечания 1 и 2 к 8.01-14)

При возврате:

CH – номер верхней граничной строки курсора

CL – номер нижней граничной строки курсора

DH – номер ряда (счет от верхнего ряда с номера 00h)

DL – номер колонки (счет от левой колонки с номера 00h)

Содержимое регистра AX может быть не сохранено.

8.01-14 INT 10\AH=05h – выбор активной экранной страницы

При вызове:

AH = 05h

AL – номер запрашиваемой экранной страницы

Примечание 1: если запрашиваемый номер больше номера последней экранной страницы в данном видеорежиме, то выбор не будет подтвержден функцией INT 10\AH=0Fh. Количество экранных страниц для видеорежимов VGA можно узнать из таблицы, возвращаемой функцией INT 10\AX=1B00h (приложение A.10-2, смещение 29h), а для видеорежимов SVGA – из таблицы, возвращаемой функцией INT 10\AX=4F01h (приложение A.10-7, смещение 1Dh).

Примечание 2: экранные страницы нумеруются от 00h, номер последней страницы на единицу меньше числа имеющихся экранных страниц. Наиболее часто используемый текстовый видеорежим 03h предоставляет 4 экранных страницы от 00h до 03h.

Примечание 3: переключение экранных страниц согласованно с перенаправлением курсора манипулятора "мышь" на выбранную страницу выполняет обработчик прерывания INT 33\AX=001Dh (8.03-47), устанавливаемый драйвером "мыши".

8.01-15 INT 10\AH=06h-07h – прокрутка экранного окна

Прокруткой принято называть перемещение текста вверх или вниз в пределах всего экрана или только "окна", то есть прямоугольной области, составляющей часть воспроизводимого на экране изображения. Строки, выдвигаемые из-под кромки "окна", не содержат текста и заполняются пробелами заданного цвета. Эта же процедура при AL=00h позволяет очистить от текста и заполнить заданным цветом любой прямоугольный участок изображения, в том числе и весь экран.

При вызове:

AH = 06h – направление прокрутки вверх,
= 07h – направление прокрутки вниз
AL – интервал прокрутки в строках (или 00h, чтобы очистить окно)
BH – биты 4 – 7: цвет заполнения (столбцы 1 и 4 таблицы A.10-5)
CH – ряд, CL – колонка левого верхнего угла окна
DH – ряд, DL – колонка нижнего правого угла окна

Примечание 1: эта процедура действует только в текстовых видеорежимах.
Примечание 2: эта процедура затрагивает только ту экранную страницу, которая является активной в данный момент.
Примечание 3: некоторые версии BIOS при исполнении данной процедуры могут изменять содержимое регистров BP и DS.

8.01-16 INT 10\AH=08h – чтение знака из позиции курсора

При вызове:

AH = 08h
BH – номер экранной страницы (примечания 1 и 2 к 8.01-14)

При возврате:

AH – цвет, как в приложении A.10-5 (только в текстовых режимах)
AL – считанный знак в коде ASCII

Примечание 1: в графических видеорежимах правильно распознаются только знаки белого цвета. Если знак не распознан, то AL=00h.

8.01-17 INT 10\AH=09h-0Ah – запись знака в позицию курсора

При вызове:

AH = 09h
AL – код ASCII того знака, который надлежит записать
BH – номер экранной страницы (примечания 1 и 2 к 8.01-14)
BL – байт цвета, показанный в приложении A.10-5.
CX – число, сколько раз подряд надо записать указанный знак.

Примечание 1: индицируются все знаки, включая 0Dh (CR), 0Ah (LF), 08h (BS) и другие служебные символы, упоминаемые в разделе A.02-08.

Примечание 2: в графических видеорежимах с количеством цветов менее 256, при установленном бите 7 в регистре BL запись знака осуществляется посредством исполнения операции "исключающее ИЛИ" (XOR).

Примечание 3: в графических видеорежимах число в регистре CX не должно быть больше, чем число знаков, которые можно разместить справа от позиции курсора в том же ряду.

Примечание 4: обработчик прерывания INT 10\AH=0Ah записывает знаки так же, но игнорирует байт цвета в BL. Введенные знаки будут иметь цвет такой же, как у остального текста.

Примечание 5: позиция курсора не смещается независимо от количества повторов записи знака.

Примечание 6: если установлен графический 256-цветный видеорежим (например, 13h), то в BH следует указывать байт цвета фона, а в BL – байт цвета знака.

8.01-18 INT 10\AH=0Bh – цвет фона или рамки

Для графических видеорежимов эта процедура задает цвет фона, а для текстовых видеорежимов – цвет рамки экрана.

При вызове:

AH = 0Bh
BX – биты 2, 1, 0 = красный, зеленый, синий соответственно

Примечание 1: рамка экрана на многих современных LCD-дисплеях отображается неправильно (в виде полоски) или вообще не отображается.

8.01-19 INT 10\AH=0Ch – рисование точки

При вызове:

AH = 0Ch
AL – байт цвета, показанный в приложении A.10-5.
BH – номер экранной страницы (примечания 1 и 2 к 8.01-14)
CX – номер столбца
DX – номер строки

Примечание 1: рисование точки выполняется только в графических видеорежимах.

Примечание 2: в графических видеорежимах с количеством цветов менее 256, при установленном бите 7 в регистре AL запись точки осуществляется посредством исполнения операции "исключающее ИЛИ" (XOR).

Примечание 3: если установленный видеорежим поддерживает только одну экранную страницу, то содержимое регистра BH игнорируется.

Примечание 4: вызов INT 10\AH=0Ch удобен для рисования линий, но для заполнения площадей изображения предпочтительна более быстрая прямая запись в видеопамять (8.01-39).

8.01-20 INT 10\AH=0Dh – считывание цвета точки

При вызове:

AH = 0Dh
BH – номер экранной страницы (примечания 1 и 2 к 8.01-14)
CX – номер столбца
DX – номер строки

При возврате:

AL – байт цвета

Примечание 1: считывание цвета точки выполняется только в графических видеорежимах.

Примечание 2: если установленный видеорежим поддерживает только одну экранную страницу, то содержимое регистра BH игнорируется.

8.01-21 INT 10\AH=0Eh – телетайпная запись знака

Обработчик прерывания INT 10\AH=0Eh заносит знак в текущую позицию курсора и затем перемещает курсор в следующее знакоместо. Если в заполняемом ряду знаков нет свободных знакомест, то курсор переводится в следующий ряд.

При вызове:

AH = 0Eh
AL – код ASCII знака, который надлежит записать
BH – номер экранной страницы (примечания 1 и 2 к 8.01-14)
BL – цвет знака (только в графических видеорежимах)

Примечание 1: служебные коды, перечисленные в разделе A.02-08, включая 07h (BEL) и 08h (BS), не индицируются, а исполняются как команды.

Примечание 2: в текстовых режимах цвет знака остается тем же, каким он был установлен ранее для того же знакоместа.

8.01-22 INT 10\AH=0Fh – выяснение видеорежима

При вызове:

AH = 0Fh

При возврате:

AH – число знакомест в ряду или в строке

AL – код видеорежима, объясняемый в приложении A.10-1
BH – номер экранной страницы (примечания 1 и 2 к 8.01-14)

Примечание 1: если код видеорежима был задан с установленным битом 7 (не очищать экран), то и возвращен он будет с установленным битом 7.

Примечание 2: эта функция не определяет коды видеорежимов SVGA, и для всех текстовых видеорежимов SVGA возвращает либо AL=07h (монохромный видеорежим), либо AL=03h (цветной видеорежим).

8.01-23 INT 10\AX=1003h – переключение роли бита 7

В текстовых видеорежимах бит 7 в байте цвета (A.10-5) может быть использован либо для задания мерцания, либо в качестве бита яркости фона: это зависит от состояния управляющего бита, которое можно изменить вызовом INT 10\AX=1003h.

При вызове:

AX = 1003h
BX – вид операции:
= 0000h – разрешить управление яркостью фона
= 0001h – разрешить управление мерцанием знаков.

Примечание 1: состояние управляющего бита отображается битом 5 в байте по адресу 0040:0065h в области данных BIOS (A.10-6), а также битом 5 в байте со смещением 2Dh в блоке данных (A.10-2), возвращаемом обработчиком прерывания INT 10\AX=1B00h.

8.01-24 INT 10\AX=1010h – установка яркости цветов

Обработчик прерывания INT 10\AX=1010h записывает в регистр цифро-аналогового преобразователя уровни от 0 до 3Fh, определяющие яркости каждого из трех основных цветов в той комбинации цветов, за которую отвечает данный регистр.

При вызове:

AX = 1010h
BX – номер регистра цифро-аналогового преобразователя
CH – новое значение уровня зеленого цвета
CL – новое значение уровня синего цвета
DH – новое значение уровня красного цвета

Примечание 1: для записи уровней доступны все регистры цифро-аналогового преобразователя видеокарты, но не все из них задействованы: это

зависит от установленного видеорежима. В частности, цвет фона в 16-цветных видеорежимах определяется регистрами 0 – 7.

8.01-25 INT 10\AX=1015h – считывание уровней яркости

Обработчик прерывания INT 10\AX=1015h считывает из регистра цифро-аналогового преобразователя видеокарты записанные там уровни, определяющие яркости каждого из трех основных цветов в той комбинации цветов, за которую отвечает запрошенный регистр.

При вызове:

AX = 1015h
BX – номер регистра цифро-аналогового преобразователя

При возврате:

CH – значение уровня зеленого цвета
CL – значение уровня синего цвета
DH – значение уровня красного цвета
содержимое регистра AX может быть изменено.

8.01-26 INT 10\AX=1018h – маска цвета

При вызове:

AX = 1018h
BL – новая маска, которую надлежит установить

Примечание 1: биты 0 – 2 маски включают синий, зеленый и красный цвета фона, биты 3 – 5 делают то же самое для знаков переднего плана. Состояние битов 6 и 7 безразлично. Нормальная маска должна быть байтом FFh: все цвета включены.

Примечание 2: команда CLS (3.05) не возвращает маску в нормальное состояние.

8.01-27 INT 10\AX=1100h – загрузка шрифта в текстовом видеорежиме

При вызове:

AX = 1100h
BH – число байтов, представляющих в шрифте один знак.
BL – идентификатор загружаемого блока (примечание 1 к 8.01-28)
CX – число загружаемых или замещаемых знаков
DX – смещение в блоке, начиная с которого надо загружать знаки
ES:BP – указатель (адрес) загружаемой таблицы знаков

Примечание 1: предполагается, что полная таблица шрифта содержит FFh знаков.

Примечание 2: каждый байт в коде знака представляет одну строку на экране, так что число байтов, представляющих каждый знак (число в регистре BH), равно числу строк в знакоместе.

Примечание 3: при загрузке шрифта устанавливается текстовый видеорежим, соответствующий загружаемому шрифту, но видеобуфер не очищается.

Примечание 4: если предстоит загружать несколько блоков шрифта, для них надо заранее, при загрузке DOS, выделить участки памяти с помощью драйвера DISPLAY.SYS (5.02-02), иначе можно будет пользоваться только выделяемым по умолчанию блоком с идентификатором 00h.

Примечание 5: обработчик прерывания INT 10\AX=1110h также загружает шрифт и требует те же исходные данные в регистрах, но переопределяет состояние видеоконтроллера. Вызывать INT 10\AX=1110h следует сразу после установки видеорежима и экранной страницы 0.

8.01-28 INT 10\AX=1103h – переключение шрифтов

Обработчик прерывания INT 10\AX=1103h переключает знакогенератор на другой шрифт, который должен быть заранее загружен в один из блоков памяти знакогенератора. Знакогенераторы EGA-совместимых и VGA-совместимых видеокарт позволяют активизировать два блока, давая тем самым возможность показывать знаки двух шрифтов одновременно. Выборка знака из того или другого шрифта будет зависеть от бита 3 в байте цвета, указываемом, в частности, в регистре BL при выводе знаков с помощью INT 10\AH=09h или INT 10\AH=0Eh.

При вызове:

AX = 1103h
BL – идентификатор блока шрифта в памяти знакогенератора

Примечание 1: идентификатор блока шрифта представляет собой байт, в котором выделены 2 поля: одно составляют биты 4,1,0, а другое – биты 5,3,2. В каждое поле записывается номер от 0 до 7 блока памяти, содержащего загруженный шрифт. Если номера в обоих полях совпадают, то будет адресован один шрифт, и тогда бит 3 в байте цвета (A.10-5) будет определять яркость. В частности, в EGA-совместимых видеокартах, обслуживающих только 4 шрифта, их блоки адресуются идентификаторами 00h, 05h, 0Ah, 0Fh.

Примечание 2: возможность активизации двух шрифтов одновременно нужно проверять по значению 9-го байта в таблице статической функциональности (A.10-3), адрес которой можно получить посредством INT 10\AX=1B00h.

Примечание 3: для активизации двух шрифтов в поля идентификатора блока шрифта должны быть внесены разные номера блоков. При этом

выборка знаков, у которых бит 3 в байте цвета сброшен в нуль, будет производиться из того блока, номер которого вписан в первое поле (биты 4, 1, 0), а выборка знаков, у которых бит 3 в байте цвета установлен в единицу, будет производиться из того блока, номер которого вписан во второе поле (биты 5, 3, 2).

8.01-29 INT 10\AX=1104h – загрузка стандартного шрифта 8x16

Обработчик прерывания INT 10\AX=1104h загружает из постоянной памяти BIOS тот самый шрифт, который принимается по умолчанию и отображает американскую кодовую страницу CP437. При загрузке устанавливается текстовый видеорежим 3, формат которого (80x25) соответствует данному шрифту.

При вызове:

AX = 1104h
BL – идентификатор загружаемого блока (примечание 1 к 8.01-28)

Примечание 1: обработчик прерывания INT 10\AX=1114h принимает те же исходные данные и делает то же самое, но при этом переопределяет состояние видеоконтроллера. Вызывать INT 10\AX=1114h следует сразу после установки видеорежима и экранной страницы 0.

Примечание 2: если предстоит загружать несколько блоков шрифта, для них надо заранее, при загрузке DOS, выделить участки памяти с помощью драйвера DISPLAY.SYS (5.02-02), иначе можно будет пользоваться только выделяемым по умолчанию блоком с идентификатором 00h.

Примечание 3: в постоянной памяти BIOS также имеются шрифты форматов 8x8 и 8x14. Шрифт 8x8 загружается аналогичным образом посредством INT 10\AX=1102h (и AX=1112h), а монохромный шрифт 8x14 – посредством INT 10\AX=1101h (и AX=1111h). Вызовы с AX=1111h и AX=1112h переопределяют состояние видеоконтроллера.

8.01-30 INT 10\AX=1121h – загрузка шрифта для графического видеорежима

При вызове:

AX = 1121h
BL – число рядов: 01h – 14 рядов, 02h – 25 рядов, 03h – 43 ряда, а 00h означает, что число рядов указано в регистре DL.
CX – число байтов на один знак
DL – число рядов, если BL = 00h, иначе DL игнорируется
ES:BP – указатель на первый байт загружаемого шрифта

Примечание 1: непосредственно перед вызовом INT 10\AX=1121h необходимо заново провести установку графического видеорежима.

Примечание 2: шрифты для графических видеорежимов не требуют подготовки блоков знакогенератора. Вместо этого указатель на загруженный шрифт записывается в таблицу прерываний по адресу 0000:010Ch (иногда этот указатель называют вектором INT 43).

Примечание 3: определяемые пользователем конфигурации знаков 80h – FFh для шрифтов формата 8x8 графического видеорежима могут быть загружены с помощью INT 10\AX=1120h. Поскольку формат шрифта фиксирован, содержимое регистров BL, CX и DL игнорируется, таблица шрифта имеет фиксированную длину 400h, а указатель на нее (называемый иногда вектором INT 1F) записывается в таблицу прерываний по адресу 0000:007Ch.

8.01-31 INT 10\AX=1124h – стандартный шрифт графического видеорежима

Обработчик прерывания INT 10\AX=1124h предназначен для загрузки стандартного шрифта 8x16, отображающего американскую кодовую страницу CP437, при работе в графических видеорежимах. Используется таблица шрифта, содержащаяся в постоянной памяти BIOS. Указатель на первый байт таблицы шрифта заносится в таблицу прерываний по адресу 0000:010Ch.

При вызове:

AX = 1124h

Примечание 1: непосредственно перед вызовом INT 10\AX=1124h необходимо заново произвести установку графического видеорежима.

Примечание 2: точно так же INT 10\AX=1122h загружает шрифт формата 8x14.

Примечание 3: точно так же INT 10\AX=1123h загружает шрифт формата 8x8.

8.01-32 INT 10\AX=1130h – запрос сведений о шрифте

При вызове:

AX = 1130h

BH – вид запрашиваемых данных:

= 00h – указатель из адреса 0000:007Ch таблицы прерываний
= 01h – указатель из адреса 0000:010Ch таблицы прерываний
= 02h – шрифт 8x14 из области BIOS (CP437)
= 03h – шрифт 8x8 из области BIOS, знаки 00h – 7Fh (CP437)
= 04h – шрифт 8x8 из области BIOS, знаки 80h – FFh (CP437)
= 05h – шрифт 9x14 из области BIOS (CP437)
= 06h – стандартный шрифт 8x16 из области BIOS (CP437)
= 07h – шрифт 9x16 из области BIOS (CP437)

При возврате:

ES:BP – указатель на первый байт таблицы запрошенного шрифта

CX – число байтов на знак в используемом шрифте
DL – номер последнего ряда знаков в используемом шрифте

Примечание 1: данные в регистрах CX и DL относятся к шрифту, используемому для индикации в данный момент (а не к запрашиваемому шрифту).

8.01-33 INT 10\AH=13h – запись строки знаков

Обработчик прерывания INT 10\AH=13h выводит на экран строку знаков, которая может быть представлена либо просто последовательностью байтов кода ASCII, либо так, как заполняется экранная память в текстовых режимах, то есть последовательностью с перемежением, в которой за каждым байтом знака следует байт цвета (A.10-5). В последнем случае содержимое BL игнорируется.

При вызове:
AH = 13h
AL – биты 7 – 2 сбросить в нуль
– бит 1 – строка с перемежением байтов знака и байтов цвета
– бит 0 – перемещать курсор по ходу записи строки.
BH – номер экранной страницы (примечания 1 и 2 к 8.01-14)
BL – байт цвета (A.10-5), если строка содержит только знаки
CX – число знаков в строке, которую надо записать
DH – ряд знаков, в котором надо начать запись
DL – колонка, с которой надо начать запись
ES:BP – указатель на первый байт строки, которую надо записать

Примечание 1: служебные коды ASCII, перечисленные в разделе A.02-08, не индицируются, а исполняются как команды.
Примечание 2: в некоторых моделях старых видеокарт возврат (Backspace) и перевод строки (CR) правильно исполняются только при записи на ту экранную страницу, которая индицируется в данный момент.

8.01-34 INT 10\AX=1B00h – информация о статусе видеосистемы

При вызове:
AX = 1B00h
BX = 0000h
ES:DI – указатель на место для записи 64-байтового блока данных
При возврате:
AL = 1Bh, если BIOS поддерживает данную функцию
ES:DI – указатель на первый байт записанного блока данных, содержание которого показано в приложении A.10-2.

8.01-35 INT 10\AX=4F00h – информация о VBE-расширениях BIOS

Обработчик прерывания INT 10\AX=4F00h вызывают для того, чтобы выяснить, имеются ли в данном компьютере VBE-расширения BIOS, и какие видеорежимы SVGA этими расширениями поддерживаются. Если VBE-расширения отсутствуют, то все функции INT 10\AX=4Fxxh на данном компьютере не поддерживаются.

При вызове:

AX = 4F00h
ES:DI – указатель на место для записи таблицы длиной 512 байт

При возврате:

AL = 4Fh – если значение другое, то VBE-расширения отсутствуют
AH = 00h – успешное завершение, таблица записана
= 01h – таблицу не удалось записать
= 02h – функция не поддерживается аппаратурой компьютера
= 03h – в установленном видеорежиме функция не работает
ES:DI – указатель на первый байт таблицы SVGA BIOS (A.10-4).

8.01-36 INT 10\AX=4F01h – информация о видеорежиме SVGA

При вызове:

AX = 4F01h
CX – номер видеорежима SVGA
ES:DI – указатель на место для записи таблицы длиной 256 байт

При возврате:

AL = 4Fh – если значение другое, то функция не поддерживается
AH = 00h – успешное завершение, таблица записана
= 01h – миссия не исполнена, таблицу не удалось записать
ES:DI – указатель на первый байт записанной таблицы (A.10-7).

8.01-37 INT 10\AX=4F02h-4F03h – задать или выяснить SVGA-видеорежим

При вызове:

AX – цель вызова:
= 4F02h – установить новый видеорежим
= 4F03h – получить сведения о видеорежиме
BX – только для AX = 4F02h: SVGA-видеорежим (A.10-1), включая
бит 14: – задействовать линейный кадровый буфер
бит 15: – не обнулять видеопамять

При возврате:

AL = 4Fh – если значение другое, то функция не поддерживается
AH – код завершения, как для 8.01-36

BX – номер установленного видеорежима (A.10-1), включая
бит 14: – линейный кадровый буфер задействован
бит 15: – содержимое видеопамяти сохранено

Примечание 1: координированное переключение видеорежима вместе со сменой параметров управления манипулятором "мышь" обеспечивается вызовом драйвера "мыши" через прерывание INT 33\AX=0028h.

Примечание 2: для некоторых дисплеев приемлемы не все видеорежимы SVGA. Необходимо заранее знать, будет ли Ваш дисплей работать в том режиме, который Вы собираетесь установить.

Примечание 3: переключение видеорежимов сопровождается гашением экрана дисплея на 1 – 2 секунды, дискомфортным для зрительного восприятия. Потому лишних переключений желательно избегать.

8.01-38 INT 10\AX=4F04h – сохранить или восстановить SVGA-видеорежим

При вызове:

AX = 4F04h
CX – записываемая часть конфигурации:
= 0001h – состояние аппаратуры: установлен бит 0
= 0002h – видеоданные BIOS: установлен бит 1
= 0004h – цветностные регистры: установлен бит 2
= 0008h – состояние SVGA: установлен бит 3
= 000Fh – полная конфигурация: установлены биты 0 – 3
DL – подфункция:
= 00h – определить размер буфера для записи состояния
= 01h – сохранить состояние SVGA-видеоконтроллера
= 02h – восстановить состояние SVGA-видеоконтроллера
ES:BX – указатель на начало буфера в памяти (только для подфункций 01h и 02h, причем для 02h он должен быть заполнен данными)

При возврате из подфункции DL = 00h:

BX – требуемое число 64-байтовых блоков памяти

При возврате из подфункции DL = 01h:

ES:BX – указатель на начало буфера с данными видеорежима

8.01-39 INT 10\AX=4F05h – управление окнами видеопамяти

В старых компьютерах для видеопамяти была предоставлена часть адресного пространства в пределах A000:0000h – B000:FFFFh. Расположения задействованных адресных зон в этой области для EGA-совместимых и VGA-совместимых видеорежимов приведены в таблице A.10-1. Однако встроенная видеопамять современных видеокарт рассчитана на обеспечение видеорежимов

SVGA и потому имеет гораздо больший объем, для которого выделенная часть адресного пространства слишком узка. Чтобы преодолеть проблему ограниченности адресного пространства, SVGA BIOS видеокарт организует в предоставленном диапазоне адресов скользящие "окна", через которые открывается доступ к запрашиваемым участкам обширной видеопамяти.

Количество, размер и расположение "окон" в адресуемой процессором памяти зависят и от видеокарты, и от избранного видеорежима. Конкретные данные надо выяснять по таблице A.10-7, получаемой посредством вызова INT 10\AX=4F01h (8.01-36), но обычно организованы одно или два 64-килобайтных "окна": "окно A" A000:0000h – A000:FFFFh и "окно B" B000:0000h – B000:FFFFh. Обработчик прерывания INT 10\AX=4F05h сообщает расположение участка видеопамяти, открытого в запрашиваемом "окне", и позволяет это расположение изменять.

При вызове:

AX = 4F05h
BH = 00h – задать положение участка доступа в видеопамяти
= 01h – выяснить положение участка доступа в видеопамяти
BL = 00h – запрос по отношению к окну A
= 01h – запрос по отношению к окну B
DX – положение участка доступа (только при BH = 00h)

При возврате:

AL = 4Fh – если значение другое, то функция не поддерживается
AH – код завершения, как для 8.01-36
DX – положение участка доступа (только после BH = 01h)

Примечание 1: положение участка доступа в видеопамяти выражается числом шагов адресации, причем размер шага не фиксирован, он дается в слове со смещением 04h в таблице A.10-7, получаемой посредством вызова INT 10\AX=4F01h (8.01-36).

Примечание 2: вызвать обработчик прерывания INT 10\AX=4F05h можно также командой CALL FAR (7.03-08) с адресом, приведенным в двойном слове со смещением 0Ch в таблице A.10-7.

Примечание 3: интерпретация данных, посылаемых в адрес отображаемого "окна", зависит от типа модели, выражаемого байтом 1Bh в таблице A.10-7, причем некоторые модели организации видеопамяти допускают несколько вариантов. В частности, графическая EGA-модель предоставляет 3 варианта интерпретации посылаемого байта:

– режим 00h стирания или записи с учетом битовой маски и маски цвета (байт представляет состояния 8 пикселов);
– режим 01h копирования из одного места видеопамяти в другое (в командах учитываются только адреса);
– режим 02h "заливки" 8 последовательных пикселов (4 младших бита посылаемого байта выражают цвет).

Способы задания режима интерпретации и посылки байтов масок перечислены в примечаниях 3 и 4 к таблице А.14-1.

8.01-40 INT 10\AX=4F06h – логическая длина строки

Обработчик прерывания INT 10\AX=4F06h позволяет установить длину строки кратной степени двойки, что существенно упрощает и ускоряет пересчет координат ценой приемлемого сокращения используемой емкости видеопамяти. Данная функция действует как в графических, так и в текстовых видеорежимах.

При вызове:

AX = 4F06h
BL = 00h – задать фактическую длину строк в пикселах
= 01h – выяснить фактическую длину строк
= 02h – задать фактическую длину строк в байтах
= 03h – выяснить максимальную длину строк
CX – задаваемая длина строк (только при BL = 00h и BL = 02h)

При возврате:

AL = 4Fh – если значение другое, то функция не поддерживается
AH – код завершения, как для 8.01-36
BX – запрошенная длина строк в байтах
CX – запрошенная длина строк в пикселах
DX – максимальное число строк при запрошенной длине строки

Примечание 1: длина сроки может превышать номинальное значение для данного видеорежима, но не может превышать максимального значения. Если задаваемая длина меньше номинальной, то будет установлено ближайшее приемлемое значение.

8.01-41 INT 10\AX=4F07h – начало отображаемого участка видеопамяти

Обработчик прерывания INT 10\AX=4F07h позволяет осуществлять прокрутку изображения на экране, а также переход на другую экранную страницу в пределах имеющейся видеопамяти. Данная функция действует как в графических, так и в текстовых видеорежимах.

При вызове:

AX = 4F07h
BX = 0000h – сместить положение начала в момент вызова
= 0001h – выяснить положение начала отображаемого участка
= 0080h – сместить положение начала в обратном ходе по кадру
CX – номер левого пиксела в строке (при BX = 0000h и 0080h)
DX – номер первой строки (при BX = 0000h и 0080h)

При возврате:

AL = 4Fh – если значение другое, то функция не поддерживается
AH – код завершения, как для 8.01-36
CX – номер левого пиксела в строке (только после BX = 0001h)
DX – номер первой строки (только после BX = 0001h)

8.01-42 INT 11 – получить перечень имеющихся устройств

При вызове ничего готовить не нужно
При возврате:

AX – слово-перечень, расшифровка в приложении A.11-1

Примечание 1: при работе на третьем (низшем) уровне привилегий в защищенном режиме и в режиме V86 процессоры могут вызывать INT 11 в случае размещения операндов по адресам, не кратным числу байтов в формате этих операндов. Такой контроль осуществляется при условии установления бита 12h в регистре флагов, а также бита 12h в управляющем регистре CR0 (примечание 6 к A.11-4). Программы, пользующиеся таким контролем, должны перехватывать вызовы INT 11 в таблице прерываний защищенного режима.

8.01-43 INT 12 –объем обыкновенной памяти до 1 Мб

При вызове ничего готовить не нужно
При возврате:

AX – объем обыкновенной памяти в килобайтах (примечания 2 и 3)

Примечание 1: объем обыкновенной памяти считывается из слова по адресу 0040:0013h в области данных BIOS (приложение A.01-1). Помимо того, объем записан в ячейки 15h и 16h CMOS RAM (примечание 1 к A.14-1).

Примечание 2: для выяснения наличия оперативной памяти сверх 1 Мб следует пользоваться функциями INT 15\AH=88h, INT 15\AX=E801h или INT 15\AX=E820h.

Примечание 3: для выяснения наличия свободной памяти, которая может быть предоставлена в пользование программе по запросу, следует обратиться к INT 21\AH=48h (примечание 1 к 8.02-50).

8.01-44 INT 13\AH=00h\0Dh – сброс контроллера дисковода

При сбросе в начальное состояние контроллер заново заполняет свои регистры данными из таблицы параметров соответствующего дисковода (A.08-2, A.13-1).

Сброс контроллера в начальное состояние необходим после каждого неудачного обращения к дисководу на гибких или жестких магнитных дисках, и только потом попытка обращения может быть повторена.

При вызове:

AH = 00h – обращение к флоппи-дисководу (на гибких дисках)
= 0Dh – обращение к HDD (на жестких магнитных дисках)
DL – номер дисковода (примечание 1)

При неудачном завершении флаг CF установлен, в AH – код завершения (A.06-1). Сброс флага CF означает успешное завершение

Примечание 1: нумерация дисководов на гибких магнитных дисках начинается с нуля: 00h – первый, 01h – второй и так далее, а нумерация дисководов на жестких магнитных дисках – с 80h: 80h – первый, 81h – второй, и так далее. Нумерация дисководов (физических дисков) не связана с буквенными обозначениями логических дисков, которых может быть несколько на каждом дисководе.

Примечание 2: если к одному контроллеру подключены два дисковода, то при сбросе контроллера в начальное состояние происходит перевод головок на нулевую дорожку в обоих дисководах. Когда нужна только рекалибровка дисковода переводом головок на нулевую дорожку без сброса контроллера, тогда нужно вызывать INT 13\AH=11h (все прочие спецификации вызова такие же).

8.01-45 INT 13\AH=01h – статус предыдущего обращения к дисководу

При вызове:

AX = 0100h
DL – номер дисковода (примечание 1 к 8.01-44)

При возврате:

AH – байт статуса, расшифровываемый по таблице A.06-1

Примечание 1: некоторые BIOS возвращают статус предыдущего обращения в AL.

8.01-46 INT 13\AH=02h – считывание секторов в память

При вызове:

AH = 02h
AL – число считываемых секторов (примечание 1)
CH – младшие 8 бит от номера цилиндра (дорожки)
CL – биты 0-5: номер начального сектора от 1 до 63,
– биты 6-7: старшие 2 бита от номера цилиндра (дорожки)
DH – номер головки (примечание 2)

DL – номер дисковода (примечание 1 к 8.01-44)
ES:BX – указатель на буфер для данных

При неудачном завершении флаг CF установлен, в AH – код завершения (A.06-1).
Сброс флага CF означает успешное завершение, и тогда

AL – число считанных секторов (примечание 5)
ES:BX – указатель на буфер со считанными с диска данными.

Примечание 1: задаваемое в AL число секторов не должно быть нулем. Некоторые старые BIOS требуют, чтобы число в AL не превосходило бы числа оставшихся секторов на заказанной дорожке (еще об этом в 4.22).

Примечание 2: старые системы BIOS воспринимают только 4 младших бита в DH, то есть позволяют указать не более 16 головок. Современные системы BIOS (с сигнатурой A0h в блоке данных A.13-1) принимают преобразованные параметры CHS, допуская до 256 головок. В обоих случаях должные предельные значения параметров для конкретного дисковода возвращает обработчик INT 13\AH=08h (8.01-49).

Примечание 3: при возникновении ошибок считывания с флоппи-дисков рекомендуют повторять попытки считывания по крайней мере трижды со сбросом контроллера дисковода в начальное состояние перед каждой новой попыткой.

Примечание 4: при работе в операционной системе Windows прямой доступ к дискам посредством INT 13 возможен только если диск закрыт для конкурентного доступа операцией LOCK (примечание 2 к INT 21\AX=440Dh).

Примечание 5: некоторые старые BIOS не возвращают число в AL. Современные системы BIOS при возникновении ошибок считывания (код = 11h) возвращают в AL длину скорректированного пакета данных.

Примечание 6: INT 13\AH=0Ah (с теми же спецификациями вызова) осуществляет считывание секторов на жестких магнитных дисках вместе с 22-байтовым продолжением, содержащим от 4 до 7 байтов кода для коррекции ошибок. Обработчик прерывания INT 13\AH=0Ah не исправляет ошибок и прекращает процесс считывания сразу же после обнаружения ошибки в последнем считанном секторе.

8.01-47 INT 13\AH=03h – запись данных в сектора диска

При вызове:

AH = 03h
AL – ненулевое число секторов, которые надлежит записать
CH, CL, DH, DL – то же, что для INT 13\AH=02h (8.01-46)
ES:BX – указатель на буфер с подготовленными данными для записи

При неудачном завершении флаг CF установлен, в AH – код завершения (A.06-1).

Сброс флага CF означает успешное завершение, и тогда
AL – число записанных секторов

Примечание 1: все, что сказано в примечаниях 1 – 4 к 8.01-46, действительно также и для INT 13\AH=03h.
Примечание 2: после записи сигналограмму можно сверить с данными в буфере с помощью INT 13\AH=04h (прочие спецификации вызова такие же).
Примечание 3: запись секторов вместе с кодом коррекции ошибок осуществляется с помощью INT 13\AH=0Bh (прочие спецификации вызова такие же).

8.01-48 INT 13\AH=05h – низкоуровневое форматирование дорожки

Низкоуровневому форматированию можно подвергать только такие диски, которые не имеют постоянной разметки дорожек. Поэтому INT 13\AH=05h можно применять по отношению к гибким магнитным дискам, но нельзя применять по отношению к современным жестким магнитным дискам: их заводская разметка может быть необратимо повреждена.

Перед проведением низкоуровневого форматирования необходимо подготовить в памяти компьютера таблицы дополнительных данных с помощью INT 10\AH=18h (8.01-54), а для дисководов устаревших типов – с помощью INT 10\AH=17h. В частности, параметры форматирования дискет считываются из таблицы, адрес которой записан в ячейку 0000:0078h (A.08-2).

При вызове:
AH = 05h
AL – число секторов, которые надлежит форматировать
CH – номер дорожки
DH – номер головки
DL – номер дисковода (примечание 1 к 8.01-44)
ES:BX – указатель на буфер, содержащий по 4 байта данных на каждый форматируемый сектор: 1-й байт – номер дорожки, 2-й байт – номер головки, 3-й байт – номер сектора, 4-й байт – размер сектора (значения (00h, 01h, 02h, 03h соответствуют размерам сектора 128, 256, 512, 1024 байт)
При неудачном завершении флаг CF установлен, в AH – код завершения (A.06-1).
Сброс флага CF означает успешное завершение.

Примечание 1: в блоке данных, на который указывает ES:DX, счет дорожек и головок начинается с нуля, а счет секторов – с единицы.
Примечание 2: конечные значения номеров секторов, головок и дорожек следует определять посредством INT 13\AH=08h (8.01-49). Эти значения могут отличаться от реальных, но они всегда соответствуют тем

преобразованиям (A.13-1), которые выполняет над ними система BIOS данного компьютера.

8.01-49 INT 13\AH=08h – определение параметров дисковода

При вызове:

AH = 08h
DL – номер дисковода (примечание 1 к 8.01-44)

При неудачном завершении флаг CF установлен, в AH – код завершения (A.06-1).
Сброс флага CF означает успешное завершение, и тогда

BL – тип дисковода, только для дисководов на сменных дисках:
= 01h – дисковод для дискет емкостью 360 кбайт,
= 02h – дисковод для дискет емкостью 1.2 Мбайт,
= 03h – дисковод для дискет емкостью 720 кбайт,
= 04h – дисковод для дискет емкостью 1.44 Мбайт,
= 06h – дисковод для дискет емкостью 2.88 Мбайт,
= 10h – какой-либо иной дисковод.
CH – младшие 8 бит максимального номера цилиндра (дорожки)
CL – биты 0-5: номер последнего сектора на дорожке,
– биты 6-7: старшие 2 бита от максимального номера цилиндра
DH – максимальный номер головки
DL – число подобных дисководов в компьютере (примечание 2)
ES:DI – адрес таблицы параметров (A.08-2), выдаваемый только для дисководов на гибких магнитных дисках.

Примечание 1: статус успешного завершения AH=00h может быть возвращен даже если запрошенный дисковод не существует. Чтобы убедиться в достоверности полученных данных, необходимо сначала проверить флаг CF или число, возвращенное в регистре DL.

Примечание 2: некоторые системы BIOS не возвращают в регистре DL число больше чем два. Подозрение на наличие большего числа дисководов следует проверять отдельно с помощью INT 13\AH=15h (8.01-52).

Примечание 3: сигнатура A0h в байте со смещением 03h таблицы A.13-1 означает, что для накопителей на жестких магнитных дисках обработчик прерывания INT 13\AH=08h выдает не реальные, а преобразованные значения параметров CHS. В таком случае при вызовах всех старых функций прерывания INT 13 (8.01-46 – 8.01-54) следует руководствоваться именно преобразованными, а не реальными значениями параметров CHS.

8.01-50 INT 13\AH=0Ch – смещение головки дисковода на заданную дорожку

При вызове:

AH = 0Ch

CH, CL, DH, DL – то же, что для INT 13\AH=02h (8.01-46)

При неудачном завершении флаг CF установлен, в AH – код завершения (A.06-1).
Сброс флага CF означает успешное завершение.

8.01-51 INT 13\AH=10h – проверка готовности жесткого магнитного диска

Байт статуса, возвращаемый после вызова INT 13\AH=10h, показывает, существует ли запрошенный дисковод и готов ли он принять к исполнению очередное задание.

При вызове:

AH = 10h

DL – номер дисковода на жестких магнитных дисках
(80h – первый дисковод, 81h – второй, и так далее)

При возврате:

AH – байт статуса, расшифровываемый по таблице A.06-1.

Сброс флага CF означает успешное завершение.

8.01-52 INT 13\AH=15h – проверка типа дисковода

При вызове:

AH = 15h

DL – номер дисковода (примечание 1 к 8.01-44)

При неудачном завершении флаг CF установлен, в AH – код завершения (A.06-1).
Если флаг CF сброшен, то AH – код типа дисковода:

= 00h – запрошенный дисковод не существует
= 01h – флоппи-дисковод без датчика смены диска
= 02h – любой дисковод с датчиком смены диска
= 03h – дисковод на жестких магнитных дисках

Если код дисковода AH=03h, то в регистрах CX:DX возвращается четырехбайтовое число 512-байтовых секторов на этом диске.

Примечание 1: дисководы на сменных жестких магнитных дисках могут быть причислены к типу 01h или 02h.

Примечание 2: обработчик прерывания INT 13\AH=15h не полагается на хранящиеся в памяти сведения, он запрашивает контроллер дисковода заново.

8.01-53 INT 13\AH=16h – обнаружение смены диска

При вызове:

AH = 16h

DL – номер дисковода (примечание 1 к 8.01-44)

При возврате:

если диск не был сменен, то флаг CF сброшен и AH = 00h

если флаг CF установлен и AH = 06h, то диск был сменен

если флаг CF установлен, но значение в AH другое, то это значение – код завершения из таблицы A.06-1.

Примечание 1: перед первой засылкой запроса INT 13\AH=16h на дисковод неизвестного типа нужно заранее с помощью INT 13\AH=15h убедиться в том, что этот дисковод имеет датчик смены дисков.

Примечание 2: датчик смены дисков обычно реагирует на открывание или закрывание заслонки слота в дисководе, так что может сообщать о смене диска, которая в самом деле не произошла.

Примечание 3: о каждом факте смены диска датчик "докладывает" только один раз, потому что он сбрасывается в исходное состояние после каждого вызова INT 13\AH=16h.

Примечание 4: современные системы BIOS позволяют регистрировать смену дисков посредством INT 13\AH=49h с теми же (кроме AH) спецификациями, но с более широким кругом опрашиваемых дисководов, включающим любые дисководы с номерами 80h и выше. В том, что эта функция в Вашем компьютере поддерживается, нужно заранее убедиться с помощью INT 13\AH=41h (8.01-55).

8.01-54 INT 13\AH=18h – задание параметров форматирования для дискет

При вызове:

AH = 18h

CH, CL, DH, DL – то же, что для INT 13\AH=02h (8.01-46)

При возврате:

AH = 00h – форматирование с заданными параметрами возможно

= 01h – функция недоступна

= 0Ch – тип диска не поддерживается или неизвестен

= 80h – сменный диск в дисководе отсутствует

ES:DI – указатель на таблицу параметров A.08-2

Примечание 1: обработчик прерывания INT 13\AH=18h не записывает возвращаемый указатель на таблицу параметров в адрес 0000:0078h (вектор INT 1E, A.12-1), эту операцию надо делать в вызывающей программе.

Примечание 2: при попытках задавать параметры форматирования для накопителей на жестких магнитных дисках обработчик прерывания INT 13\AH=18h устанавливает флаг CF и выдает код статуса AH=01h. Параметры форматирования для устаревших 5.25" дискет и для дискет емкостью 720 кб следует задавать посредством INT 13\AH=17h.

8.01-55 INT 13\AH=41h – проверка поддержки дополнительных функций

Поддержка дополнительных функций прерывания INT 13 получила широкое распространение с 1997 года. Благодаря этой поддержке реализованы свойства, которые уже стали привычными в современных компьютерах, в частности, возможность загрузки с оптических дисков и адресация LBA (примечание 4 к A.13-6) для доступа к дискам большой емкости.

При вызове:

AH = 41h
BX = 55AAh
DL – номер дисковода (примечание 1 к 8.01-44)

При неудаче флаг CF установлен, в AH – код завершения (A.06-1), причем
AH = 01h означает, что функция не поддерживается.

При успешном завершении флаг CF сброшен, и тогда
AH = 01h – версии 1.x дополнительных функций INT 13;
= 20h – версия 2.0 дополнительных функций INT 13;
= 21h – версия 2.1 дополнительных функций INT 13.
BX = AA55h означает, что BIOS включает расширения INT 13
CX – признаки поддержки следующих функций:
бит 0 – INT 13\AH=42h-44h,47h,48h;
бит 1 – INT 13\AH=45h,46h,48h,49h, INT 15\AH=52h;
бит 2 – INT 13\AH=48h,4Eh;
бит 3 – поддержка расширенного пакета (примечание 2).

Примечание 1: при возврате данные в регистрах AL и DH могут быть утрачены.
Примечание 2: все расширения INT 13 поддерживают запросы с 10-байтовыми пакетами дисковой адресации (с 4-байтовыми адресами для дисков емкостью до 127 Гбайт), а поддержка расширенного пакета означает возможность пользования 20-байтовыми пакетами с 8-байтовыми адресами для дисков емкостью свыше 127 Гбайт. Структура пакетов показана в приложении A.13-4.

8.01-56 INT 13\AH=42h – расширенная функция чтения с диска

При вызове:

AH = 42h
DL – номер дисковода (примечание 1 к 8.01-44)
DS:SI – указатель на пакет дисковой адресации (A.13-4)

При неудачном завершении флаг CF установлен, в AH – код завершения (A.06-1).
Сброс флага CF означает успешное завершение.

Примечание 1: при возврате в адрес 02h в пакете дисковой адресации вписывается количество считанных блоков данных.

Примечание 2: указатель на считанный массив данных представлен 4-байтовым (двойным) словом по адресу 04h в пакете дисковой адресации.

Примечание 3: вызов INT 13\AH=47h с теми же (кроме AH) спецификациями заставляет дисковод начать перемещение головки к заданной дорожке. Процессор успеет многое сделать за 2 – 3 миллисекунды, пока головка движется к цели. Если вызывать INT 13\AH=47h заблаговременно, то считывание не будет сопряжено с ожиданием установления головки, и программа будет выполняться быстрее.

8.01-57 INT 13\AH=43h – расширенная функция записи на диск

При вызове:

AH = 43h
AL = 00h – не исполнять сверку сигналограммы,
= 02h – проводить сверку после записи.
DL – номер дисковода (примечание 1 к 8.01-44)
DS:SI – указатель на пакет дисковой адресации (A.13-4)

При неудачном завершении флаг CF установлен, в AH – код завершения (A.06-1).
Сброс флага CF означает успешное завершение

Примечание 1: указатель на массив подлежащих записи данных должен быть представлен по адресу 04h в пакете дисковой адресации.

Примечание 2: при возврате в пакет дисковой адресации вписывается слово со смещением 02h, означающее количество успешно записанных или успешно сверенных блоков данных.

Примечание 3: версии до 2.0 расширений INT 13 использовали флаг AL = 01h для проведения сверки после записи. Если сверка запрошена, но не поддерживается, то BIOS устанавливает флаг CF и возвращает AH = 01h (= неверная функция).

Примечание 4: сверку сигналограммы можно провести отдельно посредством вызова INT 13\AH=44h с теми же спецификациями, за исключением AH и того, что значение в AL будет проигнорировано.

8.01-58 INT 13\AH=45h – блокировка и разблокировка дисковода

В программах встречаются такие последовательности обращений дисководу, которые нельзя прерывать во избежание потери данных. Типичный пример такой процедуры – дефрагментация. На время выполнения таких процедур дисковод следует блокировать. Пока дисковод блокирован, из него нельзя извлечь сменный диск. В многозадачных операционных системах блокировка исключает обращения к диску со стороны других программ. Допустимы до 255 уровней вложенности вызовов процедур, требующих монопольного "владения" диском. Каждая такая процедура должна завершаться операцией разблокирования. Разблокированному состоянию диска соответствует уровень вложенности 00h.

При вызове:

AX = 4500h – блокировать дисковод: увеличить уровень на 1
= 4501h – разблокировать дисковод: уменьшить уровень на 1
= 4502h – выяснить уровень вложенности блокирования
DL – номер дисковода (примечание 1 к 8.01-44)

При неудачном завершении флаг CF установлен, в AH – код завершения (A.06-1)
При успешном завершении флаг CF сброшен, и тогда

AL – уровень вложенности блокирования.

8.01-59 INT 13\AH=46h – выталкивание сменного диска из дисковода

При вызове:

AX = 4600h
DL – номер дисковода (примечание 1 к 8.01-44)

При неудачном завершении флаг CF установлен, в AH – код завершения (A.06-1).
Если флаг CF сброшен, то миссия завершена успешно

Примечание 1: обработчик прерывания INT 13\AH=46h использует INT 15\AH=52h чтобы проверить, не занят ли данный диск операциями обмена данными с кэш-буфером или обслуживанием других программ.

8.01-60 INT 13\AH=48h – запрос параметров дисковода

При вызове:

AH = 48h
DL – номер дисковода (примечание 1 к 8.01-44)
DS:SI – указатель на буфер для данных (примечание 1)

При неудачном завершении флаг CF установлен, в AH – код завершения (A.06-1).
Если флаг CF сброшен, то миссия завершена успешно, и тогда

DS:SI – указатель на буфер с таблицей данных (A.13-2)

Примечание 1: первое слово в буфере должно объявлять его длину: 001Ah – для расширений INT 13 версий 1.x, 001Eh – для версий 2.x, 0049h – для версий 3.x. Если указана длина буфера для более старой версии, то новые версии расширений "обрежут" возвращаемую таблицу по формату для старой версии.

8.01-61 INT 13\AX=4A00h – эмулировать диск по данным с оптического диска

Эта функция создает виртуальный диск на основании образа реального диска, записанного на оптический диск.

При вызове:

AX = 4A00h

DS:SI – указатель на пакет спецификаций загрузки A.15-1

При неудачном завершении флаг CF установлен, в AH – код завершения (A.06-1).
Если флаг CF сброшен, то миссия завершена успешно

Примечание 1: обработчик прерывания INT 13\AX=4C00h принимает такие же спецификации (кроме AX) и делает то же самое, но сверх того инициирует процесс загрузки операционной системы с созданного виртуального диска.

8.01-62 INT 13\AH=4Bh – подфункции эмуляции дисковода

При вызове:

AH = 4Bh

AL = 00h – прекратить эмуляцию дисковода

= 01h – выдать статус эмуляции дисковода

DL – номер эмулируемого дисковода (примечание 1 к 8.01-44)

DS:SI – указатель на место для пакета данных длиной 13h байт.

При неудачном завершении флаг CF установлен, в AH – код завершения (A.06-1).
Если флаг CF сброшен, то миссия завершена успешно, и тогда

DS:SI – указатель на запрошенный пакет данных (A.15-1)

Примечание 1: подфункция AL = 00h при указании номера дисковода DL = 7Fh прекращает эмуляцию всех виртуальных дисков.

Примечание 2: если эмуляция не осуществляется, то флаг CF сброшен.

8.01-63 INT 13\AH=4Dh – считывание секторов оптического диска

При вызове:

AX = 4D00h

DS:SI – указатель на командный пакет (A.15-2)

При неудачном завершении флаг CF установлен, в AH – код завершения (A.06-1).

Если флаг CF сброшен, то миссия завершена успешно

Примечание 1: возвращаемые данные записываются в подготовленный буфер, адрес которого должен быть указан в командном пакете в двойном слове со смещением 02h (A.15-2).

Примечание 2: обычно с помощью INT 13\AH=4Dh считывают загрузочный каталог, структура которого показана в таблице A.15-3.

8.01-64 INT 13\AH=4Eh – конфигурирование дисковода

При вызове:

AH = 4Eh
AL = 00h – вести упреждающее считывание в буфер дисковода
= 01h – запретить упреждающее считывание в буфер
= 02h – задать режим PIO с максимальной скоростью передачи
= 03h – задать нулевой режим PIO
= 04h – задать режим PIO, принимаемый по умолчанию
= 05h – разрешить режим DMA с максимальным номером
= 06h – запретить использование режима DMA
DL – номер дисковода (примечание 1 к 8.01-44)

При неудачном завершении флаг CF установлен, в AH – код завершения (A.06-1).
Если флаг CF сброшен, то миссия завершена успешно, и тогда

AL = 00h – команда изменила режим только указанного дисковода
= 01h – команда повлияла на режим работы других устройств, подключенных к тому же контроллеру.

Примечание 1: передача данных с прямым доступом в память (DMA) и режим программного управления вводом-выводом (PIO) взаимно несовместимы. Задание режима DMA означает отказ от программного управления PIO, выбор PIO означает отказ от DMA.

8.01-65 INT 14\AH=00h – параметры последовательного порта

При вызове:

AH = 00h
AL – биты 7-5: значения от 000 до 111 задают скорости передачи 110, 150, 300, 600, 1200, 2400, 4800, 9600 бит/с.
– биты 4-3: значения 00 или 10 – проверка не производится, 01 – проверка по нечетности, 11 – проверка по четности.
– бит 2: если сброшен – 1 стоповый бит, установлен – 2 бита.
– биты 1-0: 00 означает 5 битов в слове, 01 – 6 битов в слове, 10 – 7 битов в слове, 11 – 8 битов в слове.
DX – номер последовательного порта (0000h – 0003h)

При возврате:

AH – статус линии (A.14-2)
AL – статус модема, если он подключен к запрошенному порту.

8.01-66 INT 14\AH=01h – посылка байта в последовательный порт

При вызове:

AH = 01h
AL – посылаемый байт
DX – номер последовательного порта (0000h – 0003h)

При возврате:

AH – статус линии (A.14-2)

Примечание 1: ошибка передачи обнаруживается по установленному биту 7 в байте статуса (превышение времени ожидания отклика).

8.01-67 INT 14\AH=02h – прием байта данных через последовательный порт

При вызове:

AX = 0200h
DX – номер последовательного порта (0000h – 0003h)

При возврате:

AH – статус линии (A.14-2)
AL – принятый байт, если сброшен бит 7 в регистре AH

8.01-68 INT 14\AH=03h – выяснение статуса последовательного порта

При вызове:

AX = 0300h
DX – номер последовательного порта (0000h – 0003h)

При возврате:

AH – статус линии (A.14-2)
AL – статус модема, если он подключен к запрошенному порту.

8.01-69 INT 15\AH=52h – выяснение занятости дисковода

Функция INT 15\AH=52h показывает, занят ли дисковод в данный момент обменом данными или нет. Ее вызывает, в частности, обработчик прерывания INT 13\AH=46h, чтобы определить, можно ли вытолкнуть сменный диск.

При вызове:

AH = 52h
DL – номер дисковода (примечание 1 к 8.01-44)

При возврате в регистре AH – код завершения из таблицы A.06-1,
если флаг CF сброшен, то дисковод свободен,
если флаг CF установлен, то дисковод занят.

Примечание 1: прежде чем вызывать INT 15\AH=52h, надо с помощью INT 13\AH=41h определить, поддерживается ли эта функция в данном компьютере.

Примечание 2: по умолчанию BIOS устанавливает "заглушку", которая на вызовы INT 15\AH=52h всегда отвечает CF=0. Вызовы INT 15\AH=52h перехватывает программа кэширования, и только тогда ответ будет отражать реальное состояние процесса обмена данными.

8.01-70 INT 15\AX=5301h – активизация управления электропитанием.

В момент включения компьютера программное управление электропитанием (APM) не активизируется автоматически. При работе компьютера в реальном режиме для активизации управления электропитанием необходим вызов обработчика прерывания INT 15\AX=5301h.
При вызове:
AX = 5301h
BX = 0000h (идентификатор APM-расширения системы BIOS).
При неудаче флаг CF установлен, в AH – код завершения (A.06-1).
Если флаг CF сброшен, то миссия завершена успешно.

Примечание 1: обработчик прерывания INT 15\AX=5301h заставляет систему APM эмулировать спецификации APM версии 1.0. Если нужно вызывать операции, не определенные в версии 1.0, то их следует заранее разблокировать с помощью INT 15\AX=530Eh (8.01-72).

8.01-71 INT 15\AX=5307h – переключение режимов электропитания.

Программное выключение наиболее распространенных компьютеров конструктивного исполнения ATX выполняется путем перевода их блока электропитания в дежурный режим. При этом электропитание продолжает подаваться только на те узлы, которые обеспечивают возможность автоматического включения компьютера. Разумеется, программное выключение невозможно без аппаратной поддержки со стороны блока электропитания и материнской платы.
При вызове:
AX = 5307h
BX = 0001h (идентификатор всех APM-управляемых устройств)
CX = 0003h (код запроса программного выключения)
При неудаче флаг CF установлен, в AH – код завершения (A.06-1),

а при удачном завершении все остальное не имеет значения.

Примечание 1: операция выключения определена в спецификации APM версии 1.2. Если система управления электропитанием эмулирует операции версии 1.0 (примечание 1 к 8.01-70), то возможность программного выключения электропитания необходимо сначала разблокировать с помощью INT 15\AX=530Eh (8.01-72).

Примечание 2: в устаревших компьютерах конструктивного исполнения AT вызов данной операции допустим, но игнорируется.

8.01-72 INT 15\AX=530Eh – запрос версии управления электропитанием.

Чтобы обеспечить совместимость с имеющимися операционными системами, в спецификации управления электропитанием (APM) заложена возможность эмуляции более новыми версиями APM BIOS тех совокупностей операций, которые были способны исполнять старые версии APM. Программа, которой предстоит управлять электропитанием, должна заранее запросить у APM BIOS эмуляцию желаемой версии APM, а потом принять к исполнению возвращаемое APM BIOS сообщение о том, какую ближайшую версию APM данная APM BIOS способна воспроизвести.

При вызове:

AX = 530Eh
BX = 0000h (идентификатор APM-расширения системы BIOS)
CX – версия APM, эмуляция которой запрашивается (примечание 1)

При возврате:

при неудаче флаг CF установлен, в AH – код завершения (A.06-1),
если флаг CF сброшен, то в AX – версия APM, которой будет соответствовать предоставляемое множество операций.

Примечание 1: в регистре CX целую и дробную части номера версии APM надо записывать в разные байты. В частности, для запроса версии 1.2 надо указать CX=0102h. В таком же формате APM BIOS возвратит номер фактически эмулируемой версии в регистре AX.

Примечание 2: данная операция определена начиная со спецификации APM версии 1.1. Если возвращен код завершения AH=80h или AH=86h, а вызов INT 15\AX=5301h (8.01-70) завершился как-либо иначе, значит, BIOS данного компьютера реализует только APM версии 1.0.

8.01-73 INT 15\AH=83h,86h – управление счетом времени в таймере BIOS

При вызове:

AX = 8300h – запустить счет и отметить его окончание изменением состояния маркерного байта (примечание 3)

AX = 8600h – запустить счет и остановить вызывающий процесс до окончания счета
CX – старшие 16 бит 32-битовой задержки в микросекундах
DX – младшие 16 бит 32-битовой задержки в микросекундах
ES:BX– указатель на маркерный байт (только для AX = 8300h)

При возврате:

флаг CF сброшен при успешном завершении
флаг CF установлен при ошибке или запуске, когда счет уже идет
AH – код завершения из таблицы A.06-1.

Примечание 1: вероятный шаг счета времени – 977 микросекунд.
Примечание 2: чтобы прервать уже начатый счет, надо вызвать INT 15\AH=8301h.
Примечание 3: обычно маркерный байт – это байт 0040:00A0h (A.12-1). Некоторые системы BIOS по завершении счета устанавливают в единицу его старший бит, другие записывают в маркерный байт число 80h.
Примечание 4: из "окна DOS" системы Windows таймер BIOS недоступен.

8.01-74 INT 15\AH=84h – считывание состояния джойстика.

При вызове:

AH = 84h
DX = 0000h – считать состояния переключателей
= 0001h – считать сигналы положения манипуляторов

При неудачном завершении флаг CF установлен, в AH – код завершения (A.06-1).
Если после исполнения подфункции DX=0000h флаг CF сброшен, то состояния переключателей возвращаются в битах 4 – 7 регистра AL.
Если после исполнения подфункции DX=0001h флаг CF сброшен, то

AX – значение x-координаты 1-го манипулятора
BX – значение y-координаты 1-го манипулятора
CX – значение x-координаты 2-го манипулятора
DX – значение y-координаты 2-го манипулятора

Примечание 1: при типичном сопротивлении датчиков 250 кОм возвращаемые значения координат обычно находятся в пределах 0000h – 01A0h.
Примечание 2: если игровой порт в компьютере отсутствует, то подфункция DX=0001h возвращает нулевые значения координат, а подфункция DX=0000h возвращает AL=00h, что соответствует разомкнутому состоянию переключателей.

8.01-75 INT 15\AH=85h – вызов по нажатию клавиши SysRq

Когда пользователь нажимает клавишу SysRq, обработчик прерывания INT 09 вызывает INT 15\AH=85h. Для INT 15\AH=85h обработчик прерывания должен быть поставлен драйвером видеокарты с тем, чтобы обеспечить распечатку выводимого на экран изображения в том формате, в котором работает видеокарта. Но если этот обработчик не поставлен, то сработает "заглушка", которая заранее ставится системой BIOS: она просто возвратит управление вызывающей программе так, как если бы вызов был удовлетворен.

При вызове:

AH = 85h
AL = 00h – вызов при нажатии на клавишу SysRq
= 01h – вызов при отпускании клавиши SysRq
флаг CF при вызове должен быть сброшен

Если вызов удовлетворен, то флаг CF сброшен и AH = 00h.
При неудаче флаг CF установлен, в AH – код завершения (A.06-1).

8.01-76 INT 15\AH=87h – копирование с доступом в расширенную память

Обработчик прерывания INT 15\AH=87h осуществляет копирование блоков данных размером до 64 кбайт в пределах линейно адресуемой памяти до 16 Мбайт. Копирование осуществляется в защищенном режиме при блокированных аппаратных прерываниях, так что готовить таблицу прерываний перед вызовом INT 15\AH=87h не нужно. Тем не менее для копирования посредством INT 15\AH=87h нужно заранее подготовить таблицу GDT из шести дескрипторов согласно показанному здесь шаблону:

Резервируемый дескриптор:	`00 00 00 00 00 00 00 00`
Резервируемый дескриптор:	`00 00 00 00 00 00 00 00`
Дескриптор сегмента-источника:	`ss ss aa aa aa 93 00 00`
Дескриптор сегмента назначения:	`ss ss dd dd dd 93 00 00`
Резервируемый дескриптор:	`00 00 00 00 00 00 00 00`
Резервируемый дескриптор:	`00 00 00 00 00 00 00 00`

В показанном шаблоне знаком "a" отмечено поле линейного адреса сегмента источника, знаком "d" – поле линейного адреса сегмента назначения, а знаком "s" – поля размера сегментов. Поскольку размер копируемого участка памяти задается в регистре CX количеством копируемых двухбайтовых слов, постольку размеры сегментов, выражаемые в байтах, должны быть не меньше, чем (2•CX)–1. И в полях размеров, и в полях адресов первыми следуют младшие байты, а последними – старшие. Байты атрибутов и в дескрипторе-источнике, и в дескрипторе

назначения должны быть равны 93h. Более детальные пояснения структуры дескрипторов даны в приложении A.12-2.

Резервируемые дескрипторы, заполненные нулями, обработчик прерывания INT 15\AH=87h задействует потом для обслуживания операций защищенного режима. По завершении копирования процессор будет переведен обратно в реальный режим, и будет восстановлено состояние, обеспечивающее продолжение исполнения вызывающей программы.

При вызове:

AH = 87h
CX – число копируемых слов, не более 7FFFh
ES:SI – указатель на подготовленную таблицу GDT.

При неудачном завершении флаг CF установлен, в AH – код завершения (A.06-1).
Если флаг CF сброшен, то миссия завершена успешно.

Примечание 1: из-за исполнения копирования при блокированных аппаратных прерываниях пользование INT 15\AH=87h может привести к пропуску вызовов от внешних устройств. Копирование с доступом в расширенную память посредством драйвера HIMEM.SYS (A.12-4) этим недостатком не обладает.

Примечание 2: драйвер HIMEM.SYS перехватывает прерывание INT 15\AH=87h и не дает прикладным программам доступ к обработчику, который установлен системой BIOS. Тем не менее драйвер HIMEM.SYS позволит пользоваться прерыванием INT 15\AH=87h для доступа к ограниченному участку расширенной памяти, если этот участок зарезервирован заранее указанием параметра /INT15 (5.04-01).

8.01-77 INT 15\AH=88h – размер расширенной памяти до 16 Мбайт

При вызове:

AH = 88h

При неудачном завершении флаг CF установлен, в AH – код завершения (A.06-1).
Если флаг CF сброшен, то миссия завершена успешно, и тогда

AX – размер памяти в килобайтах, имеющейся за пределом 100000h, то есть сверх 1 Мегабайта.

Примечание 1: если адресное пространство сверх 1 Мегабайта не является непрерывным, то будет показан размер до первого промежутка.

Примечание 2: обработчик прерывания INT 15\AH=88h считывает сведения о размере памяти из байтов 30h и 31h CMOS-памяти системы BIOS.

Примечание 3: вызовы обработчика прерывания INT 15\AH=88h, устанавливаемого системой BIOS, перехватываются драйверами HIMEM.SYS и EMM386.EXE.

Примечание 4: для выяснения размера памяти сверх 16 Мбайт следует использовать INT 15\AX=E801h\E881h (8.01-79) или INT 15\AX=E820h (8.01-80).

8.01-78 INT 15\AH=89h – переключение процессора в защищенный режим.

Обработчик прерывания INT 15\AH=89h осуществляет процесс перехода в защищенный режим, включая операции занесения селекторов сегментов в сегментные регистры и перепрограммирования контроллеров прерываний для работы с таблицей прерываний защищенного режима.

Чтобы исполнение вызывающей программы продолжилось после переключения процессора в защищенный режим, нужно заранее подготовить глобальную таблицу дескрипторов (GDT) и таблицу прерываний (IDT) для защищенного режима. Таблица IDT заполняется 8-байтовыми дескрипторами адресов перехода ("вентилями") для вызова обработчиков прерываний защищенного режима. Размер и линейный адрес расположения таблицы IDT записываются в дескриптор сегмента IDT, входящий в состав таблицы GDT.

Таблица GDT, подготавливаемая для обработчика прерывания INT 15\AH=89h, включает восемь 8-байтовых сегментных дескрипторов, причем их расположение должно соответствовать следующему шаблону:

Резервируемый дескриптор:	`00 00 00 00 00 00 00 00`
Дескриптор сегмента GDT:	`3F 00 aa aa aa 00 00 00`
Дескриптор сегмента IDT:	`FF 03 aa aa aa F2 00 00`
Дескриптор сегмента DS:	`ss ss aa aa aa 92 0s aa`
Дескриптор сегмента ES:	`ss ss aa aa aa 92 0s aa`
Дескриптор сегмента SS:	`ss ss aa aa aa 92 0s aa`
Дескриптор сегмента CS:	`ss ss aa aa aa 9A 0s 00`
Резервируемый дескриптор:	`00 00 00 00 00 00 00 00`

В показанном шаблоне знаком "a" отмечены поля линейного адреса каждого сегмента, а знаком "s" – поля размера сегментов. Резервируемые дескрипторы заполняют нулями. В качестве примера в шаблоне показаны конкретные значения байта атрибутов, а также размера для сегментов GDT и IDT. Детальная расшифровка структуры дескрипторов приведена в приложении A.12-2.

При вызове:

AH = 89h
BL – номер прерывания для IRQ 0 (примечание 2)
BH – номер прерывания для IRQ 8 (примечание 2)
ES:SI – указатель на заполненную таблицу GDT

При неудаче флаг CF установлен, в AH – код завершения (A.06-1).

При успешном переходе в защищенный режим флаг CF сброшен, AH = 00h, в сегментные регистры процессора занесены селекторы сегментов из таблицы GDT. Значение в регистре BP может быть изменено.

Примечание 1: чтобы после перехода в защищенный режим исполнение программы продолжилось со следующей команды, дескриптор сегмента CS в составе таблицы GDT должен указывать на тот же сегмент, который определен сегментным адресом в регистре CS при реальном режиме.

Примечание 2: назначаемые номера прерываний в регистрах BL и BH должны быть кратны 8 (младшие 3 бита обнулены). Запросам по линиям IRQ 1 – IRQ 7 будут назначены следующие 7 номеров прерываний вслед за номером в регистре BL, а запросам по линиям IRQ 9 – IRQ F – следующие 7 номеров вслед за номером в регистре BH. При назначении номеров нужно учитывать, что номера до 1Fh могут быть задействованы исключениями, вызываемыми процессором.

Примечание 3: вследствие перепрограммирования контроллеров прерываний и изменения формата адресации после перехода в защищенный режим становится невозможен прямой вызов всех обработчиков прерываний, предназначенных для использования в реальном режиме.

8.01-79 INT 15\AX=E801h,E881h – размер расширенной памяти.

При вызове:

AX = E801h или E881h

При неудачном завершении флаг CF установлен. Если флаг CF сброшен, то

AX – размер расширенной памяти в области между 1 Мбайт и 16 Мбайт (в килобайтах)

BX – размер расширенной памяти в области сверх 16 Мбайт (в 64-килобайтных блоках)

CX – размер конфигурированной памяти в области между 1 Мбайт и 16 Мбайт (в килобайтах)

DX – размер конфигурированной памяти в области сверх 16 Мбайт (в 64-килобайтных блоках)

Примечание 1: если при удачном завершении AX = BX = 0000h, то полный размер расширенной памяти следует считывать из регистров CX и DX.

Примечание 2: вызов INT 15\AX=E881h, в отличие от INT 15\AX=E801h, правильно отображает величины свыше 4 Гбайт, выводя старшие значащие цифры в разрядах 16 – 31 32-разрядных регистров EBX и EDX. Способ доступа к ним из программ реального режима показан в разделе 7.02-06.

Примечание 3: функции INT 15\AX=E801h\E881h не поддерживаются устаревшими версиями BIOS, разработанными до 1995 года.

8.01-80 INT 15\AX=E820h – карта распределения оперативной памяти.

Карта распределения представляет собой последовательность 20-байтных дескрипторов, каждый из которых описывает одну из областей памяти, выделенную с определенной целью системой BIOS. Каждый дескриптор начинается с 8-байтового стартового адреса, далее следует 8-байтовая длина соответствующей области памяти, и кончается дескриптор 4-байтовым кодом назначения области памяти (примечание 1). За один вызов считывается один дескриптор, так что INT 15\AX=E820h приходится вызывать несколько раз. Первый вызов требует установления в регистре EBX нулевого (00000000h) адреса считывания карты памяти, а возвращает ненулевое значение, необходимое для следующего вызова. Окончание цикла вызовов отмечается либо возвратом нулевого значения в регистре EBX, либо установлением флага CF.

При вызове:

AX = E820h
EBX – адрес считывания карты памяти
ECX – размер буфера для дескриптора, не менее 20 байт
EDX = 534D4150h – сигнатура "SMAP"
ES:DI – указатель на буфер для записи дескриптора

При неудачном завершении значение в EAX отлично от сигнатуры 534D4150h, флаг CF установлен, в AH – код завершения (A.06-1).

При успешном завершении EAX = 534D4150h "SMAP", и тогда:

EBX – следующий адрес считывания карты памяти
ECX – фактическая длина записанных в буфер данных
ES:DI – указатель на заполненный буфер

Примечание 1: определены следующие коды назначения областей памяти:

01h – память, предоставляемая операционной системе;
02h – память, резервированная BIOS (системная ROM);
03h – области таблиц ACPI (после считывания свободные);
04h – области энергонезависимой системной памяти ACPI.

Области, характеризуемые другими кодами, следует считать резервированными системой BIOS. Интервалы адресного пространства, не включенные в карту распределения памяти, аппаратно не задействованы примененным комплектом микросхем.

Примечание 2: некоторые системы BIOS требуют, чтобы при вызове старшие разряды 16 – 31 регистра EAX были обнулены (EAX = 0000E820h).

Способ доступа к EAX и другим 32-разрядным регистрам из программ реального режима показан в разделе 7.02-06.

Примечание 3: некоторые системы BIOS игнорируют значение в регистре ECX и всегда возвращают 20 байт данных.

Примечание 4: если вызов INT 15\AX=E820h компьютером не поддерживается, то рекомендуется попытаться вызвать INT 15\AX=E801h (8.01-79), а в случае повторной неудачи – INT 15\AH=88h (8.01-77).

8.01-81 INT 16\AH=03h – темп и задержка повторения для клавиатуры

При вызове:

AX = 0300h – установить значения, принимаемые по умолчанию
= 0305h – установить темп из BL, задержку из BH
= 0306h – вывести действующие значения темпа и задержки
BL – (для AX=0305h): код темпа повторения (примечание 1)
BH – (для AX=0305h): код задержки (примечание 2)

При возврате:

BL – код установленного темпа повторения (примечание 1)
BH – код установленной задержки (примечание 2)
AH – содержимое может быть не сохранено

Примечание 1: допустимые значения кода темпа от 00h до 1Fh соответствуют темпам повторения от 30 раз в секунду до 2 раз в секунду.

Примечание 2: допустимые значения кода задержки от 00h до 03h соответствуют значениям задержки от 0.25 секунды до 1 секунды.

8.01-82 INT 16\AH=05h – введение кода нажатия клавиши в буфер клавиатуры

При вызове:

AH = 05h
CH – скэн-код нажатия клавиши из таблицы A.02-1
CL – код ASCII соответствующего знака из таблицы A.02-1

При возврате:

AL – код завершения из таблицы A.06-1
AH – содержимое может быть не сохранено

Примечание 1: старые версии BIOS не поддерживают эту функцию.

Примечание 2: введение некоторых кодов (например, кода клавиши Enter) в буфер клавиатуры может вызвать исполнение соответствующих функций

Примечание 3: с помощью INT 16\AH=05h нельзя вводить коды "функциональных" клавишей (Shift, Ctrl, Alt и т.п.).

8.01-83 INT 16\AH=10h – получение кода клавиши из буфера клавиатуры

Обработчик прерывания INT 16\AH=10h обслуживает наиболее распространенные модели 101\108-клавишных клавиатур, а также совместимые клавиши на других клавиатурах. Он изымает коды клавиши из буфера клавиатуры и записывает их в регистры AH и AL, причем ожидает нажатия любой клавиши, если буфер клавиатуры в момент вызова пуст.

При вызове:

AH = 10h

При возврате:

AH – скэн-код клавиши из таблицы A.02-1
AL – код ASCII соответствующего знака из таблицы A.02-1

Примечание 1: обработчик прерывания INT 16\AH=20h делает то же самое плюс реагирует на дополнительные клавиши, которые имеются только на редко используемых 122-клавишных клавиатурах.

Примечание 2: в очень старых компьютерах с 84\86-клавишными клавиатурами для исполнения той же функции необходимо вызывать INT 16\AH=00h. В более "свежих" компьютерах INT 16\AH=00h действует так же и не реагирует на те клавиши, которых в старых клавиатурах нет.

8.01-84 INT 16\AH=11h – определение кода клавиши в буфере клавиатуры

Обработчик вызовов INT 16\AH=11h обслуживает наиболее распространенные модели 101\108-клавишных клавиатур, а также совместимые клавиши на других клавиатурах. Он не удаляет коды клавишей из буфера клавиатуры, а копирует их в регистры AH и AL, и не ждет нажатия клавиши, когда буфер пуст.

При вызове:

AH = 11h

При возврате:

если флаг ZF установлен, то буфер клавиатуры пуст.
если флаг ZF сброшен, то
AH – скэн-код из таблицы A.02-1
AL – код ASCII из таблицы A.02-1

Примечание 1: обработчик прерывания INT 16\AH=21h делает то же самое плюс реагирует на дополнительные клавиши, которые имеются только на редко используемых 122-клавишных клавиатурах.

Примечание 2: в старых компьютерах с 84\86-клавишными клавиатурами для исполнения той же функции необходимо вызывать INT 16\AH=01h.

В более "свежих" компьютерах INT 16\AH=01h действует так же и не реагирует на те клавиши, которых в старых клавиатурах нет.

8.01-85 INT 16\AH=12h – считывание состояний флагов клавиатуры

Обработчик вызовов INT 16\AH=12h считывает слово флагов из области данных BIOS, обычно из адреса 0040:0017h, показанного в таблице A.02-3

При вызове:

AH = 12h

При возврате:

AX – слово флагов:

- бит 0: – нажата правая клавиша Shift
- бит 1: – нажата левая клавиша Shift
- бит 2: – нажата любая из клавиш CTRL
- бит 3: – нажата любая из клавиш ALT
- бит 4: – переключатель Scroll Lock включен
- бит 5: – переключатель Num Lock включен
- бит 6: – переключатель Caps Lock включен
- бит 7: – переключатель Insert включен
- бит 8: – левая клавиша CTRL нажата
- бит 9: – левая клавиша ALT нажата
- бит 10: – правая клавиша CTRL нажата
- бит 11: – правая клавиша ALT нажата
- бит 12: – клавиша Scroll Lock нажата
- бит 13: – клавиша Num Lock нажата
- бит 14: – клавиша Caps Lock нажата
- бит 15: – клавиша SysRq нажата (примечание 2)

Примечание 1: в очень старых компьютерах с 84\86-клавишными клавиатурами аналогичную роль играет функция INT 16\AH=02h. Она отображает флаги только в битах 0 – 7 регистра AL. В более "свежих" компьютерах INT 16\AH=02h действует так же и не показывает те флаги, которых в старых компьютерах нет.

Примечание 2: нажатие клавиши SysRq фиксируется в слове флагов только когда удерживается нажатой любая (левая или правая) клавиша ALT.

8.01-86 INT 17\AH=00h – посылка одного знака на принтер

При вызове:

AH = 00h
AL – код посылаемого знака
DX – номер порта от 0000h до 0002h, соответствующий LPT1 – LPT3

При возврате:

AH – статус порта (и принтера) из таблицы A.14-3.

8.01-87 INT 17\AH=01h – инициализация параллельного порта

При вызове:

AH = 01h

DX – номер порта от 0000h до 0002h, соответствующий LPT1 – LPT3

При возврате:

AH – статус порта (и принтера) из таблицы A.14-3.

Примечание 1: возвращаемый при инициализации байт статуса не всегда отражает состояние правильно. Более надежные сведения можно получить, если после небольшой задержки вызвать INT 17\AX=0200h.

8.01-88 INT 17\AX=0200h – определить статус параллельного порта

Вызовы INT 17\AX=0200h обслуживаются как старыми версиями BIOS, так и более современными версиями EPP BIOS (Enhanced Parallel Port BIOS). В устаревших компьютерах EPP BIOS отсутствует, и тогда будет возвращен только байт статуса в регистре AH (A.14-3). Точно так будут реагировать и современные компьютеры, если предустановленное значение в регистре BX не равно 5050h. Но вызовы с предустановленными значениями BX=5050h и CH=45h перехватывает EPP BIOS, и реакция будет иной: в регистрах DX:BX (или в ES:BX) будет возвращен адрес вызова дополнительных функций, показанных в таблице A.14-4, для обслуживания параллельных портов согласно спецификации IEEE 1284.

При вызове:

AX = 0200h

BX = 5050h, если вызов адресован EPP BIOS

CH = 45h, если вызов адресован EPP BIOS

DX – номер порта от 0000h до 0002h, соответствующий LPT1 – LPT3

При возврате:

AH – статус порта (и принтера) из таблицы A.14-3.

Статус AH = 03h при установленном (CY) флаге CF возвращает EPP BIOS, если она не поддерживает запрошенный LPT порт.

Статус AH = 00h при сброшенном (NC) флаге CF возвращает EPP BIOS, если она поддерживает запрошенный LPT порт, и тогда:

CX:AL = 4550:50h – сигнатура EPP BIOS версий 1.x

CX:AL = 5050:45h – сигнатура EPP BIOS версий 3.x

EPP BIOS версий 1.x и 3.x возвращают ES без изменений и

DX:BX – сегмент: смещение для вызова функций EPP BIOS

EPP BIOS версии 7 изменяет сегмент ES и возвращает:

DX – базовый адрес для операций ввода-вывода EPP BIOS
ES:BX – сегмент: смещение для вызова функций EPP BIOS

Примечание 1: если вызов специально не адресован к EPP BIOS, то в регистрах BX и CH могут быть любые значения, кроме 5050h и 45h.

Примечание 2: изменение сегментного адреса в регистре ES является отличительным признаком наличия EPP BIOS последней 7-й версии, обеспечивающей управление мультиплексором LPT-порта.

8.01-89 INT 18 – бездисковая загрузка компьютера

При вызове ничего подготавливать не требуется.

Система BIOS вызывает обработчика прерывания INT 18 тогда, когда не находит в компьютере диска, с которого можно было бы продолжить загрузку. В старых компьютерах вызов INT 18 запускал интерпретатор языка BASIC, записанный в постоянную память BIOS. При вызове INT 18 современные компьютеры выводят на экран сообщение об отсутствии встроенного интерпретатора языка BASIC, после чего компьютер останавливается. Однако вызов INT 18 может быть перехвачен другим обработчиком, поставляемым в составе постоянного запоминающего устройства платы расширения. Таким способом, в частности, осуществляют бездисковую загрузку компьютеров через локальную сеть.

8.01-90 INT 19 – операции загрузки компьютера с диска

Обработчик прерывания INT 19 выполняет важную часть процедуры загрузки компьютера: он копирует boot-сектор с загрузочного раздела диска в адрес 0000:7C00h и передает управление туда. Сначала происходит обращение к тому диску, который указан первым в последовательности загрузочных дисков, установленной программой BIOS SETUP. С флоппи-дисков boot-сектор считывается непосредственно, а на жестких магнитных дисках адрес boot-сектора загрузочного раздела определяется из таблицы разделов в MBR (A.13-5). Если адресуемый диск недоступен, то производится попытка обращения к следующему диску из установленной последовательности. Если все попытки обращения к дискам закончатся неудачно, последним шагом будет вызов INT 18 (8.01-89).

Однако обработчик прерывания INT 19 не очищает память и не восстанавливает таблицу прерываний. Поэтому неподготовленные вызовы INT 19 обычно кончаются зависанием компьютера. При необходимости инициировать перезагрузку компьютера из исполняемой программы следует пользоваться не вызовом INT 19, а либо командой FEh контроллера клавиатуры (примечание 3 к A.11-3), либо дальним переходом (CALL FAR, 7.03-08) по адресу F000:FFF0h

(примечание 4 к A.12-1), где находится вход в программу загрузки у AT-совместимых компьютеров.

8.01-91 INT 1A\AH=00h – системное время

При вызове:

AH = 00h

При возврате:

AL – не нуль, если данный запрос – первый после полуночи.
CX – старшие 2 байта времени в тактах, 1800B0h тактов в сутки
DX – младшие 2 байта времени в тактах, 18.2 такта в секунду

Примечание 1: счет тактов обнуляется в полночь.

Примечание 2: число, возвращаемое в AL, используется операционной системой для счета дней. Если первый после полуночи запрос системного времени поступит не от операционной системы, то счет дней может быть нарушен.

Примечание 3: системное время можно установить с помощью INT 1A\AH=01h. При вызове INT 1A\AH=01h задаваемое число тактов должно быть аналогичным образом записано в регистрах CX и DX.

8.01-92 INT 1A\AH=02h – часы реального времени

При вызове:

AH = 02h

При возврате: если флаг CF установлен, то возвращаемые данные не верны
Если флаг CF сброшен, то

CH – часы
CL – минуты
DH – секунды
DL = 00h – стандартное время ("зимнее" в северном полушарии)
= 01h – время светового дня ("летнее" в северном полушарии)

Примечание 1: часы, минуты и секунды представляются в упакованном десятичном формате, по две десятичные цифры на байт.

Примечание 2: часы реального времени можно установить с помощью INT 1A\AH=03h. Перед вызовом следует записать задаваемые значения в регистры CH, CL, DH и DL в той же форме.

8.01-93 INT 1A\AH=04h – дата

При вызове:

AH = 04h

При возврате: если флаг CF установлен, то возвращаемые данные не верны

Если флаг CF сброшен, то

CH – столетие
CL – год
DH – месяц
DL – день

Примечание 1: все данные представляются в упакованном десятичном формате, по две десятичные цифры на байт.

Примечание 2: дату и связанные с ней данные можно установить с помощью INT 1A\AH=05h. Перед вызовом следует записать задаваемые значения в регистры CH, CL, DH и DL в той же форме.

8.01-94 INT 1A\AH=06h – установление времени регулярного действия

Обработчик прерывания INT 1A\AH=06h взводит "будильник" системы BIOS, который после этого каждый день в заданное время будет вызывать прерывание INT 4A. Желаемое действие, как предполагается, будет выполнять обработчик прерывания INT 4A, который должен быть заранее написан пользователем и загружен.

При вызове:

AH – 06h
CH – час
CL – минуты
DH – секунды

При возврате: если флаг CF сброшен, то миссия завершена успешно, а если флаг CF установлен, то произошла ошибка, например, "будильник" уже взведен.

Примечание 1: часы, минуты и секунды представляются в упакованном десятичном формате, по две десятичные цифры на байт.

Примечание 2: если задано значение FFh, то вызов INT 4A происходит при каждом приращении соответствующей величины. В частности, при CH=FFh вызов происходит каждый час, а при CH=CL=FFh – каждую минуту.

Примечание 3: "будильник" продолжает оставаться взведенным до тех пор, пока не будет сброшен посредством INT 1A\AH=07h, причем для сброса не нужны никакие другие приготовления, кроме AH = 07h.

8.01-95 INT 1B – реакция на нажатие "CTRL-Break"

При вызове ничего подготавливать не требуется.

Вызов INT 1B производится обработчиком прерывания INT 09 каждый раз, когда контроллер клавиатуры сообщает о нажатии клавишной комбинации

CTRL-Break. Устанавливаемый по умолчанию обработчик прерывания INT 1B взводит в состояние логической единицы флаг в области данных BIOS (бит 7 в байте со смещением 71h в таблице A.02-3), а потом возвращает управление вызывающей программе. Но состояние этого флага проверяют обработчики прерываний, установленные MS-DOS, когда их вызывает исполняемая программа. Обнаружив взведенное состояние флага, они вызывают обработчика прерывания INT 23. Именно он останавливает исполнение текущей программы и ответственен за все варианты последующего развития событий (1.01, 1.03, 8.02-83).

Подмена адреса обработчика прерывания INT 1B в таблице прерываний на адрес, указывающий на команду возврата IRET (7.03-30) – это распространенный прием предохранения любого ответственного блока программы от прерывания исполнения по нажатию клавишной комбинации CTRL-Break.

8.01-96 INT 1C – перехват тактов счетчика системного времени

При вызове ничего подготавливать не требуется.

Вызов INT 1C производится обработчиком прерывания INT 08 на каждом такте, то есть 18.2 раза в секунду. Устанавливаемый по умолчанию обработчик прерывания INT 1C просто возвращает управление вызывающей программе. Однако любой резидентной программе предоставлено право загрузить свой собственный обработчик прерывания INT 1C, который мог бы регулярно ее вызывать, обеспечивая тем самым возможность обслуживания развивающихся во времени процессов.

8.02 Обработчики от MS-DOS7 (INT 20 – INT 2E)

8.02-01 INT 20 – завершение исполнения программы

Обработчик прерывания INT 20 восстанавливает указатели в таблице прерываний по данным из префикса сегмента программы (PSP, A.07-1), высвобождает память, которую программа использовала, восстанавливает состояния стека и сегментных регистров, и затем передает управление туда, куда указывает INT 22 (8.02-82). Но INT 20 требует наличия сегментного адреса PSP в регистре CS: и не дает возможности оставить код ошибки (errorlevel, 3.15-03), характеризующий обстоятельства завершения программы. Чтобы избавиться от этих ограничений, рекомендуют завершать исполнение программ вызовом INT 21\AH=4Ch (8.02-55).

При вызове:

сегментный адрес в регистре CS: должен указывать на начало PSP.

При возврате:

возврата не будет.

Примечание 1: при исполнении программ в среде отладчика Debug.exe INT 20 и INT 21\AH=4Ch действуют неодинаково: INT 20 передает управление отладчику, тогда как INT 21\AH=4Ch закрывает сеанс работы с отладчиком и передает управление DOS.

Примечание 2: завершить программу без оставления кода ошибки можно также вызовом INT 21\AH=00h. MS-DOS7 поддерживает INT 21\AH=00h ради сохранения совместимости со старыми программами.

8.02-02 INT 21\AH=01h – получение знака из канала ввода

Обработчик прерывания INT 21\AH=01h считывает знак из стандартного канала ввода STDIN (STandarD INput), записывает считанный знак в регистр AL и посылает копию считанного знака в стандартный канал вывода STDOUT (STandarD OUTput). Оба эти канала – STDIN и STDOUT – могут быть перенаправлены, но по умолчанию STDIN получает сигналы с клавиатуры, а STDOUT пересылает все на дисплей, и тогда при вызове INT 21\AH=01h компьютер начинает ожидать нажатия клавиши, а после нажатия отображает соответствующий знак на экране.

При вызове:

AH = 01h

При возврате:

AL – код ASCII считанного знака

Примечание 1: возвращаемый код ASCII представлен правыми двумя цифрами четырехзначных шестнадцатеричных чисел в таблице A.02-1.

Примечание 2: нельзя вызывать INT 21\AH=01h из командных файлов, которые исполняются с перенаправления: такие вызовы перехватывают из канала STDIN команды, предназначенные интерпретатору.

Примечание 3: функция INT 21\AH=01h проверяет состояние флага, взводимого клавишными комбинациями CTRL-C и CTRL-Break (8.01-95). Если окажется, что этот флаг взведен, то будет вызван обработчик прерывания INT 23.

Примечание 4: обработчик прерывания INT 21\AH=03h аналогичным образом считывает знак из канала STDAUX, который связан с портом COM1.

8.02-03 INT 21\AH=02h – посылка знака в канал вывода

Обработчик прерывания INT 21\AH=02h посылает знак в стандартный канал вывода STDOUT, который по умолчанию пересылает все на дисплей, если не произведено перенаправление этого канала куда-либо еще.

При вызове:

AH = 02h
DL – код ASCII посылаемого знака

При возврате:

AH – код посланного знака (кроме 09h, который заменяется на 20h).

Примечание 1: перенаправление канала STDOUT в файл производится без проверок наличия дискеты в дисководе, заполненности этой дискеты, защиты ее от записи и т.д. Но если в момент вызова дисковод занят, то обработчик прерывания INT 21\AH=02h будет ждать, пока дисковод завершит исполнение предыдущей операции.

Примечание 2: функция INT 21\AH=02h проверяет состояние флага, взводимого клавишными комбинациями CTRL-C и CTRL-Break (8.01-95). Если окажется, что этот флаг взведен, то будет вызван обработчик прерывания INT 23.

Примечание 3: обработчик прерывания INT 21\AH=06h делает то же, только не проверяет флаг CTRL-C\CTRL-Break. Однако вызов INT 21\AH=06h действует совершенно по-другому при DL=FFh (8.02-04).

Примечание 4: подобно тому, как INT 21\AH=02h посылает знак в канал STDOUT, INT 21\AH=04h посылает знак в канал STDAUX (порт COM1), а INT 21\AH=05h – на принтер, подключенный к порту LPT1.

8.02-04 INT 21\AH=06h – копирование знака из канала ввода

Обработчик прерывания INT 21\AH=06h при условии DL = FFh делает почти то же, что INT 21\AH=01h, но:

– не проверяет флаг CTRL-C\CTRL-BREAK,
– не удаляет код знака из буфера клавиатуры,
– не вызывает ожидания нажатия клавиши,
– не посылает копию выводимого кода в канал STDOUT
– дополнительно воспринимает расширенные коды клавиш.

Под расширенными кодами клавиш имеются ввиду те, которые отличаются только скэн-кодами, а коды ASCII для всех таких клавиш (правые две цифры из чисел в таблице A.02-1) имеют значение 00h или E0h. Если BIOS сообщает любое из этих значений, то обработчик прерывания INT 21\AH=06h сбрасывает флаг ZF и устанавливает AL=00h. Это означает, что была нажата клавиша, которой

соответствует расширенный код, и что следует вызвать INT 21\AH=06h повторно: тогда в регистре AL будет возвращен скэн-код (левые две цифры из чисел в таблице A.02-1), по которому такие нажатия клавиш различаются.

Коды ASCII, отличающиеся от 00h и E0h, записываются в регистр AL сразу, и тогда скэн-код не возвращается; повторный вызов INT 21\AH=06h допустим, но он будет исполняться как отдельный, не зависимый от предыдущего.

При вызове:
AH = 06h
DL = FFh (для других значений – примечание 3 к 8.02-03)
При возврате: если флаг ZF установлен, то отказ: канал STDIN пуст
Если флаг ZF сброшен, то
AL – код знака (ASCII или скэн-код)

Примечание 1: INT 21\AH=07h делает то же самое, но игнорирует DL, вызывает ожидание нажатия клавиши, не меняет состояния флага ZF, и удаляет считанный код из буфера клавиатуры.
Примечание 2: INT 21\AH=08h делает то же самое, что INT 21\AH=07h, но сверх того вызывает INT 23 после нажатия CTRL-C или CTRL-BREAK.
Примечание 3: нельзя вызывать INT 21\AH=06h-08h из командных файлов, которые исполняются с перенаправления: такие вызовы перехватывают из канала STDIN команды, предназначенные интерпретатору.
Примечание 4: когда установлен флаг поддержки иероглифических языков (байт со смещением 3Ch в таблице A.07-1), тогда INT 21\AH=06h-08h могут возвращать частично сформированные двухбайтовые коды.

8.02-05 INT 21\AH=09h – посылка строки знаков в канал STDOUT

При вызове:
AH = 09h
DS:DX – указатель на начало строки, оканчивающейся знаком "$"
При возврате:
AL = 24h (код знака "$", которым кончается строка)

Примечание 1: посылка знаков прекращается, как только будет встречен первый знак "$" (24h), причем сам знак "$" в канал не посылается.
Примечание 2: функция INT 21\AH=09h проверяет состояние флага, взводимого клавишными комбинациями CTRL-C и CTRL-Break (8.01-95). Если окажется, что этот флаг взведен, то будет вызван обработчик прерывания INT 23.
Примечание 3: послать строку знаков можно также посредством INT21\AH=40h (8.02-36).

8.02-06 INT 21\AH=0Ah – буферированный ввод из канала STDIN

Когда канал STDIN не перенаправлен, обработчик прерывания INT 21\AH=0Ah ожидает нажатия клавиш, и после каждого нажатия записывает код клавиши в заранее подготовленный буфер. Если первоначально буфер был не пуст, то новые коды можно записывать после сохраняемого прежнего содержания. Записываемые коды знаков по мере их поступления копируются в канал STDOUT для отображения на экране. Запись прекращается при поступлении кода 0Dh в результате нажатия клавиши ENTER. Когда канал STDIN пересылает данные, перенаправленные из файла, то запись происходит безостановочно до тех пор, пока не будет встречен первый байт 0Dh. В любом случае запись прекращается, когда буфер оказывается заполнен.

При вызове:

AH = 0Ah

DS:DX – указатель на начало буфера, где заранее записаны 2 байта:

с адреса 00h: полный размер буфера в байтах;

с адреса 01h: с какого смещения записывать новые данные.

При возврате:

DS:DX – указатель на заполненный буфер, где 2 байта содержат:

с адреса 00h: – полный размер буфера в байтах;

с адреса 01h: – число фактически заполненных байтов.

Примечание 1: если первый байт буфера содержит код 00h, то обработчик прерывания INT 21\AH=0Ah не ждет нажатия клавиши, а сразу же возвращает управление вызывающей программе.

Примечание 2: счет записанных в буфер байтов начинается с байта 02h, причем в это число не входит байт 0Dh, которым кончается строка.

Примечание 3: функция INT 21\AH=0Ah проверяет состояние флага, взводимого клавишными комбинациями CTRL-C и CTRL-Break (8.01-95). Если окажется, что этот флаг взведен, то будет вызван обработчик прерывания INT 23.

Примечание 4: нельзя вызывать INT 21\AH=0Ah из командных файлов, которые исполняются с перенаправления: такие вызовы перехватывают из канала STDIN команды, предназначенные интерпретатору.

8.02-07 INT 21\AH=0Bh – определение статуса канала ввода

Обработчик прерывания INT 21\AH=0Bh не выполняет пересылку данных, а только позволяет определить, имеются ли в канале STDIN подлежащие пересылке данные. Если канал STDIN не перенаправлен, то результат показывает, пуст ли буфер клавиатуры. Если предполагается получать перенаправление из файла, то

результат покажет, произведено ли перенаправление на самом деле и не исчерпано ли уже все содержимое файла.

При вызове:

AH = 0Bh

При возврате:

AL = 00h, если канал STDIN пуст, или

AL = FFh, если в канале STDIN имеется хотя бы один байт.

Примечание 1: функция INT 21\AH=0Bh проверяет состояние флага, взводимого клавишными комбинациями CTRL-C и CTRL-Break (8.01-95). Если окажется, что этот флаг взведен, то будет вызван обработчик прерывания INT 23.

8.02-08 INT 21\AH=0Ch – очистить буфер клавиатуры и считать знак

Обработчик прерывания INT 21\AH=0Ch используется для того, чтобы очистить буфер клавиатуры, а затем автоматически вызвать любую функцию из числа INT 21\AH=01h, 06h, 07h, 08h, 0Ah, в зависимости от того, какой код помещен в регистр AL. Все другие особенности исполнения определяются вызываемой функцией.

При вызове:

AH = 0Ch

AL – код вызываемой функции ввода: 01h, 06h, 07h, 08h или 0Ah

другие регистры – как нужно для вызываемой функции ввода

При возврате: то же, что должна возвратить вызванная функция ввода

Примечание 1: нельзя вызывать INT 21\AH=0Ch с указанными выше кодами в AL из командных файлов, которые исполняются с перенаправления: такие вызовы перехватывают из канала STDIN команды, предназначенные интерпретатору.

Примечание 2: если в AL указать не один из перечисленных выше кодов, а какой-нибудь другой, то буфер клавиатуры будет очищен, но управление будет возвращено без попытки считывания данных.

8.02-09 INT 21\AH=0Dh – запись содержимого из буферов на диск

Сброс буферов выполняют каждый раз перед обращением к другому диску. Помимо того, вызовом INT 21\AH=0Dh устанавливается принимаемый по умолчанию адрес области DTA (8.02-16).

При вызове:

AH = 0Dh

При возврате: если флаг CF сброшен, то запись проведена успешно

Примечание 1: вызов INT 21\AH=0Dh не корректирует соответствующим образом записи в каталогах диска. Для этого надо закрывать относящиеся к файлам номерные ссылки посредством INT 21\AH=3Eh (8.02-34).

8.02-10 INT 21\AH=0Eh – назначение "текущего" логического диска

Функция INT 21\AH=0Eh задает тот логический диск, к которому по умолчанию будет отнесено действие дисковых операций, если в их спецификации прямое указание на конкретный логический диск отсутствует.

При вызове:

AH = 0Eh
DL – номер назначаемого логического диска (примечание 1)

При возврате:

AL = предельно допустимое число логических дисков, задаваемое командой LASTDRIVE (4.17, 4.18).

Примечание 1: в отличие от нумерации дисководов, нумерация логических дисков производится по порядку назначенных им буквенных обозначений: 00h = A:, 01h = B:, 02h = C:, и т.д.

Примечание 2: при вызове INT 21\AH=0Eh из драйверов возвращаемое в AL число может быть неверным, так как оно считывается из "списка списков" (таблица A.01 2, смещение 21h), а заносится туда только после окончания загрузки всех драйверов командами DEVICE (4.06, 4.07).

Примечание 3: номер принимаемого по умолчанию логического диска можно выяснить с помощью INT 21\AH=19h (8.02-15).

8.02-11 INT 21\AH=11h – поиск первого подходящего файла по данным FCB

Обработчик прерывания INT 21\AH=11h использует устаревшую спецификацию FCB (File Control Block), которая непригодна для доступа к файлам на дисках с файловой системой FAT-32, но вполне пригодна для поиска в текущем каталоге, в том числе на дисках с файловой системой FAT-32. Допустимые форматы FCB показаны в таблице A.09-5. Исходные данные в FCB могут содержать знак подстановки "?" (2.01-03). Подготовку исходных данных в формате FCB удобно осуществлять с помощью INT 21\AH=29h (8.02-19).

По окончании поиска некоторые из возвращаемых результатов дополняют данные в FCB, как показано в таблице A.09-5. Но основная часть сведений о найденном файле возвращается в области DTA (Data Transfer Area) размером 128 байт, которая по умолчанию расположена в PSP (A.07-1) начиная со смещения 80h, но может быть перемещена с помощью INT 21\AH=1Ah (8.02-16). Формат возвращаемых данных в DTA зависит от того, какой формат FCB – обычный или

расширенный – был использован при вызове; оба варианта формата возвращаемых данных показаны в таблице A.09-1.

При вызове:

AH = 11h

DS:DX – указатель на блок FCB, обычный или расширенный (A.09-5)

При возврате:

AL – код завершения из таблицы A.06-1.

Область DTA (A.09-1) со сведениями о первом найденном файле.

Примечание 1: чтобы получить сведения о метке диска, нужно из корневого каталога вызвать INT 21\AH=11h с использованием FCB расширенного формата и байта атрибутов 08h.

Примечание 2: чтобы получить сведения о каталоге, нужно из вышестоящего каталога вызвать INT 21\AH=11h с использованием FCB расширенного формата и байта атрибутов 10h.

Примечание 3: если предполагается продолжить поиск дальше с помощью INT 21\AH=12h (8.02-12), то все данные в FCB и DTA, в том числе возвращенные, должны быть сохранены без изменений.

Примечание 4: поиск без обращения к FCB возможен с помощью INT 21\AH=4Eh.

8.02-12 INT 21\AH=12h – поиск следующего файла согласно данным FCB

Обработчик прерывания INT 21\AH=12h следует вызывать для продолжения поиска после успешного завершения миссии INT 21\AH=11h (8.02-11) или после успешного завершения предыдущего вызова INT 21\AH=12h. Данные, оставленные в FCB и DTA предшествовавшей операцией поиска, должны быть сохранены без изменений: они являются исходными для продолжения поиска. Результат поиска возвращается точно так же, как после вызова INT 21\AH=11h (8.02-11).

При вызове:

AH = 12h

DS:DX – указатель на блок FCB, обычный или расширенный (A.09-5).

Данные в области DTA, оставленные предыдущей операцией поиска.

