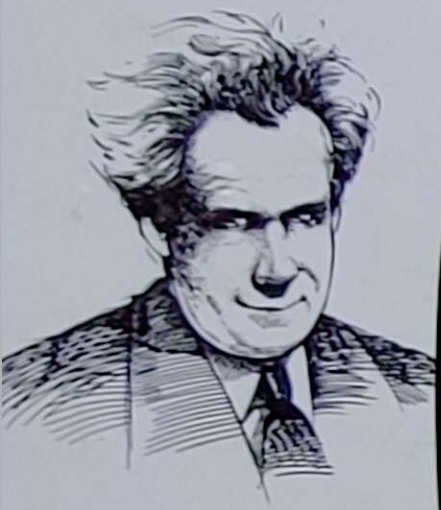
2'88

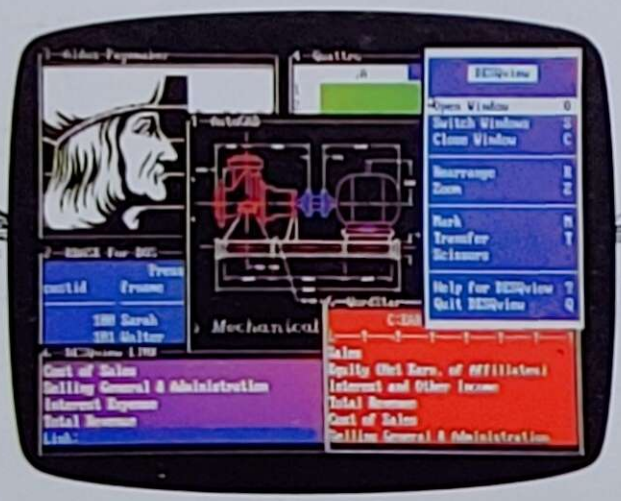
ISSN 0235-3520

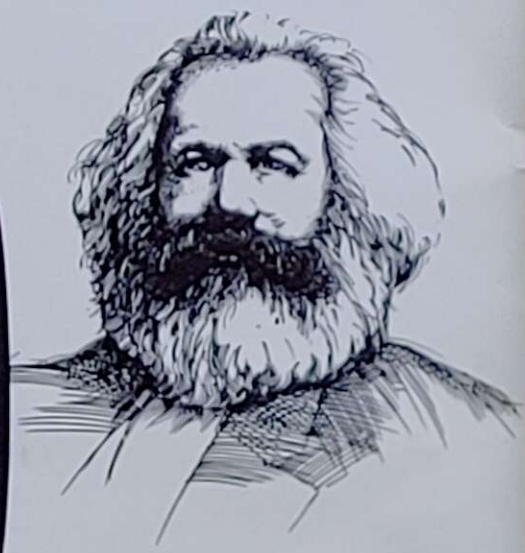
В МИРЕ ПЕРСОНАЛЬНЫХ КОМПЬЮТЕРОВ

„Не только результат исследования, но и ведущий к нему путь должен быть истинным“

К. Маркс

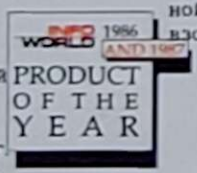






Именно эти слова К. Маркса приводит С. Эйзенштейн в одном из своих фильмов. Вот с такой точки зрения мы и предлагаем Вам, вдумчивому пользователю персонального компьютера, взглянуть на повышение производительности при использовании нескольких прикладных систем, взаимодействующих друг с другом и с Вами.

Премии



"Лучший программный продукт года" – премия, присужденная журналом InfoWorld в 1986 и 1987 гг. по результатам опроса читателей.

"Лучшая операционная среда" – премия, присужденная в 1986 и 1987 гг. по результатам опроса посетителей (свыше 80 000 чел.) выставки Comdex на конкурсе разработчиков программного обеспечения, проводимом журналом PC Tech Journal.

Наиболее полезным программным продуктом 1987 г. назвал ее Jerry Pournelle в журнале Byte.

"Лучшая альтернатива OS/2" – премия, присужденная в 1987 г. редакцией журнала PC Magazine.

В апреле 1988 г. система DESQview 2.0 на испытаниях, проведенных Национальной лабораторией тестирования программного обеспечения, созданной журналом PC World, превзошла по своим характеристикам систему Windows 386.

При испытаниях, проведенных журналом InfoWorld, система получила оценку 9,1 балла из 10.

Преимущества

Возможность параллельного выполнения нескольких программ и быстрого переключения с одной из них на любую из остальных (в том числе на Windows 2.0, Ventura Publisher, AutoCAD, 1-2-3, dBASEIII).

Возможность параллельного выполнения программ при использовании стандартной памяти емкостью 640К или расширяемой (EMS 4.0) памяти.

Возможность взаимодействия с каждой из параллельно выполняемых программ посредством окон на экране дисплея.

Возможность легкого и быстрого обмена информацией между параллельно выполняемыми программами.

Облегчение взаимодействия с ОС MC-DOS за счет применения меню и подсказок.

Возможность набора телефонных номеров.

Упрощение использования различных программ за счет встроенных макросредств, обеспечивающих произвольное программирование клавиш.

Возможность использования интерфейса API (Application Program Interface), в который включен ряд средств, присущих OS/2: порождение подзадач, межзадачные коммуникации, разделяемые программы и т. д.

И все это при использовании стандартной клавиатуры или манипулятора "мышь"

Истина

Наш голос не самый громкий среди голосов, обещающих повышение производительности. Но мы уверены, что объективный анализ фактов покажет то, что хорошо известно нам:

Нет более простого и быстрого пути повысить производительность, чем воспользоваться системой DESQview 2.0.