При возврате:

AL – код завершения из таблицы A.06-1, причем код FFh означает, что больше подходящих файлов нет.

Область DTA (A.09-1) со сведениями об очередном найденном файле.

Примечание 1: все примечания к INT 21\AH=11h (8.02-11) в равной мере относятся также и к INT 21\AH=12h.

8.02-13 INT 21\AH=13h – удаление файлов согласно спецификации FCB

Обработчик прерывания INT 21\AH=13h использует устаревшую спецификацию FCB (File Control Block), которая тем не менее пригодна для удаления одного или нескольких файлов из текущего каталога, в том числе на дисках с файловой системой FAT-32. Допустимые форматы FCB показаны в таблице A.09-5. Для удаления нескольких файлов допускается использование в FCB знака подстановки "?" (2.01-03). Подготовку исходных данных в формате FCB удобно осуществлять с помощью INT 21\AH=29h (8.02-19).

Подлежащие удалению файлы должны быть заранее закрыты (8.02-34), с них должны быть сняты атрибуты HSR (A.09-2), и диск, на котором находятся файлы, не должен быть защищен от записи.

При вызове:

AH = 13h

DS:DX – указатель на блок FCB, обычный или расширенный (A.09-5)

При возврате:

AL – код завершения из таблицы A.06-1, причем код FFh означает, что соответствующие данной спецификации файлы не найдены.

Примечание 1: благодаря наличию байта атрибутов в расширенном FCB (A.09-5) возможно удаление меток диска и файлов с атрибутом R (только для чтения). Возможно также удаление подкаталогов, но содержащиеся в них файлы не удаляются, а становятся потерянными кластерами.

Примечание 2: удаляемый файл не стирается, просто соответствующая ему запись в каталоге делается недействительной путем замены первого знака в имени файла кодом E5h.

Примечание 3: удаление с помощью INT 21\AH=13h не затрагивает те записи в каталоге, которые содержат продолжение длинного имени файла.

Примечание 4: удаление файлов без использования спецификации FCB производится с помощью INT 21\AH=41h (8.02-37).

8.02-14 INT 21\AH=17h – переименование согласно спецификации FCB

При вызове:

AH = 17h

DS:DX – указатель на неоткрытый блок FCB (A.09-5), обычный или расширенный. Прежнее имя записывается на своем обычном месте, новое имя – на 10h байт дальше, то есть точно под первым в следующей строке дампа. Конкретные значения смещений приведены в таблице A.09-5. Оба имени должны быть записаны в нормализованной форме, например, с помощью

INT 21\AH=29h (8.02-19). Требуемая длина буфера – 28 байт для обычного и 35 байт для расширенного FCB.

При возврате:

AL = 00h – успешное переименование, или
= FFh – неудачное завершение: не найдены подходящие файлы, или файл защищен HRS-атрибутами, или файл с заданным новым именем уже существует.

Примечание 1: если используются знаки подстановки "?" (2.01-03), то в обоих именах (или суффиксах) они должны занимать одинаковые места: там будут сохранены знаки из прежнего имени или суффикса.

Примечание 2: в спецификации FCB не предусмотрено указание пути, поэтому переименование производится только в текущем каталоге. Вне текущего каталога переименование возможно посредством функции INT 21\AH=56h (8.02-62), не использующей спецификацию FCB.

Примечание 3: при использовании расширенного FCB возможно переименование подкаталогов из вышестоящего каталога с указанием байта атрибутов 10h, а также меток тома – из корневого каталога с указанием байта атрибутов 08h.

8.02-15 INT 21\AH=19h – выяснение "текущего" логического диска

Функция INT 21\AH=19h возвращает номер того логического диска, который принимается по умолчанию, когда для дисковой операции конкретная спецификация диска не указана.

При вызове:

AH = 19h

При возврате:

AL – номер "текущего" логического диска (примечание 1 к 8.02-10).

Примечание 1: сменить назначение "текущего" логического диска можно с помощью функции INT 21\AH=0Eh (8.02-10).

8.02-16 INT 21\AH=1Ah,2Fh – задание и выяснение адреса области DTA

Область передачи данных DTA (Data Transfer Area) – это буфер длиной 128 байт, используемый обработчиками прерываний INT 21\AH=11h, 12h, 4Eh, 4Fh. По умолчанию область DTA расположена в PSP исполняемой программы начиная со смещения 80h (примечание 6 к A.07-1).

При вызове:

AH = 1Ah – задать новый адрес области DTA
= 2Fh – выяснить действующий адрес области DTA

DS:DX – новый адрес области DTA (только при вызове с AH = 1Ah)

При возврате:

ES:BX – действующий адрес DTA (только после вызова с AH = 2Fh)

Примечание 1: форматы данных в области DTA показаны в таблице A.09-1.

Примечание 2: принимаемый по умолчанию адрес области DTA (PSP:0080h) восстанавливается при каждом вызове INT 21\AH=0Dh (8.02-09).

8.02-17 INT 21\AH=1Ch – получение сведений о диске

При вызове:

AH = 1Ch

DL – номер диска (примечание 1)

При возврате:

AH – байт-идентификатор носителя записи:
= F8h – жесткий магнитный диск,
= F9h – дискета емкостью 1.2 Мбайт или 720 кбайт,
= FAh – виртуальный RAM-диск,
= FDh – дискета емкостью 360 кбайт,
= F0h – все другие, включая дискеты емкостью 1.44 Мбайт.

CX – длина сектора в байтах;

DS:BX – указатель на байт-идентификатор, записанный в память; данные в DX и AL не сохраняются.

Примечание 1: здесь используется смещенная нумерация логических дисков, при которой 00h – это "текущий" (принимаемый по умолчанию) диск, а потом следуют 01h = A:, 02h = B:, 03h = C:, и т.д.

Примечание 2: при запросе данных недействующего или несуществующего диска INT 21\AH=1Ch не объявляет об ошибке, а просто не изменяет значения в AH, BX и DS. Но данные в DX и в AL утрачиваются.

Примечание 3: аналогичным образом вызов INT 21\AH=1Bh сообщает сведения о текущем диске, игнорируя номер в регистре DL. И INT 21\AH=1Ch, и INT 21\AH=1Bh, – устаревшие функции, не поддерживающие большинство современных типов носителей.

8.02-18 INT 21\AH=25h – записать новый указатель в таблицу прерываний

При вызове:

AH = 25h

AL – номер прерывания

DS:DX – указатель, который надлежит записать

Примечание 1: при записи нового указателя прежний указатель утрачивается. Если его надо сохранить, то побеспокоиться об этом надо заранее.

Примечание 2: обработчик прерывания INT 21\AH=25h, поставляемый MS-DOS7, можно использовать при работе в реальном режиме и в режиме V86, но при работе в защищенном режиме его использовать нельзя.

8.02-19 INT 21\AX=2901h – разбор имени для записи в FCB

Обработчик прерывания INT 21\AX=2901h преобразует имя файла в нормализованный вид и формирует неоткрытый блок FCB (File Control Block) обыкновенного типа (A.09-5). Имя и суффикс записываются отдельно в свои поля блока FCB, причем буквы переводятся в заглавные, знаки подстановки "звездочка" (2.01-03) заменяются должным числом знаков "?". Если длина имени меньше 8 знаков, а длина суффикса меньше 3 знаков, то оставшиеся знакоместа заполняются кодами пробела (20h). Разбор заканчивается на первом встреченном знаке пробела, косой черты, или на байте 0Dh (завершение на байте 00h не допускается).

Перед именем нельзя указывать путь, но допускается указание буквы диска (например, A:Config.sys). Буква диска переводится в номер и записывается перед именем в поле номера диска. Если второй знак в представленной строке не является двоеточием, то считается, что диск не указан, и тогда в поле номера диска в FCB записывается код 00h, что означает "текущий" диск.

При вызове:

AH = 2901h
DS:SI – указатель на подлежащую разбору строку
ES:DI – указатель на место для размещения FCB длиной 21 байт

При возврате:

AL = 00h – успешное завершение, знаки подстановки не встречены
= 01h – успешное завершение со знаками подстановки
= FFh – неудачное завершение, неверная спецификация
DS:SI – указатель на первый "неразобранный" знак
ES:DI – указатель на сформированный блок FCB

Примечание 1: если в дальнейшем формируемый блок FCB будет использован для доступа к файлу, то длина буфера должна быть 36 байт.

Примечание 2: INT 21\AX=2903h делает то же самое, но только не перезаписывает прежнее содержание поля номера диска в блоке FCB.

8.02-20 INT 21\AH=2Ah – определение системной даты

При вызове:

AH = 2Ah

При возврате:

AL – день недели (00h = воскресенье)

CX – год (в пределах 1980 – 2099)
DH – месяц
DL – день

Примечание 1: все значения – в упакованном десятичном формате, по две десятичные цифры на байт.
Примечание 2: для задания новой даты нужно подготовить данные в AL, CX, DH и DL так же, как показано выше, и затем вызвать INT 21\AH=2Bh. При неудачном завершении дата не будет изменена, а в AL будет возвращено значение FFh.

8.02-21 INT 21\AH=2Ch – определение системного времени

При вызове:
AH = 2Ch
При возврате:
CH – часы
CL – минуты
DH – секунды
DL – сотые доли секунды

Примечание 1: все значения – в упакованном десятичном формате, по две десятичные цифры на байт.
Примечание 2: некоторые компьютеры считают значение в DL шагами по 0.05 секунды; встречаются и такие, которые всегда возвращают DL=00h.
Примечание 3: чтобы задать время, нужно подготовить данные в CH, CL, DH и DL так же, как показано выше, и затем вызвать INT 21\AH=2Dh. При неудачном завершении время не будет изменено, а в AL будет возвращено значение FFh.

8.02-22 INT 21\AH=30h – определение версии DOS

При вызове:
AX = 3000h – возвратить в BH идентификатор разработчика DOS.
= 3001h – возвратить в BH флаг версии DOS.
При возврате:
AL.AH – номер версии
BH – идентификатор разработчика или флаг версии DOS.

Примечание 1: идентификаторы разработчиков: 00h – IBM, 66h – ФизТехСофт, EEh – DR-DOS, EFh – Novell, FDh – FreeDOS, FFh – Microsoft.
Примечание 2: если в флаге версии установлен бит 3, то это специальная версия для поставки DOS в микросхемах постоянной памяти.

Примечание 3: возвращаемый номер версии считывается из слова со смещением 40h в PSP (A.07-1) вызывающей программы. Если установлен драйвер SETVER.EXE (5.01-02), и имя вызывающей программы внесено в таблицу этого драйвера, то в момент запуска программы в ее PSP вместо действительного номера версии DOS будет записан "подставной" номер из этой таблицы.

Примечание 4: чтобы получить заведомо не подмененный номер DOS, нужно вызывать INT 21\AX=3306h (8.02-27).

8.02-23 INT 21\AH=31h – завершение исполнения с оставлением TSR-модуля

При вызове:

AH = 31h
AL – оставляемое значение кода ошибки (примечание 3 к 8.02-55)
DX – размер резидентного модуля в 16-байтовых параграфах, не менее 6 параграфов, отсчитывается от начала PSP.

Примечание 1: вызов INT 21\AH=31h высвобождает занятую программой память, за исключением резидентного модуля, восстанавливает указатели в таблице прерываний по данным из PSP, восстанавливает состояния сегментных регистров и стека, а затем передает управление по адресу, определяемому указателем INT 22 (8.02-82). Но функция INT 21\AH=31h не закрывает открытые файлы, не высвобождает область окружения и ту память, которая предоставлена программе посредством прерывания INT 21\AH=48h (8.02-50). При необходимости эти операции должна заранее выполнить сама завершаемая программа.

Примечание 2: устаревшее прерывание INT 27 делает то же самое, но не позволяет оставить ненулевые значения кода ошибки (errorlevel) и ограничивает размер резидентного модуля величиной 64 кбайт.

8.02-24 INT 21\AH=32h – адрес блока параметров диска (DPB)

При вызове:

AH = 32h
DL – номер диска (примечание 1 к 8.02-17)

При возврате:

AL = FFh, – запрошен сетевой или недействительный диск;
= 00h, – запрос выполнен успешно, и тогда
DS:BX – указатель на блок параметров диска (DPB, A.03-1)

Примечание 1: INT 21\AH=1Fh тоже возвращает в DS:BX указатель на блок DPB, но только для "текущего" диска; значение в DL игнорируется.

Примечание 2: и INT 21\AH=32h, и INT 21\AH=1Fh стараются обновить блок параметров посредством считывания данных с диска. При неудаче считывания следует вызов INT 24 с его вопросом "Abort, Retry, Fail?". Если такой исход в данный момент нежелателен, то указатель на блок DPB следует искать через "список списков" (примечание 1 к A.01-2).

Примечание 3: и INT 21\AH=32h, и INT 21\AH=1Fh нельзя применять по отношению к дискам с файловой системой FAT-32; вместо них следует вызывать INT 21\AX=7302 (8.02-79).

8.02-25 INT 21\AX=3300h – определение состояния флага BREAK

При вызове:

AX = 3300h

При возврате:

DL = 00h – флаг BREAK сброшен (OFF),
= 01h – флаг BREAK установлен (ON).

Примечание 1: флаг BREAK находится в области текущих данных DOS (A.01-3, смещение 17h).

Примечание 2: функция INT 21\AX=3301 позволяет изменить состояние флага BREAK и принимает его из регистра DL в тех же обозначениях.

8.02-26 INT 21\AX=3305h – выяснение загрузочного диска

При вызове:

AX = 3305h

При возврате:

DL – номер загрузочного диска (примечание 1 к 8.02-17)

Примечание 1: номер логического диска, с которого компьютер был загружен, считывается из "списка списков" (A.01-2, смещение 43h).

8.02-27 INT 21\AX=3306h – истинная версия DOS

При вызове:

AX = 3306h

При возврате:

AL = FFh, если версия DOS меньше 5.00
BL.BH – версия DOS
DH – флаги:
бит 3 – DOS в микросхемах постоянной памяти
бит 4 – DOS загружена в область HMA
DL – номер ревизии версии DOS.

Примечание 1: номер версии, возвращаемый данной функцией, не подменяется данными драйвера SETVER.EXE (в отличие от INT 21\AH=30h).

Примечание 2: эта функция не поддерживается версиями DOS меньше 5.00. Помимо AL=FFh, о версии DOS меньше 5.00 могут свидетельствовать значения в BL менее 05h или в BH более 64h.

8.02-28 INT 21\AH=34h – адрес флага InDOS

Флаг InDOS – это однобайтовый счетчик вложенных вызовов функций DOS: значение в нем увеличивается на единицу при каждом вызове любой функции прерывания INT 21 и уменьшается на единицу при завершении ее исполнения. Состояния флага InDOS и флага критической ошибки (A.01-3) нужно проверять каждый раз перед вызовом функции DOS из любого резидентного модуля: если хотя бы один из этих флагов имеет ненулевое значение, то такой вызов небезопасен.

Рекомендуется вызвать функцию INT 21\AH=34h при первом исполнении программы, запомнить адрес флага InDOS, и тогда для проверки значения этого флага в дальнейшем уже не потребуется вызывать функцию DOS.

При вызове:

AH = 34h

При возврате:

ES:BX – указатель на флаг InDOS

Примечание 1: флаг критической ошибки представляет собой байт, который расположен перед флагом InDOS (A.01-3). Указатель на флаг критической ошибки можно получить с помощью INT 21\AX=5D06h.

8.02-29 INT 21\AH=35h – получение адреса обработчика прерывания

При вызове:

AH = 35h

AL – номер прерывания

При возврате:

ES:BX – адрес вызова обработчика прерывания.

8.02-30 INT 21\AH=36h – определение свободного пространства на диске

При вызове:

AH = 36h

DL – номер диска (примечание 1 к 8.02-17)

При возврате:

AX – число секторов в кластере
BX – число свободных кластеров
CX – число байтов в секторе
DX – полное число кластеров на диске

Примечание 1: свободное дисковое пространство – это произведение AX•BX•CX
Примечание 2: полное дисковое пространство – это произведение AX•CX•DX
Примечание 3: "потерянные" кластеры причисляются к занятым.
Примечание 4: при обращении к недействующему или несуществующему диску возвращается значение AX = FFFFh. При обращении к оптическому дисководу CD-ROM возвращаемые данные заведомо не верны.
Примечание 5: обращения к дискам с файловой системой FAT-32 допустимы, но возвращаемые результаты ограничены величиной 2048 Мбайт; для дисков большего объема получить точные данные можно с помощью INT 21\AX=7303h (8.02-80).

8.02-31 INT 21\AH=39h-3Ah – создание или удаление пустого подкаталога

При вызове:
AH = 39h – создать подкаталог
= 3Ah – удалить пустой подкаталог
DS:DX – указатель на строку с именем адресуемого подкаталога. Строка должна заканчиваться байтом 00h. Перед именем адресуемого подкаталога в строке может быть указан путь. Длина строки – до 64 байт.

При неудачном завершении флаг CF установлен, в AL – код завершения (A.06-1)
При успешном завершении флаг CF сброшен, AX не сохраняется.

Примечание 1: каталог удалить нельзя, если он корневой, если он не пуст, а также если он отмечен как текущий в таблице CDS (A.03-3).
Примечание 2: в отличие от обычных каталогов, объем корневого каталога ограничен. Если он полон, то подкаталог в нем создать нельзя.

8.02-32 INT 21\AH=3Bh – переназначение "текущего" каталога

Обработчик прерывания INT 21\AH=3Bh перезаписывает принимаемый по умолчанию путь в таблице CDS (A.03-3).

При вызове:
AH = 3Bh
DS:DX – указатель на строку с именем заданного каталога. Строка должна заканчиваться байтом 00h. Перед именем заданного

каталога в строке может быть указан путь, включая букву диска. Длина строки – до 64 байт.

При неудачном завершении флаг CF установлен, в AL – код завершения (A.06-1).
При успешном завершении флаг CF сброшен, AX не сохраняется.

Примечание 1: если вызов INT 21\AH=3Bh адресован другому диску, то "текущий" диск и "текущий" каталог не изменяются, но заданный каталог станет "текущим", когда станет "текущим" адресуемый диск.

Примечание 2: с помощью функции INT 21\AH=47h (8.02-49) можно выяснить, какой каталог является "текущим" в данный момент.

8.02-33 INT 21\AH=3Dh – выдача номерной ссылки для доступа

Номерная ссылка (handle) – это предоставляемое программе шестнадцатеричное число, ассоциированное с блоком описания в таблице SFT (System File Table, A.01-4). Каждый блок описания в таблице SFT относится к объекту: либо к существующему файлу, либо к выделенной области XMS-памяти, либо к каналу доступа. Обработчик прерывания INT 21\AH=3Dh выясняет, есть ли в таблице SFT блок описания, относящийся к запрошенному объекту. Если есть, то отмечаемое в этом блоке описания число ссылок увеличивается на единицу, байт с номером блока описания вводится в таблицу JFT (Job File Table, A.07-1, смещение 18h) исполняемой программы, и порядковый номер этого байта в таблице JFT становится той самой номерной ссылкой. Если для запрошенного объекта нет блока описания в таблице SFT, то обработчик прерывания INT 21\AH=3Dh его создает, и далее выполняет ту же последовательность операций, кончающуюся выдачей номерной ссылки. Файлы, доступ к которым открыт действующей номерной ссылкой, принято называть открытыми.

При вызове:

AH = 3Dh
AL – права доступа из таблицы A.09-4
DS:DX – указатель на имя объекта, за которым следует байт 00h

При неудачном завершении флаг CF установлен, в AX – код завершения (A.06-1).
Если при возврате флаг CF сброшен, то

AX – предоставленная номерная ссылка

Примечание 1: чтобы получить номерную ссылку на канал управления драйвером нужно использовать имя, записанное начиная со смещения 0Ah в заголовке драйвера (A.05-1). Пути поиска заголовков резидентных модулей показаны в примечании 2 к A.01-2, в 8.03-12 и во введении к разделу 8.03.

Примечание 2: знаки подстановки в имени объекта не допускаются.

Примечание 3: при открытии файла указатель места доступа устанавливается на начало файла.

Примечание 4: при любой комбинации атрибутов файл получит номерную ссылку и будет открыт для доступа, однако получить номерную ссылку на каталог прямым вызовом INT 21\AH=3Dh нельзя.

Примечание 5: при вызове INT 21\AH=3Dh посредством серверной функции INT 21\AX=5D00h (8.02-68) возможно задать в регистре CL маску атрибутов (A.09-2) для файла, который надо открыть.

Примечание 6: номерные ссылки, наследуемые от родительской программы, наследуют те же права доступа к объекту. Условия наследования также задаются байтом прав доступа в регистре AL (A.09-4).

Примечание 7: для открытия доступа к файлам на дисках с файловой системой FAT-32 нужно получать номерные ссылки с помощью INT 21\AX=6C00h (8.02-78).

8.02-34 INT 21\AH=3Eh – аннулирование номерной ссылки

Обработчик прерывания INT 21\AH=3Eh уменьшает на единицу число ссылок на тот же объект в относящемся к нему блоке описания в таблице SFT (A.01-4) и удаляет номер этого блока описания из таблицы JFT (A.07-1) исполняемой программы. Если объект – файл, и если он был изменен, то данные из буферов записываются на диск, и записи в каталоге диска соответствующим образом корректируются. Если дублирующих номерных ссылок на тот же файл данная программа не имеет, то она утрачивает доступ к этому файлу, для нее он становится "закрыт". Однако операционная система будет считать файл "закрытым" только тогда, когда станет равно нулю число ссылок на него в относящемся к нему блоке описания в таблице SFT: только тогда этот файл можно удалить, и он перестает быть поводом для блокировки в дисководе того сменного диска, на котором он записан. В таблице SFT блок описания с нулевым количеством ссылок на объект тоже считается закрытым (точнее, недействительным).

При вызове:

AH = 3Eh

BX – номерная ссылка, которую надо аннулировать

При неудачном завершении флаг CF установлен, в AX – код завершения (A.06-1).
При успешном завершении флаг CF сброшен, данные в AX утрачены

Примечание 1: аннулировать номерную ссылку можно также посредством INT 21\AH=68h и INT 21\AH=6Ah; все другие спецификации вызова этих функций (кроме AH) такие же.

Примечание 2: при подготовке программы к завершению исполнения можно закрыть все открытые ею файлы сразу с помощью INT 21\AX=5D01h (8.02-69).

Примечание 3: номерные ссылки на участки расширенной памяти, выделенные драйвером EMM386.EXE (5.04-02), следует аннулировать только посредством INT 67\AH=45h (8.03-61).

8.02-35 INT 21\AH=3Fh – считывание данных из файла или из канала доступа

При вызове:

AH = 3Fh
BX – номерная ссылка на файл или на канал доступа
CX – число байтов, которые нужно считать
DS:DX – указатель на место для записи данных

При неудачном завершении флаг CF установлен, в AX – код завершения (A.06-1).
Если флаг CF сброшен, то

AX – число фактически считанных байтов
DS:DX – указатель на блок записанных данных

Примечание 1: каждая программа автоматически наследует открытую номерную ссылку 0000h, которая дает возможность считывания данных из канала STDIN, если, конечно, канал готов их предоставить. Но если канал STDIN не перенаправлен, и если в момент вызова его буфер пуст, то запускается процедура заполнения буфера с ожиданием ввода знаков с клавиатуры. Вводимые знаки одновременно отображаются на экране, их число не ограничено тем, которое предустановлено в регистре CX. Ввод прекращается по нажатию клавиши ENTER (CR), и тогда указанное в регистре CX число знаков считывается из буфера канала в адрес DS:DX. Оставшиеся в буфере знаки доступны для считывания последующими вызовами функции INT 21\AH=3Fh.

Примечание 2: считывание из файла начинается с того места, куда показывает указатель места доступа (8.02-38) в SFT (A.01-4), причем после успешного считывания положение указателя должным образом корректируется.

Примечание 3: если запрошенное число байтов в CX выводит указатель места доступа за пределы файла, то констатируется успешное завершение, но число считанных байтов в AX будет меньше числа в CX.

8.02-36 INT 21\AH=40h – посылка данных в файл или в канал доступа

При вызове:

AH = 40h

BX – номерная ссылка на файл или на канал доступа
CX – число байтов, которые надо послать.
DS:DX– указатель на буфер с данными, которые надо послать
При неудачном завершении флаг CF установлен, в AX – код завершения (A.06-1).
Если флаг CF сброшен, то
AX– – число фактически посланных байтов

Примечание 1: каждая программа автоматически наследует 4 открытые номерные ссылки для посылки данных: 0001h – в канал STDOUT, 0002h – в канал STDERR, 0003h – в канал STDAUX (порт COM1), 0004h – в канал STDPRN (порт LPT1). Канал STDOUT перенаправляемый, но по умолчанию выводит знаки на экран дисплея. Канал STDERR нельзя перенаправить, он всегда выводит знаки на экран дисплея.

Примечание 2: запись в файл начинается с того места, куда показывает указатель места доступа (8.02-38) в SFT (A.01-4), причем после успешной записи положение указателя должным образом корректируется. Если запись данных в файл выводит указатель места доступа за пределы файла, то длина файла автоматически увеличивается. Однако внесенные изменения не будут записаны на диск, пока номерная ссылка не удалена (8.02-48).

Примечание 3: если задать CX = 0, то пересылки данных не будет, но файл будет удлинен или укорочен согласно положению указателя места доступа.

Примечание 4: на дисках с файловой системой FAT-32 размер файла может быть увеличен сверх 2 Гбайт, при условии что файл открыт вызовом INT 21\AX=6C00h с установленным флагом расширенного размера.

Примечание 5: если после исполнения число посланных байтов в AX оказывается меньше заданного числа байтов в CX, то наиболее вероятной причиной этого является отсутствие свободного места на диске.

8.02-37 INT 21\AH=41h – удаление закрытого файла

При вызове:
AH = 41h
DS:DX – указатель на строку с именем файла, перед которым может быть указан путь. Строка должна кончаться байтом 00h.
При неудачном завершении флаг CF установлен, в AX – код завершения (A.06-1).
При успешном завершении флаг CF сброшен, данные в AX утрачены

Примечание 1: удаляемый файл не стирается, просто соответствующая ему запись в каталоге делается недействительной путем замены первого знака в имени файла кодом E5h.

Примечание 2: удаление с помощью INT 21\AH=41h не затрагивает те записи в каталоге, которые содержат продолжение длинного имени файла.

Примечание 3: не-закрытые файлы тоже можно удалить, но тогда не будут аннулированы относящиеся к ним номерные ссылки, и это может привести к потере данных на диске. Если подлежащий удалению файл открыт, его надо сначала закрыть посредством INT 21\AH=3Eh (8.02-34).

Примечание 4: знаки подстановки в имени файла не допускаются, за исключением вызовов INT 21\AH=41h через серверную функцию INT 21\AX=5D00h; в дополнение к тому серверная функция позволяет указать в регистре CL маску атрибутов (A.09-2) для удаляемых файлов.

Примечание 5: удаление файлов из текущего каталога возможно также посредством вызова INT 21\AH=13h (8.02-13).

8.02-38 INT 21\AH=42h – перемещение указателя места доступа

Место начала считывания или записи данных определяется указателем в SFT (15h в таблице A.01-4), который ведет счет смещения относительно начала файла. Обработчик прерывания INT 21\AH=42h позволяет изменить значение этого указателя и тем самым сместить положение точки доступа к файлу.

При вызове:

AH = 42h
AL – точка отсчета:
= 00h – начало файла
= 01h – текущее положение указателя
= 02h – конец файла
BX – номерная ссылка на файл
CX:DX – 4-байтовое смещение положения относительно точки отсчета

При неудачном завершении флаг CF установлен, в AX – код завершения (A.06-1).
При успешном завершении флаг CF сброшен, и тогда

DX:AX – новое положение точки доступа относительно начала файла.

Примечание 1:-при выборе точек отсчета 01h и 02h смещение в CX:DX должно быть числом со знаком, и тогда возможно получить указатель, показывающий на место до начала файла. В таких случаях флаг CF бывает сброшен, но ошибки возникают при попытках доступа.

Примечание 2: если новое значение указателя показывает на место за концом файла, то при первой следующей операции записи длина будет автоматически приведена в соответствие с положением указателя.

Примечание 3: вызов INT 21\AH=42h со значениями AX=4202h и CX=DX=0000h возвращает в регистрах DX:AX длину файла в байтах.

Примечание 4: для файлов на дисках с файловой системой FAT-32 допускаются значения указателя места доступа, превышающие 2048 Мбайт, но только если файл открыт посредством INT 21\AX=6C00h при установленном флаге расширенного размера.

8.02-39 INT 21\AX=4300h – определение атрибутов файла

При вызове:

AX = 4300h

DS:DX – указатель на имя файла, за которым следует байт 00h

При неудачном завершении флаг CF установлен, в AX – код завершения (A.06-1).
Если флаг CF сброшен, то

CX – атрибуты файла (A.09-2)

Примечание 1: с помощью INT 21\AX=4301h можно приписать файлу атрибуты, подготовленные в регистре CX. При возврате содержимое AX утрачивается. Все другие спецификации такие же.

8.02-40 INT 21\AX=4400h – сведения о номерной ссылке

Номерная ссылка (handle) – это предоставляемое программе шестнадцатеричное число, ассоциированное с блоком описания в таблице SFT (System File Table, A.01-4). Каждый блок описания в таблице SFT относится к объекту: либо к существующему файлу, либо к выделенной области XMS-памяти, либо к каналу доступа. Обработчик прерывания INT 21\AX=4400h дает возможность выяснить, к какому объекту относится данная ссылка, и узнать некоторые его характеристики.

При вызове:

AX = 4400h

BX – номерная ссылка

При неудачном завершении флаг CF установлен, в AX – код завершения (A.06-1).
При успешном завершении флаг CF сброшен, и тогда

DX – слово признаков (примечание 1). Состояние AX не сохраняется.

Примечание 1: если в слове признаков, возвращаемом в регистре DX, бит 7 сброшен, то его надо расшифровывать по таблице A.04-2 как относящееся к файлу, а если бит 7 установлен, то слово надо расшифровывать согласно таблице A.05-2.

Примечание 2: в слове признаков A.05-2, относящемуся к каналу доступа, состояния битов 0 – 7 можно изменять с помощью INT 21\AX=4401h. Слово признаков надо подготовить в регистре

DX, биты 8 – 15 должны быть сброшены в нуль, в AX записано число 4401h, все другие спецификации вызова такие же.

8.02-41 INT 21\AX=4402h-4403h – управление драйверами

Основная идея управления вводом-выводом (IOCTL) состоит в том, что DOS предоставляет драйверам место для размещения управляющих данных, а программам дает возможность доступа к этим данным. Миссию обеспечения такого доступа исполняют обработчики прерываний INT 21\AX=4402h-4403h. Если конкретный драйвер поддерживает программное управление вводом-выводом, то его управляющие данные адресуются номерной ссылкой, которая выдается на имя этого драйвера обработчиком прерывания INT 21\AH=3Dh или INT 21\AX=6C00h. Узнать о том, поддерживает ли драйвер такое управление, можно по состоянию битов 6 и 7 в слове признаков драйвера (A.05-2).

При вызове:

AX – функция:
= 4402h – считывание управляющих данных
= 4403h – запись управляющих данных
BX – номерная ссылка, полученная на имя драйвера.
CX – число байтов, которые надо считать или записать
DS:DX – указатель на буфер с данными или для получения данных

При неудачном завершении флаг CF установлен, в AX – код завершения (A.06-1).
Если флаг CF сброшен, то

AX – число пересланных байтов

Примечание 1: формат управляющих данных специфичен для каждого драйвера. В частности, таблица A.15-4 показывает примеры форматов команд для драйверов стандартных дисководов CD/DVD-ROM.

Примечание 2: управляющие данные для дисковых драйверов, которые нельзя адресовать номерной ссылкой, могут быть считаны с помощью INT 21\AX=4404h и записаны с помощью INT 21\AX=4405h. Эти функции используют такие же спецификации вызова, за исключением регистров AX и BX: состояние BH игнорируется, а BL адресует драйвера по номерам дисков (примечание 1 к 8.02-17).

8.02-42 INT 21\AX=4406h-4407h – готовность доступа по номерной ссылке

При вызове:

AX – функция:
= 4406h – проверка готовности к считыванию данных
= 4407h – проверка готовности к записи данных
BX – номерная ссылка (8.02-33)

При неудачном завершении флаг CF установлен, в AX – код завершения (A.06-1).
Если флаг CF сброшен, то регистр AL показывает результат:

AL = FFh – означает готовность
= 00h – неготовность к выполнению запрошенной операции.

Примечание 1: если указатель места доступа к файлу установлен на конец файла не операциями считывания или записи, а посредством INT 21\AH=42h (8.02-38), то INT 21\AX=4406h может ошибочно констатировать готовность к считыванию, возвращая AL=FFh.

Примечание 2: обработчик прерывания INT 21\AX=4407h не проверяет, вставлена ли в дисковод дискета и имеется ли на ней свободное место.

8.02-43 INT 21\AX=4408h – проверка возможности смены диска

При вызове:

AX = 4408h
BL – номер диска (примечание 1 к 8.02-17)

При неудачном завершении флаг CF установлен, в AX – код завершения (A.06-1).
Если флаг CF сброшен, то

AX = 0000h означает, что данный диск – сменный,
= 0001h – что диск несменяемый.

8.02-44 INT 21\AX=4409h – атрибуты драйвера логического диска

По атрибутам драйвера можно выяснить, является ли диск реальным или фиктивным, локальным или сетевым, доступен ли диск посредством функций BIOS или нет. Расшифровка слова атрибутов дана в таблице A.05-2.
При вызове:

AX = 4409h
BL – номер диска (примечание 1 к 8.02-17)

При неудачном завершении флаг CF установлен, в AX – код завершения (A.06-1).
Если флаг CF сброшен, то

DX – слово атрибутов (A.05-2). Состояние AX не сохраняется.

Примечание 1: биты в DX отображают способности драйвера, но не обязательно соответствуют свойствам диска, номер которого запрошен в BL. Например, установленный бит 11 (поддержка сменных дисков) означает лишь, что среди обслуживаемых данным драйвером накопителей могут быть накопители на сменных дисках. Но отсюда не следует, что запрошенный диск обязательно является сменным.

8.02-45 INT 21\AX=440Ah – проверка сетевого доступа по номерной ссылке

При вызове:

AX = 440Ah
BX – номерная ссылка (8.02-33)

При неудачном завершении флаг CF установлен, в AX – код завершения (A.06-1). Если флаг CF сброшен, то DX >= 8000h означает доступ по ссылке через сеть.

8.02-46 INT 21\AX=440Dh – родовой вызов дисковых подфункций

INT 21\AX=440Dh фактически представляет собой шаблон для вызова множества подфункций. Тип подфункции определяется кодом в регистре CL. Регистр CH при вызове содержит код категории: CH=08h соответствует дисковым подфункциям, которые унаследованы от предыдущих версий DOS и применимы к дискам с файловыми системами FAT-12 и FAT-16. Код категории CH=48h соответствует дисковым подфункциям, которые впервые введены в MS-DOS7 и позволяют обращаться к дискам с файловой системой FAT-32.

Здесь представлены только подфункции категории CH=48h, причем только те из них, которые одинаково применимы к дискам с файловыми системами FAT-12, FAT-16, FAT-32 и не требуют поддержки расширенного набора функций прерывания INT 13 (8.01-55) со стороны системы BIOS компьютера. Посредством вызова INT 21\AX=4411h (8.02-47) можно проверить, поддерживается ли конкретная подфункция Вашим компьютером.

Общая форма представления данных для всех подфункций – это блок данных, указатель на который помещается в регистры DS:DX. Распределение данных в этом блоке показано в приложении A.04 для ряда подфункций отдельно. Особые условия для некоторых подфункций оговорены ниже в примечаниях.

При вызове:

AX = 440Dh
BL – номер диска (примечание 1 к 8.02-17)
CX – подфункция:
= 4840h – задать параметры диска из таблицы (A.04-3)
= 4841h – записать дорожку логического диска (A.04-4)
= 4842h – форматировать и сверить дорожку диска (A.04-5)
= 4846h – задать серийный номер тома (A.04-1)
= 4847h – установить флаг доступа (примечание 1)
= 4848h – установить состояние блокировки (примечание 2)
= 4849h – вытолкнуть диск из дисковода (без блока данных)
= 4860h – считать таблицу параметров диска (A.04-3)
= 4861h – считать дорожку логического диска (A.04-4)
= 4862h – сверить дорожку логического диска (A.04-5)

= 4866h – считать метку тома и тип FAT (A.04-1)
= 4867h – считать флаг доступа (примечание 1)
= 4868h – определить тип дискеты (примечание 3)
DS:DX – указатель на блок исходных данных, если он требуется

При неудачном завершении флаг CF установлен, в AX – код завершения (A.06-1).
Если флаг CF сброшен, и если подфункция должна возвращать данные, то
DS:DX – указатель на блок возвращаемых данных.

Примечание 1: подфункция CX=4847h принимает, а подфункция 4867h возвращает в блоке данных DS:DX только один байт доступа (смещение 01h). Любое ненулевое значение означает, что доступ разрешен.

Примечание 2: в блоке данных в байте со смещением 00h подфункция CL=48h принимает код операции: 00h – заблокировать диск, 01h – разблокировать, 02h – доложить статус блокировки (так же, как INT 13\AH=45h, 8.01-58). Статус выражает уровень вложенности блокирования и возвращается в байте 01h блока данных.

Примечание 3: в блоке данных в байте со смещением 00h подфункция CL=4868h возвращает 01h, если дискета соответствует типу дисковода, или 00h при всех других типах носителей, а в байте со смещением 01h возвращает код типа дискеты: 02h – дискета 720 кбайт, 07h – дискета 1.44 Мбайт, 09h – дискета 2.88 Мбайт.

8.02-47 INT 21\AX=4411h – поддержка функций родового вызова

Обработчик прерывания INT 21\AX=4411h сообщает, поддерживается ли указанная подфункция родового вызова системой BIOS, аппаратными средствами и загруженными драйверами в конкретном компьютере.

При вызове:
AX = 4411h
BL – номер диска (примечание 1 к 8.02-17)
CX – код подфункции, как для INT 21\AX=440Dh (8.02-46).
При неудачном завершении флаг CF установлен, в AL – код завершения (A.06-1).
Если флаг CF сброшен, и AX=0000h, то функция поддерживается

Примечание 1: поддержку самой проверки посредством INT 21\AX=4411h, как и других функций управления вводом-выводом, можно определить по слову признаков (A.05-2) соответствующего драйвера.

8.02-48 INT 21\AH=45h\46h – выдача дубликата номерной ссылки

Обработчик прерывания INT 21\AH=45h копирует указатель из одной ячейки таблицы JFT (примечание 3 к таблице A.07-1) в ближайшую свободную ячейку той

же таблицы JFT. Номера обеих этих ячеек становятся ссылками на один и тот же объект. Ссылку-дубликат создают для перераспределения указателей, для сохранения указателя, а иногда только для того, чтобы ее аннулировать, вызывая тем самым запись файла из памяти на диск без закрытия файла.

Вызов INT 21\AH=46h тоже выполняет копирование указателя, но помещает копию в заданную программистом ячейку таблицы JFT, замещая там прежний указатель. Например, если номерную ссылку 0001h, относящуюся к каналу STDOUT, сделать дубликатом номерной ссылки на файл, то вместо вывода данных на экран дисплея будет происходить запись данных в этот файл. Именно так DOS осуществляет перенаправление каналов ввода и вывода (2.04-02 – 2.04-05).

При вызове:

AH = 45h – дублирование без замещения имеющихся ссылок
= 46h – дублирование с замещением заданного указателя
BX – действующая номерная ссылка, которую надо продублировать
CX – номерная ссылка, которая должна стать дубликатом другой ссылки на тот же объект (только для вызова с AH=46h)

При неудачном завершении флаг CF установлен, в AL – код завершения (A.06-1).
Если флаг CF сброшен, то миссия выполнена успешно, и тогда

AX – автоматически назначенный дубликат номерной ссылки (только после вызова с AH=45h).

Примечание 1: если при вызове INT 21\AH=46h в регистре CX указана действующая ссылка на другой открытый файл, то он автоматически будет закрыт.

Примечание 2: смещение места доступа по одной номерной ссылке влечет такое же смещение по дублирующей ссылке, потому что соответствующие им ячейки таблицы JFT содержат указатели на один и тот же блок описания в таблице SFT (A.01-4).

8.02-49 INT 21\AH=47h – определение текущего каталога

При вызове:

AH = 47h
DL – номер диска (примечание 1 к 8.02-17)
DS:SI – адрес 64-байтового буфера для пути и имени каталога

При неудачном завершении флаг CF установлен, в AL – код завершения (A.06-1).
Если флаг CF сброшен, содержимое AX утрачено, буфер DS:SI заполнен; конец записи в буфере отмечен байтом 00h.

Примечание 1: возвращаемый в буфере путь не содержит имени диска и начальной обратной косой черты.

Примечание 2: изменить текущий каталог можно посредством INT 21\AH=3Bh (8.02-32).

8.02-50 INT 21\AH=48h – выделение блока памяти для программы

При вызове:

AH = 48h
BX – размер запрашиваемого блока в 16-байтовых параграфах

При неудачном завершении флаг CF установлен, в AL – код завершения (A.06-1).
Если флаг CF сброшен, то

AX – сегментный адрес выделенного блока
BX – размер наибольшего свободного блока в памяти.

Примечание 1: запрос блока размером BX=FFFFh обречен на неудачу, но при этом обработчик прерывания INT 21\AH=48h возвратит в регистре BX размер всего свободного пространства обыкновенной памяти.

Примечание 2: DOS может отвечать отказом с кодом завершения 08h ("имеющейся памяти недостаточно") на любой запрос о выделении памяти во время исполнения программ с суффиксом *.COM, потому что по умолчанию таким программам предоставляется все имеющееся пространство памяти (подробнее – в примечании 5 к A.12-7). Программы с суффиксом *.COM должны декларировать размер используемой ими памяти посредством INT 21\AH=4Ah, иначе все остальное пространство памяти DOS не будет считать свободным.

8.02-51 INT 21\AH=49h – высвобождение блока памяти

При вызове:

AH = 49h
ES – сегментный адрес высвобождаемого блока

При неудачном завершении флаг CF установлен, в AL – код завершения (A.06-1).
Если флаг CF сброшен, то миссия завершена успешно.

8.02-52 INT 21\AH=4Ah – изменение размера выделенного блока памяти

При вызове:

AH = 4Ah
BX – новый размер блока в 16-байтовых параграфах
ES – сегментный адрес блока, размер которого надо изменить

При неудачном завершении флаг CF установлен, в AL – код завершения (A.06-1).
Если флаг CF сброшен, то миссия завершена успешно, и тогда

BX – максимальное число параграфов, которое можно предоставить

Примечание 1: если памяти недостаточно, и запрошенное число параграфов предоставить нельзя, то блок будет увеличен настолько, насколько это возможно.

8.02-53 INT 21\AH=4Bh – загрузка программы для исполнения

При вызове:

AH = 4Bh
AL = 00h – загрузка и запуск программы на исполнение
= 01h – загрузка без инициализации исполнения
= 03h – загрузка оверлея (заменяемого фрагмента кода)
DS:DX – указатель на строку с полной спецификацией вызова программы. Имя программы должно быть приведено с суффиксом. Строка должна кончаться байтом 00h.
ES:BX – указатель на блок параметров из таблицы A.07-2.

При неудачном завершении флаг CF установлен, в AL – код завершения (A.06-1).
При успешном завершении флаг CF сброшен; данные в BX и DX утрачены.

Примечание 1: подфункция AL=00h копирует окружение вызывающей программы в сегмент окружения загружаемой программы и создает для нее новый PSP (A.07-1), причем в нем строка вызова заканчивается байтом 00h (а не байтом 0Dh, как обычно).

Примечание 2: миссия проверки наличия достаточной свободной памяти для загружаемой программы возложена на вызывающий процесс.

Примечание 3: подфункция AL=01h обслуживает загрузку программы точно так же, но не инициирует ее исполнение. Чтобы позже это смогла сделать вызывающая программа, ей в составе того же исходного блока данных (A.07-2) подфункция AL=01h возвращает адрес вызова загруженной программы и указатель на вершину ее стека.

Примечание 4: если программа начинается с байтов MZ или ZM, то она ставится на исполнение как файл формата *.EXE. Чтобы программа была поставлена на исполнение как файл *.COM, ее первыми байтами не должны быть MZ, NE, LE, LX, W3, W4, PE, PL, MP, P2, P3, ZM.

Примечание 5: чтобы посредством INT 21\AH=4Bh исполнить batch-файл, нужно загрузить интерпретатор COMMAND.COM (6.04) с параметром /C, указав после него имя подлежащего исполнению batch-файла.

Примечание 6: подфункция AL=03h загружает в память оверлей, не проверяя принадлежность выделенной памяти вызывающей программе. При этом окружение и префикс сегмента программы не создаются, исполнение загруженного оверлея не начинается, и состав параметров в блоке, на который указывают регистры ES:BX, должен быть другим: первое слово – сегментный адрес загрузки оверлея, слово со смещением 02h – смещение оверлея в сегменте.

8.02-54 INT 21\AX=4B05h – задание условий исполнения

Функция INT 21\AX=4B05h используется обработчиками, которые перехватывают вызовы INT 21\AX=4B00h чтобы подготовить загруженную программу к исполнению, включая указание требуемого номера версии DOS.

При вызове:

AX = 4B05h

DS:DX – указатель на дескриптор условий исполнения A.07-3

При неудачном завершении флаг CF установлен, в AL – код завершения (A.06-1).
При удачном завершении флаг CF сброшен, в AX записано число 0000h

Примечание 1: между вызовом INT 21\AX=4B05h и моментом передачи управления загруженной программе не допускаются вызовы никаких прерываний.

Примечание 2: если DOS загружена в область верхней памяти (HMA), то при возврате из INT 21\AX=4B05h линия A20 отключается.

8.02-55 INT 21\AH=4Ch – завершение исполнения с указанием уровня ошибки

При вызове:

AH = 4Ch

AL – шестнадцатеричный код уровня ошибки (примечание 3)

При возврате:

возврата не будет

Примечание 1: перед вызовом INT 21\AH=4Ch все установленные программой сетевые блокировки должны быть сняты.

Примечание 2: вызов INT 21\AH=4Ch закрывает все открытые процессом файлы и высвобождает всю используемую процессом память, но при условии, что указатель на родительский процесс в PSP (смещение 16h в таблице A.07-1) не указывает на сегмент того же PSP. Последнее есть признак невыгружаемой резидентной программы; ею может быть, например, командный интерпретатор COMMAND.COM.

Примечание 3: код уровня ошибки (errorlevel) записывается в слово со смещением 14h в области текущих данных DOS, показанной в таблице A.01-3. Он может быть считан функцией INT 21\AH=4Dh (8.02-56) или проверен операцией "if errorlevel..." (3.15-03).

Примечание 4: в среде отладчика Debug.exe INT 21\AH=4Ch закрывает сеанс работы с отладчиком и передает управление DOS. Если сеанс должен быть продолжен, то завершать исполнение отлаживаемой программы надо иначе, например, посредством точек останова или вызовом INT 20 (8.02-01).

8.02-56 INT 21\AH=4Dh – считывание кода уровня ошибки

При вызове:

AH = 4Dh

При возврате:

AH – тип завершения:
= 00h – нормальный (8.02-01, 8.02-55)
= 01h – по нажатию Ctrl-C, Ctrl-Break (8.01-95, 8.02-83)
= 02h – по возникновению критической ошибки (8.02-84)
= 03h – с оставлением TSR-модуля (8.02-23, 8.02-86).

AL – шестнадцатеричное значение кода уровня ошибки (errorlevel).

Примечание 1: код уровня ошибки характеризует завершение предыдущей программы, за исключением резидентных программ и программ, исполняемых в фоновом режиме.

Примечание 2: код уровня ошибки находится в слове со смещением 14h в области текущих данных DOS (A.01-3), причем после вызова INT 21\AH=4Dh он автоматически обнуляется, так что считать его дважды нельзя. Многократное считывание кода уровня ошибки возможно из batch-файлов (3.15-03).

Примечание 3: встроенные команды интерпретатора COMMAND.COM не оставляют кода уровня ошибки и не изменяют значение, оставленное после исполнения предшествовавшей программы.

8.02-57 INT 21\AH=4Eh – поиск первого подходящего файла

Обработчик прерывания INT 21\AH=4Eh оставляет найденные сведения в области DTA, которая по умолчанию находится в префиксе сегмента программы (PSP, A.07-1) начиная со смещения 0080h. Расположение области DTA можно изменять и определять посредством INT 21\AH=1Ah,2Fh (8.02-16). Формат возвращаемых в DTA данных показан в таблице A.09-1 (колонка F4E).

При вызове:

AX = 4E00h
CH = 00h
CL – маска атрибутов файла, показанная в таблице A.09-2.
DS:DX – указатель на строку с именем искомого файла. Имени может предшествовать путь. В имени допускается наличие знаков подстановки. Строка должна кончаться байтом 00h.

При неудачном завершении флаг CF установлен, в AL – код завершения (A.06-1). Если флаг CF сброшен, то в DTA записан блок данных (A.09-1).

Примечание 1: биты 0 и 5 в маске атрибутов файла (A.09-2) игнорируются. Установление битов 1, 2 и 4 в маске атрибутов файла не исключает

поиска файлов, не имеющих соответствующих атрибутов. Установление бита 3 (метка тома) исключает поиск файлов.

Примечание 2: поиск в текущем каталоге возможен также с помощью INT 21\AH=11h.

8.02-58 INT 21\AH=4Fh – поиск следующего подходящего файла

При вызове:

AH = 4Fh

В области DTA – сохраненные результаты предыдущего поиска

При неудачном завершении флаг CF установлен, в AL – код завершения (A.06-1).
Если флаг CF сброшен, то в DTA обновлен блок данных (A.09-1).

Примечание 1: для продолжения поиска обработчик прерывания INT 21\AH=4Fh использует данные, полученные в результате предшествующего вызова INT 21\AH=4Eh или INT 21\AH=4Fh, и сохраняемые с того момента без изменений в области DTA (8.02-16). Во избежание возникновения различий между этими данными и реальным расположением файлов нельзя выполнять операции переименования, перемещения или удаления файлов, пока попытки продолжения поиска не прекращены.

8.02-59 INT 21\AH=52h – адрес "списка списков"

При вызове:

AH = 52h

При возврате:

ES:BX – указатель на "список списков" (A.01-2)

Примечание 1: возвращаемый сегментный адрес в ES указывает на блок данных DOS, создаваемый загрузчиком IO.SYS.

Примечание 2: если прикладная программа исполняется не в реальной, а в эмулируемой среде DOS, то надо быть готовым к тому, чтобы получить и отвергнуть заведомо недействительные значения в ES:BX, например, 0000:0000h или FFFF:FFFFh.

8.02-60 INT 21\AH=54h – считывание состояния флага сверки

При вызове:

AH = 54h

При возврате:

AL = 00h – флаг сверки сброшен (OFF),
= 01h – флаг сверки установлен (ON)

Примечание 1: при установленном флаге сверки после каждой операции записи на диск выполняется сверка сигналограммы (подробнее – в разделе 3.33). По умолчанию флаг сверки сброшен.

Примечание 2: задать новое состояние флага сверки можно посредством вызова INT 21\AH=2Eh, причем перед вызовом новое состояние должно быть указано в AL так же, как его возвращает INT 21\AH=54h.

8.02-61 INT 21\AH=55h – создание производного PSP

Обработчик прерывания INT 21\AH=55h создает производный (дочерний) префикс сегмента программы (PSP, A.07-1), в котором в поле адреса родительского PSP указан адрес PSP вызывающей программы. Помимо того, в новом PSP заполняются поля адресов обработчиков прерываний (INT 22, 23, 24), в поля таблицы JFT, соответствующие наследуемым номерным ссылкам, вносятся номера блоков описания таблицы SFT (A.01-4), а в самих этих блоках описания счетчики ссылок получают приращение на единицу. Созданный PSP готов к тому, чтобы обслужить исполнение COM-файла.

При вызове:

AH = 55h
DX – сегментный адрес, начиная с которого надо разместить PSP
SI – объем памяти для записи в поле со смещением 02h в PSP

При возврате содержимое AL не сохраняется.

Примечание 1: до обращения к INT 21\AH=55h необходимо побеспокоиться о том, чтобы сегмент, где будет размещен новый PSP, был бы выделен операционной системой посредством INT 21\AH=48h (8.02-50). Подлежащий исполнению COM-файл надо будет скопировать в этот сегмент начиная со смещения 100h. После этого процедура передачи управления включает переопределение идентификатора текущего процесса с помощью INT 21\AH=50h (примечание к 8.02-73) и дальний переход на адрес 100h относительно начала созданного PSP.

8.02-62 INT 21\AH=56h – корректировка записей в каталогах

Посредством корректировки записей в каталогах можно выполнять переименование файлов и каталогов, а также перемещение файлов из одного каталога в другой на том же диске. Перемещение записей о файлах из одного каталога в другой выполняется гораздо быстрее, чем копирование файлов.

При вызове:

AH = 56h

DS:DX – указатель на строку с именем существующего объекта, файла или каталога (примечание 4). Имени может предшествовать путь. Строка должна кончаться байтом 00h.

ES:DI – указатель на строку с новым именем (примечание 4) или с другим путем. Строка должна кончаться байтом 00h.

При неудачном завершении флаг CF установлен, в AL – код завершения (A.06-1).
Если флаг CF сброшен, то миссия завершена успешно.

Примечание 1: если DS:DX указывает на файл, а ES:DI – на существующий каталог того же диска, то соответствующая файлу запись будет перемещена в этот каталог без копирования файла.

Примечание 2: если ES:DI указывает на новое имя, а DS:DX – на существующий объект – закрытый файл или каталог, то этот объект будет переименован. Но переименование открытых файлов не допускается.

Примечание 3: обработчик прерывания INT 21\AH=56h не присваивает перемещенным и переименованным файлам атрибут "A" (подлежит архивированию).

Примечание 4: чтобы переименовать группу файлов, используя в именах знаки подстановки (2.01-03), необходимо вызывать INT 21\AH=56h через серверную функцию INT 21\A=5D00h (8.02-68). Помимо знаков подстановки, серверная функция принимает в регистре CL маску атрибутов (A.09-2), и в случае успешного завершения возвращает AL=12h (= подходящих файлов больше нет).

8.02-63 INT 21\AX=5700h – время и дата последнего изменения файла

При вызове:

AX = 5700h

BX – номерная ссылка на файл (8.02-33)

При неудачном завершении флаг CF установлен, в AL – код завершения (A.06-1).
Если флаг CF сброшен, то:

CX – время последнего изменения файла, причем
биты 15-11 – часы от 00 до 23;
биты 10-5 – минуты;
биты 4-0 – секунды.

DX – дата последнего изменения файла, причем
биты 15-9 – год, отсчитываемый от 1980-го;
биты 8-5 – месяц;
биты 4-0 – день.

Примечание 1: переустановить время и дату последнего изменения файла можно с помощью INT 21\AX=5701h: задаваемые значения принимаются из регистров CX и DX в той же форме.

Примечание 2: если командой ACCDATE (4.01) не заблокировано обслуживание сведений о времени создания файлов, то эти сведения можно аналогичным образом получить с помощью INT 21\AX=5706h и задать с помощью INT 21\AX=5707h. Дополнительно эти функции используют регистр SI для кода времени в 10-миллисекундных единицах.

Примечание 3: если командой ACCDATE (4.01) не заблокировано обслуживание сведений о последней дате доступа к файлам, то эту дату можно получить в DX с помощью INT 21\AX=5704h и задать с помощью INT 21\AX=5705h, причем при вызове надо задавать CX=0000h.

8.02-64 INT 21\AX=5800h – стратегия выделения памяти

При вызове:

AX = 5800h

При неудачном завершении флаг CF установлен, в AL – код завершения (A.06-1).
Если флаг CF сброшен, то

AX – код принятой стратегии, причем в этом слове

биты 7 и 6:

= 00 – использовать обыкновенную память;
= 01 – использовать расширенную память;
= 10 – использовать обыкновенную память, только если расширенная недоступна;

биты 1 и 0:

= 00 – выделять первый подходящий участок;
= 01 – выделять участки оптимального размера;
= 10 – выделять последний подходящий участок.

биты 15-8 и 5-2 должны быть сброшены в нуль.

Примечание 1: обработчик прерывания INT 21\AX=5801h принимает в регистре BX такой же код стратегии и позволяет его переустановить.

Примечание 2: каждая программа, которая изменяет стратегию выделения памяти, при завершении исполнения должна восстановить прежнюю стратегию.

Примечание 3: для пользования расширенной памятью задание соответствующей стратегии необходимо, но не достаточно: нужно еще в файле CONFIG.SYS ввести команду DOS=UMB (4.08).

8.02-65 INT 21\AH=59h – расширенные сведения об ошибке

При вызове:

AH = 59h
BX = 0000h

При возврате: содержимое регистров CL, DX, SI, BP и DS утрачивается,

BH – класс ошибки из таблицы A.06-2
BL – рекомендуемые действия из таблицы A.06-3
CH – место возникновения ошибки из таблицы A.06-4
AX – код завершения из таблицы A.06-1; если AX=0022h, то
ES:DI – адрес идентификатора (примечание 2 к A.06-1).

Примечание 1: предоставляемые сведения считываются из области текущих данных DOS (A.01-3).

Примечание 2: сведения об ошибке можно записать в область текущих данных DOS с помощью INT 21\AX=5D0Ah; перед вызовом надо указать в DS:DX адрес блока данных (A.07-4), из которого будут взяты значения для регистров AX, BX, CX, DX, DI и ES.

8.02-66 INT 21\AH=5Ah – создание и открытие временного файла

При вызове:

AH = 5Ah
CX – атрибуты файла согласно таблице A.09-2
DS:DX – указатель на строку с путем, заканчивающимся знаком обратной косой черты "\", за которым следуют 13 байтов 00h – это место для автоматически формируемого имени.

При неудачном завершении флаг CF установлен, в AL – код завершения (A.06-1).
Если флаг CF сброшен, то:

AX – номерная ссылка на созданный файл;
DS:DX– указатель на строку с путем и именем файла.

Примечание 1: присваиваемое файлу имя уникально, конфликт имен исключается.

Примечание 2: корневые каталоги дисков имеют ограниченный объем, и когда этот объем заполнен, файл в корневом каталоге создать нельзя.

Примечание 3: к моменту завершения работы программы, создавшей временный файл, этот файл должен быть закрыт и удален.

8.02-67 INT 21\AH=5Bh – создание и открытие нового файла

При вызове:

AH = 5Bh
CX – атрибуты файла согласно таблице A.09-2

DS:DX – указатель на строку с именем файла. Перед именем может быть указан путь. Строка должна кончаться байтом 00h.

При неудачном завершении флаг CF установлен, в AL – код завершения (A.06-1).
Если флаг CF сброшен, то состояние DS:DX сохранено, и тогда

AX – номерная ссылка на созданный файл.

Примечание 1: обработчик прерывания INT 21\AH=5Bh не создаст файл, если файл с таким же именем в указанном каталоге уже существует.

Примечание 2: корневые каталоги дисков имеют ограниченный объем, и когда этот объем заполнен, файл в корневом каталоге создать нельзя.

Примечание 3: обработчик прерывания INT 21\AH=3Ch делает то же самое на базе тех же спецификаций (кроме AH), но при наличии одноименного файла в каталоге назначения не сообщает об ошибке, а "обрезает" этот файл до нулевой длины. В результате пропадает ссылка на первый кластер файла, и восстановить его оказывается намного сложнее, чем после обычного удаления с помощью INT 21\AH=13h (8.02-13) или INT 21\AH=41h (8.02-37).

8.02-68 INT 21\AX=5D00h – серверное обслуживание вызовов

Обработчик прерывания INT 21\AX=5D00h предоставляет шаблон, по которому можно вызвать любую функцию прерывания INT 21, причем эта функция будет исполняться как отдельный процесс, включая возможности избирательного и многократного исполнения. В частности, при серверном вызове INT 21\AH=3Dh предоставляется возможность задать маску атрибутов (A.07-2) для открываемого файла, а при серверном вызове операций переименования (INT 21\AH=56h) и удаления (INT 21\AH=41h) – возможность переименовывать и удалять группы файлов по маске имени с использованием знаков подстановки (2.01-03).

При вызове:

AX = 5D00h

DS:DX – указатель на блок данных, показанный в таблице A.07-4; там записываются состояния всех регистров, подготовленные для вызова запрашиваемой функции.

При возврате состояние определяется той функцией, вызов которой был запрошен.

Примечание 1: правильность исходных данных при серверном вызове не проверяется. Вызов несуществующей функции из-за неверного значения для регистра AH обычно ведет к зависанию компьютера.

Примечание 2: передаваемые по ссылкам имена надо заранее перевести в каноническую форму с помощью INT 21\AH=60h (8.02-72).

8.02-69 INT 21\AX=5D01h – закрытие всех файлов указанного процесса

Обработчик прерывания INT 21\AX=5D01h записывает из буферов на диск все данные, относящиеся к закрываемым файлам, и обновляет соответствующие записи в каталогах. Если остаются открытыми файлы, доступ к которым осуществляется по сети, то производится обращение к обработчику прерывания INT 2F\AX=1107h.

При вызове:

AX = 5D01h

DS:DX – указатель на блок данных, показанный в таблице A.07-4, причем там принимаются во внимание только идентификаторы компьютера и процесса (смещения 12h и 14h), а значения для всех регистров игнорируются.

При неудачном завершении флаг CF установлен, в AX – код завершения (A.06-1). Если флаг CF сброшен, то миссия завершена успешно.

8.02-70 INT 21\AX=5D06h – адрес области SDA

Функции DOS записывают данные об исполняемом процессе, которые бывают нужны тем же или другим функциям DOS, вызываемым позднее из того же процесса. Однако исполняемый процесс может быть прерван другим процессом – вызванной резидентной программой или обработчиком прерывания. Функции DOS, вызываемые этим другим процессом, запишут данные о другом исполняемом процессе, искажая прежние данные, которые относились к прерванному процессу. Эти искажения воспрепятствуют правильному возобновлению прерванного процесса. Во избежание подобных конфликтов необходимые данные надо копировать в безопасное место и потом восстанавливать. Поэтому область записи данных, необходимых для возобновления прерванного процесса, названа Swappable Data Area (SDA), то есть областью перекачиваемых данных. Элементы структуры данных в области SDA показаны в таблице A.01-3. Обработчик прерывания INT 21\AX=5D06h предоставляет адрес области SDA и сведения о том, какую ее часть в каком случае надо сохранять.

При вызове:

AX = 5D06h

При неудачном завершении флаг CF установлен, в AX – код завершения (A.06-1). Если флаг CF сброшен, то:

DS:SI – указатель на начало области SDA (A.01-3),

CX – размер в байтах всей области SDA, включая сведения о процессе и стеки DOS. Всю область SDA надо сохранять для правильного возобновления прерванной функции DOS.

DX – размер в байтах части области SDA со сведениями о процессе. Эту часть надо сохранять, когда прерванный процесс не является функцией DOS.

Примечание 1: сохранение данных из SDA не требуется, если прерывающий процесс не пользуется вызываемыми функциями DOS.

Примечание 2: INT 21\AX=5D06h – это тоже функция DOS, и ее опасно вызывать, когда прерывание уже произошло, а данные из области SDA еще не сохранены. Поэтому рекомендуется вызывать INT 21\AX=5D06h заранее, во время инициализации резидентной программы или драйвера, чтобы возвращенными сведениями можно было бы пользоваться потом в любой момент.

8.02-71 INT 21\AX=5F08h – скрыть логический диск

При вызове:

AX = 5F08h

DL – номер диска (примечание 1 к 8.02-10)

При неудачном завершении флаг CF установлен, в AX – код завершения (A.06-1). Если флаг CF сброшен, то миссия завершена успешно.

Примечание 1: обработчик прерывания INT 21\AX=5F08h вносит изменения не в разметку физического диска, а только в структуры таблиц DOS (A.03), проверяя при этом наличие данных, обеспечивающих возможность восстановления доступности диска в дальнейшем. Если диск неправильно зарегистрирован или недействителен, то миссия INT 21\AX=5F08h завершается неудачно.

Примечание 2: обработчик прерывания INT 21\AX=5F07h принимает такие же спецификации вызова (кроме AX), но выполняет обратную операцию: снова делает действительным скрытый логический диск.

8.02-72 INT 21\AH=60h – приведение пути и имени в канонический вид

Действие обработчика прерывания INT 21\AH=60h идентично действию команды TRUENAME, описанному в разделе 3.29. Преобразование спецификаций пути и имени в канонический вид является условием успешного выполнения многих функций DOS и позволяет заранее выявить ошибки, которые потом могли бы привести к нежелательным последствиям.

При вызове:

AH = 60h

DS:SI – указатель на строку до 64 байт с именем, которому может предшествовать путь. Строка должна кончаться байтом 00h.

ES:DI – указатель на 128-байтовый буфер для размещения результата преобразования.

При успешном завершении флаг CF сброшен, данные в AX утрачены, строка приведена в канонический вид и записана в подготовленный буфер.

При неудаче флаг CF установлен, буфер не изменен, в AX– код завершения:

AX = 0002h – элемент спецификации неправилен или отсутствует;
= 0003h – ошибка композиции или неверная буква диска.

Примечание 1: существование предъявленных пути и имени не проверяется.
Примечание 2: нельзя применять INT 21\AH=60h по отношению к сетевым путям.

8.02-73 INT 21\AH=62h – сегментный адрес PSP исполняемой программы

DOS использует сегментный адрес PSP в качестве идентификатора того процесса, который исполняется компьютером в данный момент. Смена идентификатора – ключевая операция для осуществления многозадачного режима исполнения программ, она включает считывание текущего адреса PSP из SDA (A.01-3) с помощью INT 21\AH=62h, запоминание его, и запись в SDA нового значения. По завершении миссии процесса управление должно быть передано назад прерванной программе с восстановлением прежнего значения идентификатора в SDA. В ряде других ситуаций также приходится обращаться к INT 21\AH=62h для выяснения истинного адреса PSP (пример – в разделе 9.07-02).

При вызове:

AH = 62h

При возврате:

BX – сегментный адрес PSP исполняемой программы.

Примечание 1: новое значение сегментного адреса PSP записывается в SDA с помощью INT 21\AH=50h, принимающего это новое значение из регистра BX. Помимо AH=50h и BX, другие исходные данные не требуются.

8.02-74 INT 21\AX=6501h – сведения о национальной адаптации

При вызове:

AX = 6501h
BX – кодовая страница в шестнадцатеричной форме, или иначе
= FFFFh – запрос сведений об установленной кодовой странице
CX – размер буфера для блока данных, не менее 29h байт
DX – идентификатор страны, или иначе
= FFFFh – для страны, под которую DOS адаптирована сейчас
ES:DI – указатель на буфер для блока данных, не менее 29h байт

При неудачном завершении флаг CF установлен, в AX – код завершения (A.06-1).
При успешном завершении флаг CF сброшен, и тогда

ES:DI – указатель на буфер, заполненный блоком данных (A.02-4)
CX – размер записанного в буфер блока данных.

Примечание 1: вызовы INT 21\AX=6502h, 6504h, 6505h, 6506h используют те же исходные данные (кроме AX), но в буфере ES:DI возвращают только один 4-байтовый указатель, записываемый начиная со смещения 01h:

INT 21\AX=6502h возвращает указатель на таблицу соответствия заглавных букв, в первом слове которой записан ее размер, а далее следуют 128 заглавных эквивалентов для знаков с номерами от 80h до FFh.

INT 21\AX=6504h возвращает указатель на таблицу соответствий, которая имеет такую же структуру, но применяется только по отношению к именам файлов.

INT 21\AX=6505h возвращает указатель на таблицу ограничений для имен файлов (A.02-5).

INT 21\AX=6506h возвращает указатель на таблицу, в первом слове которой записан ее размер, а далее следуют 256 байтов, задающих порядок сортировки для знаков с номерами от 00h до FFh.

Примечание 2: сведения, касающиеся других стран и других кодовых страниц, которые не установлены в данный момент, не выдаются, если не установлена резидентная программа NLSFUNC.EXE (5.02-03).

Примечание 3: параметры адаптации можно записать заново посредством INT 21\AX=7002h, принимающего в DS:SI указатель на таблицу A.02-4 с новыми данными, а в CX – длину этой таблицы, обычно 0026h байт. При успешном завершении флаг CF сброшен, а в CX – количество записанных байт. При неудачном завершении флаг CF установлен, AX = 7000h означает, что функция не поддерживается, а другие значения кода завершения соответствуют таблице A.06-1.

8.02-75 INT 21\AX=6521h – перевод национальных букв в заглавные

При вызове:

AX = 6521h
CX – длина преобразуемой строки
DS:DX – указатель на строку, которую надо преобразовать

При неудачном завершении флаг CF установлен, в AX – код завершения (A.06-1).
При успешном завершении флаг CF сброшен, строка преобразована.

Примечание 1: INT 21\AX=6522h делает то же самое, но игнорирует число в CX и требует, чтобы конец строки был отмечен байтом 00h.

Примечание 2: для перевода одного знака используют INT 21\AX=6520h. Код знака принимается и возвращается в DL, данные в CX и DS игнорируются.

8.02-76 INT 21\AH=67h – изменение размера таблицы номерных ссылок

По умолчанию программа может пользоваться одновременно не более чем двадцатью номерными ссылками: это число определяется длиной таблицы JFT (смещение 18h в таблице A.07-1). Обработчик прерывания INT 21\AH=67h создает таблицу JFT произвольной длины за пределами префикса сегмента программы (PSP), снимая тем самым ограничение на количество номерных ссылок, вызываемое стандартным размещением таблицы JFT.

При вызове:

AH = 67h

BX – предельное количество используемых номерных ссылок

При неудачном завершении флаг CF установлен, в AX – код завершения (A.06-1). Если флаг CF сброшен, то миссия завершена успешно.

Примечание 1: если таблица JFT располагается в PSP, и задаваемое число в BX не превосходит 20, то никакие действия не предпринимаются.

Примечание 2: если таблица JFT уже выведена за пределы PSP, содержит не более 20 номерных ссылок, и задаваемое в BX значение также не превышает 20, тогда таблица JFT копируется обратно в PSP.

Примечание 3: если имеющиеся в таблице JFT номерные ссылки не вмещаются в тот уменьшенный размер, который задан в регистре BX, то миссия кончается неудачно с AX=0004h (открыто слишком много файлов).

Примечание 4: число открываемых файлов ограничено не только таблицей JFT, но также и таблицей SFT (A.01-4), длина которой определяется командой FILES (4.12) в конфигурационном файле CONFIG.SYS.

Примечание 5: при любом размере и расположении таблицы JFT производные (дочерние) процессы наследуют не более 20 номерных ссылок.

8.02-77 INT 21\AX=6900h – считывание метки тома и типа FAT

Вызов INT 21\AX=6900h возвращает такую же таблицу A.04-1, какую возвращает INT 21\AX=440Dh\CX=4866h, но только не производит обращения к обработчику критической ошибки в случае неудачи считывания данных с носителя. Все возможные ошибки будут отражены кодом завершения в регистре AX.

При вызове:

AX = 6900h
BH = 00h
BL – номер диска (примечание 1 к 8.02-17)
DS:DX – указатель на 26-байтовый буфер для блока данных

При неудачном завершении флаг CF установлен, в AX – код завершения (A.06-1).
Если флаг CF сброшен, то данные в AH утрачены, и

DS:DX – указатель на буфер с блоком данных (A.04-1).

Примечание 1: возвращаемый блок данных, начиная со смещения 02h, копируется из байтов 27h – 3Dh расширенной таблицы BPB (A.03-4).

Примечание 2: возвращаемое значение 0005h кода завершения означает, что расширенная таблица BPB для данного диска отсутствует.

Примечание 3: с помощью INT 21\AX=6900h нельзя запрашивать данные о дисках, доступ к которым осуществляется по сети (код завершения 0001h).

Примечание 4: обработчик прерывания INT 21\AX=6901h выполняет обратную операцию: он принимает в DS:DX указатель на блок данных A.04-1 и записывает эти данные в расширенную таблицу BPB. Все другие спецификации вызова (кроме AX) такие же.

8.02-78 INT 21\AX=6C00h – открытие файла для доступа

Номерная ссылка для доступа к объекту может быть получена посредством любой из двух функций: INT 21\AH=3Dh (8.02-33) и INT 21\AX=6C00h, однако последняя предоставляет расширенные возможности задать свойства номерной ссылки и предписываемые действия.

При вызове:

AX = 6C00h
BH – флаги:
  бит 4 – разрешить размер сверх 2 Гбайт (для FAT-32)
  бит 5 – сообщать об ошибке, не вызывая INT 24h
  бит 6 – исполнять запись сразу, без буферирования
BL – условия доступа согласно таблице A.09-4
CX – атрибуты файла (A.09-2), если его предстоит создать
DH = 00h
DL – предписываемое действие:
  = 01h – открыть существующий файл, а если такой файл не существует, то сообщить об ошибке;
  = 10h – открыть новый файл, а если одноименный файл уже существует, то сообщить об ошибке;
  = 11h – открыть файл; если такого файла нет, то создать его;

= 12h – открыть новый файл, а если одноименный файл уже существует, то удалить его и создать заново.

DS:SI – указатель на строку с именем файла, которому может предшествовать путь. Знаки подстановки в имени файла не допускаются. Строка должна кончаться байтом 00h.

При неудачном завершении флаг CF установлен, в AX – код завершения (A.06-1).
Если флаг CF при завершении сброшен, то:

AX – номерная ссылка на открытый файл

CX – результат действия:

= 0001h – открыт существующий файл

= 0002h – открыт файл, который не существовал и был создан

= 0003h – открыт файл, созданный взамен удаленного файла

Примечание 1: предписываемое действие DL = 11h не поддерживается на дисках, доступ к которым осуществляется по сети.

Примечание 2: после выполнения операций с дисками, доступ к которым осуществляется по сети, статус в регистре CX не возвращается.

Примечание 3: если файл создается заново, то задаваемые в регистре BX состояния флагов заносятся в таблицу SFT (A.01-4).

Примечание 4: указатель места доступа устанавливается на начало файла.

Примечание 5: наличие атрибутов H (скрытый) и S (системный) не является препятствием для открытия существующего файла.

8.02-79 INT 21\AX=7302h – копирование блока параметров диска (DPB)

Обработчик прерывания INT 21\AX=7302h копирует в предоставленный буфер расширенный блок параметров (DPB, A.03-1) запрошенного диска, имеющего файловую систему FAT-12, FAT-16 или FAT-32.

При вызове:

AX = 7302h

DL – номер запрашиваемого диска (примечание 1 к 8.02-17)

CX – длина предоставляемого буфера, не менее 3Dh байт

ES:DI – указатель на предоставляемый буфер для данных

При неудачном завершении флаг CF установлен, в AX – код завершения (A.06-1).
Если флаг CF сброшен, то таблица DPB скопирована (примечание 2)

Примечание 1: в отличие от подобных функций INT 21\AH=1Fh и INT 21\AH=32h (8.02-24), функция INT 21\AH=7302h недоступна в предыдущих версиях DOS и нуждается в том, чтобы система BIOS компьютера поддерживала расширенный набор функций прерывания INT 13 (8.01-55).

Примечание 2: блок данных в буфере начинается со слова, означающего его длину, а данные из DPB следуют начиная со смещения DI+02.

Примечание 3: в скопированной таблице DPB место адреса заголовка драйвера (слово со смещением 13h) заполнено значением FFFFh.

Примечание 4: при вызове обработчик прерывания INT 21\AX=7302h старается обновить данные DPB путем считывания блока BPB с диска. Если попытка считывания кончается неудачно, то происходит вызов обработчика критических ошибок – прерывания INT 24.

8.02-80 INT 21\AX=7303h – таблица свободного дискового пространства

Обработчик прерывания INT 21\AX=7303h выдает сведения о наличии свободного пространства на дисках с файловыми системами FAT-12, FAT-16 и FAT-32. Сведения выдаются в виде блока данных A.13-7, копируемого в подготовленный буфер. В отличие от величин, возвращаемых функцией INT 21\AH=36h (8.02-30), величины в блоке данных A.13-7 не ограничены значением 2048 Мбайт.

При вызове:

AX = 7303h
CX – длина буфера для данных, не менее 34h байт
DS:DX – указатель на строку, определяющую диск (примечание 2)
ES:DI – указатель на буфер для данных; при вызове в буфере слово со смещением 02h должно быть 0000h.

При неудачном завершении флаг CF установлен, в AX – код завершения (A.06-1). Если флаг CF сброшен, то в буфере ES:DI – блок данных A.13-7.

Примечание 1: в отличие от подобной функции INT 21\AH=36h (8.02-30), функция INT 21\AH=7303h недоступна в предыдущих версиях DOS и нуждается в том, чтобы система BIOS компьютера поддерживала расширенный набор функций прерывания INT 13 (8.01-55).

Примечание 2: строка определяет запрашиваемый диск в той же форме, в которой он определен в таблице CDS (A.03-3), например, C:\ для локальных дисков и \\SERVER\Share для дисков, доступных по сети. Строка должна кончаться байтом 00h.

8.02-81 INT 21\AX=7305h – расширенные операции считывания и записи

Обработчик прерывания INT 21\AX=7305h представляет 5 операций считывания и записи с адресацией по номерам секторов, отсчитываемым отдельно от начала каждого логического диска. Эти операции подобны функциям INT 25 и

INT 26 (8.02-85), однако применимы не только к дискам с файловыми системами FAT-12 и FAT-16, но также к дискам с файловой системой FAT-32.

При вызове:

AX = 7305h
DL – номер логического диска (примечание 1 к 8.02-17)
DS:BX – указатель на пакет дисковой адресации (примечание 2).
SI – операция:
= 0000h – считывание
= 0001h – запись произвольных данных
= 2001h – запись данных таблицы FAT
= 4001h – запись данных каталога
= 6001h – запись данных в файл

При неудачном завершении флаг CF установлен, в AX – код завершения (A.06-1). Если флаг CF сброшен, то операция завершена успешно.

Примечание 1: в отличие от подобных функций INT 25 и INT 26 (8.02-85), функция INT 21\AH=7305h недоступна в предыдущих версиях DOS и нуждается в том, чтобы система BIOS компьютера поддерживала расширенный набор функций прерывания INT 13 (8.01-55).

Примечание 2: пакет дисковой адресации имеет длину 10 байт и содержит:
00h – двойное слово: номер сектора, с которого надо начать;
04h – слово: сколько секторов надо считать или записать;
06h – двойное слово: адрес буфера с данными или для данных.

8.02-82 INT 22 – адрес завершающей передачи управления

Соответствующий прерыванию INT 22 адрес перехода указывает не на обработчика, а на точку возврата в родительский процесс, то есть туда, куда должно быть передано управление после окончания работы текущей программы. Обычно адрес INT 22 указывает на команду, стоящую вслед за вызовом функции INT 21\AH=4Bh (8.02-53), которая запустила на исполнение текущую программу.

Прерывание INT 22 не вызывают непосредственно, потому что родительский процесс не может правильно возобновиться без восстановления состояния стека и многого другого, в том числе адресов для прерываний INT 22 – INT 24 в таблице прерываний. Все необходимые подготовительные операции входят в процедуру завершения работы текущей программы, выполняемую обработчиками прерываний INT 20 и INT 21\AH=4Ch (8.02-55). Они считывают адрес INT 22 из таблицы прерываний, копируют на его место адрес INT 22 для родительского процесса, хранимый в двойном слове со смещением 0Ah в префиксе сегмента текущей программы (PSP, A.07-1), а затем выполняют дальний переход по адресу, заранее считанному из таблицы прерываний.

8.02-83 INT 23 – останов исполнения текущей программы

Распознав нажатия клавишных комбинаций Ctrl-C или Ctrl-Break, обработчик прерывания INT 09 вызывает обработчика прерывания INT 1B (8.01-95), который взводит в состояние логической единицы флаг Ctrl-Break по адресу 0040:0071h в области данных BIOS (A.02-3). Состояние этого флага проверяют обработчики прерываний, установленные MS-DOS, когда их вызывает исполняемая программа. Обнаружив взведенное состояние флага, они вызывают обработчика прерывания INT 23. Именно он останавливает исполнение текущей программы.

Код обработчика прерывания INT 23 написан в расчете на то, что вызывающим процессом для него является функция DOS. Поэтому прямой вызов INT 23 из пользовательских программ не допускается, возможен только косвенный вызов через INT 1B (8.01-95).

Дальнейшее развитие событий частично зависит от обстоятельств вызова, но в большинстве ситуаций обработчик прерывания INT 23 ставит пользователя перед выбором: продолжить или завершить исполнение приостановленной программы. Возможные ответные действия пользователя описаны в разделе 1.03. Если будет избран вариант "завершить", то обработчик прерывания INT 23 закроет файлы, открытые приостановленной программой, высвободит занимаемую ей память, задаст нулевое значение возвращаемого уровня ошибки, установит флаг CF в состояние CY, и затем передаст управление родительскому процессу – тому, который вызвал на исполнение закрываемую программу. Если пользователем будет избран вариант "продолжить", то произойдет возврат к исполнению кода того обработчика прерывания, из которого произошел вызов INT 23, причем состояния регистров и флагов будут сохранены. Завершив свою миссию, этот обработчик возвратит управление вызывавшей его текущей программе.

8.02-84 INT 24 – обработчик критических ошибок

Когда система BIOS или порты неадекватно реагируют на обращения функций DOS к аппаратуре компьютера, тогда эти функции DOS вызывают INT 24 – обработчика критических ошибок. Обработчик критических ошибок анализирует переданные ему сведения об ошибке, но не очень надеется на себя; обычно он адресует пользователю свой знаменитый вопрос "Abort, Retry, Fail?", а потом передает решение пользователя для исполнения той функции DOS, которая вызвала INT 24.

Код обработчика прерывания INT 24 написан в расчете на то, что вызывающим процессом для него является функция DOS. Поэтому прямой вызов INT 24 из пользовательских программ не допускается.

При возврате состояния регистров (кроме AL) восстанавливаются по сохраненным в стеке значениям, а в регистре AL возвращается код действия:

AL = 00h – вернуться в вызывающую программу с ошибкой (fail);
= 01h – повторить исполнение запрошенной операции (retry);
= 02h – прекратить исполнение вызывающей программы (abort);
= 03h – констатировать системный отказ, остановить процессор.

Примечание 1: автоматическая выдача ответа FAIL и безостановочное продолжение исполнения обеспечиваются предварительной установкой флага в ячейке со смещением 2Ah в области текущих данных DOS (A.01-3). Этим пользуется, в частности, интерпретатор COMMAND.COM, когда принимает недокументированный параметр /F (6.04).

8.02-85 Считывание (INT 25) и запись (INT 26) с прямым доступом к диску

Обработчики прерываний INT 25 и INT 26 осуществляют доступ только в пределах логических дисков с файловыми системами FAT-12 и FAT-16, причем с адресацией по номерам секторов в обход файловой системы. На каждом логическом диске счет секторов начинается заново с номера 00000000h. Адрес буфера и конкретные номера секторов передаются в составе пакета дисковой адресации, показанного в примечании 2 к 8.02-81.

При вызове:

AL – номер логического диска (примечание 1 к 8.02-10)
CX = FFFFh (примечание 3)
DS:BX – пакет дисковой адресации (примечание 2 к 8.02-81).

При возврате состояния регистров BX, CX, DX, DI, SI не сохраняются.

При неудачном завершении флаг CF установлен, в AL – код завершения (A.06-1), в AH – статус ошибки, показанный в таблице A.06-5.

Если флаг CF сброшен, то миссия завершена успешно.

Примечание 1: после вызова INT 25 и INT 26 в вершине стека остается слово, отображающее исходное состояние флагов. Состояние стека должно быть восстановлено вызывающей программой. Нужно либо вытолкнуть это слово из стека, либо записать прежнее значение в регистр SP.

Примечание 2: обработчикам прерываний INT 25 и INT 26 доступны те логические диски, для которых драйвер способен обеспечить 32-разрядный обмен, что отмечается установлением бита 1 в слове атрибутов в заголовке драйвера (A.05-2).

Примечание 3: вызовы INT 25 и INT 26 со значением CX < FFFFh допустимы только по отношению к логическим дискам емкостью менее 32

Мбайт без кластерной структуры, которые маркируются идентификатором 04h из таблицы A.13-6. При обращении к таким дискам пакет дисковой адресации не употребляется, в регистры записывается следующее:

CX – число секторов
DX – начальный сектор
DS:BX – указатель на буфер с данными или для данных.

Примечание 4: для обращений к дискам с файловой системой FAT-32 обработчики прерываний INT 25 и INT 26 непригодны, вместо них следует использовать INT 21\AX=7305h (8.02-81).

8.02-86 INT 27 – завершение исполнения с оставлением TSR-модуля

При вызове:

CS – сегментный адрес PSP
DX – размер резидентного модуля в байтах, от 60h до FFF0h байт, отсчитываемый от начала PSP.

Примечание 1: вызов INT 27h высвобождает всю занятую программой память, за исключением резидентного модуля, восстанавливает указатели в таблице прерываний по данным из PSP, но не закрывает открытые файлы: это должна сделать сама программа до вызова INT 27h.

Примечание 2: обработчик прерывания INT 21\AH=31h (8.02-23) делает то же самое, но сверх того позволяет оставить ненулевые значения кода уровня ошибки (errorlevel) и не ограничивает размер резидентного модуля величиной 64 кбайт.

8.02-87 INT 28 – механизм фонового исполнения программ

DOS вызывает INT 28 в темпе тактов 18 раз в секунду в те интервалы времени, когда исполняемые операции ввода (INT 21\AH=01h-0Ch) ожидают ввода знаков с клавиатуры. В этих интервалах ожидания DOS фактически мается без дела. Устанавливаемый по умолчанию обработчик прерывания INT 28 представляет собой одну команду IRET (7.03-30), которая просто возвращает управление вызывающей программе. Перехват вызовов INT 28 позволяет активизировать другую программу, предназначенную для исполнения в фоновом режиме, пока основная программа ожидает ввода знаков пользователем. Перехватывающий обработчик вызовов INT 28 получает в регистрах SS:SP адрес, указывающий на вершину стека DOS, а при возврате должен восстановить состояния всех регистров.

Примечание 1: программы, активизируемые посредством INT 28, должны прежде всего проверять состояние флага InDOS (01h в таблице A.01-3), причем адрес этого флага должен быть найден заранее с помощью

INT 21\AH=34h (8.02-28) и сохранен с момента инициализации программы. Флаг InDOS в момент вызова INT 28 нормально имеет значение 01h; если значение больше 01h, то в фоновой программе нельзя вызывать на исполнение никакие функции DOS.

Примечание 2: программы, предназначенные для фонового исполнения путем перехвата INT 28 при значении 01h флага InDOS, могут пользоваться функциями DOS, однако не допускаются обращения к функциям INT 21\AH=01h-0Ch, а также обращения к номерным ссылкам устройства CON (обычно это ссылки 0000h – 0002h).

8.02-88 INT 29 – неперенаправляемое отображение знака на экране

К вызову INT 29 прибегают тогда, когда нужно вывести сообщение на экран дисплея, несмотря на то, что канал STDOUT может быть перенаправлен и что в качестве средства ввода-вывода может быть заявлено какое-либо иное устройство вместо принимаемого по умолчанию устройства CON (консоли).

При вызове:

AL – код ASCII знака, который надо отобразить на экране

При возврате содержимое регистра BX может быть утрачено.

Примечание 1: обработчик прерывания INT 29 осуществляет индикацию знака на экране посредством вызова INT 10\AH=0Eh (8.01-21).

Примечание 2: о поддержке INT 29 свидетельствует установление бита 4 в слове атрибутов драйвера устройства CON (A.05-2).

Примечание 3: неперенаправляемое отображение строки знаков можно осуществить также через номерную ссылку 0002h на канал STDERR посредством INT 21\AH=40h (8.02-36). В отличие от INT 29, такой вывод знаков всегда поддерживается драйвером устройства CON.

8.02-89 INT 2E – передача команды командному интерпретатору

Посредством вызова INT 2E программа передает командную строку на исполнение командному интерпретатору COMMAND.COM, причем новый резидентный модуль командного интерпретатора не загружается. Команда передается тому самому резидентному модулю, который запустил на исполнение вызывающую программу.

При вызове:

DS:SI – адрес командной строки в том формате, в каком она вписывается в PSP (A.07-1) начиная со смещения 80h. В первом байте строки должна быть указана ее длина. Конец строки отмечается байтом 0Dh, который не входит в счет длины строки.

При возврате:

AX – код завершения (A.06-1);
состояния всех других регистров, кроме CS:IP, не сохраняются.

Примечание 1: перед вызовом INT 2E вызывающая программа должна убедиться в том, что имеется достаточно свободной памяти для размещения подгружаемых модулей командного интерпретатора.

Примечание 2: вызванный прерыванием INT 2E командный интерпретатор работает со своими исходными переменными окружения, которые могут отличаться от переменных окружения вызывающей программы. Все изменения переменных окружения, которые могут произойти в ходе исполнения, не будут доступны вызывающей программе.

Примечание 3: не следует вызывать INT 2E из программ, которые запускаются из batch-файлов.

8.03 Обработчики от драйверов и резидентных программ

В ходе загрузки любого резидентного обработчика прерывания должна быть обеспечена возможность его вызова посредством адреса, занесенного в таблицу прерываний. Однако прежде чем любая программа сможет воспользоваться этим адресом, она должна убедиться, что в конкретной ячейке таблицы прерываний записан именно адрес вызова. Такой вопрос не стоит по отношению к модулям, поставляемым системами BIOS и DOS: их адреса – заведомо на своих местах. Но драйвер может быть не загружен, и тогда обращение к соответствующей ячейке, не содержащей адреса, почти наверняка приведет к "зависанию" компьютера.

Сначала, когда драйверов было не слишком много, каждому из них выделяли свою ячейку в таблице прерываний, и тогда определять факт загрузки драйвера можно было по наличию сигнатуры в определенной позиции относительно записанного в ячейку сегментного адреса. Именно так определяют, например, факт загрузки драйвера EMM386.EXE (примечание 1 к 8.03-62), "предки" которого известны с 1983 года.

Потом драйверов становилось все больше, и на ограниченном пространстве таблицы прерываний стали возникать конфликты. Помимо того, надо было устранить опасность "зависания" компьютера при обращении к отсутствующим модулям. Для решения обеих этих проблем была предложена идея обращения ко всем резидентным модулям драйверов посредством мультиплексного прерывания INT 2F. Предполагается, что каждый участвующий драйвер перехватывает вызовы INT 2F, образуя тем самым звено в цепи ссылок (подробнее – в A.07-5). Вызов передается по цепи ссылок от одного обработчика к другому, и каждый из них анализирует идентификатор, записанный в регистр AH. Если обработчик не сочтет

идентификатор "своим", то вызов в неизменном виде продолжит путешествие по цепи ссылок. Последней будет ссылка на команду IRET, которую первоначально загрузила DOS. Команда IRET возвратит управление вызывающей программе, не изменяя состояния регистров. Любое изменение их первоначального состояния будет означать, что один из обработчиков посчитал идентификатор "своим". Следовательно, запрошенный резидентный модуль уже загружен.

Разумеется, мультиплексное прерывание можно использовать не только для опознания. Оно дает возможность обращаться к основным функциям резидентных модулей без риска зависания компьютера, если требуемый модуль не загружен. Однако прослеживание цепи ссылок при каждом вызове существенно замедляет работу программ. Во избежание задержек многие драйверы передают через INT 2F прямой адрес вызова своих специфических функций, чтобы далее программы могли пользоваться этими функциями, не обращаясь к мультиплексному прерыванию. Именно так строит свое взаимодействие с программами, например, драйвер Himem.sys (8.03-22, 8.03-23).

Практика пользования прерыванием INT 2F получила распространение, но не могла предотвратить конфликты из-за несогласованного назначения одинаковых фиксированных идентификационных кодов многим разным драйверам и резидентным модулям. Во избежание таких конфликтов было предложено разрешить любому резидентному модулю присваивать себе первый попавшийся идентификационный код, который будет найден незанятым в момент загрузки. Эта идея реализована мультиплексным прерыванием INT 2D (подробнее – в A.07-6), которое избавляет от конфликтов программы, разрабатываемые в настоящее время.

Вместе с тем старые драйверы с давно известными идентификационными кодами продолжают использовать прерывание INT 2F. Более того, отдельные резидентные модули, которые в прежних версиях DOS загружались с помощью драйверов, теперь вошли в состав ядра DOS, но ради сохранения совместимости по-прежнему вызываются посредством INT 2F. По этой причине проверка факта загрузки модулей с идентификационными кодами AH = 05h, 08h и 12h не имеет смысла – они фактически представляют функции DOS. Чтобы не потерять удобство порядкового поиска материалов, вызовы этих и нескольких других функций описаны ниже в разделе 8.03 наряду с вызовами функций загружаемых драйверов.

8.03-01 INT 2F\AX=0501h – преобразование кода завершения в сообщение

При вызове:

AX = 0501h

BX – преобразуемый код завершения (A.06-1)

При возврате: если флаг CF сброшен, то

AL = 00h – сообщение нужно дополнить буквой диска

= 01h – сообщение готово к посылке на экран дисплея

ES:DI – указатель на строку сообщения с байтом 00h в конце.
При неудаче флаг CF установлен: значит, для данного кода нет сообщения.
При любом исходе состояния флагов и регистров AX, DI, ES не сохраняются.

8.03-02 INT 2F\AX=0801h – добавление логического диска

Обработчик прерывания INT 2F\AX=0801h добавляет подготовленную таблицу к последовательности таблиц DDT (Disk Data Tables, A.03-2), и исправляет соответствующим образом ссылки в других таблицах DDT, относящихся к тому же физическому дисководу. С этого момента добавленный логический диск становится открыт для доступа с помощью дисковых драйверов, входящих в состав ядра DOS.

При вызове:
AX = 0801h
DS:DI – указатель на подготовленную таблицу DDT (A.03-2)
При возврате состояния регистров AX, BX, SI, ES не сохраняются.

Примечание 1: не следует применять INT 2F\AX=0801h по отношению к дискам с устанавливаемой файловой системой (IFS, 5.08-01), к сетевым и другим дискам, если драйверы из состава ядра DOS не способны эти диски обслужить.
Примечание 2: образец таблицы DDT можно получить с помощью INT 2F\AX=0803h.

8.03-03 INT 2F\AX=0802h – отсылка запроса дисковому драйверу

Обработчик прерывания INT 2F\AX=0802h посылает запросы к драйверам логических дисков, входящих в состав ядра DOS. Запросы должны относиться к дискам, обслуживаемым этими драйверами и представленным в таблицах DDT (Disk Data Tables, A.03-2). Номер запрашиваемого диска вместе с кодом операции (A.05-3) указываются в блоке данных запроса (A.05-4).

При вызове:
AX = 0802h
ES:BX – указатель на блок данных запроса (A.05-4)
При возврате результаты отображаются в блоке данных запроса в соответствии с запрошенной операцией (A.05-3 – A.05-7).

Примечание 1: после вызова INT 2F\AX=0802h в вершине стека остается слово, отображающее исходное состояние флагов. Состояние стека должно быть восстановлено вызывающей программой. Нужно либо вытолкнуть это слово из стека, либо записать прежнее значение в регистр SP.

Примечание 2: неудачное завершение отображается кодами в байтах со смещением 03h и 04h в блоке данных запроса (A.05-4), но в любом случае вызов обработчика критических ошибок (INT 24) не производится.

8.03-04 INT 2F\AX=0803h – определение адреса первой таблицы DDT

При вызове:

AX = 0803h

При возврате:

DS:DI – указатель на начало первой таблицы DDT (A.03-2)

Примечание 1: последовательность таблиц DDT несложно проследить, поскольку каждая таблица начинается с 4-байтового указателя на следующую таблицу DDT (A.03-2).

8.03-05 INT 2F\AX=1202h – указатель на адрес обработчика прерывания

При вызове:

AX = 1202h

Номер запрашиваемого прерывания – в верхнем регистре стека

При возврате:

ES:BX – указатель на адрес обработчика прерывания;
значение в регистре AX утрачивается; состояние стека не изменяется.

Примечание 1: при работе в реальном режиме эта функция просто умножает номер прерывания на 4.

8.03-06 INT 2F\AX=1212h – определение длины строки

При вызове:

AX = 1212h

ES:DI – указатель на строку, длину которой надо определить. Строка должна заканчиваться байтом 00h.

При возврате:

CX – длина строки, включая последний байт 00h.

Примечание 1: обработчик прерывания INT 2F\AX=1225h делает то же самое, но только принимает указатель на строку из регистров DS:SI.

8.03-07 INT 2F\AX=1213h – перевод строчной буквы в заглавную

При вызове:

AX = 1213h

Код ASCII строчной буквы – в верхнем регистре стека.

При возврате состояние стека не изменяется, и тогда
AL – код ASCII соответствующей заглавной буквы.

8.03-08 INT 2F\AX=1214h – сравнение четырехбайтовых указателей

При вызове:
AX = 1212h
DS:SI – первый указатель
ES:DI – второй указатель
При возврате:
флаг ZF установлен и флаг CF сброшен, если указатели одинаковы;
флаг ZF сброшен и флаг CF установлен, если указатели различны.

8.03-09 INT 2F\AX=1216h – определение адреса блока описания в таблице SFT

Обработчик прерывания INT 2F\AX=1216h принимает номер запрашиваемого блока описания в таблицах SFT (A.01-4) и возвращает указатель на этот блок, предоставляя тем самым возможность доступа к нему.

При вызове:
AX = 1216h
BX – номер запрашиваемого блока описания в таблице SFT
При успешном завершении флаг CF сброшен, значение в AX утрачено, и тогда
ES:DI – указатель на блок описания в SFT,
BX – относительный номер блока описания в той таблице SFT, в которой этот блок описания содержится.
Если при возврате флаг CF установлен, то миссия потерпела неудачу.

Примечание 1: вероятная причина неудачного завершения – запрос номера, превышающего предел, установленный командой FILES (4.12).
Примечание 2: указатель на номер блока описания в таблице SFT можно найти по известной номерной ссылке с помощью INT 2F\AX=1220h (8.03-11).

8.03-10 INT 2F\AX=121Eh – сопоставление имен файлов

При вызове:
AX = 121Eh
DS:SI – указатель на строку с именем первого файла;
ES:DI – указатель на строку с именем второго файла;
обе строки должны кончаться байтом 00h.
При возврате флаг ZF установлен, если имена эквивалентны.
Флаг ZF сброшен, если имена разные.

8.03-11 INT 2F\AX=1220h – указатель на байт в таблице JFT

Таблица JFT (Job File Table, примечание 3 к A.07-1) содержит номера блоков описания, находящихся в другой таблице – SFT (System File Table, A.01-4) и описывающих состояние открытых для доступа объектов – каналов или файлов. Доступ к этим объектам осуществляется по номерным ссылкам, известным вызывающей программе. Путь от номерной ссылки до соответствующего объекту блока описания начинается с вызова INT 2F\AX=1220h, возвращающего указатель на номер блока описания в таблице SFT. Затем по этому номеру с помощью INT 2F\AX=1216h (8.03-09) можно будет найти адрес соответствующего объекту блока описания в таблице SFT.

При вызове:

AX = 1220h

BX – номерная ссылка

Если при возврате флаг CF сброшен, то

ES:DI – адрес того байта таблицы JFT, в котором записан искомый номер блока описания из таблицы SFT.

При неудаче флаг CF установлен, AL = 06h (- неверная номерная ссылка).

Примечание 1: значение FFh в том байте таблицы JFT, на который указывает возвращенный адрес в регистрах ES:DI, означает неактивное (закрытое) состояние запрошенной номерной ссылки.

8.03-12 INT 2F\AX=122Ch – путь к цепи драйверных заголовков

Вызов INT 2F\AX=122Ch позволяет проследить всю цепочку заголовков загруженных драйверов, потому что каждый заголовок начинается с 4-байтового указателя на следующий заголовок. Заголовок последнего драйвера начинается со слова FFFFh.

При вызове:

AX = 122Ch

При возврате:

BX:AX – указатель на начало заголовка второго драйвера (первый драйвер – это драйвер виртуального устройства NUL, находящийся в "списке списков", A.01-2).

8.03-13 INT 2F\AX=1500h – число имеющихся оптических дисководов

При вызове:

AX = 1500h

BX = 0000h

При успешном завершении в регистре AL сигнатура FFh (примечание 1), и тогда:

BX – число имеющихся оптических дисководов
CX – номер диска, присвоенный первому оптическому дисководу (0002h = C:, 0003h = D:, и т.д.).

Примечание 1: возврат исходного состояния AL = 00h означает, что не загружен резидентный модуль программы доступа к оптическим дискам (5.08-03 или 5.08-04), который должен отвечать на вызов функции INT 2F\AX=1500h.

Примечание 2: обработчик прерывания INT 2F\AX=1500h "конфликтует" с драйвером GRAPHICS.COM и, помимо того, может ошибаться в букве первого оптического дисковода, когда установлен драйвер INTERLNK.EXE.

8.03-14 INT 2F\AX=1501h – адреса драйверов оптических дисководов

При вызове:
AX = 1501h
ES:BX – указатель на буфер, по 5 байт на каждый дисковод
При возврате:
ES:BX – указатель на буфер, заполненный данными: для каждого дисковода 1-й байт – номер обслуживаемого драйвером устройства, а следующие за ним 4 байта – адрес соответствующего драйверного заголовка. Длина блока данных определяется числом дисководов, которое можно узнать посредством функции INT 2F\AX=1500h (8.03-13).

Примечание 1: прежде чем вызывать обработчик прерывания INT 2F\AX=1501h, входящий в состав резидентного модуля программы доступа к оптическим дискам (5.08-03 или 5.08-04), нужно сначала вызовом INT 2F\AX=150Bh (8.03-17) проверить, загружен ли и активен ли этот резидентный модуль.

Примечание 2: при работе в "окне DOS" операционной системы WINDOWS вызов INT 2F\AX=1501h возвращает AX=0000h и неверные адреса.

8.03-15 INT 2F\AX=1505h – чтение таблицы содержания оптического диска

При вызове:
AX = 1505h
CX – номер оптического диска (0002h = C:, 0003h = D:, и т.д.)
DX – индекс сектора (примечание 2)
ES:BX – указатель на буфер размером 2048 байт
При неудачном завершении флаг CF установлен, AH = 00h
AL – вид ошибки:

= 15h – неверно указан номер диска
= 21h – дисковод занят или диск в нем отсутствует

Если флаг CF сброшен, то в буфере – таблица содержания, и тогда

AL – тип дескриптора:
= 01h – стандартный дескриптор тома;
= FFh – последний дескриптор тома;
= 00h – все другие виды дескрипторов тома.

Примечание 1: прежде чем вызывать обработчик прерывания INT 2F\AX=1505h, входящий в состав резидентного модуля программы доступа к оптическим дискам (5.08-03 или 5.08-04), нужно сначала вызовом INT 2F\AX=150Bh (8.03-17) проверить, загружен ли и активен ли этот резидентный модуль.

Примечание 2: оптический диск может содержать несколько дескрипторов тома, индекс сектора 0000h соответствует первому дескриптору, индекс сектора 0001h – второму дескриптору, и т.д.

8.03-16 INT 2F\AX=1508h-1509h – доступ к секторам оптического диска

При вызове:

AX = 1508h – операция считывания
= 1509h – операция записи
CX – номер оптического диска (0002h = C:, 0003h = D:, и т.д.)
DX – число секторов, которые надо считать или записать
ES:BX – указатель на буфер с данными или для данных
SI:DI – номер сектора, с которого надо начать

При неудачном завершении флаг CF установлен, и тогда причина неудачи

AL = 0Fh – неверно указан номер диска
= 15h – дисковод занят или диск в нем отсутствует.

Если флаг CF сброшен, то операция прошла успешно, после операции считывания буфер заполнен считанными данными.

Примечание 1: прежде чем вызывать обработчики прерываний INT 2F\AX=1508h и INT 2F\AX=1509h, входящие в состав резидентного модуля программы доступа к оптическим дискам (5.08-03 или 5.08-04), нужно сначала вызовом INT 2F\AX=150Bh (8.03-17) проверить, загружен ли и активен ли этот резидентный модуль.

Примечание 2: вызовы INT 2F\AX=1508h-1509h при работе в "окне DOS" системы Windows всегда возвращают AL = 15h (диск занят).

Примечание 3: ранние версии программ доступа к оптическим дискам (5.08-03 и 5.08-04) не поддерживали операцию записи. Кроме того, многие оптические дисководы аппаратно неспособны осуществлять запись.

8.03-17 INT 2F\AX=150Bh – запрос об обслуживании диска.

При вызове:

AX = 150Bh

CX – номер диска (0002h = C:, 0003h = D:, и т.д.).

При возврате:

BX = ADADh, – подтверждение активности резидентного модуля программы доступа к оптическим дискам (5.08-03 или 5.08-04).

AX = 0000h – значит, запрошен диск, не обслуживаемый модулем программы доступа к оптическим дискам.

8.03-18 INT 2F\AX=150Dh – номера оптических дисков

При вызове:

AX = 150Dh

ES:BX – указатель на буфер, по одному байту на диск

При возврате:

ES:BX – указатель на буфер, заполненный номерами дисков (02h = C:, 03h = D:, и т.д.). Конец списка номеров отмечен байтом 00h.

Примечание 1: прежде чем вызывать обработчик прерывания INT 2F\AX=150Dh, входящий в состав резидентного модуля программы доступа к оптическим дискам (5.08-03 или 5.08-04), нужно сначала вызовом INT 2F\AX=150Bh (8.03-17) проверить, загружен ли и активен ли этот резидентный модуль.

8.03-19 INT 2F\AX=150Fh – считывание каталога оптического диска

При вызове:

AX = 150Fh

CH = 00h – считывать каталог "как он есть", без изменений

= 01h – считывать, преобразуя в каноническую форму (A.09-6)

CL – номер оптического диска (02h = C:, 03h = D:, и т.д.)

ES:BX – указатель на строку с путем, кончающуюся байтом 00h

SI:DI – адрес буфера, не менее 255 байт для прямого копирования.

При неудачном завершении флаг CF установлен, AX = код завершения (A.06-1).

При успешном завершении флаг CF сброшен, буфер SI:DI заполнен, и тогда

AX – формат диска:

= 0000h – диск формата High Sierra,

= 0001h – диск формата ISO 9660.

Примечание 1: прежде чем вызывать обработчик прерывания INT 2F\AX=150Fh, входящий в состав резидентного модуля программы доступа к оптическим дискам (5.08-03 или 5.08-04), нужно сначала вызовом

INT 2F\AX=150Bh (8.03-17) проверить, загружен ли и активен ли этот резидентный модуль.

8.03-20 INT 2F\AX=160Ah – тест операционной среды OS Windows.

При вызове:

AX = 160Ah

При возврате:

AX = 0000h, если OS Windows отвечает на вызов INT 2F\AX=160Ah.
BH:BL – версия OS Windows
CX – вид установки (0002h = стандартный, 0003h = улучшенный)

Примечание 1: возврат ненулевого значения в регистре AX не означает, что действие происходит вне "окна DOS" операционной системы Windows. Дело в том, что среди настроек OS Windows имеется флаг "Prevent DOS programs from detecting Windows" (= не позволять программам DOS обнаруживать Windows). Когда этот флаг установлен, OS Windows не отвечает на вызов INT 2F\AX=160Ah.

8.03-21 INT 2F\AX=1687h – пробный запрос к серверу DPMI.

Серверы DPMI обслуживают исполнение программ в режиме V86 и по запросу активизируют функции прерывания INT 31. Эти функции дают программам возможность устанавливать свои обработчики прерываний защищенного режима, а также запрашивать выделение дополнительных ресурсов у того драйвера или у той операционной системы, которыми организована работа в режиме V86. Серверы DPMI для работы в среде DOS (QDPMI.SYS, CWSDPMI.EXE и др.) сейчас применяют редко, так как самый доступный сервер DPMI предоставлен операционной системой Windows в эмулируемой среде ее "окна DOS". Там пробный запрос к серверу DPMI всегда срабатывает, даже когда система Windows "не хочет" обнаруживать себя легально (примечание 1 к 8.03-20). Помимо прочего, положительная реакция на пробный запрос является достаточным свидетельством того, что процессор компьютера работает в режиме V86. По перечисленным причинам к функции INT 2F\AX=1687h приходится обращаться, несмотря на то, что сервер DPMI в состав MS-DOS7 не входит, и пользование другими функциями DPMI в этой книге не рассматривается.

При вызове:

AX = 1687h

При возврате:

AX = 0000h – подтверждение того, что сервер DPMI загружен
BX – бит 0 = 1b означает поддержку 32-битовых программ
CL – тип процессора (02h – 80286, 03h – 80386, 04h – 80486,...)

DH:DL – версия сервера DPMI
SI – число параграфов в блоке данных сервера DPMI
ES:DI – адрес входа для активизации функций прерывания INT 31

Примечание 1: в отличие от среды, формируемой драйверами DPMI для DOS, в среде "окна DOS" операционной системы Windows нет доступа к таймеру BIOS (8.01-73) и к функциям VCPI (8.03-71 – 8.03-73).

Примечание 2: операционная система "Open DOS" фирмы Caldera отличается от всех других версий DOS тем, что в ней сервер DPMI включен в состав драйвера EMM386.EXE.

8.03-22 INT 2F\AX=4300h – тест доступности функций XMS-драйвера.

При вызове:
AX = 4300h
При возврате:
если AL = 80h, то XMS-драйвер (HIMEM.SYS, 5.04-01) установлен;
при любом другом значении AL функции XMS-драйвера недоступны.

8.03-23 INT 2F\AX=4310h – адрес вызова функций XMS-драйвера

Обработчик прерывания INT 2F\AX=4310h возвращает адрес для вызова командой CALL FAR (7.03-08) функций XMS- драйвера, перечисленных в таблице A.12-3.

При вызове:
AX = 4310h
При возврате:
ES:BX – адрес вызова XMS-драйвера (HIMEM.SYS, 5.04-01).

Примечание 1: обработчик прерывания INT 2F\AX=4310h устанавливается драйвером HIMEM.SYS, поэтому нужно сначала проверить, загружен ли этот драйвер, посредством вызова INT 2F\AX=4300h (8.03-22).

Примечание 2: для вызова INT 2F\AX=4310h и других функций драйвера HIMEM.SYS требуется наличие до 256 байт свободного пространства стека.

8.03-24 INT 2F\AX=4B52h – операции драйвера KEYRUS.COM (5.02-05)

При вызове:
AX = 4B52h (= 'KR')
BL – операция:
= 00h – проверка, загружен ли драйвер KEYRUS.COM

= 4Ch – ввод знаков 0 – 127 (американского алфавита)
= 90h – ввод знаков 128 – 255 (национального алфавита)

При возврате значение в ES утрачивается.
Если драйвер KEYRUS.COM установлен, то операция 00h возвращает:

AL = 82h
BH:BL – номер версии драйвера KEYRUS.COM

8.03-25 INT 2F\AX=AD00h – тест загрузки драйвера DISPLAY.SYS.

При вызове:

AX = AD00h

При возврате:

если AL = FFh, то драйвер DISPLAY.SYS (5.02-02) установлен.
Состояние регистра BX может быть изменено.

8.03-26 INT 2F\AX=AD01h-AD02h – активная кодовая страница

Вызовы прерываний INT 2F\AX=AD01h-AD02h обслуживаются резидентным модулем драйвера DISPLAY.SYS (5.02-02). Перед обращением к нему необходимо с помощью INT 2F\AX=AD00h (8.03-25) убедиться, что он установлен.

При вызове:

AX = AD01h – сменить кодовую страницу знакогенератора
= AD02h – сообщить номер установленной кодовой страницы
BX – новая кодовая страница (A.02-2), только при AX=AD01h.

При успешном завершении флаг CF сброшен, подфункция AX=AD02h возвращает:

BX – номер установленной кодовой страницы.

При неудаче флаг CF установлен, содержимое регистров AX и BX не сохраняется.

8.03-27 INT 2F\AX=AD03h – сведения о кодовых страницах знакогенератора

Вызов прерывания INT 2F\AX=AD03h обслуживается резидентным модулем драйвера DISPLAY.SYS (5.02-02). Перед обращением к нему необходимо с помощью INT 2F\AX=AD00h (8.03-25) убедиться, что он установлен.

При вызове:

AX = AD03h
ES:DI – указатель на буфер для блока данных
CX – размер буфера в байтах (A.02-6)

При неудаче флаг CF установлен. Вероятная причина: буфер слишком мал.
При удачном завершении флаг CF сброшен, и тогда

ES:DI – указатель на блок данных, показанный в разделе A.02-6.

8.03-28 INT 2F\AX=AD80h – тест доступности функций драйвера KEYB.COM.

При вызове:

AX = AD80h

Если драйвер KEYB.COM (5.02-04) установлен, то AL = FFh, и тогда

BH:BL – номер версии драйвера

ES:DI – указатель на блок служебных данных драйвера.

Содержимое регистра AH может быть изменено.

8.03-29 INT 2F\AX=AD81h – задание кодовой страницы для клавиатуры

Вызов прерывания INT 2F\AX=AD81h обслуживается резидентным модулем драйвера KEYB.COM (5.02-04). Перед обращением к нему необходимо с помощью INT 2F\AX=AD80h (8.03-28) убедиться в том, что он установлен.

При вызове:

AX = AD81h

BX – номер кодовой страницы

При неудаче флаг CF установлен, AX = 0001h если кодовая страница недоступна.
Если флаг CF сброшен, то миссия завершена успешно.

8.03-30 INT 2F\AX=AD82h-AD83h – раскладки клавиатуры

Вызовы функций INT 2F\AX=AD82h-AD83h обслуживаются резидентным модулем драйвера KEYB.COM (5.02-04). Перед обращением к нему необходимо с помощью INT 2F\AX=AD80h (8.03-28) убедиться в том, что он установлен.

При вызове:

AX = AD82h – задать раскладку клавиатуры

AX = AD83h – выяснить действующую раскладку клавиатуры

BL – операция (только при AX = AD82h):

= 00h – задать американскую раскладку

= FFh – задать национальную раскладку

При неудаче флаг CF установлен, вероятная причина – неверное значение в BL.
При удачном завершении флаг CF сброшен, и тогда только после AX=AD83h:

BL = 00h – активна американская раскладка (знаки 32 - 127),

= FFh – активна национальная раскладка (знаки 128 - 255).

8.03-31 INT 33\AX=0000h – установка начального состояния драйвера "мыши"

Приведение в начальное состояние означает установление таких значений параметров, которые принимаются по умолчанию. Счетчики смещений "мыши" и вращения ее колеса прокрутки сбрасываются в нуль, координаты курсора задаются

на центр экрана, для отображения назначается экранная страница 0, и курсор делается невидимым (сделать его видимым можно с помощью INT 33\AX=0001h). Помимо прочего, вызов INT 33\AX=0000h позволяет определить, установлен ли драйвер "мыши", и если установлен, то какую именно "мышь" он обслуживает.

При вызове:

AX = 0000h

Если при возврате AX = 0000h, то драйвер "мыши" не установлен (примечание 2).
Если при возврате AX = FFFFh, то драйвер "мыши" установлен, и тогда

CX = 0000h – применена не-двухкнопочная "мышь";
= 0002h (и FFFFh тоже) – используется 2-кнопочная "мышь";
= 0003h – 3-кнопочная "мышь" Mouse Systems или Logitech.

Примечание 1: если видеорежим был изменен, то перед установлением начального состояния нужно сбросить флаг смены видеорежима (примечание 2 к INT 33\AX=0028h, 8.03-52).

Примечание 2: MS-DOS7 заполняет свободные ячейки таблицы прерываний (до INT 3F) отсылками к команде IRET (7.03-30). Когда драйвер "мыши" не загружен, эта команда IRET просто возвращает неизмененное состояние регистра AX. Определить, установлен ли драйвер "мыши", не приводя его в начальное состояние, можно с помощью функции INT 33\AX=0021h (8.03-49).

8.03-32 INT 33\AX=0001h-0002h – переключение отображения курсора "мыши"

При вызове:

AX = 0001h – разрешить отображение курсора "мыши"
= 0002h – запретить отображение курсора "мыши"

Примечание 1: запрещать отображение курсора "мыши" нужно при завершении работы любой программы, вызывавшей отображение курсора "мыши". Помимо того, полезно запрещать отображение курсора перед каждым изменением содержания изображения, но только это лучше делать локально посредством INT 33\AX=0010h (8.03-42).

Примечание 2: если вызывать INT 33\AX=0002h несколько раз, то потребуется столько же раз вызывать INT 33\AX=0001h, чтобы снова сделать курсор видимым. Число несброшенных запретов отображения можно выяснить с помощью INT 33\AX=002Ah (8.03-53).

8.03-33 INT 33\AX=0003h – положения кнопок, курсора и колеса прокрутки

При вызове:

AX = 0003h

При возврате:

BH – 8-битовый поворот колеса прокрутки (примечание 1 к 8.03-33)
BL – байт состояния кнопок (примечание 2 к 8.03-33)
CX – горизонтальная X-координата (примечание 1 к 8.03-53)
DX – вертикальная Y-координата (примечание 1 к 8.03-53)

Примечание 1: поворот – это число со знаком, возвращаемое если колесо прокрутки у "мыши" имеется и если драйвер способен обслуживать такую "мышь". Оба эти условия надо проверять вызовом функции INT 33\AX=0011h (8.03-43). Положительные значения поворота соответствуют направлению "к себе". Счетчик поворота сбрасывается в нуль после каждого вызова функций INT 33\AX=0003h, INT 33\AX=0005h и INT 33\AX=0006h (8.03-35).

Примечание 2: удержание нажатыми левой и правой кнопок "мыши" отмечается в байте состояния установлением в единицу соответственно битов 0 и 1. Средней кнопке соответствует бит 2, если установленный драйвер поддерживает работу с трехкнопочной "мышью".

Примечание 3: в текстовых видеорежимах значения координат выражаются числами, кратными размеру знакоместа.

8.03-34 INT 33\AX=0004h – изменение расположения курсора "мыши"

При вызове:
AX = 0004h
CX – X-координата курсора (0000h – 0280h в видеорежиме 3)
DX – Y-координата курсора (0000h – 00C0h в видеорежиме 3)

Примечание 1: в текстовых видеорежимах координаты округляются до ближайшего меньшего значения, кратного размеру знакоместа.

Примечание 2: максимальные значения координат курсора для разных видеорежимов можно узнать с помощью INT 33\AX=0026h (8.03-51) или с помощью INT 33\AX=0031h (8.03-54).

8.03-35 INT 33\AX=0005h–0006h – сообщения о кнопках и о колесе прокрутки

При вызове:
AX = 0005h – запрос о нажатиях кнопок или о повороте колеса
= 0006h – запрос об отпускании кнопок или о повороте колеса
BX = 0000h – запрашивается сообщение о левой кнопке;
= 0001h – запрашивается сообщение о правой кнопке;
= 0002h – запрашивается сообщение о средней кнопке;
= FFFFh – запрос о повороте колеса прокрутки.

При возврате:
AH – 8-битовое значение поворота колеса (примечание 1 к 8.03-33)

AL – байт состояния кнопок (примечание 2 к 8.03-33)
BX – после запроса сообщения о кнопках: число нажатий или отпусканий запрошенной кнопки с момента посылки точно такого же предыдущего запроса;
– после запроса о колесе прокрутки: 16-битовое значение поворота колеса прокрутки (примечание 1 к 8.03-33);
CX – горизонтальная X-координата курсора в момент последнего запрошенного события (нажатия или отпускания кнопки или поворота колеса прокрутки);
DX – вертикальная Y-координата курсора в момент последнего запрошенного события (нажатия или отпускания кнопки или поворота колеса прокрутки).

Примечание 1: возврат значений BX = CX = DX = 0000h означает, что с момента посылки точно такого же предыдущего запроса запрашиваемое событие не произошло ни одного раза.

8.03-36 INT 33\AX=0007h-0008h – границы перемещения курсора "мыши"

При вызове:
AX = 0008h – задание границ по вертикали
= 0007h – задание границ по горизонтали,
CX – наименьшее значение координаты (примечание 2 к 8.03-34)
DX – наибольшее значение координаты (примечание 2 к 8.03-34)

Примечание 1: если в момент вызова курсор "мыши" находился за задаваемым пределом, то он перемещается к соответствующей границе.

8.03-37 INT 33\AX=0009h – вид курсора "мыши" в графическом видеорежиме

При вызове:
AX = 0009h
BX – горизонтальный сдвиг (от–16 до +16) вершины курсора
CX – вертикальный сдвиг (от–16 до +16) вершины курсора
ES:DX – указатель на пакет масок (примечание 1).

Примечание 1: пакет масок включает маску экрана и маску курсора. Маска экрана занимает 16 слов, начиная со смещения 00h, а маска курсора – тоже 16 слов, начиная со смещения 20h. Каждое слово в маске определяет 16 пикселов, расположенных горизонтально в ряд, причем наименее значимому биту соответствует самый правый пиксел. Маска экрана накладывается на содержимое видеопамяти посредством поэлементной логической операции AND ("И"), а затем на

полученный результат накладывается маска курсора посредством поэлементной логической операции XOR ("исключающее ИЛИ").

Примечание 2: действующий в данный момент сдвиг вершины курсора можно узнать посредством функции INT 33\AX=002Ah (8.03-53).

8.03-38 INT 33\AX=000Ah – вид курсора "мыши" в текстовом видеорежиме

При вызове:

AX = 000Ah
BX = 0000h – программный вариант задания вида курсора
= 0001h – аппаратный вариант задания вида курсора
CX – маска экрана (BX=0000h) или начальная строка (BX=0001h)
DX – маска курсора (BX=0000h) или конечная строка (BX=0001h).

Примечание 1: если выбран программный вариант, то маска экрана накладывается на содержимое видеопамяти посредством поэлементной логической операции AND ("И"), а затем на полученный результат накладывается маска курсора посредством поэлементной логической операции XOR ("исключающее ИЛИ"). Старшие байты масок из регистров CH и DH накладываются на байт цвета в видеопамяти (A.10-5). В частности, при CX=0000h форма курсора соответствует знаку, код ASCII которого задан в регистре DL, цвет знака определяется битами 0 – 3 из DH согласно таблице A.10-5, цвет фона – битами 4 – 6 из DH, а от бита 7 в DH зависит мерцание. Если заданы ненулевые значения в битах 4 – 6 регистра CH, то цвет становится зависимым от содержимого видеопамяти, так что курсор получается хорошо заметным на любом фоне.

Примечание 2: если выбран аппаратно определенный вариант, то курсор имеет вид мерцающей полоски или мерцающего прямоугольника. В частности, при CX=0002h и DX=0003h полоска высвечивается над знаком в строке, а при CX=0003h, DX=0004h – под знаком как подчеркивание.

8.03-39 INT 33\AX=000Bh – показания счетчиков движения "мыши"

При вызове:

AX = 000Bh

При возврате:

CX – смещение по горизонтали с момента предыдущего вызова
DX – смещение по вертикали с момента предыдущего вызова

Примечание 1: смещения курсора "мыши" измеряются в шагах ("mickeys"). Шаг смещения – это наименьшее воспринимаемое изменение положения

"мыши". Драйверы фирмы Microsoft связывают положительное число шагов с направлением движения вниз и вправо. Соотношение между величиной шага и размером пиксела можно изменять с помощью INT 33\AX=000Fh (8.03-41).

Примечание 2: INT 33\AX=0027h при вызове тоже требует указать только AX, и возвращает те же смещения в CX и DX, но сверх того возвращает в зависимости от варианта отображения курсора (8.03-38) в регистре AX – маску экрана или начальную строку курсора, а в регистре BX – маску курсора или конечную строку курсора.

8.03-40 INT 33\AX=000Ch – вызов резидентной программы драйвером "мыши"

При вызове:

AX = 000Ch

CX – условия вызова:

- бит 0: – при движении "мыши"
- бит 1: – при нажатии левой кнопки
- бит 2: – при отпускании левой кнопки
- бит 3: – при нажатии правой кнопки
- бит 4: – при отпускании правой кнопки
- бит 5: – при нажатии средней кнопки
- бит 6: – при отпускании средней кнопки
- бит 7: – при повороте колеса прокрутки

ES:DX – адрес вызова программы командой CALL FAR (7.03-08)

Примечание 1: в регистре CX могут быть заявлены несколько условий, и тогда вызов резидентной программы будет производиться при выполнении любого из этих условий. Состояния битов 8 – 15 в регистре CX не учитываются, а состояния битов 5 – 7 учитываются только теми драйверами, которые поддерживают соответствующие функции "мыши".

Примечание 2: при вызове программы драйвер оставляет в регистрах следующее:

- AX – условия вызова (как в CX при вызове драйвера);
- BH – поворот колеса прокрутки (примечание 1 к 8.03-33)
- BL – байт состояния кнопок (примечание 2 к 8.03-33)
- CX – горизонтальная координата (примечание 2 к 8.03-34)
- DX – вертикальная координата (примечание 2 к 8.03-34)
- SI – смещение по горизонтали (примечание 1 к 8.03-39)
- DI – смещение по вертикали (примечание 1 к 8.03-39)

Примечание 3: обработчик прерывания INT 33\AX=0014h заменяет ранее заданные условия вызова и адрес на новые, которые надо точно так же указать в регистрах CX и ES:DX. В отличие от INT 33\AX=000Ch,

обработчик прерывания INT 33\AX=0014h возвращает в CX и в ES:DX замененные прежние значения условий вызова и адреса.

Примечание 4: если программа установила адрес вызова, то при ее завершении установленный адрес должен быть дезактивирован посредством повторного вызова INT 33\AX=000Ch с CX=0000h.

Примечание 5: драйверы "мыши" фирмы Microsoft способны при разных условиях вызывать до четырех резидентных программ. Первая из них должна быть заявлена с помощью INT 33\AX=000Ch, а последующие – с помощью INT 33\AX=0018h (8.03-45).

8.03-41 INT 33\AX=000Fh – чувствительность перемещений "мыши"

При вызове:

AX = 000Fh
CX – число шагов на 8 пикселов по горизонтали (по умолчанию 8)
DX – число шагов на 8 пикселов по вертикали (по умолчанию 16).

8.03-42 INT 33\AX=0010h – локальный запрет отображения курсора.

При перемещении курсора "мыши" по экрану драйвер "мыши" восстанавливает изображение на каждом прежнем месте расположения курсора. Чтобы эти действия не мешали обновлению изображения, индикацию курсора "мыши" на обновляемом участке изображения желательно заранее запретить. В отличие от полного запрета посредством INT 33\AX=0002h (8.03-32), локальный запрет позволяет сделать мерцание курсора совсем незаметным. По завершении обновления заявленного участка индикацию курсора восстанавливают вызовом INT 33\AX=0001h (8.03-32).

При вызове:

AX = 0010h
CX – горизонтальная X-координата левого верхнего угла участка
DX – вертикальная Y-координата левого верхнего угла участка
SI – горизонтальная X-координата нижнего правого угла участка
DI – вертикальная Y-координата нижнего правого угла участка

8.03-43 INT 33\AX=0011h – проверка поддержки колеса прокрутки

При вызове:

AX = 0011h

При возврате:

AX = 574Dh – эта сигнатура подтверждает, что драйвер способен обслуживать "мышей" с колесом прокрутки (примечание 1)
CX – установленное состояние бита 0 означает, что используемая "мышь" снабжена колесом прокрутки.

Содержимое регистра BX может быть изменено.

Примечание 1: драйверы GMOUSE.COM (5.03-01) версии 9.06 и выше отвечают на вызов INT 33\AX=0011h возвратом сигнатуры AX = FFFFh. Она означает, что драйвер не обслуживает колесо прокрутки, но возвращает в регистре BX число активных кнопок "мыши".

8.03-44 INT 33\AX=0016h-0017h – сохранение состояния драйвера "мыши"

При вызове:

AX = 0016h – запись состояния драйвера в буфер
= 0017h – восстановление состояния по данным из буфера
BX – размер буфера для записи состояния (примечание 2).
ES:DX – адрес буфера (при AX=0017h он должен быть заполнен)

При возврате:

ES:DX – адрес буфера с данными состояния драйвера "мыши".

Примечание 1: восстановление состояния драйвера "мыши" должно происходить в том же видеорежиме, в котором сделана та запись состояния, по которой будет производиться восстановление.

Примечание 2: размер буфера для записи состояния драйвера "мыши" следует определить с помощью INT 33\AX=0015h: требуемый размер буфера в байтах будет возвращен в регистре BX.

8.03-45 INT 33\AX=0018h – вызов добавочных программ драйвером "мыши"

"Мышиные" драйверы фирмы Microsoft вызывают в зависимости от условий до 4-х резидентных программ. Первая из них должна быть заявлена с помощью INT 33\AX=000Ch (8.03-40), а последующие – с помощью INT 33\AX=0018h.

При вызове:

AX = 0018h
CX – условия вызова:
бит 0: – при движении "мыши"
бит 1: – при нажатии левой кнопки
бит 2: – при отпускании левой кнопки
бит 3: – при нажатии правой кнопки
бит 4: – при отпускании правой кнопки
бит 5: – во время удержания клавиши SHIFT
бит 6: – во время удержания клавиши CTRL
бит 7: – во время удержания клавиши ALT
ES:DX – адрес вызова программы командой CALL FAR (7.03-08)

При успешном завершении AX = 0018h, при неудаче AX = FFFFh.

Примечание 1: дополнительные программы вызываются только в случая нажатия на клавиатуре клавиш SHIFT, CTRL или ALT, так что хотя бы один из битов 5 – 7 в регистре CX должен быть установлен в единицу. Состояния старших битов 8 – 15 в регистре CX игнорируются. Остальные биты 0 – 4 в регистре CX учитываются по условию "ИЛИ", как при INT 33\AX=000Ch (8.03-40, примечание 1).

Примечание 2: при вызове программы на исполнение драйвер "мыши" оставляет в регистрах величины, перечисленные в примечании 2 к 8.03-40.

Примечание 3: чтобы прекратить обращения к ранее заявленной резидентной программе, нужно вызвать INT 33\AX=0018h повторно с тем же кодом условий в регистре CX, но с адресом 0000:0000h в ES:DX.

Примечание 4: поддержка INT 33\AX=0018h введена фирмой Microsoft в драйверы "мыши" начиная с версии 6.0. Драйверы "мыши", поставляемые другими фирмами, могут не поддерживать INT 33\AX=0018h.

8.03-46 INT 33\AX=0019h – адрес обращения к резидентной программе.

В ответ на вызов INT 33\AX=0019h драйвер "мыши" возвращает адрес обращения к той дополнительной резидентной программе, которая уже заявлена через INT 33\AX=0018h (8.03-45) и должна реагировать на условия, определенные значением в регистре CX. По возвращаемым данным любая программа, которая заменяет адрес обращения, сможет потом восстановить функционирование первоначально загруженной дополнительной резидентной программы.

При вызове:

AX = 0019h

CX – поисковое слово условий вызова (8.03-45)

Если не удалось найти такую резидентную программу, которая соответствует поисковому слову условий вызова, то при возврате CX = 0000h.

Если подходящая резидентная программа найдена, то при возврате BX:DX – адрес обращения к ней, а CX – то слово условий вызова (8.03-45), с которым эта программа была первоначально заявлена.

8.03-47 INT 33\AX=001Dh-001Eh – страница экрана для курсора "мыши"

При вызове:

AX = 001Dh – переадресация на другую экранную страницу

= 001Eh – выяснение адресуемой курсором экранной страницы

BX – номер новой страницы (только для INT 33\AX=001Dh)

При возврате из INT 33\AX=001Eh:

BX – номер адресуемой экранной страницы.

8.03-48 INT 33\AX=001Fh-0020h – включение и отключение драйвера "мыши"

При вызове:

AX = 001Fh – прекращение действия драйвера "мыши"
= 0020h – возобновление действия драйвера "мыши"

При успешном завершении значение в AX сохраняется, при неудаче AX = FFFFh.

Примечание 1: вызов INT 33\AX=001Fh восстанавливает прежние указатели на обработчики прерываний INT 10 и INT 74, которые были установлены до загрузки драйвера "мыши".

Примечание 2: вызов INT 33\AX=0020h возвращает указатели, поставленные драйвером "мыши", только если они были сняты посредством INT 33\AX=001Fh.

Примечание 3: при успешном завершении INT 33\AX=001Fh возвращает в регистрах ES:BX адрес обработчика прерывания INT 33, который был установлен до загрузки драйвера "мыши". Если записать этот адрес обратно в таблицу прерываний, то возможность вызова драйвера "мыши" через прерывания будет полностью исключена.

8.03-49 INT 33\AX=0021h – выяснение установлен ли драйвер "мыши"

При вызове:

AX = 0021h

Если драйвер "мыши" не установлен, то при возврате обычно AX = 0021h.

Если при возврате AX = FFFFh, то драйвер "мыши" установлен, и тогда

BX – число кнопок "мыши", причем для двухкнопочной "мыши" может быть возвращено значение BX = FFFFh

Примечание 1: MS-DOS7 заполняет свободные ячейки таблицы прерываний (до INT 3F) отсылками к команде IRET (7.03-30). Когда драйвер "мыши" не загружен, эта команда IRET просто возвращает неизмененное состояние регистра AX. Выяснить, установлен ли драйвер "мыши" и сразу же привести его в начальное состояние можно с помощью функции INT 33\AX=0000h (8.03-31). Функция INT 33\AX=0021h не устанавливает драйвер в начальное состояние, но тем не менее сбрасывает счетчик поворота колеса прокрутки.

8.03-50 INT 33\AX=0024h – выяснение типа "мыши" и номера линии IRQ

При вызове:

AX = 0024h

При неудаче AX = FFFFh. Если в AX любое другое значение, то

BH.BL – номер версии драйвера "мыши"

CH = 00h – "мышь" не подключена
= 01h – "мышь" подключена через карту расширения
= 02h – "мышь" для последовательного порта
= 04h – "мышь" для порта PS/2,
= 05h – "мышь" фирмы Hewlett-Packard
CL – шестнадцатеричный номер линии IRQ для "мыши" (для "мышей", подключенных к порту PS/2, возвращается CL = 00h)

8.03-51 INT 33\AX=0026h – предельные значения координат курсора "мыши"

При вызове:
AX = 0026h
При неудаче в BX ненулевое значение. Если же при возврате BX = 0000h, то:
CX – максимальная X-координата в установленном видеорежиме
DX – максимальная Y-координата в установленном видеорежиме

Примечание 1: обработчик прерывания INT 33\AX=0026h выдает координаты для того случая, когда перемещение курсора "мыши" не ограничено "окном". Иначе следует вызывать INT 33\AX=0031h(8.03-54).
Примечание 2: прежде чем определять максимальные значения координат, нужно с помощью INT 33\AX=0032h (8.03-55) выяснить, поддерживает ли установленный драйвер "мыши" функцию INT 33\AX=0026h.

8.03-52 INT 33\AX=0028h – изменение видеорежима драйвером "мыши"

Видеорежим определяет не только геометрические соотношения, но также форматы данных в видеопамяти. Чтобы обращения драйвера "мыши" к видеопамяти были бы скорректированы согласно установленному видеорежиму, нужно переключать видеорежимы посредством драйвера "мыши". Только тогда смена видеорежима не нарушит перемещение и отображение курсора "мыши".

При вызове:
AX = 0028h
CX – код нового видеорежима из таблицы A.10-1
DH – вертикальный размер знакоместа (примечание 1)
DL – горизонтальный размер знакоместа (примечание 1)
При успешном завершении CL = 00h. При неудаче в CL – ненулевое значение.

Примечание 1: чтобы установить принимаемый по умолчанию размер знакоместа для данного видеорежима, нужно при вызове задать DH = DL = 00h. Если запрошенный видеорежим не поддерживает изменение размеров знакоместа, то значения в DH и в DL игнорируются.

Примечание 2: вызов с CX=0000h не переустанавливает видеорежим, но сбрасывает флаг смены видеорежима. Этот флаг надо сбрасывать перед приведением драйвера "мыши" в начальное состояние.

Примечание 3: прежде чем переустанавливать видеорежим, нужно с помощью INT 33\AX=0032h (8.03-55) выяснить, поддерживает ли установленный драйвер "мыши" функцию INT 33\AX=0028h.

Примечание 4: перечень видеорежимов, поддерживаемых драйвером "мыши", можно получить посредством нескольких последовательных вызовов INT 33\AX=0029h. Первый вызов принимает CX=0000h и возвращает в CX код первого видеорежима. С этим кодом в CX производится следующий вызов и возвращает в CX код второго видеорежима, и так далее. Некоторые драйверы возвращают в DS:DX указатель на строку описания видеорежима, другие возвращают DS:DX=0000:0000h. Окончание цикла выдачи кодов отмечается возвратом CX=0000h.

8.03-53 INT 33\AX=002Ah – параметры отображения курсора "мыши"

При вызове:

AX = 002Ah

При возврате:

AX – число запретов отображения (INT 33\AX=0002h, 8.03-32).
BX – сдвиг вершины курсора относительно X-координаты
CX – сдвиг вершины курсора относительно Y-координаты
DX – тип "мыши", как в CH после INT 33\AX=0024h (8.03-50).

Примечание 1: координаты курсора определяют положение левой верхней угловой точки курсорного блока. Вершина курсора – это точка, куда указывает курсор. Ее сдвиг относительно левой верхней угловой точки может быть от–127 до +127 по обеим координатам.

Примечание 2: прежде чем выяснять параметры отображения курсора, нужно с помощью INT 33\AX=0032h (8.03-55) выяснить, поддерживает ли установленный драйвер "мыши" функцию INT 33\AX=002Ah.

8.03-54 INT 33\AX=0031h – пределы изменения координат курсора "мыши"

Обработчик прерывания INT 33\AX=0031h сообщает допустимые пределы изменения координат курсора "мыши", когда его перемещение ограничено "окном", заданным посредством INT 33\AX=0007h-0008h (8.03-36).

При вызове:

AX = 0031h

При возврате:

AX – минимальное значение X-координаты (по горизонтали)
BX – минимальное значение Y-координаты (по вертикали)
CX – максимальное значение X-координаты (по горизонтали)
DX – максимальное значение Y-координаты (по вертикали)

Примечание 1: когда доступная курсору площадь экрана не ограничена "окном", тогда предпочтительно пользоваться INT 33\AX=0026h (8.03-51).

Примечание 2: прежде чем определять пределы изменения координат, нужно с помощью INT 33\AX=0032h (8.03-55) выяснить, поддерживает ли установленный драйвер "мыши" функцию INT 33\AX=0031h.

8.03-55 INT 33\AX=0032h – доступные функции драйвера "мыши"

При вызове:
AX = 0032h

При возврате:
AX – слово, в котором установление в единицу каждого бита свидетельствует о поддержке одной из функций драйвера:
бит 3: – поддерживается INT 33\AX=0031h
бит 10: – поддерживается INT 33\AX=002Ah
бит 11: – поддерживается INT 33\AX=0029h
бит 12: – поддерживается INT 33\AX=0028h
бит 14: – поддерживается INT 33\AX=0026h
содержимое регистров BX, CX, DX может быть изменено.

8.03-56 INT 4A – вызов обработчика ежедневного действия

Устанавливаемый по умолчанию обработчик прерывания INT 4A просто возвращает управление вызывающей программе. В роли вызывающей программы выступает система BIOS, которая с определенной периодичностью вызывает INT 4A, если время вызова задано с помощью INT 1A\AH=06h (8.01-94). Посредством перехвата INT 4A другим, специально установленным обработчиком можно организовать исполнение определенных действий в заданное время.

Примечание 1: при написании альтернативного обработчика прерывания INT 4A следует учитывать нереентерабельность DOS и принимать соответствующие меры в случае необходимости (8.02-70, 8.02-87).

8.03-57 INT 67\AH=41h – получение сегментного адреса кадра страниц

Драйвер EMM386.EXE (5.04-02) по умолчанию организует 4 страницы доступа к расширенной памяти, сгруппированные в один 64-килобайтный кадр. Обычно этот кадр располагается начиная с сегментного адреса E000h.

При вызове:

AH = 41h

При возврате:

AH – код завершения из таблицы A.06-1; если AH = 00h, то
BX – начальный сегментный адрес кадра страниц.

Примечание 1: перед обращением к INT 67\AH=41h нужно с помощью INT 67\AH=46h (примечание 1 к 8.03-62) убедиться в том, что драйвер EMM386.EXE (5.04-02), обслуживающий вызовы INT 67, загружен и активизирован.

8.03-58 INT 67\AH=42h – получение числа страниц расширенной памяти

Разметке на 16-килобайтные страницы EMS-памяти подлежит адресное пространство от 1088 до 32768 кбайт, за исключением участков, которые резервированы (параметром /L, 5.04-02) для доступа посредством XMS-драйвера (5.04-01).

При вызове:

AH = 42h

При возврате:

AH – код завершения из таблицы A.06-1; если AH = 00h, то
BX – число свободных страниц расширенной памяти
DX – полное число страниц расширенной памяти.

Примечание 1: перед обращением к INT 67\AH=42h нужно с помощью INT 67\AH=46h (примечание 1 к 8.03-62) убедиться в том, что драйвер EMM386.EXE (5.04-02), обслуживающий вызовы INT 67, загружен и активизирован.

8.03-59 INT 67\AH=43h – выделение расширенной памяти и номерной ссылки

В отличие от INT 21\AH=3Dh (8.02-33), драйвер EMM386.EXE (5.04-02) выделяет и обслуживает ссылки, относящиеся только к участкам расширенной памяти. Каждый такой участок содержит целое число 16-килобайтных (логических) страниц. Для идентификации отдельной страницы расширенной памяти необходимо указывать номерную ссылку и, кроме того, номер логической страницы в пределах относящегося к данной ссылке участка.

При вызове:

AH = 43h
BX – запрашиваемое (ненулевое) число логических страниц расширенной памяти, по 16 кбайт каждая.

При возврате:

AH – код завершения из таблицы A.06-1; если AH = 00h, то
DX – номерная ссылка на выделенный участок расширенной памяти.

Примечание 1: перед обращением к INT 67\AH=43h нужно с помощью INT 67\AH=46h (примечание 1 к 8.03-62) убедиться в том, что драйвер EMM386.EXE (5.04-02), обслуживающий вызовы INT 67, загружен и активизирован.

Примечание 2: драйвер EMM386.EXE по умолчанию выделяет до 64 ссылок. Параметром "h" (5.04-02) этот предел может быть увеличен до 255.

Примечание 3: согласно стандарту LIM EMS 4.0 та же операция может быть выполнена посредством вызова INT 67\AX=5A00h. Спецификации вызова (кроме AX=5A00h) такие же, но INT 67\AX=5A00h допускает запрос нулевого количества логических страниц.

8.03-60 INT 67\AH=44h – отображение логической страницы на физическую

Здесь под отображением подразумевается установление такого соответствия, при котором любое обращение к 16-килобайтной (физической) странице в пределах до 1 мегабайта автоматически переадресуется к другой (логической) странице в адресном пространстве за пределами 1 мегабайта.

При вызове:

AH = 44h
AL – номер физической страницы для отображения;
BX – номер логической страницы, которую надо отобразить, или =FFFFh – для высвобождения указанной физической страницы;
DX – номерная ссылка, к которой относится логическая страница.

При возврате:

AH – код завершения (A.06-1); при успешном завершении AH = 00h.

Примечание 1: перед обращением к INT 67\AH=44h нужно с помощью INT 67\AH=46h (примечание 1 к 8.03-62) убедиться в том, что драйвер EMM386.EXE (5.04-02), обслуживающий вызовы INT 67, загружен и активизирован.

Примечание 2: счет страниц (и физических, и логических) начинается с нуля.

Примечание 3: физические страницы 00h – 03h входят в состав кадра, расположение которого показывает INT 67\AH=41h (8.03-57). Расположение физических страниц с номерами 04h и выше следует

определять посредством INT 67\AX=5800h (8.03-70). Если запрашиваемая физическая страница размещается ниже границы 640 кбайт, то соответствующее ей пространство памяти должно быть заранее выделено операционной системой той программе, которая будет пользоваться этой страницей.

8.03-61 INT 67\AH=45h – аннулирование ссылки и высвобождение памяти

Когда номерная ссылка и соответствующий ей участок расширенной памяти становятся больше не нужны, тогда программа, которой эта ссылка выделена, должна сообщить об этом драйверу EMM386.EXE (5.04-02), чтобы данный участок расширенной памяти стал считаться свободным.

При вызове:

AH = 45h

DX – номерная ссылка, которую надо аннулировать

При возврате:

AH – код завершения (A.06-1); при успешном завершении AH = 00h.

Примечание 1: аннулировать с помощью INT 67\AH=45h можно только те ссылки, которые были выделены драйвером EMM386.EXE (5.04-02) посредством INT 67\AH=43h (8.03-59) или INT 67\AX=5A00h. Другие номерные ссылки следует аннулировать посредством INT 21\AH=68h, INT 21\AH=6Ah или INT 21\AH=3Eh (8.02-34).

Примечание 2: доступ к тем страницам расширенной памяти, которые соответствовали аннулированной ссылке, нельзя возобновить посредством повторного запроса ссылки.

8.03-62 INT 67\AH=46h – выяснение версии драйвера EMS- памяти

При вызове:

AH = 46h

При возврате:

AH – код завершения из таблицы A.06-1; если AH = 00h, то

AL – номер версии действующего драйвера EMM386.EXE (5.04-02).

Примечание 1: способ обнаружить факт загрузки драйвера EMM386.EXE состоит в считывании двухбайтового сегментного адреса обработчика прерывания INT 67 из ячейки 0000:019Eh и в проверке наличия имени драйвера (EMMXXXX0) в поле имени, начинающемся со смещения 000Ah относительно считанного сегментного адреса. До подтверждения факта загрузки драйвера EMM386.EXE ни одну из функций INT 67 вызывать нельзя, так как адрес в указанной ячейке

может быть недействителен (например, 0000h). Когда факт загрузки драйвера EMM386.EXE подтвержден, тогда возврат значения AH = 00h обработчиком прерывания INT 67\AH=46h будет свидетельствовать о том, что драйвер EMM386.EXE находится в активном, действующем состоянии.

Примечание 2: если факт загрузки драйвера EMM386.EXE подтвержден, и при этом обработчик прерывания INT 67\AH=46h возвращает в регистре AH ненулевой код завершения, значит, драйвер EMM386.EXE находится в неактивном состоянии. В таком случае с помощью INT 67\AX=FFA5h (8.03-74) можно найти адрес API драйвера EMM386.EXE, и, обратившись по этому адресу, перевести драйвер в активное состояние.

8.03-63 INT 67\AH=4Bh – число ссылок для доступа к расширенной памяти

При вызове:

AH = 4Bh

При возврате:

AH – код завершения из таблицы A.06-1; если AH = 00h, то
BX – число ссылок, задействованных драйвером EMM386.EXE.

Примечание 1: перед обращением к INT 67\AH=4Bh нужно с помощью INT 67\AH=46h (примечание 1 к 8.03-62) убедиться в том, что драйвер EMM386.EXE (5.04-02), обслуживающий вызовы INT 67, загружен и активизирован.

Примечание 2: драйвер EMM386.EXE по умолчанию выделяет до 64 ссылок. Параметром "h" (5.04-02) этот предел может быть увеличен до 255.

8.03-64 INT 67\AH=4Ch – число логических страниц номерной ссылки

При вызове:

AH = 4Ch
DX – номерная ссылка на участок расширенной памяти

При возврате:

AH – код завершения из таблицы A.06-1; если AH = 00h, то
BX – число логических страниц в участке расширенной памяти, относящемся к указанной номерной ссылке.

Примечание 1: вызов INT 67\AH=4Ch обслуживает только те номерные ссылки, которые выданы посредством INT 67\AH=43h (8.03-59) драйвером EMM386.EXE (5.04-02). Перед обращением к INT 67\AH=4Ch нужно с помощью INT 67\AH=46h (примечание 1 к 8.03-62)

убедиться в том, что драйвер EMM386.EXE загружен и активизирован.

8.03-65 INT 67\AH=4Eh – сохранение состояния расширенной памяти.

При написании кода резидентных модулей, пользующихся расширенной памятью, следует иметь ввиду, что любая программа, прерываемая вызовом резидентного модуля, тоже имела право пользоваться расширенной памятью. Чтобы после завершения миссии резидентного модуля исполнение прерванной программы можно было бы продолжить, исходное отображение страниц расширенной памяти нужно восстановить. Для этого предусмотрены 4 подфункции прерывания INT 67\AH=4Eh, позволяющие выяснить размер блока памяти для записи состояния, осуществить запись данных об отображении страниц в этот блок памяти, а впоследствии восстановить исходное отображение страниц по записанным заранее данным.

При вызове:

AH = 4Eh
AL – подфункция:
= 00h – сохранение данных о текущем состоянии
= 01h – восстановление прежнего состояния по записи
= 02h – последовательное исполнение подфункций 00h и 01h
= 03h – определение размера блока памяти для записи
ES:DI – адрес для записи данных (только для подфункций 00h и 02h)
DS:SI – адрес блока данных (только для подфункций 01h и 02h)

При возврате:

AH – код завершения из таблицы A.06-1; если AH = 00h, то
AL – размер блока данных в байтах (только после подфункции 03h)

Примечание 1: вызов INT 67\AH=4Eh обслуживается драйвером расширенной памяти EMM386.EXE (5.04-02) версии не ниже 4.00, так что сначала следует с помощью INT 67\AH=46h (примечание 1 к 8.03-62) убедиться в том, что драйвер EMM386.EXE загружен, активизирован, и что его версия соответствует требуемой.

Примечание 2: начиная с версии 3.00 драйвера EMM386.EXE сохранение и восстановление состояния отображения расширенной памяти возможны с помощью функций INT 67\AH=47h и INT 67\AH=48h соответственно. Эти функции сохраняют данные в памяти, выделенной драйверу EMM386.EXE (отдельный блок данных не нужен), сохраняют состояние только одного 64-килобайтного кадра страниц и требуют указания в регистре DX номерной ссылки, предоставленной драйвером EMM386.EXE запрашивающему резидентному модулю.

8.03-66 INT 67\AX=5000h – соответствие логических и физических страниц

Посредством прерывания INT 67\AX=5000h драйвер EMM386.EXE (5.04-02) принимает новую таблицу соответствия логических и физических страниц, относящихся к одной номерной ссылке. Один вызов INT 67\AX=5000h эквивалентен нескольким последовательным вызовам INT 67\AH=44h (8.03-60).

При вызове:

AX = 5000h
CX – число записей в таблице соответствия (примечание 3)
DX – номерная ссылка
DS:SI – указатель на начало таблицы соответствия

При возврате:

AH – код завершения (A.06-1); при успешном завершении AH = 00h.

Примечание 1: вызов INT 67\AH=5000h обслуживает только те номерные ссылки, которые выданы посредством INT 67\AH=43h (8.03-59) драйвером EMM386.EXE. Перед обращением к INT 67\AH=5000h нужно с помощью INT 67\AH=46h (примечание 1 к 8.03-62) убедиться в том, что драйвер EMM386.EXE (5.04-02) загружен и активизирован.

Примечание 2: если запрашиваемая физическая страница размещается ниже границы 640 кбайт, то соответствующее ей пространство памяти должно быть заранее выделено операционной системой той программе, которая будет пользоваться этой страницей.

Примечание 3: каждая запись в таблице соответствия занимает 4 байта и состоит из двух слов, из которых второе – это номер физической страницы (обычно 0000h – 0003h). Для операции установления соответствия (mapping) первым словом в записи должен быть номер логической страницы. Для операции разрыва соответствия (unmapping) в первом слове записи нужно указывать значение FFFFh.

Примечание 4: функция INT 67\AX=5001h делает то же самое и принимает те же спецификации, за исключением того, что вторым словом в каждой записи таблицы соответствия должен быть сегментный адрес соответствующей физической страницы.

8.03-67 INT 67\AH=51h – изменение числа выделенных логических страниц

При вызове:

AH = 51h
BX – требуемое полное число логических страниц
DX – номерная ссылка

При возврате:

AH – код завершения из таблицы A.06-1; если AH = 00h, то

BX – новое полное число страниц, доступных по данной ссылке.

Примечание 1: вызов INT 67\AH=51h обслуживает только те номерные ссылки, которые выданы посредством INT 67\AH=43h (8.03-59) драйвером EMM386.EXE. Перед обращением к INT 67\AH=51h нужно с помощью INT 67\AH=46h (примечание 1 к 8.03-62) убедиться в том, что драйвер EMM386.EXE (5.04-02) загружен и активизирован.

Примечание 2: требуемое число страниц может быть больше или меньше числа тех, которые уже доступны по данной ссылке. Номера добавляемых логических страниц будут следовать за номерами тех, которые были доступны раньше. Если же вызов INT 67\AH=51h уменьшает задействованное число страниц, то высвобождение памяти будет происходить начиная со страниц с наибольшим номером.

8.03-68 INT 67\AH=55h-56h – переход и вызов подпрограммы в EMS-памяти

При исполнении операций дальней передачи управления (JMP FAR и CALL FAR) в расширенной памяти доступ к логической странице назначения необходимо подготавливать заранее. В связи с этим драйвер EMM386.EXE предоставляет две функции, сочетающие дальнюю передачу управления с заменой таблицы соответствия логических и физических страниц для одной номерной ссылки. Функция INT 67\AH=55h заменяет таблицу соответствия и выполняет дальний переход (JMP FAR), а функция INT 67\AH=56h заменяет таблицу соответствия и выполняет дальний вызов подпрограммы (CALL FAR). Параметры перехода и вызова передаются обработчику прерывания в составе заранее подготовленного блока данных, показанного в приложении A.12-6.

При вызове:

AX = 5500h для осуществления дальнего перехода
= 5600h для осуществления дальнего вызова подпрограммы
DX – номерная ссылка, выданная драйвером EMM386.EXE
DS:SI – указатель на блок данных A.12-6.

При возврате:

AH – код завершения (A.06-1); при успешном завершении AH = 00h.

Примечание 1: пользоваться операциями INT 67\AH=55h-56h могут только те программы, которые реально исполняются в расширенной памяти.

Примечание 2: операции INT 67\AX=5501h и INT 67\AX=5601h также исполняют переход и вызов подпрограммы, но отличаются тем, что принимают в составе таблиц соответствия не номера физических страниц, а их сегментные адреса.

Примечание 3: операция INT 67\AX=5602h не требует никаких других исходных сведений (кроме AX) и возвращает в регистре BX число байтов,

которое займут адреса возврата, сохраняемые в стеке операциями вызова подпрограммы (INT 67\AX=5600h-5601h).

8.03-69 INT 67\AX=5700h-5701h – копирование данных или обмен данными

Операции INT 67\AX=5700h-5701h копируют данные или выполняют обмен данных между участками памяти, доступными через разные номерные ссылки и находящимися как в расширенной, так и в обыкновенной памяти. Вся адресация осуществляется через дескриптор, показанный в таблице A.12-5.

При вызове:

AX = 5700h – копирование данных из одного участка в другой
= 5701h – обмен данными между участками памяти
DS:SI – указатель на дескриптор, показанный в таблице A.12-5

При возврате:

AH – код завершения (A.06-1); при успешном завершении AH = 00h.

Примечание 1: вызовы INT 67\AH=57h обслуживают только те номерные ссылки, которые выданы посредством INT 67\AH=43h (8.03-59) драйвером EMM386.EXE. Перед обращением к INT 67\AH=57h нужно с помощью INT 67\AH=46h (примечание 1 к 8.03-62) убедиться в том, что драйвер EMM386.EXE (5.04-02) загружен и активизирован.

8.03-70 INT 67\AX=5800h – сегментные адреса физических страниц

При вызове:

AX = 5800h
ES:DI – указатель на буфер для данных

При возврате:

AH – код завершения из таблицы A.06-1; если AH = 00h, то
CX – число записей в буфере (по 4 байта каждая);
ES:DI – указатель на заполненный буфер (примечание 2)

Примечание 1: перед обращением к INT 67\AH=58h нужно с помощью INT 67\AH=46h (примечание 1 к 8.03-62) убедиться в том, что драйвер EMM386.EXE (5.04-02), обслуживающий вызовы INT 67, загружен и активизирован.

Примечание 2: каждая запись в буфере состоит из двух слов: первое слово – сегментный адрес физической страницы, а второе – номер этой физической страницы.

Примечание 3: число физических страниц и, следовательно, требуемый размер буфера, могут быть определены заранее с помощью INT 67\AX=5801h, которое точно так же возвращает число записей

в регистре CX, но не заполняет буфер данными и игнорирует содержимое ES:DI.

8.03-71 INT 67\AX=DE06h – определение физического адреса страницы.

Данная служебная функция VCPI-серверов дает возможность составить представление о том, как механизм страничной переадресации в 32-разрядных процессорах преобразует адресное пространство в области UMB после установления защищенного режима работы процессора. Поскольку роль сервера VCPI в MS-DOS7 играет драйвер EMM386.EXE (5.04-02), постольку перед обращением к INT 67\AX=DE06h нужно убедиться, что драйвер EMM386.EXE загружен и активизирован (примечание 1 к 8.03-62), а также что обслуживание функций VCPI не запрещено параметром /noVCPI (5.04-02).

При вызове:

AX = DE06h

CX – номер страницы в пределах до 1 Мегабайта (примечание 1)

Успешное завершение отмечается значением AH = 00h, и тогда

EDX – физический адрес запрошенной страницы (примечание 2).

Неудача отмечается ненулевым кодом завершения в AH, причем

AH = 8Bh обычно означает ошибку в указании номера страницы.

Примечание 1: в отличие от функций LIM EMS, функции VCPI оперируют страницами памяти размером 4 кбайт – теми же, с которыми работает механизм страничной переадресации в 32-разрядных процессорах. Поэтому здесь номер страницы – это ее линейный адрес, сдвинутый вправо на 12 бит. Например, ячейке памяти D400:1ABCh будет соответствовать линейный адрес D5ABCh и, следовательно, номер страницы CX=00D5.

Примечание 2: способ доступа к EDX и другим 32-разрядным регистрам из программ DOS показан в разделе 7.02-06.

8.03-72 INT 67\AX=DE07h – считывание из управляющего регистра CR0.

В отличие от операции считывания состояния регистра CR0 командой MOV (примечание 1 к 7.03-58), данная служебная функция VCPI-серверов действует в защищенном режиме на третьем уровне привилегий. Поскольку роль сервера VCPI в MS-DOS7 играет драйвер EMM386.EXE (5.04-02), постольку перед обращением к INT 67\AX=DE07h нужно убедиться, что драйвер EMM386.EXE загружен и активизирован (примечание 1 к 8.03-62), а также что обслуживание функций VCPI не запрещено параметром /noVCPI (5.04-02).

При вызове:

AX = DE07h

При возврате ненулевое значение AH означает неудачу, но если AH = 00h, то

EBX – считанное состояние управляющего регистра CR0.

Примечание 1: способ доступа к EBX и другим 32-разрядным регистрам из программ DOS показан в разделе 7.02-06.

8.03-73 INT 67\AX=DE08h-DE09h – обращение к отладочным регистрам

В отличие от операций доступа к отладочным регистрам посредством команды MOV (примечание 1 к 7.03-58), данные служебные функции VCPI-серверов действуют в защищенном режиме на третьем уровне привилегий. Поскольку роль сервера VCPI в MS-DOS7 играет драйвер EMM386.EXE (5.04-02), постольку перед обращением к INT 67\AX=DE07h–DE09h нужно убедиться, что драйвер EMM386.EXE загружен и активизирован (примечание 1 к 8.03-62), а также что обслуживание функций VCPI не запрещено параметром /noVCPI (5.04-02).

При вызове:

AX = DE08h – считывание из отладочных регистров в буфер

= DE09h – запись из буфера в отладочные регистры

ES:DI – указатель на 32-байтовый буфер с данными или для данных

При возврате:

AH – код завершения (A.06-1); при успешном завершении AH = 00h.

Примечание 1: буфер заполняется данными по 4 байта на регистр, начиная с регистра DR0 и кончая DR7, причем данные из регистров DR4 и DR5 не выводятся. При записи соответствующие им 8 байт игнорируются. Роль каждого отладочного регистра описана в разделе A.11-5.

8.03-74 INT 67\AX=FFA5h – получение адреса API драйвера EMM386.EXE

Данная функция, введенная стандартом LIM EMS версии 4.2, уникальна тем, что исполняется резидентным модулем драйвера EMM386.EXE (5.04-02) даже тогда, когда он находится в неактивном состоянии и игнорирует все другие запросы. Это не исключает, однако, необходимости проверять факт загрузки драйвера (примечание 1 к 8.03-62) перед вызовом INT 67\AX=FFA5h.

При вызове:

AX = FFA5h

При возврате:

AH = 84h – сигнатура успешного завершения, и тогда

BX:CX – адрес вызова API-функций драйвера EMM386.EXE.

Примечание 1: переход командой CALL FAR (7.03-08) по адресу BX:CX вызывает исполнение операции, определяемой значением в регистре AX:

AX = 0100h – перевод EMM386.EXE в активное состояние;
AX = 0101h – перевод EMM386.EXE в пассивное состояние;
AX = 0500h – выведение сообщения о текущем состоянии.

Если UMB-блоки или EMS-страницы задействованы, то запрос на перевод EMM386.EXE в пассивное состояние не будет исполнен.

8.03-75 INT 70 – INT 77: обслуживание запросов IRQ 8 – IRQ 15

Эта группа обработчиков прерываний обслуживает запросы, поступающие по линиям IRQ 8 – IRQ 15 от устройств компьютера ко второму контроллеру прерываний, который, в свою очередь, посылает вызовы в линию IRQ 2 первого контроллера прерываний (8.01-09). Поступление запроса по каждой из линий IRQ 8 – IRQ 15 может быть заблокировано путем посылки в порт A1h бита маски, указанного в третьей колонке приведенной ниже таблицы. Некоторые линии выделены для приема запросов от определенных устройств, указанных в четвертой колонке таблицы, но остальные готовы принять запрос от любого устройства, если оно сконфигурировано для посылки запроса по этой линии и поддерживается драйвером, загружающим обработчик соответствующего прерывания.

Вызов	Линия	Маска	Источник запросов	Примечания
INT 70	IRQ 8	бит 0	Такты реального времени	*1
INT 71	IRQ 9	бит 1	-	-
INT 72	IRQ 10	бит 2	-	-
INT 73	IRQ 11	бит 3	-	-
INT 74	IRQ 12	бит 4	-	*2
INT 75	IRQ 13	бит 5	Арифметический сопроцессор	-
INT 76	IRQ 14	бит 6	Первый IDE-контроллер	-
INT 77	IRQ 15	бит 7	-	*3

Примечание 1: вызовы INT 70 следуют 1024 раз в секунду. Во многих системах BIOS следование тактов выключено и включается только в интервалах ожидания (INT 15\AH=83h, 8.01-73).

Примечание 2: вероятным источником запросов по линии IRQ 12 является "мышь", подключенная к порту PS2, если она используется.

Примечание 3: наиболее вероятным источником запросов по линии IRQ 15 является второй IDE-контроллер или SCSI-контроллер, если хотя бы один из них имеется.

ГЛАВА 9 Примеры композиции исполняемых файлов

Все примеры интерпретируемых, исполняемых и конфигурационных файлов, представленных в 9-й главе, проверены на нескольких разных AT-совместимых компьютерах с процессорами от 486SL (1992-го года) до Pentium-D (2006-го года). Тем не менее предусмотреть все заранее невозможно. Правильнее рассматривать представленные примеры как рекомендуемые схемы, из которых каждому предоставлена возможность выбрать и использовать то, что ему нужно. Несмотря на различия в частностях, с некоторыми базовыми действиями и базовыми условиями придется иметь дело каждому, кто намерен работать в DOS.

Для преобразования книжного текста в файл нужно запустить на исполнение программу редактирования, которая могла бы сохранять неформатированный текст. Из тех программ редактирования, которые поставляются в составе операционной системы Windows, программы WORD и WORDPAD для этого не годятся. Подойдут программы NOTEPAD и EDIT.COM, но для введения национального текста в комментариях годится только EDIT.COM, и то при условии правильной установки драйверов национальной адаптации для DOS (примеры – в разделах 9.01 и 9.04).

В меню FILE программы редактирования пункт NEW открывает новый пустой файл и предоставляет Вам возможность ввести текст. Если не оговорено иное, тексты файлов из этой книжки нужно вводить построчно без отступа слева, вплотную к левой кромке текстового поля. Набранный текст надо сохранить командой SAVE AS из меню FILE программы редактирования, при этом для файла будут запрошены имя и суффикс. Поскольку суффикс определяет роль файла (2.01-02), постольку batch-файлы должны получить суффикс *.BAT, командные файлы для отладчика DEBUG.EXE – суффикс *.SCR, разные конфигурационные файлы – суффиксы *.SYS, *.MNU или *.EXT.

Длинные файлы рекомендуется сохранять по мере их набора, первый раз командой SAVE AS, и потом несколько раз до завершения набора – командой SAVE. Если файл приходится корректировать "на ходу" или набирать "с чистого листа", то полезно проводить тестирование набираемого текста по частям, как показано в разделе 9.07-02. Миссия программы редактирования завершается, когда файл с заданным Вами именем полностью набран, проверен и сохранен.

Для успешного исполнения или интерпретации набранного файла должны быть подготовлены определенные условия. Если в конкретных случаях не оговорено иное, то по умолчанию предполагается следующее:

- имя резидентного командного интерпретатора (Command.com или его заменяющего) вместе с полным путем к нему должны быть определены в переменной окружения COMSPEC;
- не допускается присвоение атрибутов H (скрытый) и S (системный) ни одной из программ, которые предстоит вызывать по ходу исполнения набранного файла (примечание 1 к 9.11-02);
- все программы, которые приходится вызывать по ходу исполнения, должны быть размещены в каталогах, пути к которым перечислены в значении переменной окружения PATH (2.02-02);
- ни в текущем каталоге, ни вдоль путей, записанных в переменную окружения PATH, не должно быть одноименных программ, непригодных для работы в среде используемой версии MS-DOS;
- переменная окружения TEMP должна указывать путь к существующему каталогу, предназначенному для записи временных файлов и предоставляющему достаточно свободного места для этой цели.

Важно подчеркнуть, что в каталоге для временных файлов не должно быть таких файлов, которые нельзя было бы удалить или перезаписать. Перед запуском программ полезно командой SET проверить установленные значения переменных окружения. И значения переменных окружения, и другие условия исполнения программ задаются заранее в конфигурационных файлах, разные образцы которых приведены в разделах 9.04 и 9.09. Но начинать надо, конечно, с самых простых конфигурационных файлов, показанных в следующем разделе 9.01.

9.01 Примеры простых конфигурационных файлов.

Процесс загрузки MS-DOS7 зависит от многих факторов, и его результат может выглядеть по-разному. Загрузчик – файл IO.SYS – принимает во внимание параметры, записанные в файл MSDOS.SYS (5.01-01), а также состав группы предоставляемых ему файлов DOS. Полный список файлов DOS, которые должны находиться в корневом каталоге загрузочного диска, приведен в разделе 9.11-02. В этом списке имеются два необязательных, но очень важных конфигурационных файла: CONFIG.SYS и AUTOEXEC.BAT. Именно от них зависит та конфигурация MS-DOS7, которая получится в результате загрузки. MS-DOS7 способна загружаться и без них, но принимаемая по умолчанию конфигурация слишком бедна и служить образцом никак не может.

Чтобы MS-DOS7 предстала перед пользователем эффективной, удобной и дружественной, пользователю надо самому побеспокоиться о составлении соответствующих конфигурационных файлов. Для работы на современных компьютерах даже простые конфигурации содержат драйверы расширенной памяти, "мыши" и оптического дисковода. Невозможно представить себе удобство

пользования MS-DOS7 без подходящего файл-менеджера. Показываемые в данном разделе примеры простых конфигурационных файлов включают также стандартную загрузку национальных кодовых страниц.

Простые конфигурационные файлы потенциально пригодны для загрузки MS-DOS7 со сменных носителей, однако область их целесообразного применения – это установка MS-DOS7 на жесткий магнитный диск: либо временная установка после форматирования диска, либо постоянная установка при выборочной альтернативной загрузке нескольких операционных систем (9.11-02).

9.01-01 Файл CONFIG.SYS: простой вариант.

Файл CONFIG.SYS – это интерпретируемый командный файл: каждая его строка представляет собой команду. Термин "интерпретируемый" означает, что команды исполняются процессором не непосредственно, а через особую программу – командный интерпретатор. По отношению к командам из файла CONFIG.SYS интерпретатором является загрузчик IO.SYS.

В приведенном ниже файле CONFIG.SYS загрузка всех драйверов показана в явной форме согласно предпочтительному порядку. Предполагается, что драйверы находятся в каталоге \DOS\DRV. Если Вы используете другую структуру каталогов, то спецификации путей надо будет соответствующим образом изменить. Обратите внимание, что в спецификациях путей не указана буква диска. Такие спецификации годятся для загрузки с любого диска.

```
device=\DOS\DRV\Himem.sys
device=\DOS\DRV\Emm386.exe ram v
dos=high,umb,noauto
buffershigh=20,0
fileshigh=30
lastdrivehigh=Z
fcbshigh=1,0
stackshigh=9,256
numlock off
country=007,866,\DOS\DRV\Country.sys
devicehigh=\DOS\DRV\Dblbuff.sys
devicehigh=\DOS\DRV\Ifshlp.sys
devicehigh=\DOS\DRV\Setver.exe
devicehigh=\DOS\DRV\Display.sys con=(ega,,1)
devicehigh=\DOS\DRV\Oakcdrom.sys /D:CD001
installhigh=\DOS\DRV\Mscdex.exe /D:CD001 /E /L:O /M:13
installhigh=\DOS\DRV\Mouse.com
shell=\Command.com \ /E:2016 /L:511 /U:255 /p
```

Пояснения к композиции всех использованных здесь команд имеются в главах 4 ("Конфигурационные команды") и 5 ("Драйверы"). Большинство драйверов взято из каталогов \Windows и \Windows\Command операционных систем Windows-95/98, за исключением драйверов MOUSE.COM (5.03-02) из поставки MS-DOS6.22 и OAKCDROM.SYS (5.09-01), который скопирован с загрузочной дискеты, создаваемой операционными системами Windows-95/98. Существует много других драйверов "мыши" и драйверов оптических дисков (GMOUSE.COM, VIDE-CDD.SYS, ECSCDIDE.SYS и др.), которые способны работать с разными моделями устройств и также могут быть применены здесь.

Порядок строк в файле CONFIG.SYS должен подчиняться следующему общему правилу: драйверы, обеспечивающие поддержку исполнения любой функции, должны быть загружены раньше, чем возникнет потребность в исполнении этой функции. Драйверы, обеспечивающие доступ к расширенной памяти (HIMEM.SYS и затем EMM386.EXE) должны быть загружены командами DEVICE раньше, чем этот доступ потребуется для исполнения команд DEVICEHIGH и INSTALLHIGH. Команды INSTALL и INSTALLHIGH, используемые для загрузки исполняемых драйверов, размещают после всех команд DEVICE и DEVICEHIGH, но перед строкой с командой SHELL. Строка загрузки драйвера SETVER.EXE должна предшествовать строке загрузки любого другого драйвера, которому требуется подмена версии DOS. Хотя таких драйверов в данном файле CONFIG.SYS нет, загрузка SETVER.EXE позволит в дальнейшем использовать программы из других версий DOS (PRINT.EXE, QBASIC.EXE, TREE.COM и др.).

Важную роль играет параметр NOAUTO команды DOS в 3-й строке: он исключает загрузку по умолчанию ряда драйверов (DBLBUFF.SYS, DRVSPACE.BIN, HIMEM.SYS, IFSHLP.SYS) и их поиск. В значительной степени благодаря параметру NOAUTO становится возможным использование MS-DOS7 как самостоятельной операционной системы.

Еще обратите внимание на путь к файлу COMMAND.COM, указываемый в последней строке: он сокращен до одного знака обратной косой черты. Этого достаточно, чтобы найти файл COMMAND.COM в корневом каталоге текущего диска, но недостаточно для полноценного определения пути к нему в переменной окружения %COMSPEC%. Конечно, можно указать букву диска и здесь. Примеры строк загрузки файла COMMAND.COM с указанием буквы диска для переменной %COMSPEC% приведены в разделах 4.26 и 6.04. Однако бывает неудобно заменять букву диска в нескольких местах каждый раз, когда данный комплект конфигурационных файлов приходится использовать для загрузки MS-DOS с другого диска. Поэтому здесь задание буквы диска отложено до исполнения последнего конфигурационного файла AUTOEXEC.BAT (9.01-02). Это позволяет обойтись корректированием буквы диска только в одном месте и открывает возможность автоматизации процесса определения того диска, с которого производится загрузка (такие примеры показаны в разделах 9.01-03 и 9.09-02).

9.01-02 Файл AUTOEXEC.BAT: простой вариант.

Поскольку функциональные возможности загрузчика IO.SYS ограничены, постольку некоторые конфигурационные операции нельзя представить командами файла CONFIG.SYS. Такие операции должны быть представлены в другом конфигурационном файле – AUTOEXEC.BAT. Он тоже является командным интерпретируемым файлом, но его миссия состоит в том, чтобы задействовать возможности более мощного командного интерпретатора – COMMAND.COM – для выполнения ряда заключительных конфигурационных операций.

Представленный здесь пример файла AUTOEXEC.BAT предполагает наличие на загрузочном диске структуры каталогов: каталога \TEMP для временных файлов и каталога \DOS с подкаталогами \DOS\OTH, \DOS\MS7, \DOS\VC4, \DOS\DRV. Такая структура подойдет и для дискет, и для жестких магнитных дисков. Но если Вы используете другую структуру каталогов, то все пути и ссылки должны быть соответствующим образом изменены. Также надо иметь ввиду, что файл написан для случая загрузки с диска C:. Если потребуется загружать компьютер с другого диска, то букву диска C: во второй строке файла необходимо будет заменить на букву того диска, с которого придется загружаться. Заметьте, что это единственное упоминание буквы загрузочного диска, которое надо корректировать. После этого во всех остальных местах нужная буква диска будет проставлена автоматически.

Ниже приведен текст предлагаемой простой версии файла AUTOEXEC.BAT.

```
@echo off
set dsk=C:
set comspec=%dsk%\Command.com
if not exist TEMP\nul %comspec% nul /f /c md TEMP
if exist TEMP\nul set Temp=%dsk%\TEMP
prompt $p$g
set dircmd= /A /O:GNE /P
path ;
path=%dsk%\DOS\VC4;%dsk%\DOS\OTH;%dsk%\DOS\MS7;%dsk%\
Mode.com con codepage prepare=((866) %dsk%\DOS\DRV\Ega3.cpi)
Mode.com con codepage select=866
Keyb.com ru,866,%dsk%\DOS\DRV\Keybrd3.sys
set VC=%dsk%\DOS\VC4
Vc.com /TSR /no2E /noswap
```

Первая строка файла отключает отображение строк на экране дисплея, как это обычно делается почти во всех batch-файлах. Во второй строке буква текущего диска записывается в переменную окружения %dsk%. При наборе нельзя оставлять пробелы в конце этой строки. Многочисленные ссылки на переменную %dsk% в последующих строках введут букву диска в состав всех спецификаций, где ее

наличие требуется. Именно благодаря ссылкам оказывается возможным указывать букву текущего диска только один раз (во второй строке). Третья строка вводит букву диска в значение переменной COMSPEC, которое не было сформировано должным образом при интерпретации файла CONFIG.SYS. С этого момента MS-DOS7 подготовлена к смене текущего диска.

Строки 4 и 5 занимаются каталогом TEMP для временных файлов. Сначала проверяется наличие этого каталога. Если он не существует, то предпринимается попытка его создать. Потом его наличие проверяется еще раз, и в случае успеха путь к нему записывается в переменную TEMP. Такая процедура гарантирует создание каталога для временных файлов на любом записываемом диске. С другой стороны, отсутствие значения у переменной TEMP после такой процедуры будет однозначно свидетельствовать о загрузке с незаписываемого диска.

Далее следует группа операций присвоения значения другим переменным окружения: %DIRCMD%, %PROMPT%, %PATH%. Показанные значения представляют собой примеры, которые надо корректировать согласно Вашим задачам и фактической структуре каталогов на Вашем диске. Предполагается, что значение переменной %PATH% (2.02-02) включает пути ко всем программам, которые будут вызваны в последующих строках: MODE.COM, KEYB.COM и VC.COM, а также все те пути, которые Вы захотите добавить.

Вызовы программ MODE.COM и KEYB.COM активизируют желаемую национальную кодовую страницу для знакогенератора и желаемую таблицу раскладки для клавиатуры. Если Вам нужна не русская кодовая страница 866, а какая-либо другая, то Вы вправе ее сменить, но при этом необходимо проверить по таблице A.02-2, содержат ли указанные здесь файлы данных (EGA3.CPI и KEYBRD3.SYS) требуемые национальные данные. Если нет, то эти файлы нужно будет заменить тоже. Разумеется, при выборе американской (437-й) кодовой страницы все строки с вызовами программ MODE.COM и KEYB.SYS становятся не нужны, потому что эта кодовая страница устанавливается по умолчанию.

Последние две строки файла AUTOEXEC.BAT служат для запуска файл-менеджера Volcov Commander (6.25). Другие файл-менеджеры, например, Norton Commander (NC.EXE) и Dos Navigator (DN.EXE), могут быть запущены на исполнение аналогичным образом. Если Вы не намерены использовать файл-менеджер и хотите управляться с MS-DOS7 из "голой" командной строки, то последние две строки файла AUTOEXEC.BAT будут просто не нужны.

9.01-03 Автоматическое определение буквы диска в ходе загрузки.

Было бы удобно иметь такой комплект конфигурационных файлов, которые сами определяли бы букву текущего диска по ходу загрузки. Такую адаптацию несложно реализовать, если Вы уже ассемблировали программу Reassign.com,

предложенную в разделе 9.06, или уже имеете какую-либо функционально эквивалентную программу. Нужно всего лишь заменить операцию задания конкретной буквы диска ("set dsk=C:") во 2-й строке файла AUTOEXEC.BAT (9.01-02) следующими двумя строками:

```
set dsk=33
\DOS\OTH\Reassign.com dsk
```

В любом случае путь перед именем вызываемой программы должен указывать на тот каталог, где эта программа реально находится. В результате исполнения этих двух командных строк буква текущего диска становится значением переменной окружения %dsk%, а потом она автоматически вводится во все остальные спецификации в ходе дальнейшего исполнения файла AUTOEXEC.BAT.

Автоматическое определение текущего диска возможно и без использования дополнительных программ, исключительно с помощью стандартных средств MS-DOS, но реализация получается не настолько простой и, помимо прочего, требующей доступа к записываемому диску. Поэтому приведенный ниже вариант файла AUTOEXEC.BAT нельзя использовать при загрузке MS-DOS7 с дисков CD-ROM. Предпочтительный вариант его использования – однократная установка на жесткий магнитный диск после форматирования.

```
@echo off
if exist ..\nul goto L19
prompt=@echo off$_Set dsk$q$N:$_goto L7
%comspec% /f /c $.bat > $.bat
type Autoexec.bat >> $.bat
for %%Z in ("del A" "ren $.bat A" "A") do %%Zutoexec.bat
:L7
set comspec=%dsk%\Command.com
set Temp=%dsk%\TEMP
if not exist %Temp%\nul md %Temp%
prompt $p$g
set dircmd= /A /O:GNE /P
path ;
path=%dsk%\DOS\VC4;%dsk%\DOS\OTH;%dsk%\DOS\MS7;%dsk%\
Mode.com con codepage prepare=((866) %dsk%\DOS\DRV\Ega3.cpi)
Mode.com con codepage select=866
Keyb.com ru,866,%dsk%\DOS\DRV\Keybrd3.sys
set VC=%dsk%\DOS\VC4
Vc.com /TSR /no2E /noswap
:L19
```

Строки с 8-й по 18-ю представленного варианта файла AUTOEXEC.BAT выполняют обычные операции, какие были описаны в разделе 9.01-02. Но строки 2

– 6 и расстановка меток специфичны для данного варианта. Для упрощения ориентации цифры в составе меток совпадают с номерами соответствующих строк.

Во второй строке представленного файла производится проверка того, имеется ли каталог, являющийся родительским по отношению к текущему. Если такой каталог имеется, то текущий каталог заведомо не является корневым, и тогда сразу происходит переход на конечную метку L19. Значит, вне корневого каталога данный batch-файл фактически не будет исполняться. Он будет исполняться только после того, как будет перемещен в корневой каталог диска. Только тогда проверка во 2-й строке позволит пройти к исполнению 3-й строки.

В третьей строке команда PROMPT (3.22) задает новую форму приглашения командной строки, которая выводится лишь однажды, при запуске командного интерпретатора в 4-й строке. Там командный интерпретатор формально исполнит пустой batch-файл $.BAT, только что созданный операционной системой при подготовке к перенаправлению, указанному правее в той же 4-й строке. Результат будет записан в тот же batch-файл $.BAT. Поскольку в нем первоначально ничего не было, туда запишется только приглашение командной строки, то самое, которое было задано в 3-й строке. Для определенности допустим, что текущим является диск D:, тогда содержимое batch-файла $.BAT после исполнения 4-й строки будет выглядеть так:

```
@echo off
Set dsk=D:
goto L7
```

Обратите внимание, что буква диска во второй строке файла $.BAT не была задана в команде PROMPT, это реальная буква текущего диска, вписанная операционной системой при формировании приглашения командной строки.

Команда TYPE в следующей 5-й строке файла AUTOEXEC.BAT считывает весь текст файла AUTOEXEC.BAT и через перенаправление вывода приписывает его снизу к показанным выше трем строкам файла $.BAT.

В 6-й строке файла AUTOEXEC.BAT цикл FOR исполняет подряд три операции, образуемые посредством последовательных подстановок трех значений локальной переменной %%Z. Первая подстановка дает команду

```
del Autoexec.bat
```

и файл AUTOEXEC.BAT перестает существовать. Затем в результате второй подстановки получается команда

```
ren $.bat Autoexec.bat
```

и прежний файл $.BAT оказывается переименован в AUTOEXEC.BAT. Последняя третья подстановка дает

```
Autoexec.bat
```

то есть команду исполнить новый файл AUTOEXEC.BAT начиная с его первой строки. Поскольку имени вызываемого на исполнение файла не предшествует команда CALL, постольку возврат к прежде исполнявшемуся и уже не существующему файлу не предусмотрен. Заметьте, что такую подмену исполняемого файла одноименным другим файлом нельзя осуществить, если не задавать все связанные с подменой операции в одной строке.

Новый файл AUTOEXEC.BAT начнется с тех самых показанных выше трех строк файла $.BAT. При их исполнении буква текущего диска станет значением переменной окружения %DSK%, а затем произойдет безусловный переход на метку L7. Тем самым группа операций само-модификации в строках 3 – 6 исходного текста файла AUTOEXEC.BAT будет навсегда обойдена. Начиная с метки L7 продолжится исполнение тривиальных операций файла AUTOEXEC.BAT, в ходе которых найденная буква текущего диска будет автоматически вписана во все спецификации, где она должна быть указана.

9.02 Командные файлы для отладчика DEBUG.EXE

9.02-01 Сохранение и восстановление boot-сектора

Boot-сектор – это первый сектор на каждом логическом диске. Он содержит BPB – блок параметров (A.03-4) диска, а также фрагмент исполняемого машинного кода и спецификации файлов, которым должно быть передано управление при загрузке компьютера. Запись boot-сектора стандартного формата выполняется при каждом форматировании диска программой FORMAT.COM (6.15), а также когда диск делается загрузочным с помощью программы SYS.COM (6.24). Но существуют программы, записывающие и использующие нестандартные boot-секторы, которые нельзя воссоздать обычными средствами MS-DOS7.

Сохранение в файле копии boot-сектора может потребоваться для обеспечения возможности его восстановления после вирусного заражения, повреждения сигналограммы или после перезаписи boot-сектора при организации альтернативной загрузки нескольких операционных систем (пример – в разделе 9.11-02). Командный интерпретатор COMMAND.COM прямого доступа к boot-сектору не обеспечивает, но для отладчика DEBUG.EXE (6.05) это – простая задача, решаемая исключительно с помощью его встроенных команд. Ниже приведен пример командного файла, который предназначен для исполнения отладчиком DEBUG.EXE с тем, чтобы скопировать содержимое boot-сектора с логического диска C:.

```
L CS:100 2 0 1
r BX
0000
r CX
200
n Bootsect.dat
w CS:100
q
```

Первая строка содержит команду считывания (6.05-10) boot-сектора логического диска C: и записи его в память начиная с адреса CS:100. Строки 2 – 5 вводят в регистры BX:CX длину (00000200h = 512 байт) получившегося в памяти блока данных, который надлежит записать в файл. Строка 6 задает имя для будущего файла. Строка 7 содержит команду записи (6.05-19) блока данных из памяти в файл. Последняя строка – это команда ("Q") выхода из сеанса работы с отладчиком DEBUG.EXE.

Представленный текст надо набрать с помощью программы текстового редактора (NOTEPAD или EDIT.COM) и сохранить результат в файле с подходящим именем (пусть он называется SAVEBOOT.SCR), созданном в текущем каталоге. Потом этот файл должен быть послан на исполнение отладчику DEBUG.EXE, но нужно иметь ввиду, что исполнение должно происходить в среде MS-DOS, а не в "окне DOS" операционной системы WINDOWS, потому что там прямой доступ к диску перекрыт. Послать командный файл на исполнение можно, например, так:

```
DEBUG.EXE < SAVEBOOT.SCR
```

После исполнения в текущем каталоге появится новый файл BOOTSECT.DAT длиной 512 байт, который содержит точную копию boot-сектора с логического диска C:. Аналогично можно скопировать в файл boot-сектор с любого другого существующего логического диска (6.05-10). В частности, для копирования boot-сектора с диска A:, первая строка в файле SAVEBOOT.SCR должна выглядеть так:

```
L CS:100 0 0 1
```

Конечно, перед обращением к сменному диску нужно убедиться в том, что этот сменный диск физически имеется в дисководе.

Обратная операция записи данных boot-сектора из файла на диск выполняется с помощью еще более простого командного файла:

```
n Bootsect.dat
L CS:100
w CS:100 2 0 1
q
```

Здесь первая строка объявляет имя файла, содержащего блок данных boot-сектора. Этот файл должен находиться в текущем каталоге. Вторая строка считывает данные из файла в память, а третья строка записывает данные из памяти в сектор диска, в данном случае в boot-сектор диска C:.

Представленный текст должен быть набран с помощью текстового редактора и сохранен под подходящим именем, например, RESTBOOT.SCR. Поскольку его исполнение предполагает прямой доступ к диску, постольку следует работать в среде MS-DOS (а не в "окне DOS" операционной системы Windows). Для исполнения командный файл RESTBOOT.SCR нужно послать отладчику DEBUG.EXE через перенаправление ввода, например, так:

```
DEBUG.EXE < RESTBOOT.SCR
```

С помощью описанных здесь командных файлов восстановление boot-сектора становится простым делом. Тем не менее следует предостеречь от попыток трансплантации boot-сектора на другой диск. Даже при трансплантации между дискетами одного формата возникают проблемы с уникальностью серийного номера и с меткой тома, потому что эта метка дублируется в корневом каталоге. Вообще boot-сектор уникален и не должен подходить ни к какому другому диску кроме того, с которого он был считан.

9.02-02 Сохранение главной загрузочной записи (MBR) в файле

Каждый раз при обычном включении Ваш компьютер загружает установленную операционную систему с жесткого магнитного диска. Эта привычная процедура включает несколько этапов, и одним из важнейших этапов является обращение к главной загрузочной записи (MBR), которая указывает на расположение загрузочного раздела диска и направляет процесс загрузки дальше. Как любая другая запись на диске, главная загрузочная запись подвержена естественной деградации, а также может быть случайно повреждена или намеренно испорчена компьютерными вирусами. В таких случаях приходится загружать компьютер со сменного диска и восстанавливать главную загрузочную запись.

Для восстановления главной загрузочной записи операционные системы Windows-95/98/ME предоставляют программу FDISK.EXE (6.13), но эта программа не восстанавливает таблицу разделов диска, которая находится вместе с главной загрузочной записью в том же физическом секторе диска (A.13-5). Особые виды главной загрузочной записи, которые используются другими операционными системами, программой DDO (Dynamic Drive Overlay) и boot-менеджерами, также не могут быть восстановлены программой FDISK.EXE.

Между тем существует простое и универсальное решение: сохранить файл с копией главной загрузочной записи на сменном носителе – дискете, флэш-карте

или оптическом диске. Тогда Вы всегда сможете восстановить главную загрузочную запись вместе с таблицей разделов по данным из сохраненного файла.

MS-DOS7 не предоставляет специальных средств копирования главной загрузочной записи, но зато предоставляет отладчик DEBUG.EXE (6.05), с помощью которого можно написать и тут же исполнить последовательность операций, в том числе и такую, которая копирует главную загрузочную запись в файл. Простой вариант текста командного файла с такой последовательностью операций выглядит так:

```
a 100
mov         DX,0080      ;запрос головки 00h дисковода 80h
mov         CX,0001      ;запрос сектора 01h на цилиндре 00h
mov         BX,0200      ;загружать начиная со смещения 0200h
mov         AX,0201      ;запрос 1 сектора и функции AH=02h
int         13           ;считывание MBR в память (8.01-46)
mov         [01F6],AH    ;код завершения - в ячейку памяти
ret                      ;кончить исполнение команды "g =100"
org         130          ;начать набор заново с ячейки 130
mov         AL,[01F6]    ;код завершения - в регистр AL
mov         AH,4C        ;4C - операция завершения программы
int         21           ;исполнение операции DOS (8.02-55)

g =100
n Mbr.dat
r CX
0200
r BX
0000
w CS:0200
d 01F6,L1
g =130
```

Представленный здесь текст надо набрать с помощью программы редактирования, как рекомендовано во вводной статье к главе 9. Комментарии справа от знака точки с запятой не исполняются отладчиком DEBUG.EXE, и их можно не набирать. Обратите внимание на наличие пустой строки между командами "INT 21" и "G =100": она играет важную роль (7.01-04) и должна там быть. Закончив набор, сохраните текст в файле, например, под именем READ_MBR.SCR.

Если Ваш компьютер загружается не с логического диска C:, а с диска D:, E: или другого логического диска, то нужно проверить, на каком физическом

дисководе находится загрузочный логический диск, например, посредством команды

```
FDISK.EXE /status
```

Из выведенной на экран таблицы Вы определите номер загрузочного физического дисковода, и если это будет не дисковод номер 1, то запрос во второй строке файла READ_MBR.SCR нужно будет исправить: для случая загрузки с дисковода номер 2 надо будет указать код дисковода 81h, для случая загрузки с дисковода номер 3 – код 82h, и так далее.

Теперь уместно пояснить, как действует файл READ_MBR.SCR. Его первая строка "A 100" переключает отладчик DEBUG.EXE в режим ассемблирования с размещением кодов начиная с адреса CS:0100h. Пока сохраняется режим ассемблирования, отладчик позволяет вводить комментарии после знака точки с запятой, и роль команд в строках 2 – 12 ясна из этих комментариев. Более детальные сведения по каждой ассемблерной команде приведены в главе 7. Отладчик DEBUG.EXE переводит команды в две последовательности машинных кодов: одна записывается в память начиная с ячейки 100h, а вторая, объявленная командой ORG в строке 9, записывается начиная с ячейки 130h. Запись кодов в память продолжается до тех пор, пока отладчик не встретит пустую строку 13. Она воспринимается как приказ выйти из режима ассемблирования. Команды в последующих строках не дополняют последовательность машинных кодов, а исполняются сразу.

Команда "G =100" (6.05-07) в 14-й строке запускает на исполнение первую последовательность машинных кодов с адреса CS:0100h. В ходе исполнения главная загрузочная запись копируется с диска в память и размещается там начиная с ячейки 0200h. Код завершения этой операции временно сохраняется в произвольно выбранной свободной ячейке 01F6h. Исполнение первой последовательности машинных кодов прекращает команда RET (7.03-73) из 8-й строки, которая возвратит управление обратно отладчику DEBUG.EXE.

Отладчик продолжит исполнение с 15-й строки, с команды "N" (6.05-12). Она подготавливает имя для файла, который должен быть создан. Вы можете изменить это имя или указать перед ним путь, если нужно создать файл не в текущем каталоге. В строках 16 – 19 длина создаваемого файла (00000200h) вводится в регистры BX и CX. Команда "W" (6.05-19) в 20-й строке считывает данные из памяти, начиная с адреса CS:0200h, и записывает их файл, имя и длина которого заданы заранее. Предпоследняя команда "D 01F6,L1" (6.05-04) из 21-й строки покажет на экране код завершения операции считывания с диска, сохраненный в ячейке 01F6h.

Последняя команда "G =130" в 22-й строке запустит на исполнение вторую подготовленную последовательность машинных кодов. В ходе ее исполнения код завершения, сохраненный в ячейке 01F6h, считывается в регистр AL, а потом

происходит вызов обработчика прерывания INT 21\AH=4Ch (8.02-55), прекращающего сеанс работы с отладчиком DEBUG.EXE. При этом код из регистра AL становится возвращаемым кодом уровня ошибки, который потом можно проверить по условию "if errorlevel" (3.15-03).

Перед исполнением командного файла READ_MBR.SCR нужно убедиться в том, что он находится в текущем каталоге на доступном для записи диске. Если главная загрузочная запись будет сохранена в файле MBR.DAT, то файла с таким именем в каталоге назначения не должно быть, иначе он будет перезаписан без предупреждения. На этом приготовления заканчиваются, и Вы можете послать командный файл на исполнение, например, так:

```
DEBUG.EXE < READ_MBR.SCR
```

При исполнении выводится сообщение "Program terminated normally" (= программа завершилась успешно), но оно означает только то, что отладчик снова взял управление на себя после исполнения первой последовательности подготовленных машинных кодов. Исход попытки считывания с диска выражается кодом завершения – двузначным шестнадцатеричным числом в предпоследней выведенной на экран строке. Ненулевое значение кода означает неудачу попытки доступа к диску. При таком исходе файл MBR.DAT будет создан, но он будет содержать только "мусор". По условию "If errorlevel 1" (3.15-03) его следует автоматически удалить. Расшифровка ненулевых значений кода завершения по таблице A.06-1 поможет выяснить причину неудачи.

Значение 00h кода завершения служит свидетельством того, что считывание главной загрузочной записи с диска произведено успешно. В таком случае новый файл с предлагаемым именем MBR.DAT будет содержать копию главной загрузочной записи и таблицы разделов диска. Образец дампа файла MBR.DAT приведен на рис. 12 (в разделе A.13-5).

9.02-03 Восстановление главной загрузочной записи (MBR)

Возникновение сбоев в главной загрузочной записи – событие относительно редкое, но от него не застрахован никто. Однажды такое может случиться и с Вами. Тогда надо проверить и исключить несколько других гипотез: неверные установки параметров BIOS, потерю данных в CMOS-памяти, повреждение boot-сектора и т.д. Самый убедительный эксперимент – повторное считывание главной загрузочной записи в файл, как было показано в разделе 9.02-02, и сравнение полученной копии с той, которую Вы сделали заблаговременно и сохранили в другом файле. Сравнение можно выполнить с помощью программы FC.EXE (6.12). Если будет выявлено различие, то главную загрузочную запись надо будет перезаписать или восстановить.

MS-DOS не предоставляет специальных средств восстановления главной загрузочной записи по сохраненной в файле копии, но предоставляет возможность написать командный файл, который заставит отладчика DEBUG.EXE выполнить требуемую работу. Допустим, что этот файл называется WriteMBR.SCR, что файл с копией первоначальной главной загрузочной записи называется MBR.DAT, и что оба этих файла находятся в текущем каталоге. Тогда текст командного файла WriteMBR.SCR может выглядеть, например, так:

```
a 100
mov         DX,0080      ;запрос головки 00h дисковода 80h
mov         CX,0001      ;запрос сектора 01h на цилиндре 00h
mov         BX,0200      ;брать данные от смещения 0200h
mov         AX,0301      ;запрос 1 сектора и функции AH=03h
int         13           ;запись MBR на диск (8.01-47)
mov         [01F6],AH    ;код завершения - в ячейку памяти
ret                      ;кончить исполнение команды "g"

n MBR.DAT
L CS:0200
g =0100
d 01F6,L1
q
```

Команда "A 100" в первой строке представленного командного файла переводит отладчик DEBUG.EXE в режим ассемблирования, так что команды из строк 2 – 8 не исполняются сразу, а переводятся в машинные коды и записываются в память, начиная с адреса CS:0100. Пока отладчик DEBUG.EXE работает в режиме ассемблирования, он позволяет снабжать каждую строку комментариями, так что миссия команд в строках 2 – 8 вполне ясна из этих комментариев. Встретив пустую строку 9, отладчик DEBUG.EXE выходит из режима ассемблирования, принимает командой "N" (6.05-12) в 10-й строке имя того файла, который надо загрузить, и командой "L" (6.05-10) в 11-й строке загружает этот файл в память начиная с адреса CS:0200.

Команда "G" (6.05-07) в 12-й строке запускает на исполнение подготовленную последовательность машинных кодов, начиная с адреса CS:0100h. Исполнение будет продолжаться до встречи с командой RET (7.03-73) в 8-й строке, которая вернет управление отладчику DEBUG.EXE. Тогда исполнение файла WriteMBR.SCR продолжится со строки, следующей за командой "G", то есть с команды "D 01F6,L1" в 13-й строке; она выведет на экран код завершения, оставленный системой BIOS после вызова прерывания INT13. Последняя 14-я строка с командой "Q" закроет сеанс работы отладчика DEBUG.EXE.

Конечно, если Ваш компьютер загружается с физического дисковода, которому присвоен не первый номер, то код дисковода "80h" во второй строке должен быть изменен точно так же, как описано в разделе 9.02-02. Когда Вы уверены, что код дисковода указан правильно, что главную загрузочную запись действительно необходимо восстановить, и что все необходимые файлы должным образом подготовлены в текущем каталоге, тогда командный файл WriteMBR.SCR можно послать на исполнение, например, так:

```
DEBUG.EXE <  WriteMBR.SCR
```

После исполнения на экране отдельной строкой будет показан двухзначный шестнадцатеричный код завершения. Значения кода завершения несложно расшифровать по таблице A.06-1. Значение "00h" будет свидетельствовать о том, что восстановление главной загрузочной записи выполнено успешно.

Примечание 1: проводить эксперименты с файлом WriteMBR.SCR над используемыми дисководами нельзя, потому что из-за случайной ошибки можно потерять доступ к имеющимся логическим дискам. Экспериментировать можно только если в дисководе заведомо не записано ничего такого, что стоило бы сохранять.

Примечание 2: при исполнении файла WriteMBR.SCR необходим прямой доступ к диску. Поэтому главную загрузочную запись нельзя восстанавливать из "окна DOS" операционной системы WINDOWS, это надо делать в среде MS-DOS7.

9.03 Примеры batch-файлов

Интерпретатор COMMAND.COM принимает обычные командные файлы через перенаправление ввода так же, как это делает отладчик DEBUG.EXE (9.02). Однако batch-файлы представляют особый класс командных файлов, которые интерпретатор COMMAND.COM распознает и принимает к исполнению прямо из командной строки, без перенаправления. Более того, в batch-файлах интерпретатор COMMAND.COM исполняет несколько важных команд (3.02, 3.14, 3.21, 3.27), не принимаемых к исполнению из обычных командных файлов и из командной строки. Из-за этих причин для исполнения интерпретатором COMMAND.COM сейчас предназначают только такие командные файлы, которые относятся к классу batch-файлов.

Наиболее простым batch-файлом в этой книжке является файл AUTOEXEC.BAT, описанный в разделе 9.01-02. Поэтому здесь далее приведены чуть менее простые примеры batch-файлов. Они показывают приемы batch-программирования, которые могут оказаться полезными для решения многих других задач помимо тех, которые представлены здесь.

9.03-01 Batch-файл Arc.bat для архивирования.

В компьютерном жаргоне под архивированием понимают сжатие группы файлов в объединенный файл-архив. Такой смысл "прилип" к этому слову давно, когда компьютеры были очень ненадежны, и файлы приходилось часто сохранять (точнее, save = спасать!) в виде объединенного потока на магнитной ленте. С того времени многое изменилось, но интерес к программам архивирования не иссяк. Сжатие объема файлов сейчас актуально из-за ограниченной скорости передачи данных по сетям. Помимо того, представление нескольких файлов одним архивом ускоряет их копирование и дефрагментацию дисков, уменьшает потери дискового пространства в кластерах, а также оказывается очень удобным при поставке больших программных комплектов.