ТРЕБОВАНИЯ, ПРЕДЪЯВЛЯЕМЫЕ СИСТЕМОЙ DESQview

Персональный компьютер фирмы IBM или любой другой 100%-но совместимый с ним персональный компьютер (на основе любого из следующих микропроцессоров фирмы Intel: 8086, 8088, 80286, 80386) с монохромным или цветным дисплеем или персональный компьютер семейства PS/2. Память: 640 Кб (рекомендуется); сама система DESQview занимает 145 Кб памяти. Расширенная память (дополнительно): либо плата памяти, совместимая с AboveBoard, либо усовершенствованная плата памяти, совместимая с AST RAMpage, либо плата памяти EMS 4.0. НМД: либо два НГМД, либо один НГМД и один НЖМД. Плата видеоадаптера (дополнительно): Hercules, CGA, EGA, VGA. Манипулятор "мышь" (дополнительно): совместимый с манипулятором "мышь" фирмы Microsoft. Модем (дополнительно): совместимый с модемами фирмы Hayes. Операционная система: PC-DOS 2.0 – 3.3 или MS-DOS 2.0 – 3.2. Программное обеспечение: большинство программ для операционных систем PC-DOS и MS-DOS, а также программы для систем Windows 1.03 – 2.03, GEM 1.1 – 3.0, IBM TopView 1.1. Носитель: система DESQview поставляется на гибких магнитных дисках (диаметром 5 1/4" или 3 1/2").

Торговые марки принадлежат соответствующим владельцам: IBM, OS/2, PS/2, 1-2-3, Windows, dBASE III, Ventura Publisher, AutoCAD, MS-DOS

За подробной информацией о системе DESQview и других изделиях фирмы Quarterdeck, пожалуйста, обращайтесь по адресу:

Quarterdeck

150 Pico Boulevard, Santa Monica, CA 90405
(213) 392-9851

2'88 PC WORLD-USSR

В МИРЕ ПЕРСОНАЛЬНЫХ КОМПЬЮТЕРОВ

IDG COMMUNICATION «РАДИО И СВЯЗЬ»

В номере

В $\frac{2404000000\text{-}178}{046\,(01)\text{-}88}$ КБ-29-17-88

От издательства

ДОРОГИЕ ЧИТАТЕЛИ!

Перед вами второй выпуск сборника "В мире персональных компьютеров". В него вошли статьи, ориентированные на читателей, имеющих различный уровень подготовки в области программирования и вычислительной техники.

Сборник открывается подборкой статей, освещающих перспективы развития технического и программного обеспечения персональных компьютеров. Подготовленный А. И. Масаловичем обзор докладов Всесоюзной школы-семинара "ПЭВМ-88" дает представление о производстве и применении персональных компьютеров в нашей стране. В статье Э. Бендера и К. Гринберга описывается микропроцессор 80386 фирмы Intel, появление которого создало условия для выпуска 32-разрядных персональных компьютеров, совместимых с ПК AT. Этот же микропроцессор используется в самых мощных персональных компьютерах нового семейства PS/2 фирмы IBM. Особенности семейства компьютеров PS/2 и связанные с ним планы фирмы IBM обсуждаются в статьях этого же раздела.

Значительное место отведено информации, представляющей интерес для программирующих на языке Бейсик. В статье Д. Уилльямза о компиляторе Quick Basic фирмы Microsoft показано, что Бейсик может успешно конкурировать с другими языками программирования, причем процесс разработки программ на нем уже сейчас оснащен эффективными средствами отладки на уровне исходного текста программы и редактором текста, проверяющим синтаксис в процессе ввода программы, а предварительная компиляция сокращает время компиляции настолько, что оно становится практически незаметным. Несколько небольших заметок поможет программистам решить задачи, часто возникающие на практике. Не забыты и проблемы, связанные с интерпретаторами Бейсика. Так, в статье Г. Перри рассказывается, как добиться повышения быстродействия интерпретируемых программ. В статье Р. Олфорда объясняется, от чего зависит быстродействие самого компьютера.

Для тех, кто еще не имеет достаточного опыта в эксплуатации своего компьютера, может оказаться полезной статья об обмене данными между ПК AT и XT, в которой подробно рассматриваются проблемы использования дисководов для двух типов дискет.

Для более опытных программистов предназначен материал раздела "Программирование на машинном уровне". В статье А. В. Козлова приведен текст программы драйвера клавиатуры и видеоадаптера персонального компьютера, решающей проблему включения в набор используемых символов букв русского алфавита. Статья Х. Бродерза учит писать программы на языке ассемблера, позволяющие обмениваться сообщениями.

В разделе "Вокруг DOS" можно выделить статью Д. Рауэлла, в которой приведены пакетные файлы, помогающие автоматизировать многие процедуры, многократно выполняемые программистами, а также заметку И. А. Настенко о его программе "DOS-интегратор", облегчающей жизнь начинающим программистам и ведущей с ними диалог от имени DOS на русском языке.

Попытки стандартизации языка запросов баз данных для персональных компьютеров сделаны всеми ведущими фирмами. О таких попытках, связанных с использованием языка SQL, и о существующих проблемах в этой области рассказывается в статье С. Мейса. Интересен опыт использования СУБД DataFlex фирмы Data Access, которым делятся на страницах нашего сборника Б. К. Норкин и С. К. Сомов.

Часть материалов, публикуемых в сборнике, посвящена научно-инженерным приложениям персональных компьютеров.

Одной из самых интригующих тем исследований в настоящее время является тема искусственного интеллекта. Не избежали искушения и составители настоящего сборника. Автор статьи "Будущее Пролога" Р. Льюис предсказывает Прологу блестящее будущее как языку для разработки экспертных систем. Однако пока Пролог борется со своим конкурентом – языком Лисп, А. Р. Кулмагамбетов и Л. Т. Ямпольский доказали, что экспертную систему можно разработать и с использованием Бейсика (см. статью "Экспертная психодиагностическая система").