Из множества известных алгоритмов архивирования только два – ZIP и RAR – заслужили широкое признание рядовых пользователей. Алгоритм ZIP разработан Филиппом Кацем в начале 1990-х годов и с тех пор воспроизведен в программах многих других авторов. Рекордной степени сжатия он не обеспечивает, но имеет два других достоинства - скорость и совместимость. Возможность распаковки ZIP-архивов не определяется версией той программы, которая их сформировала. Программы формирования и распаковки ZIP-архивов можно свободно скачать из сети Интернет, например, с сайта http://comp.site3k.net/comp/pkzip.html .

Алгоритм RAR, разработанный Евгением Рошалем в середине 1990-х годов, отличается повышенной степенью сжатия и возможностью исправления частично поврежденных архивов при условии заблаговременного введения избыточности. Но нужно иметь ввиду, что новые версии программы WINRAR формируют архивы, которые нельзя распаковать с помощью прежних версий. Эта несовместимость может подвести Вас в ответственный момент или поставить в неудобное положение перед Вашими адресатами. Чтобы избежать таких неожиданностей, RAR-архивы лучше формировать с помощью не новой, но достаточно хорошей программы RAR.EXE версии 2.50 (1999-го года), которую можно свободно скачать, например, с сайта http://dosprogram.narod.ru/arc/index.html .

Для удобства пользования архивами файл-менеджер Volcov Commander раскрывает их по нажатию кнопки "мыши" и позволяет обращаться с их содержимым как с файлами в обычных каталогах (подробнее – в разделе 6.25-03). Создавать архивы через меню файл-менеджера также намного удобнее, чем из командной строки. Тем не менее посредничество файл-менеджера от ошибок не избавляет, а разбираться с их причинами иногда даже становится труднее. В связи с этим возникла идея написать простой командный файл для проверки параметров, которые файл-менеджер передает вызываемой программе архивирования. Идея была воплощена в виде batch-файла, то есть такого текстового файла, каждая строка которого является командной строкой для интерпретатора

COMMAND.COM. Проверочный batch-файл ARC.BAT предотвращал ошибочные вызовы, а выводимые им сообщения помогали быстро исправить ситуацию.

С годами файл ARC.BAT рос и обзаводился новыми проверками, так что теперь их число достигло семи. Когда встал вопрос о том, какой файл предложить в качестве несложного и вместе с тем полезного примера batch-файла, то самым подходящим оказался именно ARC.BAT.

Принцип действия файла ARC.BAT исходит из предположения, что в активной панели файл-менеджера Volcov Commander пользователь набирает правой кнопкой "мыши" группу файлов и каталогов, которые надо поместить в новый архив, а левой кнопкой "мыши" в той же панели выделяет один файл или каталог, имя которого будет присвоено создаваемому файлу архива. Пример такого выделения группы файлов показан на рис. 5 (в разделе 6.25-01). После этого достаточно щелкнуть мышкой по пункту меню, вызывающему файл ARC.BAT – и архив готов. Если в этот момент открыта только одна панель файл-менеджера, то архив будет помещен в тот же каталог. Если открыты обе панели, то каталог назначения укажет противоположная (неактивная) панель.

В ходе изложенной процедуры все состояния панелей и имена выделенных файлов передаются посредством макрокоманд файл-менеджера Volcov Commander, знаки вызова которых объяснены в разделе 6.25-02. Эти знаки указывают в строке вызова batch-файла ARC.BAT в определенном порядке, чтобы каждый возвращаемый элемент данных был отражен отдельным формальным параметром batch-файла. В частности, для создания архивов типа RAR строка вызова batch-файла ARC.BAT в файле меню VC.MNU должна иметь следующий вид:

```
@Arc.bat  RAR  !:  !~\  !~  !~@  %:  %~\
```

При исполнении этой строки файл-менеджером Volcov Commander каждый знак вызова макрокоманды будет заменен тем элементом данных, который эта макрокоманда возвратит. Потом строка со всеми выполненными заменами будет передана для исполнения командному интерпретатору COMMAND.COM, который установит должное соответствие между формальными параметрами batch-файла и имеющимися в командной строке элементами данных. В тексте batch-файла места подстановки значений формальных параметров обозначаются цифрами от 0 до 9 с предшествующим знаком процента (2.03-03), причем цифра – это порядковый номер элемента данных в командной строке. При показанном порядке элементов данных формальные параметры получат вот какие назначения:

%0 – имя исполняемого файла (ARC.BAT)
%1 – RAR (или ZIP): тип создаваемого архива
%2 – буква диска из активной панели
%3 – путь к каталогу, открытому в активной панели
%4 – имя файла, выделенного левой кнопкой "мыши"

%5 – файл-список имен, выделенных правой кнопкой
%6 – буква диска из неактивной панели
%7 – путь к каталогу, открытому в неактивной панели

Перечисленные сведения, полученные от файл-менеджера, дополняются теми, которые в файле ARC.BAT будут запрошены у операционной системы. Все они учитываются при проведении проверок с целью выявления обстоятельств, препятствующих успешному осуществлению архивирования. В зависимости от результатов проверок либо будет вызвана программа архивирования, либо на экран будет выведено понятное сообщение об ошибке. Ниже приведен полный текст batch-файла ARC.BAT, содержащий командные строки для выполнения всех необходимых проверок.

```
@echo off
set V1=02
if %1"==ZIP" set V1=Pkzip.exe
if %1"==RAR" set V1=Rar.exe
if %6"==" set V1=02
if %V1%==02 echo Parameters are invalid or not defined!
if %V1%==02 goto END
set V2=08
for %%Z in (%path%) do if exist %%Z\%V1% set V2=%%Z\%V1%
rem  ================== Line 10 ==================
if %V2%==08 echo The %V1% archiver hasn't been found!
if %V2%==08 goto END
set V3=13
for %%Z in (%path%) do if exist %%Z\Find.exe set V3=%%Z\Find.exe
if %V3%==13 echo The Find.exe utility hasn't been found!
if %V3%==13 goto END
if %7"==" echo Archive in inactive panel must be closed!
if %7"==" goto END
if %4"==.." echo Name for the archive isn't chosen!
rem  ================== Line 20 ==================
if %4"==.." goto END
%V3% /C /I /V ".%1" %5 | %V3% ": 0" > nul
if not errorlevel 1 echo Chosen file(s) - already %1-archive(s)!
if not errorlevel 1 goto END
%V3% /I "%4.%1" %5 > nul
if not errorlevel 1 if %2%3"==%6%7" echo Conflicting filenames!
if not errorlevel 1 if %2%3"==%6%7" goto END
ctty nul
%comspec% /f /c copy %V3% %6%7%4.%1 /Y | %V3% "1 f"
rem  ================== Line 30 ==================
ctty con
```

```
if errorlevel 1 echo Non-writable target disk or overwrite denied
if errorlevel 1 goto END
del %6%7%4.%1
if %1==RAR %V2% a -s- -rr -ems- -w%5\.. %6%7%4.%1 @%5
if %1==RAR if errorlevel 1 goto END
for %%Z in (1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1) do echo.
if %1==ZIP %V2% %6%7%4.%1 -ex -P -wHS -jhsr @%5
if %1==ZIP if errorlevel 1 goto END
rem  ================== Line 40 ==================
echo Archive %4.%1 is written into the %6%7 directory.
:END
```

По своей структуре файл ARC.BAT представляет собой просто последовательность проверок. Циклов и подпрограмм в нем нет. Все переходы выполняются только в случае неисполнения условий проверок и только на единственную конечную метку END. Чтобы было легче искать строки по номерам, каждая десятая строка содержит комментарий с объявлением ее номера.

Первая строка файла ARC.BAT выключает ECHO-флаг, чтобы специально выводимые сообщения не терялись в множестве повторов исполняемых строк. В строках 2 – 7 проверяется значение 1-го формального параметра и наличие значения у 6-го формального параметра. Если эти условия не выполнены, то выводится сообщение о том, что параметры не определены или неверно заданы, и на том исполнение завершается. Но если условия выполнены, то значением переменной V1 становится имя той программы архивирования, которую потом предстоит вызывать.

Следующая проверка в строке 9 выясняет, имеется ли нужная программа архивирования в каком-либо из тех каталогов, пути к которым указаны в значении переменной PATH (2.02-02). Подготовка должного значения переменной PATH входит в число общих условий исполнения batch-файлов, оговоренных во вводной статье к главе 9. Примеры присвоения значений переменной PATH даны во всех комплектах конфигурационных файлов, представленных в этой книге. Однако это не снимает с пользователя ответственности за то, чтобы требуемая программа архивирования все же могла бы быть найдена вдоль хотя бы одного из указанных там путей. Если ее не удастся найти, исполнение завершится сообщением об ошибке. Если же программа архивирования найдена, то ее имя вместе с предшествующим полным путем станет значением переменной V2.

Далее в 14-й строке аналогичным образом происходит поиск программы FIND.EXE (6.14), которая необходима для проведения нескольких последующих проверок. Если программа FIND.EXE будет найдена, то ее имя вместе с предшествующим полным путем станет значением переменной V3.

В строках 17 и 18 файла ARC.BAT проверяется наличие значения у седьмого формального параметра, который должен задавать путь к каталогу, открытому в неактивной панели файл-менеджера. Значение 7-го параметра бывает не определено, когда в неактивной панели открыт не каталог, а другой архив. В таком случае сообщение, выводимое файлом ARC.BAT, укажет на то, что архив в неактивной панели следует закрыть.

В 19-й строке проверяется то имя, которое пользователь должен был выделить левой кнопкой "мыши", чтобы оно было присвоено создаваемому архиву. Нередко выделение остается на строке с двумя точками, которая в панелях файл-менеджера Volcov Commander обозначает вышестоящий каталог. В таком случае на экран будет выведено сообщение о неправильном выборе имени будущего архива.

В 22-й строке файла ARC.BAT выполняется проверка записей в том файле-списке, посредством которого файл-менеджер должен передать программе архивирования имена архивируемых файлов. Напомним, что значением переменной V3, дважды подставляемым в 22-й строке, является имя файла FIND.EXE с предшествующим полным путем к нему. При первом вызове программа FIND.EXE считает в списке имена файлов, не являющиеся архивами того же вида, какой предстоит создать. Дело в том, что наличие файла-архива в составе группы архивируемых файлов допустимо, но повторное сжатие только архивов бессмысленно: степень сжатия от этого не увеличивается, а надежность сохранения данных снижается. Результат подсчета передается через промежуточное перенаправление, и второй вызов программы FIND.EXE выясняет, не равен ли результат подсчета нулю. Если выяснится, что других файлов в подготовленной группе нет, то исполнение закончится выведением на экран сообщения о том, что выделенные пользователем файлы уже являются архивами желаемого типа.

Проверка в строке 25 выясняет, не содержится ли файл архива с избранным пользователем именем и суффиксом в составе группы файлов, подлежащих архивированию. Если да, то новый архив должен создаваться обязательно не в том же каталоге, иначе одноименный исходный файл будет перезаписан раньше, чем его содержимое будет включено в состав создаваемого архива. В случае выявления такой ситуации на экран выводится сообщение о конфликте имен в текущем каталоге.

В строке 29 предпринимается попытка записать файл с именем создаваемого архива в тот каталог, куда предстоит поместить создаваемый архив. Обращение к каталогу назначения производится со всеми мерами предосторожности, какие должны быть приняты при обращении к недоступному диску: команда CTTY NUL в 28-й строке предотвратит выдачу панических сообщений, а вызов командного интерпретатора с параметром /f гарантирует безостановочное продолжение исполнения. Напомним, что значение переменной %COMSPEC%, подставляемое в 29-й строке, представляет имя командного интерпретатора с предшествующим

полным путем к нему, причем оно присваивается переменной %COMSPEC% автоматически при первом запуске командного интерпретатора (6.04).

Попытка записи в 29-й строке может завершиться неудачно, когда в каталоге назначения уже находится одноименный файл, имеющий атрибуты HRS (6.01), а также когда каталог назначения находится на незаписываемом или на защищенном от записи диске. Исход попытки записи отображается сообщением, которое команда COPY направит в канал STDOUT. Но в 29-й строке оно перенаправлено программе FIND.EXE. Если она зафиксирует неудачу, то исполнение завершится вследствие невозможности создания архива с заданным именем в указанном каталоге. Если же попытка записи окажется успешной, то записанный файл сразу же будет удален командой DEL (3.09) в 34-й строке. Тем самым будет устранено последнее препятствие, которое могло бы помешать исполнению миссии программы архивирования.

При вызове каждой программы архивирования нужно учитывать, как эта программа выводит на экран свои сообщения. Архиватор RAR.EXE формирует свое экранное поле, не пользуясь каналом STDOUT, и экран потом надо очистить. Но если это сделать командой CLS, то последующее заключительное сообщение будет выведено в верхней строке экрана и сразу скроется под панелями файл-менеджера. Поэтому после вызова программы RAR.EXE в 35-й строке очистку экрана приходится выполнять смещением курсора на 20 строк вниз командой FOR в 37-й строке. Помимо того, панели файл-менеджера должны быть развернуты не на полную длину. Их длина регулируется "мышью" или клавишами стрелок вверх-вниз в то время, пока удерживаются нажатыми клавиши ALT-F11 или ALT-F12. Установленный размер панелей надо зафиксировать в файле VC.INI нажатием клавишной комбинации SHIFT-F9.

Архиватор PKZIP выводит сообщения через канал STDOUT, и потому его вызов производится в 38-й строке, то есть после очистки экрана. Благодаря всем выполненным проверкам неудачный исход операций архивирования крайне маловероятен, но вообще исключать его нельзя. На этот случай предусмотрены переходы на конечную метку END со строк 36 и 39, и тогда сообщение об ошибке, выводимое программой архивирования, останется на экране после окончания исполнения команд файла ARC.BAT. А при удачном завершении миссии любого архиватора команда ECHO в 41-й строке выведет на экран финальное сообщение о том, как архив назван и в каком каталоге он создан.

Готовый batch-файл ARC.BAT надо поместить в каталог, путь к которому указан в значении переменной PATH (2.02-02), причем желательно в тот же каталог, где находятся файлы меню файл-менеджера Volcov Commander. Пример текста меню VC.MNU со строками вызова batch-файла ARC.BAT показан в разделе 6.25-02.

9.03-02 Batch-файл для исследования дисков.

Когда нужно обследовать незнакомый компьютер, то первый вопрос относится к дискам: сколько их и доступны ли они. Существуют программы MSD.EXE (из комплекта MS-DOS6.22) и NDIAGS.EXE (из пакета программ Norton Utilities), которые призваны решать такие проблемы. Обе упомянутые программы совместимы с MS-DOS7, но обе они не сообщают статус носителя в дисководе - вставлен ли сменный диск, форматирован ли, доступен ли для записи и т.д.

Предлагаемый ниже batch-файл – назовем его DISK.BAT – выдает краткий, но исчерпывающий отчет о статусе дисков в дисководах. Основная идея состоит в том, чтобы проверить возможность считывания путем считывания метки тома, а затем проверить доступность для записи путем записи той же метки обратно. Такая процедура безопасна, потому что и успех, и неудача операции записи не влекут за собой изменения содержания диска.

С другой стороны, поставленная задача дает повод показать в файле DISK.BAT несколько нетривиальных приемов batch-программирования, в частности:

– использование подпрограмм в составе batch-файла;
– обеспечение безостановочного тестирования недоступных дисков;
– предотвращение выведения нежелательных сообщений;
– формирование временных командных файлов;
– запись сообщений из канала STDOUT в переменную окружения.

Для упрощения поиска строк в файле DISK.BAT каждая десятая строка содержит комментарий с объявлением ее номера. Кроме того, метки названы по номерам соответствующих строк; например, метка L28 отмечает 28-ю строку, где кончается основная часть программы и начинается одна из подпрограмм.

Ниже следует текст batch-файла DISK.BAT.

```
@echo off
if %1"==&" if not %2"==" goto L%2
if %Path%"==" %0 & 79 4 PATH
if %Temp%"==" %0 & 79 4 TEMP
Call %0 & 28 A Attrib D Debug F Find L Label
if VA"==" goto L87
set V1=%Path%
set Path=%1
set V2=%Path%
rem  ================== Line 10 ==================
set Path=%V1%
set V1=
Call %0 & 39 A B C D E F G H I J K L M N O P Q R S
```

```
if %V1%"==" if not %1"==" %0 & 79 5
if %1"==" set V1=A C D E F G O R
ctty nul
%VA% -H -R -S %Temp%\tmp.*
echo e 100 'call %%1 & 46' 20 > %Temp%\tmp.scr
echo w >> %Temp%\tmp.scr
rem  ================== Line 20 ==================
echo q >> %Temp%\tmp.scr
set Dircmd=
for %%Z in (%V1%) do call %0 & 53 %%Z: %1
del %Temp%\tmp.*
ctty con
for %%Z in (A D F L 1 2) do set V%%Z=
goto L87
:L28
shift
rem  ================== Line 30 ==================
shift
set VL=
for %%Z in (%path%) do if exist %%Z\%2.exe set VL=%%Z\%2.exe
if %VL%"==" echo Error: the %2.exe utility hasn't been found!
if %VL%"==" set VA=
if not %VL%"==" set V%1=%VL%
if not %4"==" goto L28
goto L87
:L39
rem  ================== Line 40 ==================
shift
if %V2%"==%2" set V1=%2
if %V2%"==%2:" set V1=%2
if not %3"==" goto L39
goto L87
:L46
set V1=%6
if not %7"==" set V1=%6 %7
if not %8"==" set V1=%6 %7 %8
rem  ================== Line 50 ==================
if %5"==has" set V1=NO NAME
goto L87
:L53
%comspec% /f /c Dir /-p %3\nul > %Temp%\tmp.txt
%VF% "Volume in d" < %Temp%\tmp.txt > %Temp%\tmp.bat
if errorlevel 1 %0 & 79 6 %3 %4
%VF% "Directory of " < %Temp%\tmp.txt
```

```
if errorlevel 1 %0 & 79 7 %3
%VF% "0 bytes free" < %Temp%\tmp.txt
rem  ================== Line 60 ==================
if not errorlevel 1 %0 & 79 8 %3
%VD% %Temp%\tmp.bat < %Temp%\tmp.scr
call %Temp%\tmp.bat %0
set V2=if errorlevel 20 del %Temp%\tmp.bat
%comspec% /f /c for %%Z in ("%VL% %3%V1%" "%V2%") do %%Z
%VF% "Volume Seria" < %Temp%\tmp.txt
if errorlevel 1 if not exist %Temp%\tmp.bat %0 & 79 9 %3
if errorlevel 1 if exist %Temp%\tmp.bat %0 & 73 1 %3
if not errorlevel 1 if not exist %Temp%\tmp.bat %0 & 73 2 %3
rem  ================== Line 70 ==================
if not errorlevel 1 if exist %Temp%\tmp.bat %0 & 73 3 %3
goto L87
:L73
if %3"==1" if %3"==A:" echo Disk %4 (%V1%) is writable > con
if %3"==1" if not %3"==A:" echo Disk %4 (%V1%) is a RAM-disk > con
if %3"==2" echo Disk %4 (%V1%) is write-protected > con
if %3"==3" echo Disk %4 (%V1%) is writable > con
Dir /a:ARD /-p /v %4\ | %Dsk%\DOS\MS7\Find.exe "otal d" > con
:L79
rem  ================== Line 80 ==================
if %3"==4" echo The %4 variable is not defined
if %3"==5" echo Wrong parameter, it must be a letter-name or none
if %3"==6" if not %5"==" echo Letter %4 doesn't refer to a disk > con
if %3"==7" echo Disk %4 has no media inside > con
if %3"==8" echo Disk %4 is probably a CD-ROM (no free space) > con
if %3"==9" echo Disk %4 is not formatted > con
:L87
```

Вторая строка в файле DISK.BAT проверяет значение 1-го параметра. Если это значение – амперсанд, то выполняется переход, а если нет, то будет продолжено исполнение основной части программы. Обратите внимание еще на то, что адрес перехода ("goto L%2") не фиксирован, а зависит от второго параметра. Такое построение позволяет включить несколько подпрограмм в состав основного batch-файла, иначе пришлось бы использовать отдельные файлы для каждой подпрограммы.

Команды в строках 3 – 22 выполняют подготовительные операции. Сначала проверяется наличие значений у переменных PATH и TEMP: первая из них должна указывать путь к файлам MS-DOS7, а вторая – к каталогу для временных файлов.

Если хотя бы одна из этих переменных не определена, происходит переход на метку L79, где выводится соответствующее сообщение об ошибке, и на том исполнение программы завершится. Системные пути должны быть заданы заранее в конфигурационных файлах, и такие примеры даны во всех комплектах конфигурационных файлов, представленных в этой книге.

Следующая проверка в строке 5 вызывает подпрограмму, размещенную начиная со строки 28. Она должна записать в переменные окружения пути к каждому файлу, который потребуется при исполнении файла DISK.BAT. Имена этих файлов перечислены там же, в строке 5. Имена будут переданы подпрограмме через ее формальные параметры. Основная операция извлечения нужного пути из значения переменной PATH выполняется в 33-й строке. В случае неудачи из 34-й строки выводится сообщение об ошибке. Пока перечень формальных параметров не опустеет, из 37-й строки будет происходить переход на начало подпрограммы, на метку L28, чтобы найти путь к следующему файлу. В результате пути к файлам Attrib.exe, Debug.exe, Find.exe, Label.exe станут значениями переменных окружения VA, VD, VF и VL соответственно.

После возврата из подпрограммы продолжится исполнение операций в строках 7 – 11, где буква диска, которая может быть задана пользователем в командной строке, преобразуется в заглавную. Однако вместо буквы диска по ошибке может быть указан иной знак. На этот случай предусмотрен вызов в 13-й строке еще одной подпрограммы, которая размещена в строках 39 – 45 и проверяет знак на принадлежность к списку букв. Если знак будет отвергнут, то исполнение завершится переходом от 14-й строки к метке L79 и выдачей сообщения об ошибке. Если заданная пользователем буква имеется в списке, то она станет значением переменной V1, и тогда далее будет исследован только этот диск. Если же буква диска в строке вызова файла DISK.BAT вообще не была указана, то будет исследована группа дисков, буквенные обозначения которых станут значением переменной V1 в 15-й строке.

При исследовании дисков в каталоге для временных файлов придется создавать три файла: TMP.SCR, TMP.TXT, TMP.BAT. Чтобы гарантировать беспрепятственное создание временных файлов, операция в строке 17 снимает атрибуты с одноименных файлов, которые к тому моменту могут существовать. Напомним, что значением переменной VA, подставляемым в 17-й строке, является имя программы ATTRIB.EXE с полным предшествующим путем. Первый из временных файлов – TMP.SCR – создается перенаправлениями в строках 18 – 21; содержание и назначение этого файла будут пояснены позднее.

Команда FOR в 23-й строке запускает основной цикл исследования дисков, представленных значением переменной V1. Собственно исследование выполняет подпрограмма, размещаемая в строках 53 – 72. Вызов этой подпрограммы для исследования одного диска выглядит так:

```
call %0 & 53 %%Z: %1
```

где слово "call" означает команду возврата в тот же цикл FOR по окончании каждого предыдущего исследования. Параметр "%0" означает подстановку имени исполняемого batch-файла: DISK.BAT или другого, если он будет назван иначе. Параметры "& 53" направляют переход, выполняемый из второй строки, на метку L53. Вместо параметра %%Z: команда FOR подставит букву того конкретного диска, который должен быть исследован при данном вызове подпрограммы L53.

Обратите внимание, что до входа в цикл FOR, в 16-й строке стоит команда CTTY NUL (3.07), которая разрывает все принимаемые по умолчанию связи функций ввода-вывода с клавиатурой и дисплеем. С этого момента будут действовать только те связи, которые указаны явно. Разрыв принимаемых по умолчанию связей позволит избежать появления на экране бесчисленных сообщений об ошибках при попытках доступа к недействующим и несуществующим дискам. Однако для отслеживания ошибок набора самого файла DISK.BAT выведение сообщений об ошибках может быть полезным, и тогда команду CTTY NUL надо будет временно сделать недействительной, поставив перед ней в той же 16-й строке команду REM (3.24). Когда Вы будете уверены, что ошибок набора нет, тогда исходный вид 16-й строки можно будет восстановить.

Подпрограмма исследования диска начинается с метки :L53. Первая проверка производится в 54-й строке и начинается с команды

```
%Comspec% /f /c Dir /-p %3\nul
```

где слово "%Comspec%" при исполнении заменяется значением переменной COMSPEC, то есть именем используемого командного интерпретатора с полным предшествующим путем к нему. Следовательно, каждая проверка начнется с загрузки еще одного резидентного модуля командного интерпретатора. Передаваемый ему параметр "/f" заставит его работать безостановочно, автоматически отвечая "FAIL" на все вопросы, которые могут возникнуть при неудачных попытках доступа. Параметр "/c" означает, что загружаемый резидентный модуль должен будет исполнить только одну команду "DIR", и потом сразу выгрузиться из памяти, возвратив управление для продолжения исполнения файла DISK.BAT. Сообщение, которое резидентный модуль пошлет в канал STDOUT, в 54-й строке перенаправлено во временный файл TMP.TXT.

Содержание временного файла TMP.TXT проверяется программой FIND.EXE четыре раза: в 55-й, 57-й, 59-й и 66-й строках. Напомним, что имя программы FIND.EXE вместе с путем к ней вводится в эти строки путем подстановки значения переменной VF. Первая проверка выявляет несуществующие диски, вторая – случаи отсутствия сменного диска в дисководе, третья проверка выявляет оптические дисководы (CD-ROM). Если условия проверки не выполнены, то

происходит переход на метку L79, где на экран выводится соответствующее сообщение. Через первые три проверки пройдут только те диски, которые существуют, вставлены в дисковод и не являются оптическими дисками CD-ROM.

Строка, выделенная программой FIND.EXE при первой проверке в 55-й строке, записывается во временный файл TMP.BAT; его содержимое, к примеру, может выглядеть так:

```
Volume in drive D is EXTENDED1
```

После первых трех проверок, в 62-й строке, содержимое файла TMP.BAT передается как данные отладчику DEBUG.EXE, который принимает команды из заранее созданного временного файла TMP.SCR. Файл TMP.SCR, созданный посредством перенаправлений из строк 18 – 21, содержит следующее:

```
e 100 'call %1 & 46' 20
w
q
```

Первая команда "e 100" (6.05-05) из файла TMP.SCR заставляет отладчик DEBUG.EXE перезаписать начальную часть загруженной в него строки данных, которая тогда принимает следующий вид:

```
call %1 & 46 ve D is EXTENDED1
```

Вторая команда "w" (6.05-19) из файла TMP.SCR заставляет отладчика DEBUG.EXE записать измененную строку данных обратно в файл TMP.BAT, а третья команда из файла TMP.SCR закрывает сеанс работы с отладчиком DEBUG.EXE.

После внесенных изменений в файле TMP.BAT оказывается команда CALL (3.02) командного интерпретатора COMMAND.COM, в результате исполнения которой будет вызвана программа, заданная значением первого параметра %1. Это так и происходит, когда в 63-й строке файл TMP.BAT ставится на исполнение, причем его первым параметром становится параметр %0 batch-файла, то есть сам файл DISK.BAT. Таким образом, в 63-й строке происходит рекурсивный вызов файла DISK.BAT, но при этом второй параметр задает переход из второй строки на метку L46, а 6-й параметр – метку тома испытуемого диска. Далее команды, показанные в строках 47 – 51 файла DISK.BAT, заменяют прежнее значение переменной окружения V1 меткой тома испытуемого диска. Тем самым миссия подпрограммы L46 оказывается выполненной, и интерпретатор возвращается к исполнению строки 64 подпрограммы L53.

Команда в строке 64 файла DISK.BAT подготавливает значение переменной V2 с единственной целью: избежать переноса строки 65, которая иначе была бы слишком длинной. Цикл FOR в 65-й строке исполняет две операции, определяемые

значениями переменных VL, V1 и V2. Первая операция – попытка записи метки тома обратно на диск, а вторая – условное удаление файла TMP.BAT, если оставленное после попытки записи значение кода уровня ошибки (errorlevel) отражает неудачное завершение попытки записи. С этого момента существование файла TMP.BAT является свидетельством успешной записи метки тома, и, следовательно, доступности испытуемого диска для записи.

Проверка наличия серийного номера диска в файле TMP.TXT, выполняемая в 66-й строке, добавляет к существованию файла TMP.BAT еще один аргумент, выраженный кодом уровня ошибки (errorlevel). Полученных двух аргументов оказывается достаточно для того, чтобы в строках 67 – 71 идентифицировать статус исследуемого диска. После этого исполнение подпрограммы исследования диска завершается вызовом подпрограммы L73 или L79, которые выводят на экран соответствующие сообщения.

Подпрограмма L73 вызывается в случае успешного завершения, когда после исследования можно в 78-й строке исполнить команду, показывающую размер диска и процент использованного дискового пространства. Подпрограмма L79 вызывается тогда, когда статус исследуемого диска не дает надежды на получение дополнительных сведений о нем. После вывода сообщения о результате исследования управление возвращается к продолжению исполнения цикла FOR в строке 23, из которого подпрограмма исследования диска была вызвана.

Цикл FOR в 23-й строке продолжит вызовы подпрограммы L53 для исследования каждого диска, включенного в список подлежащих исследованию дисков. Когда все диски будут исследованы, команда DEL в строке 24 удалит оставшиеся временные файлы, команда CTTY CON в 25-й строке восстановит нормальные связи функций ввода-вывода с аппаратурой компьютера, а цикл FOR в 26-й строке уничтожит все локальные переменные окружения. Из 27-й строки основной части программы происходит переход на конечную метку L87, и на этом исполнение batch-файла DISK.BAT завершается.

При пользовании batch-файлом DISK.BAT нужно иметь ввиду, что ему необходим прямой доступ к исследуемым дискам. Поэтому исполнять файл DISK.BAT в "окне DOS" операционной системы Windows нельзя, его можно исполнять только в среде MS-DOS7 или MS-DOS8. Помимо того, должны быть удовлетворены все пять общих условий исполнения batch-файлов, оговоренных во вводной статье к главе 9.

9.04 Конфигурации с перебазированием на RAM-диск.

Предлагаемая здесь пара конфигурационных файлов (CONFIG.SYS и AUTOEXEC.BAT) предоставляет выбор из двух альтернатив: обычный вариант загрузки MS-DOS7 и вариант с перебазированием MS-DOS7 на виртуальный RAM-диск, создаваемый в выделенной области памяти компьютера. Виртуальный RAM-диск работает гораздо быстрее, чем любой реальный диск. Пользование виртуальным диском снижает нагрузку на реальный диск и уменьшает его износ. Однако при проведении восстановительных и настроечных работ к пользованию виртуальным диском приходится прибегать по другой, более прозаической причине. Пока операционная система работает со сменного загрузочного диска, извлечь из дисковода этот диск нельзя, он просто мешает использовать дисковод для считывания требуемых программ с других дисков. Необходимо перебазировать операционную систему хоть куда-нибудь, и при таких обстоятельствах самой подходящей кандидатурой становится виртуальный RAM-диск.

Для образования виртуальных RAM-дисков в составе MS-DOS7 имеется драйвер RAMDRIVE.SYS (5.05-01). Общей проблемой для многих драйверов RAM-дисков, включая RAMDRIVE.SYS, является невозможность произвольного задания буквы диска. Драйверы просто берут первую свободную букву, которую предоставляет им MS-DOS. В результате RAM-диску оказывается присвоена буква, следующая за буквой последнего логического диска. Однако число логических дисков в разных компьютерах – величина непостоянная, так что буква RAM-диска заранее неизвестна, ее еще надо определить. С этой целью в состав MS-DOS8 включены два специальных файла: batch-файл Findramd.bat и исполняемый файл Findramd.exe. Представленная здесь пара конфигурационных файлов решает ту же задачу иначе, исключительно с помощью стандартных средств MS-DOS7 и без привлечения каких-либо дополнительных файлов.

9.04-01 Файл CONFIG.SYS с драйвером RAMDRIVE.SYS.

Эта версия файла CONFIG.SYS представляет пример относительно простой блочной структуры. Первый блок с зарезервированным именем [menu] предлагает две альтернативы. При его исполнении на экран дисплея выводятся заголовки двух пунктов меню, и пользователю предстоит сделать выбор клавишами стрелок вверх (UP) или вниз (DOWN). Когда выбор подтвержден нажатием клавиши ENTER, загрузчик IO.SYS записывает выбранную альтернативу – "relocation" или "ordinary" – в значение переменной окружения CONFIG, а затем переходит к исполнению команд из одноименного блока файла CONFIG.SYS.

```
[menu]
numlock off
menuitem=relocation, Relocate DOS onto a 5.6 Mb RAM-disk
menuitem=ordinary, MS-DOS 7.10, ordinary loading
menudefault=relocation,20

[relocation]
include=ordinary
devicehigh=\DOS\DRV\Ramdrive.sys 5600 /E

[ordinary]
accdate c- d- e- r-
device=\DOS\DRV\Himem.sys /v
device=\DOS\DRV\Emm386.exe ram v
dos=high,umb,noauto
buffershigh=30,0
fileshigh=30
lastdrivehigh=Z
fcbshigh=1,0
stackshigh=8,256
country=007,866,\DOS\DRV\Country.sys
devicehigh=\DOS\DRV\Dblbuff.sys
devicehigh=\DOS\DRV\Ifshlp.sys
devicehigh=\DOS\DRV\Setver.exe
devicehigh=\DOS\DRV\Atapimgr.sys /W:6 /NDR /T:5 /LUN
devicehigh=\DOS\DRV\Oakcdrom.sys /D:CD001

[common]
installhigh=\DOS\DRV\Ctmouse.exe
installhigh=\DOS\DRV\Keyrus.com
shell=\Command.com \ /E:2016 /L:511 /U:255 /p
```

Блок [ordinary] очень похож на содержимое файла CONFIG.SYS, представленного в разделе 9.01-01, но есть два различия. Во-первых, введена загрузка драйвера ATAPIMGR.SYS (5.07-01), открывающего доступ к дискам DVD. Второе, более существенное различие состоит в том, что ни в блоке [ordinary], ни во всем файле CONFIG.SYS нет загрузки драйвера MSCDEX.EXE (5.08-03), потому что здесь она повлияла бы на присвоение букв логическим дискам. Эквивалентный драйвер SHSUCDX.COM (5.08-04) будет загружен, но позже, в ходе интерпретации строк файла AUTOEXEC.BAT (9.04-02).

Разумеется, все пути, указываемые в файле CONFIG.SYS, должны соответствовать действительному размещению драйверов, и это соответствие надо

будет так или иначе обеспечить. Если Вы изберете альтернативу [ordinary], то MS-DOS7 загрузится как обычно и продолжит работать с загрузочного диска.

Блок [relocation] состоит всего из двух строк. Команда "include=" в первой строке заставляет загрузчик IO.SYS исполнить все командные строки блока [ordinary]. Потом вторая строка блока [relocation] загружает драйвер RAMDRIVE.SYS (5.05-01). Он создает в расширенной памяти RAM-диск емкостью 5600 кбайт. Такой размер диска можно себе позволить на компьютерах, имеющих память не менее 8 Мбайт, то есть фактически на всех современных и даже не очень современных компьютерах.

Последний блок в файле CONFIG.SYS имеет зарезервированное имя [common]. Блок с таким именем исполняется всегда, независимо от избранной альтернативы. Там загружаются драйвер "мыши" Ctmouse.exe (5.03-03) и комбинированный драйвер Keyrus.com (5.02-05) для переключения символов знакогенератора и раскладок клавиатуры. Последняя строка в блоке [common] передает управление командному интерпретатору COMMAND.COM: ему предстоит завершить конфигурирование исполнением команд из файла AUTOEXEC.BAT (9.04-02).

9.04-02 Файл AUTOEXEC.BAT с перебазированием на RAM-диск.

Приведенный здесь вариант файла AUTOEXEC.BAT написан для случая загрузки MS-DOS7 с диска A:, обязательно содержащего каталог \DOS. Диск A: может быть представлен дискетой в реальном дисководе или может быть эмулирован с копии, записанной на диск CD-ROM. Если Вы хотите использовать для загрузки компьютера другой диск, то надо будет

– изменить назначаемую букву текущего диска в строке 5;
– исключить этот диск из списка в строке 3 секции "_relocation";
– исправить условия команд "IF" в строках 4 и 5 секции "_relocation".

Все перечисленные правки, конечно, можно автоматизировать, если определить букву текущего диска с помощью программы Reassign.com (9.06), например так, как предложено в разделе 9.01-03. Однако здесь лучше акцентировать внимание на другом: на принципе поиска назначенного RAM-диску буквенного обозначения, тем более что подобные сценарии загрузки, как правило, разворачиваются именно с диска A:.

```
@echo off
if %1"==J" if not %2"==" goto _%2
prompt $p$g
set dircmd= /A /O:GNE /P
set dsk=A:
set comspec=%dsk%\Command.com
goto _%config%
```

```
:_relocation
echo.
echo Seeking RAM-disk as the last valid disk...
for %%Z in (C D E F G H I J K L) do call \Autoexec.bat J test %%Z:
if A:==%dsk% echo RAM-disk is not found!
if A:==%dsk% goto _ordinary
echo RAM-disk is assumed to be %dsk%
echo.
set comspec=%dsk%\Command.com
\DOS\MS7\Xcopy.exe \*.* %dsk%\ /S /E
%dsk%\Autoexec.bat J ordinary

:_ordinary
%dsk%
cd \
if not exist TEMP\nul %comspec% nul /f /c md TEMP
if exist TEMP\nul set Temp=%dsk%\TEMP
if %Temp%"==" echo Note: the TEMP variable is left not defined!
Lh \DOS\DRV\Shsucdx.com /D:?CD001 /L:N /~+ /R /Q
set VC=%dsk%\DOS\VC4
path ;
path=%VC%;%dsk%\DOS\OTH;%dsk%\DOS\MS7
Vc.com /TSR /no2E /noswap
goto _end

:_test
echo Testing disk %3 ...
%comspec% nul /f /c if exist %3\nul cd DOS
if exist ..\nul set dsk=%3
if exist ..\nul echo              valid
if not exist ..\nul echo              inaccessible
cd \
:_end
```

Эта версия файла AUTOEXEC.BAT начинается с условного перехода в строке 2, обеспечивающего возможность рекурсивного вызова подпрограмм. Когда файл AUTOEXEC.BAT интерпретируется впервые, этот переход не происходит. Тогда далее производятся обычные присвоения значений переменным окружения. В 7-й строке происходит переход на метку, записанную в значение переменной %CONFIG% еще при исполнении файла CONFIG.SYS (9.04-01). Этим значением может быть либо "relocation", если Вы избрали перебазирование на RAM-диск, либо "ordinary", если Вы предпочли DOS, работающую с загрузочного диска.

Если значением переменной %CONFIG% является слово "ordinary", то будут интерпретироваться команды из секции "_ordinary". Они включают присвоение значений переменным %Temp% и %Path%, загрузку драйвера SHSUCDX.COM для доступа к дискам CD/DVD-ROM, а также файл-менеджера VC.COM. Кончается такая загрузка тем, что DOS продолжает работать с загрузочного диска.

Если же значением переменной %CONFIG% является слово "relocation", то будут интерпретироваться команды из секции "_relocation". В третьей строке этой секции исполняется цикл FOR, рекурсивно вызывающий подпрограмму тестирования из секции "_test" того же самого файла AUTOEXEC.BAT. Тестированию подвергаются диски, перечисленные в скобках в строке того же цикла FOR. Тестирование выполняется с целью обнаружить последнюю букву диска, которой соответствует реально доступный, действующий диск. Пока драйверы IFS и сетевые драйверы не загружены, найденная последняя "действующая" буква диска – это как раз та, которая присвоена RAM-диску. Именно по этой причине, кстати, загрузка драйвера SHSUCDX.COM была отложена и производится только во время исполнения команд из секции "_ordinary".

Секция "_test" файла AUTOEXEC.BAT принимает букву диска, который надо тестировать, через свой третий параметр %3. Во второй строке секции "_test" проверяется наличие корневого каталога тестируемого диска, причем эта проверка применима даже к недоступным и к несуществующим дискам. Если корневой каталог на проверяемом диске "отзывается", то на текущем загрузочном диске происходит переход в каталог \DOS. А в следующей строке проверяется наличие родительского каталога по отношению к текущему: у каталога \DOS родительский каталог имеется, а у корневого – нет. Если смена текущего каталога произошла, то буква тестируемого диска становится значением переменной %dsk%. Команда в последней строке секции "_test" снова делает текущим корневой каталог загрузочного диска.

При циклическом исполнении подпрограммы тестирования, буквы дисков из подготовленного списка, соответствующие доступным дискам, последовательно сменяют друг друга в значении переменной %dsk%. Но буква, соответствующая последнему действующему диску, останется зафиксированной в значении переменной %dsk% к моменту окончания цикла тестирования. Это будет как раз та буква, которая присвоена RAM-диску.

После окончания цикла FOR, вызывавшего подпрограмму тестирования, исполнение командных строк в секции "_relocation" продолжится. Значение переменной %COMSPEC% будет переопределено и на сей раз станет указывать на корневой каталог RAM-диска. Затем программа XCOPY.EXE скопирует все не-скрытые файлы вместе со всей структурой каталогов с загрузочного диска на RAM-диск. Скрытые системные файлы (IO.SYS и MSDOS.SYS) к этому моменту

уже сделали свое дело и для работы MS-DOS7 больше не нужны. Имейте ввиду, что программе XCOPY.EXE для копирования необходимо, чтобы в одном каталоге с ней (то есть в \DOS\MS7) находился еще файл XCOPY32.EXE (6.26).

Последняя строка в секции "_relocation" вызывает на исполнение не сам файл AUTOEXEC.BAT, а его копию, созданную программой XCOPY.EXE в корневом каталоге RAM-диска, потому что значение переменной %DSK% теперь уже другое и указывает именно на RAM-диск. По той же причине вся адресация с участием переменной %DSK% далее будет относиться к RAM-диску.

Исполнение копии файла AUTOEXEC.BAT начнется с первой строки в секции "_ordinary". Эта операция играет важную роль: она переназначает текущий диск с загрузочного на RAM-диск. С этого момента загрузочный диск становится не нужен: все файлы на нем закрыты, и его можно вынуть из дисковода. Теперь MS-DOS7 будет работать с RAM-диска. Дальнейшие операции в секции "_ordinary" будут выполняться так, как будто MS-DOS7 нормально загружается с RAM-диска.

9.05 Примеры простейших программ

Хотя отладчик DEBUG.EXE (6.05) – не самый удобный инструмент для создания исполняемых программ, тем не менее писать программы COM-формата на нем все-таки можно. COM-формат замечателен тем, что программа "раскладывается" для исполнения в памяти компьютера точно так, как она представлена в файле. Поэтому начинать надо с программ COM-формата и, конечно, с самых простых. В представленных здесь простейших программах нет выделенных элементов структуры, нет обращений к дискам, к файлам, к пользователю. Это позволяет не забивать Вам голову многими ограничениями, с которыми необходимо считаться при составлении более сложных программ.

Далее приведены примеры двух простейших программ, написанных спонтанно для решения неожиданно возникшей проблемы. Позднее стало известно, что независимо от меня в тех же целях были написаны более совершенные программы. Это подтвердило актуальность задач, но не изменило намерения привести здесь свои, самые примитивные версии. Ведь на начальном этапе главное – не в совершенстве программ. Самый главный результат – Ваше умение их понимать и создавать.

9.05-01 Программа коррекции яркости синего цвета.

Поводом для написания самой простой программы послужила замена старого дисплея на кинескопе новым жидкокристаллическим дисплеем. Из-за различия их модуляционных характеристик привычный темно-синий фон панелей файл-менеджера стал слишком ярким, вызывающим зрительное раздражение.

Чтобы возвратить панелям комфортный вид, была написана программа BLUE.COM, снижающая уровень яркости синего цвета в цифро-аналоговом преобразователе видеокарты. Программа рассчитана только на видеорежим 03 и не имеет настроек. Тем не менее она оказалась полезной. Может быть, она будет полезна и Вам тоже.

Файл программы BLUE.COM получается в результате исполнения отладчиком DEBUG.EXE последовательности команд из командного файла BLUE.SCR, который надо набрать с помощью программы редактирования текстов (как показано во вводной статье к главе 9) и потом переслать отладчику через перенаправление ввода, например, так:

```
Debug.exe < Blue.scr
```

Командный файл BLUE.SCR должен содержать следующие строки:

```
A 100
MOV         AX,1010      ;100 Функция установки яркости (8.01-24)
MOV         BX,0001      ;103 Номер регистра преобразователя
MOV         CX,0015      ;106 CL - уровень фона синего цвета,
MOV         DX,0000      ;109        CH -зеленого, DH - красного
INT         10           ;10C Вызов функции BIOS
MOV         AX,4C00      ;10E Завершение программы (8.02-55)
INT         21           ;111 Вызов функции DOS

N Blue.com
R BX
0000
R CX
0013
W
Q
```

Первая строка командного файла BLUE.SCR переводит отладчик DEBUG.EXE в режим ассемблирования начиная с адреса CS:0100. Следующие семь строк определяют все действия программы BLUE.COM. Смысл каждого действия ясен из комментария, приведенного после знака точки с запятой в той же строке. Девятая строка оставлена пустой (7.01-04), она заставит отладчик DEBUG.EXE выйти из режима ассемблирования. Затем команда "N" (6.05-12) объявляет имя создаваемого исполняемого файла, а его длина (00000013h = 19 байт) вводится в регистры BX и CX. Команда "W" (6.05-19) в предпоследней строке вызывает запись набранных машинных команд в файл BLUE.COM. Наконец, команда "Q" в последней строке завершит сеанс работы отладчика DEBUG.EXE.

При исполнении команд файла BLUE.SCR отладчик DEBUG.EXE показывает на экране листинг. По нему можно контролировать правильность набора файла BLUE.SCR, сверяя смещение каждой команды с должным значением, которое представлено номером в начале комментария к той же строке. Методика сверки листинга детально изложена в разделе 9.07-01. Когда ошибок в листинге нет, полученный файл BLUE.COM можно запускать на исполнение, дальнейшее тестирование для настолько простых программ не требуется. Если установленный уровень яркости синего цвета Вас не удовлетворит, то надо будет изменить численное значение, вводимое в регистр CX в 4-й строке файла BLUE.SCR, и затем еще раз послать его через перенаправление ввода отладчику DEBUG.EXE.

Программу BLUE.COM запускают из той строки файла AUTOEXEC.BAT, которая предшествует вызову файл-менеджера. Помимо того, встречаются программы, задающие стандартные уровни яркости заново. К ним относятся, в частности, SCANDISK.EXE (6.21) и просмотрщик LXPIC.EXE, упоминаемый в одной из строк файла VC.EXT (6.25-03). После исполнения таких программ файл BLUE.COM приходится запускать снова. Это удобно делать автоматически, вписав вызов BLUE.COM в следующую строку batch-файла или того же файла VC.EXT. Здесь такие примеры не показаны, потому что целесообразность коррекции яркости синего цвета зависит от индивидуальных особенностей зрительного восприятия. Тем не менее цветокоррекция, выполняемая программой BLUE.COM, отображена на рис. 3 (в разделе 6.09) и на рис. 5 (в разделе 6.25-01).

9.05-02 Программа выключения электропитания.

Еще один повод для написания очень простой программы появился в 1999 году, когда на смену компьютерам в корпусах AT пришли компьютеры в корпусах ATX. Оказалось, что все они рассчитаны на программное выключение электропитания, что действие кнопки выключателя на передней панели системного блока зависит от настроек BIOS Setup, и его не всегда удается предугадать. В DOS программы выключения электропитания никогда не было. Пришлось ее написать, и она получила название TURN_OFF.COM.

Исполняемый файл программы TURN_OFF.COM получается в результате исполнения отладчиком DEBUG.EXE последовательности команд из файла TURN_OFF.SCR, который должен быть набран с помощью программы редактирования текстов (как показано во вводной статье к главе 9) и потом переслан отладчику через перенаправление ввода, например, так:

```
Debug.exe < Turn_off.scr
```

Ниже приведен полный текст файла TURN_OFF.SCR.

```
a 100
mov         AX,5301      ;100 Активизация APM реального режима
mov         BX,0000      ;103 Введем идентификатор APM BIOS
int         15           ;106 Вызов процедуры активизации APM
mov         AX,530E      ;108 Запрос версии APM BIOS
mov         BX,0000      ;10B Введем идентификатор APM BIOS
mov         CX,0102      ;10E Запросим версию 1.2 APM
int         15           ;111 Вызов процедуры запроса версии
mov         AX,5307      ;113 Запрос смены режима электропитания
mov         BX,0001      ;116 Введем идентификатор всех приборов
mov         CX,0003      ;119 Запрос операции выключения
int         15           ;11C Вызов процедуры смены режима
mov         AX,4C00      ;11E Код процедуры завершения
int         21           ;121 Вызов процедуры завершения

n turn_off.com
r BX
0000
r CX
0023
w
q
```

Из текста программы видно, что она действительно очень проста: в ней нет условных переходов, используются лишь два вида ассемблерных команд, полная длина исполняемого файла составляет всего лишь 35 (23h) байт.

В первой строке команда "a 100" (6.05-02) переводит отладчик DEBUG.EXE в режим ассемблирования. В следующих 13 строках происходит набор машинных команд. В этих строках правее знака точки с запятой имеются комментарии, из которых становится понятна роль каждой команды. Дополнительные сведения о прерываниях INT 15\AH=53h приведены в разделах 8.01-70 – 8.01-72.

После 14-й строки текста следует пустая строка, которая обязательно должна быть в командном файле: она выведет отладчик из режима ассемблирования. Затем команда "n" (6.05-12) объявляет имя создаваемого исполняемого файла, а его длина вводится в регистры BX и CX. Команда "w" (6.05-19) в предпоследней строке вызывает запись набранных машинных команд в файл. Наконец, команда "q" в последней строке завершает сеанс работы отладчика DEBUG.EXE.

Настолько простые программы фактически нечего отлаживать, достаточно их набрать без ошибок. Для контроля правильности набора комментарий в каждой строке начинается с номера: это смещение вводимой в данной строке команды. Смещение надо сверять по выводимому отладчиком листингу. Методика сверки

листинга детально изложена в разделе 9.07-01. На всякий случай полезно проверить длину созданного файла либо командой DIR (3.10), либо по данным в панелях файл-менеджера Volcov Commander (6.25).

При пользовании программой TURN_OFF.COM нужно иметь ввиду, что большинство компьютеров выпуска 1990-х годов не оснащено средствами APM BIOS, и потому проигнорирует заложенные в программу вызовы. Кроме того, нельзя напрямую обращаться к APM BIOS из "окна DOS" операционной системы Windows. Программу TURN_OFF.COM следует вызывать только из среды DOS, причем ее удобно вызывать через меню файл-менеджера, например так, как показано в последних строках файла VC.MNU в разделе 6.25-02.

С позиции пользования, конечно, было бы целесообразно предусмотреть в программе TURN_OFF.COM выдачу сообщения об ошибках, например, о вызове из "окна DOS" или об отсутствии в компьютере средств APM BIOS. Но тогда программа не была бы настолько проста. Однако исправить ситуацию несложно. Например, опознавание среды Windows реализовано в строках 13A – 141 файла из раздела 9.10-02. Многочисленные варианты выведения сообщений об ошибках показаны в разделах 9.06, 9.08 и 9.10. Ознакомившись с упомянутыми примерами, Вы сможете сами потом усовершенствовать файл TURN_OFF.COM.

Примечание 1: если в момент выключения электропитания кэш-буфер драйвера SMARTDRV.EXE (5.06-01) не пуст, то остающиеся там данные пропадают. Это может случиться как при выключении посредством программы TURN_OFF.COM, так и при неожиданном пропадании напряжения в электросети. Если драйвер SMARTDRV.EXE все же приходится применять, то можно намеренно сбрасывать данные из кэш-буфера на диск перед вызовом программы TURN_OFF.COM, но лучше кэширование операций записи вообще запретить.

9.06 Программа ввода данных в переменную окружения

Давным-давно DOS разрабатывалась как операционная система для широкого круга пользователей, и с этих позиций было сформировано множество выполняемых ею команд. Но оно уже не вполне соответствует той роли, которую DOS играет сейчас. Данный раздел представляет небольшую программу для выполнения трех операций, которых очень недостает в наборе команд DOS. В частности, именно эта программа позволяет осуществить адаптивный сценарий загрузки MS-DOS7, описанный в разделе 9.09.

Предлагаемая программа заменяет значение существующей переменной окружения другими данными, и потому названа Reassign.com. Она действует в кооперации со встроенной командой SET (3.26) командного интерпретатора, не

дублируя ее функции. Команда SET создает переменную окружения, первая цифра значения которой определяет операцию для программы Reassign.com:

1 – сообщить размер набольшего свободного блока XMS-памяти;
2 – принять ввод данных с клавиатуры;
3 – сообщить букву текущего логического диска.

Последующие знаки в значении подготовленной переменной могут быть любыми, но их наличие определяет место для тех данных, которые возвратит программа Reassign.com. Если этого места недостаточно для размещения запрошенной величины, то на экран будет выведено сообщение об ошибке. Если результат не заполнит все предоставленное место, то оставшаяся часть будет заполнена знаками пробела (20h).

При загрузке DOS на незнакомый компьютер буквенное обозначение текущего диска, назначаемое системой BIOS, бывает заранее неизвестно. Операция 3 программы Reassign.com поможет определить, какая буква назначена тому логическому диску, с которого производится загрузка:

```
set disk=33
Reassign disk
echo Current disk is %disk%
```

Подготовленное значение 33 переменной disk будет заменено буквой текущего диска, например, D:. Когда буква текущего диска известна, вся дальнейшая адресация в конфигурационных файлах может быть адаптирована автоматически.

Следующей проблемой адаптивных процедур загрузки является определение размера RAM-диска, на который должна быть перенесена DOS, поскольку заранее объем памяти компьютера обычно неизвестен. Но функция 1 программы Reassign.com даст ответ:

```
set xms=11111
Reassign xms
echo Largest free XMS memory block is %xms% kb
```

Последняя строка приведенного примера покажет результат на экране. Теперь Вы готовы задать размер RAM-диска, и программа Reassign.com примет задаваемое Вами значение посредством своей функции 2:

```
echo Specify the size of RAM-disk in kb:
set ramdisk=22222
Reassign ramdisk
Tdsk.exe R: %ramdisk% 512 /M /F:2
```

В ходе исполнения операции 2 программа Reassign.com пригласит Вас ввести желаемое значение. Неправильно набранные знаки можно убрать нажатием

клавиши Backspace. Ввод данных закончится по нажатию клавиши ENTER. В последней строке приведенного примера введенное Вами значение будет подставлено в перечень параметров программы Tdsk.exe (5.05-02), которая создаст RAM-диск указанного объема. Конечно, найдется много других поводов применить программу Reassign.com, помимо тех, которые показаны здесь.

Важно отметить, что все представленные выше примеры не будут работать из командных строк файл-менеджеров NC (Norton Commander), VC (Volcov Commander) и тому подобных. Причина в том, что файл-менеджеры, как правило, исполняют каждую строку в отдельном пространстве окружения, вследствие чего значения переменных, заданные командой SET, от предшествующей строки к следующей не передаются. Но в обычной "чистой" командной строке DOS и в составе batch-файлов команды исполняются в общем пространстве окружения, так что ввод данных во всех представленных примерах будет происходить правильно.

Программа REASSIGN.COM получается в результате исполнения отладчиком DEBUG.EXE последовательности команд. Эту последовательность надо записать в текстовый командный файл REASSIGN.SCR с помощью программы редактирования текстов, как показано во вводной статье к главе 9. Словесные комментарии при наборе текста можно опустить. Обратите внимание на пустую строку в ассемблерном тексте – восьмую с конца. Она выводит отладчик из режима ассемблирования (7.01-04) и потому в файле REASSIGN.SCR обязательно должна быть сохранена. Затем файл REASSIGN.SCR надо переслать на исполнение отладчику DEBUG.EXE через перенаправление ввода, например, так:

```
Debug.exe < Reassign.scr
```

В результате исполнения командного файла отладчик должен создать в текущем каталоге исполняемый файл REASSIGN.COM длиной 992 байта. Необходимый для этого командный файл REASSIGN.SCR содержит следующие строки:

```
a 100
      ;************* Программа Reassign.com **************
      ;********* Секция 1: начальные приготовления
         ; 110 - адрес перехода со строки 104
cmp           SP,2010       ; 100 Выделено менее 8 кбайт?
jbe           0110          ;*104 Если да, оставим как есть
mov           SP,1FFE       ; 106 Вершину стека - на 8 кбайт
mov           BX,0200       ; 109 Запросим 8 кбайт памяти
mov           AH,4A         ; 10C Вызов функции создания
int           21            ; 10E          свободного MCB
mov           DX,03A8       ;=110 Сообщение 4 (help)
cmp byte ptr  [005D],20     ; 113 1-й байт в FCB заполнен?
jz            0151          ;*118 Если нет, выведем help
```

```
cmp byte ptr [005D],3F     ; 11A 1-й байт - знак вопроса?
jz           0151          ;*11F Если да, выведем help
cld                        ; 121 Зададим счет вверх (DF=UP)
      ;********** Секция 2: Результаты поиска переменной
call         0173          ;*122 Вызовем подпрограмму поиска
cmp byte ptr [0165],F7     ; 125 Ошибки 1-го и 2-го циклов?
ja           0151          ;*12A Выйдем, если имеются
mov          DX,0345       ;*12C Введем адрес 2-го сообщения
les          DI,[00F8]     ; 12F Адрес переменной - в ES:DI
ES:                        ; 133 Считываем 1-й знак
mov          BL,[DI]       ; 134        значения переменной
cmp          BL,31         ; 136 Нижняя граница: функция 1
jb           0151          ;*139 Выйдем, если значение меньше
cmp          BL,33         ; 13B Верхняя граница: функция 3
ja           0151          ;*13E Выйдем, если значение больше
shl          BL,1          ; 140 x2, т.к. адрес - это 2 байта
mov          BH,00         ; 142 Подготовим BX к переходу
call         [BX+0105]     ;*144 Вызов основных функций
jz           0151          ;*148 ZR - вывод сообщения
jb           015F          ;*14A CY - errorlevel из DH
      ;************* Секция 3: завершение программы.
         ; 151 - переходы от 118, 11F, 12A, 139, 13E, 148
         ; 15F - адрес перехода со строк 14A, 14F
call         01F9          ;*14C Подпрограмма копирования
jmp          015F          ;*14F Переход к завершению
mov          BX,DX         ;=151 DS:DX - адрес сообщения
mov          CX,[BX-02]    ; 153 CX = число знаков
mov          BX,0001       ; 156 BX = ссылка на STDOUT
mov          AH,40         ; 159 Вызов функции посылки
int          21            ; 15B      сообщения в STDOUT
mov          DH,F0         ; 15D Errorlevel = F0h
mov          AL,DH         ;=15F Восстановим errorlevel
mov          AH,4C         ; 161 Вызов функции
int          21            ; 163     завершения программы
      ;************** Секция 4: блок данных.
         ; 165 - адрес записи в 125, 1D1, 1D6, 1F4, 205
dw           00FC
         ; 167 = (2*31 + 105) в строке 144, 221, 22D
         ; 169 - в строках 225, 236, 240, 247, 252, 280
dw           0213,029A,02DC,0000,0000,0000
      ;************** Секция 5: поиск и проверка PSP
         ; 173 - адрес вызова от строк 122, 1A3
         ; 19F - адрес перехода со строки 197
         ; 1A6 - переходы от строк 17A, 183, 18F, 19D
```

```
xor          DI,DI        ;=173 Обнулим регистр DI
ES:                       ; 175 Начинается ли сегмент
cmp word ptr [DI],20CD    ; 176             с команды CD20?
jnz          01A6         ;*17A Выйдем, если PSP не опознан
ES:                       ; 17C Имеется ли команда CD21 в
cmp word ptr [0050],21CD  ; 17D               ячейке 0050h?
jnz          01A6         ;*183 Выйдем, если PSP не опознан
push         ES           ; 185 Сохраним адрес PSP
ES:                       ; 186 Загрузим сегментный адрес
mov          ES,[002C]    ; 187     окружения в регистр ES
call         01B3         ; 18B Подпрограмма поиска имени
pop          ES           ; 18E Восстановим адрес PSP в ES
jz           01A6         ;*18F Выйдем, если имя не найдено
mov          AX,ES        ; 191 Сегментный адрес PSP - в AX
mov          DI,0016      ; 193 Равен ли адрес в ES:[0016]
scasw                     ; 196    адресу данного PSP в AX?
jnz          019F         ;*197 Если нет, это PSP-родитель
mov          DI,0010      ; 199 Равен ли адрес в ES:[0010]
scasw                     ; 19C    адресу данного PSP в AX?
jz           01A6         ;*19D Если да, ничего мы не нашли
ES:                       ;=19F Загрузим в ES сегментный
mov          ES,[DI-02]   ; 1A0     адрес родительского PSP
call         0173         ; 1A3 Вызовем проверку этого PSP
ret                       ;=1A6 Возврат из подпрограммы
      ;************* Секция 6: подпрограмма поиска имени
         ; 1A7 - адрес перехода со строк 1C8, 1CF
         ; 1B3 - адрес вызова со строки 18B
         ; 1DF - адрес перехода со строк 1B1, 1BA
mov          CX,0100      ;=1A7 Зададим число сравнений
mov          AL,00        ; 1AA Сравниваем со значением 00
repnz                     ; 1AC Повтор, пока не найдем 00
scasb                     ; 1AD Сравниваем [ES:DI] с AL=00
cmp          CX,0000      ; 1AE Если число сравнений
jz           01DF         ;*1B1    исчерпано, кончаем поиск
mov          DX,0318      ;=1B3 Точка входа в цикл поиска
ES:                       ; 1B6 Если [DI]==00h, значит
cmp byte ptr [DI],00      ; 1B7      окружение просмотрено
jz           01DF         ;*1BA      все, кончаем поиск
mov          SI,005D      ; 1BC Адрес имени - в DS:SI
mov          CX,000A      ; 1BF Зададим число сравнений в CX
repz                      ; 1C2 Повтор до первого различия
cmpsb                     ; 1C3 Сравниваем [DS:SI] с [ES:DI]
cmp byte ptr [SI-01],20   ; 1C4 Имя кончается пробелом?
jnz          01A7         ;*1C8 Перейдем к следующему имени
```

```
ES:                         ; 1CA Стоит ли после имени
cmp byte ptr  [DI-01],3D    ; 1CB            знак равенства?
jnz           01A7          ;*1CF Перейдем к следующему имени
sub byte ptr  [0165],04     ;*1D1 Сдвиг адреса записи
mov           SI,[0165]     ;*1D6 Адрес записи - в регистр SI
mov           [SI],DI       ; 1DA Адрес значения переменной
mov           [SI+02],ES    ; 1DC Сегментный адрес окружения
ret                         ;=1DF Возврат из подпрограммы
      ;******** Секция 7: подпрограмма копирования
         ; 1E0 - адрес перехода со строки 210
         ; 1E5 - адрес перехода со строки 1F1
         ; 1F3 - адрес перехода со строк 1E9, 1EE
         ; 1F9 - адрес вызова со строки 14C
         ; 1FB - адрес перехода со строки 202
         ; 204 - адрес перехода со строки 1FF
CS:                         ;=1E0 Загрузим адрес источника
lds           SI,[00F8]     ; 1E1          в регистры DS:SI
ES:                         ;=1E5 Есть ли еще место
cmp byte ptr  [DI],00       ; 1E6       по адресу назначения?
jz            01F3          ;*1E9 Если нет - в следующий цикл
cmp byte ptr  [SI],00       ; 1EB Есть ли знаки в источнике?
jz            01F3          ;*1EE Если нет - в следующий цикл
movsb                       ; 1F0 Скопируем один байт
jmp           01E5          ;*1F1 Проверим следующий байт
CS:                         ;=1F3 Сменим адрес
add word ptr  [0165],0004   ; 1F4        назначения
mov           AL,20         ;=1F9 Точка входа в подпрограмму
ES:                         ;=1FB Есть ли место
cmp byte ptr  [DI],00       ; 1FC      по адресу заполнения?
jz            0204          ;*1FF Если нет, кончим заполнять
stosb                       ; 201 Пробел - в адрес назначения
jmp           01FB          ;*202 Проверим следующий байт
CS:                         ;=204 Загрузим номер
mov           SI,[0165]     ; 205             ячейки в SI
CS:                         ; 209 Загрузим адрес назначения
les           DI,[SI]       ; 20A           в регистры ES:DI
cmp           SI,00F8       ; 20C Номер ячейки - последний?
jnz           01E0          ;*210 Если нет - возврат в цикл
ret                         ; 212 Возврат из подпрограммы
      ;****** Секция 8: функция 1, объем XMS-памяти
         ; 213 - адрес вызова в ячейке 167
         ; 256 - адрес перехода со строк 23E, 247, 250
         ; 25B - адрес перехода со строки 263
         ; 269 - адрес перехода со строки 277
```

```
        ; 275 - адрес перехода со строки 272
        ; 284 - адрес перехода со строки 27E
        ; 285 - адрес перехода со строк 21A, 234
mov           AX,4300       ;=213 Точка входа в функцию 1
int           2F            ; 216 Проверим, установлен ли
cmp           AL,80         ; 218     XMS-драйвер Himem.sys?
jnz           0285          ;*21A Выйдем, если не установлен
mov           AX,4310       ; 21C Запрос адреса вызова
int           2F            ; 21F Адрес вызова - в ES:BX
mov           [0167],BX     ; 221 Смещения адреса вызова
mov           [0169],ES     ; 225 Сегмент адреса вызова
mov           BL,00         ; 229 Зададим 00h в регистре BL
mov           AH,08         ; 22B Код запроса объема
call far      [0167]        ; 22D Запрос к Himem.sys
cmp           BL,00         ; 231 Запрос исполнен успешно?
jnz           0285          ;*234 Выйдем, если не успешно
mov byte ptr  [0169],00     ; 236 Зададим errorlevel = 00
cmp           AX,1900       ; 23B   если XMS-памяти для
jb            0256          ;*23E   6 Mb диска недостаточно
inc byte ptr  [0169]        ; 240 Зададим errorlevel = 01
cmp           AX,4900       ; 244   если 5600 Kb XMS-диск
jb            0256          ;*247   может быть создан
inc byte ptr  [0169]        ; 249 Зададим errorlevel = 02
cmp           AX,8C00       ; 24D   если XMS-диск ограничен
jb            0256          ;*250   объемом 16 Mb, а если
inc byte ptr  [0169]        ; 252   нет, то errorlevel = 03
mov           BP,SP         ;=256 Фиксируем состояние SP
mov           BX,000A       ; 258 Зададим делитель
xor           DX,DX         ;=25B Очистим DX перед делением
div           BX            ; 25D Разделим на 0Ah
push          DX            ; 25F Затолкнем остаток в стек
cmp           AX,0000       ; 260 Процесс деления закончен?
jnz           025B          ;*263 Если нет - следующий цикл
les           DI,[00F8]     ; 265 Адрес переменной - в ES:DI
pop           AX            ;=269 Вытолкнем разряд из стека
add           AL,30         ; 26A Переведем цифру в код ASCII
mov           SI,DI         ; 26C SI фиксирует факт записи
ES:                         ; 26E Есть ли свободное место
cmp byte ptr  [DI],00       ; 26F     по адресу назначения?
jz            0275          ;*272 Обойдем запись, если нет
stosb                       ; 274 Запишем цифру, увеличим DI
cmp           SP,BP         ;=275 Все остатки вытолкнуты?
jb            0269          ;*277 Если нет, вытолкнем еще
mov           DX,036F       ;*279 3-е сообщение об ошибке
```

```
cmp          DI,SI        ; 27C Была ли запись пропущена?
jz           0284         ;*27E
mov          DH,[0169]    ; 280 Считаем errorlevel в DH
ret                       ;=284
mov          DH,FF        ;=285 Errorlevel = FF
stc                       ; 287 Завершение с ошибкой
ret                       ; 288 Возврат из подпрограммы
     ;******* Секция 9: функция 2, ввод с клавиатуры
        ; 289 - адрес перехода со строки 2AD
        ; 29A - переходы от 169, 28D, 291, 295, 2B4, 2BD
        ; 2BF - адрес перехода со строки 2A8
        ; 2C7 - адрес перехода со строк 2A3, 2C3
ES:                       ;=289 Сначала проверим,
cmp byte ptr [DI],00      ; 28A        заполнен ли буфер
jz           029A         ;*28D Если да, ждем 1Bh, 0Dh, 08h
cmp          AL,20        ; 28F Знаки ниже 20h в
jb           029A         ;*291      переменную не вводим
cmp          AL,3D        ; 293 Знак равенства в
jz           029A         ;*295     переменную не вводим
int          29           ; 297 Отобразим введенный знак
stosb                     ; 299 Копируем знак в буфер ES:DI
mov          AH,10        ;=29A Точка входа в функцию 2
int          16           ; 29C Считываем знак с клавиатуры
mov          DH,FF        ; 29E Errorlevel = FFh
cmp          AH,01        ; 2A0 Нажата ли клавиша ESC?
jz           02C7         ;*2A3 Если да, то не копируем
cmp          AH,1C        ; 2A5 Нажата ли клавиша ENTER?
jz           02BF         ;*2A8 Если да, выходим и копируем
cmp          AH,0E        ; 2AA Нажата клавиша BackSpace?
jnz          0289         ;*2AD Если нет, вернемся в цикл
ES:                       ; 2AF Проверим, не будет ли адрес
cmp byte ptr [DI-01],3D   ; 2B0   в регистре DI указывать
jz           029A         ;*2B4   на точку до начала буфера
mov          AX,2008      ; 2B6 Вывод знака возврата и
call         02D2         ;*2B9       пробела через INT 29
dec          DI           ; 2BC Уменьшим указатель в DI
jmp          029A         ;*2BD Вернемся к началу цикла
cmp          DI,[00F8]    ;=2BF Изменилось ли число в DI?
jz           02C7         ;*2C3 Если нет, оставим DH = FFh
mov          DH,00        ; 2C5 Если да, установим DH = 00
mov          AX,0A0D      ;=2C7 Вывод знаков перевода
call         02D2         ;*2CA       строки через INT 29
cmp          DH,20        ; 2CD Установим флаги
cmc                       ; 2D0 Реверсируем флаг переноса
```

```
ret                         ; 2D1 Возврат из подпрограммы
      ;***** Секция 10: функция 2, подпрограмма вывода
         ; 2D2 - адрес вызова со строк 2B9, 2CA
         ; 2D5 - адрес возврата в цикле со строки 2D9
mov           CX,0003       ;=2D2 Установим число повторений
int           29            ;=2D5 Выведем знак на экран
xchg          AL,AH         ; 2D7 Обменяем коды знаков
loop          02D5          ;*2D9 Проверим условие цикла
ret                         ; 2DB Возврат из подпрограммы
      ;***** Секция 11: функция 3, определение буквы диска
         ; 2DC - записан в ячейку 16B
         ; 2EE - адрес перехода со строки 2E9
         ; 2F6 - адрес перехода со строки 30C
         ; 305 - адрес перехода со строки 2FE
         ; 30E - адрес перехода со строк 2F9 и 308
mov           AH,19         ;=2DC Точка входа в функцию 3
int           21            ; 2DE Определим текущий диск
mov           DL,AL         ; 2E0 Скопируем номер диска в DL
add           AL,41         ; 2E2 Преобразуем в букву диска
stosb                       ; 2E4 Скопируем букву в ES:DI
ES:                         ; 2E5      и увеличим число в DI
cmp byte ptr  [DI],00       ; 2E6 Проверим, заполнен ли буфер
jz            02EE          ;*2E9 Если да, не добавим ":"
mov           AL,3A         ; 2EB Введем код ":" в AL
stosb                       ; 2ED Скопируем ":" в буфер ES:DI
mov           DH,00         ;=2EE Предустановим errorlevel
push          DI            ; 2F0 Сохраним DI в стеке
mov           AX,0803       ; 2F1 Запрос адреса 1-й DDT
int           2F            ; 2F4 Адрес 1-й DDT - в DS:DI
cmp           [DI+04],AL    ;=2F6 Это флоппи-диск?
ja            030E          ;*2F9 Если нет, то дальше не ищем
cmp           [DI+04],AH    ; 2FB Если дисковод тот же,
jz            0305          ;*2FE         то его не учитываем
mov           AH,[DI+04]    ; 300 Запомним номер дисковода
inc           DH            ; 303 Увеличим число дисководов
cmp word ptr  [DI],FFFF     ;=305 Данная DDT - последняя?
jz            030E          ;*308 Если да, то дальше не ищем
lds           DI,[DI]       ; 30A Адрес следующей DDT в DS:DI
jmp           02F6          ;*30C Исследуем DDT дальше
pop           DI            ;=30E Восстановим DI
push          CS            ; 30F Восстановим первоначальное
pop           DS            ; 310               состояние DS
cmp           DH,FF         ; 311 Сбросим флаг ZF в NZ
clc                         ; 314 Сбросим флаг переноса
```

```
ret                          ; 315 Возврат из подпрограммы
      ;********************* Секция 12: тексты сообщений
dw            002B
        ; 318 - 1-е сообщение, вызвано от 1B3
db            0D 0A 'ERROR: specified name hasn'
db            27 't been found' 0D 0A
dw            0028
        ; 345 - 2-е сообщение, вызвано от 12C
db            0D 0A 'ERROR: invalid value of the'
db            20 'variable' 0D 0A
dw            0037
        ; 36F - 3-е сообщение, вызвано от 279
db            0D 0A 'ERROR: no space for new value,'
db            20 'old one is too short' 0D 0A
dw            0138
        ; 3A8 - 4-е сообщение (help), вызвано от 110
db            0D 0A 09 'Reassign.com overwrites value'
db            20 'of existing variable' 0D 0A 'Usage:'
db            0D 0A 09 'Reassign Anyname' 0D 0A 'Anyn'
db            'ame - name example (up to 8 letters) o'
db            'f a variable' 0D 0A 'The first in its'
db            20 'value must be a digit 1, 2 or 3 - i'
db            't defines function:' 0D 0A 09 '1 - get'
db            20 'size of largest free XMS block' 0D 0A
db            09 '2 - accept keyboard' 27 's input' 0D
db            0A 09 '3 - get current disk letter' 0D 0A
        ; 4E0

n Reassign.com
rbx
0000
rcx
03E0
w
q
```

Ассемблерный текст программы Reassign.com начинается в секции 1 с высвобождения памяти, которая заведомо не потребуется при исполнении программы (подробнее – в примечании 5 к A.12-7). Затем в строках 110 – 11F выполняется проверка параметров командной строки, которые командный интерпретатор автоматически впишет в первый блок FCB (примечание 4 к A.07-1), начиная со смещения 005Dh, причем там строчные буквы уже переведены в

заглавные. Если ячейка 005Dh не заполнена или если туда вписан знак вопроса, то Reassign.com выведет на экран сообщение помощи (help) из секции 12 и вернет управление командному интерпретатору, оставив код уровня ошибки (errorlevel) 240. Во всех других случаях будет считаться, что начиная с ячейки 005Dh в блок FCB вписано имя переменной окружения, значение которой надлежит заменить.

Секция 2 начинается в строке 122 с вызова процедуры поиска переменной с указанным именем в текущем и нижележащих пространствах окружения. Процедура поиска включает подпрограмму проверки PSP из секции 5, а также подпрограмму поиска имени из секции 6. Подлинность PSP проверяется в строках 175 – 183 по характерным сигнатурам в ячейках [0000] и [0050]. Когда подлинность PSP подтверждена, адрес пространства окружения из ячейки [002C] передается подпрограмме поиска имени, вызываемой из строки 18B.

Подпрограмма поиска заданного имени просматривает все пространство окружения и выходит с установленным флагом ZF, если заданное имя не удается найти. Если же переменная найдена, то полный адрес ее значения, включающий сегмент и смещение, записывается в ячейку памяти, причем адрес этой ячейки вычисляется вычитанием 4 байт из указателя в ячейке [0165]. Благодаря такому смещению адреса значений переменной из разных пространств окружения не "затирают" друг друга, а располагаются в свободном конце префикса сегмента программы последовательно, один за другим.

При возврате в подпрограмму проверки PSP со сброшенным флагом ZF процесс поиска надо будет продолжить в нижележащем ("родительском") пространстве окружения. Для этого необходимо сначала найти сегментный адрес нижележащего PSP. В качестве кандидатов рассматриваются сегментные адреса в ячейках ES:[0016] и ES:[0010], и затем по отношению к избранному сегментному адресу подпрограмма проверки PSP в строке 1A3 рекурсивно вызовет сама себя. Все процессы проверки и поиска повторятся в нижележащем PSP и в относящемся к нему пространстве окружения. Рекурсивный спуск в нижележащие PSP будет продолжаться дальше и завершится только тогда, когда очередной PSP окажется недействительным или когда в пространстве окружения очередного PSP заданное имя переменной не удастся найти.

После выхода из цепи рекурсивных вызовов продолжится исполнение команд в секции 2. Там в строке 12F адрес значения искомой переменной, найденный в ближайшем ("верхнем") пространстве окружения, записывается в регистры ES:DI. Этот адрес прямо указывает на первую цифру значения, которая определяет основную миссию программы Reassign.com: какое именно новое наполнение должна получить указанная переменная окружения. Допустимых функций всего три, и им соответствуют коды знаков 31h, 32h, 33h. Если код первой цифры в значении переменной будет какой-нибудь другой, то Reassign.com выведет сообщение "Неверное значение переменной" и вернет управление командному

интерпретатору. Но если все проверки успешно пройдены, то в строке 144 произойдет вызов подпрограммы, исполняющей запрошенную функцию. Адрес точки входа в подпрограмму будет взят из списка 0167 в секции 4.

Исполнение функции 1 начинается со смещения 0213h в секции 8. Сначала вызовом INT 2F\AX=4300h (8.03-22) производится проверка, загружен ли драйвер HIMEM.SYS (5.04-01). Если драйвер не загружен, то Reassign.com возвратит управление командному интерпретатору, не выводя на экран никакого сообщения и оставив код уровня ошибки (errorlevel) 255. Если же драйвер загружен, то следующим будет вызов INT 2F\AX=4310h (8.03-23) с намерением получить адрес точки входа для запросов к драйверу. Возвращаемый полный адрес используется в строке 22D для запроса функции 08h драйвера Himem.sys (A.12-3) посредством команды CALL FAR. Запрошенная функция возвращает несколько параметров, в том числе размер наибольшего свободного блока XMS-памяти. По нему в строках 236 – 252 определяется оставляемый программой код уровня ошибки, позволяющий потом автоматически задать приемлемый объем RAM-диска. В строках 256 – 263 этот размер переводится в форму десятичного числа. В строках 265 – 277 цифры этого числа переводятся в знаки кода ASCII и записываются в то место, где было прежнее значение переменной в текущем ("верхнем") пространстве окружения. Если места оказывается недостаточно, Reassign.com выводит на экран сообщение "прежнее значение было слишком короткое" и возвращает управление командному интерпретатору, оставляя код уровня ошибки (errorlevel) 240. Если же места достаточно, то происходит возврат из подпрограммы в заключительную секцию 3. Оттуда из строки 14C вызывается подпрограмма копирования, которая по мере надобности дополняет записанное значение пробелами и копирует его в нижележащие пространства окружения. На том миссия функции 1 завершается.

Исполнение функции 2 начинается со строки 029A в секции 9. Вводимые с клавиатуры знаки воспринимаются и записываются в то место, где было прежнее значение переменной в текущем ("верхнем") пространстве окружения. Ввод знаков посредством прерывания INT 16\AH=10h (в строке 29C) и выведение их на экран посредством прерывания INT 29 (в строке 297) не подвержены перенаправлениям ввода-вывода на уровне DOS. Команды в строках 2AF – 2BD заполняют место предыдущего знака кодом пробела, когда пользователь нажимает клавишу Backspace. Reassign.com выходит из цикла ввода знаков по нажатию клавиши ENTER или клавиши ESC. При нажатии клавиши ESC команды в строках 2CD – 2D0 устанавливают флаги так, что после возврата из подпрограммы копирование нового значения в нижележащие пространства окружения не осуществляется. Переменная сохранит свое прежнее значение, и программа завершится с кодом уровня ошибки (errorlevel) 255. После нажатия клавиши ENTER возврат из подпрограммы и все дальнейшие операции будут происходить так же, как было описано выше применительно к функции 1. Переменная получит новое значение, и программа завершится с кодом уровня ошибки (errorlevel) 000.

Исполнение функции 3 начинается со строки 2DC в секции 11. Буква диска запрашивается через прерывание INT 21\AH=19h, переводится в код ASCII и записывается в то место, где было прежнее значение переменной. Если там есть место еще хотя бы для одного знака, то к букве диска приписывается двоеточие.

Поскольку "текущий" диск заведомо существует, постольку коду уровня ошибки в функции 3 дана другая миссия – определение числа флоппи-дисководов в компьютере. Это нужно знать адаптивным процедурам загрузки, потому что в компьютерах с одним флоппи-дисководом все обращения к диску B: MS-DOS автоматически отсылает к единственному имеющемуся флоппи-дисководу A:. Задачу дополнительно осложняет возможность эмуляции флоппи-дисковода при загрузке компьютера с компакт-диска, не учитываемая в тех данных, которые предоставляет CMOS-память системы BIOS.

Для выяснения числа флоппи-дисководов команды в строках 2F1 – 30C обращаются к таблицам DDT (Disk Data Table, A.03-2). Только в них имеются сведения о принадлежности каждого логического диска конкретному физическому дисководу. По последовательности таблиц DDT подсчитывается число флоппи-дисководов, причем повторные отсылки к одному и тому же физическому дисководу не засчитываются. Потом в любом случае выполняется возврат из подпрограммы, и Reassign.com завершается исполнением заключительных команд из секции 3. Полученные значения кода уровня ошибки помогают правильно выбрать порядок тестирования дисков в ходе загрузки (пример – в разделе 9.09-02).

Более детальные пояснения к командам даны в комментариях, которые приведены в каждой ассемблируемой строке после знака точки с запятой.

В заключение полезно обратить внимание на модульную структуру программы Reassign.com. Каждая функция выполняется отдельной подпрограммой, не зависимой от остальных. Более того, список адресов подпрограмм в секции 4 содержит три незаполненных позиции (16D, 16F, 171). Если Вас не устраивает предлагаемый набор функций, Вы можете вписать туда адреса вызова Ваших собственных подпрограмм (вместе с тем надо будет изменить верхнюю границу допустимых номеров в строке 13B). Но прежде чем дело дойдет до модернизации программы Reassign.com, следует побеспокоиться о том, чтобы имеющийся ассемблерный текст был бы набран без ошибок. Он не настолько прост, как тексты предшествовавших программ из раздела 9.05, так что было бы опрометчиво сразу запускать на исполнение тот файл, который создаст отладчик Debug.exe в ответ на перенаправление ему ассемблерного текста.

Избежать нежелательных осложнений Вам помогут приведенные в следующем разделе 9.07 полезные советы по обнаружению ошибок в ассемблерных текстах и по тестированию "сырых" исполняемых файлов.

Примечание 1: если в нижележащем пространстве окружения имеется одноименная переменная с другим значением, то программа REASSIGN.COM переопределит и эту переменную тоже, причем ее новое значение может быть обрезано по длине прежнего. Повторного пользования одним и тем же именем в разных контекстах желательно избегать.

9.07 Рекомендации по проверке и испытанию программ

Обычная практика проверки программ, написанных на языке ассемблера, включает анализ листинга, исправление отмеченных там ошибок, и последующее пошаговое отлаживание программы. Исправлять ошибки необходимо в любом случае, но при этом рассчитывать на существенную помощь отладчика DEBUG.EXE не приходится, потому что он не выявляет не-синтаксические ошибки и не расставляет адреса по меткам автоматически, как это делают более совершенные ассемблеры.

При всех очевидных недостатках отладчика DEBUG.EXE встречаются ситуации, когда реальной альтернативы ему нет. Так бывает, когда отладка выполняется в условиях неопределенности или сопряжена с воздействием на системные установки BIOS и DOS. Поэтому умение пользоваться скромными возможностями отладчика DEBUG.EXE может оказаться очень полезным.

9.07-01 Анализ листинга.

По ходу исполнения команд, из которых состоит ассемблерный текст, отладчик DEBUG.EXE выводит на экран листинг. В нем отмечены выявленные отладчиком ошибки. Листинг смещается по экрану быстро, так что строку с отмеченной ошибкой легко пропустить. Чтобы рассмотреть выведенную на экран часть листинга внимательнее, Вы можете в любой момент приостановить исполнение, нажав на клавишу PAUSE. Нажатие на любую другую клавишу возобновит ассемблирование, но потом оно снова может быть временно остановлено в любой нужный момент. Такие манипуляции клавишами были достаточны при работе на старых компьютерах, но современные компьютеры смещают листинг по экрану слишком быстро. Поэтому бывает удобнее анализировать листинг после перенаправления его в отдельный файл, например, так:

```
Debug.exe < Reassign.scr > Listing.txt
```

Выведенный файл Listing.txt теперь можно распечатать или просмотреть с помощью любого текстового редактора. В качестве примера фрагмент листинга программы Reassign.com (из раздела 9.06) показан на рис. 6.

```
0C5B:0134       mov         BL,[DI]     ; 134        значения переменной
0C5B:0136       cmp         BL,31       ; 136 Нижняя граница: функция 1
0C5B:0139       jb          0151        ;*139 Выйдем, если значение меньше
0C5B:013B       cmp         BL,33       ; 13B Верхняя граница: функция 3
0C5B:013E       ja          0151        ;*13E Выйдем, если значение больше
0C5B:0140       shl         BL,1        ; 140 x2, т.к. адрес - это 2 байта
0C5B:0142       mov         BH,00       ; 142 Подготовим BX к переходу
0C5B:0144       call        [BX+005]    ;*144 Вызов основных функций
0C5B:0147       jz          0151        ;*148 ZR - вывод сообщения
0C5B:0149       jb          015F        ;*14A CY - errorlevel из DH
0C5B:014B             ;************* Секция 3: завершение программы.
0C5B:014B                ; 151 - переходы от 118, 11F, 12A, 139, 13E, 148
0C5B:014B                ; 15F - адрес перехода со строк 14A, 14F
0C5B:014B       call        01F9        ;*14C Подпрограмма копирования
0C5B:014E       imp         015F        ;*14F Переход к завершению
                ^ Error
0C5B:014E       mov         BX,DX       ;=151 DS:DX - адрес сообщения
0C5B:0150       mov         CX,[BX-02]  ; 153 CX = число знаков
0C5B:0153       mov         BX,0001     ; 156 BX = ссылка на STDOUT
0C5B:0156       mov         AH,40       ; 159 Вызов функции посылки
0C5B:0158       int         21          ; 15B      сообщения в STDOUT
0C5B:015A       mov         DH,F0       ; 15D Errorlevel = F0h
0C5B:015C       mov         AL,DH       ;=15F Восстановим errorlevel
0C5B:015E       mov         AH,4C       ; 161 Вызов функции
0C5B:0160       int         21          ; 163     завершения программы
```

Рис. 6

Каждая выявленная ошибка отмечена в листинге отдельной строкой с направленной вверх стрелочкой и словом "ошибка" (^Error). Стрелочка указывает на тот знак предыдущей строки, который отладчик не может интерпретировать. Стрелочка вверх на рис. 6 показывает на ошибку в предыдущей строке: действительно, там вместо команды "jmp" набрано "imp". Однако иногда стрелочка указывает на конец строки или на знак точки с запятой, с которого начинается комментарий. Так бывает, когда в спецификации ассемблерной команды отсутствует какой-то необходимый элемент. Выяснить "пропажу" помогут сведения из главы 7, где приведены все допустимые варианты спецификаций ассемблерных команд, которые "понятны" отладчику DEBUG.EXE.

В любом случае строка с выявленной ошибкой не ассемблируется, и вся последующая адресация становится неправильной. По этой причине выявленные отладчиком ошибки должны быть исправлены в первую очередь, и до их исправления какие-либо дальнейшие действия не имеют смысла.

Информативный листинг позволяет выявлять даже такие ошибки, которые отладчик не обнаруживает. Но для этого надо заранее постараться, чтобы сделать листинг достаточно информативным: вписать правильные значения адресов в комментарии к ассемблерному тексту, причем предпочтительно к каждой строке.

В разделах 9.05, 9.06, 9.08 и 9.10 почти каждая строка любого ассемблерного текста содержит комментарий, в котором сразу же после знака точки с запятой указано смещение соответствующей машинной команды. Его надо сверять со смещением в адресе, с которого начинается строка листинга. Если смещения неодинаковы, значит, в предыдущие строки вкралась ошибка. На рис. 6 смещения

различны начиная со строки 147, и потому ошибку следует искать в предшествующей строке 144. Действительно, сопоставление строки 144 на рис. 6 с исходным текстом выявляет ошибку: набрано "call [BX+005]" вместо "call [BX+0105]". Отладчик эту ошибку пропустил. Такие ошибки часто происходят из-за неправильного набора текстовых сообщений и данных. В любом случае ошибки, вызывающие расхождение смещений, нельзя оставлять без исправления.

Не выявляемые отладчиком ошибки также встречаются в адресах, содержащихся в составе некоторых команд. Если комментарий к строке начинается со знака звездочки, значит, в этой строке имеется команда с адресом назначения. Он должен быть точно равен фактическому смещению первого байта адресуемой команды или адресуемого блока данных. Для облегчения зрительного поиска строк назначения комментарии к ним начинаются со знака равенства. Помимо того, в комментариях под заголовками секций ассемблерного текста перечислены позиции всех команд, в которых упоминается каждый адрес назначения из данной секции.

Последним важным значением смещения является то, которое отмечает в листинге пустую строку, выводящую отладчик DEBUG.EXE из режима ассемблирования. В ассемблерном тексте из раздела 9.06 эта строка – 8-я с конца. Как правило, ассемблирование начинается с адреса 100h, и потому смещение пустой строки должно быть точно на 100h байт больше, чем полная длина файла компилируемой программы. Если нужно сохранить весь ассемблированный код в файле, то перед записью в файл надо ввести в регистр CX число, которое на 100h байт меньше смещения пустой строки в листинге. Если же Ваша цель – обнаружить вероятные ошибки, то надо сравнить смещение пустой строки в листинге с числом, вводимым в регистр CX. В частности, в ассемблерном тексте из раздела 9.06, в третьей строке с конца, в регистр CX вводится число 03E0h. Следовательно, в листинге смещение пустой строки должно быть 04E0h: это будет свидетельством отсутствия смещений адресации в сформированной последовательности команд.

9.07-02 Интерактивное отлаживание при составлении программ.

Когда нужно что-либо уточнить, тогда бывает нетрудно набрать вручную 5 – 7 строк ассемблерного текста и тут же их исполнить. Длинные последовательности команд набирать вручную не удается, их приходится набирать отдельно с помощью программы редактирования, а потом посылать полученные файлы отладчику через перенаправление ввода. Но когда ввод перенаправлен, отладчик перестает воспринимать команды с клавиатуры, и преимущества интерактивного взаимодействия с пользователем пропадают. Это обычно принимают как должное, потому что ничего другого не дано. Тем не менее такое мнение ошибочно.

Через перенаправление ввода отладчик готов принять и исполнить команды, которые отменят это самое перенаправление и вернут ему способность

интерактивного взаимодействия. Требуемая последовательность команд может выглядеть, например, так:

```
CS:                         ; Восстановление содержания JFT
mov word ptr [0018],0101    ;          в PSP отлаживаемой программы
mov          AH,62          ; Запрос сегментного адреса
int          21             ;               PSP отладчика DEBUG.EXE
mov          DS,BX          ; Сегментный адрес - в регистр DS
mov word ptr [0018],0101    ; Восстановление JFT в PSP отладчика
int          20             ; Возврат к командной строке отладчика
```

Первые две командные строки восстанавливают ссылки на первый блок описания таблицы SFT в ячейках таблицы JFT (примечание 3 к A.07-1) со смещениями CS:0018h и CS:0019h. Эти ячейки относятся к каналам STDIN и STDOUT, а первый блок описания таблицы SFT (A.01-4) активизирует драйвер консоли (CON), обслуживающий ввод с клавиатуры и вывод сообщений на дисплей. Следовательно, выполненная подстановка восстановит нормальное взаимодействие команд отлаживаемой программы с клавиатурой и дисплеем.

Однако отладчик DEBUG.EXE не руководствуется ссылками, вносимыми в ту копию своего префикса PSP, которую он создает для отлаживаемой программы. Для отладчика изменения должны быть внесены не в копию, а в его собственный PSP (A.07-1), сформированный интерпретатором COMMAND.COM. Сегментный адрес этого PSP возвращает в регистре BX функция INT 21/AH=62h (8.02-73), вызываемая в 4-й строке. В 5-й и 6-й командных строках найденный сегментный адрес используется для того, чтобы выполнить аналогичную подстановку в собственную таблицу JFT отладчика DEBUG.EXE. Тем самым оказываются подготовлены все условия для восстановления интерактивного взаимодействия отладчика с пользователем. Возврат из процесса исполнения ассемблированных команд выполнен без обычно используемой команды RET (не так, как в примерах из раздела 9.02), а посредством обработчика прерывания INT 20 (8.02-01). В отличие от команды RET, этот обработчик сохраняет положение вершины стека, что бывает полезно для продолжения отладки испытуемой программы.

Во избежание путаницы с командами отлаживаемой программы показанную выше группу команд целесообразно приписывать в форме машинных кодов к концу посылаемого файла. Адрес размещения этих кодов в памяти может быть любым, лишь бы он был не занят. Допустим, что ячейки за смещением F00h заведомо свободны. Тогда строки, которыми нужно заменить обычные последние команды "w" и "q" в посылаемом файле, будут иметь следующий вид:

```
e F00 2E C7 06 18 00 01 01 B4 62 CD 21 8E DB C7 06 18 00 01 01 CD 20
g=F00
```

Если послать через перенаправление ввода командный файл, заканчивающийся этими двумя строками, то будут выполнены все содержащиеся в нем команды, но исполнение завершится выходом не в командную строку интерпретатора Command.com, а в командную строку отладчика Debug.exe. Очевидное преимущество такого завершения состоит в том, что теперь не нужно заранее определять длину того файла, который должен быть сформирован отладчиком. Фактическая длина любого ассемблированного кода будет показана в листинге, и ее можно ввести в регистр CX с клавиатуры. Еще одно преимущество в том, что снимаются ограничения (примечание 1 к 6.05) на отладку команд, обращающихся к каналам STDIN и STDOUT. Наконец, отпадает необходимость формировать заново исполняемый файл отлаживаемой программы после каждого внесенного исправления: все коррективы до окончательного завершения отладки теперь достаточно вносить в исходный ассемблерный текст отлаживаемой программы.

Перечисленные преимущества особенно полезны, когда программу формируют по частям, последовательно дополняя уже отлаженный ассемблерный текст новыми группами подлежащих отладке команд. Последовательное отлаживание модулей способствует своевременному обнаружению не-синтаксических ошибок и снижает трудоемкость исправления их последствий. Преимущества последовательной модульной компоновки в полной мере проявили себя при написании тех программ, которые представлены готовыми в разделах 9.06, 9.08 и 9.10.

9.07-03 Отлаживание с использованием batch-файла.

Когда все ошибки из листинга исправлены, и последний прогон ассемблирования новых ошибок не выявил, тогда, значит, настала пора разбираться со всей остальной массой ошибок в ходе отлаживания и тестирования.

Полученный при последнем прогоне ассемблирования файл программы, как правило, еще слишком "сырой", запускать его на исполнение опасно. Для отлаживания его можно передать отладчику DEBUG.EXE прямо из командной строки, как показано во вводной статье к разделу 6.05. Но лучше начинать отлаживание с помощью специально подготовленного batch-файла. Во-первых, отлаживание приходится повторять, и batch-файл избавляет от необходимости набирать строки каждый раз заново. Во-вторых, batch-файл обеспечивает общее пространство окружения. В-третьих, в batch-файле можно задать дополнительные процедуры, которые сделают результат более информативным.

Общая композиция испытательного batch-файла включает подготовительную часть, основную команду отлаживания проверяемой программы, и заключительную часть с анализом и выведением результатов на экран дисплея. При конкретном воплощении каждой части, конечно, необходимо учитывать

особенности проверяемой программы. Ниже приведен пример batch-файла, написанного для отлаживания функции 2 программы Reassign.com из раздела 9.06.

```
@echo off
set input=22222
echo Original value of input=%input%
Debug.exe Reassign.com input < Reassign.scr
set E=
set Z=00
set N=
:ErrCycle
for %%Y in (0 1 2 %N%) do if errorlevel %E%%%Y%Z% set E=%E%%%Y
if %Z%"==" goto OUT
if %Z%"==0" set Z=
if %Z%"==00" set Z=0
set N=3 4 5
if not %E%"==2" if not %E%"==25" set N=%N% 6 7 8 9
goto ErrCycle
:OUT
echo Errorlevel is %E%
echo New value of input=%input%
pause
```

Первые три строки в предложенном batch-файле составляют подготовительную часть, а 4-я строка запускает процесс отлаживания. Чтобы не потерять ориентацию в последовательности проверяемых машинных команд, полезно заранее заготовить распечатку ассемблерного текста отлаживаемого файла с комментариями.

Вызываемый в четвертой строке отладчик DEBUG.EXE получает группу параметров из командной строки и, кроме того, командный файл через перенаправление ввода. Здесь основная роль параметров командной строки – их участие в заполнении префикса сегмента (PSP) отлаживаемой программы. В простейшем случае файл Reassign.com может быть вообще пуст, но важно, чтобы его имя вместе с последующими параметрами, если они имеются, было бы вписано в соответствующие поля PSP (A.07-1), обеспечив таким образом формирование должного окружения отлаживаемой программы. Если файл Reassign.com не пуст, то содержащийся в нем код будет выложен отладчиком в памяти начиная с адреса CS:0100, и этим целесообразно пользоваться при отладке больших программ.

Перенаправление ввода в четвертой строке заставит отладчика ассемблировать команды из строк файла Reassign.scr. Получающийся машинный код будет выложен в память начиная с любого указанного адреса, причем как поверх того

кода, который изначально скопирован из файла Reassign.com, так и в дополнение к нему. Поэтому файл Reassign.scr может содержать только неотлаженную часть ассемблерного текста составляемой программы. Это не относится к предлагаемым здесь программам: готовый ассемблерный текст нет смысла делить. Однако в любом случае при необходимости отлаживания ассемблерного текста, посылаемого через перенаправление ввода, он должен заканчиваться теми двумя строками, которые предложены в разделе 9.07-02 и которые по окончании ассемблирования обеспечат переход отладчика к приему команд с клавиатуры.

Ожидая ввода команд с клавиатуры, отладчик DEBUG.EXE покажет свое приглашение – мигающий знак подчеркивания – и предоставит пользователю действовать дальше по своему усмотрению. Лучше начинать отлаживание программы по частям (или по подпрограммам) с помощью команд "G" (= Go, 6.05-07) и "P" (= Proceed, 6.05-14). Проверка по частям менее трудоемка, чем пошаговая. Полезно записывать состояния значимых флагов, регистров и ячеек памяти в критически важных точках, например, в моменты выхода из подпрограмм. Эти сведения упростят поиск причин возможных сбоев. Если будет выявлена ошибка в какой-либо части, то ее нужно будет потом более детально обследовать по группам машинных команд, а затем и по отдельным шагам.

Если Вы сочтете целесообразным закончить сеанс отлаживания, то достаточно набрать команду "Q" (= Quit) и нажать клавишу ENTER. При таком завершении отладчик всегда оставляет нулевой код уровня ошибки. Но когда важно правильно передать тот код уровня ошибки (errorlevel), который оставляет отлаживаемая программа, тогда надо завершать сеанс вызовом обработчика прерывания INT 21\AH=4Ch (8.02-55), то есть так же, как отлаживаемую программу следовало бы закрывать при исполнении вне "оболочки" отладчика DEBUG.EXE.

По окончании каждого сеанса отлаживания исполняются команды в завершающей части испытательного batch-файла, призванные показать последствия сеанса. Результаты исполнения программы Reassign.com выражаются кодом уровня ошибки и изменением значения одной из переменных окружения. Иногда для "отлова" кода уровня ошибки исполняют тестируемую программу в "оболочке" командного интерпретатора COMMAND.COM, запущенного с параметром /Z (6.04), однако из такой "оболочки" значения переменных окружения не передаются. Поэтому здесь испытательный batch-файл в строках 5 – 16 дополнен процедурой, которая определяет оставленное значение кода уровня ошибки. Потом команды в строках 17 – 18 выводят на экран значение кода уровня ошибки и той переменной, которую должна была изменить программа Reassign.com. Последняя строка с командой PAUSE служит только для того, чтобы выведенные на экран значения не были бы сразу же скрыты под панелями файл-менеджера.

Для проверки другой функции испытуемой программы нужно изменить параметры, задаваемые в подготовительной части испытательного batch-файла, а

иногда и в основной строке вызова испытуемой программы. Это можно делать через формальные параметры batch-файла (2.03-03). В показанном выше примере формальные параметры batch-файла для простоты не задействованы, требуемые изменения предполагается вводить прямо в текст с помощью программы редактирования. После сохранения batch-файла с внесенными изменениями он снова готов к тестированию очередной функции испытуемой программы.

Вообще любая программа должна быть подвергнута трем сериям испытаний. Испытания первой серии проверяют все функции испытуемой программы в нормальных условиях. Испытания второй серии проверяют реакцию испытуемой программы на неординарные условия: неверные спецификации, отсутствие обязательных параметров, недопустимые значения исходных данных. Наконец, испытания третьей серии проверяют работоспособность программы в пределах того ограниченного разнообразия конфигураций компьютеров, которое предопределено ее назначением. Конкретный состав тестов, конечно, сильно зависит от характера задачи и масштаба притязаний, но принцип одинаков для всех программ, даже таких небольших, как предложенная в разделе 9.06 программа Reassign.com. Когда набранная Вами программа успешно пройдет все три серии испытаний, только тогда Вашу работу над ней можно будет считать законченной.

9.08 Давайте попробуем написать драйвер

Внутри незнакомых компьютеров становится намного уютнее, когда RAM-диск со знакомой, загруженной Вами версией DOS всегда имеет одно и то же буквенное обозначение, например, букву R:. Это можно сделать с помощью драйвера, который заставит DOS выделить буквы для несуществующих, фиктивных дисков так, чтобы следующая выделяемая RAM-диску буква была бы именно той, которую Вы хотите назначить.

Известны по крайней мере 3 попытки написать такой драйвер. Не все получившиеся драйверы одинаково удачны, но те, которые удачны, не являются свободно распространяемыми, и их код не раскрыт. В данном разделе представлен ассемблерный текст драйвера, решающего ту же задачу, но не копирующего предыдущие попытки ее решения. Текст написан полностью заново специально для того, чтобы Вы с помощью отладчика DEBUG.EXE могли бы сделать себе такой драйвер и между делом освоили бы основные особенности построения драйверов в DOS. Предлагаемый драйвер очень маленький: всего 503 байта. Для определенности давайте назовем его SkipDsk.sys.

Отладчик DEBUG.EXE создаст файл драйвера SkipDsk.sys в ходе исполнения команд файла SkipDsk.scr, который должен быть послан отладчику через перенаправление ввода, например, так:

```
Debug.exe < SkipDsk.scr
```

Файл SkipDsk.scr надо будет заранее набрать с помощью текстового редактора, как описано во вводной статье к главе 9, по представленному ниже тексту. Сопровождающие текст комментарии набирать не обязательно, но числовые значения смещений справа от знаков точки с запятой все же лучше не опускать для упрощения последующей проверки и отладки.

```
a 0000
      ;************* Секция 1: заголовок драйвера
         ; 000 - место для адреса следующего драйвера
dw           FFFF,FFFF
         ; 004 - атрибуты драйвера (A.05-2)
dw           2202
         ; 006 - адрес входа в программу стратегии
dw           0031
         ; 008 - адрес входа в программу прерывания
dw           006E
         ; 00A - число дисков, обращения от 058, 113
db           00
         ; 00B - идентификатор (7 байтов)
db           53,6B,69,70,44,73,6B
      ;************* Секция 2: данные
         ; 012 - адрес запроса, обращения от 032, 073
         ; 014 - сегмент запроса, обращение от 037
dw           0000,0000
         ; 016 - блок BPB (A.03-4), указан в 093 - 0C1
db           00,02,FF,01,00,01,40,00,00,22,F0,01,00
db           12,00,01,00,01,00,00,00,00,00,00,00
         ; 02F - метка, от строк 03D, 04D, 052, 087, 0D8
dw           FEFE
      ;****** Секция 3: резидентная программа стратегии
         ; 031 - указан в слове 006
CS:                       ;=031 Точка входа
mov          [0012],BX    ;*032 Запомним смещение блока
CS:                       ; 036    данных запроса (A.05-3)
mov          [0014],ES    ;*037 Запомним сегментный адрес
retf                      ; 03B    и вернем управление DOS
      ;******** Секция 4: отклик на "проверку носителя"
         ; 03C - адрес перехода со строки 084
```

```
        ; 05C - адрес перехода со строки 06C
CS:                         ;=03C Подготовлено ли в CS:002Fh
cmp byte ptr  [002F],FE     ; 03D       смещение записи CDS?
jnb           008E          ;*042 Возврат назад, если нет
mov           AH,52         ; 044 Запросим "Список Списков"
int           21            ; 046 Адрес возвращен в ES:BX
ES:                         ; 048 Загрузим адрес 1-й записи
les           BX,[BX+16]    ; 049      CDS в регистры ES:BX
CS:                         ; 04C Вычислим смещение CDS
add           BX,[002F]     ;*04D     1-го фиктивного диска
CS:                         ; 051 FFh в ячейке 002Fh значит
mov byte ptr  [002F],FF     ;*052   что проверка выполнялась
CS:                         ; 057 Считаем число записей CDS
mov           AH,[000A]     ; 058       для фиктивных дисков
cmp           AH,01         ;=05C Сравним это число с 01h
jb            008E          ;*05F Выходим, если дисков нет
ES:                         ; 061 Внесем недействительные
mov word ptr  [BX+43],0000  ; 062      атрибуты в запись CDS
add           BX,0058       ; 067 Найдем следующую CDS
dec           AH            ; 06A Число оставшихся циклов
jmp           005C          ;*06C Повтор для следующей CDS
     ;******** Секция 5: программа прерывания
        ; 06E - указан в слове 008
        ; 08E - адрес перехода от 042, 05F, 082, 13C
pushf                       ;=06E Точка входа
push          ES            ; 06F Сохраним состояния
push          BX            ; 070        флагов и регистров
push          AX            ; 071
CS:                         ; 072 Загрузим адрес данных
les           BX,[0012]     ;*073           запроса в ES:BX
ES:                         ; 077 Зададим статус
mov word ptr  [BX+03],8007  ; 078     "непригодный носитель"
ES:                         ; 07D Проверим код операции
cmp byte ptr  [BX+02],01    ; 07E     в блоке данных запроса
ja            008E          ;*082 Выходим, если больше 01h
jz            003C          ;*084 Если 01h, то - в секцию 4
CS:                         ; 086 Если меньше 01h, то эта
cmp byte ptr  [002F],FE     ;*087    инициализация - первая?
jz            00C3          ;*08C Если да - к инициализации
pop           AX            ;=08E Восстановим
pop           BX            ; 08F     состояния регистров
pop           ES            ; 090
popf                        ; 091 Восстановим флаги
retf                        ; 092   и вернем управление DOS
```

```
      ;******** Секция 6: адреса таблицы BPB для дисков
         ; 093 - упомянут в строках 11E, 12A
dw           0016,0016,0016,0016,0016,0016,0016,0016
dw           0016,0016,0016,0016,0016,0016,0016,0016
dw           0016,0016,0016,0016,0016,0016,0016,0016
      ;********** Секция 7: нерезидентная часть драйвера
         ; 0C3 - адрес перехода от строки 08C
push         DX            ;=0C3 Сохраним
push         DS            ; 0C4       состояния регистров
push         SI            ; 0C5
cld                        ; 0C6 Счет - на приращение
ES:                        ; 0C7 Занесем в AH номер диска,
mov          AH,[BX+16]    ; 0C8     который предлагает DOS
mov          AL,41         ; 0CB Преобразуем номер диска в
add          AL,AH         ; 0CD           букву диска в AL
CS:                        ; 0CF Заменим этой буквой диска
mov          [01E0],AL     ;*0D0        букву в сообщении 2
mov          AL,58         ; 0D3 Найдем смещение записи CDS
mul          AH            ; 0D5   первого фиктивного диска
CS:                        ; 0D7 Запишем смещение
mov          [002F],AX     ;*0D8        в ячейку CS:[002F]
      ;******* Секция 8: считывание из командной строки
         ; 0E2 - адрес перехода от строки 0E7
         ; 0E9 - адрес перехода от строки 0EE
ES:                        ; 0DB Указатель на аргументы
lds          SI,[BX+12]    ; 0DC   командной строки в DS:SI
mov          DX,013F       ;*0DF Смещение 1-го сообщения
lodsb                      ;=0E2 Загрузим в AL знак и
cmp          AL,20         ; 0E3     организуем цикл поиска
jb           00F8          ;*0E5      первого пробела после
ja           00E2          ;*0E7             имени драйвера
lodsb                      ;=0E9 Загрузим в AL знак и
cmp          AL,20         ; 0EA     организуем цикл поиска
jb           00F8          ;*0EC      первой значащей буквы
jz           00E9          ;*0EE      после первого пробела
cmp          AL,43         ; 0F0 Буква меньше чем C: ?
jb           00F8          ;*0F2 Если да, то - в секцию 9
cmp          AL,59         ; 0F4 Буква меньше чем Y: ?
jbe          0102          ;*0F6 Если да, то - в секцию 10
      ;********* Секция 9: выведение на экран сообщений
         ; 0F8 - переходы от строк 0E5, 0EC, 0F2, 10C
push         CS            ;=0F8 Подготовим DS = CS для
pop          DS            ; 0F9        адресации сообщений
mov          AH,09         ; 0FA Вызовем функцию выведения
```

```
int          21            ; 0FC         сообщений на экран
mov          AL,00         ; 0FE После ошибки - 0 дисков
jmp          010E          ;*100 Готовимся к возврату в DOS
      ;************* Секция 10: размер резидентной части
         ; 102 - адрес перехода со строки 0F6
         ; 10E - адрес перехода со строки 100
inc          AL            ;=102 Получим букву RAM-диска
mov          DX,01C3       ; 104 Смещение 2-го сообщения
CS:                        ; 107 Рассчитаем число дисков
sub          AL,[01E0]     ;*108     которое надо запросить
jb           00F8          ;*10C Если меньше 0, то ошибка
ES:                        ;=10E Запишем число дисков
mov          [BX+0D],AL    ; 10F      в блок данных запроса
CS:                        ; 112 Запишем число дисков
mov          [000A],AL     ; 113       в заголовок драйвера
mov          AH,00         ; 116 Вычислим смещение для
cmp          AL,00         ; 118    указателя на 1-й байт
jz           0121          ;*11A    после резидентной части
shl          AX,1          ; 11C        драйвера по формуле
add          AX,0093       ;*11E              (2*AX + 093h)
      ;************* Секция 11: заполнение блока запроса
         ; 121 - адрес перехода со строки 11A
ES:                        ;=121 Запишем размер резидентной
mov          [BX+0E],AX    ; 122       части в блок запроса
ES:                        ; 125 Припишем сегмент к размеру
mov          [BX+10],CS    ; 126          резидентной части
ES:                        ; 129 Смещение для указателей на
mov word ptr [BX+12],0093  ;*12A    на BPB фиктивных дисков
ES:                        ; 12F Припишем сегмент к
mov          [BX+14],CS    ; 130          указателям на BPB
ES:                        ; 133 Объявим успех в статусе
mov word ptr [BX+03],0100  ; 134     в блоке данных запроса
pop          SI            ; 139 Восстановим
pop          DS            ; 13A        состояния регистров
pop          DX            ; 13B
jmp          008E          ;*13C Перейдем для возврата DOS
      ;*********** Секция 12: сообщения об ошибках
         ; 13F - 1-е сообщение, указано в строке 0DF
db           0D 0A "SkipDsk: diskletter isn" 27 "t found"
db           20 "or out of range" 0D 0A
db           "Example:" 0A "device=A:\SkipDsk.sys Q:"
db           0D 0A 09 09 09 "Q: - diskletter (C: - Y:)"
db           20 "to be skipped" 0D 0A 0A 24
         ; 1C3 - 2-е сообщение, указано в строке 104
```

```
db            0D 0A "SkipDsk: diskletters below" 20
        ; 1E0 - заменяемая буква диска, от 0D0, 108
db            "A: are assigned yet" 0D 0A 0A 24
        ; 1F7 - контрольная точка, конец драйвера

m 0000 L01F7 0100
n SkipDsk.sys
rBX
0000
rCX
01F7
w
q
```

В отличие от обычных исполняемых файлов, драйверы загружаются начиная со смещения 0000h, то есть без резервирования места для префикса сегмента программы (PSP, A.07-1). Именно поэтому приведенный выше текст начат с команды "a 0000", а не как обычно с "a 100". Если стартовый адрес ассемблирования указать неправильно, то неизбежно возникает путаница при расчете смещений для ссылок и адресов переходов.

Еще одно специфическое свойство драйверов для DOS состоит в том, что у них две точки входа, причем ни одна из них не совпадает со стартовым адресом размещения кода драйвера в памяти. Первая точка входа относится к программе стратегии, которая служит для приема блока данных запроса и для инициализации исполнения этого запроса тем физическим устройством, которое драйвер обслуживает. Вторая точка входа относится к так называемой программе прерывания, используемой для сбора результатов и формирования ответа на каждый ранее выданный запрос. Во время между вызовами упомянутых программ, входящих в состав драйвера, операционная система и соответствующие физические устройства работают независимо и делают каждый свое дело. Адреса точек входа должны быть объявлены в строго определенном месте в заголовке драйвера: для программы стратегии - в слове со смещением 0006h, для программы прерывания - в слове со смещением 0008h. Эти и другие обязательные элементы драйверного заголовка перечислены в таблице A.05-1.

Резидентная часть программы стратегии у драйвера SkipDsk.sys составляет секцию 3 приведенного выше ассемблерного текста. Она очень проста и включает лишь операции записи сегментного адреса и смещения блока данных запроса (A.05-3) в приготовленное место в секции данных. Нерезидентной части у данной программы стратегии вообще нет.

Все действия по подготовке ответов на запросы DOS исполняются программой прерывания, которая начинается в секции 5. Сначала в стеке сохраняются

состояния флагов и регистров процессора: это тоже является специфической обязанностью каждого драйвера. Затем проверяется код запрашиваемой операции. Поскольку драйвер SkipDsk.sys "обслуживает" только фиктивные диски, постольку операции с кодом больше 01h должны всегда кончаться возвратом в DOS со статусом "непригодный носитель". Но операции с кодами 00h и 01h вызывают исполнение команд, показанных в секциях 7 - 11 или в секции 4.

Первый запрос DOS к драйверу всегда содержит команду инициализации с кодом 00h. Поскольку инициализация запрашивается только один раз сразу после загрузки драйвера, постольку соответствующие секции 7 – 11 находятся за пределами резидентной части драйвера. Секция 7 подготавливает данные для последующего использования, секция 8 считывает из командной строки букву диска – последнюю, которую надо пропустить, секция 9 всегда стоит наготове для выведения сообщений об ошибках на экран. Тексты сообщений об ошибках заготовлены заранее в секции 12. В секции 10 рассчитывается число фиктивных дисков, которые надо создать, а также размер резидентной части драйвера. В секции 11 блок данных запроса заполняется данными, которые надо возвратить операционной системе. В соответствии с этими данными DOS дополнит свои таблицы DPB (A.03-1) и CDS (A.03-3) так, что буквы фиктивных дисков, включая ту букву, которая заявлена в командной строке, будут считаться занятыми. Когда позднее предложение к инициализации будет послано драйверу RAM-диска, ему будет выделена следующая свободная буква, которая заранее предопределена драйвером SkipDsk.sys.

В момент назначения буквы RAM-диску те фиктивные диски, которые "выдуманы" драйвером SkipDsk.sys, операционная система должна считать полноценными и действующими. Однако потом такой статус фиктивных дисков становится помехой для назначения буквенных обозначений сетевым и IFS-дискам. Список доступных дисков оказывается забит "мусором", в котором трудно разобраться. Поэтому перед драйвером SkipDsk.sys ставится следующая задача: аннулировать действительный статус фиктивных дисков. Чтобы решить эту задачу, драйвер SkipDsk.sys должен быть вызван на исполнение хотя бы еще один раз.

Повторную активизацию драйвера SkipDsk.sys удобно произвести при поступлении от DOS запроса на операцию "проверка носителя" (код 01h), потому что эта операция предшествует всем другим запросам, обращенным к дисководам на сменных дисках, а также потому что проверка носителя будет автоматически запрошена после окончания интерпретации строк файла CONFIG.SYS. В ответ на этот запрос драйвер SkipDsk.sys исполняет команды из своей резидентной секции 4. Они определяют адрес CDS-записи, соответствующей первому фиктивному диску, а затем вписывают во все CDS-записи фиктивных дисков слово атрибутов 0000h. Логические диски с таким словом атрибутов считаются недействительными, они не отображаются в списке доступных дисков, и соответствующие им буквенные обозначения могут быть задействованы другими драйверами.

Когда прежняя буква фиктивного диска предоставлена другому драйверу, тогда снова вписывать слово атрибутов 0000h в ту же CDS-запись уже нельзя. Во избежание этого одна из команд секции 4 заносит в ячейку со смещением 02Fh метку FFh. Наличие этой метки проверяется при каждом вызове программы прерывания драйвера SkipDsk.sys, чтобы предотвратить многократное исполнение как инициализации, так и цикла вписывания атрибутов. Благодаря тому любые повторные обращения не нарушат правильное функционирование других драйверов, получивших в свое распоряжение прежние буквы фиктивных дисков.

Более детальные объяснения роли отдельных команд приведены в комментариях к строкам файла SkipDsk.scr.

Как в большинстве других ассемблерных текстов, в файле SkipDsk.scr необходимо обратить внимание на пустую строку, девятую от конца файла. Она вызывает выход отладчика DEBUG.EXE из режима ассемблирования и потому обязательно должна быть в набранном Вами тексте. Следующая строка с командой "m" (6.05-11) смещает весь транслированный машинный код на 100h байт дальше, освобождая тем самым область PSP. Только после этого можно командой "n" (6.05-12) назначить имя для создаваемого драйвера, потому что оно записывается как раз в область PSP. В заключительных шести строках файла SkipDsk.scr длина создаваемого драйвера вносится в регистры BX:CX, после чего драйвер SkipDsk.sys записывается на диск командой "w" (6.05-19).

Сформированный драйвер SkipDsk.sys нельзя считать пригодным к использованию без тщательной проверки выводимого отладчиком листинга, о которой написано в разделе 9.07-01. Последнее смещение в левой колонке листинга выводится в ответ на пустую строку, девятую от конца ассемблерного текста. Поскольку ассемблирование было начато с адреса 0000, постольку последнее значение смещения должно быть 01F7h, то есть точно равным полной длине создаваемого драйвера, указываемой в 8-й и 3-й строках от конца ассемблерного текста. Если листинг не выявляет никаких ошибок, то созданный драйвер SkipDsk.sys имеет шансы успешно пройти тестирование.

При проведении испытаний нужно принять во внимание, что ради достижения простоты и малого размера драйвера SkipDsk.sys его способности намеренно ограничены. Он не допускает наличия косой черты перед буквой диска в командной строке, не допускает замены пробелов в командной строке знаками табуляции (09h). Он не может перераспределять буквы дисков, которые уже назначены. Наконец, он не сможет работать с версиями DOS ниже MS-DOS4.0.

В практике пользования драйвером SkipDsk.sys переназначение букв, выделенных фиктивным дискам, может потребоваться до завершения интерпретации файла Config.sys. В таких случаях запрос операции "проверка носителя" надо спровоцировать намеренно, например, как это сделано в третьей строке следующего фрагмента файла Config.sys:

```
devicehigh=\DOS\DRV\SkipDsk.sys Q:
devicehigh=\DOS\DRV\Ramdrive.sys 2400 /E
install=\Command.com nul /low /f /c vol Q:
```

В приведенном примере буква Q: будет выделена последнему фиктивному диску, а RAM-диск получит следующую букву R:. В последней строке обращение командного интерпретатора к фиктивному диску Q: заставит DOS послать драйверу SkipDsk.sys запрос операции "проверка носителя". С этого момента все буквы, выделенные фиктивным дискам, станут свободными и могут быть повторно выделены другим драйверам, которые загружаются позднее командами "Install" или "Installhigh". Еще один подобный пример использования драйвера SkipDsk.sys показан в разделе 9.09-01.

9.09 Адаптивная загрузка на незнакомый компьютер.

Предлагаемые в этом разделе варианты конфигурационных файлов CONFIG.SYS и AUTOEXEC.BAT, помимо обычной загрузки MS-DOS7, позволяют перебазировать действующую MS-DOS7 на RAM-диск или на физический диск, а также осуществить загрузку без драйверов, применяемую для тестирования памяти компьютера и для смены прошивок микросхем памяти BIOS.

Главная особенность предлагаемых здесь конфигурационных файлов состоит в том, что выбор лучшего из альтернативных вариантов загрузки пользователь сможет сделать не "вслепую", а на основании результатов тестирования. Если Вы предпочитаете перебазирование MS-DOS7 на RAM-диск, то его объем будет определен по результатам исследования доступной XMS-памяти. Если Вы предпочтете перебазирование MS-DOS7 на физический диск, то заранее узнаете, какие диски имеются в компьютере, их объем и процент занятого дискового пространства на каждом из записываемых дисков.

Проведение тестирования в то время, когда загрузка операционной системы еще не закончена, неизбежно сопряжено с рядом ограничений. Из-за них здесь нельзя воспроизвести те же процедуры, которые реализованы в файле DISK.BAT (9.03-02). По той же причине не удается обойтись без привлечения драйверов и программ, не входящих в комплект поставки MS-DOS7. С другой стороны, для тестирования в процессе загрузки можно принять ряд упрощающих допущений. В частности, допустимо считать, что исследование проводится с загрузочного диска, что структура каталогов на этом диске известна, и что известно, какие операции на каждом этапе загрузки DOS еще не осуществлены. Благодаря последнему допущению, кстати, нет необходимости включать в перечень исследуемых дисков RAM-диски и дисководы CD-ROM.

Предлагаемые здесь конфигурационные файлы предназначены для обеспечения загрузки MS-DOS7 со сменного носителя на AT-совместимые компьютеры с

заранее неизвестным составом оборудования. Предоставляемая возможность адаптировать процесс загрузки к оперативно выясняемым особенностям каждого конкретного компьютера экономит много времени при проведении сложных восстановительных работ.

9.09-01 Файл CONFIG.SYS: подготовка адаптивной загрузки.

Предлагаемая здесь версия файла CONFIG.SYS предоставляет выбор из пяти вариантов загрузки MS-DOS7, показанных ниже в тексте файла в разделе [menu]. Некоторые из вариантов включают исполнение таких операций, которые не могут быть реализованы стандартными средствами MS-DOS7. В частности, создание RAM-диска адаптируемого объема обеспечивается свободно распространяемым драйвером TDSK.EXE (5.05-02) версии 2.42. Точных аналогов этот драйвер не имеет. Кроме того, привлечен драйвер SKIPDSK.SYS (9.08), который Вы могли бы сотворить самостоятельно. Потенциально он может быть заменен платным драйвером JDRIVE.SYS фирмы JAMSOFT.

Справочные материалы по остальным примененным драйверам приведены в главе 5. Эти драйверы либо входят в поставку Windows-95/98, либо имеют близкие аналоги, и их отбор не столь критичен. Конечно, каждая замена должна сопровождаться корректировкой параметров командной строки так, как того требует конкретный драйвер.

Ниже приведен текст предлагаемой версии файла CONFIG.SYS.

```
[menu]
numlock off
menuitem=L047, User-configurable real mode
menuitem=L048, User-configurable V86 mode
menuitem=L029, Quick boot in real mode
menuitem=L031, Quick boot in V86 mode
menuitem=L104, Real mode without drivers
menudefault=L029,20

[L047]
device=\DOS\DRV\Himem.sys /v
device=\DOS\DRV\Umbpci.sys
include=L104
country=007,866,\DOS\DRV\Country.sys
devicehigh=\DOS\DRV\Dblbuff.sys
devicehigh=\DOS\DRV\Ifshlp.sys
devicehigh=\DOS\DRV\Setver.exe
devicehigh=\DOS\DRV\Dvs.sys /D:CD001
devicehigh=\DOS\DRV\Skipdsk.sys Q:
```

```
device=\DOS\DRV\Tdsk.exe 0
install=\Command.com nul /low /f /c vol Q:
installhigh=\DOS\DRV\Shsucdx.com /D:?CD001 /L:N /~+ /R /Q
installhigh=\DOS\DRV\Ctmouse.exe
installhigh=\DOS\DRV\Keyrus.exe

[L048]
device=\DOS\DRV\Himem.sys /v
device=\DOS\DRV\Jemm386.exe X=TEST noems verbose
include=L104
country=007,866,\DOS\DRV\Country.sys
devicehigh=\DOS\DRV\Dblbuff.sys
devicehigh=\DOS\DRV\Ifshlp.sys
devicehigh=\DOS\DRV\Setver.exe
devicehigh=\DOS\DRV\Dvs.sys /D:CD001
devicehigh=\DOS\DRV\Skipdsk.sys Q:
device=\DOS\DRV\Tdsk.exe 0
install=\Command.com nul /low /f /c vol Q:
installhigh=\DOS\DRV\Shsucdx.com /D:?CD001 /L:N /~+ /R /Q
installhigh=\DOS\DRV\Ctmouse.exe
installhigh=\DOS\DRV\Keyrus.exe

[L029]
include=L047

[L031]
include=L048

[L104]
accdate C- D- E- F-
dos=high,umb,noauto
buffershigh=30,0
fileshigh=30
lastdrivehigh=Z
fcbshigh=1,0
stackshigh=9,256

[common]
shell=\Command.com \ /E:2016 /L:511 /U:255 /p
```

Текст файла CONFIG.SYS начинается с секции [menu], содержащей 5 пунктов, каждый со своим заголовком и именем от L029 до L104. Эти имена совпадают с названиями секций файла CONFIG.SYS, которые должны быть интерпретированы

при выборе одноименного пункта меню. Помимо того, имя избранного пункта меню автоматически становится значением переменной окружения CONFIG; ее значение используется впоследствии для перехода к соответствующей части другого конфигурационного файла – AUTOEXEC.BAT. С этой целью имена пунктов меню выбраны такими же, как имена соответствующих меток в файле AUTOEXEC.BAT (9.09-02).

При выборе пунктов меню L029, L031, L047 и L048 загружаются почти одинаковые комплекты драйверов, различающиеся только строкой 2 в секциях [L047] и [L048]. Драйвер UMBPCI.SYS (5.04-04) в секции [L047] обеспечивает доступ за пределы обыкновенной памяти, сохраняя работу центрального процессора в реальном режиме, тогда как в секции [L048] драйвер JEMM386.SYS (примечание 4 к 5.04-02) для обеспечения такого же доступа переключает процессор в режим V86. Обе упомянутые секции включают загрузку драйвера XMS-памяти, драйвера для доступа к дисководам CD/DVD-ROM, драйвера "мыши" и ряда других драйверов. Следует отметить, что драйвер RAM-диска TDSK.EXE, в отличие от большинства других драйверов, загружается в обыкновенную память, и что размер RAM-диска здесь не задан. Если образование RAM-диска будет сочтено целесообразным, то его размер будет задан позже, в ходе интерпретации строк файла AUTOEXEC.BAT (9.09-02).

Существенные различия между конфигурациями, определяемыми пунктами меню L029 – L048, также проявят себя позже, в ходе интерпретации строк файла AUTOEXEC.BAT. Исключением является только конфигурация, задаваемая пунктом меню L104. Соответствующая этому пункту секция [L104] файла CONFIG.SYS не содержит ничего, кроме установок параметров DOS. Файл CONFIG.SYS кончается секцией [common], которая исполняется при любом варианте загрузки. В ней имеется лишь одна командная строка, загружающая командный интерпретатор COMMAND.COM.

Все пути, указываемые в строках файла CONFIG.SYS, не содержат ссылок на букву загрузочного диска и пригодны для загрузки с любого диска. Тем не менее фактическое размещение используемых драйверов и программ по каталогам загрузочного диска должно соответствовать указанным путям. Изменения структуры каталогов в принципе допустимы, но они затрагивают также условия исполнения файла AUTOEXEC.BAT (9.09-02). Потому такие решения не следует принимать несогласованно.

Приведенный здесь текст надо набрать с помощью программы текстового редактора, как показано во вводной статье к главе 9, и сохранить под именем CONFIG.SYS. Получаемый файл предназначен для размещения в корневом каталоге сменного загрузочного носителя.

9.09-02 Файл AUTOEXEC.BAT с операциями адаптации.

Все специфичные операции исследования дисков и памяти компьютера выполняются в ходе интерпретации строк файла AUTOEXEC.BAT. При исполнении этих операций используется программа REASSIGN.COM (9.06), которую Вы можете сотворить самостоятельно. Полноценной замены для нее, к сожалению, нет.

Для упрощения ориентации в предлагаемом файле AUTOEXEC.BAT используемые в нем метки включают номера строк: например, метка L029 означает 29-ю строку, считая от начала файла. Кроме того, строки, номера которых кратны 10, отмечены соответствующими комментариями.

Локальные переменные файла AUTOEXEC.BAT названы V0 – V8. Одна из них – V8 – постоянной миссии не имеет. Назначение остальных переменных следующее:

V0 – список дисков, подлежащих тестированию,
V1 – размер наибольшего свободного блока XMS-памяти,
V2 – рекомендуемый размер RAM-диска,
V3 – диск, на который предстоит перебазировать DOS,
V4 – диск, с которого производится загрузка компьютера,
V5 – статус доступности загрузочного диска для записи,
V6 – список недоступных или незаписываемых дисков,
V7 – список дисков, доступных для записи.

Ниже приведен текст предлагаемой версии файла AUTOEXEC.BAT.

```
@echo off
if %1"==J" if not %2"==" goto L%2
prompt $p$g
path ;
path=\DOS\MS7;\DOS\OTH
set V0=C D E F
set V4=33
Reassign.com V4
if errorlevel 2 set V0=%V0% B
       rem  ================== Line 10 ==================
if errorlevel 1 set V0=%V0% A
set comspec=%V4%\Command.com
set V1=11111111
set V2=300
Reassign.com V1
if errorlevel 1 if not errorlevel 2 set V2=5600
if errorlevel 2 if not errorlevel 3 set V2=16000
if errorlevel 3 if not errorlevel 100 set V2=32000
```

```
if errorlevel 100 for %%Z in (1 2) do set V%%Z=
       rem  ================== Line 20 ==================
if errorlevel 100 if not %config%"==L104" set config=L047
if not %config%"==L104" goto %config%
for %%Z in (3 5 6 7) do set V%%Z=
for %%Z in (%V0%) do call \Autoexec.bat J 117 %%Z: Q
if %V5%"==F" echo Warning: current disk %V4% is not writable!
if not %V7%"==" echo         Writable disk(s): %V7%
if %V7%"==" echo Warning: no writable disks have been found!
goto L104
:L029
       rem  ================== Line 30 ==================
:L031
set V3=
set ramdrive=?:
%V4%\DOS\DRV\Tdsk.exe %V2% /E /M /F:2
if not %ramdrive%"==?:" if not %ramdrive%"==::" set V3=%ramdrive%
if not %V3%"==" if not %V2%==300 goto L086
if not %V3%"==" if %V2%"==300" goto L104
set V1=
echo RAM-disk arranging procedure has failed!
       rem  ================== Line 40 ==================
goto L047
:L042
for %%Z in (1 1 1 1 1 1 1) do echo=
ctty nul
call %V4%\Autoexec.bat J 117 %V3% S
ctty con
:L047
:L048
for %%Z in (1 1 1 1 1 1 1) do echo=
       rem  ================== Line 50 ==================
for %%Z in (5 6 7) do set V%%Z=
ctty nul
for %%Z in (%V0%) do call %V4%\Autoexec.bat J 117 %%Z: V
ctty con
if %ramdrive%"==" set ramdrive=R:
if not %V7%"==" echo Writable disk(s): %V7%
if %V7%"==" echo Writable disks have not been found!
if not %V1%"==" echo Available XMS-memory is %V1% kb
if %V1%"==" echo XMS-memory is unavailable
       rem  ================== Line 60 ==================
echo  Select one of the shown keys and then press ENTER:
if not %V7%"==" echo      %V7% - set DOS onto the chosen disk
```

```
if not %V6%"==" echo      %V6% - retry inaccessible disk(s)
if not %V1%"==" echo      %ramdrive% - set a RAM-disk for DOS
echo      ESC - leave DOS on %V4%, no relocation
echo=
:L067
set V3=2
Reassign.com V3
       rem  ================== Line 70 ==================
if not errorlevel 128 if %V3%"==" goto L067
if errorlevel 128 set V3=
if errorlevel 128 goto L104
set V8=%path%
path %V3%
set V3=%path%:
path=%V8%
if %V3%"==%ramdrive%" if not %V1%"==" goto L031
set V8=N
       rem  ================== Line 80 ==================
for %%Z in (%V6%) do if %V3%==%%Z set V8=F
if %V8%==F goto L042
for %%Z in (%V7%) do if %V3%==%%Z set V8=T
if %V8%==T if %V3%"==%V4%" goto L104
if not %V8%==T goto L067
:L086
ctty nul
for %%Z in (. OTH MS7 VC4) do Attrib -h -r -s %V3%\DOS\%%Z\*.*
for %%Z in (. ..\TEMP OTH MS7 VC4) do md %V3%\DOS\%%Z
       rem  ================== Line 90 ==================
for %%Z in (Autoexec.bat Command.com) do copy /B \%%Z %V3%\DOS /Y
ctty con
if not exist %V3%\DOS\Command.com set V3=
if %V3%"==" echo Relocation attempt has failed!
if %V3%"==" goto L104
echo Copying the following files to disk %V3%
for %%Z in (OTH MS7 VC4) do copy /B \DOS\%%Z\*.* %V3%\DOS\%%Z /Y
set comspec=%V3%\DOS\Command.com
%V3%
       rem  ================== Line 100 ==================
for %%Z in (OTH MS7 VC4) do \DOS\MS7\Attrib +r \DOS\%%Z\*.ini > nul
set V4=%V3%
%V3%\DOS\Autoexec.bat J 104
:L104
if %V3%"==" for %%Z in (%V7%) do set V3=%%Z
if not %V3%"==" if not exist %V3%\Temp\nul md %V3%\Temp
```

```
if not %V3%"==" if exist %V3%\Temp\nul set Temp=%V3%\TEMP
if %Temp%"==" echo Warning: the TEMP variable is not defined!
set dircmd= /A /O:GNE /P
       rem  ================= Line 110 ==================
path=%V4%\DOS\OTH;%V4%\DOS\MS7;%V4%\DOS\VC4;%V4%\;%V4%\DOS
%V4%\DOS\OTH\Blue.com
if not %config%==L104 set VC=%V4%\DOS\VC4
for %%Z in (0 1 2 3 4 5 6 7 8) do set V%%Z=
if not %config%==L104 %VC%\Vc.com /TSR /no2E /noswap
goto END
:L117
%comspec% nul /f /c if exist %3\nul cd \DOS
if not exist ..\NUL if %4"==Q" goto END
       rem  ================= Line 120 ==================
if not exist ..\NUL goto L152
%comspec% nul /f /c call %0 J 141 %3 %4
%V4%
if not exist ..\NUL if %3"==%V4%" set V5=
if not exist ..\NUL if %4"==S" goto END
if not exist ..\NUL if not %V7%"==" set V7=%3 %V7%
if not exist ..\NUL if %V7%"==" set V7=%3
if not exist ..\NUL goto END
cd \
       rem  ================= Line 130 ==================
if not %4"==Q" echo Disk %3 is not writable! > con
if %3"==%V4%" set V5=F
if not %4"==S" if not %V6%"==" set V6=%V6% %3
if not %4"==S" if %V6%"==" set V6=%3
if not %4"==S" goto END
echo If disk %3 is write-protected, close protection > con
echo hole in its cartridge. Press any key to continue > con
pause < con > nul
goto END
       rem  ================= Line 140 ==================
:L141
set TEMP=
%3
ver | shift
if not %4"==" goto END
cd %V4%\
if not %3"==V" goto END
echo Disk %2 is writable: > con
dir /a:ARD /-p /v %2\ | %V4%\DOS\MS7\Find.exe "otal d" > con
       rem  ================= Line 150 ==================
```

```
goto END
:L152
cd \DOS
set V8=if errorlevel 15 if not errorlevel 16 cd \
if %4"==S" set V8=if errorlevel 20 cd \
%comspec% /f /c for %%Z in ("Label.exe %3trial" "%V8%") do %%Z
if not exist ..\nul if not %4"==S" goto END
if not %4"==S" if not %V6%"==" set V6=%V6% %3
if not %4"==S" if %V6%"==" set V6=%3
       rem  ================= Line 160 ==================
set V8=Disk %3 either is unformatted or has an improper format
if exist ..\nul set V8=Drive %3 probably has no media inside
if %4"==V" set V8=Disk %3 is either unformatted or not inserted
echo %V8% > con
cd \
if not %4"==S" goto END
echo Insert proper media and press any key to continue > con
pause < con > nul
:END
```

Особенности данного файла AUTOEXEC.BAT начинаются с условного перехода во 2-й строке, который позволяет рекурсивно вызывать встроенные подпрограммы. Однако условия перехода заданы так, что при первой интерпретации файла AUTOEXEC.BAT этот переход заведомо не исполняется.

Далее производится присвоение значений нескольким переменным окружения. Значение переменной PATH, присваиваемое в 5-й строке, не окончательное: оно будет действовать только до перебазирования DOS, а потом будет заменено на значение, включающее окончательные спецификации диска.

Программа REASSIGN.COM вызывается в первый раз в 8-й строке для того, чтобы определить текущий диск. Оставляемый программой код уровня ошибки (errorlevel) помогает скомпоновать список дисков, подлежащих тестированию. Второй раз программу REASSIGN.COM вызывают в 15-й строке, чтобы выяснить доступность XMS-памяти и определить размер наибольшего свободного XMS-блока. На сей раз оставляемый код уровня ошибки подсказывает рекомендуемый размер RAM-диска, который позднее может быть создан. Если же обнаружится, что XMS-память вообще недоступна, то переменной CONFIG в 21-й строке будет дано новое значение, которое предотвратит попытки создания RAM-диска, заведомо обреченные на неудачу.

Важную роль играет команда GOTO в 22-й строке: она выполняет переход к частям файла AUTEXEC.BAT, соответствующим выбору пользователя в

конфигурационном меню, поскольку этот выбор отображен значением переменной CONFIG. Если избран вариант быстрой загрузки, то переход выводит на метки L029 или L031. Там в 34-й строке драйвер TDSK.EXE создает RAM-диск предписанного объема и делает букву RAM-диска значением переменной RAMDRIVE. Уместно напомнить: букву RAM-диска в значение переменной записывает только драйвер TDSK.EXE версии 2.42. В случае применения более ранней версии или других подобных драйверов (BITDISK.EXE, SRDISK.EXE, XMSDSK.EXE) надо просто присвоить значение R: переменной RAMDRIVE посредством команды SET (3.26). При этом, понятно, способность адаптации к непредвиденному назначению RAM-диску другой буквы будет утрачена.

Когда предписанный объем RAM-диска мал, тогда в 37-й строке выполняется переход к завершающей части файла AUTOEXEC.BAT, так что RAM-диск будет использован только для записи временных файлов. Но обычно объем RAM-диска вполне достаточен для размещения DOS, и тогда в 36-й строке происходит переход на метку L086, где осуществляется перебазирование DOS на RAM-диск.

Процедура перебазирования DOS построена так, чтобы ее можно было бы применить не только к RAM-диску, но и к любому непустому физическому диску. Для того она начинается со снятия файловых атрибутов в каталогах назначения, потому что иначе обычные средства DOS не смогут перезаписать одноименные файлы в каталогах назначения и даже не смогут обнаружить их наличие. Операция в строке 89 создает типичную структуру каталогов назначения, если изначально она не существовала. Затем в каталог %V3%\DOS копируются файлы AUTOEXEC.BAT и COMMAND.COM. Если они скопированы успешно, то в строке 97 остальные файлы DOS копируются в соответствующие каталоги. Переменным COMSPEC и V4 присваиваются новые, скорректированные значения. Команда в строке 103 передает управление не файлу AUTOEXEC.BAT, а его копии, находящейся в каталоге назначения %V3%\DOS, причем так, что возврат к исполнению первоначального файла никогда уже не произойдет. Исполнение копии файла AUTOEXEC.BAT начнется со строки 105.

Если пользователь с самого начала изберет конфигурируемый вариант загрузки, то переход из 22-й строки приведет к метке L047 или L048, где начинается процедура исследования имеющихся дисков. Исследование выполняет подпрограмма L117, являющаяся частью того же файла AUTOEXEC.BAT. Ее рекурсивно вызывает цикл FOR из строки 53, отдельно для каждого тестируемого диска. Подпрограмма L117 предпринимает первую попытку доступа к исследуемому диску в строке 118 чтобы выяснить, доступен ли он для чтения. Если диск не читается, то следует переход на метку L152, где следующая проверка в строке 156 должна отличить несуществующие накопители от тех, в которых носитель записи не форматирован или просто не вставлен. Несуществующие накопители игнорируются, а все остальные причисляются к списку недоступных

дисков, представляемому значением переменной V6. Поскольку статус таких дисков может измениться, их проверку потом можно будет повторить.

Доступные для чтения диски затем проверяются на доступность для записи подпрограммой L141, которую вызывает из строки 122 специально загружаемый модуль командного интерпретатора COMMAND.COM. Решающее испытание происходит в строке 144, где промежуточное перенаправление предполагает создание временного файла на испытуемом диске. Если временный файл не удастся создать, то не будет исполнена команда SHIFT в той же строке, нумерация формальных параметров не изменится, подпрограмма L141 завершится в строке 145, и команды в строках 133 – 134 занесут букву испытанного диска в список недоступных дисков. Если же временный файл удастся создать, то он сразу же будет автоматически удален, но команда SHIFT будет исполнена, команда ECHO в строке 148 выведет на экран подтверждающее сообщение, и команда DIR в 149-й строке покажет использование дискового пространства. В строке 151 исполнение подпрограммы L141 закончится, и буква испытанного записываемого диска в строках 126 – 127 будет добавлена к списку доступных дисков, представляемому значением переменной V7.

Исследование диска подпрограммой L117 заканчивается либо в строке 128 либо в строке 135 – это зависит от результата тестирования. В обоих случаях управление возвращается к циклу FOR в строке 53. Он продолжит вызывать подпрограмму L117 до тех пор, пока не подойдет к концу список подлежащих исследованию дисков, представляемый значением переменной V0. Когда все диски проверены, результаты исследования вместе с перечнем предлагаемых альтернатив будут выведены на экран дисплея командами в строках 56 – 65. Пользователю предлагается выбрать наиболее удачный вариант загрузки. Ответ пользователя воспримет программа Reassign.com (9.06) в строке 69. Поскольку ответ может быть выражен как заглавной, так и строчной буквой, команды в строках 74 – 77 преобразуют любую принятую букву в заглавную, а потом условные команды в строках 78 – 85 выполнят волю пользователя.

Проверка в строке 78 выясняет, не выбрал ли пользователь вариант перебазирования DOS на RAM-диск. Если да, то следует переход на метку L031, и все дальнейшие события повторят сценарий быстрого перебазирования на RAM-диск точно так, как он был описан выше. Если пользователь принял иное решение, то цикл FOR в строке 81 выяснит, не выражает ли оно желание повторно тестировать один из недоступных дисков. В таком случае последует переход на метку L042, а там команда в строке 45 снова вызовет подпрограмму L117 для повторной проверки избранного диска. Четвертый формальный параметр "S" подпрограммы L117 задает режим более тщательной проверки, отличающийся критериями тестов и выдачей детальных советов по обеспечению доступа к диску. Подпрограмма L117 сделает паузу, чтобы можно было сменить диск или закрыть отверстие защиты от записи в его картридже. Затем в строке 53 последует обычный

вызов той же подпрограммы L117 для того, чтобы обновить списки недоступных и доступных для записи дисков. Пользователю будет показан результат и снова будет предложен выбор предпочтительного варианта загрузки.

Выбор записываемого диска для перебазирования DOS регистрируется циклом FOR в 83-й строке. В таком случае исполнение проследует через метку L086 в ту часть файла AUTOEXEC.BAT, где выполняется перебазирование, причем оно будет выполнено точно так же, как описанное выше перебазирование на RAM-диск. Важно отметить, что перебазирование DOS не делает избранный диск загрузочным, а просто позволяет продолжить сеанс работы в DOS после изъятия загрузочного носителя записи из того устройства, которое было использовано для загрузки компьютера. Если избранный диск уже был загрузочным, то это свойство не будет утрачено, потому что перебазирование DOS не вносит изменений ни в boot-сектор, ни в комплект файлов корневого каталога.

Диск, с которого компьютер был загружен, вполне может быть доступен для записи, и тогда соответствующая ему буква будет включена в список, представленный значением переменной V7. Однако перебазирование DOS на этот диск не имеет смысла, и его надо предотвратить. Для этого служит проверка в строке 84, которая направляет дальнейшее исполнение на метку L104. Тем самым процедура перебазирования DOS будет обойдена. Так как загрузочный диск был признан доступным для записи, он же будет использован как место для записи временных файлов. К сожалению, на обычных дискетах емкостью 1.44 Мб места для временных файлов всегда недостаточно. Поэтому пользователю предоставлена еще одна возможность обойти процедуру перебазирования DOS: нажатие клавиши ESC вызывает переход от строки 73 на ту же метку L104, но в таком случае будет предпринята попытка перенаправить запись временных файлов на какой-нибудь другой пригодный для записи диск.

Последний пункт в загрузочном меню (9.09-01) - это загрузка без драйверов и без резидентных программ. Такой вариант тоже приводит к той же самой метке L104 через переход со строки 28. Но перед этим в 24-й строке будет вызвана подпрограмма L117, причем ее 4-й параметр Q задаст режим быстрого исполнения. Детальное исследование дисков и вывод промежуточных сообщений будут пропущены. Единственная цель такого исследования – подготовка данных для предупреждающих сообщений, выводимых в строках 25 – 27.

Все варианты загрузки сходятся в конечной части файла AUTOEXEC.BAT на метке L104. Цикл FOR в строке 105 назначает записываемый диск для временных файлов, если такое назначение не состоялось раньше. Операции в строках 106 – 114 присваивают окончательные значения переменным TEMP, DIRCMD и PATH, а также удаляют локальные переменные V0 – V8. Если не избран специальный вариант загрузки L104, то на исполнение будет вызван файл-менеджер Volcov

Commander. Последней операцией файла AUTOEXEC.BAT является безусловный переход на конечную метку END.

Представленный вариант файла AUTOEXEC.BAT надо поместить в корневой каталог сменного загрузочного носителя. В других каталогах того же носителя должны быть размещены все те файлы, которые приходится вызывать из строк файла AUTOEXEC.BAT. Предполагается, что в корневом каталоге находится командный интерпретатор Command.com, в каталоге \DOS\MS7 – файлы Attrib.exe, Find.exe и Label.exe, в каталоге \DOS\OTH – файлы Reassign.com и Blue.com, в каталоге \DOS\DRV – файл Tdsk.exe (версии 2.42), в каталоге \DOS\VC4 – файлы Vc.com и Vc.ovl. Разумеется, в тех же каталогах допустимо наличие других файлов. Если Вы намерены использовать иное расположение файлов или иную структуру каталогов, то в тексте файла AUTOEXEC.BAT все отличающиеся пути и ссылки должны быть соответствующим образом скорректированы.

9.10 Эксперименты с линейной адресацией

Широко распространено мнение, что современные 32-разрядные процессоры при работе в реальном режиме не позволяют обращаться к памяти сверх 1088 кбайт. Хотя официальные справочные данные это мнение не опровергают, тем не менее оно неверно. С начала 1990-х годов известны сообщения об использовании недокументированных свойств 32-разрядных процессоров для доступа к расширенной памяти. Автором идеи считают Томаса Родена.

Анализ кода популярного драйвера Himem.sys (5.04-01) показывает, что там имеются все элементы, необходимые для осуществления обмена данными с расширенной памятью в реальном режиме. Вероятно, драйвер Himem.sys именно так и действует, но официальных подтверждений того нет. Несмотря на солидный возраст вопроса, идея линейной 32-разрядной адресации в реальном режиме осталась на уровне малоизвестных частных предложений и примеров.

Сейчас проблема состоит не в раскрытии сокровенной истины, а в демонстрации конкретных путей осуществления тех возможностей, которые уже давно предоставлены программам реального режима. С этой целью в разделе 9.10 приведены тексты двух небольших программ: GS_limit.com снимает сегментную защиту для регистра GS, а GS_dump.com выводит дамп участка памяти по заданному 32-разрядному линейному адресу. Действие программ наглядно иллюстрирует рис. 7: та область памяти, которая только что была закрыта сегментной защитой, становится доступной после вызова программы GS_limit.com.

```
D:\MSDOSTXT\TRIAL>GS_dump.com 10FE0

 GS= 0010
 00010FE0   F1 32 1E 33 F3 32 F4 32 F5 32 F6 32 F7 32 F8 32  ±2.3≤2⌠2J2÷2≈2°2
 00010FF0   F9 32 FA 32 FB 32 FC 32 FD 32 FE 32 FF 32 00 33  ·2·2√2ⁿ2²2■2 2.3
 00011000       - above GS limit

D:\MSDOSTXT\TRIAL>GS_limit.com off

       GS segment limit protection is turned OFF

D:\MSDOSTXT\TRIAL>GS_dump.com 10FE0

 GS= 0008
 00010FE0   F1 32 1E 33 F3 32 F4 32 F5 32 F6 32 F7 32 F8 32  ±2.3≤2⌠2J2÷2≈2°2
 00010FF0   F9 32 FA 32 FB 32 FC 32 FD 32 FE 32 FF 32 00 33  ·2·2√2ⁿ2²2■2 2.3
 00011000   30 04 90 0C 11 04 32 00 00 00 01 00 00 00 00 00  0.P...2.........
 00011010   00 EB 0E 69 6F 6E 61 6C 56 43 56 49 45 57 20 20  .ы.ionalVCVIEW..
 00011020   45 58 54 20 00 00 00 00 00 00 00 00 00 00 9C 79  EXT...........by
 00011030   62 33 82 03 F7 02 00 00 4D 41 43 38 36 36 20 20  b3B.≈...MAC866..
 00011040   54 42 4C 20 00 00 00 00 00 00 00 00 00 00 01 00  TBL.............
 00011050   C2 28 83 03 10 01 00 00 00 00 00 00 00 00 00 00  ┬(Г.............

D:\MSDOSTXT\TRIAL>
```

Рис. 7

Тот, кто решит воспользоваться предлагаемыми программами, получит уникальный шанс увидеть все "закоулки" и "завороты" 32-разрядного адресного пространства.

9.10-01 Программа включения/отключения сегментной защиты.

Как только вычисляемый процессором линейный адрес выходит за границы сегмента, происходит срабатывание сегментной защиты, и на том обычно рушатся все надежды воспользоваться возможностями префикса разрядности адреса (7.02-07) в реальном режиме. Но ситуация не настолько безнадежна, как сначала кажется. Все современные процессоры, начиная с модели 80386, позволяют изменять размер сегмента, и его можно задать равным верхнему пределу адресации. Тогда любой вычисленный линейный адрес будет сочтен допустимым, и сегментная защита фактически будет отключена.

Отключение сегментной защиты – хорошо это или плохо? С одной стороны – хаос и крах субординации в многозадачных операционных системах. С другой стороны – новые перспективы для сервисных и диагностических программ, действующих в однозадачной операционной среде. Как любой эффективный инструмент, отключение сегментной защиты может быть и очень полезным, и очень опасным. Именно потому разработчики процессоров подошли к вопросам управления сегментной защитой с большой осторожностью. Они оставили такую возможность только для программ защищенного режима, действующих на высшем (нулевом) уровне привилегий.

Когда в защищенном режиме на высшем уровне привилегий уже действует ядро операционной системы, тогда у всех других программ просто нет шансов изменить ситуацию. Но пока компьютер работает в реальном режиме, шансы есть: нужно лишь "нырнуть" в защищенный режим, изменить там предел сегмента в "теневом" регистре процессора, а потом "вынырнуть" обратно. После этого в реальном режиме можно будет пользоваться линейной 32-разрядной адресацией относительно соответствующего сегментного регистра без риска "напороться" на срабатывание сегментной защиты. Такова, вкратце, основная мысль, реализуемая предлагаемой ниже программой.

В качестве "подопытного" сегментного регистра взят регистр GS, потому что программы реального режима редко обращаются к нему. Помимо того, правильно составленные программы не нарушают границы сегментов намеренно. В результате все испробованные автором программы, не предназначенные специально для выявления состояния сегментной защиты, работали совершенно одинаково как при установленной, так и при снятой защите сегмента GS.

Вследствие выбора сегментного регистра GS предлагаемая программа получила название GS_limit.com. Она отличается от аналогов, во-первых, тем, что способна выполнять свою миссию не только в "голой" DOS, но и после загрузки драйвера Himem.sys. Во-вторых, программа GS_limit.com способна не только снимать защиту сегмента GS, но и восстанавливать ее обратно. Для устранения предела сегмента GS надо ввести команду

```
GS_limit  off
```

Восстановление стандартного 64-килобайтного предела сегмента GS производится командой

```
GS_limit  on
```

Если ни один из двух упомянутых параметров (OFF или ON) не указан, то программа выведет справку помощи (help).

Чтобы скомпилировать исполняемый файл программы GS_limit.com, нужно сначала набрать с помощью программы редактирования приведенный ниже ассемблерный текст и сохранить его в отдельном файле, например, под именем GS_limit.scr. Словесные комментарии в ассемблируемых строках при наборе текста можно опустить. Обратите внимание на пустую строку в ассемблерном тексте – восьмую с конца. Она служит командой выхода из режима ассемблирования (7.01-04) и потому в файле GS_limit.scr обязательно должна быть сохранена. Потом файл GS_limit.scr надо переслать на исполнение отладчику Debug.exe через перенаправление ввода, например, так:

```
Debug.exe < GS_limit.scr
```

В результате исполнения содержащихся в ассемблерном тексте команд отладчик создаст в текущем каталоге исполняемый файл GS_limit.com длиной 662 байта. Для этого файл GS_limit.scr должен содержать следующие строки:

```
a 100
      ;************ Программа GS_limit.com *************
      ;********** Секция 1: проверка памяти и параметров
         ; 110 - адрес перехода со строки 104
cmp           SP,2010       ; 100 Выделено менее 8 кбайт?
jbe           0110          ;*104 Если да, оставим как есть
mov           SP,1FFE       ; 106 Вершину стека - на 8 кбайт
mov           BX,0200       ; 109 Запросим 8 кбайт памяти
mov           AH,4A         ; 10C Вызов функции создания
int           21            ; 10E          свободного MCB
cmp byte ptr  [005E],46     ;=110 Имеется ли параметр "F" ?
jz            0148          ;*115 Если да, то переход вперед
cmp byte ptr  [005E],4E     ; 117 Имеется ли параметр "N" ?
jz            0148          ;*11C Если да, то переход вперед
mov           DX,0317       ;*11E Если параметров нет, то
mov           AL,01         ; 121   переход на индикацию
jmp           020A          ;*123   сообщения 0317 и на выход
      ;********** Секция 2: данные, указатели, дескрипторы
         ; 126 - заполнен от 17D, вызван от 187, 1F7
         ; 128 - заполнен от 181, обращения от 190, 1E5
         ; 12A - псевдодескриптор GDT, обращение от 1CA
         ; 12C - линейный адрес GDT, получен от 1B2
                           ; 126 Место адреса вызова Himem
dw            0000,0000
                           ; 12A Размер GDT (3 дескриптора)
db            18 00
                           ; 12C Адрес GDT (смещение = 130)
db            30 01 00 00
                           ; 130 "Пустой" дескриптор 0000
db            00 00 00 00 00 00 00 00
                           ; 138 4-Гбайтный дескриптор 0008
db            FF FF 00 00 00 93 8F 00
                           ; 140 64-кбайтный дескриптор 0010
db            FF FF 00 00 00 93 00 00
      ;******** Секция 3: проверки процессора и защиты
         ; 148 - адрес перехода со строк 115, 11C
         ; 162 - адрес перехода со строки 158
pushf                      ;=148 Загрузим в стек 2 копии
pushf                      ; 149  исходного состояния флагов
pop           CX           ; 14A Выгрузим копию в CX
```

```
xor           CH,70         ; 14B Инвертируем биты 0C, 0D, 0E
push          CX            ; 14E Перешлем измененные биты
popf                        ; 14F       через стек в регистр
pushf                       ; 150       флагов, а потом снова
pop           AX            ; 151       через стек в AX
popf                        ; 152 Вернем исходные состояния
xor           AH,CH         ; 153 Установим несовпавшие биты
test          AH,40         ; 155 Наш процессор 16-битовый?
jz            0162          ;*158 Если нет, проверим защиту
mov           AL,04         ; 15A Если да, то переход на
mov           DX,024F       ;*15C       выход с индикацией
jmp           020A          ;*15F       сообщения 024F
test          AH,30         ;=162 Установлен защищенный режим?
jz            016F          ;*165 Если нет, следуем дальше
mov           AL,08         ; 167 Если да, то переход на
mov           DX,0271       ;*169       выход с индикацией
jmp           020A          ;*16C       сообщения 0271
      ;********* Секция 4: подготовка адресной линии A20
         ; 16F - адрес перехода со строки 165
         ; 18B - адрес перехода со строки 176
mov           AX,4300       ;=16F Проверим, установлен ли
int           2F            ; 172      драйвер Himem.sys
cmp           AL,80         ; 174 Если нет, перейдем и
jnz           018B          ;*176   обратимся к A20 напрямую
mov           AX,4310       ; 178 Если да, запросим адрес
int           2F            ; 17B    вызова функций Himem.sys
mov           [0126],BX     ;*17D Сохраним в памяти
mov           [0128],ES     ;*181    полученный адрес вызова
mov           AH,05         ; 185 Вызовем функцию активизации
call far      [0126]        ;*187    адресной линии A20
call          021C          ;=18B Проверим A20 и перейдем,
jnz           01A7          ;*18E    если линия A20 активна
mov word ptr  [0128],0001   ;*190 Метка: линия A20 не активна
mov           AL,FF         ; 196 Запросим активизацию линии
call          022F          ;*198    A20 у подпрограммы 022F
call          021C          ;*19B Проверим A20 и перейдем,
jnz           01A7          ;*19E    если линия A20 активна
mov           AL,02         ; 1A0 Если линия A20 не активна,
mov           DX,029C       ;*1A2    то переход на выход с
jmp           020A          ;*1A5    индикацией сообщения 029C
      ;********** Секция 5: подготовка таблицы GDT
         ; 1A7 - адрес перехода со строк 18E, 19E
                            ;=1A7 Префикс 32-битового операнда
db            66
```

```
xor           AX,AX         ; 1A8 Обнулим регистр EAX
mov           AX,CS         ; 1AA Запишем CS в AX
mov           CL,04         ; 1AC Сдвиг на 4 бита превратит
db            66
shl           AX,CL         ; 1AF  сегментный адрес в линейный
db            66
add           [012C],AX     ;*1B2 Вычислим линейный адрес GDT
      ;******** Секция 6: подготовка селектора и прерываний
         ; 1C3 - адрес перехода со строки 1BE
mov           CX,0008       ; 1B6 0008 - 4-Гбайтный селектор
cmp byte ptr  [005E],46     ; 1B9 Указан ли параметр "F" ?
jz            01C3          ;*1BE Если да, оставим CX=0008
mov           CX,0010       ; 1C0 Если нет, пусть CX=0010
cli                         ;=1C3 Запретим прерывания
mov           AL,80         ; 1C4 Отсылка байта 80h
out           70,AL         ; 1C6     в порт 70h
in            AL,71         ; 1C8     запретит NMI
      ;********** Секция 7: переходы к PM и обратно
                            ; 1CA = Lgdt fword ptr [012A]
db            0F 01 16 2A 01
                            ; 1CF = mov EAX,CR0
db            0F 20 C0
or            AL,01         ; 1D2 Установим бит защищенного
                            ; 1D4    режима (= mov CR0,EAX)
db            0F 22 C0
                            ; 1D7 Загрузим GS (=mov GS,CX)
db            8E E9
and           AL,FE         ; 1D9 Сбросим бит защищенного
                            ; 1DB    режима (= mov CR0,EAX)
db            0F 22 C0
      ;********** Секция 8: возврат к прежнему состоянию
         ; 1F5 - адрес перехода со строки 1EC
mov           AL,7F         ; 1DE Посылка байта 7Fh
out           70,AL         ; 1E0     в порт 70h
in            AL,71         ; 1E2     снова разрешает NMI
sti                         ; 1E4 Разрешим прерывания
cmp word ptr  [0128],0001   ;*1E5 Посмотрим метку линии A20
jb            01FB          ;*1EA Если 0000, ничего не делаем
ja            01F5          ;*1EC Если >, обратимся к Himem
mov           AL,FD         ; 1EE Если =0001, восстановим
call          022F          ;*1F0    состояние линии A20 с
jmp           01FB          ;*1F3    помощью подпрограммы 022F
mov           AH,06         ;=1F5 Вызов функции драйвера Himem
call far      [0126]        ;*1F7    для перекрытия линии A20
```

```
      ;********** Секция 9: вывод сообщения и возврат в DOS
         ; 1FB - адрес перехода со строк 1EA, 1F3
         ; 208 - адрес перехода со строки 203
         ; 20A - адрес перехода со строк 123, 15F, 16C, 1A5
mov           DX,02E9       ;=1FB Сообщение "Предел поставлен"
cmp byte ptr  [005E],46     ; 1FE Указан ли параметр "F" ?
jnz           0208          ;*203 Если да, то заменить на
mov           DX,02B9       ; 205      сообщение "Предел снят"
mov           AL,00         ;=208 Нулевой уровень ошибки
push          AX            ;=20A Точка входа для выведения
mov           AH,40         ; 20B          сообщений на экран
mov           BX,DX         ; 20D Считаем в регистр CX
mov           CX,[BX-02]    ; 20F     число выводимых знаков
mov           BX,0001       ; 212 0001 = ссылка на STDOUT
int           21            ; 215 Вывод сообщения в STDOUT
pop           AX            ; 217 Вернем уровень ошибки в AL
mov           AH,4C         ; 218 Вызов функции
int           21            ; 21A       завершения программы
      ;******** Секция 10: подпрограмма проверки линии A20
         ; 21C - адрес вызова из строк 18B, 19B
push          DS            ;=21C Сохраним состояние DS
xor           SI,SI         ; 21D Обнулим регистр SI
mov           DS,SI         ; 21F Теперь DS:SI=0000:0000
mov           DI,F0F1       ; 221 Установим ES:DI=F0F1:F0F0
mov           ES,DI         ; 224    чтобы получился адрес
dec           DI            ; 226    F0F10h + F0F0h = 100000h
cld                         ; 227 Зададим счет на увеличение
mov           CX,0010       ; 228 Зададим предел 16 байт и
repz                        ; 22B   повторение до несовпадения
cmpsb                       ; 22C Адреса "свернуты" ?
pop           DS            ; 22D Возврат результата через
ret                         ; 22E       состояние флага ZF
      ;****** Секция 11: подпрограмма управления линией A20
         ; 22F - адрес вызова из строк 198, 1F0
push          AX            ;=22F Сохраним команду в стеке
call          0241          ;*230 Ждем готовности контроллера
mov           AL,D1         ; 233 D1 - первый байт команды,
out           64,AL         ; 235       посылаемый в порт 64h
call          0241          ;*237 Ждем готовности контроллера
pop           AX            ; 23A Вернем команду в AX
out           60,AL         ; 23B      и пошлем ее в порт 60h
call          0241          ;*23D Подождем срабатывания
ret                         ; 240      контроллера и вернемся
      ;******** Секция 12: подпрограмма ожидания готовности
```

```
        ; 241 - адрес вызова из строк 230, 237, 23D
        ; 245 - адрес цикла из строки 249
push        CX          ;=241 Сохраним CX в стеке
mov         CX,FFFF     ; 242 Запишем в CX число циклов
in          AL,64       ;=245 Считаем байт из порта 64h
test        AL,02       ; 247 Проверим состояние 2-го бита
loopnz      0245        ;*249 Повторим, если порт занят
pop         CX          ; 24B Восстановим состояние CX
ret                     ; 24C Возврат из подпрограммы
     ;********** Секция 13: сообщения
db          20 00
        ; 24F - 1-е сообщение, вызываемое из строки 15C
db          0D 0A 09 "16-bit processor"
db          20 "can" 27 "t suit" 0D 0A
db          29 00
        ; 271 - 2-е сообщение, вызываемое из строки 169
db          0D 0A 09 "GS_limit can" 27 "t run"
db          20 "in protected mode" 0D 0A
db          1B 00
        ; 29C - 3-е сообщение, вызываемое из строки 1A2
db          0D 0A 09 "Line A20 control error" 0D 0A
db          2E 00
        ; 2B9 - 4-е сообщение, вызываемое из строки 205
db          0D 0A 09 "GS segment limit"
db          20 "protection is turned OFF" 0D 0A
db          2C 00
        ; 2E9 - 5-е сообщение, вызываемое из строки 1FB
db          0D 0A 09 "GS segment limit"
db          20 "protection is restored" 0D 0A
db          7F 00
        ; 317 - 6-е сообщение, вызываемое из строки 11E
db          0D 0A "GS_limit.com removes the GS segment"
db          20 "limit protection or can restore it back"
db          0D 0A "Usage examples:" 0A 0D 09 09 "GS_lim"
db          "it off" 0A 0D 09 09 "GS_limit on" 0D 0A
        ; 396 Конец программы

n GS_limit.com
rBX
0000
rCX
0296
w
q
```

Начинается файл в секции 1 с обычных процедур высвобождения излишней памяти (примечание 5 к А.12-7) и проверки наличия параметров. Форма представления параметров в командной строке намеренно выбрана так, чтобы они оказались автоматически вписанными в первый блок FCB (примечание 4 к А.07-1) в составе PSP, причем сразу в нормализованном виде. Если ни один из параметров не указан, то в строке 123 происходит переход в секцию завершения, где выводится справка помощи, и затем программа завершается, оставляя код ошибки 01.

Если необходимый параметр указан, то исполнение продолжается со строки 148 в секции 3, где проверяется пригодность процессора для решения поставленной задачи, а также необходимое условие: работа процессора в реальном режиме. Смысл выполняемых проверок изложен в примечаниях 2 и 3 к разделу А.11-4. Если какое-либо из упомянутых условий не выполняется, то происходит переход в секцию завершения, где выводится на экран соответствующее сообщение об ошибке, и затем программа завершается, оставляя код ошибки 04 или 08.

Когда проверка состояния процессора пройдена, исполнение продолжается со строки 16F в секции 4, где выполняется подготовка линии A20 шины адреса. Открывать линию A20 шины адреса приходится по-разному в зависимости от того, установлен ли драйвер Himem.sys или нет. Если драйвер установлен, то необходимо действовать через его функцию AH=05 (А.12-3), как показано в строке 185. Если же драйвер Himem.sys не установлен, то приходится вызывать подпрограмму 022F из секции 11, которая попытается открыть линию A20 "руками" контроллера клавиатуры (подробнее – в примечании 1 к А.11-3). Исходное и получившееся состояния линии A20 проверяются в строках 18B и 19B вызовами подпрограммы 021C из секции 10. Если попытки открыть линию A20 будут неудачны, то в строке 1A5 произойдет переход в секцию завершения, на экран будет выведено сообщение об ошибке, и на том исполнение программы закончится, оставив код уровня ошибки 02. Как правило, линию A20 так или иначе удается открыть, и тогда исполнение программы продолжается со строки 1A7 в секции 5.

Команды секции 5 подготавливают псевдодескриптор таблицы GDT, из которого будет считан ее адрес в регистр GDTR. В строке 1AC сегментный адрес CS сдвигом на 4 бита влево преобразуется в линейный адрес CS. Операция суммирования в строке 1B2 формирует линейный адрес таблицы GDT в шаблоне псевдодескриптора GDT (в строках 12A – 12F секции 2).

Расшифровку структуры каждого дескриптора в таблице GDT можно посмотреть в разделе А.12-2. Второй и третий дескрипторы в таблице GDT предназначены для сегментов данных, причем второй дескриптор (строка 138, селектор 0008), задающий 4-Гигабайтный размер сегмента, служит для снятия защиты с сегмента GS. Третий дескриптор (строка 140, селектор 0010), задающий стандартный 64-килобайтный размер сегмента, служит для восстановления сегментной защиты. Перед переходом в защищенный режим в регистре CX должен

быть подготовлен селектор именно того дескриптора, который соответствует поставленной задаче. Команды в первых четырех строках секции 6 (1B6 – 1C0) выбирают тот или другой селектор – 0008 или 0010. Команды в последних строках секции 6 непосредственно предшествуют переходу в защищенный режим и служат для запрета прерываний, включая NMI (подробнее – в примечании 1 к 8.01-03).

Секция 7 начинается с команды LGDT (= Load Global Descriptor Table), загружающей псевдодескриптор таблицы GDT в специальный (предназначенный только для него) регистр GDTR. Отладчик Debug.exe о существовании команды LGDT, конечно, не догадывается, и потому ее код (0F 01 16) введен как данные инструкцией DB. Последние 2 байта в команде LGDT – 2A 01 – это смещение псевдодескриптора относительно сегментного адреса в регистре DS, то есть просто номер строки 012A, с которой начинается шаблон псевдодескриптора в секции 2.

Собственно переход в защищенный режим можно было бы выполнить вызовом INT 15\AH=89h (8.01-78), но тогда пришлось бы готовить более громоздкую таблицу GDT, а потом, помимо прочего, перепрограммировать оба контроллера прерываний обратно для работы с таблицей прерываний реального режима. Чтобы избежать лишних сложностей, здесь переход в защищенный режим выполнен установлением в единицу младшего бита PE (= Protection Enable) в управляющем регистре CR0 (A.11-4). Для этого в строке 1CF содержимое регистра CR0 считывается, в нем устанавливается бит PE, а потом в строке 1D4 оно записывается обратно в регистр CR0. Команды считывания и записи в регистр CR0, описанные в примечании 1 к разделу 7.03-58, отладчику Debug.exe неизвестны, и потому в строках 1CF и 1D4 они введены как данные инструкцией DB.

Конечно, для полноценной работы в защищенном режиме одного установления бита PE недостаточно, но в данном случае большего не требуется. В защищенном режиме программа GS_limit.com должна выполнить всего одну операцию: записать в регистр GS тот селектор (0008 или 0010), который уже подготовлен в регистре CX. Код команды копирования содержимого регистра CX в регистр GS вводится в строке 1D7 инструкцией DB, потому что отладчик Debug.exe "не знает" команд обращения к регистру GS (они показаны в примечании 2 к разделу 7.03-58). Запись селектора в регистр GS автоматически повлечет за собой то, ради чего написана программа GS_limit.com: занесение в "теневой" регистр процессора данных о размере сегмента GS из того дескриптора таблицы GDT, на который укажет подготовленный селектор.

Когда вершина достигнута, пора побеспокоиться о том, чтобы аккуратно с нее спуститься. Прежде всего надо сбросить бит PE в том коде, который был считан из управляющего регистра CR0 и с тех пор хранится в регистре EAX. Команда в строке 1DB запишет этот код обратно в регистр CR0 и осуществит тем самым возврат процессора в реальный режим. Потом команды в секции 8 снимут запрет прерываний и восстановят исходное состояние линии A20 адресной шины, причем

благодаря проверке в строке 1E5 для восстановления состояния линии A20 будут применены именно те средства, с помощью которых это состояние было изменено.

В начальных строках секции 9 производится выбор подходящего сообщения об успешном завершении программы. Команды в строках 20B – 215 выводят это сообщение на экран. Последним следует вызов стандартной функции завершения программы в строках 218 – 21A.

После своего завершения программа GS_limit.com оставит код уровня ошибки, который иногда представляет отдельный интерес и дает повод использовать программу GS_limit.com не только по своему прямому назначению, но также, например, для определения режима работы процессора:

```
GS_limit  on > nul
if errorlevel 7 echo Processor runs in V86 mode
if not errorlevel 7 echo Processor runs in real mode
```

Последняя особенность завершения программы GS_limit.com состоит в том, что независимо от характера выполненной операции число, оставляемое в регистре GS, сохраняет статус селектора. Это дает возможность экспериментировать с адресацией до тех пор, пока любой акт записи в регистр GS не вернет его содержимому обычный статус сегментного адреса.

9.10-02 Программа вывода дампа по линейному адресу.

Представленная здесь программа GS_dump.com показывает на экране 128-байтовый дамп участка памяти почти так же, как это делает отладчик Debug.exe в ответ на команду "D" (6.05-04). Главное отличие состоит в том, что для выведения дампа программа GS_dump.com использует не обычные адреса, состоящие из сегмента и смещения, а 32-битовые линейные адреса, позволяющие обращаться к любому участку 4-Гигабайтового адресного пространства, если, конечно, сегментная защита заранее отключена программой GS_limit.com (9.10-01).

Когда сегментная защита действует, программа GS_dump.com покажет дамп в "разрешенных" пределах относительно сегментного адреса в регистре GS. Если же заданный адрес вызовет срабатывание защиты, то на экран будет выведено соответствующее сообщение об ошибке.

Помимо очевидного демонстрационного эффекта, программа GS_dump.com предоставляет ответы на ряд нетривиальных вопросов, неизбежно возникающих при попытках практически осуществить линейную адресацию в DOS.

В командной строке вслед за именем программы GS_dump.com должен быть указан линейный адрес, содержащий до 8 шестнадцатеричных цифр, например:

```
GS_dump.com  FFFE0
```

Указанный адрес воспринимается как абсолютный, то есть отсчитываемый от нуля независимо от числа в регистре GS. Кроме адреса, в командной строке может быть указан один из двух допустимых дополнительных параметров:

A – не изменять состояние линии A20 адресной шины;
R – обнулить сегментный регистр GS.

Командная строка с дополнительным параметром выглядит, например, так

```
GS_dump.com  FFFE0  A
```

Когда линия A20 не активна, вызов программы GS_dump.com с параметром "A" покажет "заворот" адресного пространства с адреса 100000h. По умолчанию программа GS_dump.com активизирует линию A20 и всегда показывает адресное пространство открытым независимо от ее исходного состояния.

Чтобы скомпилировать исполняемый файл программы GS_dump.com, нужно набрать с помощью программы редактирования приведенный ниже ассемблерный текст и сохранить его в отдельном файле, например, под именем GS_dump.scr. Словесные комментарии в ассемблируемых строках при наборе текста можно опустить. Обратите внимание на пустую строку в ассемблерном тексте – восьмую с конца. Она выводит отладчик из режима ассемблирования (7.01-04) и потому в файле GS_dump.scr обязательно должна быть сохранена. Потом файл GS_dump.scr надо переслать на исполнение отладчику Debug.exe через перенаправление ввода, например, так:

```
Debug.exe < GS_dump.scr
```

В результате исполнения содержащихся в ассемблерном тексте команд отладчик создаст в текущем каталоге исполняемый файл GS_dump.com длиной 1291 байт. Во избежание ошибок, конечно, нужно будет выполнить все проверки, о которых написано в разделе 9.07.

```
a 100
      ;************* Программа GS_dump.com **************
      ;********* Секция 1: начальные приготовления
         ; 110 - адрес перехода со строки 104
cmp           SP,2010       ; 100 Выделено менее 8 кбайт?
jbe           0110          ;*104 Если да, оставим как есть
mov           SP,1FFE       ; 106 Вершину стека - на 8 кбайт
mov           BX,0200       ; 109 Запросим 8 кбайт памяти
mov           AH,4A         ; 10C Вызов функции создания
int           21            ; 10E          свободного MCB
push          CS            ;=110 Подготовим
pop           DS            ; 111           DS = CS
db            66
```

```
xor           BX,BX         ; 113 Подготовим EBX=0
      ;********* Секция 2: проверка процессора
          ; 12F - адрес перехода со строки 125
pushf                       ; 115 Загрузим в стек 2 копии
pushf                       ; 116   исходного состояния флагов
pop           CX            ; 117 Выгрузим копию в CX
xor           CH,70         ; 118 Инвертируем биты 0C, 0D, 0E
push          CX            ; 11B Перешлем измененные биты
popf                        ; 11C        через стек в регистр
pushf                       ; 11D        флагов, а потом снова
pop           AX            ; 11E        через стек в AX
popf                        ; 11F Вернем исходные состояния
xor           AX,CX         ; 120 Установим несовпавшие биты
test          AH,40         ; 122 Наш процессор 16-битовый?
jz            012F          ;*125 Если нет, проверим защиту
mov           AL,02         ; 127 Если да, то переход на
mov           DX,03D6       ;*129        выход с индикацией
jmp           02BF          ;*12C        сообщения 03D6
test          AH,30         ;=12F Установлен защищенный режим?
jz            0163          ;*132 Если нет, обойдем секцию 3
      ;********* Секция 3: проверки защищенного режима
         ; 14B - адрес перехода со строки 141
mov word ptr  [03D4],0483   ;*134 Изменим адрес сообщения
mov           AX,1687       ; 13A Вызов функции обнаружения
int           2F            ; 13D              сервера DPMI
or            AX,AX         ; 13F Сервер DPMI загружен?
jnz           014B          ;*141 Если нет, перейдем дальше
mov           AL,08         ; 143 Если да, то выход из
mov           DX,03FB       ;*145     программы с индикацией
jmp           02BF          ;*148     сообщения 03FB
push          DS            ;=14B Сохраним DS в стеке
mov           DS,BX         ; 14C Сегмент таблицы прерываний
mov           DS,[019E]     ; 14E Загрузим сегмент EMM
cmp word ptr  [0014],4249   ; 152 Версия EMM от фирмы IBM ?
pop           DS            ; 158 Восстановим сегмент DS
jz            0163          ;*159 Продолжим, если EMM от IBM
mov           AL,02         ; 15B Если нет, то выход из
mov           DX,041E       ;*15D      программы с индикацией
jmp           02BF          ;*160      сообщения 041E
      ;********* Секция 4: получение линейного адреса
         ; 163 - адрес перехода со строк 132, 159
         ; 16A - адрес назначения цикла со строки 19C
         ; 17B - адрес перехода со строки 175
         ; 188 - адрес перехода со строки 17E
```

```
        ; 194 - адрес перехода со строк 16F, 179
mov          CX,0004       ;=163 Установим сдвиг 4 бита
cld                        ; 166 Счет SI на увеличение
mov          SI,005D       ; 167 Зададим стартовый адрес
lodsb                      ;=16A Загрузим один байт в AL
sub          AL,30         ; 16B Преобразуем знак ASCII
cmp          AL,09         ; 16D Если это десятичная цифра,
jbe          0194          ;*16F      то добавим ее к EBX
sub          AL,07         ; 171 Преобразуем коды букв A-F
cmp          AL,0A         ; 173 Отсечем знаки до буквы A
jb           017B          ;*175 Перейдем выяснять причину
cmp          AL,0F         ; 177 Выберем буквы A-F и
jbe          0194          ;*179      добавим к числу в EBX
cmp          SI,005E       ;=17B Это первая итерация?
jnz          0188          ;*17E Если нет, посмотрим дальше
mov          AL,01         ; 180 Если да, то выход из
mov          DX,04F1       ;*182      программы с индикацией
jmp          02BF          ;*185      сообщения 04F1
cmp          AL,E9         ;=188 Последний знак - пробел 20h?
jz           019E          ;*18A Тогда все, больше знаков нет
mov          AL,01         ; 18C Если знак - не пробел, то
mov          DX,04CB       ;*18E      выход из программы с
jmp          02BF          ;*191      выводом сообщения 04CB
                           ;=194 Префикс 32-битового операнда
db           66
shl          BX,CL         ; 195 Сдвиг EBX 4 бита влево
or           BL,AL         ; 197 Вставим знак в регистр BL
cmp          SI,0064       ; 199 Достигнута ли 8-я итерация?
jbe          016A          ;*19C Если нет, повторить цикл
      ;********* Секция 5: активизация линии A20
        ; 19E - адрес перехода со строки 18A
        ; 1C3 - адрес перехода со строки 1AC
        ; 1D3 - адрес перехода со строки 1A3
        ; 1D8 - адрес перехода со строки 1C6
cmp byte ptr [006D],41     ;=19E Имеется ли параметр "A"?
jz           01D3          ;*1A3 Если да, оставим линию A20
mov          AX,4300       ; 1A5 Теперь посмотрим,
int          2F            ; 1A8      установлен ли
cmp          AL,80         ; 1AA      драйвер Himem.sys
jnz          01C3          ;*1AC Если нет, то переход
push         BX            ; 1AE Сохраним значение BX
mov          AX,4310       ; 1AF Найдем адрес вызова
int          2F            ; 1B2      функций Himem.sys
mov          [03C0],BX     ;*1B4 Запомним адрес вызова
```

```
mov           [03C2],ES     ;*1B8      функций Himem.sys
mov           AH,05         ; 1BC Вызовем функцию
call far      [03C0]        ;*1BE      активизации линии A20
pop           BX            ; 1C2 Восстановим значение BX
call          02D3          ;=1C3 Активна ли линия A20?
jnz           01D8          ;*1C6 Если да, то переход
mov word ptr  [03C2],0001   ;*1C8 Если нет, поставим метку
mov           AL,FF         ; 1CE Активизируем линию A20
call          02EB          ;*1D0      подпрограммой 02EB
call          02D3          ;=1D3 Активна ли линия A20?
jz            01DD          ;*1D6 Если да, дезактивируем
mov byte ptr  [0445],24     ;=1D8      сообщение 0445
      ;******* Секция 6: индикация сегментного адреса GS
         ; 1DD - адрес перехода со строки 1D6
         ; 1E6 - адрес перехода со строки 1E2
cmp byte ptr  [006D],52     ;=1DD Параметр "R" указан?
jz            01E6          ;*1E2 Если да, не считываем GS
                            ; 1E4  = MOV SI,GS
db            8C EE
                            ;=1E6  = MOV GS,SI
db            8E EE
                            ; 1E8  = PUSH GS
db            0F A8
pop           [03D0]        ;*1EA Запомним GS в 3D0
call          0339          ;*1EE Пошлем 0Dh 20h в STDOUT
call          0349          ;*1F1 Выведем 3D0 на экран
mov           DX,0445       ;*1F4 Выведем на экран
call          0330          ;*1F7       сообщение 0445
mov           DX,045B       ;*1FA Выведем на экран
call          0330          ;*1FD       сообщение 045B
      ;********* Секция 7: вычисление начала отсчета
db            66
xchg          SI,BX         ; 201 Переведем адрес в ESI
mov           BX,SI         ; 203 Вернем в BX только 2 байта
and           BX,000F       ; 205 Выберем самую правую цифру
neg           BL            ; 209 Получим начало отсчета
and           SI,FFF0       ; 20B Теперь ESI - базовый адрес
db            66
mov           DI,SI         ; 210 Скопируем ESI в EDI
db            66
xchg          [03D0],DI     ;*213 Обменяем GS с EDI
db            66
shl           DI,CL         ; 218 В EDI - линейный адрес GS
mov           DX,[03D4]     ;*21A Подготовим номер сообщения
```

```
db            66
sub           SI,DI          ; 21F Вычислим смещение в ESI
jnb           0226           ;*221 Если меньше нуля, сменим
mov           DX,0460        ;*223   номер сообщения на 0460
      ;********* Секция 8: замена флагов и указателей
         ; 226 - адрес перехода со строки 221
                              ;=226 Префикс 32-битного операнда
db            66
pushf                        ; 227 Затолкаем EFLAGS в стек
mov           BP,SP          ; 228 Указатель стека - в BP
mov           AL,[BP+02]     ; 22A Считаем и сохраним
mov           [03C4],AL      ;*22D      третий байт флагов
and byte ptr  [BP+02],FE     ; 230 Сбросим флаг выравнивания
db            66
popf                         ; 235 EFLAGS - обратно во флаги
in            AL,21          ; 236 Считаем маску порта 21h
mov           [03C5],AL      ; 238       и запомним ее
or            AL,60          ; 23B Установим биты 05 и 06
out           21,AL          ; 23D Запретим IRQ5 и IRQ6
mov           [03C8],DS      ;*23F Заполним сегменты в адресах
mov           [03CC],DS      ;*243     обработчиков прерываний
push          BX             ; 247 Сохраним BX в стеке
mov           AL,0D          ; 248 Заменим адреса обработчиков
call          03A0           ;*24A       прерывания INT 0D
call          03A0           ;*24D       и прерывания INT 0E
pop           BX             ; 250 Восстановим содержимое BX
      ;********* Секция 9: пробное считывание
         ; 251 - адрес назначения цикла со строки 289
mov           [03CE],SI      ;=251 Сохраним SI для сравнения
db            65 67
lodsb                        ; 257 Попытаемся считать байт
cmp           SI,[03CE]      ;*258 Изменилось ли значение SI?
jnz           0266           ;*25C Если да - в основной цикл
call          0342           ;*25E Индицируем линейный адрес
call          0321           ;*261 Вычислим следующий адрес
jmp           0286           ;*264 Переход к проверке адреса
      ;********* Секция 10: основной цикл вывода строки
         ; 266 - адрес перехода со строки 25C
         ; 26E - адрес назначения цикла со строки 273
         ; 278 - адрес назначения цикла со строки 27D
         ; 286 - адрес перехода со строки 264
                              ;=266 Префикс 32-битного операнда
db            66
dec           SI             ; 267 Восстановим индекс в SI
```

```
call          0349          ;*268 Индицируем линейный адрес
call          0318          ;*26B Промежуточная коррекция
call          0362          ;=26E Индицируем байт из AL
inc           BL            ; 271 Увеличим счет байтов на 1
loop          026E          ;*273 1-й цикл вывода строки
call          0309          ;*275 Промежуточная коррекция
call          0379          ;=278 Индицируем байт кодом ASCII
inc           BL            ; 27B Увеличим счет байтов на 1
loop          0278          ;*27D 2-й цикл вывода строки
call          0339          ;*27F Пошлем 0Dh 20h в STDOUT
mov           DX,[03D4]     ;*282 Заменим адрес сообщения
cmp           BL,80         ;=286 Данная строка последняя?
jb            0251          ;*289 Если нет - к следующей
      ;********* Секция 11: возврат к исходному состоянию
         ; 2B4 - адрес перехода со строки 2AB
mov           AL,0D         ; 28B Восстановим обработчики
call          03A0          ;*28D      прерывания INT 0D
call          03A0          ;*290      и прерывания INT 0E
mov           AL,[03C5]     ;*293 Считаем прежнюю маску и
out           21,AL         ; 296      пошлем ее в порт 21h
db            66
pushf                       ; 299 Считаем EFLAGS в стек
mov           BP,SP         ; 29A Указатель стека - в BP
mov           AL,[03C4]     ;*29C Считаем и восстановим
mov           [BP+02],AL    ; 29F    прежний 3-й байт флагов
db            66
popf                        ; 2A3 Восстановим EFLAGS
cmp word ptr  [03C2],0001   ;*2A4 Проверим метку линии A20
jb            02BA          ;*2A9 Если 0000, A20 не трогаем
ja            02B4          ;*2AB К A20 - с помощью Himem.sys
mov           AL,FD         ; 2AD Восстановим A20 с помощью
call          02EB          ;*2AF       подпрограммы 02EB
jmp           02BA          ;*2B2 Переход к завершению
mov           AL,06         ;=2B4 Вызов функции закрытия A20
call far      [03C0]        ;*2B6       драйвером Himem.sys
      ;******** Секция 12: завершение программы
         ; 2BA - адрес перехода со строк 2A9, 2B2
         ; 2BF - выход со строк 12C, 148, 160, 185, 191
mov           DX,0442       ;=2BA Сообщение "конец строки"
mov           AL,00         ; 2BD Код успешного завершения
push          AX            ;=2BF Сохраним код ошибки
mov           AH,09         ; 2C0 Вывод завершающего
int           21            ; 2C2        сообщения на экран
pop           AX            ; 2C4 Восстановим код ошибки
```

```
mov          AH,4C        ; 2C5 Вызов функции завершения
int          21           ; 2C7        и возврат в DOS
      ;******* Секция 13: обработчики исключений 0D и 0E
         ; 2C9  - должен быть указан в ячейке 3CA
         ; 2CC  - должен быть указан в ячейке 3C6
mov          DX,04A5      ;=2C9 Адрес вызова Int 0E
mov          BP,SP        ;=2CC Адрес вызова Int 0D
add word ptr [BP+00],+03  ; 2CE Корректируем адрес
iret                      ; 2D2    возврата и возвращаемся
      ;******** Секция 14: проверка состояния линии A20
         ; 2D3 - адрес вызова из строк 1C3, 1D3
push         DS           ;=2D3 Сохраним состояния
push         CX           ; 2D4        регистров DS и CX
db           66
xor          SI,SI        ; 2D6 Обнулим регистр ESI
push         SI           ; 2D8 Сохраним в стеке SI=0
mov          DS,SI        ; 2D9 Теперь DS:SI=0000:0000
mov          DI,F0F1      ; 2DB Установим ES:DI=F0F1:F0F0
mov          ES,DI        ; 2DE    чтобы получить адрес
dec          DI           ; 2E0    F0F10h + F0F0h = 100000h
cld                       ; 2E1 Зададим счет на увеличение
mov          CX,0010      ; 2E2 Предел счета - 16 байт
repz                      ; 2E5 Повторяем до несовпадения
cmpsb                     ; 2E6 Адреса "свернуты" ?
pop          SI           ; 2E7 Восстановим содержимое
pop          CX           ; 2E8   регистров и возвращаемся,
pop          DS           ; 2E9    сохраняя результат в
ret                       ; 2EA     состоянии флага ZF
      ;***** Секция 15: подпрограмма управления линией A20
         ; 2EB - адрес вызова из строк 1D0, 2AF
push         AX           ;=2EB Сохраним команду в стеке
call         02FD         ;*2EC Ждем готовности
mov          AL,D1        ; 2EF 1-й байт команды
out          64,AL        ; 2F1      посылаем в порт 64h
call         02FD         ;*2F3 Ждем готовности
pop          AX           ; 2F6 2-й байт команды из AX
out          60,AL        ; 2F7      посылаем в порт 60h
call         02FD         ;*2F9 Ждем пока контроллер
ret                       ; 2FC     сработает и возвращаемся
      ;********* Секция 16: ожидание готовности контроллера
         ; 2FD - адрес вызова из строк 2EC, 2F3, 2F9
         ; 301 - адрес циклического возврата со строки 305
push         CX           ;=2FD Сохраним CX в стеке
mov          CX,FFFF      ; 2FE Запишем в CX число циклов
```

```
in            AL,64         ;=301 Считаем состояние порта 64h
test          AL,02         ; 303 Проверим бит готовности
loopnz        0301          ;*305 Если порт не готов, повторим
pop           CX            ; 307 Восстановим CX и вернемся
ret                         ; 308      в вызывающую  программу
      ;********* Секция 17: промежуточная коррекция
         ; 309 - адрес вызова из строки 275
         ; 318 - адрес вызова из строки 26B
sub           BL,10         ;=309 Вернем счет на строку назад
push          BX            ; 30C Сохраним BX в стеке
mov           BL,10         ; 30D 10h = величина приращения
db            66
add           [03D0],BX     ;*310 Корректируем линейный адрес
db            66
sub           SI,BX         ; 315 Скорректируем смещение
pop           BX            ; 317 Восстановим BX
call          0393          ;=318 Дважды пошлем знак
call          0393          ;*31B          пробела в STDOUT
mov           CL,10         ; 31E Предустановим счет байтов
ret                         ; 320 Возврат из подпрограммы
      ;********* Секция 18: перевод строки дампа
         ; 321 - адрес вызова из строки 261
         ; 330 - адрес вызова из строк 1F7, 1FD
         ; 339 - адрес вызова из строк 1EE, 27F
add           BL,10         ;=321 Увеличим счет байтов на 16
push          BX            ; 324 Сохраним BX в стеке
mov           BL,10         ; 325 Зададим приращение на 16
db            66
add           [03D0],BX     ;*328 Коррекция линейного адреса
db            66
add           SI,BX         ; 32D Коррекция смещения
pop           BX            ; 32F Восстановим содержимое BX
mov           DI,DX         ;=330 Скопируем адрес в DI
mov           AH,09         ; 332 Функция вывода
int           21            ; 334      сообщения в STDOUT
mov byte ptr  [DI],24       ; 336 Дезактивируем сообщение
mov           AL,0D         ;=339 Пошлем команду возврата к
call          0395          ;*33B      началу строки в STDOUT
call          0393          ;*33E Пошлем пробел в STDOUT
ret                         ; 341 Возврат из подпрограммы
      ;********* Секция 19: индикация числа из ячейки 3D0
         ; 342 - адрес вызова из строки 25E
         ; 349 - адрес вызова из строк 1F1, 268
         ; 354 - адрес перехода со строки 35E
```

```
         ; 361 - адрес перехода со строки 347
mov           DI,DX         ;=342 Дезактивировано ли
cmp byte ptr  [DI],24       ; 344      запрошенное сообщение?
jz            0361          ;*347 Если да, не выводим его
mov           AL,0A         ;=349 Пошлем команду перевода
call          0395          ;*34B      строки в STDOUT
push          BX            ; 34E Сохраним содержимое BX
xor           BL,BL         ; 34F Отключим проверку предела
mov           DI,03D3       ; 351 В DI - адрес старшего байта
mov           AL,[DI]       ;=354 Выведем байт в AL
call          0368          ;*356 Вызовем трансляцию байта AL
dec           DI            ; 359 Перейдем к следующему байту
cmp           DI,03D0       ;*35A Это последний байт?
jnb           0354          ;*35E Если нет, повторим цикл
pop           BX            ; 360 Восстановим содержимое BX
ret                         ;=361 Возврат из подпрограммы
      ;********* Секция 20: подпрограмма трансляции AL
         ; 362 - адрес вызова из строки 26E
         ; 368 - адрес вызова из строки 356
call          0393          ;=362 Воспроизведем пробел
db            67 65
lodsb                       ; 367 GS:[32-битовый адрес]
push          CX            ;=368 Сохраним содержимое CX
mov           CL,04         ; 369 Сдвиг 4 бита зададим в CL
shl           AH,CL         ; 36B Очистим младшие 4 бита AH
shl           AX,CL         ; 36D Разделим полубайты из AL
shr           AL,CL         ; 36F Очистим старшие 4 бита AL
pop           CX            ; 371 Восстановим содержимое CX
call          0384          ;*372 Вызов индикации AH
call          0384          ;*375 Вызов индикации AL
ret                         ; 378 Возврат из подпрограммы
      ;********* Секция 21: индикация одного знака
         ; 379 - адрес вызова из строки 278
         ; 384 - адрес вызова из строк 372, 375
         ; 38E - адрес перехода со строк 37E, 382, 38A
         ; 393 - адрес вызова из строк 318, 31B, 33E, 362
         ; 395 - адрес вызова из 33B, 34B, переход от 391
                            ;=379 Считаем в AL один знак из
db            67 65
lodsb                       ; 37B      GS:[32-битный адрес]
cmp           AL,20         ; 37C Значение до 20h или больше?
ja            038E          ;*37E Значения AL < 20h заменим
mov           AL,2E         ; 380      точками и перейдем к
jmp           038E          ;*382      проверке границ
```

```
xchg        AH,AL       ;=384 Вход для индикации AH-AL
add         AL,30       ; 386 Переведем 0 - 9 в ASCII
cmp         AL,39       ; 388 Это 0 - 9 или A - F ?
jbe         038E        ;*38A Знаки 0 - 9 оставим как есть
add         AL,07       ; 38C A - F переведем в ASCII
cmp         BL,80       ;=38E Проверим границы
jb          0395        ;*391 Если за пределами границ,
mov         AL,20       ;=393      заменим знаки пробелами
push        AX          ;=395 Сохраним значения
push        DX          ; 396      регистров AX и DX
mov         AH,02       ; 397 Вызовем функцию
mov         DL,AL       ; 399      вывода одного
int         21          ; 39B      знака в STDOUT
pop         DX          ; 39D Восстановим значения
pop         AX          ; 39E      регистров AX и DX
ret                     ; 39F Возврат из подпрограммы
     ;********* Секция 22: замена обработчиков исключений
        ; 3A0 - адрес вызова со строк 24A, 24D, 28D, 290
mov         AH,35       ;=3A0 вызов функции считывания
int         21          ; 3A2   адреса обработчика в ES:BX
mov         DI,AX       ; 3A4 В DI подготовим адрес
shl         DI,1        ; 3A6   ячейки памяти: DI = AX•2
shl         DI,1        ; 3A8 350D•4=D434    350E•4=D438
sub         DI,D06E     ;*3AA D434-D06E=3C6  D438-D06E=3CA
push        DS          ; 3AE Сохраним содержимое
push        DX          ; 3AF       регистров DS и DX
lds         DX,[DI]     ; 3B0 DS:DX - указатель на ячейку
mov         AH,25       ; 3B2 Вызов функции записи
int         21          ; 3B4       адреса обработчика
pop         DX          ; 3B6 Восстановим содержимое
pop         DS          ; 3B7       регистров DS и DX
mov         [DI],BX     ; 3B8 Сохраним в памяти адрес
mov         [DI+02],ES  ; 3BA        прежнего обработчика
inc         AL          ; 3BD В AL - следующий номер
ret                     ; 3BF Возврат из подпрограммы
     ;******** Секция 23: адреса и данные
        ; 3C0 - адрес Himem, в строках 1B4, 1BE, 2B6
        ; 3C2 - сегмент Himem, в строках 1B8, 1C8, 2A4
db          00 00 00 00
        ; 3C4 - третий байт EFLAG, в строках 22D, 29C
db          00
        ; 3C5 - маска порта 21h, в строках 238, 293
db          00
        ; 3C6 - адрес этой ячейки вычислен в строке 3AA
```

```
        ; 3C8 - сегмент обработчика 0Dh, вписан из 23F
db          CC 02 00 00
        ; 3CA - адрес этой ячейки вычислен в строке 3AA
        ; 3CC - сегмент обработчика 0Eh, вписан из 243
db          C9 02 00 00
        ; 3CE - место хранения SI, доступ из 251, 258
db          00 00
        ; 3D0 - линейный адрес - 1EA, 213, 310, 328, 35A
        ; 3D3 - старший байт линейного адреса - 351
db          00 00 00 00
        ; 3D4 - место адреса сообщения - 134, 21A, 282
db          71 04
      ;******** Секция 24: сообщения
        ; 3D6  1-е сообщение, от строки 129
db          0D 0A 'Error: 16-bit machine can' 27
db          't suit' 0D 0A 24
        ; 3FB  2-е сообщение, от строки 145
db          0D 0A 'Error: can' 27 't run under WINDOWS'
db          0D 0A 24
        ; 41E  3-е сообщение, от строки 15D
db          0D 0A 'Error: incompatible EMM386 version'
        ; 442  4-е виртуальное сообщение, от строки 2BA
db          0D 0A 24
        ; 445  5-е сообщение, от строк 1D8, 1F4
db          09 'Line A20 is disabled' 24
        ; 45B  6-е сообщение, от строки 1FA
db          'GS=' 20 24
        ; 460  7-е сообщение, от строки 223
db          09 '- below GS base' 24
        ; 471  8-е сообщение, адрес в строке 3D4
db          09 '- above GS limit' 24
        ; 483  9-е сообщение, от строки 134
db          09 ' GS range or privilege violation' 24
        ; 4A5  10-е сообщение, от строки 2C9
db          09 'Page isn' 27 't initialized or is'
db          20 'swapped' 24
        ; 4CB  11-е сообщение, от строки 18E
db          0D 0A 'Address error: invalid character(s)' 0A
        ; 4F1  12-е сообщение (help), от строки 182
db          0D 0A 09 'GS_dump.com - linear GS address'
db          20 'dump utility' 0D 0A 'Usage examples:'
db          0D 0A 09 09 'GS_dump  002FABCD' 0D 0A 09 09
db          'GS_dump  002FABCD  A' 0D 0A 09 09
db          'GS_dump  002FABCD  R' 0D 0A 20
```

```
db            '002FABCD  - linear address example, up to'
db            20 '8 hexadecimal digits long' 0D 0A 09
db            'A  - option: don' 27 't try to enable' 20
db            'line A20' 0D 0A 09 'R  - option: reset GS'
db            20 'to zero' 0D 0A 24
          ; 60B

n GS_dump.com
rBX
0000
rCX
050B
w
q
```

Программа GS_dump.com начинается в первой секции с обычной процедуры высвобождения лишней выделенной памяти (примечание 5 к A.12-7). Проверки процессора и защищенного режима во второй секции выполняются так же, как в описанной выше программе GS_limit.com (9.10-01). Однако в случае констатации защищенного режима программа GS_dump.com не завершается сразу, а переходит к более детальной проверке в секции 3. Если выяснится, что защищенный режим установлен не операционной системой Windows, а совместимой версией драйвера EMM386.EXE (5.04-02), то исполнение будет продолжено.