В будущем году намечается выпуск четырех таких сборников. В них предполагается достаточно подробно рассмотреть проблемы использования локальных сетей на персональных компьютерах с описанием необходимой аппаратуры и программного обеспечения. Будет также продолжена публикация сообщений, предназначенных для оказания практической помощи программистам с разным уровнем подготовки и для информирования профессионалов о новостях в мире персональных компьютеров.

Напоминаем, что сборники формируются на основе материалов из журналов, выпускаемых фирмой IDG Communications (США), и статей советских авторов, посвященных проблемам применения персональных компьютеров в различных областях знаний и в быту.

Присылайте ваши статьи и сообщения! Тематика и стиль изложения материала станут понятными после того, как вы прочтете настоящий сборник. Следуйте образцам, которые вам покажутся лучшими.

Добро пожаловать в мир информационной технологии

илучшие пожелания читателям сборника "В мире
ксональных компьютеров"! Мы верим, что Вы быстро и
фективно улучшите Ваши возможности по
именению компьютеров, присоединившись к
мирной аудитории этого важного издания IDG.

24 года International Data Corp., отдел исследований
аркетинга IDG стал признанным лидером, прекрасно
бирающимся в вопросах применения, маркетинга, стра-
ического и тактического планирования для осущест-
ения коммерческой и административной деятельности
азвития индустрии информационной технологии. Наши
деления, обеспечивающие информационной технологией
ь мир, находятся в 22 крупнейших странах.

ециализируясь в определении тенденций развития
ынка и размещения новой продукции, IDC предлагает
нкретные ценные услуги клиентам, стремящимся
йти на новые рынки. Если Вы хотите узнать об этих
слугах подробнее, пожалуйста, обращайтесь к нам.

Best wishes to you, the readers of PC World U.S.S.R. We believe you will accelerate and enhance your ability to use computers beneficially by joining the global audience of this important International Data Group publication.

For 24 years, International Data Group's research and marketing division, International Data Corporation, has been the acknowledged leader in providing invaluable insights and guidance on applications, markets, strategies, and tactical planning to business enterprises, governments, and the information technology industry. We cover the world of information technology with offices in 22 major countries.

IDC, which specializes in emerging market trends and new product positioning, offers particularly valuable services to clients with global objectives. Should you wish to know more about these services, please contact us.

Telefax: 626-4205; Telex: 95-1168; Telephone: (617) 872-8200. International Data Corporation, 5 Speen Street, Framingham, MA USA 01701.

Состояние и перспективы развития профессиональных ПК

(ПО МАТЕРИАЛАМ ШКОЛЫ-СЕМИНАРА "ПЭВМ-88")

А. И. МАСАЛОВИЧ

I Всесоюзная школа-семинар "Разработка и внедрение в народное хозяйство персональных ЭВМ" ("ПЭВМ-88"), проходившая 17 – 20 мая этого года в г. Минске, явилась событием во многих отношениях исключительным. Во-первых, она собрала беспрецедентное количество участников. Было представлено более 400 докладов, тезисы которых заняли пять объемистых томов. В работе школы приняло участие более 1000 человек. Во-вторых, одновременно с открытием школы-семинара в Минске открылись две выставки – выставка ПК, выпускаемых в СССР, и выставка программного обеспечения для ПК. Наконец, школа-семинар "ПЭВМ-88" интересна тем, что это первая в стране конференция, целиком посвященная проблеме персональных компьютеров.

По оценкам некоторых специалистов, общее число ПК в мире составляло в конце 1986 г. 35 млн штук. Охвативший мир бум ПК не обошел и нашу страну. План текущей пятилетки предусматривает выпуск к 1990 г. 1,1 млн ПК различного назначения. Много это или мало? Прогноз развития ЭВМ массового применения, сделанный Межотраслевым научно-техническим комплексом "Персональные ЭВМ" (МНТК "ПЭВМ"), оценивает потребность народного хозяйства СССР в ПК в 28 млн штук (без учета бытовых компьютеров), из них для организации центров обучения и досуга требуется 16,5 млн, для административно-управленческих работ – 9,5 млн, для решения задач автоматизации проектирования и научных исследований – 2 млн. Как видно из приведенных цифр, перед советской индустрией микрокомпьютеров стоят большие и сложные задачи.

Анализ различных аспектов поставленных задач и связанных с ними научных проблем, поиск путей их решения явились предметом обсуждения на девяти секциях школы-семинара "ПЭВМ-88".

ТЕХНИЧЕСКИЕ СРЕДСТВА ПК

В настоящее время в стране выпускается более двух десятков типов ПК. Большая часть их была представлена на выставке, развернутой в дни работы школы-семинара. Характеристики представленных на выставке ПК приведены в таблице. Из таблицы видно, что наиболее привлекательными для профессионального применения характеристиками обладает ПК ЕС 1841.

Три с половиной года назад были начаты работы по созданию профессиональных ПК Единой системы, совместимых с ПК фирмы IBM. В ПК ЕС 1840 и ЕС 1841 использован микропроцессор К1810ВМ86, обладающий возможностями известного процессора Intel 8086. Перспективная модель ЕС 1842, разработка которой близится к завершению, включает в себя модернизированный вариант этого процессора К1810ВМ86М с повышенной тактовой частотой. Использование контроллера виртуальной памяти и программной эмуляции команд процессора Intel 80286 позволяет достичь программной совместимости ПК ЕС 1842 и IBM PC/AT.

Ведутся работы по созданию 32-разрядного ПК Единой системы. Он будет базироваться на комплекте БИС с числом выводов 128 – 172, изготавливаемых по КМОП-технологии с 2-мкм технологическими нормами. Ожидаемая производительность на операциях типа "регистр – регистр" 5 млн операций/с.