В 4-й секции производится считывание линейного адреса из командной строки в регистр EBX. По ходу считывания выполняется преобразование знаков кода ASCII с анализом правильности считываемой последовательности. Если там будет обнаружено что-либо иное, кроме шестнадцатеричных цифр, то программа завершится выдачей сообщения об ошибке.

В 5-й секции программы GS_dump.com осуществляется активизация линии A20 адресной шины, причем точно так же, как это делается в секции 4 описанной выше программы GS_limit.com (9.10-01).

Оригинальные и принципиально важные операции в программе GS_dump.com начинаются с подготовки регистра GS в секции 6. Дело в том, что до первого акта записи в регистр GS число в нем может сохранять статус селектора. Чтобы избавиться от неопределенности, в строке 1E6 выполняется запись в регистр GS. Даже когда содержимое регистра GS желательно сохранять, в него записывают то самое число, которое заранее, в строке 1E4, было считано из того же регистра GS. После операции записи любое число в сегментном регистре процессор будет интерпретировать как обычный сегментный адрес.

В последующих строках секции 6 вызовом подпрограммы 0349 считанный сегментный адрес GS выводится на экран. Далее, в строках 201 – 20B секции 7, из

указанного пользователем линейного адреса вычисляются байт начала отсчета и базовый адрес, то есть линейный адрес первого байта в строке дампа. В строке 218 сдвиг на 4 бита влево преобразует сегментный адрес GS в линейный адрес. Разность базового линейного адреса и линейного адреса GS, вычисляемая в строке 21F, представляет собой смещение, необходимое для считывания данных по тому абсолютному адресу, который был указан пользователем в командной строке.

Прежде чем подавать вычисленные адреса процессору для исполнения считывания, нужно сделать еще несколько важных приготовлений, потому что операция считывания по 32-битовому линейному адресу может привести к генерации исключений INT 0D, INT 0E (примечания 6 и 7 к разделу 8.01-09) и INT 11 (примечание к 8.01-42). Если не принять должных мер, любое из этих исключений повлечет "зависание" компьютера. Исключение INT 11 можно предотвратить – достаточно сбросить в нуль бит контроля выравнивания (12h) в регистре EFLAGS (A.11-4). С этой целью в строках 226 – 235 регистр EFLAGS считывается в стек, исходное состояние 3-го считанного байта запоминается в отдельной ячейке памяти (03C4), прямо в стеке флаг выравнивания сбрасывается в нуль, и потом результат записывается обратно из стека в регистр EFLAGS.

Исключения INT 0D и INT 0E предотвратить нельзя, так как процессор неизбежно будет их генерировать при попытках доступа к защищенным областям памяти. Можно предусмотреть перехват этих исключений, но дело дополнительно осложнено тем, что при работе в реальном режиме их трудно отличить от вызовов прерываний, которые поступают от устройств по линиям IRQ 5 и IRQ 6 (8.01-09). Во избежание путаницы программа GS_dump.com на время вывода дампа запрещает поступление внешних вызовов по линиям IRQ 5 и IRQ 6. Для этого операции в строках 236 – 23D считывают маску порта 21h (A.14-1), сохраняют ее первоначальное состояние в ячейке 03C5, устанавливают в ней биты 05 и 06, и в таком виде отправляют ее обратно в порт 21h.

Обработчики исключений INT 0D и INT 0E в программе GS_dump.com заготовлены заранее в секции 13. Они представляют собой простую отсылку с коррекцией адреса возврата, который процессор оставляет в стеке. Коррекция исключает "зависание" в цикле исполнения одной и той же команды, а также обеспечивает правильное возобновление исполнения программы со следующей команды. Чтобы задействовать подготовленные обработчики исключений, операции в строках 23F и 243 секции 8 вписывают фактический сегментный адрес в ячейки памяти 03C8 и 03CC, предназначенные для хранения указателей на обработчики прерываний. Затем два вызова подпрограммы 03A0 меняют местами адреса из таблицы прерываний и те, которые были подготовлены в ячейках 03C8 и 03CC. С этого момента программа GS_dump.com готова правильно реагировать на генерацию процессором исключений INT 0D и INT 0E.

Поскольку правила доступа в адресном пространстве изменяются с дискретностью не менее 16 байт, постольку о доступности всех байтов одной строки дампа можно судить по одному пробному акту считывания, выполняемому в строке 257 секции 9. Когда запрошенный первый байт строки дампа доступен, тогда процессор, исполняя команду LODSB (7.03-53) в строке 257, изменяет на единицу смещение в регистре SI. Но когда запрос вызывает срабатывание защиты, тогда команда LODSB вообще не исполняется, и смещение в регистре SI приращения не получает. Операция в строке 258 сравнивает текущее значение смещения в регистре SI с предыдущим, сохраняемым в ячейке 03CE. Равенство значений свидетельствует о срабатывании защиты в процессоре.

Если срабатывание защиты в процессоре произошло, то любые попытки доступа к памяти в пределах ближайших 16 байт бесполезны. В таких строках дампа дальнейшие операции секции 9 выводят линейный адрес с сообщением о срабатывании защиты и вычисляют 32-битовое смещение первого байта для следующей строки дампа. Затем следует переход к проверке номера строки дампа в строке 286. Если данная строка дампа – не последняя, то произойдет возврат к началу процедуры пробного считывания в секции 9, где те же операции будут повторены для следующей строки дампа.

Если значения SI, сравниваемые в строке 258, неодинаковы, то в строке 25C произойдет переход к формированию строки дампа в секции 10. Оно включает индикацию базового линейного адреса вызовом подпрограммы 0349, цикл 26E – 273 для вывода на экран 16 байт дампа и цикл 278 - 27D для вывода тех же 16 байт в виде знаков кода ASCII. Если данная строка дампа – не последняя, то на шаге 289 произойдет возврат в секцию 9 к пробному считыванию следующей строки.

После вывода последней строки дампа программа переходит к операциям секции 11, восстанавливающим исходное состояние всех тех элементов, у которых оно было изменено. Двумя вызовами подпрограммы 03A0 восстанавливаются исходные адреса обработчиков прерываний INT 0D и INT 0E. Операции в строках 293 – 2A3 восстанавливают прежнюю маску порта 21h и прежнее состояние регистра EFLAGS. Операции в строках 2A4 – 2B6 восстанавливают прежнее состояние линии A20 шины адреса.

Когда все должное восстановлено, исполняются операции завершающей секции 12. При успешном завершении финальное сообщение не выводится, и код уровня ошибки устанавливается равным нулю. Но при неудачном завершении те же операции в строках 2BF – 2C4 используются для выведения сообщения об ошибке. Последним следует вызов функции завершения программы в строке 2C7.

Примечание 1: код уровня ошибки, который программа GS_dump.com оставляет в случае своего неудачного завершения, может представлять отдельный интерес. В частности, наибольший код уровня ошибки 8 программа GS_dump.com оставит при попытке исполнения в среде

"окна DOS" операционной системы Windows. Регистрация актов завершения с уровнем ошибки 8 (3.15-03) позволит предотвращать запуск таких программ, которые в "окне DOS" исполнять нельзя (например, программы Turn_off.com из раздела 9.05-02).

9.11 Альтернативные варианты загрузки MS-DOS7

В отношении обеспечения сохранности данных и основной операционной системы компьютера эксперименты в среде DOS наиболее безопасны тогда, когда DOS загружают с внешнего, отдельного дисковода или со сменного носителя записи – дискеты, флэш-карты или компакт-диска. Так обычно действуют при выездном обслуживании компьютеров. Но при систематическом пользовании DOS такой путь, описанный в разделе 9.11-01, недостаточно надежен, потому что в этой роли срок службы всех записываемых сменных носителей невелик.

Стационарную выборочную загрузку компьютера осуществляют с помощью особых программ (boot-менеджеров) или с помощью загрузочных модулей некоторых операционных систем. В разделах 9.11-02 и 9.11-03 рассмотрено использование с этой целью загрузочного модуля операционной системы Windows-XP, а также самой MS-DOS7: ведь она изначально была задумана как стартовая площадка для операционных систем Windows-95/98. "Способности" MS-DOS7 к исполнению роли boot-менеджера скромны, но у нее есть полезное свойство: совместимость с загружаемыми расширениями системы BIOS. Может быть, для выборочной загрузки компьютера Вам придется применить MS-DOS7.

9.11-01 Загрузка MS-DOS7 со сменных носителей.

Обычная дискета емкостью 1.44 Мбайт становится загрузочной, когда на нее записаны загрузочный сектор (boot-сектор) и системные файлы DOS. В среде MS-DOS7 это делает программа SYS.COM (6.24). В среде Windows Вы можете скачать самораспаковывающийся файл-образ загрузочной дискеты с MS-DOS7 из сети Интернет, например, с хоста http://1gighost.com/ed/jamiephiladelphia/ или с сайта http://anbcomp.com/files/bootdisk/ . Предпочтительны варианты без перебазирования MS-DOS7 на RAM-диск (файлы boot95b.exe, boot98c.exe, boot98sc.exe), потому что примененная в стандартных загрузочных дискетах процедура перебазирования на многих современных компьютерах работает некорректно. По той же причине не рекомендуется задавать формирование загрузочной дискеты в ходе форматирования. Но отмеченного недостатка нет у загрузочных дискет с MS-DOS8, которые формирует Windows-XP.

Получив обычную загрузочную дискету, Вы едва ли будете удовлетворены её скудным набором программ. Минимально-приемлемый комплект придется составлять самостоятельно, причем для MS-DOS7 многое можно взять из дистрибутива Windows-95/98. Образы загрузочных дискет с расширенными комплектами программ выложены, например, на сайтах http://www.multiboot.ru/ и http://www.netbootdisk.com/ . Рекомендации автора по составлению базовых комплектов выражены перечнем драйверов и программ в конфигурационных файлах, приведенных в разделах 6.25, 9.01, 9.04 и 9.09 данной книги. Разумеется, при любой комплектации все спецификации и пути вызова в конфигурационных файлах должны точно соответствовать наличию и размещению тех же драйверов и программ на дискете.

Перед загрузкой компьютера с дискеты нужно еще проверить настройки программы BIOS Setup (о ее вызове – в разделе 1.01). На странице "Main" программы BIOS Setup должен быть правильно указан тип имеющегося флоппи-дисковода. На странице "Boot" следует установить такой порядок загрузочных устройств ("Boot Device Priority"), чтобы флоппи-дисковод был бы опрошен раньше, чем жесткий магнитный диск. В компьютерах 1990-х годов порядок обычно задавался буквенными обозначениями дисков, и тогда список должен начинаться с буквы A:, обозначающей флоппи-дисковод. В современных компьютерах флоппи-дисковод ("1st Floppy Drive") надо поставить на первое место в списке устройств на сменных носителях ("Removable Drives"), и загрузке с них присвоить больший приоритет, чем загрузке с жестких магнитных дисков.

К сожалению, операционная система работает с дискеты очень медленно и притом интенсивно ее изнашивает. Ресурса дискеты хватит всего на 10 – 20 часов такой эксплуатации. Работа пойдет быстро и надежно, если конфигурационные файлы организуют перебазирование DOS с дискеты на RAM-диск (примеры – в разделах 9.04 и 9.09). Но и в последнем случае дискету нельзя признать удачным носителем для операционной системы. Ленивое перебазирование DOS с дискеты, аккомпанируемое скрежетом дисковода, вызывает раздражение. Помимо прочего, емкость дискеты по нынешним меркам слишком мала.

Оптические диски выглядят намного привлекательнее дискет и по емкости, и по скорости считывания, и по сроку службы. Однако отличной сохраняемостью обладают только оптические диски с постоянной сигналограммой, тиражируемые с матриц на промышленных технологических линиях. У тех "болванок", которые Вы записываете на своем дисководе, физическая основа сигналограммы совсем другая. Сигналограммы на них деградируют при считывании, так что их ресурс не намного отличается от ресурса дискет. По быстродействию разница тоже невелика, так как позиционирование головки у оптических дисководов занимает примерно 0.1 секунды, а именно оно лимитирует темп доступа к случайно размещенным файлам. Из-за малого ресурса и низкого темпа доступа перебазирование DOS с оптических дисков на RAM-диск не менее актуально, чем при загрузке с дискет.

Дополнительные сложности возникают из-за отличия файловых систем оптических дисков от тех, которые "понятны" ядру DOS. По этой причине DOS нельзя загрузить с оптического диска непосредственно. На оптический диск нужно заранее записать образ другого диска – загрузочного диска с "понятной" файловой системой FAT-12 или FAT-16. Система BIOS компьютера должна быть способна представить (эмулировать) этот образ как физический диск, и только тогда с него можно будет загрузить DOS. Если оригиналом для записанного образа послужила загрузочная дискета, то эмулируемый системой BIOS фиктивный диск станет диском A:, а реальный флоппи-дисковод получит буквенное обозначение B:.

Почти все программы записи оптических дисков позволяют создавать загрузочные оптические диски и способны считывать подлежащий записи образ с загрузочных дискет. Для исполнения этой роли вполне пригодны конфигурации, описанные в разделах 9.04 и 9.09, предусматривающие перебазирование DOS на RAM-диск (5.05) и загрузку драйверов оптических дисков (5.08-04 и 5.09-04). Обеспечивая доступ за пределы записанного на оптический диск образа дискеты, драйвера оптических дисков устраняют проблему ее недостаточной емкости.

Для загрузки с оптического диска оптический дисковод должен быть зарегистрирован как загрузочное устройство. Чтобы убедиться в этом, надо открыть страницу BOOT программы BIOS Setup. Там в списке загрузочных устройств ("Boot Device Priority") разные версии системы BIOS указывают либо класс прибора (CD-ROM), либо конкретный тип оптического дисковода. Там же надо установить такой порядок загрузочных устройств в списке, чтобы оптический дисковод был бы опрошен системой BIOS раньше, чем жесткий магнитный диск. Потом остается только сохранить новые настройки и успеть вставить загрузочный оптический диск в дисковод до начала стартовых тестов системы BIOS.

В современных компьютерах система BIOS может не зарегистрировать загрузочное устройство из-за неточностей в параметрах интерфейса. В частности, для установки параметров интерфейса IDE нужно сначала открыть страницу MAIN программы BIOS Setup, и из нее выйти на страницу "IDE Configuration". Там параметру "Onboard IDE Operate Mode" надо дать значение "Compatible Mode", а параметр "Combined Mode Option" привести в соответствие с типом шины подключенных приборов. Оптические дисководы внутреннего исполнения обычно снабжают параллельной шиной P-ATA. Если приборов с последовательной шиной S-ATA в компьютере нет, то параметру "Combined Mode Option" следует дать значение "P-ATA only". В иных случаях лучше выбрать значение "P-ATA+S-ATA". Еще нужно иметь ввиду, что внесенные изменения повлияют на список загрузочных устройств не сразу, а только после перезагрузки компьютера.

Чтобы установить параметры интерфейса USB, надо открыть страницу "Advanced" программы BIOS Setup, и из нее выйти на страницу "USB Configuration". Там параметр "USB Function" (или "USB Controller") должен быть

поставлен в состояние "Enabled". Этого достаточно для доступа к приборам на шине USB с помощью драйверов (5.07-05). Но те системы BIOS, которые способны загружать компьютер с шины USB, предоставляют дополнительные настройки. Так, в последних версиях BIOS фирмы AMI на той же странице имеется параметр "Legacy USB support". Для обеспечения загрузки ему надо присвоить значение "Enabled". Напомним, что это может вызвать конфликты с драйверами шины USB (5.07-05), так что строки их загрузки из конфигурационных файлов лучше убрать. Если имеются отдельные параметры для контроллера USB 2.0, то их тоже стоит поставить в положение "Enabled", а скорость обмена – в положение "HiSpeed".

Шина USB допускает довольно сложные варианты подключения устройств, и потому на регистрацию внешних загрузочных устройств может повлиять ограничение состава стартовых тестов. Чтобы снять это ограничение, надо открыть страницу BOOT программы BIOS Setup, из нее выйти на страницу "Boot Settings Configuration", и там поставить параметр "Quick Boot" в положение "Disabled". Длительность стартовых процедур, понятно, немного увеличится.

Когда на шине USB системой BIOS зарегистрировано хотя бы одно устройство, тогда в современных компьютерах программа BIOS Setup со своей страницы "USB Configuration" дает возможность открыть страницу "USB Mass Storage Device Configuration". На ней приведен список зарегистрированных устройств, и показан примененный к каждому из них способ эмуляции. Ошибки установления способа эмуляции могут служить еще одной причиной срыва процесса загрузки компьютера. По умолчанию способ эмуляции определяется автоматически, и тогда ошибки бывают вызваны тем, что носитель записи в подключенном USB-устройстве отсутствует, или не форматирован, или просто вставлен не слишком оперативно, когда стартовые тесты уже начались.

Как правило, способ эмуляции должен соответствовать классу устройства: накопитель на жестких магнитных дисках надо определить как "Hard Disk", оптический дисковод - как "CD-ROM", флоппи-дисковод – как "Floppy". Но иногда заранее неизвестно, к какому классу устройств отнести, например, флэш-карту. В таких случаях надо исходить из того, что флэш-карты емкостью от 512 Мбайт и выше форматируются как жесткий магнитный диск в процессе изготовления. А для флэш-карт меньшей емкости в современных системах BIOS предусмотрен особый вид эмуляции – "Forced FDD", при котором флэш-карта будет представлена как "Big Floppy" независимо от того, как она форматирована на самом деле.

Если Вы хотите изменить формат или заново нанести разметку носителя записи, то способ эмуляции обязательно должен точно соответствовать классу устройства. Оставлять определения "Forced FDD" или "Auto" в таких случаях нельзя. Новые, еще не размеченные жесткие магнитные диски регистрируются системой BIOS, но не получают буквенных обозначений. Для их разметки и форматирования лучше использовать современные программы, например, "Partition

Magic" версии не ниже 8.01. Один из первичных разделов надо обязательно сделать активным. После перезагрузки компьютера разделы получат буквенные обозначения, и тогда активный раздел можно будет сделать загрузочным с помощью программы SYS.COM (6.24). Конфигурация с перебазированием DOS на RAM-диск для жестких магнитных дисков не обязательна.

Большинство систем BIOS не принимает под свое управление те накопители, в которых носители записи не эмулированы, а реально форматированы как "Big Floppy", то есть без MBR. Обычно доступ к ним обеспечивается устанавливаемым драйвером, например, ASPIDISK.SYS (5.07-03, 5.07-05). Но на такие носители можно записать MBR с помощью программ, упоминаемых в примечании 5 к статье 6.13. После перезагрузки компьютера система BIOS примет носитель записи с MBR под свое управление как жесткий магнитный диск. С этого момента с ним можно обращаться как с жестким диском, в том числе можно сделать его загрузочным.

Когда хотя бы один накопитель воспринят системой BIOS как жесткий магнитный диск, тогда программа BIOS Setup со страницы BOOT разрешает выйти на страницу "Hard Disk Drives". Там приведен перечень всех имеющихся в компьютере дисководов на жестких магнитных дисках. Но только первый из них попадет в список загрузочных устройств. Если Вы хотите, чтобы компьютер загружался с внешнего дисковода, то его следует поставить на первое место на странице "Hard Disk Drives". То же наименование дисковода сразу появится в списке загрузочных устройств на странице "Boot Device Priority". Там жесткий магнитный диск не обязательно должен быть первым, но нужно, чтобы в момент загрузки в устройствах с большим приоритетом не было бы носителей записи.

Способ эмуляции "Hard Disk" годится для накопителей на сменных носителях, но при условии, что один и тот же носитель с форматом жесткого магнитного диска находится в накопителе постоянно с момента загрузки. Другой носитель в том же адаптере невозможно будет прочитать. Накопитель, эмулируемый как "Hard Disk", получит буквенное обозначение жесткого магнитного диска. Но если в момент загрузки он не содержит сменного носителя, то он не будет зарегистрирован, и система BIOS не выделит ему буквенного обозначения. Последнее бывает полезно, когда USB-адаптер имеет слоты для нескольких типов флэш-карт, которыми Вы не пользуетесь, и лишние буквы дисков не хотите им выделять.

Накопители, эмулируемые как "Floppy" или как "Forced FDD" получают буквенные обозначения независимо от того, имеются ли в них сменные носители в момент загрузки. Те современные системы BIOS, с которыми автору пришлось иметь дело, не позволяли заменять в них сменные носители и выделяли им только буквы A: и B:. Если в компьютере несколько таких накопителей, то буквенные обозначения получали первые два из списка на странице "Removable Drives" программы BIOS Setup, а остальные оказывались недоступны. Загрузочным может стать только тот накопитель, который Вы поставите на первое место в этом списке.

Сейчас широко распространены адаптеры для флэш-карт и твердотельные аналоги дисков, играющие роль накопителей с интерфейсом USB. Для загрузки компьютера с таких приборов они должны быть взяты под управление BIOS так же, как реальные дисковые накопители. Сначала надо задать способ эмуляции "Hard Disk". Если носитель записи оказывается доступен, значит, он форматирован как жесткий магнитный диск. Если носитель недоступен, следует предпочесть способ эмуляции "Forced FDD". В последнем случае надо проверить место соответствующего накопителя в списке устройств на странице "Removable Drives" программы BIOS Setup. Его надо поставить там на первое или на второе место, иначе он не получит буквенного обозначения, и к нему нельзя будет обращаться.

В носителях с форматом жесткого магнитного диска следует проверить, имеет ли первичный раздел статус активного (загрузочного) раздела. Поскольку программа "Partition Magic" работает только с реальными дисководами, постольку в твердотельных накопителях проверять и изменять статус раздела приходится с помощью старой программы FDISK.EXE, запускаемой с параметрами /fprmt и /actok (6.13). Многие версии систем BIOS не загружают компьютер из разделов с файловой системой FAT-12, однако программа FDISK.EXE не позволяет размечать иначе разделы, размер которых не превышает 16 Мбайт. При необходимости надо будет считать таблицу разделов (9.02-02), заменить в ней идентификатор файловой системы (A.13-6) с 01h на 06h, и потом записать таблицу разделов обратно (9.02-03). После такой подмены программа FORMAT.COM (6.15), запущенная с параметром /z:1, сформирует в этом разделе желаемую файловую систему FAT-16.

На следующем этапе подготовки загрузочного носителя на него можно записать boot-сектор и системные файлы с помощью программы SYS.COM (6.24), и далее формировать конфигурацию загрузки DOS. Предпочтительны конфигурации с перебазированием DOS на RAM-диск (9.04, 9.09), потому что у твердотельных ячеек памяти ресурс числа актов изменения состояния ограничен, и скорость доступа невелика. Согласно избранной конфигурации на носителе формируют структуру каталогов, и заполняют их необходимыми файлами.

Когда загрузочный твердотельный носитель сформирован, соответствующий ему накопитель надо поставить на первое место на странице "Hard Disk Drives" или на странице "Removable Drives" – в зависимости от того, какой способ эмуляции был применен. При необходимости на этом этапе способ эмуляции "Hard Disk" может быть заменен на "Forced FDD". Когда наименование твердотельного накопителя появилось на странице "Boot Device Priority" программы BIOS Setup, следует поставить его там на место с самым высоким приоритетом из всех устройств, которые будут готовы обеспечивать процесс загрузки. После выхода из программы BIOS Setup с сохранением параметров начнется процесс загрузки DOS с подготовленного твердотельного носителя.

9.11-02 Использование Windows-2000/XP для запуска MS-DOS7.

MS-DOS7 оставляет мало места для выбора: при старте с жесткого магнитного диска ей обязательно должен быть предоставлен первичный раздел с файловой системой FAT-16 или FAT-32. В отличие от того операционные системы Windows-2000/XP позволяют устанавливать себя и в первичные, и в не-первичные разделы с файловой системой FAT-32 или NTFS. Совсем безвыходных ситуаций не бывает. Даже если у Вас весь диск размечен как один раздел NTFS, можно программой Partition Magic высвободить место для DOS-раздела, а потом с помощью boot-менеджера (например, System Commander) выбирать ту или иную операционную систему. Но здесь будет рассмотрен вариант с использованием только собственного загрузочного модуля систем Windows-2000/XP.

Структуры разметки дисков, содержащие первичный раздел с файловой системой FAT-32, обычно бывают сформированы при установке операционных систем Windows-2000/XP "поверх" ранее установленной Windows-95/98 с сохранением возможности ее загрузки. Если после включения компьютера вслед за отчетом POST появляется загрузочное меню со строкой "Previous operating system", и если эта предыдущая операционная система именно Windows-95/98, значит, Вы имеете дело с одним из подобных вариантов структуры, и тогда для обеспечения альтернативной загрузки MS-DOS7 нужно править не загрузочные спецификации Windows-2000/XP, а сохраненные конфигурационные файлы Windows-95/98. Пример такой правки показан в разделе 9.11-03.

Если в ходе загрузки Windows-2000/XP загрузочное меню не появляется или не содержит строки "Previous operating system", то надо будет выяснить тип файловой системы на том диске, который назначен загрузочным в спецификациях программы BIOS Setup. Обычно – это диск C:. В окне программы Explorer (Проводник) нужно вывести контекстное меню загрузочного диска и в нем выбрать пункт "Properties" (Свойства). На открывшейся вкладке будет указан тип файловой системы. Если там написано "NTFS", то организовать альтернативную загрузку MS-DOS7 без boot-менеджера и без переразметки диска, видимо, нельзя. Но если там написано "FAT", то можно обойтись корректировкой записей в файле BOOT.INI. Конечно, помимо того потребуется перенести системные файлы MS-DOS7 и заново сформировать конфигурационные файлы для нее.

К правке записей в файле BOOT.INI существует официально предписанный путь: в меню "Start" ("Пуск") надо выбрать пункт "Settings" ("Настройка"), оттуда выйти в "Control Panel" ("Панель управления"), там выбрать папку "Performance and maintenance" ("Производительность и обслуживание"), потом – папку "System" ("Система"), в ней – вкладку "Advanced" ("Дополнительно"), там – нажать кнопку "Startup and Recovery Settings" ("Параметры загрузки и восстановления"), и тогда откроется окно, где, наконец, имеется кнопка "Edit" ("Правка"). В том же окне надо установить ненулевое время индикации загрузочного меню. По завершении правки

файл BOOT.INI следует сохранить, и потом дважды закрывать "окна" кнопкой "OK". Помимо того, изменять содержание файла BOOT.INI можно программой Msconfig.exe. Ее запускают из командной строки в окне, которое открывается при выборе пункта "Run" ("Выполнить") в меню кнопки "Start" ("Пуск").

Синтаксис в файле BOOT.INI такой же, как в файлах MSDOS.SYS (5.01-01) и CONFIG.SYS (9.04-01, 9.09-01). В каждой строке - отдельная спецификация. Ее наименование отделено от значения знаком равенства. Заголовки секций выделены квадратными скобками. В файле BOOT.INI две секции: в первой - параметры загрузки, а во второй - список установленных операционных систем. Ниже показан пример файла BOOT.INI, обеспечивающего загрузку трех операционных систем:

```
[boot loader]
timeout=30
default=multi(0)disk(0)rdisk(0)partition(3)\WINDOWS
[operating systems]
multi(0)disk(0)rdisk(0)partition(3)\WINDOWS="Microsoft Windows XP...
multi(0)disk(0)rdisk(0)partition(2)\WINNT="Microsoft Windows 2000...
C:\bootsect.dos="Microsoft DOS 7.10"
```

В представленном примере пятая и шестая строки обрезаны по размеру страницы, в реальном файле они длиннее. Но это не существенно: концы усеченных строк в реальном файле в любом случае не должны быть изменены. Из содержания файла понятно, что он обеспечивал раздельную загрузку операционных систем Windows-XP и Windows-2000, поставленных в разделы 2 и 3 жесткого магнитного диска, так что первый раздел диска оставался свободным. Оставлять первый загрузочный раздел свободным не обязательно, но полезно, так как это дает возможность установить MS-DOS7 отдельно от других операционных систем. Последняя строка в приведенном примере – именно та, которую Вам предстоит вписать самостоятельно для обеспечения загрузки MS-DOS7, в частности, с диска C:. Слова, заключенные в двойные кавычки, на процесс загрузки не повлияют, они представляют лишь название пункта меню.

Загрузчик NTLDR, интерпретирующий строки файла BOOT.INI, поймет вписанную Вами последнюю строку как команду искать в корневом каталоге загрузочного диска образ boot-сектора Bootsect.dos. Если его там нет, то его предстоит создать, причем в среде MS-DOS7, загрузившись с восстановительной дискеты для Windows-95/98. Сначала надо сохранить имеющийся boot-сектор загрузочного раздела жесткого магнитного диска в файл, как показано в разделе 9.02-01, затем – обновить содержимое boot-сектора с помощью программы SYS.COM (6.24), скопировать это новое содержимое в файл Bootsect.dos, и потом восстановить из файла прежнее содержимое boot-сектора, также по рекомендациям раздела 9.02-01. Вместе с обновлением boot-сектора программа SYS.COM

скопирует в корневой каталог файлы COMMAND.COM и IO.SYS. Загрузчику IO.SYS будет передано управление в результате исполнения кода boot-сектора.

Очередная Ваша задача состоит в том, чтобы все другие файлы, необходимые для MS-DOS7, тоже были бы найдены на своих должных местах. В частности, в корневом каталоге загрузочного диска должны быть следующие файлы:

Io.sys	– скрытый системный файл: загрузчик и ядро MS-DOS7;
Msdos.sys	– скрытый системный файл: параметры загрузки (5.01-01);
Config.sys	– конфигурационный файл (9.01-01, 9.04-01, 9.11-03);
Autoexec.bat	– конфигурационный файл (9.01-02, 9.04-02, 9.11-03);
Command.com	– файл только для чтения: командный интерпретатор.

Три конфигурационных файла из приведенного перечня – MSDOS.SYS, CONFIG.SYS и AUTOEXEC.BAT – Вам предстоит написать; в скобках в строках перечня указаны номера разделов данной книги, где даны образцы их составления.

Помимо файлов корневого каталога, для загрузки и последующей работы MS-DOS7 необходимы еще драйверы, упоминаемые в строках конфигурационных файлов, а также различные программы из числа описанных в главе 6. Всех их размещают в структуре каталогов DOS, которую также предстоит создать на том же диске. Примеры конфигурационных файлов, представленные в этой книге, написаны в расчете на одну и ту же структуру каталогов: драйверы помещены в каталог \DOS\DRV, файлы из поставки MS-DOS7 – в каталог \DOS\MS7, файловый менеджер – в каталог \DOS\VC4, все остальные файлы – в каталог \DOS\OTH. Конкретный вид структуры каталогов DOS и состав размещаемых в ней файлов Вы вправе задать иначе, но в любом случае они должны быть точно согласованы со всеми ссылками в примененных Вами конфигурационных файлах.

Какому примеру конфигурационных файлов следовать – тоже вопрос Вашего выбора. Иногда бывает достаточен простейший вариант, подобный показанному в разделе 9.01. Чаще полезны варианты с перебазированием MS-DOS7 на RAM-диск, подобные описанным в разделах 9.04 и 9.09. Специальные примеры конфигурационных файлов для желающих экспериментировать с DOS и другими операционными системами показаны в разделе 9.11-03. На основе упомянутых примеров несложно комбинировать такие конфигурации загрузки, которые наиболее точно отвечают стоящим перед Вами задачам.

Примечание 1: в ходе установки операционных систем Windows-2000/XP "поверх" установленных ранее MS-DOS7 или Windows-95/98 командному интерпретатору COMMAND.COM, который остается в корневом каталоге, оказываются приписаны атрибуты H (скрытый) и S (системный). Из-за них обращения к командному интерпретатору из batch-файлов не будут исполняться. Упомянутые атрибуты необходимо снять с помощью программы ATTRIB.EXE (6.01).

9.11-03 MS-DOS7 в роли boot-менеджера.

Операционные системы Windows-95/98 используют MS-DOS7 в процессе загрузки, но не раскрывают секретов отдельного конфигурирования MS-DOS7. Между тем раздельное конфигурирование вполне возможно, и оно полезно не только ради самой MS-DOS7, но и для осуществления выборочной альтернативной загрузки ряда других операционных систем, способных стартовать из среды MS-DOS7.

Приведенный здесь образец файла CONFIG.SYS предназначен для Windows-95/98, но, помимо того, обеспечивает загрузку двух конфигураций MS-DOS7 и еще двух других операционных систем. Для старта показанных загрузчиков операционных систем QNX и Linux необходим диск с файловой системой FAT-16. Если Вам нужна только альтернативная загрузка MS-DOS7 и Windows-95/98, то все строки, относящиеся к системам QNX и Linux, нужно просто исключить, и тогда можно будет использовать загрузочный диск с файловой системой FAT-32.

Файл CONFIG.SYS начинается с секции [menu]. Выбор пункта меню направляет процесс интерпретации к секциям [L08] – [L25]: каждая из них соответствует одной из альтернатив. Секции названы по именам меток в файле Autoexec.bat. Пустые строки между секциями введены только для удобства восприятия, при наборе текста файла их можно пропускать.

```
[menu]
numlock off
menuitem=L08, Real mode MS-DOS7
menuitem=L09, Protected mode MS-DOS7
menuitem=L16, Microsoft's Windows-98
menuitem=L24, QNX v.6.0
menuitem=L25, Linux Slackware v.3.5
menudefault=L16,20

[L08]
device=\DOS\DRV\Himem.sys /v
device=\DOS\DRV\Umbpci.sys
include=S08
include=S09

[S08]
accdate c- d- e-
dos=high,umb,noauto
buffershigh=30,0
fileshigh=30,0
```

```
lastdrivehigh=Z
multitrack=On
fcbshigh=1,0
stackshigh=9,256

[L09]
device=\DOS\DRV\Himem.sys /v
device=\DOS\DRV\Emm386.exe ram v
include=S08
include=S09

[S09]
country=007,866,C:\DOS\DRV\Country.sys
devicehigh=\DOS\DRV\Dblbuff.sys
devicehigh=\DOS\DRV\Ifshlp.sys
devicehigh=\DOS\DRV\Setver.exe
devicehigh=\DOS\DRV\Atapimgr.sys /W:6 /T:5 /LUN
devicehigh=\DOS\DRV\Oakcdrom.sys /D:CD001
installhigh=\DOS\DRV\Shsucdx.com /D:CD001 /L:N /~+ /R /Q

[L16]
device=\WINDOWS\Himem.sys
include=S08
Country=007,866,C:\WINDOWS\COMMAND\Country.sys
devicehigh=\WINDOWS\Dblbuff.sys
devicehigh=\WINDOWS\Ifshlp.sys
devicehigh=\WINDOWS\Setver.exe
devicehigh=\WINDOWS\COMMAND\Display.sys con=(ega,,1)

[L24]
device=\QNX\boot\bin\loadqnx.sys C:\QNX\boot\fs\qnxbas.ifs

[L25]
device=\DOS\DRV\Himem.sys
include=S08
install=\linux\loadlin.exe @\linux\linparam.scr

[common]
installhigh=\DOS\DRV\Mouse.com
shell=C:\COMMAND.COM C:\ /E:2016 /L:511 /U:255 /p
```

Секции [L08] и [L09] в приведенном файле CONFIG.SYS загружают MS-DOS7 как самостоятельную операционную систему, но [L09] обеспечивает доступ к области UMB как обычно – с помощью драйвера EMM386.EXE (5.04-02), а [L08] –

с помощью драйвера UMBPCI.SYS (5.04-04) без перевода процессора в защищенный режим. Последний вариант необходим для работы с программами реального режима, такими как DUSE.EXE (5.07-05) или GS_limit.com (9.10-01).

При всех вариантах загрузки каждая операционная система пользуется отдельной структурой каталогов, наличие которой не обязательно, но желательно, потому что взаимозависимость содержания каталогов нескольких операционных систем сделала бы многовариантную загрузку менее надежной.

Секция [L16] для загрузки операционной системы Windows-95/98 содержит ряд спецификаций, которые обычно принимаются по умолчанию, но здесь написаны явно для обеспечения совместимости с обособленной загрузкой MS-DOS7. Пути в секции [L16] соответствуют обычной файловой структуре, которая создается автоматически при установке операционной системы Windows-95/98 на диск. Но если на Вашем компьютере структура каталогов другая, то пути в секции [L16] надо будет привести в соответствие с ней.

При выборе пунктов меню [L24] или [L25] управление передается загрузчикам UNIX-подобных операционных систем, которые не возвращают управление обратно к MS-DOS7. В таких случаях команды в секции [common] не будут исполнены, и единственный путь обратно лежит через команду SHUTDOWN с последующей перезагрузкой. Пути, указанные в строках секций [L24] и [L25], отражают ту структуру каталогов, которая создается на текущем диске в ходе распаковки пакетов поставки соответствующих операционных систем.

При выборе пунктов меню [L08], [L09] или [L16] интерпретация команд продолжится в секции [common]. В последней строке команда SHELL запустит интерпретатор COMMAND.COM. Он продолжит процесс загрузки согласованным исполнением команд из последнего конфигурационного файла – AUTOEXEC.BAT.

Файл AUTOEXEC.BAT получается простым, потому что варианты [L08] и [L09] обрабатываются одинаково, и множество альтернатив сужается до двух. Конкретное содержание файла AUTOEXEC.BAT может выглядеть, например, так:

```
@echo off
prompt $p$g
set dsk=C:
if not exist %dsk%\Temp\nul md %dsk%\Temp
set Temp=%dsk%\Temp
set dircmd= /A /O:GNE /P
goto %CONFIG%
:L08
:L09
Lh %dsk%\DOS\DRV\Keyrus.com
path ;
set VC=%dsk%\DOS\VC4
```

```
path=%VC%;%dsk%\DOS\OTH;%dsk%\;%dsk%\DOS\MS7
Vc.com /TSR /no2E /noswap
goto L25
:L16
path=%dsk%\WINDOWS;%dsk%\WINDOWS\COMMAND
Mode.com con codepage prepare=((866) %dsk%\WINDOWS\COMMAND\Ega3.cpi)
Mode.com con codepage select=866
Lh Keyb.com ru,866,%dsk%\WINDOWS\COMMAND\Keybrd3.sys
echo.
echo Loading Windows-98. Wait...
Win.com
:L24
:L25
```

В этом варианте файла AUTOEXEC.BAT строки 2 – 6 представляют общую часть с присвоением значений обычным переменным окружения. Важно не оставлять пробелов в конце 3-й и 5-й строк с присвоением значений переменным DSK и TEMP. В 7-й строке выполняется переход на метку, определяемую значением переменной CONFIG. Значение этой переменной неявно задает загрузчик IO.SYS в ходе интерпретации секции [menu] файла CONFIG.SYS, причем значением становится имя избранного пункта меню. Поскольку до исполнения файла AUTOEXEC.BAT дело доходит только в случаях выбора пунктов меню L08, L09 (MS-DOS7) и L16 (Windows-95/98), постольку переход из 7-й строки может произойти только на метки :L08, :L09 или :L16.

После меток :L08 и :L09 следует группа команд, обеспечивающая загрузку MS-DOS7. Команды этой группы задают пути, специфичные для MS-DOS7, и вызывают на исполнение файл-менеджер Volcov Commander.

После метки :L16 следует группа команд, аналогичным образом обеспечивающая загрузку операционной системы Windows-95/98. Здесь в переменную PATH записываются другие пути, специфичные для Windows-95/98. Следует также обратить внимание на применение тривиального варианта национальной адаптации, чтобы обеспечить правильное переключение кодировок во всех видеорежимах индикации "окна DOS". В заключительных строках инициируется загрузка Windows-95/98 посредством вызова на исполнение загрузчика – файла WIN.COM. Выведение логотипа Windows на экран дисплея не предусмотрено, вместо него на экране появится текстовое сообщение, которое по окончании загрузки сменится привычным изображением "рабочего стола".

Примечание 1: хотя операционная система Windows-XP не приспособлена для старта в среде MS-DOS7, тем не менее все необходимые условия для ее старта могут быть подготовлены с помощью программы Dostowxp.com. Вариант этой программы, модифицированный

В.Ашумовым, способен инициировать загрузку систем Windows Vista и Windows-7. И исходный, и модифицированный варианты программы выложены на сайте http://www.multiboot.ru/files.htm . Благодаря им Вы сможете запускать заранее установленные современные версии операционной системы Windows так же, как и другие операционные системы, упоминаемые в разделе 9.11-03.

Примечание 2: установка операционной системы Linux выполняется путем рекомпиляции ее ядра с теми драйверами, которые нужны для обслуживания конкретного компьютера. В сети Интернет на сервере ftp://ftp.wolfmountaingroup.org/pub/linuxware/ выложен архив linuxware-09072008.tar.gz, позволяющий рекомпилировать ядро 2.4 Linux в виде обычной программы для DOS. Это рекомпилированное ядро запускает Linux из среды DOS, сохраняет DOS, и по окончании работы Linux обеспечивает возврат обратно в DOS. Ядро 2.4 используется в версиях 8 – 10 Mandrake Linux и во многих других современных разновидностях Linux.

Приложения

A.01 Основные системные структуры данных

И система BIOS, и MS-DOS хранят необходимые им данные в выделенных областях памяти компьютера. Размещение данных в этих областях не является строго определенным, оно может зависеть как от версии системы BIOS, так и от версии MS-DOS. В общем случае доступ к системным данным должен осуществляться не путем прямой адресации, а через вызовы функций, описанных в главе 8. Помимо прочего, системные данные не будут обновлены должным образом, если не будет вызвана обслуживающая их функция.

Тем не менее прямой доступ к системным данным бывает нужен. Он позволяет узнать больше, чем разрешено узнавать пользовательским программам через служебные функции. Для поиска причин неполадок бывает нужно видеть данные такими, какими они были, без обновления. Иногда требуется вмешаться, изменить имеющиеся установки, чтобы спровоцировать желательные последствия. Конечно, все такие действия Вы можете предпринимать только на свой собственный риск, но они способны дать Вам шанс, который без риска получить нельзя.

A.01-1. Область данных BIOS

Сразу после включения компьютера система BIOS начинает собирать сведения о нем и формировать свою область данных. В AT-совместимых компьютерах область данных BIOS занимает 100h байт, начиная с адреса 0040:0000h. Приведенная ниже таблица дает общее представление о расположении данных в области данных BIOS со ссылками на другие таблицы, где более детально показаны сведения, касающиеся флоппи-дисков (A.08-1), видеосистемы (A.10-6), состава оборудования (A.11-1) и состояния клавиатуры (A.02-3).

Смещение	Длина	Содержание
00h	2	Базовый адрес ввода-вывода для порта COM-1
02h	2	Базовый адрес ввода-вывода для порта COM-2
08h	2	Базовый адрес ввода-вывода для порта LPT1
0Eh	2	Сегмент данных BIOS (0000h, если его нет)
10h	2	Состав оборудования, подробнее в A.11-1
12h	1	Отчет о результатах теста POST
13h	2	Размер базовой памяти в килобайтах
17h	39	Буфер и флаги клавиатуры, подробнее в A.02-3

Продолжение таблицы A.01-1

3Eh	7	Статус флоппи-дисководов, подробнее в A.08-1
49h	22	Данные видеорежима, подробнее в A.10-6
67h	4	Адрес рестарта (примечание 4 к A.12-1)
6Ch	4	Счет времени в тактах, начиная с полуночи
70h	1	Счет суток, сбрасываемый от INT 1A\AH=00h
71h	1	Бит 7 установлен после нажатия Ctrl-Break
72h	2	Предписываемые действия POST (примечание 1)
74h	1	Код завершения последней операции (A.06-1)
75h	1	Число дисководов на жестких магнитных дисках
77h	1	Адрес порта для жестких магнитных дисков
78h	1	Счетчик времени ожидания для порта LPT 1
7Ch	1	Счетчик времени ожидания для порта COM 1
7Dh	1	Счетчик времени ожидания для порта COM 2
80h	4	Адреса буфера клавиатуры (A.02-3)
84h	8	Регистры видеорежима, подробнее в A.10-6
8Ch	3	Статус контроллера жестких магнитных дисков
8Fh	7	Данные флоппи-контроллера, подробнее в A.08-1
96h	2	Статус клавиатуры, подробнее в A.02-3
98h	4	Указатель на флаг ожидания INT 15\AX=8300h
9Ch	4	Заданное время ожидания в микросекундах
A0h	1	Флаги системного таймера: бит 0: произведен вызов INT 15\AH=86h бит 7: время ожидания истекло
CEh	2	Счет дней с последнего включения компьютера
F0h	16	Область обмена данными для программ.

Примечание 1: при перезагрузке путем перехода на адрес F000:FFF0h (примечание 4 к A.12-1) компьютер может выполнять тест POST по-разному в зависимости от слова, заранее записанного по адресу 0040:0072h:

0000h – "холодная" перезагрузка (с проверкой памяти)

1234h – "теплая" перезагрузка (без проверки памяти)

Примечание 2: поскольку размещение сведений в области данных BIOS может зависеть от версии BIOS, постольку Вам предстоит решать, являются ли данные, считанные по определенному адресу, в действительности теми данными, которые Вы ожидаете там найти.

A.01-2 Избранные данные из Списка Списков MS-DOS

Список Списков представляет собой одну из базовых системных структур, создаваемую загрузчиком IO.SYS в момент начала загрузки MS-DOS. Указатель на начало Списка Списков можно получить с помощью INT 21\AH=52h (8.02-59). Рис. 10 (в разделе A.03-3) иллюстрирует процесс доступа к Списку Списков.

Сведения о назначении отдельных данных в Списке Списков показаны в приведенной ниже таблице.

Смещение	Длина	Содержание
– 02h	2	Сегмент дескриптора (A.12-7) 1-го блока памяти
00h	4	Указатель на первый блок DPB (примечание 1).
04h	4	Указатель на SFT (System File Table, A.01-4)
0Ch	4	Указатель на драйвер устройства CON (Console)
10h	2	Предельный размер сектора на любом диске
16h	4	Адрес первой записи в таблице CDS (A.03-3)
20h	1	Число зарегистрированных дисководов
21h	1	Число записей в таблице CDS (A.03-3)
22h	18	Заголовок драйвера NUL (примечание 2)
34h	1	Число виртуальных дисков программы Join.exe
37h	4	Указатель на таблицу драйвера SETVER.EXE
3Dh	2	Сегмент PSP последней исполненной программы
43h	1	Загрузочный диск (01h = A:, 03h = C:, и т.д.)
45h	2	Размер расширенной памяти в килобайтах.

Примечание 1: блоки DPB (Drive Parameter Blocks, A.03-1) организованы в цепочку, так что указатель на каждый следующий блок DPB находится в ячейке со смещением 19h в предыдущем блоке DPB.

Примечание 2: заголовок устройства NUL – первый в цепи заголовков драйверов. Первые 4 байта в заголовке каждого драйвера заняты указателем на заголовок следующего драйвера. Указатель на заголовок второго драйвера также можно получить с помощью INT 2F\AX=122Ch. У последнего драйвера заголовок начинается со слова FFFFh.

А.01-3. Избранные записи в области SDA

Адрес области SDA (Swappable Data Area) и ее размер можно узнать с помощью INT 21\AX=5D06h (8.02-70). В этой области MS-DOS содержит текущие данные, характеризующие ее состояние в каждый конкретный момент, в том числе основные системные стеки. В зависимости от размера стеков область SDA может занимать до нескольких килобайт.

Сохранение и последующее восстановление всей области SDA представляют собой основной механизм обеспечения реентерабельности DOS, то есть возможности вызова функций DOS обработчиками прерываний, вызванными во время исполнения функции DOS. Если в момент вызова обработчик прерывания обнаруживает, что флаг критической ошибки и флаг InDOS (в байтах области SDA со смещениями 00h и 01h) не сброшены в нуль, значит, данный вызов прервал

исполнение функции DOS, и тогда следующий вызов функции DOS может изменить данные в области SDA так, что возврат к продолжению исполнения прерванной программы станет невозможен. Хотя иногда возврат все же возможен (8.02-28, 8.02-87), тем не менее наиболее универсальный выход состоит в том, чтобы перед вызовом функций DOS записать состояние области SDA, о после окончания их исполнения восстановить состояние области SDA по сохраненной записи. Еще нужно иметь ввиду, что функция INT 21\AX=5D06h (8.02-70) тоже не-реентерабельна, и потому определить адрес области SDA надо заранее, в процессе инициализации обработчика прерывания, а в момент прерывания следует лишь пользоваться заранее подготовленным адресом.

Приводимая ниже таблица показывает размещение избранных данных в области SDA.

Смещение	Длина	Содержание
00h	1	Флаг критической ошибки ("ErrorMode")
01h	1	Флаг InDOS (INT 21\AH=34h, 8.02-28)
02h	1	Диск, вызвавший ошибку (FFh - ошибки нет)
03h	1	Место возникновения ошибки (A.06-4)
04h	2	Код завершения последней операции (A.06-1)
06h	1	Способ исправления ошибки (A.06-3)
07h	1	Класс последней ошибки (A.06-2)
08h	4	Состояние ES:DI в момент последней ошибки
0Ch	4	Адрес области DTA (8.02-16)
10h	2	Сегмент PSP - идентификатор текущего процесса
14h	2	Уровень ошибки (errorlevel) последнего процесса
16h	1	Номер текущего логического диска
17h	1	Флаг BREAK (3.01, 4.02, 8.02-25)
2Ah	1	Флаг ответа Fail при вызове INT 24 (8.02-84)
2Bh	1	Разрешенные действия INT 24 (8.02-84)
30h	1	День месяца
31h	1	Месяц
32h	2	Год, отсчитываемый от 1980 года
34h	2	Число дней с 1 января 1980 года
36h	1	День недели (0 = воскресенье)

A.01-4 Структура блоков описания в таблицах SFT

Взаимное соответствие между номерными ссылками и теми "открытыми" объектами, на которые они ссылаются, задается последовательностью системных файловых таблиц SFT (System File Tables). Название SFT не вполне корректно,

потому что объектами ссылок могут быть не только файлы, но также области памяти, каналы посылки данных и т.п.

Указатель на первую таблицу SFT помещается в ячейку со смещением 04h в "Списке Списков" (A.01-2). Таблицы SFT образуют цепь: первые 4 байта в каждой таблице содержат указатель на начало следующей таблицы SFT, за исключением последней таблицы, которая начинается со слова FFFFh. Слово со смещением 04h в каждой таблице SFT объявляет число блоков описания, содержащихся в данной таблице. Каждый блок описания соответствует одному "открытому" объекту. Общее число блоков описания во всех таблицах SFT ограничено спецификацией команды FILES (4.12) в конфигурационном файле CONFIG.SYS.

Номера блоков описания, "открытых" для конкретной программы, записываются в таблицу JFT, которая размещена начиная со смещения 18h в префиксе сегмента (PSP, A.07-1) этой программы. Соответствующие "открытые" объекты адресуются посредством номерных ссылок (INT 21\AH=3Dh, 8.02-33), которые идентифицируют блоки описания в таблицах SFT по порядку размещения их номеров в таблице JFT. Номер блока описания в таблице SFT, соответствующий определенной ссылке, можно найти с помощью INT 2F\AX=1220h (8.03-11). Затем по найденному номеру блока описания с помощью INT 2F\AX=1216h (8.03-09) можно получить адрес этого блока описания в таблице SFT. Некоторые сведения о блоках описания в таблицах SFT и соответствующих им объектах можно получить с помощью INT 21\AX=4400h (8.02-40).

Первые три блока описания в таблице SFT по умолчанию связаны с определенными объектами: 00h – с каналом AUX в порт COM1, 01h – с каналами драйвера устройства CON (консоли), 02h – с каналом PRN в порт LPT1. Эти три номера автоматически записываются в таблицу JFT каждой программы, причем порядок их размещения в JFT (01h, 01h, 01h, 00h, 02h) определяет соответствующие им номерные ссылки: 0000h – для канала STDIN, 0001h – для канала STDOUT, 0002h – для канала STDERR, 0003h – для порта COM1, 0004h – для порта LPT1. Блоки описания в таблицах SFT, которым соответствуют номерные ссылки от 0005h и далее, создаются по запросу обработчиками прерываний INT 21\AH=3Dh (8.02-33) и INT 21\AX=6C00h (8.02-78).

Первый блок описания в каждой таблице SFT начинается со смещения 06h. Начальные смещения всех следующих блоков описания несложно рассчитать, так как каждый из них имеет постоянную длину 3Bh. Приведенная ниже таблица показывает размещение данных в любом отдельном блоке описания с отсчетом смещений от его начала, причем первая колонка "CDE" (= character device entries) показывает размещение данных в блоках описания не-файловых объектов, а вторая колонка "OFE" (= ordinary file's entries) показывает размещение данных в блоках описания обычных файлов.

CDE	OFE	Длина	Содержание
00h	00h	2	Число ссылок на объект (FFFFh - нет ссылок)
02h	02h	1	Условия доступа (A.09-4)
	03h	1	Флаги (примечание 1)
	04h	1	Атрибуты файла (A.09-2)
05h	05h	2	Информационное слово объекта (A.04-2 для файлов, A.05-2 для не-файлов)
07h	07h	4	Адрес блока DPB (A.03-1) для файлов или адрес заголовка драйвера (A.05-1) для не-файлов
	0Bh	2	Номер 1-го кластера (для локальных файлов)
	0Dh	2	Время последнего изменения (8.02-63)
	0Fh	2	Дата последнего изменения (8.02-63)
	11h	4	Размер файла
	15h	4	Положение указателя в файле (8.02-38)
	19h	2	Относительный номер кластера, к которому производилось предыдущее обращение
19h		4	Адрес блока данных IFS
	1Bh	4	Сектор каталога с записью о данном файле
	1Fh	1	Номер записи о файле в секторе каталога
20h	20h	11	Имя объекта в FCB-формате (A.09-5)
	31h	2	Сегмент PSP программы, открывшей объект
	35h	2	Абсолютный номер кластера, к которому производилось предыдущее обращение
37h		4	Адрес IFS-драйвера (или 0000:0000h без IFS)

Примечание 1: байт флагов в ячейке со смещением 03h включает все, что задается в регистре BH при вызове INT 21\AX=6C00h (8.02-78) и, кроме того, бит 7, устанавливаемый для тех файлов, которые открыты посредством блоков FCB (File Control Blocks).

A.02 Коды клавиатуры и национальная адаптация

A.02-1 Коды клавишей

Задача достижения совместимости компьютера с разными типами клавиатур возложена на аппаратные средства материнской платы и на систему BIOS, потому что управление с клавиатуры должно быть обеспечено всегда, даже когда операционная система не загружена. С этой целью сразу после включения компьютера система BIOS загружает обработчики прерываний INT 09 и INT 16, обслуживающие операции, связанные с клавиатурой.

При каждом нажатии на клавишу и при каждом отпускании клавиши контроллер клавиатуры принимает и преобразует поступающие от клавиатуры сигналы, а затем выставляет код в порт 60h и посылает запрос по линии IRQ 01, вызывающий на исполнение обработчик прерывания INT 09. Некоторые клавиши заставляют контроллер передавать последовательность кодов, вызывая обработчика INT 09 несколько раз подряд. Код, считываемый обработчиком из порта 60h, обычно является скэн-кодом конкретной клавиши, но может оказаться служебным кодом. К служебным кодам относятся 00h, AAh, а также все коды от E0h до FFh. Для идентификации нажатия клавишей особое значение имеют два служебных кода, называемые префиксами:

E0h – префикс для различения клавиш, которым ради сохранения совместимости со старыми 84-клавишными клавиатурами дан одинаковый скэн-код.

E1h – префикс для клавиш с двухбайтовым скэн-кодом. Обычно такая клавиша только одна: Pause/Break (примечание 6 к А.02-1).

Считывание префикса подготавливает обработчик прерывания INT 09 к особой интерпретации того скэн-кода, который будет передан ему при следующем вызове. Во 2-м столбце (INT 09) приводимой ниже таблицы показаны шестнадцатеричные скэн-коды нажатия клавиш, считываемые в момент вызова обработчиком прерывания INT 09. Коды отпускания отличаются от кодов нажатия только установленным в единицу старшим (седьмым) битом, и в таблице не показаны, так как их легко вывести из кодов нажатия (например, 1Eh – нажатие клавиши "A", а 9Eh – ее отпускание). Но те скэн-коды, которым предшествует передача префикса E0h или E1h, приведены во втором столбце таблицы совместно с этим префиксом, причем тот же префикс обычно предшествует и коду отпускания тех же клавишей.

Обработчики прерываний INT 09 и INT 16 преобразуют считанный из порта скэн-код в унифицированный скэн-код и в значение кода ASCII, соответствующие нажатой клавише. Именно эта пара значений предоставляется любой программе, которая пошлет запрос на ввод данных с клавиатуры посредством прерывания INT 16. Унифицированный скэн-код, как правило, повторяет скэн-код нажатия, но может быть изменен, если одновременно удерживалась нажатой какая-либо из "функциональных" клавишей: Shift, Ctrl или Alt. Каждая "функциональная" клавиша имеет свой скэн-код, все они показаны во второй колонке таблицы и принимаются во внимание обработчиком прерывания INT 09, но в буфер клавиатуры не заносятся. Состояние "функциональных" клавишей отображается иначе – через слово статуса, возвращаемое при вызове INT 16\AH=12h (8.01-85).

По номенклатуре клавишей приведенная ниже таблица соответствует широко распространенной 104-клавишной "улучшенной" клавиатуре. Слово "num" перед названием клавиши в первом столбце таблицы является отличительным признаком клавишей из цифровой группы в правой части клавиатуры, причем значения кодов для таких клавишей указаны при выключенном состоянии переключателя

NumLock. Клавиши упоминаются в порядке возрастания исходного скэн-кода (во второй колонке).

Шестнадцатеричные числа в 3-й – 6-й колонках таблицы представляют данные, которые обработчик прерывания INT 16\AH=10h (8.01-83) возвращает в регистре AX. Левые две цифры в каждом четырехзначном числе – это унифицированный скэн-код, возвращаемый в AH, а правые две цифры – это код ASCII соответствующего знака, возвращаемый в AL. Данные третьей колонки (AX) представляют одиночное нажатие, не сопровождаемое удержанием какой-либо "функциональной" клавиши. Данные в четвертой колонке (SHIFT) соответствуют случаю, когда удерживается нажатой "функциональная" клавиша SHIFT, данные в пятой колонке (CTRL) – случаю удержания "функциональной" клавиши CTRL, данные в шестой колонке (ALT) – случаю удержания "функциональной" клавиши ALT. Если в ячейке таблицы число не указано, значит, соответствующая клавишная комбинация не отображается обработчиком прерывания INT 16\AH=10h.

Клавиша	INT09	AX	SHIFT	CTRL	ALT	Примечания
Esc	01	011B	011B	011B	0100	1
1 !	02	0231	0221		7800	
2 @	03	0332	0340	0300	7900	
3 #	04	0433	0423		7A00	
4 $	05	0534	0524		7B00	
5 %	06	0635	0625		7C00	
6 ^	07	0736	075E	071E	7D00	
7 &	08	0837	0826		7E00	
8 *	09	0938	092A		7F00	
9 (	0A	0A39	0A28		8000	
0)	0B	0B30	0B29		8100	
– _	0C	0C2D	0C5F	0C1F	8200	
= +	0D	0D3D	0D2B		8300	
Backspace	0E	0E08	0E08	0E7F	0E00	1
Tab	0F	0F09	0F00	9400	A500	1, 2
Q	10	1071	1051	1011	1000	
W	11	1177	1157	1117	1100	
E	12	1265	1245	1205	1200	
R	13	1372	1352	1312	1300	
T	14	1474	1454	1414	1400	
Y	15	1579	1559	1519	1500	
U	16	1675	1655	1615	1600	
I	17	1769	1749	1709	1700	
O	18	186F	184F	180F	1800	

Продолжение таблицы A.02-1

P	19	1970	1950	1910	1900	
[{	1A	1A5B	1A7B	1A1B	1A00	1
] }	1B	1B5D	1B7D	1B1D	1B00	1
Enter	1C	1C0D	1C0D	1C0A	1C00	1
num Enter	E0 1C	E00D	E00D	E00A	A600	1, 3
Left Ctrl	1D					4
Right Ctrl	E0 1D					4
A	1E	1E61	1E41	1E01	1E00	
S	1F	1F73	1F53	1F13	1F00	
D	20	2064	2044	2004	2000	
F	21	2166	2146	2106	2100	
G	22	2267	2247	2207	2200	
H	23	2368	2348	2308	2300	
J	24	246A	244A	240A	2400	
K	25	256B	254B	250B	2500	
L	26	266C	264C	260C	2600	
; :	27	273B	273A		2700	1
' "	28	2827	2822		2800	1
` ~	29	2960	297E		2900	1
Left Shift	2A					4
SysRq	E0 2A			7200		5
\ |	2B	2B5C	2B7C	2B1C	2B00	1
Z	2C	2C7A	2C5A	2C1A	2C00	
X	2D	2D78	2D58	2D18	2D00	
C	2E	2E63	2E43	2E03	2E00	
V	2F	2F76	2F56	2F16	2F00	
B	30	3062	3042	3002	3000	
N	31	316E	314E	310E	3100	
M	32	326D	324D	320D	3200	
, <	33	332C	333C		3300	1
. >	34	342E	343E		3400	1
/ ?	35	352F	353F		3500	1
num /	E0 35	E02F	E02F	9500	A400	1, 2 ,3
Right Shift	36					4
num *	37	372A	372A	9600	3700	1, 2
Left Alt	38					4
Right Alt	E0 38					4
Spacebar	39	3920	3920	3920	3920	
Caps Lock	3A					4
F1	3B	3B00	5400	5E00	6800	
F2	3C	3C00	5500	5F00	6900	
F3	3D	3D00	5600	6000	6A00	

Продолжение таблицы A.02-1

F4	3E	3E00	5700	6100	6B00	
F5	3F	3F00	5800	6200	6C00	
F6	40	4000	5900	6300	6D00	
F7	41	4100	5A00	6400	6E00	
F8	42	4200	5B00	6500	6F00	
F9	43	4300	5C00	6600	7000	
F10	44	4400	5D00	6700	7100	
NumLock	45					4
Pause	E1 1D 45					4, 6
ScrollLock	46					4
num 7	47	4700	4737	7700	0007	7
Home	E0 47	47E0	47E0	77E0	9700	1, 3
num 8	48	4800	4838	8D00	0008	2, 7
Arrow Up	E0 48	48E0	48E0	8DE0	9800	1, 2, 3
num 9	49	4900	4939	8400	0009	7
PgUp	E0 49	49E0	49E0	84E0	9900	1, 3
num –	4A	4A2D	4A2D	8E00	4A00	1, 2
num 4	4B	4B00	4B34	7300	0004	7
LeftArrow	E0 4B	4BE0	4BE0	73E0	9B00	1, 3
num 5	4C	4C00	4C35	8F00	0005	2, 7
num 6	4D	4D00	4D36	7400	0006	7
RightArrow	E0 4D	4DE0	4DE0	74E0	9D00	1, 3
num +	4E	4E2B	4E2B	9000	4E00	1, 2
num 1	4F	4F00	4F31	7500	0001	7
End	E0 4F	4FE0	4FE0	75E0	9F00	1, 3
num 2	50	5000	5032	9100	0002	2, 7
ArrowDown	E0 50	50E0	50E0	91E0	A000	1, 2, 3
num 3	51	5100	5133	7600	0003	7
PgDn	E0 51	51E0	51E0	76E0	A100	1, 3
num 0	52	5200	5230	9200		2, 7
Ins	E0 52	52E0	52E0	92E0	A200	1, 2, 3
num .	53	5300	532E	9300		2, 7
Del	E0 53	53E0	53E0	93E0	A300	1, 2, 3
F11	57	8500	8700	8900	8B00	8
F12	58	8600	8800	8A00	8C00	8
LeftWin	E0 5B	B6E0	C2E0	CEE0	DAE0	8
RightWin	E0 5C	B7E0	C3E0	CFE0	DBE0	8
Menu	E0 5D	B8E0	C4E0	D0E0	DCE0	8

Примечание 1: обработчик прерывания INT 16\AH=00h не отвечает на комбинацию нажатий этой клавиши и "функциональной" клавиши ALT.

Примечание 2: обработчик прерывания INT 16\AH=00h не отвечает на комбинацию нажатий этой клавиши и "функциональной" клавиши CTRL.

Примечание 3: обработчик прерывания INT 16\AH=00h вместо значения E0h кода ASCII возвращает 00h, за исключением нажатий клавиш "num /" и "num Enter": для них вместо скэн-кода E0h он выдает коды 35h, и 1Ch соответственно.

Примечание 4: собственный код этой клавиши в буфер клавиатуры не заносится, но влияет на действия обработчика прерывания INT 09, например, изменяет формирование кодов других клавишей.

Примечание 5: контроллер клавиатуры отвечает на нажатие клавиши SysRq выдачей кодовой последовательности "E0 2A E0 37", а на отпускание – выдачей обратной кодовой последовательности "E0 B7 E0 AA". Не все версии BIOS выдают один и тот же код в ответ на клавишную комбинацию CTRL-SysRq.

Примечание 6: клавиша Pause/Break уникальна тем, что ее отпускание отдельно не фиксируется, код отпускания следует сразу после кода нажатия, образуя последовательность E1 1D 45 E1 9D C5. Прием такой последовательности обработчиком прерывания INT 09 влечет сброс (обнуление) буфера клавиатуры и вызов INT 1B.

Примечание 7: показанные коды данной клавиши соответствуют выключенному состоянию переключателя NumLock, а когда он включен, коды из третьей и четвертой колонок таблицы меняются местами.

Примечание 8: обработчик прерывания INT 16\AH=00h вообще не реагирует на нажатие этой клавиши.

Примечание 9: в некоторых моделях клавиатур имеются три клавиши управления электропитанием компьютера: POWER, SLEEP и WAKE UP. Этим клавишам соответствуют скэн-коды E0 5E, E0 5F, E0 63.

A.02-2 Раскладки клавиатуры и национальные кодовые страницы

В этом разделе представлены избранные сведения для осуществления национальной адаптации клавиатуры и шрифтов с помощью средств, имеющихся в поставке операционной системы Windows-95 фирмы Microsoft. Поставляемые средства включают файл данных Country.sys, три файла с раскладками клавиатур (Keyboard.sys, Keybrd2.sys и Keybrd3.sys), и четыре файла со шрифтами (Ega.cpi, Ega2.cpi, Ega3.cpi и Iso.cpi) для различных кодовых страниц.

Первая колонка (Abbr) приведенной ниже таблицы содержит сокращенные буквенные обозначения стран мира, а третья колонка (ID) – идентификаторы раскладки клавиатуры для тех же стран. Сведения из первой и третьей колонок нужны для составления командной строки запуска драйвера Keyb.com (5.02-04). Идентификатор раскладки клавиатуры необходимо указывать только тогда, когда в

одной стране используются несколько раскладок клавиатуры. Четвертая колонка (Keyb) таблицы показывает, какой файл с раскладками клавиатуры следует загружать: цифра 1 означает файл Keyboard.sys, цифра 2 – файл Keybrd2.sys, цифра 3 – Keybrd3.sys, а слово "Any" – любой из упомянутых трех файлов.

В пятой колонке приведенной ниже таблицы показан численный код страны, используемый при загрузке файла данных Country.sys (5.02-01) с помощью команды COUNTRY (4.05).

Последняя седьмая колонка таблицы содержит номера кодовых страниц для разных стран. Эти номера необходимы программе MODE.COM (6.18), которой предстоит выбрать нужную кодовую страницу из группы страниц, содержащихся в каждом файле *.CPI (пример показан в разделе 9.01-02). В файле Iso.cpi для всех кодовых страниц содержатся шрифты, рекомендуемые международной организацией по стандартизации (ISO). Шрифты, разработанные фирмой Microsoft, поставляются в файлах Ega*.cpi, каждый из которых содержит шрифты примерно для пяти кодовых страниц. В связи с этим в шестой колонке (Ega*) таблицы указано, какой из файлов Ega*.cpi следует загружать: цифра 1 означает файл Ega.cpi, цифра 2 – файл Ega2.cpi, цифра 3 – файл Ega3.cpi, а слово "Any" – любой из упомянутых здесь шрифтовых файлов.

Abbr	Страна	ID	Keyb	Код	Ega*	Кодовая страница
GR	Австрия		Any	043	Any	CP850
BE	Бельгия		1,2	032	Any	CP850
BG	Болгария	442	2	359	3	CP855
BR	Бразилия	274, 275	1,2	055	Any	CP850
CF	Канада	058	1,2	002	1	CP863
CZ	Чехия	243	Any	042	Any	CP852
DK	Дания		1,3	045	1	CP865
SU	Финляндия		Any	358	Any	CP850
FR	Франция	120, 189	1,3	033	Any	CP850
GR	Германия		Any	049	Any	CP850
GK	Греция	319	2	030	2	CP737, CP869
HU	Венгрия		Any	036	Any	CP852
IS	Исландия	161	2	354	2	CP861
IT	Италия	141, 142	Any	039	Any	CP850
LA	Латинская Америка		1	003	Any	CP850
NL	Нидерланды		1,3	031	Any	CP850
NO	Норвегия		1,2	047	1	CP865
PL	Польша		Any	048	Any	CP852
PO	Португалия		1	351	1	CP860

Продолжение таблицы A.02-2

RO	Румыния	333	2	040	Any	CP852
RU	Россия	441	2,3	007	3	CP866
SL	Словакия	245	Any	421	Any	CP852
SP	Испания		1,3	034	Any	CP850
SV	Швеция		Any	046	Any	CP850
SF	Швейцария		1,3	041	Any	CP850
TR	Турция	440, 179	2	090	2	CP857
UK	Великобритания	166, 168	Any	044	Any	CP850
US	США и Австралия		Any	001	1,3	CP437
YC	Югославия (кириллица)	118	2	038	3	CP855
YU	Югославия (латынь)	234	Any	038	Any	CP852

Примечание 1: из всех файлов с раскладками клавиатуры только Keyboard.sys содержит раскладки, используемые в пишущих машинках.

Примечание 2: файлы, поставляемые фирмой Microsoft, несовместимы с драйвером Keyrus.com (5.02-05): последний использует встроенные кодовые таблицы и раскладки клавиатуры.

Примечание 3: шрифты для некоторых других стран (в частности, для Китая, Израиля, Японии) поставляются фирмой Microsoft только в составе специальных национальных версий операционных систем.

A.02-3 Сведения о клавиатуре в области данных BIOS

В приведенной ниже таблице показано размещение сведений о клавиатуре в области данных BIOS, причем все смещения указаны относительно сегментного адреса 0040h, то есть от начала области данных BIOS.

Смещение	Длина	Содержание
17h	2	Флаги, возвращаемые INT 16\AH=12h в AX
19h	1	Обслуживание ввода по номеру знака ASCII
1Ah	2	Адрес следующего знака в буфере клавиатуры
1Ch	2	Адрес 1-й свободной ячейки буфера клавиатуры
1Eh	32	Кольцевой буфер клавиатуры
71h	1	Бит 7: флаг нажатия Ctrl-Break
80h	2	Адрес начала буфера клавиатуры (обычно 1Eh)
82h	2	Адрес байта за буфером клавиатуры (обычно 3Eh)
96h	1	Бит 0: последний принятый код – префикс E1h; бит 1: последний принятый код – префикс E0h; бит 2: была нажата правая клавиша Ctrl; бит 3: была нажата правая клавиша ALT; бит 4: установлена "улучшенная" клавиатура;

Продолжение таблицы А.02-3

		бит 6: принятый скэн-код – 1-й байт из двух.
97h	1	Бит 0: включен светодиод Scroll Lock; бит 1: включен светодиод Num Lock; бит 2: включен светодиод Caps Lock; бит 7: от клавиатуры принят флаг ошибки.

Примечание 1: представленное здесь размещение данных может зависеть от версии BIOS в Вашем компьютере.

А.02-4 Блок параметров национальной адаптации

Здесь представлена структура блока данных, возвращаемого обработчиком прерывания INT 21\AX=6501h (8.02-74). Блок данных такой же структуры принимает обработчик прерывания INT 21\AX=7002h, задающий параметры национальной адаптации (примечание 3 к 8.02-74).

Смещение	Длина	Содержание
00h	1	= 01h при возврате (примечание 1)
01h	2	Длина таблицы при возврате (примечание 1)
03h	2	Шестнадцатеричный код страны (А.02-2)
05h	2	Номер кодовой страницы (А.02-2)
07h	2	Формат даты: = 0000h – американский (mm dd yy) = 0001h – европейский (dd mm yy) = 0002h – японский (yy mm dd)
09h	5	Название валюты, оканчивающееся байтом 00h
0Eh	2	Знак отделения тысяч в числах
10h	2	Знак отделения целой и дробной частей числа
12h	2	Знак отделения даты
14h	2	Знак отделения времени
16h	1	Бит 0: ставить символ валюты после суммы; бит 1: пробел между суммой и символом валюты; бит 2: символ валюты на месте десятичной точки.
17h	1	Число знаков правее точки в денежных расчетах
18h	1	Бит 1: 24-часовой счет времени, иначе 12-часовой
19h	4	Адрес вызова программы перевода (примечание 2)
2Dh	2	Знак разделения списков данных

Примечание 1: этот элемент данных при вызове INT 21\AX=7002h игнорируется.

Примечание 2: здесь имеется ввиду программа перевода национальных строчных букв (имеющих номер ASCII больше 80h) в заглавные. Программу следует вызывать переходом по указанному адресу командой CALL

FAR (7.03-08) с номером ASCII переводимой буквы в регистре AL. Возврат результата предполагается также через регистр AL.

A.02-5 Национальные ограничения для имен

Указатель на эту таблицу национальных ограничений возвращает обработчик прерывания INT 21\AX=6505h (примечание 1 к 8.02-74).

Смещение	Длина	Содержание
00h	2	Длина таблицы (без данного слова)
03h	1	Наименьший номер знака для имен
04h	1	Наибольший номер знака для имен
06h	1	Первый номер знака из недопустимого интервала
07h	1	Последний номер знака недопустимого интервала
09h	1	Число N знаков, отмечающих конец имен
0Ah	N	Номера ASCII знаков, отмечающих конец имен

A.02-6 Сведения о загруженных кодовых страницах

В этой таблице показан блок данных, адрес которого возвращает драйвер DISPLAY.SYS (5.02-02) в ответ на вызов функции INT 2F\AX=AD03h (8.03-27).

Смещение	Длина	Содержание
00h	2	Число M страниц, заданных конфигурацией
04h	2	Число N страниц, загруженных по умолчанию
06h	2N	Номера страниц, загруженных по умолчанию
06h+2N	2M	Номера страниц, заданных конфигурацией, или FFFFh, если эти страницы еще не загружены

A.02-7 Описание "горячих" клавишей в спецификации AMIS.

Программы часто задействуют определенные клавиши для вызова своих функций, но обычно это делается без учета функций, которые уже возложены на те же клавиши загруженными ранее резидентными программами или драйверами. Такие действия в лучшем случае приводят к потере возможности вызова функций резидентных модулей. Реальный шанс избежать перехвата "горячих" клавишей открывает спецификация AMIS (A.07-6), согласно которой резидентные модули в ответ на обращение к ним через прерывание INT 2D с кодом операции AL=05h возвращают в регистрах DX:BX указатель на перечень установленных ими

"горячих" клавишей. Предоставляемые сведения доступны любой программе, которая "пожелает" ими воспользоваться.

Первый байт, имеющий смещение 00h относительно начала перечня "горячих" клавишей, определяет способ перехвата вызовов (примечание 1). Второй байт, имеющий смещение 01h относительно начала перечня, определяет число "горячих" клавишей, которые установлены данным резидентным модулем и в момент вызова не дезактивированы. Тот же второй байт определяет общую длину возвращаемого перечня, потому что после него, начиная со смещения 02h, располагаются дескрипторы по 6 байт на каждую клавишу. Состав данных в дескрипторах "горячих" клавишей показан в приведенной ниже таблице. Смещения в таблице указаны относительно начала каждого дескриптора.

Смещение	Длина	Содержание	Примечания
00h	1	Скэн-код "горячей" клавиши	*2
01h	2	Требуемые состояния флагов	*3
03h	2	Недопустимые состояния флагов	*4
05h	1	Дополнительные флаги клавиши	*5

Примечание 1: значения битов в байте со смещением 00h в перечне "горячих" клавишей дешифрируются следующим образом:

бит 0 – перехват до обработчика INT 09
бит 1 – перехват после обработчика INT 09
бит 2 – перехват до обработчика INT 15\AH=4Fh
бит 3 – перехват после обработчика INT 15\AH=4Fh
бит 4 – перехват вызовов INT 16\AH=00h,01h,02h
бит 5 – перехват вызовов INT 16\AH=10h,11h,12h
бит 6 – перехват вызовов INT 16\AH=20h,21h,22h
бит 7 – резервирован, должен быть сброшен в нуль

Примечание 2: если старший бит скэн-кода обнулен, то срабатывание клавиши регистрируется по ее нажатию, а если старший бит установлен, то срабатывание регистрируется по отпусканию клавиши. Когда срабатывание регистрируется только по комбинации состояний "функциональных" клавишей (Shift, Ctrl и т.п.), тогда вместо скэн-кода следует указывать 00h или 80h соответственно.

Примечание 3: используемое здесь слово флагов почти идентично слову флагов клавиатуры, возвращаемому обработчиком прерывания INT 16\AH=12h (8.01-85). Изменено лишь значение бита 7: здесь оно выражает активизацию любой (левой или правой) клавиши Shift. Установление в единицу любого бита в слове требуемого состояния флагов фиксирует необходимое условие срабатывания "горячей" клавиши.

Примечание 4: битам в слове недопустимых флагов клавиатуры приписан тот же смысл, что и в слове требуемых состояний (примечание 3), но здесь установление любого бита в единицу выражает условие предотвращения срабатывания данной "горячей" клавиши. Комбинирование требуемых и недопустимых условий существенно снижает вероятность ошибочных срабатываний.

Примечание 5: последний байт каждого дескриптора клавиши содержит биты дополнительных флагов, состояния которых означают следующее:

бит 0 – активизация до исполнения миссии модуля;
бит 1 – активизация после исполнения миссии модуля;
бит 2 – допустим перехват для мониторинга клавиши;
бит 3 – срабатывание блокируется другими клавишами;
бит 4 – роль данной клавиши переопределена;
бит 5 – активизация зависит от условий исполнения;

Биты 6 и 7 зарезервированы и должны быть сброшены в нуль.

A.02-8. Служебные коды стандарта ASCII.

Позиции 0 – 31 в стандартном американском коде для обмена информацией (ASCII) выделены служебным меткам и командам. Все кодовые страницы DOS унаследовали эти 32 служебных кода стандарта ASCII. Большинство служебных кодов в среде MS-DOS7 игнорируется, но некоторые из них тем не менее принимаются к исполнению.

Часть служебных кодов может быть введена посредством клавишных комбинаций, показанных в разделе 1.05. Любой служебный код можно ввести, если набрать десятичный номер кода (0 – 31) цифровыми клавишами в правой части клавиатуры, придерживая нажатой клавишу ALT.

На ввод исполняемого служебного кода может реагировать модуль ввода в составе драйвера консоли (устройства CON). Следующим обычно является командный интерпретатор. Функции BIOS предоставляют как возможность исполнения некоторых выводимых служебных кодов (8.01-21, 8.01-33), так и возможность избежать их исполнения (8.01-17). Модуль вывода в составе драйвера консоли по умолчанию не пытается избежать исполнения служебных кодов. На него можно повлиять посредством посылки ему строки параметров (8.02-41), но поводов для этого обычно нет, так как в среде DOS любая программа "вправе" обращаться не к драйверу консоли, а напрямую к функциям BIOS.

Иногда использование служебных кодов может быть полезно, но при этом надо знать, где и каким образом вводимый код будет интерпретирован. С этой целью ниже приведен перечень служебных кодов, действующих в среде MS-DOS7, с кратким описанием связанных с ними ассоциаций.

Код	Номер	Значение
00h	0	Маркер конца строк ASCIIZ, в том числе строк с записями имен и значений переменных окружения.
03h	3	Маркер "Конец текста", вызывает прерывание исполнения командных файлов (пример – в разделе 3.21).
07h	7	"Звуковой сигнал", при посылке его для воспроизведения на экране раздается короткий звуковой сигнал (beep).
08h	8	Смещает курсор на одно знакоместо влево. При посылке через драйвер консоли (CON) стирает последний знак.
09h	9	Код горизонтальной табуляции, при индикации на экране он автоматически заменяется на 8 пробелов.
0Ah	10	"Перевод строки" вызывает переход на следующую строку без возврата позиции курсора к началу строки.
0Ch	12	Команда принтерам "Вытолкнуть лист". BIOS и драйвер консоли на нее, как правило, не реагируют.
0Dh	13	"Возврат каретки" возвращает курсор к началу строки. Служит маркером конца строки в области DTA (8.02-16).
1Ah	26	Необязательная метка конца текстовых файлов, на этой метке может быть прекращено копирование файла (3.06).
1Bh	27	"Выйти" ("Escape") – маркер команд, адресуемых драйверу ANSI.SYS (если он установлен).

Примечание 1: вместе байты 0Dh 0Ah служат маркером конца строки во всех текстовых файлах, формируемых в среде DOS.

Примечание 2: нельзя исключать вероятность загрузки таких резидентных модулей, для которых будут являться командами не показанные здесь служебные коды стандарта ASCII, обычно игнорируемые в среде MS-DOS7.

А.03 Системные данные для доступа к дискам

А.03-1. Структура блока параметров дисковода (DPB)

MS-DOS хранит параметры логических дисков в блоках DPB (Drive Parameter Blocks), по одному такому блоку на каждый диск. С помощью INT 21\AX=7302h (8.02-79) можно скопировать любой блок DPB в заранее подготовленный буфер. Адреса блоков DPB предоставляются пользовательским программам обработчиками прерываний INT 21\AH=1Fh и INT 21\AH=32h (8.02-24), документированных в предшествующих версиях DOS. Ниже на рис. 8 показан весь процесс доступа к блоку DPB диска C:, включающий вызов INT 21\AH=32h, считывание адреса блока DPB из регистров DS:BX и вывод на экран дампа по

найденному адресу 00C9:13C0h. Там же на рис. 8 показан еще вывод на экран дампа блока DPB для следующего диска D: по адресу 00C9:13FDh, считанному начиная со смещения 19h из блока DPB предыдущего диска C:.

```
D:\MSDOSTXT>debug
-a100
195B:0100 mov AH,32
195B:0102 mov DL,03
195B:0104 int 21
195B:0106 nop
195B:0107
-g=100 106

AX=3200  BX=13C0  CX=0000  DX=0003  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=00C9  ES=195B  SS=195B  CS=195B  IP=0106   NV UP EI PL NZ NA PO NC
195B:0106 90            NOP
-d 00C9:13C0 L3D
00C9:13C0  02 02 00 02 0F 04 01 00-02 00 02 15 02 E3 F9 FA   ................
00C9:13D0  00 F5 01 5E 00 70 00 F8-00 FD 13 C9 00 00 00 FF   ...^.p..........
00C9:13E0  FF FF FF 00 1A 72 0D 06-2E 15 02 00 00 E3 F9 00   .....r..........
00C9:13F0  00 FA 00 00 00 56 2E 8B-36 00 00 00 00            .....V..6....
-d 00C9:13FD L3D
00C9:13F0                                        03 03 00                ...
00C9:1400  02 0F 04 01 00 02 00 02-F9 01 21 EB EC 00 D9 01   ..........!.....
00C9:1410  5E 00 70 00 F8 00 3A 14-C9 00 FC 0B FF FF FF FF   ^.p...:.........
00C9:1420  96 00 5E C3 1E 56 F9 01-00 00 21 EB 00 00 EC 00   ..^..V....!.....
00C9:1430  00 00 04 1A 00 B4 00 00-00 00                     ..........
-
```

Рис. 8

Упомянутые процедуры доступа к блокам DPB автоматически вызывают обращение к запрошенному диску с целью обновления данных в блоке DPB. Это замедляет исполнение программы и не всегда приемлемо по отношению к сменным дискам, которые могут отсутствовать в дисководе. Альтернатива состоит в получении адреса блока DPB из ячейки со смещением 45h в таблице CDS (A.03-3) того же диска.

Все блоки DPB имеют одинаковую структуру, показанную в приведенной ниже таблице, причем байты до 20h содержат те же данные, как и в предыдущих версиях DOS, а данные в ячейках со смещением больше 20h специфичны для расширенных блоков DPB в MS-DOS7.

Смещение	Длина	Содержание
00h	1	Номер логического диска (00h = A:, 02h = C:, . . .)
01h	1	Номер диска в списке дисков данного драйвера
02h	2	Размер сектора в байтах
04h	1	Наибольший номер сектора в кластере
05h	1	Число сдвигов для пересчета кластеров в секторы
06h	2	Число резервированных секторов в начале диска
08h	1	Число таблиц FAT
09h	2	Предельное число записей в корневом каталоге

Продолжение таблицы А.03-1

0Bh	2	Номер начального сектора записи данных
0Dh	2	Наибольший кластер (= число кластеров + 1)
0Fh	2	Число секторов в каждой таблице FAT
11h	2	Номер первого сектора каталога
13h	4	Адрес заголовка драйвера данного диска (А.05-1)
17h	1	Байт-идентификатор носителя (INT 21\AH=1Ch)
18h	1	= FFh до 1-го обращения к диску, = 00h - после.
19h	4	Адрес DPB следующего логического диска
1Dh	2	Начальный кластер для поиска свободного места
1Fh	2	Число свободных кластеров (FFFFh = неизвестно)
21h	2	Старшие 16 разрядов числа свободных секторов
23h	2	– Биты 0 – 3: номер активной таблицы FAT; – бит 7: не копировать FAT в неактивные FAT.
25h	2	Номер сектора с данными о FAT (примечание 2).
27h	2	Номер сектора с резервной копией boot-сектора
29h	4	Номер начального сектора первого кластера
2Dh	4	Номер последнего кластера данного диска
31h	4	Число секторов, занятых таблицами FAT
35h	4	Номер начального кластера корневого каталога

Примечание 1: таблица DPB создается путем преобразования данных блоков BPB (А.03-4) с помощью INT 21\AH=53h.

Примечание 2: значение FFFFh в ячейке со смещением 25h означает, что на данном диске нет отдельного сектора с информацией о FAT. Но если такой сектор имеется, то в нем начиная со смещения 00h записывается 4-байтовая сигнатура 61417272h, затем со смещения 04h – число свободных кластеров (или FFFFFFFFh, если число свободных кластеров неизвестно), и со смещения 08h записывается 4-байтовый номер последнего выделенного кластера.

А.03-2 Таблицы дисковых данных (DDT)

Таблицы DDT (Disk Data Tables) представляют собой базу данных, но не для пользовательских программ, а для драйверов, входящих в состав ядра DOS. Таблицы DDT создаются для локальных дисков, доступных посредством функций BIOS, включая те виртуальные диски, которые эмулированы системой BIOS с образа диска на загрузочных оптических дисках CD-ROM. Однако таблицы DDT не создаются для RAM-дисков, IFS-дисков и вообще всех дисков, доступ к которым обеспечен драйверами, устанавливаемыми из конфигурационных файлов DOS.

Совокупность таблиц DDT организована в виде цепи, в которой первым 4-байтовым словом в каждой таблице является указатель на следующую таблицу. В последнюю таблицу DDT вместо указателя первым словом вписывается маркер

FFFFh. Каждая таблица DDT имеет фиксированную длину 96h байт. Указатель на начало первой таблицы DDT можно получить с помощью INT 2F\AX=0803h (8.03-04). Этого достаточно, чтобы проследить цепочку таблиц DDT до конца.

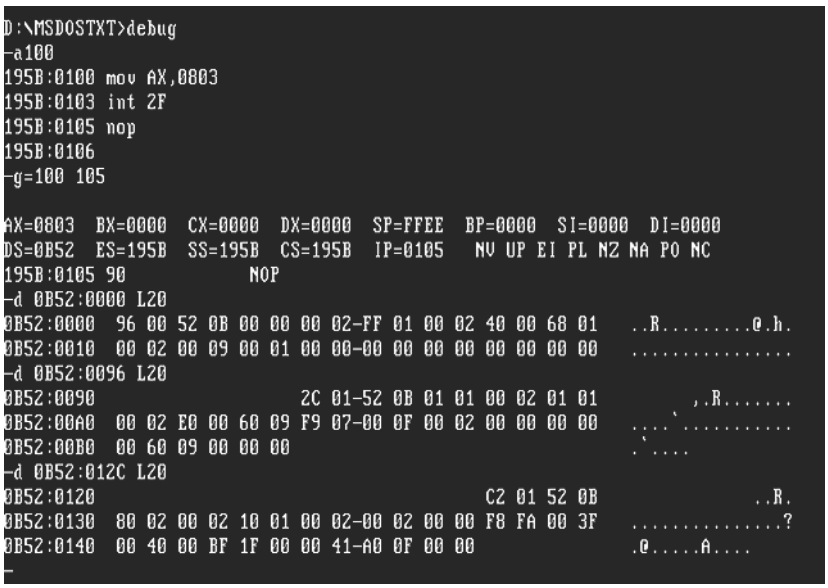


Рис. 9

На рис. 9 показан процесс доступа к таблицам DDT для дисков A:, B: и C:, включая вызов INT 2F\AX=0803h, считывание возвращаемого адреса (0B52:0000h) 1-й таблицы DDT из регистров DS:DI, выведение дампа части таблицы DDT для диска A:, считывание из первых 4 байт этого дампа адреса (0B52:0096h) таблицы DDT для следующего диска B:, выведение дампа части таблицы DDT диска B: и повторение двух последних операций по отношению к таблице DDT диска C:.

Приведенная ниже таблица A.03-2 показывает структуру данных в одной таблице DDT. Такой же формат данных используется обработчиком прерывания INT 2F\AX=0801 (8.03-02), добавляющим еще одну таблицу для нового диска к совокупности таблиц DDT.

Смещение	Длина	Содержание
00h	4	Адрес следующей DDT (или FFFFh, если ее нет)
04h	1	Номер дисковода (примечание 1 к 8.01-44)
05h	1	Номер диска (00h = A:, 02h = C:, и т.д.)
06h	25	Блок BPB используемого диска (A.03-4, до 19h)
3Bh	1	– Бит 6: файловая система FAT-16; – бит 7: на все запросы отвечать "Not Ready".
3Ch	2	Счетчик открытых файлов данного диска
3Eh	1	Тип диска (как в байте 01h таблицы A.04-3)
3Fh	2	– Бит 0: несменный жесткий магнитный диск;

Продолжение таблицы A.03-2

		– бит 1: дисковод с датчиком смены дисков; – бит 2: изменения данных BPB не допускаются; – бит 3: секторы диска – одинакового размера; – бит 4: нужно указывать LUN (примечание 1); – бит 5: на дисководе – не один логический диск; – бит 6: зарегистрирован факт смены диска; – бит 7: параметры изменены (примечание 2); – бит 8: диск переформатирован, BPB изменен; – бит 9: флаг запрета доступа (примечание 3).
43h	25	Блок данных BPB (примечание 4)
7Dh	12	11 байтов метки тома, кончающиеся байтом 00h
89h	4	Шестнадцатеричный серийный номер диска
8Dh	9	Название файловой системы, конечный байт 00h

Примечание 1: номер LUN (Logical Unit Number) служит для различения устройств, имеющих общий шинный адрес. Это необходимо, в частности, для оптических дисководов класса DVD-RAM, которые в зависимости от номера LUN могут быть представлены либо как сменный магнитный диск, либо как диск CD/DVD-ROM. Адаптеры флэш-карт представляют карты в слотах разного типа отдельными логическими дисками, также различаемыми по номеру LUN.

Примечание 2: если параметры диска изменены, то данные в таблице DDT должны быть обновлены с помощью INT 21\AX=440Dh\CX=4840h (8.02-46).

Примечание 3: флаг запрета доступа применяется только по отношению к накопителям на жестких магнитных дисках, в особенности для пресечения доступа к другим первичным разделам (помимо основного загрузочного). INT 21\AX=440Dh\CX=4867h инверсно отображает состояние флага запрета доступа, а изменить его можно с помощью INT 21\AX=440Dh\CX=4847h (8.02-46).

Примечание 4: блок данных BPB (= BIOS Parameter Block, A.03-4), начинающийся со смещения 43h, относится не к фактическому диску, а к типу дисков, принимаемому по умолчанию для данного дисковода.

A.03-3 Структура блоков данных таблицы CDS

Таблица CDS (Current Directory Structure) содержит блоки данных, в которых для каждого логического диска фиксируется путь к принимаемому по умолчанию (текущему) каталогу, а также ряд других важных характеристик. Все блоки данных CDS имеют одинаковую длину 58h байт. Адрес блока данных 1-го логического диска находится в ячейке со смещением 16h "Списка Списков" DOS (A.01-2). Там же в ячейке со смещением 21h записано общее число блоков данных в таблице CDS, заданное спецификацией команды LASTDRIVE (4.17) в файле CONFIG.SYS.

```
D:\MSDOSTXT>debug
-a 100
195B:0100 mov AH,52
195B:0102 int 21
195B:0104 nop
195B:0105
-g =100 104

AX=5200  BX=0026  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=195B  ES=00C9  SS=195B  CS=195B  IP=0104   NV UP EI PL NZ NA PO NC
195B:0104 90            NOP
-d 00C9:0026 L1A
00C9:0020                   46 13-C9 00 CC 00 C9 00 4C 00         F.......L.
00C9:0030  70 00 16 00 70 00 00 02-6D 00 C9 00 00 00 03 D2   p...p...m.......
-d D203:0000 L58
D203:0000  41 3A 5C 44 4F 53 5C 4D-53 37 00 00 00 00 00 00   A:\DOS\MS7......
D203:0010  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
D203:0020  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
D203:0030  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
D203:0040  00 00 00 00 40 46 13 C9-00 96 02 00 00 FF FF 02   ....@F..........
D203:0050  00 00 00 00 00 00 00 00                           ........
-
```

Рис. 10

Доступ к таблице CDS показан на рис. 10. Первый шаг – вызов INT 21\AH=52h, он возвращает адрес "Списка Списков" в регистрах ES:BX. Затем по полученному адресу (00C9:0026h) выводится дамп части "Списка Списков", причем длина дампа выбрана так, что последние 4 байта со смещениями 16h – 19h представляют адрес начала таблицы CDS (D203:0000h). Начиная с этого адреса выводится дамп длиной 58h, представляющий блок данных диска A:. Блоки данных всех других дисков расположены последовательно вслед за выведенным блоком с шагом 58h.

Среди блоков данных CDS имеются и такие, которые не относятся ни к одному из логических дисков. Они лишь резервируют букву для тех дисков, которые могут стать доступными потом с помощью загружаемых драйверов. Вся таблица CDS создается при интерпретации строк файла CONFIG.SYS, дополнить ее блоками данных позднее нельзя, поэтому нужное количество резервных блоков данных CDS необходимо заказывать заранее посредством команды LASTDRIVE (4.17). Структура одного блока данных CDS показана в приведенной ниже таблице.

Смещение	Длина	Содержание
00h	67	Путь к текущему каталогу (примечания 1 и 2).
43h	2	Атрибуты логического диска: бит 7: скрыть букву диска от переназначения; бит 12: диск создан программой SUBST.EXE; бит 13: диск создан программой JOIN.EXE; бит 14: диск – на физическом дисководе; бит 15: диск создан сетевым редиректором.
45h	4	Указатель на блок параметров DPB (A.03-1)
49h	2	1-ый кластер текущего каталога (примечание 5)

Продолжение таблицы A.03-3

4Fh	2	Число скрываемых знаков пути (примечание 6)
52h	4	Указатель на сетевой редиректор или IFS-драйвер, если они имеют отношение к созданию данного диска (иначе в это поле записываются нули)

Примечание 1: для локальных дисков путь к текущему каталогу включает букву диска, двоеточие, разделительную обратную косую черту, и обычную оставшуюся часть. Путь должен кончаться байтом 00h.

Примечание 2: если в слове атрибутов установлен бит 12 или бит 13, то путь, указанный в блоке данных, не является реальным путем.

Примечание 3: если в слове атрибутов биты 14 и 15 сброшены в нуль, то диск будет считаться недействительным и будет скрыт.

Примечание 4: если в слове атрибутов оба бита 14 и 15 установлены в единицу, то данный диск представляет файловую систему IFS.

Примечание 5: начальный кластер считается в пределах логического диска; для корневого каталога он имеет номер 0000h. Если доступ к диску еще не производился, то в это поле записывается значение FFFFh.

Примечание 6: DOS может сообщать только конечную часть пути, если в ячейку со смещением 4Fh записано ненулевое число знаков, отсчитываемых от начала пути, которые надлежит скрыть.

Примечание 7: в ранних версиях DOS размер блока данных CDS был 51h байт; байты 51h-57h были добавлены начиная с MS-DOS4 для обслуживания сетевых дисков и IFS-систем.

A.03-4 Структура данных блока BPB

Блок данных BPB (BIOS Parameter Block) входит в состав boot-сектора каждого логического диска. На основе данных блока BPB оперативно корректируются сведения в таблицах DPB (A.03-1) и DDT (A.03-2) при каждой смене носителя (например, дискеты) в дисководе. В разделах диска с файловой системой FAT-16 структура данных блока BPB в boot-секторе занимает 39h байт; она показана в первой колонке "F16" приведенной ниже таблицы. Те же данные в составе таблиц DDT (A.03-2) имеют несколько иное, "стандартное" расположение, показанное в третьей колонке "Std". Блоки данных стандартной структуры принимаются функцией INT 21\AX=440Dh\CX=0840h и возвращаются функцией INT 21\AX=440Dh\CX=0860h (8.02-46). Но обе эти функции обслуживают только диски с файловыми системами FAT-12 и FAT-16.

В разделах диска с файловой системой FAT-32 структура данных блока BPB в boot-секторе занимает 5Ah байт; она показана во второй колонке "F32" приведенной ниже таблицы. В составе таблиц DDT (A.03-2) применен новый, "расширенный" формат размещения тех же данных, показанный в четвертой колонке "Ext". С таким форматом данных работают новые функции MS-DOS7:

INT 21\AX=440Dh\CX=4840h для обновления данных BPB в таблицах DDT и INT 21\AX=440Dh\CX=4860h для считывания данных BPB из таблиц DDT. Эти новые функции (8.02-46) применимы к дискам с файловой системой FAT-32.

F16	F32	Std	Ext	Длина	Содержание
00h	00h			3	Команда перехода:EBh 3Ch 90h для FAT-16, EBh 5Ah 90h для FAT-32
03h	03h			8	Сигнатура создателя блока BPB
0Bh	0Bh	00h	00h	2	Размер сектора в байтах
0Dh	0Dh	02h	02h	1	Размер кластера (FFh = неизвестен)
0Eh	0Eh	03h	03h	2	Число секторов до первой FAT
10h	10h	05h	05h	1	Число таблиц FAT
11h	11h	06h	06h	2	Число записей в корневом каталоге
13h		08h	08h	2	= 0000h (примечание 1)
15h	15h	0Ah	0Ah	1	Идентификатор (примечание 2)
16h	16h	0Bh	0Bh	2	Размер FAT (примечание 3)
18h	18h	0Dh	0Dh	2	Число секторов на дорожке диска
1Ah	1Ah	0Fh	0Fh	2	Число головок
1Ch	1Ch	11h	11h	4	Номер 1-го сектора (примечание 4)
20h	20h	15h	15h	4	Число секторов (примечание 1)
	24h		19h	4	Размер FAT (примечание 3)
	28h		1Dh	2	То же, что 23h в таблице A.03-1
		1Fh		2	Число цилиндров (примечание 5)
	2Ah		1Fh	2	Версия файловой системы
		21h		1	То же, что 01h в таблице A.04-3
	2Ch		21h	4	1-ый кластер корневого каталога
		22h		2	То же, что 02h в таблице A.04-3
24h	40h			1	Номер физического дисковода
			25h	2	То же, что 25h в таблице A.03-1
26h	42h			1	Сигнатура boot-сектора (= 29h)
27h	43h		27h	4	Серийный номер диска
2Bh	47h		2Bh	11	Метка тома (или "NO NAME ")
36h	52h		36h	8	Наименование файловой системы

Примечание 1: для разделов жестких магнитных дисков размером менее 32 Мбайт четырехбайтовое число в ячейке со смещением 15h должно быть нулевым, полное число секторов для таких малых разделов должно быть указано в ячейке со смещением 08h.

Примечание 2: здесь используется идентификатор носителя, соответствующий спецификации INT 21\AH=1Ch (8.02-17). Если тип сменного диска не определен, то в поле идентификатора заносится значение 00h.

Примечание 3: в блоках BPB "расширенного" формата в ячейку со смещением 0Bh заносится значение 0000h, а число секторов в каждой таблице FAT выражается 4-байтовым значением в ячейке со смещением 19h.

Примечание 4: в блоках BPB, формируемых в разделах жестких магнитных дисков, номер начального сектора логического диска в ячейке со смещением 1Ch совпадает с номером, записанным в дескриптор соответствующего раздела (A.13-5) в ячейку со смещением 08h.

Примечание 5: слово в ячейке со смещением 1Fh и следующие байты 21h, 22h стандартного блока BPB не включены в расширенный формат BPB и не входят в блоки данных BPB в составе таблиц DDT (A.03-2).

A.04 Таблицы данных для управления вводом-выводом

A.04-1 Формат блока данных для запросов серийного номера диска

Этот формат используется при вызове прерываний INT 21\AX=6901h (8.02-77) и INT 21\AX=440Dh\CX=4846h (8.02-46) для записи серийного номера на диск, а также возвращается после вызова прерываний INT 21\AX=6900h (8.02-77) и INT 21\AX=440Dh\CX=4866h (8.02-46), считывающих серийный номер с диска.

Смещение	Длина	Содержание
00h	2	= 0000h
02h	4	Серийный номер диска в двоичной форме
06h	11	Метка тома (или "NO NAME ", если метки нет)
11h	8	При возврате: наименование файловой системы

Примечание 1: наименование "CDROM " означает файловую систему High-Sierra, а наименование "CD001 " – файловую систему ISO 9660.

A.04-2 Состав информационного слова номерной ссылки на файл

Информационное слово номерной ссылки считывается из ячейки со смещением 05h в соответствующем блоке описания таблицы SFT (A.01-4) и возвращается после запроса к обработчику прерывания INT 21\AX=4400h (8.02-40). Если запрос относится к номерной ссылке на файл, то возвращаемое слово следует интерпретировать согласно приведенной ниже таблице (данные для не-файловых ссылок показаны в таблице A.05-2). Отличительным признаком ссылки на файл является сброшенный в нуль бит 7 возвращаемого информационного слова.

Бит	Значение
15	Файл не локальный, доступен через сеть
14	Не переустанавливать дату и время при закрытии файла
11	Файл размещен на не-сменном жестком магнитном диске
7	Сброс бита 7 – признак ссылки на файл
6	Запись в файл не производилась
5 - 0	Номер диска (000000b = A:, 000001b = B:, 000010b = C:, . . .)

A.04-3 Блок данных для задания параметров ввода-вывода

Указатель на этот блок данных должен быть помещен в регистры DS:DX при вызове INT 21\AX=440Dh\CX=4840h (8.02-46) для обновления сведений в таблицах DPB (A.03-1) и DDT (A.03-2). Блок данных такой же структуры возвращает INT 21\AX=440Dh\CX=4860h (8.02-46), адрес буфера для него следует подготовить в регистрах DS:DX. Флаги в буфере в байте со смещением 00h – не возвращаемые, они выражают условия запроса и должны быть указаны там заранее.

Смещение	Длина	Содержание
00h	1	Флаги (биты 3 – 7 должны быть обнулены): бит 0: использовать данный BPB (примечание 1) бит 1: задать раскладку дорожек (примечание 2) бит 2: все секторы – одинаковые (примечание 2)
01h	1	Тип устройства: = 00h – дисковод 320 кб/360 кб = 01h – дисковод 1.2 Мб = 02h – дисковод 720 кб = 05h – жесткий магнитный диск = 06h – ленточный накопитель = 07h – другие типы (включая 1.44 Мб) = 08h – оптический дисковод = 09h – дисковод 2.88 Мб
02h	2	Атрибуты устройства: бит 0: накопитель с несменным носителем бит 1: имеется датчик смены носителя
04h	2	Число цилиндров (или число дорожек)
06h	1	Флаги носителя: = 01h – дискета 320кб/360кб = F8h – сжатый логический диск = 00h – все другие типы носителей
07h	31	Блок данных BPB (примечание 3)

Примечание 1: если бит 0 в байте флагов установлен, то обновление или копирование блока BPB (A.03-4) обязательно повлечет обращение к физическому носителю записи. Если же бит 0 в байте флагов сброшен, то обращение к носителю записи не произойдет: будет скопирована или обновлена копия блока BPB, размещенная в таблице DDT (A.03-2) начиная со смещения 43h и определяющая тип носителей, принимаемый для данного дисковода по умолчанию.

Примечание 2: биты 1 и 2 в байте флагов определяют интерпретацию необязательного субблока данных о распределении секторов на дорожке. Этот субблок размером до 256 байт может начинаться со смещения 26h для подфункции CX=0840h и со смещения 5Ch для подфункции CX=4840h. Обслуживание дисков с секторами разного размера в этой книге не рассматривается. Для подфункции CX=4860h бит 1 в байте флагов должен быть сброшен.

Примечание 3: для подфункции CX=0840h блок BPB, начинающийся со смещения 07h, должен быть стандартного формата (A.03-4), причем в нем последние 6 байтов после смещения 1Eh игнорируются, если в показанном здесь блоке данных бит 0 начального байта флагов не установлен в единицу. Для подфункции CX=4840h блок BPB, начинающийся со смещения 07h, должен соответствовать расширенной спецификации (длиной 53 байта), показанной в 4-й колонке таблицы A.03-4.

A.04-4 Формат блока данных для операций записи и считывания

Указатель на этот блок данных принимают при вызове обработчики прерываний INT 21\AX=440Dh\CX=4861h для считывания и INT 21\AX=440Dh\CX=4841h для записи (8.02-46). Обе эти операции могут быть исполнены в "окне DOS" операционной системы Windows только при условии, что конкурентный доступ к адресуемому диску заранее блокирован (8.01-58).

Смещение	Длина	Содержание
00h	1	= 00h
01h	2	Номер запрашиваемой головки
03h	2	Номер запрашиваемого цилиндра
05h	2	Номер сектора, с которого надо начать
07h	2	Сколько секторов надо считать или записать
09h	4	Адрес буфера с данными или для данных

A.04-5 Формат блока данных для операций форматирования и сверки

Указатель на этот блок данных принимают при вызове обработчики прерываний INT 21\AX=440Dh\CX=4842h для форматирования и INT 21\AX=440Dh\CX=4862h для верификации (8.02-46). Обе эти операции могут быть исполнены в "окне DOS" операционной системы Windows только при условии, что конкурентный доступ к адресуемому диску блокирован (8.01-58).

Смещение	Длина	Содержание
00h	1	Бит 0: не форматировать, но выдать код статуса бит 1: многодорожечное форматирование (только для жестких магнитных дисков)
01h	2	Номер запрашиваемой головки
03h	2	Номер запрашиваемого цилиндра
05h	2	Сколько дорожек надо форматировать или сверить

Примечание 1: если сброшен бит 1 в байте со смещением 00h, то форматируется только одна дорожка; слово со смещением 05h игнорируется.

Примечание 2: для операции верификации заявленное число дорожек в слове со смещением 05h не должно содержать более 255 секторов, бит 0 в байте со смещением 00h должен быть установлен в единицу для верификации нескольких дорожек сразу, а бит 1 в том же байте должен быть сброшен в нуль.

Примечание 3: при возврате в байт со смещением 00h записывается код статуса:

00h – данная функция поддерживается системой BIOS
01h – данная функция не поддерживается системой BIOS
02h – параметры для данного логического диска не подходят
03h – сменный диск в дисководе отсутствует,

причем возврат значения 00h в ответ на запрос кода статуса не гарантирует успешного проведения форматирования или сверки.

A.05 Структуры данных для драйверов

A.05-1 Структура заголовка драйвера

В приведенной ниже таблице показаны структуры данных для трех типов драйверных заголовков:

в колонке "B" – для ленточных и дисковых накопителей ("block" devices);
в колонке "C" – для каналов обмена данными ("character" devices);
в колонке "D" – для драйверов CD/DVD-ROM.

Адрес начала драйверного заголовка для ленточных и дисковых накопителей содержится в ячейке со смещением 13h в блоке DPB (A.03-1) соответствующего накопителя. Заголовки драйверов, относящихся к каналам обмена, могут быть идентифицированы по сигнатуре, которая начинается со смещения 0Ah, при прослеживании последовательности драйверных заголовков с адреса в ячейке 22h "Списка Списков" (A.01-2), или с адреса, возвращаемого обработчиком прерывания INT 2F\AX=122Ch (8.03-12).

B	C	D	Длина	Содержание
00h	00h	00h	4	Адрес следующего драйвера (примечание 1)
04h	04h	04h	2	Атрибуты драйвера (A.05-2)
06h	06h	06h	2	Смещение начала программы стратегии
08h	08h	08h	2	Смещение начала программы прерывания
	0Ah	0Ah	8	Поле для размещения сигнатуры драйвера
		14h	1	1-ый обслуживаемый диск (примечание 2)
0Ah		15h	1	Число дисков, обслуживаемых драйвером

Примечание 1: до загрузки данного драйвера в поле адреса следующего драйвера должно быть записано значение FFFF:FFFFh; конкретный адрес в это поле впишет DOS, когда будет загружать следующий драйвер. Если же данный драйвер окажется последним, то значение FFFFh останется и будет означать окончание цепи ссылок.

Примечание 2: до загрузки данного драйвера в поле буквы диска должно быть вписано значение 00h; позже, при инициализации программы MSCDEX.EXE (5.08-03) или программы SHSUCDX.COM (5.08-04) начальное нулевое значение будет заменено номером (примечание 1 к 8.02-17) первого из дисков, обслуживаемых данным драйвером.

A.05-2 Атрибуты драйверов

Слово атрибутов помещено в ячейку со смещением 04h в заголовке драйвера (A.05-1). Но значение большинства битов в слове атрибутов различно для драйверов каналов обмена (во второй колонке приведенной ниже таблицы) и для драйверов дисковых и ленточных накопителей (в третьей колонке). Драйверы оптических дисководов, взаимодействующие с программой MSCDEX.EXE (5.08-03) или с программой SHSUCDX.COM (5.08-04), а также драйверы фиктивных дисков, сформированных программой SUBST.EXE (6.23), формально относятся к драйверам каналов обмена: у них бит 15 установлен в единицу. Запросить слово атрибутов по известному номеру логического диска можно посредством INT 21\AX=4409h (8.02-44). Биты атрибутов, не упомянутые в приведенной ниже таблице, нормально должны быть сброшены в нуль.

Слово атрибутов драйверов каналов служит основой для формирования информационного слова номерной ссылки на не-файловые объекты, которое записано в ячейку со смещением 05h в соответствующем блоке данных таблицы SFT (A.01-4). Обработчик прерывания INT 21\AX=4400h (8.02-40) возвращает это слово в регистре DX после запроса данных о не-файловом объекте (об информационном слове номерной ссылки на файл написано в A.04-2). Отличия проявляются в битах 4 – 7 второй колонки таблицы: у слова атрибутов драйвера эти биты, как правило, сброшены в нуль. А у информационного слова не-файловой номерной ссылки бит 7 установлен в единицу и является признаком отличия от информационных слов, относящихся к номерным ссылкам на файлы.

Бит	Драйверы каналов обмена	Драйверы накопителей
0	Канал STDIN (примечание 1)	= 0 (резервировано)
1	Канал STDOUT (примечание 1)	32-битовая адресация
2	Канал NUL (примечание 1)	= 0 (резервировано)
3	Канал CLOCK (примечание 1)	= 0 (резервировано)
4	Поддержка INT 29	= 0 (резервировано)
5	Бинарный вывод (примечание 2)	= 0 (резервировано)
6	Автоматически добавлять EOF	IOCTL (примечание 3)
7	= 1 (= не-файловый объект)	IOCTL (примечание 4)
9	= 0 (резервировано)	Доступа нет (примечание 5)
11	Блокировка (примечание 6)	Блокировка (примечание 6)
12	= 0 (резервировано)	Сетевой или оптический диск
13	Поддержка вывода до занятости	IBM-несовместимый формат
14	IOCTL (примечание 7)	IOCTL (примечание 7)
15	= 1 (драйвер канала)	= 0 (драйвер накопителя)

Примечание 1: в атрибутах драйверов каналов обмена из битов 0 – 3 только один бит может быть установлен в единицу (или вообще ни один).

Примечание 2: при бинарном выводе ни один из выводимых знаков не исполняется драйвером как команда (так, как показано в разделе A.02-08).

Примечание 3: установление в единицу бита 6 означает, что драйвер обслуживает вызовы прерываний INT 21\AX=440Ch, 440Dh, 440Eh, 440Fh.

Примечание 4: установление в единицу бита 7 означает, что драйвер обслуживает вызовы прерываний INT 21\AX=4410h, 4411h.

Примечание 5: установление в единицу бита 9 означает, что доступ к дискам, обслуживаемым данным драйвером, не обеспечивается функциями BIOS. Установленное состояние бита 9 характерно для драйверов, обслуживающих IFS-диски, сетевые диски, а также диски, параметры которых подменены командой DRIVPARM (4.09).

Примечание 6: установление в единицу бита 11 означает, что драйвер способен обслуживать накопители на сменных носителях, в частности, передавая сигналы блокировки и разблокировки сменных дисков.

Примечание 7: установление в единицу бита 14 означает, что драйвер способен принимать управляющие параметры, посланные через прерывания INT 21\AX=4403h или INT 21\AX=4405h (8.02-41).

A.05-3 Избранные запросы к драйверам устройств

Взаимодействие между DOS и драйвером любого устройства осуществляется посылкой в регистрах ES:BX адреса блока данных запроса при вызове программы стратегии драйвера командой CALL FAR. Драйвер принимает код операции в блоке данных запроса и инициирует ее исполнение. Потом DOS вызывает программу прерывания драйвера, которая заполняет тот же блок полученными данными. DOS принимает результат, если байт статуса в возвращенном блоке данных подтверждает успешное завершение запрошенной операции (A.05-4).

Такие же формы блоков данных запроса принимает обработчик прерывания INT 2F\AX=0802h (8.03-03), который неявно обращается к тем драйверам дисководов, которые входят в состав ядра DOS. Этими драйверами обслуживаются дисководы, для которых имеются соответствующие таблицы DDT (A.03-2). Только к этим логическим дискам можно обращаться посредством INT 2F\AX=0802h.

1-я колонка приведенной ниже таблицы показывает размер блока данных запроса в байтах, 2-я колонка – код запрашиваемой операции, 4-я колонка – тип драйверов, к которым данная операция может быть адресована. В 5-й колонке показано, может ли данная операция быть запрошена через прерывание INT 2F\AX=0802h. Знаками "звездочка" отмечены ссылки на примечания.

Длина	Код	Операция	Тип драйверов	802	Формат данных
19h	00h	Инициализация	все	Нет	A.05-5
0Fh	01h	Проверка носителя	дисковые	Да	*2
14h	03h	Посылка строки IOCTL	*1	Нет	A.05-7
1Eh	04h	Считывание данных	все	Да	A.05-6
0Eh	05h	Копирование байта	канальные	Нет	*3
0Dh	06h	Запрос статуса ввода	канальные	Нет	A.05-4
0Dh	07h	Сброс входного буфера	канальные	Нет	A.05-4
1Eh	08h	Запись (посылка) данных	все	Да	A.05-6
1Eh	09h	Запись с верификацией	дисковые	Да	A.05-6
0Dh	0Ah	Запрос статуса вывода	канальные	Нет	A.05-4
0Dh	0Bh	Сброс выходного буфера	канальные	Нет	A.05-4
14h	0Ch	Прием строки IOCTL	*1	Нет	A.05-7

Продолжение таблицы А.05-3

0Dh	0Dh	Открывание устройства	все	Нет	А.05-4
0Dh	0Eh	Закрывание устройства	все	Нет	А.05-4
0Dh	0Fh	Сменный ли диск?	дисковые	Да	А.05-4
14h	10h	Посылка до занятости	канальные	Нет	А.05-7
0Dh	17h	Номер логического диска	дисковые	Да	А.05-4

Примечание 1: с запросами на посылку и возврат строк IOCTL можно обращаться только к тем драйверам (как дисковым, так и канальным), у которых в слове атрибутов (А.05-2) установлен в единицу бит 14.

Примечание 2: в блоке данных запроса на проверку носителя драйвер принимает идентификатор носителя в байте со смещением 0Dh и возвращает байт статуса в ячейке со смещением 0Eh. Байт статуса значит:

FFh – носитель не был сменен
01h – носитель был сменен
00h – состояние носителя не определено.

Примечание 3: в блоке данных запроса на копирование байта (неразрушающее считывание) возвращается один байт данных в ячейке со смещением 0Dh, при условии, что в возвращаемом байте статуса со смещением 04h (А.05-4) бит BUSY ("занято") не установлен.

А.05-4 Формат заголовка блока данных запроса

Описываемый формат заголовка используется при вызовах драйверов командой CALL FAR (А.05-3), а также при обращениях к ним через прерывание INT 2F\AX=0802h (8.03-03). В обоих случаях регистры ES:BX должны содержать указатель на блок данных запроса, в котором заголовок представлен байтами 00h – 0Ch. Для многих операций (06h, 07h, 0Ah, 0Bh, 0Dh, 0Eh, 0Fh, 17h) блок данных запроса не содержит ничего кроме заголовка. Структура заголовка, одинаковая для всех операций, показана в приведенной ниже таблице.

Смещение	Длина	Содержание
00h	1	Длина блока данных (колонка 1 таблицы А.05-3)
01h	1	Адресуемый логический диск (примечание 1)
02h	1	Код операции (колонка 2 в таблице А.05-3)
03h	1	Код ошибки (примечания 3 и 4)
04h	1	Статус: 01h – операция выполнена 02h – устройство занято 80h – произошла ошибка.

Примечание 1: здесь логический диск определяется своим номером в списке дисков, обслуживаемых тем драйвером, к которому происходит обращение. При обращениях через прерывание INT 2F\AX=0802h

(8.03-03) номера совпадают с номерами логических дисков: 00h = A:, 02h = C: и т.д., но только в пределах начальной группы дисков, для которых имеются соответствующие таблицы DDT (A.03-2).

Примечание 2: если операция 0Fh (Сменный ли диск?) возвращает статус 02h ("занято"), то адресуемый диск – несменный жесткий диск.

Примечание 3: код ошибки выдается только если статус имеет значение 80h, то есть подтверждает факт ошибки. Тогда расшифровывать код ошибки следует по записям для прерывания INT 2F в таблице A.06-1.

Примечание 4: операция 17h ("определить логический диск") возвращает абсолютный номер логического диска в ячейке со смещением 03h. Тип диска и его наличие в дисководе не проверяются. Если запрашиваемый номер диска выходит за рамки списка дисков, обслуживаемых данным драйвером, то возвращается значение 00h.

A.05-5 Блок данных для запроса операции инициализации

Только однажды, сразу после загрузки драйвера, DOS вызывает его командой CALL FAR (7.03-08) с запросом на инициализацию. При вызове в регистрах ES:BX указывается адрес блока данных, а его заголовке (A.05-4) – код 00h операции инициализации. Получив такой запрос, драйвер проверяет наличие и фактическое состояние всех тех аппаратных средств, которые он должен обслуживать. При этом и исходные данные, и возвращаемые результаты выходят за рамки заголовка блока данных (A.05-4). Формат выходящей за пределы заголовка части блока данных (со смещениями 0Dh – 18h) показан в приведенной ниже таблице.

Смещение	Длина	Содержание
0Dh	1	При возврате: число логических дисков, обслуживаемых данным драйвером.
0Eh	4	При вызове: адрес 1-го байта за границей участка, который может быть выделен драйверу. При возврате: адрес 1-го свободного байта после участка, занятого резидентным модулем.
12h	4	При вызове: указатель на командную строку с параметрами вызова драйвера. При возврате только для дисковых драйверов: указатель на блок данных BPB (A.03-4).
16h	1	При вызове: первый свободный номер диска, предоставляемый драйверу (A: = 00h).
17h	2	При возврате: флаг сообщения (примечание 2)

Примечание 1: канальные драйверы перед возвратом данных должны обнулять четырехбайтовое слово, начинающееся со смещения 12h.

Примечание 2: когда возвращаемый драйвером флаг сообщения об ошибке имеет значение 0000h, то никакое сообщение не выдается, а когда этот флаг имеет значение 0001h, то DOS выводит на экран сообщение: "There is an error in your CONFIG.SYS file in line..." (= В Вашем файле CONFIG.SYS имеется ошибка в строке...).

А.05-6 Блок запроса на пересылку данных

Описываемая форма блока данных запроса используется при прямом вызове драйверов (А.05-3) командой CALL FAR (7.03-08), а также при вызове посредством прерывания INT 2F\AX=0802h (8.03-03) с целью пересылки данных. При вызове в регистрах ES:BX указывается адрес блока данных запроса, а его заголовке (А.05-4) – код запрашиваемой операции: 04h, 08h или 09h. Когда запрошена операция считывания (код 04h), принимаемые данные записываются в предварительно выделенную буферную область в памяти компьютера. Операции записи (коды 08h и 09h) напротив, посылают данные из буферной области на диск или в канал вывода. Для всех упомянутых операций блоки данных запроса включают заголовок (А.05-4) и остальную часть с параметрами доступа к данным. Размещение параметров за пределами заголовка показано в приведенной ниже таблице.

Смещение	Длина	Содержание
0Dh	1	Идентификатор носителя (только для дисков)
0Eh	4	Адрес буферной области для данных
12h	2	Длина пакета данных (примечание 1)
14h	2	Номер начального сектора (примечание 2)
16h	4	Указатель на идентификатор тома (примечание 3)
1Ah	4	Номер начального сектора (примечание 2)

Примечание 1: при обращении к драйверам каналов длина считываемого или записываемого пакета данных выражается в байтах, а при обращении к дисковым драйверам – в количестве секторов.

Примечание 2: в других DOS может использоваться иной формат с четырехбайтовым номером начального сектора в ячейке 14h; отличительным признаком такого формата является длина блока данных 18h, указываемая в первом байте заголовка (А.05-4). MS-DOS7 использует 4-байтовый номер начального сектора только если драйвер поддерживает 32-битную адресацию, о чем свидетельствует установленный в единицу бит 1 в слове атрибутов драйвера (А.05-2). В таких случаях начальный номер сектора указывается в ячейке со смещением 1Ah, а в ячейку со смещением 14h записывается слово FFFFh.

Примечание 3: адрес идентификатора тома возвращается в случае возникновения ошибки 0Fh (неправомерная смена носителя записи).

А.05-7 Блок данных запроса для строковых операций

Описываемая форма блока данных запроса используется при вызове драйверов (А.05-3) командой CALL FAR (7.03-08) с целью посылки или приема строки байтов. При вызове в регистрах ES:BX указывается адрес блока данных запроса, а его заголовке (А.05-4) – код запрашиваемой операции: 03h, 0Ch или 10h. Операция 10h посылает данные в канал. Операции посылки и приема строк управляющих параметров (03h, 0Ch) могут быть адресованы только тем драйверам, которые способны пользоваться механизмом программного управления (IOCTL), о чем свидетельствует установление в единицу бита 14 в слове атрибутов (А.05-2). Запрос с кодом операции 03h "предлагает" драйверу "принять к сведению" новые значения управляющих параметров, посланные посредством функций INT 21\AX=4403h и INT 21\AX=4405h (8.02-41). Запрос с кодом операции 0Ch выдается операционной системой с тем, чтобы драйвер выдал действующие значения параметров, запрашиваемые функциями INT 21\AX=4402h и INT 21\AX=4404h (8.02-41).

Блоки данных запроса упомянутых операций имеют одинаковую структуру, включающую заголовок (А.05-4) и остальную часть. Размещение данных за пределами заголовка показано в приведенной ниже таблице.

Смещение	Длина	Содержание
0Dh	1	Идентификатор носителя (только для дисков)
0Eh	4	Адрес буферной области для данных
12h	2	При вызове: число байтов в строке При возврате: число пересланных байтов

А.06 Коды ошибок

А.06-1 Сводная таблица кодов завершения

Почти все обработчики прерываний возвращают код завершения (код ошибки). Прерывания MS-DOS обычно возвращают его в регистре AL. Операции системы BIOS могут возвращать его в регистре AH. Интерпретации кодов завершения от разных обработчиков нередко не совпадают. В приведенной ниже таблице сведены вместе почти все значения кодов завершения, которые могут встретиться при работе в MS-DOS7. Правильную интерпретацию несложно выбрать по номеру того прерывания, которое возвратило данный код завершения, или по названию резидентного модуля драйвера.

Код	Источник	Расшифровка
00h	INT 24-2F	Попытка записи на защищенный от записи диск
	Прочие	Нет ошибки, успешное завершение операции
01h	INT 13	Неверный параметр, или несуществующий диск
	INT 15	Ошибка контроля четности
	INT 16	Буфер клавиатуры уже полон
	INT 24-2F	Неизвестный драйверу номер диска
	Прочие	Неверно указаны функция или код операции
02h	INT 13	Адресная метка не найдена
	INT 15	Ошибка прерывания
	INT 24-2F	Дисковод не готов к исполнению команды
	Прочие	Файл не найден
03h	INT 13	Диск защищен от записи
	INT 15	Ошибка в управлении линией 20-го разряда адреса
	INT 24-2F	Неизвестная драйверу команда
	Прочие	Ошибка в спецификации пути
04h	INT 13	Сектор не найден или ошибка считывания
	INT 24-2F	Ошибка в данных (неверный код CRC)
	Прочие	Слишком много файлов (нет места в JFT или в SFT)
05h	INT 13	Неудача попытки приведения в начальное состояние
	INT 24-2F	Неверная длина блока данных запроса
	Прочие	Нет прав доступа
06h	INT 13	Диск сменен или не вставлен в дисковод
	INT 24-2F	Ошибка поиска
	Прочие	Неверно указана номерная ссылка
07h	INT 13	Ошибка в таблице параметров жесткого диска
	INT 24-2F	Неизвестный тип носителя записи
	Прочие	Ошибка в блоке данных управления памятью
08h	INT 13	Выход за границу области DMA
	INT 24-2F	Сектор не найден
	Прочие	Имеющейся памяти недостаточно
09h	INT 13	Выход за 64 кбайт памяти или 80h секторов диска
	INT 15	Неверный APM-идентификатор устройства
	INT 24, 2F	В принтере нет бумаги
	Прочие	Неверный адрес блока памяти
0Ah	INT 13	Обнаружен дефектный сектор
	INT 24-2F	Попытка записи кончилась неудачно
	Прочие	Неверная спецификация окружения
0Bh	INT 13	Заданная дорожка не читается или не существует
	INT 15	Прибор не поставлен под управление APM
	INT 24-2F	Попытка считывания не удалась

Продолжение таблицы A.06-1

	Прочие	Неверный формат
0Ch	INT 13	Неизвестный формат дорожки или носителя
	INT 24-2F	Отказ общего характера
	Прочие	Неверный код доступа
0Dh	INT 13	Неверное число секторов при форматировании
	INT 24-26	Нарушение условий совместного доступа
	Прочие	Неверные данные
0Eh	INT 13	Обнаружена адресная метка управляющих данных
	INT 24-2F	Попытка доступа к закрытому диску
0Fh	INT 13	Уровень арбитража DMA за пределами допуска
	INT 24-2F	Ошибочная смена диска
	Прочие	Неверный номер дисковода
10h	INT 13	Неисправимые ошибки кодов CRC или ECC
	INT 24	Блок данных FCB недоступен
	Прочие	Попытка удалить текущий каталог
11h	INT 13	Ошибки в данных скорректированы кодом ECC
	INT 24-26	Переполнение буфера регистрации доступа
	Прочие	Не то же самое устройство
12h	INT 24	Несовпадение кодовых страниц
	Прочие	Больше файлов нет, или индекс вышел за допуск
13h	INT 24-26	Нет ввода данных
	Прочие	Диск защищен от записи
14h	INT 24, 26	Имеющегося дискового пространства недостаточно
	Прочие	Обращение к неизвестному устройству
15h		Дисковод не готов к исполнению операции
16h		Неизвестная команда
17h		Ошибка, выявленная контрольным кодом CRC
18h		Неверная длина блока данных запроса
19h		Ошибка при поиске
1Ah		Неизвестный тип носителя, неизвестный формат
1Bh		Сектор не найден
1Ch		В принтере нет бумаги
1Dh		Неудача при попытке записи
1Eh		Неудача при попытке считывания
1Fh		Отказ общего характера
20h	INT 13	Отказ контроллера
	Прочие	Нарушение условий совместного доступа
21h		Попытка доступа к закрытому для доступа диску
22h		Ошибочная смена диска (примечание 2 к A.06-1)
23h		Блок данных FCB (File Control Block) недоступен
24h		Переполнение буфера совместного доступа
25h		Несоответствие кодовых страниц

Продолжение таблицы A.06-1

26h		Нет ввода для завершения файловой операции
27h		Имеющегося дискового пространства недостаточно
30h	INT 13	Датчика наличия носителя в дисководе нет
31h	INT 13	В дисководе отсутствует сменный диск
32h	INT 13	Дисковод не поддерживает носители такого формата
	Прочие	Сетевой запрос не поддерживается
33h		Запрашиваемый компьютер не отвечает
34h		Указанное сетевое имя уже используется
35h		Указанное имя сетевого устройства не найдено
36h		Сеть занята
37h		Запрошенное сетевое устройство не существует
38h		Превышен предел сетевой команды BIOS
39h		Аппаратная ошибка сетевого адаптера
3Ah		Неправильный отклик со стороны сети
3Bh		Неожиданная сетевая ошибка
3Ch		Несовместимый сетевой адаптер
3Dh		Достигнут предел длины очереди запросов на печать
3Eh		Длина очереди запросов не исчерпана
3Fh		Недостаточно места для распечатки файла
40h	INT 13	Ошибка при выполнении поиска
	Прочие	Указанное сетевое имя удалено
41h		В доступе к сети отказано
42h		Тип сетевого устройства указан неверно
43h		Запрошенное сетевое имя не найдено
44h		Достигнут предел количества сетевых имен
45h		Достигнут предел сетевой сессии BIOS
46h		Временная пауза
47h		Посланный запрос к сети не принят
48h		Сетевое перенаправление операций отложено
50h		Такой файл существует
52h		Нет возможности создать каталог
53h		Неудачный исход (fail) при вызове INT 24h
54h		Слишком много перенаправлений
55h		Дублирующие перенаправления
56h		Неверный пароль
57h		Неверный параметр
58h		Неудачное завершение передачи данных в сеть
59h		Запрошенная функция не поддерживается сетью
5Ah		Требуемый компонент системы не установлен
60h	INT 15	Запрошенный режим APM недоступен
64h	Mscdex.exe	Неизвестная ошибка
65h	Mscdex.exe	Не готов к исполнению операции

Продолжение таблицы A.06-1

66h	Mscdex.exe	EMS-память недоступна
67h	Mscdex.exe	Формат диска отличен от High Sierra и ISO-9660
68h	Mscdex.exe	Заслонка оптического дисковода открыта
80h	INT 13	Дисковод не отвечает, время ожидания истекло
	INT 67	Внутренняя ошибка
	Прочие	Неверная команда, данная функция не реализована
81h	INT 67	Неправильное функционирование оборудования
	Himem.sys	Установлено, что загружен драйвер Vdisk
82h	Himem.sys	Сбой управления линией A20 адресной шины
83h	INT 67	Неверная номерная ссылка
84h	INT 67	В запросе неверно указан код функции
85h	INT 67	Лимит номерных ссылок исчерпан
86h	INT 67	Ошибка при сохранении распределения страниц
	Прочие	Запрошенная функция не поддерживается
87h	INT 67	Количество страниц памяти недостаточно
88h	INT 67	Свободных страниц памяти слишком мало
89h	INT 67	Запрошено нулевое количество страниц памяти
8Ah	INT 67	Запрошен неверный номер логической страницы
8Bh	INT 67	Запрошен неверный номер физической страницы
8Ch	INT 67	Нет места для записи распределения страниц
8Dh	INT 67	Ошибка при сохранении распределения страниц
8Eh	INT 67	Ошибка при восстановлении распределения страниц
	Himem.sys	Общий отказ XMS-драйвера
8Fh	INT 67	Неизвестный номер подфункции
	Himem.sys	Неисправимая ошибка XMS-драйвера
90h	INT 67	Неизвестный тип атрибута
	Himem.sys	Область HMA недоступна или не существует
91h	INT 67	Запрошенное свойство не поддерживается
	Himem.sys	Область HMA уже используется
92h	INT 67	Успешное завершение с перезаписью части данных
	Himem.sys	Значение DX меньше предела /HMAMIN (5.04-01)
93h	INT 67	Длина пакета больше, чем выделенное место
	Himem.sys	Область HMA не выделена
94h	INT 67	Обычная и расширенная памяти перекрываются
	Himem.sys	Линия A20 шины адреса все еще активна
95h	INT 67	Смещения превышает размер логической страницы
96h	INT 67	Размер области выходит за пределы 1 Мегабайта
97h	INT 67	Области источника и назначения перекрываются
98h	INT 67	Неизвестный тип области источника или назначения
9Ah	INT 67	Указанная группа регистров не поддерживается
9Bh	INT 67	Все группы регистров уже распределены
9Ch	INT 67	Группа регистров DMA не поддерживается

Продолжение таблицы А.06-1

9Dh	INT 67	Память под группы регистров не выделена
9Eh	INT 67	Предназначенные каналы DMA не поддерживаются
9Fh	INT 67	Запрошенный канал DMA не поддерживается
A0h	INT 67	Ссылка с запрошенным именем не существует
	Himem.sys	Вся расширенная память уже распределена
A1h	INT 67	Номерная ссылка с таким именем уже существует
	Himem.sys	Свободных ссылок больше нет
A2h	INT 67	"Заворот" адресов вокруг границы 1 Мбайт
	Himem.sys	Неверная номерная ссылка
A3h	INT 67	Массив данных источника поврежден
	Himem.sys	Неверная номерная ссылка на источник
A4h	INT 67	Операционная система отказала в доступе
	Himem.sys	Значение смещения для источника указано неверно
A5h	Himem.sys	Неверная номерная ссылка на область назначения
A6h	Himem.sys	Смещение для области назначения указано неверно
A7h	Himem.sys	Недопустимое значение длины пакета данных
A8h	Himem.sys	Недопустимое перекрытие копируемых данных
A9h	Himem.sys	Выявлена ошибка при контроле четности
AAh	INT 13	Дисковод не готов к исполнению операции
	Himem.sys	Блок памяти не закрыт
ABh	Himem.sys	Блок памяти уже закрыт
ACh	Himem.sys	Переполнение счетчика актов закрывания
ADh	Himem.sys	Неудачное завершение операции закрывания
B0h	Himem.sys	Доступны только блоки UMB меньшего размера
	INT 13	Носитель записи в накопителе не заблокирован
B1h	Himem.sys	Свободных блоков UMB больше нет
	INT 13	Носитель записи в накопителе заблокирован
B2h	Himem.sys	Указан неверный сегмент блока UMB
	INT 13	Данный носитель записи – несменяемый
B3h	INT 13	Диск задействован, кэш-буфер записи не пуст
B4h	INT 13	Переполнение счетчика актов закрывания
B5h	INT 13	Команда на выдвижение лотка не исполнена
B6h		Носитель защищен от записи
BBh	INT 13	Неизвестная ошибка жесткого магнитного диска
CCh	INT 13	Ошибка при записи на жесткий магнитный диск
E0h	INT 13	Ошибка в регистре состояния дисковода
FFh	INT 13	Ошибка опознавания на жестком магнитном диске
	INT 15	Ошибка активизации линии A20 шины адреса
	Прочие	Подходящий файл не найден, или их больше нет, или ошибка в блоке FCB (А.09-5).

Примечание 1: если в спецификации указано, что код ошибки возвращается в регистре AX, то его старший байт – нулевой (AH = 00h).
Примечание 2: при коде ошибки 22h в регистрах ES:DI возвращается указатель на идентификатор носителя, в котором:
со смещения 00h – 12 байтов: метка тома диска с 00h в конце,
со смещения 0Ch – 4 байта: серийный номер диска (двоичный).
Примечание 3: драйвер Himem.sys возвращает код ошибки в регистре BL.
Примечание 4: коды завершения функций DOS хранятся в области SDA (A.01-3, смещение 04h). Коды завершения функций BIOS записываются в области данных BIOS (A.01-1, обычно смещение 74h).

A.06-2 Значения класса ошибки, возвращаемые MS-DOS

Приведенная ниже таблица интерпретирует значения класса ошибки, возвращаемые обработчиком прерывания INT 21\AH=59h (8.02-65) в регистре BH, а также записываемые в области текущих данных (SDA, A.01-3) в ячейку со смещением 07h.

Код	Значение
01h	Не хватает ресурсов компьютера
02h	Временная ситуация
03h	Проблемы прав доступа
04h	Внутренние проблемы программного обеспечения
05h	Сбои и отказы оборудования
06h	Системные ошибки конфигурирования
07h	Ошибки прикладных программ
08h	Неудачи поиска объектов
09h	Несовместимость форматов
0Ah	Отказы, вызванные закрытием объектов
0Bh	Ошибки, вызванные носителями записи
0Ch	Дублирование существующих имен
0Dh	Ошибки из-за невыясненных причин.

A.06-3 Коды предлагаемых действий

Приведенная ниже таблица интерпретирует коды предлагаемых действий, возвращаемые обработчиком прерывания INT 21\AH=59h (8.02-65) в регистре BL, а также записываемые в области текущих данных (SDA, A.01-3) в ячейку со смещением 06h.

Код	Рекомендация
01h	Повторить попытку
02h	Повторить попытку через некоторое время
03h	Предложить пользователю ввести данные повторно
04h	Закрыть файлы, удалить временные файлы, выйти из программы
05h	Выйти из программы немедленно
06h	Игнорировать ошибку
07h	Повторить попытку после вмешательства пользователя

A.06-4 Коды вероятного места возникновения ошибки

Приведенная ниже таблица интерпретирует коды вероятного места возникновения ошибки, возвращаемые обработчиком прерывания INT 21\AH=59h (8.02-65) в регистре CH, а также записываемые в области текущих данных (SDA, A.01-3) в ячейку со смещением 03h.

Код	Вероятная локализация
01h	Источник ошибки не определен
02h	Дисковые или ленточные накопители
03h	Локальные сети
04h	Устройства, подключенные к последовательному порту
05h	Память компьютера

A.06-5 Коды статуса ошибок ввода-вывода

Данная таблица интерпретирует коды статуса ошибок, возвращаемые в регистре AH обработчиками прерываний INT 25 и INT 26 (8.02-85).

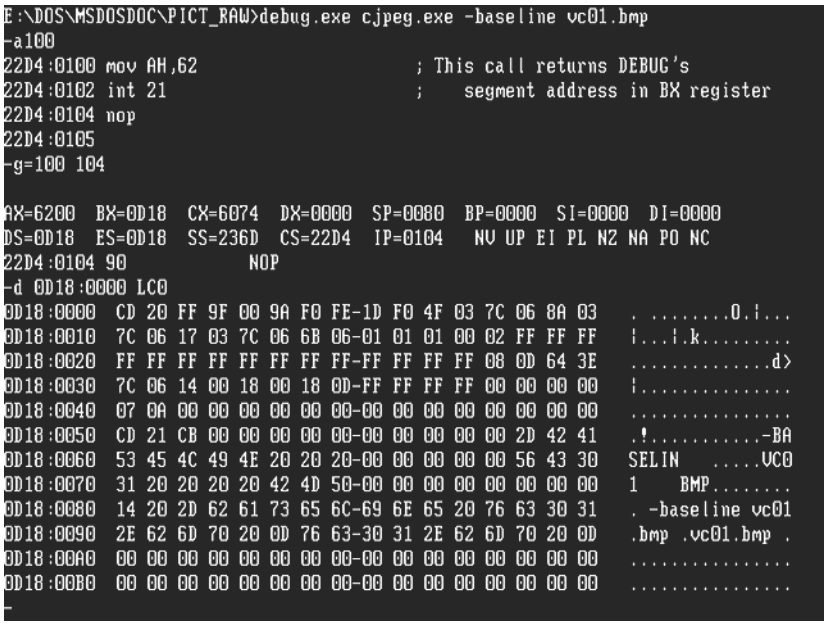
Код	Значение
01h	Неверная команда
02h	Ошибочная адресная метка
03h	Диск защищен от записи (только после INT 26)
04h	Запрошенный сектор не найден
08h	Ошибка DMA (прямого доступа к памяти)
10h	Ошибка в данных, выявляемая с помощью кода CRC
20h	Отказ контроллера
40h	Ошибка при исполнении поиска
80h	Устройство не отвечает, время ожидания истекло

А.07 Структуры данных для исполнения программ

А.07-1 Префикс сегмента программы

Когда программа загружается для исполнения в выделенный сегмент памяти, ее исполняемый код размещается начиная со смещения 100h и дальше. Предшествующая часть сегмента (смещения 00h – FFh) известна как область PSP (Program Segment Prefix), или префикс сегмента программы. Область PSP заполняется важными служебными данными, которые используются функциями DOS и могут быть использованы самой исполняемой программой.

С помощью отладчика DEBUG.EXE проще всего "подглядеть" ту область PSP, которую для него формирует командный интерпретатор COMMAND.COM. Процедура вывода части этой области PSP на экран показана на рис. 11. Оставшаяся не показанной часть этой области PSP заполнена нулями.



```
E:\DOS\MSDOSDOC\PICT_RAW>debug.exe cjpeg.exe -baseline vc01.bmp
-a100
22D4:0100 mov AH,62                     ; This call returns DEBUG's
22D4:0102 int 21                        ;    segment address in BX register
22D4:0104 nop
22D4:0105
-g=100 104

AX=6200  BX=0D18  CX=6074  DX=0000  SP=0080  BP=0000  SI=0000  DI=0000
DS=0D18  ES=0D18  SS=236D  CS=22D4  IP=0104   NV UP EI PL NZ NA PO NC
22D4:0104 90            NOP
-d 0D18:0000 LC0
0D18:0000  CD 20 FF 9F 00 9A F0 FE-1D F0 4F 03 7C 06 8A 03   . ........O.¦...
0D18:0010  7C 06 17 03 7C 06 6B 06-01 01 01 00 02 FF FF FF   ¦...¦.k.........
0D18:0020  FF FF FF FF FF FF FF FF-FF FF FF FF 08 0D 64 3E   ..............d>
0D18:0030  7C 06 14 00 18 00 18 0D-FF FF FF FF 00 00 00 00   ¦...............
0D18:0040  07 0A 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0D18:0050  CD 21 CB 00 00 00 00 00-00 00 00 00 00 2D 42 41   .!...........-BA
0D18:0060  53 45 4C 49 4E 20 20 20-00 00 00 00 00 56 43 30   SELIN    .....VC0
0D18:0070  31 20 20 20 20 42 4D 50-00 00 00 00 00 00 00 00   1    BMP........
0D18:0080  14 20 2D 62 61 73 65 6C-69 6E 65 20 76 63 30 31   . -baseline vc01
0D18:0090  2E 62 6D 70 20 0D 76 63-30 31 2E 62 6D 70 20 0D   .bmp .vc01.bmp .
0D18:00A0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0D18:00B0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
-
```

Рис. 11

Роль отлаживаемой программы на рис. 11 играет файл CJPEG.EXE. Параметры "-baseline" и "VC01.BMP" передаются отлаживаемой программе, чтобы показать пример заполнения первого и второго блоков FCB, начинающихся со смещений 5Ch и 6Ch соответственно. Расположение упомянутых и некоторых других полей данных в области PSP пояснено в приведенной ниже таблице.

Смещение	Длина	Содержание
00h	2	Команда INT 20 (для совместимости с CP/M)
02h	2	1-й занятый сегмент после сегмента данного PSP
06h	2	Размер кода в сегменте (для .COM-файлов)
0Ah	4	Запомненный адрес возврата для INT 22
0Eh	4	Запомненный адрес обработчика INT 23
12h	4	Запомненный адрес обработчика INT 24
16h	2	Сегмент родительского PSP (примечания 1 и 2)
18h	20	Таблица Job File Table (JFT, примечание 3)
2Ch	2	Сегмент области окружения данной программы
2Eh	4	SS:SP перед последним вызовом INT 21
32h	2	Число ссылок в таблице JFT (по умолчанию 20)
34h	4	Указатель на таблицу JFT (обычно PSP:0018h)
3Ch	1	= 00h (= 01h – для иероглифических клавиатур)
40h	2	Номер версии DOS для INT 21\AH=30h
50h	2	Вызова обработчика прерывания INT 21
5Ch	16	Первый блок FCB (примечание 4)
6Ch	16	Второй блок FCB (примечание 4)
80h	1	Длина командной строки (примечания 5 и 6)
81h	127	Копия командной строки и область DTA

Примечание 1: если вместо сегментного адреса родительского PSP в слове со смещением 16h записан собственный сегментный адрес PSP данной программы, то эта программа считается не имеющей "прародителя". Так отмечаются постоянно загруженные программы, например, командный интерпретатор. Эти программы невозможно закрыть вызовом прерывания INT 20 или INT 21\AH=4Ch.

Примечание 2: при защищенном режиме работы компьютера в PSP могут быть внесены дополнительные данные, перезаписывающие некоторые поля PSP, в том числе сегментный адрес родительского PSP в слове со смещением 16h. Поэтому при прослеживании цепи ссылок на родительские PSP необходимо убедиться в том, что адрес действительно указывает на PSP, например, посредством проверки характерных сигнатур: CD20h (INT 20) начиная со смещения 00h или CD21h (INT 21) начиная со смещения 50h.

Примечание 3: в момент запуска программы на исполнение таблица JFT (смещение 18h в PSP) содержит однобайтовые номера блоков описания, входящих в таблицу SFT (А.01-4). Эти номера соответствуют блокам описания тех "открытых" объектов, которые унаследованы от родительского процесса. Номерами 80h – FEh отмечаются блоки описания файлов, доступ к которым осуществляется через сеть.

Оставшееся свободное пространство JFT заполняется байтами FFh. Начальный размер JFT – 20 байтов – ограничивает число объектов, которые можно держать открытыми одновременно. Это ограничение можно преодолеть путем формирования таблицы JFT большего размера за пределами PSP (8.02-76) и помещения указателя на нее в PSP начиная со смещения 34h. Но надо иметь ввиду, что дочерние процессы все равно не смогут наследовать более 20 открытых объектов.

Примечание 4: поля, начинающиеся со смещений 5Ch и 6Ch, предназначены для неоткрытых блоков FCB (A.09-5). Туда записываются первый и второй параметры командной строки, предварительно подвергнутые "разбору" с помощью INT 21\AX=2901h (8.02-19), при этом счет параметров включает те, которые "разбору" не подлежат.

Примечание 5: область 81h – FFh заполняется копией командной строки запуска текущей программы. Заполненная часть заканчивается байтом 0Dh. Длина заполненной части записывается в байт со смещением 80h. Если длина равна 7Fh, и в байт со смещением FFh занесено значение 0Dh, то, значит, реальная длина командной строки превосходит 126 байтов, и тогда неурезанную версию командной строки следует искать в переменной окружения CMDLINE.

Примечание 6: область 80h – FFh также по умолчанию используется в качестве DTA (Data Transfer Area) при операциях поиска обработчиками прерываний INT 21\AH=11h,12h,4Eh,4Fh. Если записанная там же копия командной строки должна быть сохранена, то надо сменить адрес DTA с помощью INT 21\AH=1Ah (8.02-16).

A.07-2 Блок параметров загрузки программы

В приведенной ниже таблице показана структура блока параметров, используемого обработчиками прерываний INT 21\AX=4B00h и INT 21\AX=4B01h (8.02-53) при загрузке программы в память с целью ее последующего исполнения.

Смещение	Длина	Содержание
00h	2	Сегмент окружения дочернего процесса
02h	4	Указатель на командную строку (примечание 2)
06h	4	Адрес данных для FCB 5Ch (примечание 3)
0Ah	4	Адрес данных для FCB 6Ch (примечание 3)
0Eh	4	При возврате: адрес вершины стека SS:SP
12h	4	При возврате: адрес вызова CS:IP (примечание 4)

Примечание 1: в сегмент окружения дочернего процесса будет скопировано окружение вызывающего (родительского) процесса. Если дочернему

процессу должна быть предоставлена не копия, а доступ к окружению вызывающей программы, то в слове со смещением 00h вместо сегментного адреса следует указать значение 0000h.

Примечание 2: здесь следует указать все, что должно быть записано в PSP дочернего процесса, начиная с адреса 80h (примечание 5 к A.07-1). Строка должна начинаться с байта, указывающего ее длину, и кончаться байтом 0Dh.

Примечание 3: эта строка данных будет скопирована в соответствующий блок FCB (примечание 4 к A.07-1) в PSP дочернего процесса. Структура строки показана в колонке "N" таблицы A.09-5, заполнению подлежат первые 12 байт, потом следуют 4 байта 00h. Если данный блок FCB не должен быть заполнен, то за первым байтом 00h должны следовать 11 байтов 20h.

Примечание 4: указатели в ячейках со смещениями 0Eh и 12h относятся только к функции INT 21\AX=4B01h, которая загружает программу, но не инициирует ее исполнение. Чтобы инициировать ее исполнение позже, функция INT 21\AX=4B01h возвращает адрес вызова загруженной программы и указатель на вершину ее стека.

A.07-3 Дескриптор состояния исполнения

Приведенная ниже таблица показывает структуру данных в дескрипторе, используемом при вызове прерывания INT 21\AX=4B05h (8.02-54).

Смещение	Длина	Содержание
00h	2	= 0000h (резервировано)
02h	2	Флаги: бит 0: исполнять как программу *.EXE бит 1: загружаемый код – это оверлей
04h	4	Указатель на имя программы с байтом 00h в конце
08h	2	Сегментный адрес PSP загружаемой программы
0Ah	4	Адрес вызова загружаемой программы (CS:IP)
0Eh	4	Размер загружаемой программы, включая ее PSP

A.07-4 Блок параметров для вызова серверной функции

Поскольку серверная функция INT 21\AX=5D00h (8.02-68) исполняет каждый вызов прерывания INT 21 как отдельный процесс, постольку блок параметров, структура которого показана ниже, задает все исходные состояния регистров для исполнения той функции прерывания INT 21, которую предстоит вызвать. Перед вызовом этой функции значения из блока параметров будут скопированы в регистры автоматически.

Смещение	Длина	Содержание
00h	2	Содержимое регистра AX
02h	2	Содержимое регистра BX
04h	2	Содержимое регистра CX
06h	2	Содержимое регистра DX
08h	2	Содержимое регистра SI
0Ah	2	Содержимое регистра DI
0Ch	2	Содержимое регистра DS
0Eh	2	Содержимое регистра ES
10h	2	= 0000h (резервировано)
12h	2	Идентификатор виртуальной машины
14h	2	Идентификатор процесса, т.е. сегмент его PSP

Примечание 1: если предполагается исполнять вызываемый процесс в рамках той же действующей системы DOS, то в качестве идентификатора виртуальной машины следует указывать значение 0000h.

Примечание 2: когда показанный здесь блок параметров используется для закрывания процесса с помощью INT 21\AX=5D01h (8.02-69), тогда во внимание принимаются только слова со смещениями 12h и 14h, а все остальное содержание блока игнорируется.

A.07-5 Совместное пользование номером прерывания.

Многие драйверы и резидентные программы при загрузке должны вписывать адрес своего обработчика прерывания в определенную ячейку таблицы прерываний. Однако эта ячейка может быть уже занята адресом загруженного ранее резидентного модуля. Если прежний модуль надо заместить, то встает вопрос о том, как освободить занимаемую им память. Если новый модуль дополняет функции прежнего, то возникает проблема организации их взаимодействия. В обоих случаях необходимо, чтобы резидентный модуль предоставлял данные для организации взаимодействия в "понятной" всем другим модулям форме.

Первым шагом к организации взаимодействия резидентных модулей явился предложенный фирмой IBM протокол ISP (= Interrupt Sharing Protocol), регламентирующий наличие 16-байтового блока данных с фиксированным расположением относительно адреса вызова каждого резидентного модуля. Протокол ISP обеспечивает формирование прослеживаемой цепи ссылок на все обработчики, использующие один и тот же номер прерывания. Это дает возможность изменять порядок ссылок в цепи и удалять отдельные ссылки, что необходимо для выгрузки резидентных модулей.

Согласно протоколу ISP адрес вызова, вписываемый в таблицу прерываний, должен указывать на команду ближнего перехода на 16 байт вперед – туда, где начинается исполняемый код резидентного модуля. "Перепрыгиваемый" 16-байтный участок – это место для размещения блока данных. Состав размещаемых там данных показан в приведенной ниже таблице. Все смещения в таблице рассчитаны относительно адреса вызова обработчика прерывания.

Смещение	Длина	Содержание
00h	2	Команда перехода (EB 10) к исполняемому коду
02h	4	Адрес следующего обработчика в цепи
06h	2	Сигнатура 4Bh 42h (= "KB")
08h	1	= 00h – очередной обработчик в цепи ссылок = 80h – первоначальный обработчик прерывания
09h	2	Адрес подпрограммы выгружения модуля, оканчивающейся командой возврата RETF
0Bh	7	Резервировано (должны быть нули)

Примечание 1: далеко не все резидентные модули соответствуют протоколу ISP. Нередко его намеренно игнорируют, чтобы предотвратить прослеживание цепи ссылок или выгружение резидентного модуля.

А.07-6 Команды альтернативного мультиплексного прерывания.

При пользовании мультиплексным прерыванием INT 2F, рассматриваемым в разделе 8.03, несогласованные действия разработчиков могут приводить к назначению одинаковых идентификаторов разным резидентным модулям. Во избежание вызванных этим конфликтов Ральф Браун предложил назначать идентификаторы не заранее по усмотрению разработчиков, а автоматически в процессе загрузки модулей. Идея Ральфа Брауна реализована альтернативным мультиплексным прерыванием INT 2D и регламентирована спецификацией AMIS (Alternate Multiplex Interrupt Specification). Материал данного раздела основан на спецификации AMIS версии 3.6. Помимо того, каждый резидентный модуль, пользующийся прерыванием INT 2D, должен поддерживать формирование прослеживаемой цепи ссылок согласно протоколу ISP (А.07-5).

Идентификатор для загружаемого модуля предложено искать посредством цикла вызовов мультиплексного прерывания INT 2D с кодом операции AL=00h и с последовательным перебором идентификаторов в регистре AH, начиная с AH=00h. Если какой-либо из загруженных ранее модулей уже считает идентификатор в регистре AH "своим", то он должен установить AL=FFh, возвратить в регистрах CH:CL номер версии, а в регистрах DX:DI – указатель на сигнатуру длиной до 80 байт, оканчивающуюся кодом 00h. Выход из цикла – по условию возврата

неизмененного значения AL = 00h: значит, на заявленный идентификатор никто не претендует, и загружаемый модуль вправе присвоить его себе.

Аналогичный цикл, только с другим условием окончания, организуют чтобы определить, загружен ли какой-либо конкретный модуль. При таких проверках центральную роль играет сигнатура, указатель на которую возвращается в регистрах DX:DI. Для идентификации модуля должно быть достаточно 16 байт сигнатуры, причем первые 8 байт выделяются названию фирмы или имени разработчика, а последующие 8 байт – названию программы или драйвера, загрузивших данный резидентный модуль. Допускаются сокращенные названия. Если название короче 8 байт, его следует дополнить до 8 байт пробелами (байтами 20h). Часть сигнатуры сверх 16 байт не обязательна, но может содержать полезные сведения. Проверка по сигнатуре помогает предотвратить повторную загрузку резидентных программ. Вместе с выяснением факта загрузки интересующего модуля становится известен присвоенный ему идентификатор, по которому можно запросить другие функции того же модуля.

Поскольку вызовы мультиплексного прерывания предполагают прослеживание цепи ссылок и выполняются медленно, постольку обращаться к специфичным функциям резидентных модулей через прерывание INT 2D нецелесообразно (хотя для этого допускается использовать коды операций свыше 10h). Предпочтителен иной путь – прямое обращение к функциям модулей командой CALL FAR (7.03-08). Чтобы получить адрес прямого обращения, нужно вызвать прерывание INT 2D с идентификатором конкретного загруженного модуля в регистре AH и с кодом операции 01h в регистре AL. Искомый адрес, возвращаемый в регистрах DX:BX, надо запомнить, и тогда им можно будет пользоваться многократно.

Помимо упомянутых операций с кодами AL=00h и AL=01h, спецификация AMIS предусматривает унификацию кодов еще нескольких операций, перечисленных в первом столбце приведенной ниже таблицы. Операции с ненулевыми кодами не обязательны для исполнения. Если в ответ на вызов модуль возвращает в регистре AL код статуса 00h, значит он не поддерживает данную операцию. Напротив, возврат статуса FFh означает, что запрошенная операция поддерживается и успешно выполнена. Отдельные операции допускают возврат иных значений кода статуса, позволяющих судить о состоянии резидентного модуля. Эти и некоторые другие особенности исполнения операций пояснены в примечаниях к таблице.

Код	Значение	Примечания
00h	Проверка наличия резидентного модуля в памяти	
01h	Запрос адреса обращения к резидентному модулю	*1
02h	Деинсталляция резидентного модуля	*2
03h	Вызов резидентной программы на исполнение	*3

Продолжение таблицы А.07-6

04h	Запрос задействованных прерываний	*4
05h	Запрос списка "горячих" клавишей	*5
06h	Запрос об установленных программой драйверах	*6

Примечание 1: возврат статуса AL = 00h означает, что данный модуль не принимает обращения командой CALL FAR. Адрес в регистрах DX:BX действителен только при возврате статуса AL = FFh.

Примечание 2: при запросе деинсталляции в регистрах DX:BX должен быть указан адрес перехода после деинсталляции, хотя резидентный модуль может игнорировать этот адрес. Возвращаемые в регистре AL коды статуса (помимо 00h и FFh) означают следующее:

01h – неудача попытки деинсталляции;
02h – деинсталляция будет завершена позже;
03h – модуль не имеет деинсталлятора и остался активным;
04h – то же, что 03h, но модуль дезактивирован;
05h – попытку деинсталляции надо повторить позже;
06h – модуль дезактивирован, деинсталляция невозможна;
07h – то же, что 03h, но требуется удаление драйверов.

Возврат кодов статуса 03h, 04h или 07h означает необходимость пользования отдельной программой деинсталляции, причем для нее подлежащий удалению модуль должен возвратить в регистре BX сегментный адрес размещения своего исполняемого кода.

Примечание 3: возвращаемые в регистре AL коды статуса (помимо 00h и FFh) означают следующее:

01h – попытку активизации надо повторить позже;
02h – программа будет активизирована позже;
03h – программа уже активизирована;
04h – сбой при попытке активизации программы.

При коде статуса FFh активизируемая программа возвращает в регистре BX дополнительные сведения о своем состоянии (или просто обнуляет регистр BX). При коде статуса 04h дополнительные сведения возвращаются в регистрах BX и CX; в случае неизвестной причины сбоя эти регистры должны быть обнулены.

Примечание 4: при вызове операции 04h в регистре BL должен быть указан номер прерывания (кроме INT 2D), факт перехвата которого надлежит установить. Возвращаемые в регистре AL коды статуса (помимо 00h) означают следующее:

01h – результат проверки не определен;
02h – указанное прерывание перехвачено;
03h – то же, что 02h, плюс адрес обработчика – в DX:BX;
04h – в DX:BX – адрес списка перехваченных прерываний;
FFh – указанное прерывание не перехвачено.

При возврате кода статуса 04h номер прерывания в регистре BL игнорируется. Возвращаемый список состоит из групп по 3 байта на каждое прерывание: первый байт – номер прерывания, следующее за ним слово – смещение в том же сегменте (указанном в регистре DX) адреса вызова обработчика этого прерывания. Конец списка отмечен кодом 2Dh в позиции номера прерывания.

Примечание 5: при успешном завершении, подтверждаемом кодом статуса AL=FFh, в регистрах DX:BX возвращается указатель на список "горячих" клавишей. Содержание списка – в разделе А.02-7.

Примечание 6: операция возвращает в регистре AL число установленных данной резидентной программой драйверов, а в регистрах DX:BX – указатель на заголовок первого из этих драйверов (А.05-1). В регистре AH операция возвращает байт флагов, в котором установление в единицу отдельных битов означает следующее:

бит 0 – драйверы нельзя выгрузить из памяти;
бит 1 – драйверы не включены в цепь драйверов DOS;
бит 2 – установленные драйверы реентерабельны.

Биты 3 – 7 зарезервированы и должны быть обнулены. Если данная программа не устанавливала драйверов, то она возвращает значение AL=00h, и в этом случае содержимое регистров AH, BX и DX может быть произвольно изменено.

А.08 Таблицы параметров флоппи-дисководов

А.08-1 Сведения о флоппи-дисководах из области данных BIOS

В этой таблице приведены выборочные сведения о дисководах на гибких магнитных дисках. Все смещения указаны относительно сегментного адреса 0040h, то есть от начала области данных BIOS.

Смещение	Длина	Содержание
10h	2	Флаги: бит 0: возможна загрузка с флоппи-диска биты 6-7: число флоппи-дисководов –1
3Eh	1	Установка бита 7 обработчиком IRQ6 отмечает завершение операции флоппи-дисковода
3Fh	1	Состояние моторов флоппи-дисководов
40h	1	Счетчик времени для выключения мотора
41h	1	Биты 0-4: код завершения (примечание 2) бит 5:отказ контроллера; бит 6: ошибка при поиске;

Продолжение таблицы A.08-1

		бит 7: дисковод не готов к исполнению операции
42h	3	Регистры контроллера флоппи-дисководов
8Bh	1	Скорость передачи, заданная контроллером
8Fh	1	Бит 0: дисковод 0 поддерживает 80 дорожек; бит 2: наличие дисковода 0 зарегистрировано; бит 4: дисковод 1 поддерживает 80 дорожек; бит 6: наличие дисковода 1 зарегистрировано.
90h	1	Статус носителя в дисководе 0 биты 0-2: = 111b для дискет 3.5" бит 3: дискета емкостью 2.88 Мбайт бит 4: если = 0, тип дискеты не определен биты 6-7: скорость обмена
91h	1	Статус носителя в дисководе 1 (как в байте 90h)
94h	1	Номер дорожки, где стоит головка дисковода 0
95h	1	Номер дорожки, где стоит головка дисковода 1

Примечание 1: представленное здесь расположение данных может зависеть от версии BIOS (об этом также в A.01-1).

Примечание 2: конкретные значения байта 41h расшифровываются так, как указано в приложении A.06-1 для прерывания INT 13.

A.08-2 Параметры доступа и форматирования

Система BIOS хранит параметры доступа и форматирования для каждого флоппи-дисковода в отдельных 11-байтовых таблицах. Для любого конкретного дисковода адрес такой таблицы можно получить с помощью INT 13\AH=08h (8.01-49). Указатель на одну из этих таблиц, относящуюся к принимаемому по умолчанию ("текущему") флоппи-дисководу, записывается в ячейку 0000:0078h в таблице прерываний (он известен еще как INT 1E).

Смена данных в таблицах параметров доступа и форматирования производится вызовом INT 13\AH=18h (8.01-54), однако измененные параметры будут приняты во внимание контроллером флоппи-дисководов только после установления его начального состояния посредством INT 13\AH=00h (8.01-44).

Смещение	Длина	Содержание
00h	1	Биты 7-4: скорость перемещения головки; биты 3-0: время выгрузки головки (0Fh = 0.24 с).
01h	1	Биты 7-1: время загрузки головки (01h = 0.004 с); бит 0: = 0 – обмен без использования DMA.
02h	1	Время до остановки вращения (в тактах 1/18 с).
03h	1	= 00h – размер сектора 128 байт,

Продолжение таблицы А.08-2

		= 01h – размер сектора 256 байт, = 02h – размер сектора 512 байт, = 03h – размер сектора 1024 байт.
04h	1	Число секторов на дорожке
05h	1	Размер промежутка между секторами: =2Ah для дискет 5.25", =1Bh для дискет 3.5".
07h	1	Размер промежутков при форматировании: =50h для дискет 5.25", =6Ch для дискет 3.5".
08h	1	Заполняющий байт при форматировании (F6h)
09h	1	Время выведения головки на дорожку (в мс).
0Ah	1	Время раскрутки мотора (в тактах 1/18 с).

А.08-3 Регистрируемые BIOS типы флоппи-дисководов.

Типы имеющихся в компьютере флоппи-дисководов, указываемые в настройках программы BIOS Setup, записываются в ячейку 10h CMOS-памяти системы BIOS. Чтобы считать эти сведения из исполняемой программы, надо сначала командой OUT (7.03-66) послать адрес ячейки (10h) в порт 70h, а затем командой IN (7.03-26) считать искомый байт данных из порта 71h (еще об этом – примечание 1 к А.14-1). Считанный байт данных оказывается в регистре AL, причем биты 4 – 7 характеризуют тип первого флоппи-дисковода, а биты 0 – 3 – тип второго флоппи-дисковода, если он имеется. Значение каждой из двух групп по 4 бита расшифровывается независимо согласно следующей таблице.

Значение	Тип флоппи-дисковода
0	Дисковод отсутствует
1	Дисковод для дискет 5.25 дюйма емкостью 360 кбайт
2	Дисковод для дискет 5.25 дюйма емкостью 1,2 мегабайта
3	Дисковод для дискет 3.5 дюйма емкостью 720 кбайт
4	Дисковод для дискет 3.5 дюйма емкостью 1,44 мегабайта
5	Дисковод для дискет 3.5 дюйма емкостью 2,88 мегабайта

А.09 Структуры данных для файлов и каталогов

А.09-1 Записи в каталогах и сведения о файлах

Сведения о файлах, метках тома и подкаталогах хранятся в соответствующих записях каталогов. Структура данных в обычной 32-байтовой записи для объектов

со стандартным "коротким" именем показана в первой колонке "D" приведенной ниже таблицы. На основе этих данных процедуры поиска INT 21\AX=4E00h (8.02-57) и INT 21\AH=4Fh (8.02-58) возвращают через область DTA (8.02-16) сведения о найденных объектах; формат выдаваемых ими результатов поиска показан во второй колонке "F4E" приведенной ниже таблицы. Две другие поисковые процедуры INT 21\AH=11h (8.02-11) и INT 21\AH=12h (8.02-12) также выдают сведения о найденных объектах в области DTA, но в других форматах. Формат, показанный в третьей колонке "F1N" используется тогда, когда при запросе исходные данные представлены обычным блоком FCB (А.09-5, колонка "N"). Если же при запросе исходные данные представлены расширенным блоком FCB (А.09-5, колонка "E"), то результаты поиска выдаются в формате, показанном в четвертой колонке "F1E" приведенной ниже таблицы.

D	F4E	F1N	F1E	Длина	Содержание
			00h	1	= FFh – признак расширенного FCB
			06h	1	Атрибуты (А.09-2) для поиска
	00h	00h	07h	1	Диск, 01h = A и т.д, (примечание 1)
00h	01h	01h	08h	8	Имя объекта, дополненное пробелами до 8 байт
08h	09h	09h	10h	3	Суффикс, дополненный пробелами до 3 байт
	0Ch			1	Атрибуты (А.09-2) для поиска
	0Dh			2	Номер записи в каталоге
	0Fh			2	Начальный кластер каталога
0Bh	15h	0Ch	13h	1	Атрибуты найденные (А.09-2)
0Ch				1	Атрибуты (примечания 2 и 3)
0Dh		0Eh	15h	1	Время в 0.01 с. (примечание 3)
0Eh		0Fh	16h	2	Время создания (примечание 3)
10h		11h	18h	2	Дата создания (примечание 3)
12h		13h	1Ah	2	Дата последнего доступа к объекту
14h		15h	1Ch	2	Начальный кластер (примечание 4)
16h	16h	17h	1Eh	2	Время последнего изменения
18h	18h	19h	20h	2	Дата последнего изменения
1Ah		1Bh	22h	2	Начальный кластер (примечание 4)
1Ch	1Ah	1Dh	24h	4	Двоичный размер объекта в байтах
	1Eh			13	Имя и суффикс (примечание 5)

Примечание 1: поисковые процедуры INT 21\AX=4E00h (8.02-57) и INT 21\AH=4Fh (8.02-58) устанавливают в единицу бит 7 в этом байте, если доступ к данному диску осуществляется через сеть.

Примечание 2: этим полем пользуются операционные системы Windows-2000/XP, но сведения о записанных туда данных не раскрыты.

Примечание 3: если объект создан под управлением операционной системы DOS, то это поле остается незаполненным. При копировании объектов в среде DOS данные из этого поля не копируются.

Примечание 4: в логических дисках с файловой системой FAT-16 номер начального кластера объекта представляет собой одно слово в ячейке со смещением 1Ah, а поле 14h не используется. Но в логических дисках с файловой системой FAT-32 номер начального кластера представляет собой двойное слово, и два старших байта из этого двойного слова записаны в ячейку со смещением 14h.

Примечание 5: поисковые процедуры INT 21\AX=4E00h и INT 21\AH=4Fh не перезаписывают имя найденного объекта поверх маски искомого имени, начинающейся со смещения 01h; при возврате из процедуры имя и суффикс найденного объекта, оканчивающиеся байтом 00h, указываются отдельно начиная со смещения 1Eh.

А.09-2 Структура байта атрибутов

Байт атрибутов со смещением 0Bh в составе записи каталога (А.09-1) определяет класс объекта, к которому данная запись относится. Структура байта атрибутов показана в приведенной ниже таблице.

Бит	Значение
0	Файл только для чтения
1	Скрытый файл
2	Системный файл
3	Метка тома (= 0b для файлов и каталогов)
4	Каталог (= 0b для файлов и меток тома)
5	Файл, подлежащий архивированию
6,7	= 00b, не используются в MS-DOS

Примечание 1: значение 0Fh байта атрибутов – отличительный признак тех записей каталога, которые представляют "длинные" имена (А.09-3), назначаемые операционными системами Windows-95/98/ME.

Примечание 2: состояния битов 3 и 4 не могут быть изменены посредством INT 21\AX=4301h (8.02-39) или с помощью ATTRIB.EXE (6.01).

Примечание 3: процедуры поиска файлов INT 21\AX=4E00h (8.02-57) и INT 21\AH=4Fh (8.02-58) игнорируют состояния битов 0 и 5.

Примечание 4: обработчик прерывания INT 21\AX=6C00h (8.02-78) принимает из регистра CX слово атрибутов, в котором биты 4 и 6 – 15 должны быть сброшены в нуль, а назначение остальных битов соответствует приведенной здесь таблице.

А.09-3 Формат записей каталога, относящихся к "длинным" именам

Каждое "длинное" имя, назначенное операционными системами Windows-95/98/ME, занимает в каталоге по крайней мере несколько 32-байтовых записей. Последняя из них содержит укороченный вариант "длинного" имени и по структуре соответствует колонке "D" таблицы А.09-1. Но предшествующие записи, в которых "длинное" имя по частям записано знаками 16-битового уникода (Unicode), имеют другую структуру, показанную в приведенной ниже таблице.

Смещение	Длина	Содержание
00h	1	Порядковый номер записи (примечание 1)
01h	10	Знаки "длинного" имени, первая часть
0Bh	1	= 0Fh – запись принадлежит "длинному" имени
0Ch	1	= 00h (резервировано)
0Dh	1	Контрольная сумма (примечание 2)
10h	12	Знаки "длинного" имени, вторая часть
1Ah	2	= 0000h для всех записей "длинных" имен
1Ch	4	Знаки "длинного" имени, третья часть

Примечание 1: последняя запись, относящаяся к данному "длинному" имени, отмечается установлением в единицу бита 6 в первом байте.

Примечание 2: контрольная сумма короткого имени вычисляется сложением всех его 11 знаков со смещением суммы на один бит вправо перед суммированием с каждым очередным знаком.

А.09-4 Структура байта условий доступа

Открывая для доступа каждый очередной объект, обработчики прерываний INT 21\AH=3Dh (8.02-33) и INT 21\AX=6C00h (8.02-78) принимают байт условий доступа и записывают его в ячейку со смещением 02h в блоке описания, создаваемом для данного объекта в составе таблицы SFT (А.01-4). Структура байта условий доступа показана в приведенной ниже таблице.

Бит	Значение
1-0	Цели доступа: 00b – только для считывания 01b – только для записи 10b – для записи и считывания 11b – для пересылки и исполнения
2	Не корректировать время последнего доступа
3	= 0b (резервировано)

Продолжение таблицы А.09-4

6-4	Условия разделенного (совместного) доступа: 000b – обеспечить режим совместимости 001b – запретить совместный доступ 010b – запретить запись другим программам 011b – запретить считывание другим программам 100b – предоставить все права другим программам
7	Право доступа не наследуется дочерними процессами.

Примечание 1: условия совместного доступа принимаются во внимание только если загружена резидентная программа SHARE.EXE.

Примечание 2: при работе в предыдущих версиях DOS бит 2 должен быть сброшен в нуль.

А.09-5 Неоткрытые блоки управления файлами

Блоки управления файлами (FCB = File Control Block) представляют собой устаревшую форму спецификации, не обеспечивающую доступа за пределами "текущего" каталога и непригодную для открытия файлов на дисках с файловой системой FAT-32. Однако форма блока FCB используется рядом операций не для доступа к файлам, а просто как шаблон при поиске объектов, при их переименовании и удалении. Такие операции (INT 21\AH=11h, 12h, 13h, 17h) успешно применяют по отношению к объектам, находящимся в текущем каталоге, в том числе на дисках с файловой системой FAT-32. Для этого полная ("открытая") форма блоков FCB не нужна, достаточно частично заполненных "неоткрытых" блоков FCB, структура которых показана в приведенной ниже таблице.

MS-DOS7 допускает два варианта блоков FCB: обычные длиной до 36 байт и расширенные, длиной до 43 байт. Обычные блоки FCB применимы только к файлам, не имеющим атрибутов H (скрытый) и S (системный). Расширенные блоки FCB отличаются меткой FFh в первом байте. Они позволяют указать атрибуты объектов поиска и потому применимы не только к любым файлам, но также к меткам тома и к подкаталогам. Различия между обычными и расширенными блоками FCB имеются и у их "неоткрытых" форм. Колонка "N" приведенной ниже таблицы показывает размещение данных в неоткрытых обычных блоках FCB, а колонка "E" – в неоткрытых расширенных блоках FCB. В те байты блока FCB, которые в таблице не упомянуты, должно быть записано значение 00h.

N	E	Длина	Содержание
	00h	1	= FFh – признак расширенного блока FCB
	06h	1	Спецификация атрибутов (А.09-2) для поиска
00h	07h	1	Номер логического диска (примечание 1 к 8.02-17)
01h	08h	8	Имя или маска имени объекта (примечание 1)

Продолжение таблицы А.09-5

09h	10h	3	Суффикс или маска суффикса (примечание 1)
0Ch	13h	1	При возврате: атрибуты поиска (из байта 06h)
0Dh	14h	2	При возврате: номер записи об объекте в каталоге
0Fh	16h	2	При возврате: номер кластера текущего каталога
11h	18h	8	При вызове INT 21\AH=17h: новое имя
15h	1Ch	1	При возврате: номер диска (01h=A:, 03h=C:, и т.д.)
19h	20h	3	При вызове INT 21\AH=17h: новый суффикс

Примечание 1: в FCB все строчные буквы имени и суффикса надо заменять заглавными. Имя дополняется пробелами (20h) до номинальной длины 8 байт, суффикс – до длины 3 байта. В незаполненных блоках FCB поля имени и суффикса содержат только пробелы. Все перечисленные и некоторые другие требования к заполнению полей FCB будут строго соблюдены, если для формирования FCB воспользоваться функцией INT 21\AH=29h (8.02-19)

Примечание 2: при первом вызове всех функций, кроме INT 21\AH=17h, поля после 0Bh в обычном FCB и после 12h в расширенном FCB должны быть заполнены значением 00h. Функции поиска возвращают в этих полях данные, которые надо передавать от предшествующего вызова к следующему. Функция INT 21\AH=17h принимает из тех же полей новое имя для переименовываемого файла и требует буфера длиной 28 байт для обычного FCB и 35 байт для расширенного FCB.

Примечание 3: неоткрытые блоки FCB не подвержены ограничению, налагаемому спецификацией команды FCBS (4.10) в файле Config.sys.

А.09-6 Канонизированный формат записи для каталогов диска CD-ROM

Структуры каталогов в оптических дисках форматов High Sierra и ISO 9660 несколько отличаются, но функция INT 2F\AX=150Fh (8.03-19) приводит обе структуры к единому канонизированному виду, который показан в приведенной ниже таблице.

Смещение	Длина	Содержание
00h	1	Длина области признаков в логических блоках
01h	4	Номер начального логического блока файла
05h	2	Размер файла в логических блоках
07h	4	Длина файла в байтах
0Bh	7	Дата и время
12h	1	Флаги
13h	1	Интервал перемежения (только для файлов AVI)
14h	1	Фактор пропуска перемежения (только для AVI)

Продолжение таблицы A.09-6

15h	2	Номер последовательности установки тома
17h	1	Длина имени файла
18h	38	Имя файла, оканчивающееся байтом 00h
3Eh	2	Номер версии файла
40h	1	Число байтов в системном блоке данных
41h	220	Системный блок данных

A.10 Таблицы данных для видеосистемы

A.10-1 Некоторые видеорежимы

Видеорежимы определяют способы формирования изображений, воспроизводимых на экране дисплея. Системы BIOS и DOS обычно выводят свои данные в текстовых видеорежимах: цветном видеорежиме 03h или монохромном видеорежиме 07h. Каждой программе предоставлена возможность установить наиболее подходящий видеорежим, текстовый или графический.

Круг реализуемых видеорежимов, конечно, зависит от оборудования компьютера. Некоторые видеорежимы получили широкое распространение и стали базой для обеспечения совместимости аппаратных средств. В приведенной ниже таблице показаны только такие видеорежимы, которые почти наверняка поддерживаются любой современной видеокартой. Но устаревшие видеокарты с небольшим объемом памяти не поддерживают графические видеорежимы с высокой разрешающей способностью. Компьютеры, выпущенные до 1991 года, вообще не поддерживают все видеорежимы SVGA.

Видеорежимы EGA и VGA обозначают однобайтовым кодом, указанным в первой колонке приведенной ниже таблицы. Такие видеорежимы используют фиксированное расположение видеобуфера в адресном пространстве памяти компьютера, показанное в 5-й колонке таблицы. Установить такие видеорежимы можно с помощью INT 10\AH=00h (8.01-10).

Видеорежимы SVGA обозначают двухбайтовым шестнадцатеричным кодом, записываемым в регистр BX перед вызовом процедуры смены видеорежима INT 10\AX=4F02h (8.01-37). В приведенной ниже таблице старший полубайт кода видеорежимов SVGA не указан, потому что в нем 12-й и 13-й биты должны быть нулевыми, 14-й бит разрешает прямое обращение к кадровому буферу, а 15-й бит запрещает сброс видеопамяти при переключении видеорежима. Например, Вы можете задать BX=0102h, если хотите, чтобы видеопамять была обнулена, или можете задать BX=8102h, если хотите, чтобы содержимое видеопамяти было бы сохранено, но в обоих случаях будет установлен один и тот же видеорежим,

который обозначен как 102h в первой колонке приведенной ниже таблицы. Другие видеорежимы SVGA обозначены аналогичным образом – без указания старшего полубайта. Если с помощью той же процедуры INT 10\AX=4F02h Вам предстоит установить видеорежим EGA или VGA, то надо будет задать состояния битов 14 и 15 регистра BX согласно их роли, в младший байт регистра BX записать код устанавливаемого видеорежима, а остальные биты 8 – 13 обнулить.

В 3-й колонке приведенной ниже таблицы текстовые видеорежимы характеризованы числом знакомест в ряду и числом рядов знаков на экране. Например, цифры 80x25 в 3-й колонке означают, что Вы можете адресовать ряды 0 – 24 и знакоместа 0 – 79 в каждом ряду. Графические видеорежимы характеризуются числом строк и разрешающей способностью в пикселах, показанными в 4-й колонке. Например, разрешающая способность 640x480 означает, что Вы можете обращаться к строкам 0 – 479 и к пикселам 0 – 639 в каждой строке. Монохромные видеорежимы, как текстовые, так и графические, помечены во второй колонке таблицы значением "b/w".

Видеорежим	Цвета	Текст	Графика	Адрес	Класс
01h	16	40x25		B800	VGA
03h	16	80x25		B800	VGA
06h	b/w		640x200	B800	EGA,VGA
07h	b/w	80x25		B000	VGA
0Eh	16		640x200	A000	EGA,VGA
0Fh	b/w		640x350	A000	EGA,VGA
10h	16		640x350	A000	VGA
11h	b/w		640x480	A000	VGA
12h	16		640x480	A000	VGA
13h	256		320x200	A000	VGA
100h	256		640x400	*1	SVGA
101h	256		640x480	*1	SVGA
102h	16		800x600	*1	SVGA
103h	256		800x600	*1	SVGA
104h	16		1024x768	*1	SVGA
105h	256		1024x768	*1	SVGA
108h	16	80x60		*1	SVGA
109h	16	132x25		*1	SVGA
10Ah	16	132x43		*1	SVGA
10Bh	16	132x50		*1	SVGA
10Ch	16	132x60		*1	SVGA
110h	32k		640x480	*1	SVGA
111h	64k		640x480	*1	SVGA
112h	16M		640x480	*1	SVGA

Продолжение таблицы A.10-1

113h	32k		800x600	*1	SVGA
114h	64k		800x600	*1	SVGA
115h	16M		800x600	*1	SVGA
116h	32k		1024x768	*1	SVGA
117h	64k		1024x768	*1	SVGA
118h	16M		1024x768	*1	SVGA
119h	32k		1280x1024	*1	SVGA
11Ah	64k		1280x1024	*1	SVGA
11Bh	16M		1280x1024	*1	SVGA
120h	256		1600x1200	*1	SVGA

Примечание 1: размеры и положение "окон" доступа к видеопамяти в режимах SVGA зависят от используемой видеокарты. Параметры доступа можно выяснить вызовом INT 10\AX=4F01h (8.01-36, A.10-7).

Примечание 2: графический видеорежим 102h (800x600x16) до принятия стандарта SVGA обозначался кодом 6Ah, и посредством этого кода может быть установлен с помощью INT 10\AH=00h (8.01-10).

Примечание 3: стандарт SVGA резервирует код BX=81FFh для обозначения специального видеорежима, открывающего неограниченный доступ к встроенной памяти видеокарты.

A.10-2 Сведения о статусе видеосистемы

Здесь показана структура 64-байтового блока данных, возвращаемого обработчиком прерывания INT 10\AH=1Bh (8.01-34) и характеризующего текущее состояние видеосистемы компьютера.

Смещение	Длина	Содержание
00h	4	Адрес таблицы стат. функциональности (A.10-3)
04h	1	Установленный видеорежим
05h	2	Число элементов адресации по горизонтали
07h	2	Размер буфера регенерации в байтах
09h	2	Начальный адрес буфера регенерации
0Bh	16	8 позиций курсора для видеостраниц 0 – 7
1Bh	2	Начальная и конечная строки курсора
1Dh	1	Номер активной видеостраницы
1Eh	2	Адрес порта контроллера CRT
20h	2	Значения, посланные в порты 03x8h и 03x9h
22h	1	Номер последней строки
23h	2	Число байтов на знак в таблице шрифта
25h	1	Код активного видеоадаптера
26h	1	Код второго видеоадаптера, если он имеется

Продолжение таблицы А.10-2

27h	2	Число цветов (= 0000h для b/w видеорежимов)
29h	1	Число видеостраниц действующего видеорежима
2Ah	1	Число строк растра (примечание 1)
2Bh	1	Первый блок шрифта знакогенератора
2Ch	1	Второй блок шрифта знакогенератора
2Dh	1	Флаги текущего состояния (примечание 2)
31h	1	Видеопамять, 00h – 03h означают 64 – 256 кбайт
32h	1	Флаги, такие же как в байте 0Eh таблицы А.10-3.

Примечание 1: число строк растра определяется установлением в единицу одного бита в байте со смещением 2Ah, причем биты 0, 1, 2, 3, 4, 5, 6 соответствуют числам строк 200, 350, 400, 480, 512, 600, 768.

Примечание 2: установление битов в байте флагов 2Dh означает следующее:

бит 0 – ограничений на установление видеорежимов нет
бит 1 – включено полутоновое суммирование
бит 2 – включен монохромный знакогенератор
бит 3 – запрет загрузки произвольных палитр
бит 4 – осуществляется эмуляция курсора
бит 5 – роль бита 7 в байте цвета (А.10-5)
бит 6 – шрифты 9-точечной ширины не поддерживаются.

Если бит 5 в байте флагов сброшен в нуль, то бит 7 в байте цвета изменяет яркость фона, иначе он задает мерцание знака.

А.10-3 Формат таблицы статической функциональности

Таблица статической функциональности характеризует то множество видеорежимов, которое способна поддерживать видеокарта компьютера. Указатель на таблицу статической функциональности выдается в первых четырех байтах таблицы сведений о статусе видеосистемы (А.10-2), возвращаемой обработчиком прерывания INT 10\AH=1Bh (8.01-34).

Смещение	Длина	Содержание
00h	7	Установление битов 00h – 13h означает поддержку видеорежимов 00h – 13h. Роль битов после 13h зависит от изготовителя видеокарты.
07h	1	Биты 0 – 6 отмечают поддержку растров с числом строк 200, 350, 400, 480, 512, 600, 768.
08h	1	Доступное число блоков шрифта знакогенератора
09h	1	Число одновременно используемых шрифтов
0Ah	2	Слово поддерживаемых функций: бит 0 – произвольный выбор видеорежима бит 1 – поддержка полутонового суммирования

Продолжение таблицы А.10-3

		бит 2 – поддержка загрузки шрифтов бит 3 – возможность запрета загрузки палитр бит 4 – поддержка эмуляции курсора бит 5 – имеется встроенная палитра EGA бит 6 – встроенная цветностная палитра бит 7 – постраничные цветностные регистры бит 8 – световое перо (INT 10\AH=04h) бит 9 – сохранение и восстановление состояния бит 10 – поддержка бита 7 байта цвета (А.10-5) бит 11 – поддержка нескольких видеокарт
0Eh	1	Байт поддержки шрифтов и палитр: бит 0 – поддержка 512-знаковых шрифтов бит 1 – динамическая область сохранения бит 2 – перезагрузка текстовых шрифтов бит 3 – перезагрузка графических шрифтов бит 4 – поддержка перезагрузки палитр бит 5 – поддержка расширений видеоадаптера

А.10-4 Сведения о SVGA-расширениях системы BIOS

Здесь приведены выборочные сведения из 512-байтового блока данных, возвращаемого обработчиком прерывания INT 10\AX=4F00h (8.01-35). Сведения характеризуют программное обеспечение, "зашитое" в микросхему постоянной памяти видеокарты.

Смещение	Длина	Содержание
00h	4	Сигнатура "VESA" или "VBE2"
04h	2	Номер версии SVGA-расширений системы BIOS
06h	4	Указатель на имя фирмы-изготовителя
0Eh	4	Адрес списка поддерживаемых видеорежимов, оканчивающегося словом FFFFh
12h	2	Размер имеющейся видеопамяти, выраженный числом блоков по 64 кбайт каждый.

А.10-5 Коды цветности 16-цветных видеорежимов

Из множества доступных видеорежимов и система BIOS, и MS-DOS7 по умолчанию выбирают 16-цветный текстовый видеорежим 03h, имеющийся во всех AT-совместимых компьютерах. Далее здесь дана расшифровка четырехбитовых кодов, определяющих цвета знаков и фона в видеорежиме 03h и в других 16-цветных видеорежимах (А.10-1).

0000	0	черный	1000	8	серый
0001	1	синий	1001	9	ярко-синий
0010	2	зеленый	1010	10	ярко-зеленый
0011	3	голубой	1011	11	ярко-голубой
0100	4	красный	1100	12	ярко-красный
0101	5	пурпурный	1101	13	ярко-пурпурный
0110	6	коричневый	1110	14	желтый
0111	7	белый	1111	15	яркий белый

На основе показанных кодов формируются байты цветности (иногда называемые также байтами атрибутов). Чередующимися байтами цветности и знаковыми байтами заполняется видеопамять в текстовых видеорежимах. В каждом байте цветности биты 3 – 0 характеризуют цвет знака, а биты 6 – 4 – цвет фона. По умолчанию бит 7 характеризует не яркость фона, а мерцание знака, однако эта его роль может быть переопределена с помощью INT 10\AX=1003h (8.01-23), и тогда биты в обеих четырехбитовых группах будут иметь одинаковое значение. Роль бита 3, по умолчанию определяющего яркость знака, также может быть переопределена с помощью INT 10\AX=1103h (8.01-28) и тогда бит 3 будет направлять выбор знака к другому блоку шрифта, обеспечивая тем самым одновременное воспроизведение знаков из двух заранее загруженных шрифтов.

А.10-6 Видеоданные в области данных BIOS

В приведенной здесь таблице показаны избранные данные BIOS, которые могут иметь отношение к видеосистеме компьютера. Все значения смещений отсчитаны от сегментного адреса 0040h, то есть от начала области данных BIOS (А.01-1).

Смещение	Длина	Содержание
10h	2	Биты 4-5 определяют начальный видеорежим: 00b – выбор предоставлен видеокарте 01b – 40x25 текстовый цветной (CGA) 10b – 80x25 текстовый цветной (CGA) 11b – 80x25 монохромный текстовый.
49h	1	Установленный видеорежим (А.10-1)
4Ah	2	Число элементов адресации по горизонтали
4Ch	2	Размер видеостраницы (в байтах) в видеобуфере
4Eh	2	Начальный адрес активной видеостраницы
50h	16	Координаты курсора в каждой из 8 видеостраниц
60h	2	Начальная и конечная строки курсора (8.01-11)
62h	1	Номер активной видеостраницы
63h	2	Базовый порт CRT-контроллера (обычно 03D4h)
65h	1	Байт, посланный в порт 03B8h/03D8h:

Продолжение таблицы А.10-6

		бит 5 – управление мерцанием (8.01-23)
66h	1	Байт, посланный последним в порт 03D9h: бит 4 – яркость фона (8.01-23)
84h	1	Число элементов адресации по вертикали
85h	2	Число строк в вертикальном размере шрифта
87h	5	Флаги состояния видеокарты: бит 0: эмуляция курсора не осуществляется бит 1: применен монохромный дисплей бит 2: нужна задержка на время прогрева CRT бит 7: сохранять видеопамять при смене режима
A8h	4	Адрес блока указателей для видеорежимов VGA

Примечание 1: представленное расположение данных может быть изменено в зависимости от версии системы BIOS в Вашем компьютере.

А.10-7 Характеристики запрошенного видеорежима SVGA

Здесь представлены выборочные сведения из 256-байтового блока данных, который возвращает обработчик прерывания INT 10\AX=4F01h (8.01-36) в ответ на запрос о любом из поддерживаемых видеорежимов SVGA.

Смещение	Длина	Содержание
00h	2	Флаги: бит 0 – запрошенный режим поддерживается бит 2 – поддержка функций 8.01-21, 8.01-33 бит 3 – если сброшен, то режим монохромный бит 4 – если сброшен, то режим текстовый бит 5 – режим отличается от стандарта VGA бит 6 – переключения банков памяти нет бит 7 – линейный кадровый буфер доступен
02h	1	Окно "A": бит 0 – скользящее окно "A" существует бит 1 – окно "A" доступно для чтения бит 2 – окно "A" доступно для записи
03h	1	Окно "B": то же, что в байте 02h для окна "A"
04h	2	Шаг (в кб) смещения "окон" по видеопамяти
06h	2	Размер "окон", выраженный в килобайтах
08h	2	Начальный сегмент окна "A" (для процессора)
0Ah	2	Начальный сегмент окна "B" (для процессора)
0Ch	4	Адрес вызова подпрограммы позиционирования окон, аналогичной INT 10\AX=4F05h (8.01-39)
10h	2	Число байтов видеопамяти на 1 строку развертки

Продолжение таблицы А.10-7

12h	2	Длина строки (примечание 1)
14h	2	Размер растра по вертикали (примечание 1)
16h	1	Ширина знакоместа в пикселах
17h	1	Высота знакоместа в пикселах
18h	1	Число плоскостей видеопамяти
19h	1	Число битов видеопамяти на один пиксел
1Ah	1	Число банков памяти на видеокарте
1Bh	1	Модель заполнения видеопамяти: 00h – текстовая, попеременно знак и цвет 03h – 16-цветная графическая модель EGA 04h – графическая с "упакованными" пикселами 06h – 3 байта цветности на пиксел (HiColor) 07h – модель яркость-цветность (YUV или YIQ)
1Ch	1	Размер банка видеопамяти в килобайтах
1Dh	1	Число видеостраниц
28h	4	Физический адрес видеобуфера (VBE версии 2.0)

Примечание 1: длина строки и размер растра для графических видеорежимов выражены в пикселах, а для текстовых – числом знакомест.

А.11 Спецификации оборудования компьютера

А.11-1 Слово аппаратной конфигурации

Слово аппаратной конфигурации возвращает обработчик прерывания INT 11 (8.01-42), он считывает это слово из области данных BIOS (А.01-1) обычно по адресу 0040:0010h (адрес может зависеть от версии BIOS). Расшифровка значений битов в слове аппаратной конфигурации дана в приведенной ниже таблице.

Биты	Значение
0	Имеется флоппи-дисковод, пригодный для загрузки компьютера
1	Имеется арифметический сопроцессор
2	Под управлением BIOS имеется манипулятор "мышь"
4-5	Начальный видеорежим, расшифровываемый в таблице А.10-1
6-7	Число флоппи-дисководов минус 1, только если установлен бит 0
9-11	Число последовательных портов (COM-портов)
12	Имеется игровой порт (для джойстика)
13	Имеется встроенный модем
14-15	Число параллельных портов (LPT-портов)

А.11-2 Идентификаторы модели компьютера для драйвера HIMEM.SYS

Для обеспечения доступа к расширенной памяти компьютера драйверу HIMEM.SYS (5.04-01) необходимо правильно определить тип центрального процессора. Однако в некоторых компьютерах драйвер HIMEM.SYS не может определить тип процессора правильно, и тогда в строке вызова драйвера нужно явно указать идентификатор или кодовый номер модели компьютера.

В приведенной ниже таблице перечислены идентификаторы и соответствующие кодовые номера тех моделей компьютеров, в которых не обеспечивается определение типа центрального процессора по крайней мере некоторыми версиями драйвера HIMEM.SYS. Первое место в этой таблице (код 1) в порядке исключения занимает вполне определяемая модель IBM PC AT, потому что именно она принимается по умолчанию. Последние версии драйвера HIMEM.SYS способны правильно определять типы процессоров почти во всех перечисленных здесь моделях компьютеров, кроме Acer 1100, Wyse, и IBM 7552.

Идентификатор	Код	Модель компьютера
at	1	IBM PC AT и совместимые с ней модели
ps2	2	IBM PS/2
ptlcascade	3	Phoenix Cascade BIOS
hpvectra	4	HP Vectra (A & A+)
att6300plus	5	AT&T 6300 Plus
acer1100	6	Acer 1100
toshiba	7	Toshiba 1600 & 1200XE
wyse	8	Wyse 12.5 Mhz 286
tulip	9	Tulip SX
zenith	10	Zenith ZBIOS
at1	11	Резервировано фирмой IBM
at2	12	Резервировано фирмой IBM
css	12	CSS Labs
at3	13	Резервировано фирмой IBM
philips	13	Philips
fasthp	14	HP Vectra
ibm7552	15	IBM 7552 Industrial Computer
bullmicral	16	Bull Micral 60
dell	17	Dell XBIOS

А.11-3 Контроллер клавиатуры

Контроллер клавиатуры – это микросхема на материнской плате компьютера. В AT-совместимых компьютерах применяют разные типы контроллеров клавиатуры,

но как их роли, так и управление ими в значительной степени унифицированы. "Общение" центрального процессора с контроллером клавиатуры происходит, в основном, через порты 60h и 64h.

Порт 64h всегда открыт для считывания текущего состояния контроллера клавиатуры. Установление в единицу отдельных битов в байте состояния, принятом командой IN (7.03-26) из порта 64h, означает следующее:

бит 7 – ошибка передачи данных от клавиатуры;
бит 6 – клавиатура не отвечает контроллеру;
бит 4 – клавиатура заблокирована командой ADh;
бит 2 – самотестирование клавиатуры успешно закончено;
бит 1 – контроллер еще не успел выполнить операцию;
бит 0 – в порте 60h готов к считыванию код клавиши.

При каждом нажатии или отпускании любой клавиши контроллер выставляет соответствующий код клавиши в порт 60h и одновременно объявляет об этом, устанавливая в единицу бит 0 в порте 64h и посылая сигнал на вызов обработчика прерывания INT 09 через линию 4 своей выходной шины (примечание 1). Обработчик прерывания INT 09 (8.01-09) считывает подготовленный байт из порта 60h. Каждый акт считывания из порта 60h сбрасывает в нуль бит 0 в порте 64h.

Помимо прочего, порт 64h принимает коды операций контроллера клавиатуры, посылаемые ему посредством команды OUT (7.03-66). Поскольку контроллер работает значительно медленнее центрального процессора, постольку перед посылкой ему любого кода или любых данных необходимо ждать, пока будет сброшен в нуль бит 1 в байте состояния, считываемом из того же порта 64h: это будет означать, что контроллер завершил предыдущее действие и готов приступить к следующему. Коды наиболее важных системных операций, принимаемые контроллером клавиатуры через порт 64h, показаны в приведенной ниже таблице.

Код	Операция
ADh	Выключение (блокировка) клавиатуры
AEh	Активизация (разблокировка) клавиатуры
D1h	Открыть порт 60h для приема данных (примечание 1)
EDh	Открыть порт 60h для приема данных (примечание 2)
FEh	Сброс процессора в начальное состояние (примечание 3)

Примечание 1: при исполнении операции D1h порт 60h примет 1 байт данных и направит его в выходную шину контроллера. Биты байта данных распределяются по линиям шины следующим образом:

бит 7 – сигнал на линию данных к клавиатуре;
бит 6 – на линию тактовых импульсов к клавиатуре;
бит 4 – на линию IRQ 1 для вызова INT 09 (8.01-09);

бит 1 – к вентилю линии A20 адресной шины;
бит 0 – на сброс процессора в начальное состояние.

Активное состояние линий – нулевое, поэтому посылать байты с нулевым состоянием бита 0 нельзя (процессор будет заблокирован). По той же причине для открывания линии A20 следует посылать в порт 60h байт FFh, а для запрета доступа к HMA – байт FDh.

Примечание 2: при исполнении операции EDh порт 60h примет 1 байт данных и направит его на управление светодиодными индикаторами клавиатуры, в частности:

бит 2 – к индикатору Caps Lock
бит 1 – к индикатору Num Lock
бит 0 – к индикатору Scroll Lock

Для включения (зажигания) индикатора соответствующий бит надо установить в единицу. Не упомянутые здесь биты в посылаемом байте данных должны быть обнулены.

Примечание 3: при исполнении операций F0h – FFh контроллер направит в свою выходную шину 4 младших бита кода операции, но, в отличие от операции D1h, не будет удерживать эти состояния линий, а выдаст их только в виде 6-микросекундного импульса. В частности, при посылке в порт 64h кода операции FEh будет сформирован импульс, направляемый в линию 0 выходной шины. Он сбросит процессор в начальное состояние подобно кнопке "Reset" на лицевой панели системного блока компьютера. Пути влияния на развитие событий после сброса кратко описаны в примечании 4 к А.12-1.

А.11-4. Флаги центрального процессора.

Обычный 16-разрядный регистр флагов в современных процессорах, начиная с модели 80386, расширен до 32 бит; кроме того, введены управляющие регистры CR0, CR2 и CR3. Позже, начиная с процессора Pentium, группа управляющих регистров дополнена регистром CR4. Из упомянутых управляющих регистров только регистр CR2 не содержит флагов – он выделен для хранения линейного адреса последней команды, запросившей доступ к неразрешенной странице памяти. Назначение некоторых флагов из регистра флагов, а также из регистров CR0, CR3 и CR4 показано в приведенной ниже таблице.