Важной проблемой при разработке ПК является создание надежных и эффективных периферийных устройств. В настоящее время промышленностью освоены дисковые накопители типа "Винчестер" емкостью 5, 10 и 20 Мбайт с диаметром диска 133 мм. Средняя наработка на отказ составляет 8 – 10 тыс. ч, однако необходимы работы по ее увеличению, поскольку в мировой практике все шире используются накопители с наработкой на отказ 20 – 30 тыс. ч.

На начальном этапе находятся работы по созданию малогабаритных оптических дисковых ЗУ. Одной из причин, сдерживающих развитие оптических дисков, является отсутствие отечественных полупроводниковых лазеров с требуемыми параметрами.

СИСТЕМНОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

Персональные компьютеры ЕС 1840 в настоящее время поставляются с операционной системой М86 (основанной на принципах популярной ОС СР/М). Разработаны также ОС МИКРОС для ПК ЕС 1841, обеспечивающая совместимость с СМ-1800, и АЛЬФА-ДОС, обеспечивающая совместимость с MS-DOS. Завершаются работы по созданию UNIX-подобных операционных систем для ПК ЕС 1842. Наряду с операционными системами, аналогичными широко распространенным в мире, есть и оригинальные разработки. Особое место среди них за-

Выпускаемые в СССР ПК, представленные на выставке

(По данным, приведенным в книге "Разработка и внедрение в народное хозяйство персональных ЭВМ". Тезисы докладов I Всесоюзной школы-семинара. Минск, 1988)

Название	Процессор	Оперативная память, Кбайт	Дисплей	Внешняя память	Примечание
ЕС 1840.05	К1810ВМ86	128 – 640	Монохромный графический, 640 X 200 точек, 25 X 80 символов	Два НГМД по 360 Кбайт, 133 мм	
ЕС 1841	К1810ВМ86	512 – 2048	Монохромный графический, 16 градаций яркости или цветной, 640 X 200 точек, 25 X 80 символов	Два НГМД по 360 Кбайт; НМД на 10 или 20 Мбайт	Имеется манипулятор "мышь"
Микроша	КР580ВМ80А	32	Бытовой черно-белый телевизор, 25 X 64 символа	Бытовой кассетный магнитофон	
Электроника КР-01	К580ВМ80А	16	То же	То же	
Гранат	КР580ВМ80А	62	Бытовой черно-белый или цветной телевизор. Цветной режим: 4 цвета, 224 X 240 точек, 25 X 40 символов. Монохромный режим: 224 X 480 точек, 25 X 80 символов	–"–	
Немига (МРТИ)	К588/К1824	128	Электроника МС 6105.01, 4 цвета, 432 X 256 точек	ГМД-6022, ГМД-7012	ПК объединяются в локальную сеть
Агат-9	6502	64 – 640	На базе телевизора "Юность Ц-410", 512 X 256 точек, 32 X 64 символа	НГМД 133 мм	
Микро-16	К1810ВМ86	128 – 640	Бытовой черно-белый или цветной телевизор. Цветной режим: 4 цвета, 320 X 200 точек, 25 X 40 символов. Монохромный режим: 640 X 200 точек, 25 X 80 символов	Бытовой кассетный магнитофон, НГМД	
Львов ПК-01	К580ВМ80	64	Бытовой цветной телевизор, 4 цвета, 256 X 256 точек, 32 X 24 символа	Бытовой кассетный магнитофон	
Сура	К580ВМ80А	64	Бытовой черно-белый телевизор, 24 X (32 – 40) символов	То же	Ориентировочная цена 600 р.
Вектор 06-Ц	КР580ВМ80А	64	Бытовой черно-белый или цветной телевизор, 16 цветов – 256 X256 точек, 4 цвета – 512 X 256 точек, 80 X 25 символов	–"–	То же
Апогей БК-01	К580ВМ80А	64	Бытовой черно-белый телевизор, 25 X 64 символа	–"–	

Название	Процессор	Оперативная память, Кбайт	Дисплей	Внешняя память	Примечание
Ассистент	К1810ВМ86	128	Бытовой цветной телевизор, 16 цветов – 320 X 200 точек, 40 X 25 символов, 4 цвета – 640 X 200 точек, 80 X 20 символов	Бытовой кассетный магнитофон	
Нейрон	К1810ВМ86	256 – 1024	Монохромный графический, 640 X 200 точек, 80 X 25 символов	Два НГМД по 360 Кбайт, НМД на 5 Мбайт	
Немига	К1810ВМ86	128 – 512	Монохромный графический дисплей или бытовой телевизор, 640 X 200 точек, 25 X 80 символов	НГМД на 640 Кбайт	Имеется джойстик

нимает операционная система ОНИКС, разработанная ВЦ СОАН СССР для поддержки локальных сетей рабочих станций МРАМОР. Операционная система ОНИКС обеспечивает одновременную работу всех входящих в состав рабочей станции процессоров в многозадачном режиме.

ПРИКЛАДНОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

Пользователя ПК на первых порах ошеломляет обилие сервисного программного обеспечения, предлагаемого десятками зарубежных фирм. Представляется весьма трудным разобраться в этом потоке, а тем более сделать что-либо конкурентоспособное. Между тем опыт показывает, что отечественные программные продукты обладают некоторыми весьма привлекательными для начинающего программиста свойствами. Во-первых, они ведут диалог с пользователем на русском языке. Во-вторых, это в основном открытые системы, допускающие расширение и модификацию. В-третьих, существуют организации (например, СНПО "Алгоритм"), осуществляющие распространение программного обеспечения и обучение пользователей. Разработка базисного программного обеспечения (БПО) для ПК регламентируется ГОСТ 27201–87. Согласно стандарту в БПО входят операционные системы, системы программирования на базе языков Фортран, Паскаль, Си, Бейсик, а также пакеты программ обработки текстов, обработки таблиц, поддержки графики, СУБД и программы обмена файлами. К настоящему времени для ПК Единой системы и совместимых с ними разработаны следующие базовые пакеты прикладных программ:

АБАК – пакет обработки электронных таблиц;
ТЕЛЕТЕКСТ – пакет связи с ЕС ЭВМ;
КОМЕТА – пакет поддержки работы в системе телеобработки ЕС7920;
ТЕКСТМ86 – электронная записная книжка;
СЛОГ – пакет обработки текстовой информации;
ДЕЛОГРАФ – пакет обработки деловой графики;
РЕПЕР – система управления реляционной базой данных;
МИКРОПРИЗ – пакет автоматической генерации программ по спецификациям проблемы, заданным пользователем;
СТАТС – пакет обработки статистической информации.