Регистр	Бит	Назначение	Примечания
FLAGS	00h	CF – флаг переноса в старшем разряде	*1
FLAGS	02h	PF – четность суммы младшего байта	*1
FLAGS	04h	AF – флаг переноса из 4-го разряда	*1
FLAGS	06h	ZF – флаг равенства или нуля	*1

Продолжение таблицы А.11-4

FLAGS	07h	SF – флаг знака числа	*1
FLAGS	08h	TF – флаг пошагового исполнения	
FLAGS	09h	IF – флаг разрешения прерываний	*1
FLAGS	0Ah	DF – направление счета индексов	*1
FLAGS	0Bh	OF – флаг переполнения	*1
FLAGS	0Ch	2-битовое поле уровня привилегий	*2
FLAGS	0Eh	Флаг вложенной задачи	*3
FLAGS	0Fh	Признак процессоров 8086	*3
EFLAGS	10h	Флаг запрета отладочных прерываний	*4
EFLAGS	11h	Флаг установления режима V86	*4, *5
EFLAGS	12h	Флаг контроля выравнивания	*4, *6
EFLAGS	13h	Виртуальное разрешение прерываний	*4
EFLAGS	14h	Виртуальный запрос прерывания	*4
EFLAGS	15h	Идентификация процессора	*3, *4
CR0	00h	Флаг защищенного режима	*7
CR0	01h	Синхронизация сопроцессора (7.02-05)	*7
CR0	02h	Эмуляция сопроцессора через INT 07	*7
CR0	03h	Флаг переключения задач	*7
CR0	04h	Флаг поддержки команд сопроцессора	*7
CR0	05h	Исключение при ошибке сопроцессора	*7
CR0	10h	Защита от записи	*7, *8
CR0	12h	Разрешение выравнивания по маске	*7, *6
CR0	1Dh	Запрет сквозной записи кэша	*7
CR0	1Eh	Запрет заполнения кэша	*7
CR0	1Fh	Разрешение подкачки страниц	*7
CR3	03h	Сквозная запись при кэшировании	*7, *9
CR3	04h	Запрет страничного кэширования	*7, *9
CR4	00h	Разрешение виртуального флага V86	*7
CR4	01h	Разрешение виртуальных флагов	*7
CR4	02h	Запрет считывания счетчика времени	*7
CR4	03h	Разрешение INT 01 при вызове порта	*7

Примечание 1: роль и состояния этого флага описаны в разделе 6.05-15.

Примечание 2: биты 0Ch и 0Dh в регистре флагов выражают необходимый уровень привилегий для проведения операций ввода-вывода. В среде DOS биты 0Ch и 0Dh по умолчанию установлены: значит, прямой ввод-вывод разрешен всем процессам. Но только процессам с высшим (нулевым) уровнем привилегий дано право изменять состояния битов 0Ch и 0Dh командой POPF (7.03-68). По этому признаку можно проверить, исполняется ли текущий процесс на высшем уровне привилегий.

Примечание 3: регистр флагов позволяет грубо идентифицировать тип процессора. Невозможность сбросить флаг 0Fh в нуль – признак древних процессоров 8086. Невозможность установить в единицу бит 0Eh регистра флагов – признак 16-разрядных процессоров. Если процессор допускает запись в бит 0Eh, то он 32-разрядный и имеет регистр EFLAGS, а в нем бит 15h подскажет, способен ли данный процессор выдать таблицу своих параметров в ответ на команду CPUID (ее код 0F A2).

Примечание 4: EFLAGS – это старшие биты 32-разрядного регистра флагов. В реальном режиме доступ к ним обеспечивают команды PUSHF и POPF с префиксом 66h, как показано в разделе 7.02-06.

Примечание 5: установление режима V86 командой POPF невозможно (блокировано). Установить режим V86 из стека можно только в защищенном режиме командой IRET, когда битом 6 в байте 06h сегментного дескриптора (А.12-2) включена 32-разрядная адресация.

Примечание 6: здесь имеется ввиду выравнивание размещения операндов по адресам, кратным числу байтов в формате этих операндов. Контроль выравнивания производится только на третьем уровне привилегий при работе в защищенном режиме (подробнее – в примечании к 8.01-42). Бит 12h в регистре CR0 разрешает генерацию исключения, даже если она не разрешена битом 12h в регистре EFLAGS.

Примечание 7: управляющие регистры доступны посредством команды MOV (примечание 1 к 7.03-58). Кроме того, возможно считывание состояния регистра CR0 прерыванием INT 67\AX=DE07h (8.03-72).

Примечание 8: бит 10h служит для защиты сегментов пользовательских программ от записи со стороны операционной системы, действующей на более высоком уровне привилегий.

Примечание 9: 20 старших бит регистра CR3 хранят базовый адрес таблицы каталога страниц. Этот базовый адрес должен быть кратен размеру страницы (обычно 4 кб). Запись в регистр CR3 вызывает обновление данных в буфере TLB процессора, необходимое после внесения каждого изменения в таблицы страничной переадресации.

А.11-5 Отладочные регистры центрального процессора.

Современные процессоры, начиная с модели 80386, содержат отладочные регистры DR0 – DR7. Эти регистры позволяют аппаратно вызывать прерывание INT 01 (8.01-02) при обращениях к портам или к заранее указанным адресам памяти, в том числе при обращениях к данным и к незаписываемым носителям, на которых расставить контрольные точки нельзя. Для доступа к отладочным

регистрам можно использовать INT 67\AX=DE08h-DE09h (8.03-73), а также команду MOV (примечание 1 к 7.03-58).

Регистры DR0 – DR3 служат для записи 32-разрядных абсолютных линейных адресов назначаемых точек прерывания. Регистр DR7 определяет условия вызова прерывания INT 01. Регистр DR6 выделен для фиксации обстоятельств, при которых процессор вызвал прерывание исполнения (программные и внешние прерывания в регистре DR6 не отмечаются). Назначение отдельных битов в регистрах DR6 и DR7 показано в приведенной ниже таблице.

Регистр	Бит	Назначение	Примечания
DR6	00h	Доступ к адресу, записанному в DR0	*1
DR6	01h	Доступ к адресу, записанному в DR1	*1
DR6	02h	Доступ к адресу, записанному в DR2	*1
DR6	03h	Доступ к адресу, записанному в DR3	*1
DR6	0Dh	Зарегистрирована попытка доступа	*1
DR6	0Eh	Доступ при пошаговом исполнении	*1
DR6	0Fh	Доступ при переключении задач	*1
DR7	00h	2-битовое поле разрешения для DR0	*2
DR7	02h	2-битовое поле разрешения для DR1	*2
DR7	04h	2-битовое поле разрешения для DR2	*2
DR7	06h	2-битовое поле разрешения для DR3	*2
DR7	0Dh	Бит запрета генерации исключения	*3
DR7	10h	4-битовое поле управления для DR0	*4
DR7	14h	4-битовое поле управления для DR1	*4
DR7	18h	4-битовое поле управления для DR2	*4
DR7	1Ch	4-битовое поле управления для DR3	*4

Примечание 1: биты 1 – 3 регистра DR6 фиксируют акты обращения по адресам, заданным регистрами DR0 – DR3, даже когда генерация исключения INT 01 не разрешена битом 0Dh в регистре DR7, а также битом 10h в регистре EFLAGS (A.11-4). Бит 0Eh в регистре DR6 фиксирует состояние флага пошагового исполнения в момент обращения, а бит 0Fh – состояние флага переключения задач. Установленное состояние бита 0Dh в регистре DR6 напоминает о том, что процессор еще не генерировал исключение INT 01, хотя акт обращения уже зафиксирован (генерация исключения сбрасывает бит 0Dh в нуль).

Примечание 2: оба бита в этом 2-битовом поле разрешают регистрацию попыток доступа, но первый из них действует локально и сбрасывается при каждом переключении задачи. А второй бит действует глобально и позволяет регистрировать события, даже если они не связаны с исполнением текущей задачи.

Примечание 3: бит 0Dh в регистре DR7 запрещает не регистрацию попыток доступа, а генерацию исключения INT 01 при регистрации попыток доступа. Изменить состояние бита 0Dh в регистре DR7 можно только на высшем (нулевом) уровне привилегий или при работе в реальном режиме.

Примечание 4: первая пара битов в каждом поле управления определяет цель тех попыток доступа, которые следует регистрировать:

00 – обращение к команде для ее исполнения
01 – только запись данных в память
10 – обращение к порту (для процессоров Pentium+)
11 – обращения для записи или чтения данных.

Вторая пара битов в поле управления определяет размер зоны адресов (байт, слово или двойное слово), в пределах которой обращение к любому байту будет считаться регистрируемым событием.

A.12 Управление выделением и обслуживанием памяти

A.12-1 Карта распределения областей памяти

Приводимая ниже таблица показывает общие особенности функционального распределения областей памяти до 1 Мбайт, типичные для AT-совместимых компьютеров при их работе под управлением DOS. В каждом конкретном компьютере распределение областей памяти может несколько отличаться от показанного здесь, так как оно зависит и от версии BIOS, и от настроек BIOS Setup, и от конфигурации загрузки компьютера.

Адрес	Длина	Содержание
0000:0000	400h	Таблица прерываний реального режима
0000:0074	4	Адрес таблицы состояний видеорегистров
0000:0078	4	Адрес параметров флоппи-дисковода (A.08-2)
0000:007C	4	Адрес знаков 80-FFh графического шрифта 8x8
0000:0104	4	Адрес таблицы 1-го жесткого диска (A.13-1)
0000:010C	4	Адрес графического шрифта (8.01-30)
0000:0118	4	Адрес таблицы 2-го жесткого диска (A.13-1)
0040:0000	100h	Область данных BIOS (A.01-1)
0050:0000	1	Статус принтера для INT 05 (8.01-06)
0050:0004	1	Выбор флоппи-дисковода (A: или B:)
0050:0040	BCh	Адреса обработчиков прерываний (примечание 6)

Продолжение таблицы А.12-1

0000:7C00	200h	Область исполнения загрузочных записей
9000:FFFF	–	Граница "обыкновенной" памяти (примечание 1)
A000:0000	10000h	Окно доступа к видеопамяти (примечание 2)
B000:0000	10000h	Окно доступа к видеопамяти (примечание 2)
B800:0000	8000h	Видеобуфер для текстовых видеорежимов EGA+
C000:0000	8000h	Область видео-BIOS видеокарт (примечание 3)
C000:0070	7	Сигнатура "EXTMODE" (= поддержка SVGA)
C800:0000	4000h	Область BIOS жестких магнитных дисков
D000:0000	10000h	Область блоков UMB (5.04-04, 5.04-02)
E000:0000	10000h	Кадр страниц доступа к расширенной памяти
F000:0000	FFFFh	Область кодов BIOS компьютера (примечание 3)
F000:FFF0	–	Точка входа в программу загрузки (примечание 4)
F000:FFF5	8	Дата разработки системы BIOS
F000:FFFD	1	Контрольная сумма кода системы BIOS
F000:FFFE	1	Код модели компьютера
FFFF:0010	FFEFh	Область верхней памяти (примечание 5)

Примечание 1: граница 640 кбайт для "обыкновенной" памяти задается аппаратно микросхемой контроллера динамической памяти в составе комплекта микросхем (чипсета) материнской платы. За этой границей 384 кбайт адресного пространства резервированы для видеопамяти и для микросхем постоянной памяти с "прошивками" кодов BIOS. Свободные участки адресного пространства в резервированной области обычно бывают доступны в защищенном режиме благодаря механизму страничной переадресации в центральном процессоре.

Примечание 2: область адресов A000:0000h – B000:FFFFh предоставлена видеопамяти и может быть использована по-разному в зависимости от видеорежима (А.10-1). SVGA BIOS современных видеокарт организует в этой области одно или два "скользящих" окна доступа к обширной памяти видеокарты (подробнее в 8.01-39).

Примечание 3: через одни и те же участки адресного пространства может осуществляться доступ как непосредственно к кодам BIOS и видео-BIOS в микросхемах постоянной памяти, так и к их копиям в более быстродействующей оперативной памяти. Выбор того или другого варианта определяется установками параметров "shadowing" для соответствующих участков адресного пространства в программе BIOS Setup.

Примечание 4: адрес F000:FFF0h точки входа аппаратно обусловлен тем, что процессоры при включении электропитания устанавливают свои адресные шины в начальное состояние FFFF0h. Дальнейший ход

процесса загрузки зависит от значения байта 0Fh в CMOS RAM (примечание 1 к А.14-1) системы BIOS:

00h – обычная загрузка с тестом POST;
04h – загрузка посредством вызова INT 19 (8.01-90)
05h – сброс и переход по адресу в ячейке [0040:0067h] (А.01-1)
0Ah – переход по адресу в ячейке [0040:0067h] (А.01-1)

В отличие от первичной загрузки, при перезагрузке режим теста POST еще зависит от слова по адресу 0040:0072h (примечание 1 к А.01-1). Варианты 05h и 0Ah имеют смысл только при перезагрузке и различаются, в частности, установлением в начальное состояние контроллера прерываний.

Примечание 5: обращение к верхней памяти происходит тогда, когда при суммировании сегментного адреса со смещением образуется бит переноса, направляемый в линию А20 шины адреса. Область верхней памяти доступна в реальном режиме, но для пользования ею необходимо установить драйвер HIMEM.SYS (5.04-01), осуществляющий оперативное управление коммутацией линии А20.

Примечание 6: в области 0050:0040 – 0050:00FB MS-DOS7 сохраняет копии избранных адресов обработчиков прерываний (INT 00 – INT 1F, INT 40 – INT 43, INT 46, INT 70 – INT 77), подготовленных системой BIOS для последующей загрузки операционной системы. В основной таблице прерываний эти адреса могут быть замещены адресами других обработчиков, устанавливаемых позже драйверами, резидентными программами или самой MS-DOS7.

А.12-2 Дескрипторы сегментного доступа.

Сегментная адресация в защищенном режиме осуществляется посредством дескрипторов, в которых содержатся сведения об адресах сегментов, об их размерах и о правах доступа к ним. К моменту вызова процедур перехода в защищенный режим "глобальная" таблица дескрипторов сегментного доступа (GDT) должна быть составлена хотя бы частично. Готовить ее нужно заранее, пока процессор еще работает в реальном режиме. Состав помещаемых в таблицу дескрипторов, их количество и порядок их расположения определяются требованиями той процедуры, которая будет использовать таблицу.

Примеры таблиц GDT для разных процедур приведены в разделах 8.01-76, 8.01-78 и 9.10-01. Общим для всех таблиц GDT является то, что первый дескриптор в них должен быть заполнен нулями: он служит принимаемым по умолчанию шаблоном для дескрипторов не запрошенных сегментов и страниц памяти. Все дескрипторы сегментного доступа имеют одинаковую структуру, показанную в приведенной ниже таблице.

Смещение	Длина	Содержание
00h	2	2 байта от размера сегмента, начиная с младшего
02h	3	3 байта от адреса сегмента, начиная с младшего
05h	1	Байт прав доступа (примечание 2): бит 0: = 0 – обращений к сегменту еще не было = 1 – доступ к сегменту произошел бит 1: = 0 – чтение данных или исполнение кода = 1 – запись данных или чтение кода бит 2:– направление расширения (примечание 3) бит 3: = 0 – сегмент содержит данные = 1 – сегмент содержит исполняемый код бит 4: = 0 – маркер системных дескрипторов = 1 – маркер дескрипторов программ биты = 00 – высший уровень привилегий 5-6 = 11 – низший уровень привилегий бит 7: = 0 – сегмент надо подкачивать с диска = 1 – сегмент – в оперативной памяти
06h	1	Биты 0 – 3: старшие 4 бита от размера сегмента бит 4: – свободный бит (примечание 4) бит 5: = 0 (резервирован) бит 6: – бит разрядности (примечание 5): = 0 – 16-битовые адресация и операнды = 1 – 32-битовые адресация и операнды бит 7: – бит гранулярности: = 0 – размер сегмента в байтах = 1 – размер в единицах по 4К байт
07h	1	Старший байт адреса сегмента

Примечание 1: байты 06h и 07h в составе дескриптора принимаются во внимание процессорами не древнее модели 80386. Байты 06h и 07h следует обнулять, если программа должна сохранять работоспособность на процессорах 80286. Кроме того, нулевые значения байтов 06h и 07h являются отличительным признаком 16-разрядных программ защищенного режима, обеспечивающим их правильное исполнение на 32-разрядных процессорах.

Примечание 2: интерпретация бит 0 – 3 в байте прав доступа зависит от бита 4. Здесь показана интерпретация этих битов для дескрипторов, относящихся к пользовательским программам, включая сегменты данных и сегменты исполняемого кода. В системных дескрипторах биты 0 – 3 выражают 16 видов функций данного дескриптора.

Примечание 3: интерпретация бита 2 в байте прав доступа зависит от бита 3. Для сегментов кода нулевое значение бита 2 означает, что исполнять этот код разрешено только программам, которые имеют тот же уровень привилегий (иначе право на исполнение кода имеют также программы с более высоким уровнем привилегий). Для сегментов данных нулевое значение бита 2 означает обычное направление расширения вверх, а единичное значение – направление расширения вниз (используемое, в частности, в сегменте стека).

Примечание 4: бит 4 в байте 06h предоставлен в распоряжение программистов. В дескрипторах страниц его используют как метку запрета переопределения, например, при отображении в память адресного пространства ввода-вывода.

Примечание 5: как разрядность адресации, так и разрядность операндов определяются состоянием бита разрядности 6 в байте 06h дескриптора сегмента кода. В системных сегментах биты 4 – 6 байта 06h должны быть обнулены.

А.12-3 Избранные функции XMS-драйвера HIMEM.SYS

Перед тем, как обращаться к функциям драйвера HIMEM.SYS (5.04-01), необходимо выполнить две операции. Во-первых, нужно с помощью INT 2F\AX=4300h (8.03-22) убедиться, что драйвер HIMEM.SYS установлен. Во-вторых, нужно с помощью INT 2F\AX=4310h (8.03-23) найти указатель на точку входа в исполнительную программу XMS-драйвера. Возвращаемый указатель будет служить адресом дальнего перехода (CALL FAR, 7.03-08), посредством которого осуществляется вызов функций XMS-драйвера. Запрашиваемые функции определяются значениями в регистре AH в момент вызова, эти значения показаны в первой колонке приведенной ниже таблицы. Для многих функций дополнительные данные передаются через регистр DX; его роль показана во второй колонке таблицы. Возвращаемое содержимое регистра AX показано в четвертой колонке. Большая часть функций XMS-драйвера возвращает в регистре AX слово статуса, причем статус AX=0001h означает успешное завершение, а статус AX=0000h означает ошибку. В случае ошибки почти все функции (кроме AH=00h) возвращают в регистре BL код ошибки, показанный в таблице А.06-1, а две функции – AH=08h и AH=88h – возвращают код ошибки в регистре BL всегда.

AH	DX при вызове	Функция	AX при возврате	Приме-чания
00h		Сообщить версию XMS	Версия	*1
05h		Включить адресную шину A20	Статус	
06h		Выключить адресную шину A20	Статус	
08h		Наибольший свободный блок	Размер	*2

Продолжение таблицы А.12-3

09h	Размер	Выделить блок XMS-памяти	Статус	*3
0Ah	Ссылка	Освободить блок XMS-памяти	Статус	
0Bh		Копирование в XMS-памяти	Статус	А.12-4
0Ch	Ссылка	Закрыть блок XMS-памяти	Статус	*4
0Dh	Ссылка	Раскрыть блок XMS-памяти	Статус	
0Eh	Ссылка	Получить сведения о ссылке	Статус	*5
0Fh	Ссылка	Изменить размер блока XMS	Статус	*6
10h	Размер	Выделить блок UMB	Статус	*7,*8
11h	Адрес	Высвободить блок UMB	Статус	*7
12h	Адрес	Изменить размер блока UMB	Статус	*7,*8
88h		Наибольший свободный блок	Размер	*2,*9
89h	Размер	Выделить блок XMS-памяти	Статус	*3,*9
8Eh	Ссылка	Получить сведения о ссылке	Статус	*5,*9
8Fh	Ссылка	Изменить размер блока XMS	Статус	*6,*9

Примечание 1: в регистре DX возвращается статус области HMA: DX=0001h означает, что область HMA используется, а DX=0000h – что область HMA не задействована.

Примечание 2: функция 08h при вызове требует BL=00h, размер свободной XMS-памяти (в килобайтах) она возвращает в регистре DX, а в регистре AX возвращает размер наибольшего свободного блока. Функция 88h делает то же самое, но возвращает результаты в 32-разрядных регистрах EDX и EAX. Помимо того, функция 88h возвращает в регистре ECX наибольший физический адрес, соответствующий доступному байту XMS-памяти.

Примечание 3: функции 09h и 89h принимают запрашиваемый размер XMS блока в килобайтах, но функция 09h принимает его из регистра DX, а функция 89h – из 32-битного регистра EDX. Обе эти функции возвращают номерную ссылку на выделенный блок в регистре DX.

Примечание 4: в случае успеха в регистрах DX:BX возвращается 32-битный физический адрес блока памяти, доступ к которому был закрыт.

Примечание 5: функция 0Eh возвращает в регистре BH содержимое счетчика актов закрытия данного блока, в регистре BL – число свободных номерных ссылок, в регистре DX – размер блока памяти (в килобайтах), доступ к которому обеспечивает запрошенная номерная ссылка. Функция 8Eh делает то же самое, но возвращает число свободных номерных ссылок в регистре CX, а размер блока памяти – в 32-разрядном регистре EDX.

Примечание 6: функция 0Fh, переопределяющая размер выделенного блока XMS-памяти, принимает запрашиваемое значение (в килобайтах) из регистра BX. Функция 8Fh делает то же самое, но принимает

новый размер блока из 32-разрядного регистра EBX. При этом переопределяемый блок XMS-памяти не должен быть закрыт.

Примечание 7: чтобы реализовать функции 10h – 12h путем табличной переадресации, их исполнение (начиная с процессора i80386) передано драйверу EMM386.EXE (5.04-02), который организует переадресацию и перехватывает адрес прямого вызова HIMEM.SYS. Исполнение функций 10h – 12h без перехода в защищенный режим обеспечивает драйвер UMBPCI.SYS (5.04-04).

Примечание 8: функции 10h и 12h "понимают" запрашиваемый размер UMB-блока в 16-байтовых единицах (параграфах). Функция 12h, изменяющая размер блока, принимает запрашиваемый размер из регистра BX. Функция 10h при успешном завершении возвращает сегментный адрес выделенного UMB-блока в регистре BX, а фактический размер выделенного UMB-блока – в регистре DX. В случае неудачи, отмечаемой значением AX = 0000h, функции 10h и 12h возвращают в регистре DX размер наибольшего имеющегося UMB-блока.

Примечание 9: в отличие от функций 0xh, функции 8xh требуют драйвера HIMEM.SYS версии не ниже 3.07 и не могут быть выполнены на устаревших 16-разрядных процессорах.

А.12-4 Блок данных запроса на копирование в XMS-памяти

Указатель на этот блок данных должен быть в регистрах DS:SI при вызове функции AH = 0Bh (А.12-3) XMS-драйвера HIMEM.SYS (5.04-01). Функция AH = 0Bh копирует группу байтов из одного блока XMS-памяти (блока-источника) в другой блок XMS-памяти – блок назначения.

Смещение	Длина	Содержание
00h	4	Число копируемых байтов (обязательно четное)
04h	2	Номерная ссылка на блок-источник
06h	4	Начальное смещение в блоке-источнике
0Ah	2	Номерная ссылка на блок назначения
0Ch	4	Начальное смещение в блоке назначения

Примечание 1: если блоки источника и назначения перекрываются, то правильно исполняется только копирование вперед, то есть когда базовый адрес источника меньше, чем базовый адрес назначения.

Примечание 2: если вместо любой из номерных ссылок указано значение 0000h, то соответствующее четырехбайтовое значение начального смещения интерпретируется как указатель (сегмент : смещение) на блок в обыкновенной памяти в пределах до 1 Мегабайта.

A.12-5 Дескриптор запроса на копирование в EMS-памяти

Этот дескриптор служит для спецификации блоков источника и назначения в операциях копирования и обмена INT 67\AX=5700h-5701h (8.03-69), выполняемых драйвером EMM386.EXE (5.04-02). Как блок источника, так и блок назначения могут принадлежать страницам EMS-памяти или находиться в обыкновенной памяти. В последнем случае расположение определяется не номером EMS-страницы, а сегментным адресом блока, причем вместо соответствующей номерной ссылки должно быть указано значение 0000h.

Смещение	Длина	Содержание
00h	4	Длина запрашиваемого блока в байтах
04h	1	= 00h: источник – в обыкновенной памяти = 01h: источник – на EMS-странице
05h	2	Номерная ссылка на источник или значение 0000h, если источник – в обыкновенной памяти
07h	2	Смещение начала блока-источника на EMS-странице или относительно сегментного адреса
09h	2	Номер логической EMS-страницы источника или его сегментный адрес в обыкновенной памяти
0Bh	1	= 00h: блок назначения – в обыкновенной памяти = 01h: блок назначения – на EMS-странице
0Ch	2	Номерная ссылка на блок назначения или значение 0000h, если этот блок – в обыкновенной памяти
0Eh	2	Смещение начала блока назначения на EMS-странице или относительно сегментного адреса
10h	2	Номер EMS-страницы блока назначения или его сегментный адрес в обыкновенной памяти

Примечание 1: допускается взаимное перекрытие блоков источника и назначения при операции копирования, но в таком случае только одно направление копирования позволяет избежать потери данных.

A.12-6. Блок данных для запроса на переход в EMS-памяти.

Показанный ниже блок данных служит для спецификации параметров вызова подпрограммы в операции INT 67\AH=56h (8.03-68), выполняемой драйвером EMM386.EXE (5.04-02). В операции дальнего перехода INT 67\AH=55h (8.03-68) используется только часть показанного ниже блока до смещения 09h.

Смещение	Длина	Содержание
00h	4	Целевой адрес перехода (сегмент : смещение)
04h	1	Длина новой таблицы соответствия страниц
05h	4	Адрес новой таблицы соответствия страниц
09h	1	Длина заменяемой таблицы соответствия страниц
0Ah	4	Адрес заменяемой таблицы соответствия страниц
–	8	(резервировано для драйвера Emm386.exe)

Примечание 1: структура таблиц соответствия логических и физических страниц показана в примечании 3 к INT 67\AH=5000h (8.03-66). Сведения о действующей (заменяемой) таблице соответствия необходимы для возврата к исполнению вызывающей программы после завершения исполнения вызванной подпрограммы.

А.12-7 Дескрипторы блоков памяти.

Распределением памяти в компьютере "заведует" операционная система. Перед каждым блоком памяти, который выделяет DOS, она формирует 16-байтовый дескриптор. В зарубежной литературе его принято называть MCB (= Memory Control Block). Найти MCB просто: его сегментный адрес всегда на единицу меньше сегментного адреса того блока памяти, к которому этот MCB относится.

По дескрипторам блоков памяти DOS прослеживает всю доступную ей память (подробнее об этом – в примечании 3). Отдельным блоком памяти DOS считает свободное пространство за пределами занятых областей: ему также должен предшествовать дескриптор, отличительным признаком которого является код 0000h вместо адреса программы-владельца. Все сведения об имеющейся свободной памяти и о конкретном расположении свободных участков DOS получает из прослеживаемой последовательности дескрипторов блоков памяти.

Структура дескриптора блока памяти показана в приведенной ниже таблице.

Смещение	Длина	Содержание	Примечания
00h	1	4Dh (= M) – не-последний MCB 5Ah (= Z) – последний MCB в цепи	*1
01h	2	Сегмент программы-владельца	*2
03h	2	Размер блока памяти в параграфах	*3
05h	3	Не используются	
08h	8	Имя файла программы	*4

Примечание 1: свой основной блок системных данных DOS разделяет на субблоки, имеющие такие же дескрипторы, но с другими идентификаторами в байте со смещением 00h:

42h (= B) – субблок буферов (4.03)
44h (= D) – субблок драйвера
45h (= E) – субблок данных для драйверов
46h (= F) – субблок SFT (4.12)
49h (= I) – субблок IFS
4Ch (= L) – субблок CDS (4.17)
53h (= S) – субблок стеков (4.27)
54h (= T) – субблок переходного кода
58h (= X) – субблок FCBS (4.10).

Примечание 2: если данный блок памяти свободен, то в слово дескриптора со смещением 01h записываются нули, а если данный блок DOS выделила самой себе, то туда вместо сегментного адреса программы заносится код 0008h.

Примечание 3: размер выделяемого блока памяти здесь указывается в 16-байтовых параграфах. Сегментный адрес дескриптора следующего блока памяти на единицу больше суммы сегментного адреса данного дескриптора с числом, указанным в слове со смещением 03h. На основании этого соотношения DOS выполняет прослеживание всей цепи дескрипторов, начиная с дескриптора первого выделенного блока памяти. Сегментный адрес этого первого дескриптора указан в слове, непосредственно предшествующем "Списку Списков": оно отмечено в таблице А.01-2 как имеющее смещение –02h.

Примечание 4: имя программы заносится в дескрипторы блоков, выделенных для PSP программ, а также в дескрипторы драйверных субблоков и субблоков IFS. Начиная с байта 08h в дескрипторы некоторых блоков заносятся сигнатуры, которые означают следующее:

SC – блок принадлежит DOS и содержит код;
SD – блок принадлежит DOS и содержит данные;
SM – последний блок в области UMB;
UMB – первый блок в области UMB.

В дескрипторах других блоков байты 08h – 0Fh не используются и могут содержать "мусор".

Примечание 5: у программ с суффиксом *.COM нет заголовков, из которых DOS могла бы "узнать" размер необходимой им памяти, и DOS выделяет им все имеющееся пространство памяти. Когда свободной памяти совсем не осталось, любой запрос от резидентной программы или от обработчика, вызванного внешним прерыванием, может привести к "зависанию" компьютера. Во избежание этого программы с суффиксом *.COM должны сами вызывать INT 21\AH=4Ah

(8.02-52), чтобы DOS сформировала дескриптор, в котором неиспользуемая часть памяти была бы объявлена свободной. Примеры таких вызовов показаны в первых 6 строках ассемблерных текстов в разделах 9.06, 9.10-01 и 9.10-02.

А.13 Таблицы параметров жестких магнитных дисков

А.13-1 Параметры дисководов на жестких магнитных дисках

До 1996 года было принято адресовать операции с жесткими магнитными дисками посредством параметров CHS (Cylinder-Head-Sector). При такой адресации размер доступного дискового пространства ограничен величиной 528 Мбайт. Система BIOS хранила параметры CHS первого и второго дисководов на жестких магнитных дисках в блоках данных, адреса которых записаны в ячейки 0000:0104h и 0000:0118h соответственно. Обе эти ячейки находятся в пределах таблицы прерываний (А.12-1), и на них иногда ссылаются как на INT 41 и INT 46. Если в компьютере было больше двух дисководов на жестких магнитных дисках, то параметры CHS остальных дисководов можно было получить только посредством вызова INT 13\AH=08h (8.01-49).

К 1995 году объемы дисков уже достигли гигабайта, так что 528-мегабайтовый предел необходимо было преодолеть. Для новых программ были разработаны адресация LBA (примечание 4 к А.13-6) и новые функции прерывания INT 13 (8.01-55 – 8.01-60), а для сохранения совместимости со старыми программами обработчика прерывания INT 13\AH=08h (8.01-49) научили выдавать не реальные, а преобразованные параметры. Когда программы используют эти параметры при вызове старых функций прерывания INT 13 (8.01-46 – 8.01-54), тогда система BIOS, "подставившая" преобразованные параметры, сама выполняет обратное преобразование так, что дисковое пространство, доступное для CHS-адресации, расширяется до 8.4 Гбайт (подробнее – в примечании 2 к А.13-6).

Блоки данных, адреса которых записаны в ячейки 0000:0104h и 0000:0118h, могут содержать как реальные, так и преобразованные CHS-параметры, причем каждому варианту соответствует своя структура. Они обе показаны в приведенной ниже таблице. В первом столбце (Std) приведены смещения элементов данных в блоках с реальными CHS-параметрами. Во втором столбце (Trs) приведены смещения элементов данных в блоках, формируемых согласно уточненной спецификации, предложенной в 1995 году фирмой Phoenix для дисководов с числом цилиндров более 1024. Разрабатываемым в настоящее время программам не рекомендуется пользоваться сведениями из всех этих блоков данных.

Std	Trs	Длина	Содержание
00h	00h	2	Число цилиндров дисковода (примечание 1)
02h	02h	1	Число головок дисковода (примечание 1)
	03h	1	Сигнатура A0h – отличительный признак блоков, отвечающих спецификации фирмы Phoenix
	04h	1	Число секторов на дорожке (примечание 2)
05h	05h	2	Начало предкомпенсации записи (примечание 3)
08h	08h	1	Бит 2 – рекалибровка головок не производится бит 3 – число головок превышает 8 бит 5 – имеется карта дефектов (примечание 4) бит 6 – исключить повторы при сбоях ECC бит 7 – исключить попытки повторного доступа
	09h	2	Число дорожек (до 65536, примечание 2)
	0Bh	1	Число головок (до 16, примечание 2)
0Ch	0Ch	2	Цилиндр парковки головок (примечание 3)
0Eh	0Eh	1	Число секторов на дорожке (примечание 1)
	0Fh	1	Контрольная сумма

Примечание 1: в этих позициях даны CHS-параметры, предназначенные для использования при вызове старых функций прерывания INT 13 (8.01-46 – 8.01-54). Тем не менее некоторые версии BIOS могут указывать реальное число цилиндров свыше 1024, недопустимое при вызове старых функций прерывания INT 13.

Примечание 2: эти позиции заполнены только в преобразованном блоке данных. Записанные в них параметры отражают истинные характеристики дисковода и предназначены для использования программами, которые обращаются непосредственно к портам контроллера дисковода (а не через прерывание INT 13).

Примечание 3: современные дисководы выполняют предкомпенсацию и парковку самостоятельно. Попытки влиять на эти процессы игнорируются.

Примечание 4: карта дефектов поверхности дисков обычно записана на дорожке, номер которой на единицу превышает число цилиндров дисковода, указываемое в данной таблице в ячейке со смещением 00h.

А.13-2 Формат расширенной таблицы параметров

Представленная здесь таблица параметров запрошенного дисковода записывается в заранее подготовленный буфер обработчиком прерывания INT 13\AH=48h (8.01-60).

Смещение	Длина	Содержание
00h	2	При вызове: размер буфера (8.01-60) При возврате: длина заполненной части буфера
02h	2	Бит 0: ошибки DMA исправлять без остановок бит 1: сообщаемые данные CHS действительны бит 2: дисковод на сменных дисках, биты 4 – 6 действительны бит 3: поддержка записи с верификацией бит 4: дисковод регистрирует смену дисков бит 5: поддержка блокировки и разблокировки бит 6: диск не вставлен, CHS взяты из таблиц
04h	4	Число цилиндров в дисководе (примечание 1)
08h	4	Число головок дисковода (примечание 1)
0Ch	4	Число секторов на дорожке (счет с единицы)
10h	8	Число секторов в дисководе (оно на единицу больше, чем номер последнего сектора)
18h	2	Размер сектора в байтах
1Ah	4	Указатель на таблицу DPTE (примечание 2)
1Eh	2	= BEDDh: сигнатура наличия данных о пути
20h	1	= 2Ch: размер данных о пути, включая сигнатуру
24h	4	Системная шина (ISA или PCI) + пробел (20h)
28h	8	Тип интерфейса (примечание 3)
30h	8	Поле пути по системной шине (примечание 4)
38h	16	Поле пути к дисководу (примечание 5)
49h	1	Контрольная сумма байтов 1Eh – 48h

Примечание 1: номера дорожек, цилиндров и головок считаются от нуля, поэтому последний действительный номер на единицу меньше указанного здесь числа. Приведенные здесь параметры – реальные. При вызове старых функций прерывания INT 13 (8.01-46 – 8.01-54) следует пользоваться не реальными, а преобразованными параметрами, которые возвращает обработчик INT 13\AH=08h (8.01-49).

Примечание 2: расширения BIOS версии до 2.0 не предоставляют таблицы DPTE, а в поле указателя на нее записывают значение FFFF:FFFFh. Формат таблицы DPTE показан в приложении А.13-3. Таблица DPTE предоставляется во временном буфере, содержимое которого при последующих вызовах функций BIOS не сохраняется.

Примечание 3: в поле типа интерфейса допустимы только резервированные слова, в частности: 1394, ATA, ATAPI, SCSI, USB. Слова дополняются пробелами (знаками 20h) до полной длины поля 8 байтов.

Примечание 4: для шины ISA поле пути по системной шине содержит двухбайтовый базовый адрес порта, а байты 32h – 37h обнулены. Для шины PCI байт 30h представляет номер шины, байт 31h – номер разъема (слота), байт 32h – номер функции, байт 33h – номер контроллера; байты 34h – 37h обнулены.

Примечание 5: для интерфейса 1394 (Firewire) начиная с байта 38h указан 8-байтовый идентификатор (EUI-64). Для интерфейса ATA (IDE) значением 00h байта 38h маркируется ведущее устройство (master), а значением 01h – ведомое устройство (slave). Для интерфейса ATAPI байт 38h имеет такое же назначение, как и для интерфейса ATA, но байт 39h представляет логический номер устройства (LUN). Для интерфейса SCSI начиная с байта 38h записывается двухбайтовый идентификатор устройства (SCSI ID), а начиная с байта 3Ah – 8-байтовый логический номер устройства (LUN). Для интерфейса USB начиная с байта 38h записывается 8-байтовый серийный номер. Для всех интерфейсов предполагается, что не упомянутые здесь байты (в пределах 3Ah – 48h) обнулены.

А.13-3 Дополнительная таблица DPTE

Расширения BIOS версии 2.0 и выше в дополнение к таблице А.13-2 параметров запрошенного дисковода предоставляют дополнительную таблицу DPTE (Device Parameter Table Extension), структура которой показана ниже. Адрес таблицы DPTE возвращается в ячейке со смещением 1Ah в расширенной таблице (А.13-2) обработчиком прерывания INT 13\AH=48h (8.01-60). Этот адрес указывает на временный буфер, содержимое которого при последующих вызовах функций BIOS не сохраняется. Данные в таблице DPTE предназначены для тех программ, которые обращаются непосредственно к портам контроллера дисковода.

Смещение	Длина	Содержание
00h	2	Базовый адрес порта запрошенного устройства
02h	2	Адрес порта управляющих регистров дисковода
04h	1	Флаги: биты 0-3 = 0b, биты 5 и 7 = 1b бит 4: = 0b – ведущий диск, = 1b – ведомый бит 6: = 1b, если разрешена адресация LBA
06h	1	Биты 0-3: номер IRQ, биты 4-7 сброшены
07h	1	Число секторов в многосекторных пересылках
08h	1	Биты 0-3: канал DMA, биты 4-7: тип DMA
09h	1	Биты 0-3: тип PIO, если в слове 0Ah бит 0 = 1b
0Ah	2	Бит 0: PIO исполняется, байт 09h действителен бит 1: пересылка посредством DMA разрешена

Продолжение таблицы А.13-3

		бит 2: многосекторные пересылки разрешены бит 3: преобразование CHS осуществляется бит 4: преобразование для LBA выполняется бит 5: накопитель использует сменные носители бит 6: интерфейс ATAPI (обычно у CD-ROM) бит 7: 32-битовая пересылка разрешена бит 8: сигнал готовности интерфейса ATAPI биты 9-10: тип преобразования CHS: = 00 – посредством битовых сдвигов = 01 – посредством линейной адресации = 10, 11 – другие типы преобразования бит 11: пересылка "ultra DMA" разрешена
0Eh	1	Номер версии расширений INT 13
0Fh	1	Контрольная сумма байтов 00h – 0Eh

А.13-4 Формат пакета дисковой адресации

Этот формат пакета данных используется расширенными функциями считывания INT 13\AH=42h (8.01-56) и записи INT 13\AH=43h (8.01-57). Перед пользованием этими функциями необходимо с помощью INT 13\AH=41h (8.01-55) убедиться в том, что система BIOS их поддерживает.

Смещение	Длина	Содержание
00h	1	Длина пакета дисковой адресации (примечание 1)
02h	1	Число (до 7Fh) блоков данных (примечание 2)
04h	4	Указатель на буфер для данных (примечание 3)
08h	8	LBA-адрес 1-го блока данных (примечание 4)
10h	8	Указатель на буфер для данных (примечание 3)
18h	8	Число блоков данных (примечание 2)

Примечание 1: длина пакета равна 20h, если расширенный формат пакета поддерживается, или 10h, если расширенный формат пакета не поддерживается. О поддержке расширенного формата можно узнать по биту 3 в регистре CX после вызова INT 13\AH=41h (8.01-55).

Примечание 2: если поддерживается расширенный пакет дисковой адресации, и если байт 02h имеет значение FFh, то число подлежащих передаче блоков данных будет считано начиная со смещения 18h. При возврате число подлежащих передаче блоков данных заменяется числом фактически переданных блоков данных.

Примечание 3: если поддерживается расширенный пакет дисковой адресации, и если число в ячейке 04h имеет значение FFFF:FFFFh, то указатель на буфер будет считан начиная со смещения 10h.

Примечание 4: для дисководов, не поддерживающих адресацию LBA (примечание 4 к А.13-6), номер начального блока рассчитывается по формуле

(C•N + H)•T + S - 1

где C – заданный цилиндр, N – число головок (на 1 больше максимального номера головки), H – номер заданной головки, T – число секторов на дорожке, S – номер заданного сектора.

А.13-5 Дескрипторы разделов жестких магнитных дисков

В секторе 01h дорожки 00h на стороне 00h загрузочного физического диска находится главная загрузочная запись MBR (Master Boot Record), содержащая до 436 байт исполняемого машинного кода, 4-байтовый идентификационный код(1B8h – 1BBh) и таблицу разделов диска. Идентификационный код записывается только операционными системами Windows-NT/2000/XP и может отсутствовать. Помимо MBR, в том же секторе находится таблица разделов диска. Чтобы посмотреть сектор MBR, его надо скопировать в файл, как показано в разделе 9.02-02. Раскрыть не-текстовый файл можно в просмотрщике файлового менеджера Volcov Commander (6.25) или в отладчике DEBUG.EXE (6.05). На рис. 12 показаны фрагменты сектора MBR, скопированного с реального диска.

```
C:\DOS\SRV\BIOS>debug mbr.dat
-d 100 L20
195B:0100  EB 02 7C 01 FA 33 C0 8E-D0 8E C0 8E D8 BC 00 7C   ..|..3.........|
195B:0110  8B F4 FB BF 00 06 B9 00-01 F3 A5 BB 20 06 FF E3   ..........  ...
-d 260 L20
195B:0260  FF 01 B9 00 20 E2 FE 59-C3 0D 0A 45 72 72 6F 72   .... ..Y...Error
195B:0270  20 4C 6F 61 64 69 6E 67-20 4F 53 00 AA 55 01 00    Loading OS..U..
-d 2BE L42
195B:02B0                                           00 00                ..
195B:02C0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 80 01   ................
195B:02D0  01 02 06 3F 3F FF BF 1F-00 00 41 A0 0F 00 00 00   ...??.....A.....
195B:02E0  41 00 05 3F BF D2 00 C0-0F 00 40 BB 1C 00 00 00   A..?......@.....
195B:02F0  C1 B8 82 3F FF C7 00 92-3A 00 00 FC 00 00 55 AA   ...?....:.....U.
-
```

Рис. 12

Первый фрагмент представляет начальную часть блока исполняемого машинного кода (вариант MBR фирмы OnTrack), второй фрагмент – конечную часть того же блока с текстовым сообщением об ошибке, а третий фрагмент – таблицу разделов, отражающую разбивку на разделы жесткого магнитного диска в конкретном компьютере.

Таблица разделов состоит из четырех дескрипторов разделов, каждый длиной 16 байт. Если считать от начала сектора, то начальные смещения дескрипторов будут 1BEh, 1CEh, 1DEh и 1EEh соответственно. Но так как файл MBR.DAT на рис. 12 загружен со смещения 100h, начальные смещения дескрипторов там будут 2BEh, 2CEh, 2DEh, 2EEh. Последним словом в секторе 01h является сигнатура AA55h, отмечающая конец MBR действующего загрузочного диска.

Четыре дескриптора разделов обеспечивают возможность создания до четырех первичных разделов в каждом физическом дисководе на жестких магнитных дисках. Если фактическое число разделов меньше, то оставшиеся незадействованными дескрипторы заполняются нулями. На рис. 12 видно, что дескриптор первого раздела заполнен нулями и, следовательно, не задействован. Значит, данный жесткий диск разбит на три первичных раздела.

На загрузочном дисководе один из первичных разделов должен быть помечен как загрузочный меткой 80h в первом байте дескриптора. На рис. 12 видно, что метка 80h имеется в первом байте дескриптора второго раздела (в байте со смещением 02CEh). Значит, загрузочным является второй раздел диска.

Сведения о структуре данных, содержащихся в дескрипторах разделов, показаны в приведенной ниже таблице. Значения смещений в первом столбце таблицы отсчитываются от начала каждого дескриптора.

Смещение	Длина	Содержание
00h	1	Статус (= 80h для загрузочного раздела)
01h	1	Номер стороны (головки) начала раздела
02h	1	Начальный сектор раздела (примечание 1)
03h	1	Начальная дорожка раздела (примечание 1)
04h	1	Идентификатор файловой системы (А.13-6)
05h	1	Номер стороны (головки) конца раздела
06h	1	Конечный сектор раздела (примечание 2)
07h	1	Конечная дорожка раздела (примечание 2)
08h	4	Число секторов диска перед данным разделом
0Ch	4	Длина раздела, выраженная числом секторов

Примечание 1: в байте со смещением 02h номер начального сектора занимает только биты 0 – 5, а биты 6 и 7 представляют старшие разряды 10-битового номера начальной дорожки раздела, младшие 8 бит которого записаны в байт со смещением 03h.

Примечание 2: в байте со смещением 06h номер конечного сектора занимает только биты 0 – 5, а биты 6 и 7 представляют старшие разряды 10-битового номера конечной дорожки раздела, младшие 8 бит которого записаны в байт со смещением 07h.

Примечание 3: если дескриптор относится к разделу, в котором используется адресация LBA (примечание 4 к А.13-6), то данные о номерах головок, секторов и дорожек могут быть недействительны. Но адреса в байтах 08h – 0Fh действительны всегда.

А.13-6 Некоторые идентификаторы файловых систем

Байт со смещением 04h в каждом дескрипторе раздела (А.13-5) содержит идентификатор файловой системы, но основании которого каждая операционная система "решает", сможет ли она обеспечить доступ к данному разделу. Если идентификатор не входит в число "известных" операционной системе, то она не будет предпринимать попыток доступа и, скорее всего, даже не сообщит о наличии такого раздела пользователю. Имеются специальные идентификаторы для обозначения разделов, используемых исключительно в служебных целях и намеренно скрываемых от пользователя. В таблице А.13-6 показано соответствие между некоторыми распространенными файловыми системами и теми идентификаторами, которые их представляют.

ID	Значение
00h	Незанятое дисковое пространство
01h	Файловая система FAT-12 для разделов не свыше 16 Мбайт
04h	Устаревшая FAT-16 до 32 Мбайт без кластерной структуры
05h	Расширенный раздел с адресацией CHS (примечания 1 и 2)
06h	FAT-16 до 2 Гбайт с адресацией CHS (примечание 2)
07h	Файловая система NTFS (примечание 3)
0Bh	FAT-32 с адресацией CHS (примечание 2)
0Ch	FAT-32 с адресацией LBA (примечание 4)
0Eh	FAT-16 до 2 Гбайт с адресацией LBA (примечание 4)
0Fh	Расширенный раздел с адресацией LBA (примечания 1 и 4)
11h	Скрытая FAT-12 для размещения boot-менеджера OS/2
14h	Скрытая FAT-16 для размещения boot-менеджера OS/2
1Bh	Скрытая FAT-32 с адресацией CHS (примечание 2)
1Ch	Скрытая FAT-32 с адресацией LBA (примечание 4)
3Ch	Служебный раздел программы Partition Magic
42h	Динамический раздел операционной системы Windows Vista
43h	Менеджер загрузки (BootWizard) PTS-DOS
4Dh-4Fh	Разделы операционной системы QNX
54h	Служебный раздел DDO программы Disk Manager
64h-65h	Разделы операционной системы Novell Netware
82h	Раздел подкачки операционной системы Linux
83h	Файловая система ext2fs операционной системы Linux
84h	Раздел для восстановления состояния электропитания
85h	Расширенный раздел системы Linux (примечание 1)
A0h	Раздел сохранения состояния портативных компьютеров
A5h	Раздел операционной системы FreeBSD
A6h	Раздел операционной системы OpenBSD
A8h	Раздел UFS операционной системы MacOS

Продолжение таблицы А.13-6

A9h	Раздел операционной системы Net BSD
ABh	Загрузочный раздел операционной системы MacOS
BEh	Загрузочный раздел операционной системы Solaris
D8h, DBh	Разделы операционной системы CP/M
EBh	Файловая система BeOS (BFS1)
EEh	Раздел GPT 64-разрядных систем Windows (примечание 5)
FDh	Раздел RAID операционной системы Linux

Примечание 1: расширенный раздел – это формальная спецификация пространства для размещения нескольких не-первичных разделов (логических дисков). Параметры не-первичных разделов записываются не в MBR, а в выделенные сектора, прослеживаемые по цепи ссылок. MS-DOS не допускает замыкания этой цепи в кольцо, иначе при загрузке она входит в бесконечный цикл поиска конца у этого кольца.

Примечание 2: в дескрипторах разделов (А.13-5) каждому комплекту параметров CHS (Cylinder-Head-Sector) выделено 3 байта, так что они позволяют адресовать не более 2^24 секторов по 512 байт. Отсюда следует предел CHS-адресации в 2^23 кбайт, или 8 Гбайт. Поэтому разделы с адресацией CHS могут быть организованы только в пределах 8 Гбайт от начала дискового пространства (далее необходима адресация LBA). В таблице А.13-6 адресация CHS отмечена только у тех идентификаторов, которые выражают различия типов адресации.

Примечание 3: фирма Microsoft приписывает идентификатор 07h устанавливаемым файловым системам (IFS), то есть таким, которые не могут быть представлены пользователю без предварительного преобразования. Но фактически, помимо NTFS, идентификатором 07h бывает отмечена только редко используемая файловая система HPFS фирмы IBM.

Примечание 4: адресация LBA (Linear Block Addressing) основана на счете числа секторов от начала дискового пространства по данным из байтов 08h – 0Fh в дескрипторах разделов (А.13-5). Она позволяет преодолеть границу 8 Гбайт, присущую адресации CHS. Для осуществления адресации LBA необходимо, чтобы и дисковод, и система BIOS компьютера поддерживали расширенный набор функций прерывания INT 13 (8.01-55). Во всех современных компьютерах такая поддержка обеспечивается.

Примечание 5: данные о разделах GPT (GUID Partition Table) содержатся в расширенной записи MBR, занимающей не один сектор, а значительную часть первой дорожки физического диска. Поддержку

разделов GPT обеспечивают только 64-разрядные версии систем Windows server 2003, Windows-XP и Windows Vista.

Примечание 6: идентификаторы 21h, 23h, 26h, 31h, 33h, 34h, 36h, 71h, 73h, 74h, 76h, 86h, A1h, A3h, A4h, B1h, B3h, B4h, B6h, E5h, E6h, F3h, F6h считаются резервированными и пока, насколько известно, не используются.

A.13-7 Таблица свободного дискового пространства

Показанный в таблице блок данных со сведениями о свободном дисковом пространстве возвращает обработчик прерывания INT 21\AX=7303h (8.02-80), который позволяет запрашивать сведения о дисках с файловыми системами FAT-12, FAT-16 и FAT-32.

Смещение	Длина	Содержание
00h	2	Размер данной таблицы в байтах
02h	2	Должно быть = 0000h при вызове
08h	4	Размер сектора в байтах
0Ch	4	Число свободных кластеров
10h	4	Общее число кластеров на диске
14h	4	Число свободных физических секторов на диске
18h	4	Общее число физических секторов на диске
1Ch	4	Число доступных выделяемых элементов
20h	4	Общее число выделяемых элементов на диске

A.14 Порты

A.14-1 Адреса некоторых портов

Порты представляют аппаратную часть компьютера, к которой следует обращаться опосредованно: либо через систему BIOS, адаптированную для обслуживания конкретной материнской платы, либо через драйверы, обслуживающие платы расширения. Как правило, пользовательские программы не обращаются к портам напрямую, хотя у этого правила есть исключения. Но в любом случае полезно знать адреса некоторых портов, хотя бы для того, чтобы избегать конфликтов при конфигурировании аппаратных средств компьютера.

Приведенная ниже таблица показывает относительно стабильные особенности общего распределения адресов портов в AT-совместимых компьютерах. Тем не

менее назначение адресов отдельных портов в Вашем компьютере может отличаться от показанного здесь.

Адреса портов	Адресуемые устройства
0000h – 001Fh	1-й контроллер прямого доступа к памяти (DMA1)
0020h – 0021h	1-й контроллер прерываний (IRQ 1 – IRQ 7, 8.01-09)
0022h – 0023h	Контроллер оперативной динамической памяти
0060h – 0064h	Контроллер клавиатуры (A.11-3)
0070h	Порт приема адреса данных CMOS RAM (примечание 1)
0071h	Порт ввода-вывода данных CMOS RAM (примечание 1)
0080h	Производственный диагностический порт
00A0h – 00A1h	2-й контроллер прерываний (IRQ 8 – IRQ 15, 8.03-75)
00B2h – 00B3h	Управление электропитанием компьютера
00C0h – 00DFh	2-й контроллер прямого доступа к памяти (DMA2)
00F0h – 00FFh	Арифметический сопроцессор
0168h – 016Fh	IFS-устройства или платы расширения
0170h – 0177h	2-й IDE контроллер жестких магнитных дисков (IRQ 15)
01E8h – 01Efh	Мышь PS/2 или другие устройства (IRQ 12)
01F0h – 01F7h	1-й IDE контроллер жестких магнитных дисков (IRQ 14)
01F8h	Управление линией A20 адресной шины
0200h – 020Fh	Игровой порт, джойстик
0279h	Конфигурационная система Plug-and-play
02E8h – 02EFh	Последовательный порт COM4
02F8h – 02FFh	Последовательный порт COM2 (IRQ 3)
0300h – 031Fh	Сетевые карты, совместимые с NE2000
0330h – 0331h	Интерфейс MIDI для музыкальных инструментов
0378h – 037Ah	Параллельный порт LPT1 (IRQ 7)
03C0h – 03CFh	Порты EGA-совместимых видеокарт (примечание 2)
03C4h	Порт селектора EGA-секвенсера (примечание 3)
03C5h	Порт данных EGA-секвенсера (примечание 3)
03CEh	Порт селектора графических регистров (примечание 4)
03CFh	Порт данных для графических регистров (примечание 4)
03DAh	Порт статуса видеокарт CGA/EGA/VGA (примечание 5)
03E0h – 03E7h	Порты контроллера PCMCIA (i82365)
03E8h – 03EFh	Последовательный порт COM3
03F0h – 03F7h	Контроллер флоппи-дисководов (IRQ 6)
03F8h – 03FFh	Последовательный порт COM1 (IRQ 4)
0A79h	Порт данных конфигурационной системы Plug-and-play
0CF8h – 0CFFh	Конфигурационные порты шины PCI

Примечание 1: многие данные из CMOS RAM доступны через программу BIOS Setup (1.01), данные об аппаратуре доступны через INT 11 (8.01-42),

сведения о памяти – через INT 12 (8.01-43). Обращаться к CMOS RAM напрямую приходится для маскирования NMI (примечание 1 к 8.01-03), для получения сведений о флоппи-дисководах (байт 10h, расшифровка в таблице А.08-3), а также для задания действий BIOS после сброса процессора в начальное состояние, зависящих от значения байта 0Fh в CMOS RAM (примечание 4 к А.12-1). Для прямого доступа к данным в CMOS RAM нужно сначала послать номер запрашиваемого байта (до 7Fh) командой OUT (7.03-66) в порт 70h; затем через порт 71h можно будет считать состояние этого байта командой IN (7.03-26) или изменить его командой OUT.

Примечание 2: так как прорисовка графики посредством INT 10\AH=0Ch (8.01-19) выполняется относительно медленно, операционные системы "рисуют" свою графическую оболочку, обращаясь непосредственно к видеопамяти и к портам EGA-совместимых видеокарт. Поэтому адреса портов EGA фактически стали стандартом, хотя сами видеокарты EGA давно сошли со сцены.

Примечание 3: порт 03C5h пересылает байт, посланный из регистра AL командой OUT (7.03-66), по направлению, заранее заданному через порт 03C4h. Если в порт 03C4h заранее был послан байт AL=02h, то тогда байт, посланный в порт 03C5h, будет воспринят как маска цвета (обычно она имеет значение 0Fh).

Примечание 4: порт 03CFh пересылает байт, посланный из регистра AL командой OUT (7.03-66), по направлению, заранее заданному через порт 03CEh. Если в порт 03CEh заранее был послан байт AL=08h, то тогда байт, посланный в порт 03CFh, будет воспринят как маска битов для 8 последовательно расположенных пикселов. Если в порт 03CEh был послан байт AL=05h, то тогда байт, посланный в порт 03CFh, задаст режим записи в видеопамять (00h – 02h, примечание 3 к 8.01-39).

Примечание 5: порт 03DAh предназначен только для выдачи данных, причем в нем каждый интервал обратного хода кадровой развертки отмечается установлением в единицу бита 3. Этот факт регистрируют командой IN (7.03-26) и используют для предотвращения появления "разрывов" изображения, которые иначе были бы видны при смене содержания изображения на прямом ходу кадровой развертки.

А.14-2 Статус последовательного порта

Здесь приведена расшифровка значения битов в байте статуса, возвращаемом в регистре AH функциями INT 14\AH=00h – INT 14\AH=03h (8.01-65 – 8.01-68).

Биты	Значение
0	Принимаемые данные подготовлены
1	Ошибка переполнения
2	Ошибка по четности
3	Ошибка кадровой синхронизации
4	Зарегистрирован перерыв передачи
5	Запоминающий регистр пуст
6	Передающий сдвиговый регистр пуст
7	Ответа нет, время ожидания истекло

А.14-3 Статус принтера, подключенного к параллельному порту

Здесь приведена расшифровка значения битов в байте статуса, возвращаемом в регистре AH функциями INT 17\AH=00h – INT 17\AH=02h (8.01-86 – 8.01-88).

Биты	Значение
0	Ответа нет, время ожидания истекло
1	Только от EPP BIOS: запрошенный порт не поддерживается
2	Не используется
3	Ошибка ввода-вывода
4	Порт задействован
5	В принтере нет бумаги
6	Подтверждение получено
7	Принтер свободен, ждет команды

Примечание 1: статус AH = 03h при установленном флаге CF означает, что EPP BIOS в данном компьютере имеется, но запрошенный параллельный порт ею не поддерживается.

Примечание 2: при ответе от EPP BIOS статус AH=00h означает, что данные возвращены в регистрах (А.14-4).

А.14-4 Некоторые функции EPP BIOS

EPP BIOS – это дополнение системы BIOS компьютера, обеспечивающее передачу данных через параллельные порты LPT согласно спецификации IEEE 1284. О наличии EPP BIOS в Вашем компьютере можно узнать с помощью INT 17\AX=0200h (8.01-88); заодно можно будет выяснить базовый адрес порта LPT, версию EPP BIOS, и адрес вызова ее функций. Если по этому адресу вызова совершить дальний переход командой CALL FAR (7.03-08), то система EPP BIOS выполнит операцию, определяемую кодом команды, который должен быть в

регистре AH в момент перехода. Для последней седьмой модификации EPP BIOS номер запрашиваемого порта LPT определяется его базовым адресом в регистре DX; предыдущие версии EPP BIOS определяли порт LPT по его номеру (00h – 03h) в регистре DL. Помимо того, некоторые функции нуждаются в дополнительных условиях, показанных во второй колонке приведенной ниже таблицы или в примечаниях (отсылки к примечаниям отмечены знаком "звездочка"). Если специально не оговорено иное, то функции EPP BIOS возвращают в регистре AH байт статуса (A.14-7), вызывают потерю значения в регистре BX, возвращают флаг CF сброшенным в нуль в случае успешного завершения и, напротив, устанавливают флаг CF в единицу в случае неудачи.

AH	При вызове	Функция EPP BIOS	При возврате
00h		Определение конфигурации	A.14-5
01h	A.14-6	Задание режима	AX изменен
02h		Выяснение режима	A.14-6
03h	AL=00h	Разрешить прерывания LPT	
03h	AL=01h	Запретить прерывания LPT	
04h		Сброс EPP	AL изменен
05h	AL=адрес	Задание адреса устройства	AL изменен
06h		Считывание адреса	AL=адрес
07h	AL=байт	Передача байта данных	
08h		Передача блока данных	*1
09h		Прием байта данных	AL=байт
0Ah		Прием блока данных	*2
0Bh	AL=адрес	Адресный прием байта	AL=байт
0Ch	AL=адрес	Адресная передача байта	*3
0Dh	AL=адрес	Адресный прием блока	*2
0Eh	AL=адрес	Адресная передача блока	*1
0Fh	AL=порт	Заблокировать порт	*4
10h	AL=порт	Разблокировать порт	*4
11h	CH=00h	Не принимать прерывания	*5
11h	CH=01h	Разрешить прерывания	*5
12h	AL=00h	Подключено ли устройство?	AL=01h если да
12h	AL=01h	Добавить устройство	
12h	AL=02h	Удалить устройство	
40h		Подключен ли мультиплексор?	*6, *7
41h	AL=порт	Запрос порта мультиплексора	*6, *8
50h	AL=порт	Есть ли цепное подключение?	*6, *9
51h	AL=порт	Назначить номера по цепи	*6

Примечание 1: при вызове в DS:SI должен быть указатель на блок данных, а в CX – длина этого блока данных в байтах. Прежние версии, до

модификации 7 EPP BIOS, принимали указатель на блок данных из регистров ES:DI. При возврате в CX – число оставшихся, не пересланных байтов блока.

Примечание 2: при вызове в ES:DI должен быть указатель на буфер для приема данных, а в CX – число байтов, которые надлежит принять. При возврате в CX – число байтов буфера, оставшихся не заполненными после приема блока данных из порта.

Примечание 3: если используется модификация 7 EPP BIOS, то передаваемый байт данных должен быть в регистре CL, а предшествующие версии EPP BIOS принимали передаваемый байт из регистра DH.

Примечание 4: если устройства подключены через мультиплексор, то номер порта мультиплексора (от 1 до 8) указывают в битах 3 – 0 регистра AL, а если устройства подключены цепочкой, то номер адресуемого устройства (от 1 до 8) указывается в битах 7 – 4 регистра AL.

Примечание 5: в регистре AL нужно указать номер порта мультиплексора (01h – 08h) или 00h, если мультиплексор не используется. Для функции разрешения приема прерываний (CH = 01h) еще нужно указать в ES:DI адрес вызова обработчика прерывания.

Примечание 6: эта функция реализована начиная с 7-й модификации EPP BIOS и действует только при групповом подключении устройств либо через мультиплексор, либо цепочкой. Идентификация адресуемого порта LPT производится не по номеру, а по его базовому адресу ввода-вывода, указываемому в регистре DX.

Примечание 7: при возврате в регистре AL – номер мультиплексорного порта, который в данный момент активен, а в регистре CH – байт флагов, в котором установленное состояние бита 0 означает, что порт блокирован, а установленное состояние бита 1 – что данный порт запрашивает вызов обработчика прерывания.

Примечание 8: при возврате в регистре CH – байт флагов, в котором

бит 0 – данный порт активизирован

бит 1 – порт заблокирован

бит 2 – запрос вызова обработчика прерывания разрешен

бит 3 – порт запрашивает вызов обработчика прерывания.

Примечание 9: эта функция возвращает в BH номер модификации EPP BIOS, в BL – номер активного устройства, в CH – байт флагов (такой же, как в примечании 7), в CL – количество подключенных цепочкой устройств (или 00h, если цепочки нет), в ES:DI – указатель на строку с именем фирмы – разработчика драйвера.

А.14-5 Расшифровка байта конфигурации EPP BIOS

Функция "определение конфигурации" EPP BIOS, задаваемая значением AH=00h при вызове (А.14-4), возвращает в AL номер линии IRQ, выделенной порту LPT, в регистре BH возвращает номер версии EPP BIOS, в регистрах ES:DI возвращает указатель на название драйвера, в регистре CX возвращает базовый адрес порта LPT (только если EPP BIOS версии 1.0 – 3.0), а в регистре BL возвращает самое важное – байт конфигурации EPP BIOS. Расшифровка этого байта дана в приведенной ниже таблице.

Биты	Значение
0	Подключен мультиплексор
1	Двунаправленная передача данных поддерживается
2	Устройства подключены цепочкой (daisy chain)
3	Поддерживается спецификация ECP
4	Поддерживается программная эмуляция функций EPP BIOS
5	Функции EPP BIOS действуют
6	Поддерживается протокол передачи "fast Centronics"
7	Разводка линий порта EPP стандартная

А.14-6 Расшифровка байта режима передачи через порт EPP

Функция EPP BIOS "выяснение режима передачи" (AH=02h, А.14-4) возвращает в регистре AL байт режима, а функция "задать режим передачи" (AH=01h, А.14-4) принимает байт режима из регистра AL. Состояние регистра AH при возврате не сохраняется. Расшифровка значения отдельных битов в байте режима показана в приведенной ниже таблице.

Биты	Значение
0	Совместимый режим передачи
1	Двунаправленная передача разрешена
2	Передача согласно спецификации EPP
3	Передача согласно спецификации ECP (примечание 1)
4	Программная эмуляция функций EPP BIOS (примечание 1)
5	Передача согласно спецификации "fast Centronics" (примечание 1)
6	= 0b (резервировано)
7	Запросы на прерывания принимаются (примечание 2)

Примечание 1: биты 3 – 5 можно устанавливать в единицу только если используется 7-я модификация EPP BIOS.

Примечание 2: бит 7 не принимается функцией "задать режим передачи" (AH=01h, A.14-4), но отображается при возврате функцией "выяснение режима передачи" (AH=02h, A.14-4).

A.14-7 Расшифровка кода статуса EPP BIOS

Почти все функции EPP BIOS (кроме 01h и 02h, A.14-4) возвращают в регистре AH код статуса; интерпретация этого кода дана в приведенной ниже таблице.

Код	Значение
00h	Успешное завершение операции
02h	Данная команда или свойство не поддерживаются
03h	Запрошенный параллельный порт не поддерживается
05h	При установленном режиме данный запрос не поддерживается
06h	Неверно указана подфункция
07h	Запрошенное состояние уже установлено ранее
20h	Мультиплексор к порту LPT не подключен (BIOS фирмы AMI)
40h	Мультиплексор к порту LPT не подключен
41h	Канал мультиплексора заблокирован
80h	Нет ответа, время ожидания истекло
FFh	Неверно указанная или не поддерживаемая функция

A.15 Таблицы параметров оптических дисков

A.15-1 Формат пакета загрузочной спецификации

Здесь представлен формат пакета загрузочной спецификации, на основании которого посредством INT 13\AH=4A00h или INT 13\AX=4C00h (8.01-61) система BIOS осуществляет эмуляцию диска по его копии, считываемой с загрузочного оптического диска.

Смещение	Длина	Содержание
00h	1	Размер пакета в байтах (обычно = 13h)
01h	1	Тип копии диска (то же, что 21h в таблице A.15-3)
02h	1	Эмулируемый дисковод: 00h (A:), 80h (C:),...
03h	1	Номер эмулируемого дискового контроллера
04h	4	Номер начального логического блока копии загрузочного диска (то же, что 28h в A.15-3).

Продолжение таблицы A.15-1

08h	2	Бит 0: эмулировать ведомый дисковод IDE биты 7-1: идентификатор и номер LUN (для SCSI) биты 15-8: номер шины (для SCSI).
0Ah	2	Сегментный адрес кэш-буфера емкостью 3 кбайт или значение 0000h, если кэширования нет.
0Ch	2	Сегментный адрес загрузки boot-сектора эмулируемого диска (то же, что 22h в A.15-3).
0Eh	2	Число виртуальных 512-байтовых секторов в копии загрузочного диска (то же, что 26h в A.15-3).
10h	1	Младшие 8 бит максимального номера цилиндра эмулируемого диска (= CH в 8.01-49).
11h	1	Биты 0-5: номер последнего сектора на дорожке, биты 6-7: старшие 2 бита максимального номера цилиндра эмулируемого диска (= CL в 8.01-49).
12h	1	Число головок эмулируемого дисковода.

A.15-2 Формат командного пакета

Показанный здесь формат командного пакета используется при вызове процедуры INT 13\AH=4Dh (8.01-63) считывания группы секторов оптического диска. Таким способом считывают загрузочный каталог оптического диска.

Смещение	Длина	Содержание
00h	1	Длина командного пакета в байтах (обычно = 08h)
01h	1	Число секторов, которые надлежит считывать
02h	4	Адрес буфера размещения считываемых данных
06h	2	Номер начального сектора считываемой группы

A.15-3 Формат загрузочного каталога оптического диска

Оптические диски потенциально позволяют осуществить несколько вариантов загрузки компьютера. Необходимые для этого данные должны быть на диске в виде записей скрытого каталога, который считывают посредством INT 13\AH=4Dh (8.01-63). Как в любом каталоге, записи имеют стандартную длину 20h байт. Минимальный состав загрузочного каталога включает две обязательных записи: запись области действия (validation entry) и дескриптор копии загрузочного диска, принимаемого по умолчанию. Приведенная ниже таблица представляет структуру этих двух записей, причем записи области действия соответствует интервал смещений 00h – 1Fh, а дескриптору копии загрузочного диска – интервал смещений 20h – 3Fh.

Смещение	Длина	Содержание
00h	1	= 01h: идентификатор записи области действия
01h	1	Тип платформы: = 00h – AT-совместимые компьютеры = 01h – компьютеры класса Power PC = 02h – компьютеры Apple Macintosh
04h	24	Строка с наименованием фирмы-разработчика
1Ch	2	Контрольная сумма байтов 00h-1Fh
1Eh	2	= AA55h: метка конца записи области действия
20h	1	= 88h: признак загрузочного дескриптора
21h	1	Биты 3-0: = 0000b недействительный дескриптор = 0001b дескриптор дискеты 1.2 Мбайт = 0010b дескриптор дискеты 1.44 Мбайт = 0011b дескриптор дискеты 2.88 Мбайт = 0100b дескриптор жесткого диска бит 6: диск из дисковода с интерфейсом ATAPI бит 7: диск из дисковода с интерфейсом SCSI
22h	2	Сегментный адрес загрузки boot-сектора эмулируемого диска (при = 0000h по умолчанию принимается адрес 07C0h)
24h	1	Идентификатор файловой системы (A.13-6)
26h	2	Длина копии загрузочного диска на оптическом диске, выраженная числом виртуальных 512-байтовых секторов
28h	4	Номер логического блока, начиная с которого на оптическом диске размещена копия эмулируемого диска

Примечание 1: помимо двух обязательных записей, представленных в таблице А.15-3, загрузочный каталог может включать другие записи, группируемые в несколько секций. Каждая секция представляет отдельный вариант загрузки и содержит не менее двух записей: заголовок и дескриптор копии загрузочного диска для данного варианта загрузки. Запись заголовка начинается с байта 90h, кроме заголовка последней секции, который начинается с байта 91h, причем слово со смещением 02h в каждой записи заголовка объявляет число записей в данной секции. За дескриптором копии загрузочного диска в каждой секции могут следовать дополнительные записи. Дескрипторы, входящие в состав секций, имеют такую же структуру, как дескриптор принимаемого по

умолчанию варианта загрузки, показанный в таблице А.15-3 в интервале смещений 20h – 3Fh.

Примечание 2: допускается указывать дескрипторы незагрузочных дисков. У них в начальный байт записывается идентификатор 00h.

А.15-4 Команды, исполняемые драйверами оптических дисководов

Чтобы послать команду драйверу, нужно прежде всего получить для этого номерную ссылку. Первый шаг – определение адреса заголовка драйвера вызовом прерывания INT 2F\AX=1501h (8.03-14), исполняемого резидентными программами Mscdex.exe (5.08-03) или Shsucdx.com (5.08-04). Вторым шагом должно стать считывание 8-байтового имени канала доступа к драйверу, записанного в заголовке драйвера начиная со смещения 0Ah. Обычно именем канала доступа становится идентификатор, указываемый после параметра /D: в командной строке загрузки драйвера (например, идентификатор /D:MSCD001 в разделах 5.09-01 – 5.09-03). Третьим шагом является использование имени для получения номерной ссылки с помощью INT 21\AH=3Dh (8.02-33). Имя канала доступа должно быть написано заглавными буквами и дополнено до 8 знаков пробелами, если оно короче. Полученная номерная ссылка помещается в регистр BX перед посылкой запроса к драйверу посредством INT 21\AX=4403h или INT 21\AX=4402h (8.02-41). Еще перед посылкой запроса надо подготовить блок данных запроса, записать его адрес в регистры DS:DX, а его длину – в регистр CX. Длина блока данных запроса для разных команд показана во второй колонке приведенной ниже таблицы. В третьей колонке таблицы показан код операции, который следует записать начиная со смещения 00h в блок данных запроса. Если в результате исполнения должны быть возвращены какие-либо данные, то они, как правило, вписываются драйвером в тот же блок данных запроса.

AX	CX	Код	Операция	Примечания
4402h	05h	00h	Адрес заголовка драйвера	*2
4402h	06h	01h	Положение головки	*3
4402h	09h	04h	Статус аудиоуправления	А.15-5
4402h	05h	06h	Статус CD-дисковода	А.15-6
4402h	04h	07h	Режим считывания	*4
4402h	05h	08h	Число секторов	*2
4402h	02h	09h	Статус смены диска	*5
4402h	07h	0Ah	Число записей	*6
4402h	08h	0Bh	Найти начало записи	*7
4403h	01h	00h	Выдвинуть лоток	
4403h	02h	0100h	Разблокировать диск	
4403h	02h	0101h	Заблокировать диск	

Продолжение таблицы А.15-4

4403h	01h	02h	Сброс драйвера	*1
4403h	09h	03h	Режим воспроизведения	А.15-5
4403h	01h	05h	Вдвинуть лоток внутрь	

Примечание 1: после каждого запроса к драйверу оптического дисковода посредством INT 21\AX=4402h и до активизации драйвера с любой другой целью необходимо переустановить начальное состояние драйвера операцией 02h прерывания INT 21\AX=4403h.

Примечание 2: после исполнения операций 00h и 08h в возвращаемый блок данных запроса начиная со смещения 01h записывается 4-байтовый результат: число или адрес согласно запрошенной операции.

Примечание 3: после операции "положение головки" в возвращаемом блоке данных байт со смещением 01h выражает формат адресации: 00h – формат HSG, 01h – формат "Red Book" (кадры/секунды/минуты/пустой байт); а начиная со смещения 02h драйвер возвращает 4-байтовое число – положение головки в тех единицах, которые определяются форматом адресации.

Примечание 4: после операции "режим считывания" в возвращаемом блоке данных байт со смещением 01h выражает режим считывания: 00h – считывание с коррекцией ошибок и без выдачи корректирующего кода ECC (cooked), 01h – считывание без коррекции ошибок и с выдачей корректирующего кода ECC (raw), а в слове со смещением 02h драйвер возвращает размер сектора.

Примечание 5: после операции "статус смены диска" в возвращаемом блоке данных байт со смещением 01h означает следующее:

= 00h – факт смены диска не установлен,
= 01h – диск не был сменен,
= FFh – диск был сменен.

Примечание 6: после операции "число записей" в возвращаемом блоке данных байт со смещением 01h содержит номер первой сигналограммы (записи), байт со смещением 02h – номер последней сигналограммы, а начиная со смещения 04h там записан 4-байтовый начальный адрес первой сигналограммы (в формате "Red Book").

Примечание 7: при запросе "найти начало записи" в блоке данных запроса байт со смещением 01h должен содержать номер запрашиваемой сигналограммы. При возврате в том же блоке данных начиная со смещения 02h возвращается 4-байтовый адрес начала запрошенной сигналограммы, а в слове со смещением 06h возвращаются флаги:

бит 12 – фонограмма с преэмфазисом,
бит 13 – цифровое копирование разрешено,
бит 14 – сигналограмма с записью данных,
бит 15 – четырехканальная фонограмма.

А.15-5 Статус управления звуковоспроизведением

Если установлен бит 8 в слове статуса (А.15-6) оптического дисковода, значит, данный дисковод предоставляет возможность управления звуковоспроизведением без помощи звуковой платы.

При посылке запроса оптическому дисководу посредством функции INT 21\AX=4403h (А.15-4) в регистрах DS:DX должен быть указан адрес блока данных с задаваемыми параметрами. Представленная здесь таблица показывает структуру блока данных, который посылают при запросе операции 03h для изменения режима звуковоспроизведения. Блок данных такой же структуры драйвер оптического дисковода записывает в подготовленный буфер в ответ на запрос операции 04h о текущем статусе звуковоспроизведения, посылаемый посредством функции INT 21\AX=4402h (А.15-4). В момент выдачи запроса подготовленный буфер должен быть не пуст: код запрашиваемой операции (04h) должен быть указан там в байте со смещением 00h заранее.

Смещение	Длина	Содержание
00h	1	= 03h при AX=4403h или = 04h при AX=4402h
01h	1	Входной канал (0-3) для выходного канала 0
02h	1	Коэффициент передачи для выходного канала 0
03h	1	Входной канал (0-3) для выходного канала 1
04h	1	Коэффициент передачи для выходного канала 1
05h	1	Входной канал (0-3) для выходного канала 2
06h	1	Коэффициент передачи для выходного канала 2
07h	1	Входной канал (0-3) для выходного канала 3
08h	1	Коэффициент передачи для выходного канала 3

Примечание 1: выходные каналы 0 и 1 соответствуют левому и правому, а каналы 2 и 3 – заднему левому и заднему правому. Для выключения любого канала ему следует задать коэффициент передачи 00h.

Примечание 2: по умолчанию каждый выходной канал получает сигнал из входного канала с тем же номером, причем коэффициент передачи устанавливается максимальным (=FFh).

А.15-6 Расшифровка слова статуса оптического дисковода

При обращении к драйверу оптического дисковода с запросом операции 06h посредством прерывания INT 21\AX=4402h (А.15-4) в блоке данных запроса, начиная со смещения 01h, возвращается слово статуса оптического дисковода. Расшифровка состояний битов слова статуса дана в приведенной ниже таблице.

Бит	Значение
0	Лоток дисковода выдвинут
1	Диск в дисководе разблокирован
2	Выдача кода ECC поддерживается (примечание 1)
3	Дисковод обеспечивает запись на оптические диски
4	Поддерживается воспроизведение видеофильмов
5	Поддержка видеофонограмм с перемежением (примечание 2)
7	Поддержка предварительной выборки (примечание 3)
8	Поддержка управления звуковоспроизведением
9	Поддержка адресации по времени (формата "Red Book")
10	Дисковод занят воспроизведением фонограммы
11	Оптический диск в дисководе отсутствует
12	Дисковод с раздельными подканалами записи и считывания

Примечание 1: здесь имеется ввиду поддержка воспроизведения в режиме "raw", при котором вместе со считанными данными выдаются пакеты корректирующего кода ECC. При обычном режиме ("cooked") ошибки воспроизведения корректируются, но корректирующий код ECC на выход не выдается.

Примечание 2: здесь перемежение относится к видеофайлам, содержащим чередующиеся группы кадров изображения и звукового сопровождения.

Примечание 3: запросы предварительной выборки вызывают считывание данных с диска во встроенный буфер дисковода, чтобы впоследствии они могли бы быть доступны сразу, не требуя ожидания завершения дисковой операции.

А.16 Словарь используемых аббревиатур

ACPI – протокол BIOS по предоставлению операционной системе сведений о параметрах материнской платы (адресах портов и т.п.) в виде таблиц в выделенных областях памяти компьютера.

AGP – Accelerated Graphic Port: особый разъем для видеоплат, а также спецификация их взаимодействия с материнской платой компьютера

AH – 8-битовый регистр центрального процессора, представляющий часть (биты 8 – 15) регистра AX

AL – 8-битовый регистр центрального процессора, представляющий часть (биты 0 – 7) регистра AX.

AMIS – Alternate Multiplex Interrupt Specification: альтернативная спецификация мультиплексного прерывания (A.07-6).

ANSI – American National Standards Institute: институт стандартов США

API – Application Program Interface: службы операционной системы, предоставляемые в пользование прикладным программам

APM – Advanced Power Management: расширение системы BIOS, обеспечивающее управление электропитанием (8.01-70 – 8.01-72)

ASCII – American Standard Code for Information Interchange: американский стандартный код для обмена информацией.

ASCIIZ – строка знаков кода ASCII, заканчивающаяся одним или несколькими байтами 00h.

ASPI – Advanced SCSI Programming Interface: набор команд для интерфейса SCSI (5.07-03). Многие из этих команд реализованы также на базе интерфейсов ATAPI (5.07-01) и USB (5.07-05).

AT – Advanced Technology: модель компьютера, выпущенная фирмой IBM в 1984 году и предопределившая технические решения в последующих моделях компьютеров многих других фирм.

ATA – AT Attachment: интерфейс накопителей IDE на жестких магнитных дисках, впервые примененный в компьютерах AT.

ATAPI – ATA Packet Interface: усовершенствованный пакетный вариант интерфейса ATA (подробнее – в разделе 5.07-01).

ATX – AT eXtension: измененная спецификация габаритов и характеристик компьютерных блоков, введенная с 1998 года.

AUX – резервированное слово для адресации последовательного порта COM1.

AVI – суффикс имен аудиовизуальных файлов с чередующимися кадрами изображения и звукового сопровождения.

AX – 16-битовый регистр общего назначения, сопряженный с арифметическим устройством процессора. В 32-разрядных процессорах AX представляет часть (биты 0 – 15) расширенного регистра EAX.

b – binary: отличительная метка двоичных чисел.

BAT – суффикс имен особых командных файлов: batch-файлов, отличающихся расширенным составом команд и тем, что интерпретатор Command.com принимает их как исполняемые файлы без перенаправления ввода.

BH – 8-битовый регистр центрального процессора, представляющий часть (биты 8 – 15) регистра BX.

BIOS – Basic Input-Output System: базовая система ввода-вывода, обслуживающая материнскую плату и поставляемая вместе с ней.

BL – 8-битовый регистр центрального процессора, представляющий часть (биты 0 – 7) регистра BX.

BP – Base Pointer: 16-битовый регистр, обычно играющий роль базы для адресации в сегменте стека. В 32-разрядных процессорах BP представляет часть (биты 0 – 15) расширенного регистра EBP

BPB – BIOS Parameters Block: блок параметров диска (А.03-4).
BSD – Berkley Software Distribution: название свободно распространяемой операционной системы.
BX – 16-битовый регистр общего назначения, способный служить базой для адресации. В 32-разрядных процессорах BX представляет часть (биты 0 – 15) расширенного регистра EBX.
CD – Compact Disc: оптический диск емкостью 650-800 Мбайт.
CD-ROM – оптический диск с нестираемой сигналограммой или дисковод для таких дисков.
CDS – Current Directory Structure: таблица параметров доступа (А.03-3).
CF – (1): Carry Flag – флаг переноса в центральном процессоре.
CF – (2): Compact Flash – тип сменных карт памяти.
CGA – Color Graphic Adapter: первый видеоадаптер, формировавший цветное изображение в графических видеорежимах.
CH – 8-битовый регистр центрального процессора, представляющий часть (биты 8 – 15) регистра CX.
CHS – Cylinder-Head-Sector: один из способов адресации сигналограмм на магнитных дисках (примечание 2 к А.13-6)..
CL – 8-битовый регистр центрального процессора, представляющий часть (биты 0 – 7) регистра CX.
CMOS – Complementary Metal-Oxide Semiconductor: блок памяти BIOS, построенный на комплементарных металлоокисных микросхемах и не теряющий данные при выключении компьютера.
COM – (1): суффикс имен исполняемых файлов, не содержащих блока заголовка.
COM – (2): резервированное слово для доступа к последовательным портам.
COM – (3): Common Object Model – одна из методик программирования.
CON – console: резервированное слово для ввода данных с клавиатуры и для вывода данных на дисплей.
CP – Code Page: кодовая страница (подробнее в приложении А.02-2).
CP/M – Control Program for Microcomputers: прототип операционной системы DR-DOS фирмы Digital Research.
CPU – Central Processing Unit: центральный процессор компьютера.
CR – Control Registers: 32-разрядные управляющие регистры (А.11-4). Регистры CR0, CR2 и CR3 впервые введены в процессор 80386, а CR4 – в процессор Pentium.
CRC – Cyclic Redundancy Check: избыточный проверочный код для выявления (но не исправления) ошибок в файлах.
CRT – Cathode Ray Tube: электронно-лучевая трубка, кинескоп.
CS – Code Segment: 16-битовый сегментный регистр, задающий сегментный адрес исполняемых процессором машинных команд.

CSM – Compatibility Support Module: - дополнение к UEFI BIOS, исполняющее функции обычной BIOS чтобы загружать DOS, Windows-XP и другие OS, которые не могут стартовать из UEFI.

CWR – Control Word Register: управляющий регистр арифметического сопроцессора.

CX – 16-битовый регистр общего назначения, часто используемый в качестве счетчика. В 32-разрядных процессорах CX представляет часть (биты 0 – 15) расширенного регистра ECX.

DAC – Digital-to-Analog Converter: цифроаналоговый преобразователь.

dd – двухзначное десятичное число, обозначающее день месяца.

DDO – Dynamic Drive Overlay: резидентная программа фирмы OnTrack для преодоления предела емкости дисков 512 Мбайт в старых компьютерах.

DH – 8-битовый регистр центрального процессора, представляющий часть (биты 8 – 15) регистра DX.

DI – Destination Index: 16-битовый регистр, обычно хранящий смещение для адреса назначения. В 32-разрядных процессорах DI представляет часть (биты 0 – 15) расширенного регистра EDI.

DL – 8-битовый регистр центрального процессора, представляющий часть (биты 0 – 7) регистра DX.

DMA – Direct Memory Access: прямой доступ к памяти.

DOS – Disk-based Operating System: дисковая операционная система.

DPB – Drive Parameters Block: блок параметров дисковода (A.03-1).

DPMI – DOS Protected Mode Interface: сервисные функции для программ, исполняемых в режиме V86. DPMI реализован, в частности, "окном DOS" операционной системы Windows (8.03-21).

DPR – Data Pointer Register: регистр – указатель данных в арифметическом сопроцессоре.

DPTE – Drive Parameter Table Extension: таблица дополнительных параметров дисковода (A.13-3).

DR – (1): Digital Research – название известной фирмы, разработавшей операционные системы CP/M и DR-DOS.

DR – (2): Debug Registers – отладочные регистры DR0 – DR7, (A.11-5).

DS – Data Segment: 16-битовый сегментный регистр, задающий сегментный адрес блока данных исполняемой программы.

DTA – Data Transfer Area: область передачи данных (8.02-16, A.09-1)

DVD – Digital Versatile Disk – оптический диск повышенной емкости, обычно вмещающий до 4.7 Гигабайта данных на одной стороне.

DX – 16-битовый регистр общего назначения. В 32-разрядных процессорах DX представляет часть (биты 0 – 15) расширенного регистра EDX.

EAX – регистр общего назначения в 32-разрядных процессорах. Младшие 16 бит регистра EAX представлены регистром AX.

EBIOS – расширение системы BIOS, обеспечивающее адресацию LBA для доступа к жестким магнитным дискам большой емкости.

EBP – базовый адресный регистр в 32-разрядных процессорах. Младшие 16 бит регистра EBP представлены регистром BP.

EBX – регистр общего назначения в 32-разрядных процессорах. Младшие 16 бит регистра EBX представлены регистром BX.

ECC – Error Correcting Code: избыточный код для исправления ошибок.

ECP – Extended Capabilities Port: расширенная спецификация пересылки данных через параллельный порт LPT.

ECX – регистр общего назначения в 32-разрядных процессорах. Младшие 16 бит регистра ECX представлены регистром CX.

EDI – индексный адресный регистр в 32-разрядных процессорах. Младшие 16 бит регистра EDI представлены регистром DI.

EDX – регистр общего назначения в 32-разрядных процессорах. Младшие 16 бит регистра EDX представлены регистром DX.

EFI – Extensible Firmware Interface: спецификация 32-битных систем BIOS фирмы Intel для 64-разрядных одноядерных процессоров Itanium (2002). Модификация EFI для новых 32-разрядных многоядерных процессоров известна как UEFI (2007).

EGA – Enhanced Graphics Adapter: устаревшая видеоплата фирмы IBM, надолго определившая формы интерфейса видеоплат.

EHCI – Enhanced Host Controller Interface: алгоритм управления контроллерами шины USB версий 2.x (подробнее в 5.07-05).

EMM – Expanded Memory Manager: драйвер EMM386.EXE (5.04-02).

EMS – Expanded Memory Specification: спецификация доступа к расширенной памяти, реализуемая драйвером EMM (5.04-02).

EOF – End Of File: знак конца файла – байт 1Ah в коде ASCII.

EPP – Enhanced Parallel Port: расширение системы BIOS, обеспечивающее дополнительные функции порта LPT (A.14-4).

ES – 16-битовый сегментный регистр центрального процессора, обычно задающий сегмент для адреса назначения.

ESI – индексный адресный регистр в 32-разрядных процессорах. Младшие 16 бит регистра ESI представлены регистром SI.

ESP – регистр – указатель вершины стека в 32-разрядных процессорах. Младшие 16 бит регистра ESP представлены регистром SP.

EXE – Executable: суффикс имен исполняемых файлов, содержащих блок заголовка.

FASM – Flat ASseMbler: современный ассемблер для DOS, Windows, Linux и UNIX. Свободно выложен на сайте http://www.flatassembler.net/ .

FAT – File Allocation Table: таблица выделения дискового пространства для размещения файлов и каталогов.
FCB – File Control Block: блок управления файлом (A.09-5).
FCBS – команда резервирования памяти под блоки FCB (4.10).
FDD – Floppy Disk Drive: дисковод для гибких магнитных дисков.
FS – дополнительный 16-битовый сегментный регистр, введенный начиная с процессора 80386.
GDT – Global Descriptor Table – таблица 8-байтовых дескрипторов (A.12-2), задающих уровни привилегий программ и границы сегментов в защищенном режиме.
GDTR – системный регистр процессора, хранящий линейный адрес и размер таблицы GDT.
GS – дополнительный 16-битовый сегментный регистр, введенный начиная с процессора 80386.
GUI – Graphical User Interface: графический интерфейс пользователя – альтернатива командной строке текстовых видеорежимов.
GUID – Globally Unique IDentifier – "глобально-уникальный" 32-байтный идентификатор.
h – hexadecimal: отличительная метка шестнадцатеричных чисел.
HDD – Hard Disk Drive: накопитель на жестких магнитных дисках.
HMA – High Memory Area: область верхней памяти 1024 – 1088 кбайт.
HRS – Hidden, Read-only, System: скрытый, системный, только для чтения – типичный набор атрибутов для системных файлов.
HSG – High Sierra Group: спецификация формата оптических дисков, послужившая прототипом стандарта ISO 9660.
IBM – International Business Machines – название известной фирмы, разработавшей операционные системы PC-DOS и OS/2.
ID – identifier: идентификатор.
IDE – Integrated Drive Electronics: дисководы со встроенной электроникой. Термин IDE используется как эквивалент ATA.
IDT – Interrupt Descriptor Table – таблица прерываний защищенного режима или обозначение ее сегментного дескриптора.
IDTR – системный регистр процессора, хранящий линейный адрес и размер таблицы прерываний.
IEEE – Institute of Electrical and Electronics Engineers: институт инженеров электротехники и электроники (США), основан в 1963 г.
IFS – Installable File System: файловая система (5.08-01), доступ к которой обеспечивается загружаемым драйвером.
IML – Initial Machine Load: система начальной загрузки компьютера.
INT – Interrupt: прерывание – событие или команда (7.03-28) вызова обработчика прерывания.
I/O – Input-Output: ввод-вывод, операции приема и передачи данных.

IOCTL – Input-Output Control: управление вводом-выводом (8.02-41).
IP – Instruction Pointer: указатель команды – регистр, формирующий адрес следующей исполняемой команды. В 32-разрядных процессорах IP представляет часть (биты 0 – 15) расширенного регистра EIP.
IPR – Instruction Pointer Register: регистр – указатель команд в арифметическом сопроцессоре.
IRQ – Interrupt ReQuest: линия запроса прерывания.
ISA – Industrial Standard Architecture: стандартизованный тип шины для плат расширения, после 2000 года вытесненный шиной PCI.
ISO – International Standards Organization: международная организация по стандартизации.
ISP – Interrupt Sharing Protocol: протокол совместного пользования номером прерывания (подробнее – в А.07-5).
JFT – Job File Table: таблица открытых ссылок (примечание 3 к А.07-1).
LAN – Local Area Network: локальная компьютерная сеть.
LBA – Linear Block Addressing: линейная блочная адресация – способ адресации для дисков большой емкости (примечание 4 к А.13-6).
LCD – Liquid Crystal Display: жидкокристаллический индикатор (дисплей).
LFN – Long File Name: "длинное" имя файла (А.09-3).
LIM – Lotus-Intel-Microsoft: фирмы, разработавшие спецификацию EMS.
LPT – Line PrinTer: параллельный порт, обычно для подключения принтера.
LUN – Logical Unit Number: номер логического устройства (подробнее в примечании 1 к А.03-2).
MASM – MacroASseMbler: низкоуровневый ассемблер макрокоманд, разработанный фирмой Microsoft.
MBR – Master Boot Record: главная загрузочная запись (А.13-5).
MCB – Memory Control Block: дескриптор (А.12-7), предшествующий каждому блоку памяти, который выделила DOS.
MDA – Monochrome Display Adapter: видеоплата формирования черно-белого изображения для древних компьютеров фирмы IBM.
mm – двухзначное десятичное обозначение номера месяца в году.
MO – Magneto-Optical: магнитооптические диски или дисководы.
MS – Microsoft's: обозначение принадлежности фирме Microsoft.
MSWR – Machine Status Word Register: управляющий регистр процессора 80286. В современных процессорах он – часть регистра CR0.
NTFS – New Technology File System: файловая система для жестких магнитных дисков, используемая OS Windows NT/2000/XP.
NUL – (1): канал "в никуда", альтернатива посылке в реальный канал.
NUL – (2): обозначение байта 00h.

OEM – Original Equipment Manufacturer: поставки комплектующих элементов производителям аппаратуры по их заказам, как альтернатива retail-поставкам в розничную торговлю.

OHCI – Open Host Controller Interface: вариант алгоритма управления контроллерами шины USB версий 1.x (подробнее в 5.07-05).

OS – Operating System: операционная система.

PC – Personal Computer: персональный компьютер.

PCI – Peripheral Components Interconnect: шина соединения периферийных приборов – самый распространенный сейчас тип шины для плат расширения.

PCMCIA – PC Memory Card International Association: стандарт интерфейса, изначально разработанный для сменных блоков памяти (5.07-02).

PD – Powerful Disk: перезаписываемый оптический диск емкостью 650 Мбайт класса CD-RAM, прототип дисков DVD-RAM.

PIO – Programmed I/O: программное управление пересылкой данных для приборов с интерфейсом ATAPI.

PM – Protected Mode: защищенный режим работы процессора в компьютере

POST – Power-On Self Test: самопроверка компьютера, выполняемая его системой BIOS при каждом включении электропитания.

PRN – резервированное слово для адресации порта принтера (LPT1).

PSP – Program Segment Prefix: префикс сегмента программы (A.07-1).

PS/2 – Personal System/2: модель компьютера фирмы IBM (1987 г.)

PS2 – тип разъема для подключения манипулятора "мышь", впервые примененный в компьютерах PS/2

PTS – PhysTechSoft – фирма, разрабатывающая программное обеспечение, известная своей операционной системой PTS-DOS

RAID – Redundant Array of Inexpensive Disks: распределенная запись на несколько дисков, более быстрая и снижающая риск потери данных при условии квалифицированного обслуживания.

RAM – Random Access Memory – обычная память с произвольным доступом, альтернатива последовательному чтению с носителя

ROM – Read-Only Memory: память с постоянной сигналограммой.

SCSI – Small Computer System Interface: системная шина для малых компьютеров (5.07-03).

SFT – System File Table: системная таблица файлов (A.01-4)

SFX – SelF eXtracting: самораспаковывающийся архив или модуль.

SI – Source Index: 16-битовый регистр, обычно хранящий смещение для адреса источника данных. В 32-разрядных процессорах SI представляет часть (биты 0 – 15) расширенного регистра ESI.

SIMD – Single Instruction Multiple Data: команды, каждая из которых выполняет свою операцию над несколькими элементами данных.

SP – Stack Pointer: 16-битовый регистр – указатель вершины стека. В 32-разрядных процессорах SP представляет часть (биты 0 – 15) расширенного регистра ESP.

SS – Stack Segment: 16-битовый сегментный регистр, содержимое которого задает сегментный адрес стека.

SSE – Streaming SIMD Extensions: "потоковые" расширения состава команд SIMD в современных процессорах.

STDIN – канал ввода данных, ассоциированный по умолчанию с клавиатурой и с номерной ссылкой 0000h.

STDOUT – канал вывода данных, ассоциированный по умолчанию с дисплеем и с номерной ссылкой 0001h.

STDERR – канал, ассоциированный с номерной ссылкой 0002h и используемый для вывода сообщений об ошибках на дисплей.

SVGA – SuperVGA: усовершенствованные видеорежимы (A.10-1), предлагаемые и стандартизованные организацией VESA.

SWR – Status Word Register: регистр флагов арифметического сопроцессора (подробнее – в 7.04-08 и в 7.04-64).

TASM – TurboASseMbler: низкоуровневый ассемблер макрокоманд, разработанный фирмой Borland.

TLB – Translation Lookaside Buffer: кэш-буфер в центральном процессоре, из которого на основании поступающих линейных адресов производится выборка физических адресов памяти.

TSR – Terminate and Stay Resident: обозначение резидентных (постоянно находящихся в памяти) модулей и программ (8.02-23).

TWR – Tags Word Register: регистр тегов арифметического сопроцессора.

UEFI – Unified EFI - модификация EFI, принятая в 2007 году для новых 32-разрядных многоядерных процессоров. UEFI предусматривает графическую оболочку, поддержку сетей и сохранение совместимости с обычными BIOS посредством CSM-модуля.

UHCI – Universal Host Controller Interface: вариант алгоритма управления контроллерами шины USB версий 1.x (подробнее в 5.07-05).

UMB – Upper Memory Blocks: участки адресного пространства в пределах 640 – 1024 кбайт, используемые для загрузки драйверов.

USB – Universal Serial Bus: универсальная последовательная шина для подключения к компьютеру внешних устройств (5.07-05).

V86 – virtual 8086 mode: режим эмуляции CPU 8086 современными процессорами, работающими в защищенном режиме, с исполнением программ DOS на низшем (3-м) уровне привилегий.

VBE – Video BIOS Extensions: расширения системы BIOS для обеспечения видеорежимов SVGA, разработанные организацией VESA (8.01-35).

VC – Volcov Commander: резидентный файл-менеджер (6.25).

VCPI – Virtual Control Program Interface: протокол взаимодействия управляющих программ, обеспечивающий бесконфликтную передачу управления от одной управляющей программы к другой (подробнее – в разделе 5.04-02).

VESA – Video Electronics Standards Association: ассоциация по стандартизации видеотехники и электроники.

VGA – Video Graphics Array: видеоплата, разработанная фирмой IBM для компьютеров PS/2, а также формат развёртки 640x480, впервые реализованный видеоплатами VGA..

XMS – Extended Memory Specification: спецификация доступа к памяти, реализуемая драйвером HIMEM.SYS (5.04-01).

YIQ – представление отсчета изображения в 3-х координатах: одной яркостной и двух цветностных, соответствующих наибольшей и наименьшей цветовой разрешающей способности зрения.

YUV – представление отсчета изображения в 3-х координатах: одной яркостной и двух цветностных, используемых в диаграммах CIE (международного комитета по светотехнике).

yy – обозначение года (в MS-DOS7 год выражается четырехзначным десятичным числом).

ZIP – (1): суффикс архивных файлов, сжатых программой PKZIP.

ZIP – (2): торговая марка серии дисководов на сменных магнитных дисках, разработанных фирмой Iomega.

ZF – флаг нуля в центральном процессоре, устанавливаемый в состояние ZR при равенстве или при нулевом результате выполненной операции.