ВЫЧИСЛИТЕЛЬНЫЕ СЕТИ НА БАЗЕ ПЭВМ

Судя по тому, какому придирчивому анализу подверглись на школе-семинаре распространенные пакеты сетевого взаимодействия (такие, как Kermit, Crosstalk, EasyLAN и др.), локальные сети начинают играть все более важную роль в практике использования ПК.

К настоящему моменту накоплен опыт организации локальных сетей практически для всех основных классов отечественных ПК. Созданы локальные сети ПК "Немига", ДВК, "Электроника-85", "Электроника БК0010" и др. Завершается разработка пакета программ CP/M-Link, поддерживающего взаимодействие в локальной сети 8-разрядных ПК с ОС CP/M (или SCP для ПК "Роботрон 1715").

Наибольший интерес пользователей, вероятно, вызывают варианты обмена файлами между ПК и ЕС ЭВМ. На школе-семинаре были представлены две системы передачи файлов между ЕС ЭВМ и ПК ЕС 1840/ЕС 1841. Первая из представленных систем обеспечивает обмен через стык С2 (пакет ТЕЛЕТЕКСТ), вторая – через стык ЕС7920 (пакет КОМЕТА-ОС). Со стороны ЕС ЭВМ в состав пакета входят программы управления обменом, управления сообщениями, передачи наборов данных ЕС ЭВМ и приема файлов ПК. Данные для обмена между ЕС ЭВМ и ПК могут быть представлены как в текстовом, так и в двоичном форматах.

АСУ И АСУП НА БАЗЕ ПК

Отличительной особенностью задач АСУП является большой объем входных и справочных данных, имеющих сложную и изменяющуюся структуру. Поэтому использование ПК в АСУП оправданно при наличии на ПК дисковых накопителей достаточной емкости и средств управления базой данных. Приемлемым представляется также включение ПК в состав неоднородной локальной сети, объединяющей как ПК, так и большие ЭВМ. Однако наибольшее распространение в производстве ПК получили как основа автоматизированных комплексов управления отдельными технологическими процессами. Прежде всего это относится к производству изделий электронно-вычислительной техники и, в частности, к производству самих ПК. Персональные компьютеры здесь используются как основа измерительных систем, систем аппаратной отладки БИС, комплексов группового управления оборудованием сборки и контроля.

На предприятиях, раньше других начавших использование ПК в управлении производством, наблюдается интеграция средств АСУ различных уровней. В качестве примера можно привести интегрированные АСУ Белорусским производственным объединением вычислительной техники и информатики (ИАСУ БПО ВТИ) г. Минска. В состав ИАСУ входят: АСУ объединения, АСУ предприятия (завода), АСУ цеха. Ориентация на использование специализированных рабочих мест на базе ПК в ИАСУ БПО ВТИ для автоматизации функций управленческого и диспетчерского персонала обеспечивает ряд преимуществ перед традиционными АСУ.

ОБУЧАЮЩИЕ ПРОГРАММЫ

Такие преимущества ПК, как развитые графические средства, высокая надежность, обилие сервисных программных средств, простота освоения, обусловливают возрастающую потребность в использовании их в учебном процессе и создании автоматизированных обучающих систем (АОС) на базе ПК.

Персональные компьютеры открывают принципиально новые возможности для использования активных методов обучения. В одном из докладов школы-семинара было предложено 12 различных ролей ПК в процессе обучения: "Информатор", "Демонстратор", "Консультант", "Задачник", "Конструктор", "Соперник" и т. д. Многообразие доступных форм взаимодействия требует разработки общих методологических принципов построения АОС на базе ПК. Такие работы ведутся, например, в Московском энергетическом институте, где параллельно с исследованием социально-психологических аспектов эффективности АОС разрабатывается обобщенная концепция компьютерного обучения.

Тем, кого интересуют вопросы программирования на языке ассемблера для ПК IBM PC и XT и команды системы автоматизированного проектирования AutoCAD, издательство "Радио и связь" предложит в 1989 г. переводы с английского языка этих книг известных американских специалистов.

В то время как ведется множество дискуссий по поводу форм использования ПК в АОС, некоторые организации уже разработали и используют в учебном процессе готовые АОС. Большая их часть относится к реализации на ПК тех или иных учебных дисциплин. Так, в Ленинградском политехническом институте создан учебно-лабораторный комплекс "Организация ЭВМ, микропроцессоров и микропроцессорных систем". Однако в целом диапазон представленных на школе-семинаре АОС был достаточно широк — от технического средства обучения "Дошкольник", разработанного в Московском инженерно-физическом институте, до обучающего курса по основам работы с СУБД dBASE III Plus, предложенного группой авторов из Казани.

САПР И АРМ НА БАЗЕ ПК

"Используя ПК, вы можете в значительной степени опередить время". Во время проведения школы-семинара эта фраза постоянно была перед глазами участников — на обложке сборника тезисов и программы семинара, на информационных листах и рекламных пирамидах. Ставшая девизом школы-семинара, она, пожалуй, более всего относится к системам автоматизации проектирования. Именно в САПР использование ПК может дать наибольший экономический эффект, поскольку позволит повысить производительность труда разработчиков в 5 — 10 раз. Поэтому разработанная МНТК "ПЭВМ" "Концепция персональных ЭВМ новых поколений" предусматривает первоочередное удовлетворение потребностей САПР и АСНИ в профессиональных ПК.

Уже сейчас существует большое число программных средств, позволяющих использовать ПК в САПР: интерактивная графическая система проектирования печатных плат МАГИСТР-П, рабочая станция МРАМОР, диалоговая система конструирования программ "Ассистент" и др. Вместе с тем задачи создания эффективных средств проектирования современных БИС и СБИС на ПК, а также кремниевых компиляторов на ПК еще в полной мере не решены. Много новых проблем порождает включение ПК в структуру сложившихся САПР, поскольку соединение концептуально-разнородного программного обеспечения вызывает реакцию "отторжения". Кроме того, чтобы полностью использовать возможности ПК в САПР, необходимо построение систем проектирования на базе неоднородной локальной сети, верхний уровень которой составляют высокопроизводительные ЭВМ с размещенной на них базой данных сквозного иерархического проектирования.

ПК И ОБЩЕСТВО

В программе школы-семинара не было специальной секции, посвященной социально-экономическим аспектам распространения ПК. Однако эти вопросы обсуждались как на пленарных, так и на секционных заседаниях. Широкое внедрение в жизнь ПК является составной частью процесса информатизации общества — особого социально-экономического процесса, сопоставимого по значению с индустриализацией. Конечно, сами по себе ПК не могут решить проблему информатизации страны. Необходимы усилия по созданию глобальных вычислительных сетей, интегрированных баз данных по основным областям науки и техники, широкая разработка экспертных систем и баз знаний. Однако именно ПК позволят сделать реальностью те преимущества, которые несет с собой развитие информационных технологий.

ПОЗДРАВЛЯЕМ!!!

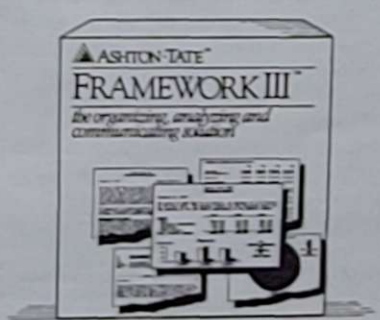
ASHTON·TATE

ВЕДУЩИЙ ПОСТАВЩИК ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ МИКРОКОМПЬЮТЕРОВ С УДОВОЛЬСТВИЕМ ПОЗДРАВЛЯЕТ ИЗДАТЕЛЕЙ ЖУРНАЛА "В МИРЕ ПЕРСОНАЛЬНЫХ КОМПЬЮТЕРОВ" В СССР И СОВЕТСКИХ ЧИТАТЕЛЕЙ С ВЫХОДОМ В СВЕТ ПЕРВОГО НОМЕРА ЭТОГО ЖУРНАЛА.

ФИРМА ASHTON-TATE ПРЕДЛАГАЕТ СЛЕДУЮЩИЕ ПРОГРАММНЫЕ ПРОДУКТЫ:

dBASE II
dBASE III PLUS
FRAMEWORK II
dBASE III PLUS LAN PACK
MULTIMATE
MULTIMATE ADVANTAGE
BYLINE
MULTIMATE GRAPHLINK
MAP-MASTER
DIAGRAM-MASTER

dBASE III
dBRUN
FRAMEWORK III™
RAPIDFILE
dBASE Mac
MULTIMATE ADVANTAGE II
MULTIMATE ON-FILE
CHART-MASTER
SIGN-MASTER



ИНТЕРКВАДРО + *UTIX* = *QUADRIX*

А. С. КАПЛАН, В. В. ЛЕОНАС, И. Г. ШЕСТАКОВ

UNIX — вот, что написано большими буквами на знамени ИНТЕРКВАДРО — первого совместного советско-франко-итальянского предприятия в области вычислительной техники и программирования. "Дитя перестройки" — ИНТЕРКВАДРО — сохранило в себе гены своих мощных и симпатичных родителей, которыми стали крупный потребитель вычислительной техники — Госагропром СССР, профессиональный научный исследователь — Госкомобразования СССР, известный разработчик и производитель средств вычислительной техники — французская фирма Aniral UTEC, активный коммерческий партнер — итальянская фирма Delta Trading. ИНТЕРКВАДРО является технико-коммерческим предприятием, ориентированным на реализацию комплексных проектов — различных автоматизированных систем (например, таких, как АСУ-офис, САПР, АСНИ, АСУТП и т. д.), сдаваемых заказчику "под ключ".

Когда речь заходит о создании крупной автоматизированной системы, это автоматически означает необходимость обеспечения контролируемого доступа к централизованной базе данных многим параллельно работающим пользователям такой системы. Кроме того, для крупных проектов особое значение приобретают вопросы совместимости программного обеспечения и дальнейшего совершенствования и развития создаваемых в рамках таких проектов аппаратно-программных комплексов. Наиболее перспективным путем преодоления препятствий такого рода являются создание и использование мобильного программного обеспечения и в первую очередь мобильных ОС, применение которых уменьшает зависимость создателя проекта и разрабатываемого в рамках проекта программного обеспечения от постоянно обновляющихся технических средств. Осознавая все это, ИНТЕРКВАДРО считает, что для фирм, ведущих работы в области системотехники, сейчас нет другой альтернативы, кроме освоения и использования ОС UNIX (или подобных ей ОС) или создания собственной ОС такого типа, наличие которой имеет большое значение не только с коммерческой, но и с научной точки зрения, определяя кредо предприятия.

Одним из вопросов, не нашедших еще однозначного ответа, является вопрос о том, как впишется ОС UNIX в многочисленные проекты, выполняемые с использованием персональных компьютеров: будет ли использование ОС UNIX ограничено файл-серверами или же переход к использованию нового поколения персональных компьютеров (построенных на основе 32-разрядных микропроцессоров, таких, например, как семейство МС 680xx фирмы Motorola и 80386 фирмы Intel) приведет к господству ОС UNIX и в мире персональных компьютеров? А может быть, все будет и иначе: кто может это сейчас надежно предсказать? Жизнь покажет, а ждать этого, видимо, осталось недолго: наступление ОС UNIX продолжается семимильными шагами, и те, кто сегодня фанатично убеждены, что персональные компьютеры — "это все, что им необходимо", скоро будут вынуждены учиться плавать в половодье реки по имени UNIX.

Лев И. Вайнберг

Генеральный директор

Персональные компьютеры, первые образцы которых начали появляться в середине 70-х гг., принесли с собой новую идеологию использования средств вычислительной техники, которая оказалась настолько привлекательной, что к концу 1987 г. ПК составляли уже около 90% общего объема мирового парка компьютеров. Даже самый поверхностный анализ современного парка ПК показывает, что львиную долю его составляют различные модели ПК типа IBM PC, используемые с операционной системой (ОС) MS DOS.

Столь широкое распространение ПК является следствием в первую очередь двух обстоятельств – новой идеологии, которую они принесли с собой, и их доступности, порожденной этой новой идеологией и невысокой ценой ПК. Быстро возраставшие возможности ПК породили иллюзию того, что автономный ПК представляет собой "панацею от всех бед", т. е. с его помощью можно решить практически любые задачи. Однако, как известно, "потребности всегда растут быстрее, чем возможности", и ПК не стали исключением из этого правила, поскольку вскоре стало очевидно, что некоторые задачи им "не по зубам". Строго говоря, это было очевидно с самого начала – опыт показывает, что широчайшее распространение личных автомобилей не привело к исчезновению автобусов, грузовиков и прочих видов автомашин, просто каждый из видов транспорта используется для различных целей. То же самое имеет место и в мире компьютеров – каждый из типов компьютеров имеет свою "экологическую нишу", определяемую классом задач, для решения которых он наилучшим образом приспособлен. Поэтому неудивительно, что, быстро освоив возможности, предоставляемые ПК, пользователи натолкнулись на ограничения, порожденные самой природой ПК. Одним из основных источников этих ограничений стала именно их "персональность", создающая впечатление, что для успешного использования ПК достаточно обеспечить пользователю возможность работы в монопольном однопрограммном режиме. Основанная на концепции монопольной однопрограммности ОС MS DOS в настоящее время вызывает у пользователей большее неудовольствие, чем, например, недостаточно высокая вычислительная мощность ПК.

Одной из очень часто встречающихся задач, решение которой крайне затруднено при использовании ПК, работающих в монопольном однопрограммном режиме, является задача разделения информации. Такая задача возникает, например, при необходимости автоматизации учрежденческой деятельности в любой достаточно крупной организации. В таких случаях, как правило, оказывается необходимым использование компьютерных сетей и централизованных баз данных. Наиболее удобным средством для организации централизованных баз данных в компьютерных сетях является файл-сервер. Он представляет собой узел компьютерной сети, в функции которого входят хранение больших объемов информации и обеспечение доступа к ней из остальных узлов этой сети. Компьютеры, используемые для создания на их основе файл-серверов, должны обладать достаточными информационными и вычислительными ресурсами. Наиболее просто и эффективно функции файл-сервера реализуются при использовании компьютера, работающего под управлением многопользовательской многопрограммной ОС разделения времени. В качестве такой ОС чаще всего используют ОС UNIX или подобные ей операционные системы.

Популярность ОС UNIX и подобных ей ОС общеизвестна. Поэтому в настоящее время любой уважающий себя производитель средств вычислительной техники считает своим долгом предложить одну из версий ОС UNIX или подобной ей ОС в качестве если не основной, то уж, по крайней мере, альтернативной ОС для производимых им компьютеров. Даже фирма IBM, в течение длительного времени не признававшая идеологию ОС UNIX, предлагает теперь подобные ей ОС: ОС PCIX – пользователям своих ПК типа XT и AT и ОС AIX – пользователям ПК модели 80 из семейства PS/2.

Новое поколение ПК, к которому, в частности, относится и ПК модели 80 семейства PS/2, строится на основе 32-разрядных микропроцессоров. Среди выпускаемых серийно 32-разрядных микропроцессоров наибольшую известность приобрели микропроцессор 80386 фирмы Intel и семейство микропроцессоров MC 680xx фирмы Motorola. Микропроцессор 80386 используется в качестве основы для старших моделей ПК семейства PS/2, выпускаемого фирмой IBM, а микропроцессоры семейства MC 680xx используются большим числом различных фирм, производящих компьютеры, среди которых можно, например, назвать такие американские фирмы, как Hewlett-Packard, Apollo, Sun Microsystems.

Семейство супермикрокомпьютеров UTEC 32, выпускаемое французской фирмой Aniral UTEC, также построено на основе использования микропроцессоров семейства MC 680xx. В это семейство супермикрокомпьютеров входят следующие модели, построенные на основе микропроцессора MC 68010: 32/15, 32/15U, 32/15T, PS 32, WS 32, PS 32 PRO. В конце 1988 г. к этому перечню добавится модель 32/30, построенная на основе микропроцессора MC 68030.

Основой супермикрокомпьютера UTEC 32/15 является микропроцессор MC 68010, работающий с тактовой частотой 10 МГц и обеспечивающий возможность адресации к ОЗУ объемом до 16 Мбайт. В качестве ВЗУ могут быть использованы НМД типа "Винчестер" объемом до 1, 2 Гбайт, НГМД объемом 1, 2 Мбайт, НМЛ катушечного типа (800/1600 бит на дюйм), НМЛ типа "Стример" объемом 60 Мбайт. К супермикрокомпьютеру UTEC 32/15 можно подключить до 16 ВУ (терминалов и/или принтеров), имеющих интерфейс RS 232. В качестве терминалов используются алфавитно-цифровые терминалы типа Wyse 75 (совместимые с терминалом DEC VT-100) или цветные графические терминалы типа Wyse 350.

На основе использования двухвходового ОЗУ фирма Aniral UTEC разработала две двухпроцессорные модели супермикрокомпьютера UTEC 32/15: 32/15U и 32/15T, последняя из которых ориентирована на работу в режиме реального времени. Использование двухпроцессорных комплексов, построенных на основе двух микропроцессоров MC 68010, позволяет существенно повысить производительность моделей 32/15U и 32/15T по сравнению с моделью 32/15.

Супермикрокомпьютер UTEC WS 32 представляет собой рабочую станцию, построенную на основе супермикрокомпьютера UTEC 32/15.

Супермикрокомпьютеры UTEC PS 32 и UTEC PS 32 PRO также построены на основе использования двухвходового ОЗУ и представляют собой двухпроцессорные комплексы. В качестве второго процессора в них используются микропроцессоры 80286 и 80386 соответственно.

Базовой операционной системой для супермикрокомпьютеров семейства UTEC 32 является многопользова-

тельская многопрограммная ОС разделения времени QUADRIX, разработанная в СП ИНТЕРКВАДРО на базе ОС UTIX и обеспечивающая совместимость с ОС UNIX версии 7. Кроме того, ОС QUADRIX дополнительно обеспечивает возможность использования букв кириллицы и совместимость с ОС ДЕМОС, версии которой имеются для различных типов компьютеров.

Основу ОС QUADRIX составляет ее ядро – резидентная часть операционной системы. Ядро ОС QUADRIX управляет процессами, памятью, устройствами и информацией. Интерфейс между программами пользователей и ядром ОС QUADRIX осуществляется с помощью системных вызовов. Будучи написано почти полностью на языке Си, ядро ОС QUADRIX включает в себя лишь небольшой набор подпрограмм, написанных на языке ассемблера (объем ассемблерных подпрограмм составляет около 10 % от общего объема ядра ОС QUADRIX).

Следующим по важности компонентом ОС QUADRIX является интерпретатор команд shell, обеспечивающий интерпретацию команд, вводимых пользователями с клавиатуры терминала, и предоставляющий пользователям возможность использования командного языка очень высокого уровня, операторами которого могут быть функции произвольной сложности, реализуемые программами или их совокупностями. Пользователям ОС UNIX и подобных ей операционных систем наверняка известны две получившие наиболее широкое распространение версии интерпретатора команд shell – так называемый "классический" интерпретатор команд shell Бурна (S. R. Bourne) и интерпретатор команд shell, разработанный в Калифорнийском университете г. Беркли. ОС QUADRIX предоставляет в распоряжение пользователя обе эти версии интерпретатора команд shell.

Одним из важнейших компонентов ОС QUADRIX является транслятор с языка Си, называемый иногда Си-компилятором. В ОС QUADRIX используется Си-компилятор, построенный по двухпроходной схеме и имеющий два дополнительных прохода – препроцессор и оптимизатор объектного кода. Помимо транслятора с языка Си в ОС QUADRIX имеются транслятор с языка Фортран 77 и интерпретатор языка Лисп, а также ассемблер. Пользователь ОС QUADRIX имеет также возможность использовать препроцессор Ратфор, обеспечивающий трансляцию программ, написанных на структурирован-

ном диалекте языка Фортран, в программы на стандартном языке Фортран.

Помимо вышеперечисленных средств в распоряжении пользователя ОС QUADRIX имеется около 200 команд, совместимых с соответствующими командами ОС UNIX. В частности, в ОС QUADRIX имеются инструментальные средства, облегчающие разработку трансляторов, создание и сопровождение больших программных комплексов, а также средства для обработки текстовой информации.

Среди наиболее существенных достоинств ОС QUADRIX необходимо назвать простоту и удобство интерфейса с пользователем, распространяющиеся на все уровни этого интерфейса: командный язык, стандартные библиотечные функции и системные вызовы. В совокупности с функционально широким набором команд и инструментальных средств, имеющихся в ОС QUADRIX, командный язык позволяет, с одной стороны, легко решать весьма сложные задачи, не прибегая к программированию, а лишь комбинируя уже имеющиеся команды и инструментальные средства, и, с другой стороны, легко создавать новые команды и инструментальные средства. Широкий по своим функциональным возможностям набор библиотечных функций с простым и удобным интерфейсом освобождает пользователя от необходимости самому реализовывать их, а компактный набор тщательно отобранных системных вызовов легко запоминается и весьма универсален.

Существенным достоинством ОС QUADRIX является ее совместимость с ОС UNIX, версии которой реализованы для большого числа компьютеров самых разных классов и типов — от микрокомпьютеров, построенных на основе 8-разрядного микропроцессора 8080 фирмы Intel, до суперкомпьютера Cray-2. Это обеспечивает высокую мобильность программ, разработанных для выполнения под управлением ОС QUADRIX.

Обеспечивая высокую мобильность прикладного программного обеспечения, предназначенного для выполнения под ее управлением, ОС QUADRIX сама обладает повышенной мобильностью. Повышение мобильности ОС QUADRIX является результатом использования при ее реализации нескольких приемов. Среди этих приемов необходимо отметить, во-первых, реализацию ОС практически полностью на машинно-независимом языке высокого уровня Си, а во-вторых, использование концепции максимального облегчения ядра ОС, достигаемого путем структуризации ОС и за счет отказа от внесения в ядро ОС ряда компонентов, традиционно в него входящих.

Нельзя не упомянуть еще об одном достоинстве ОС QUADRIX, заключающемся в легкости внесения в нее изменений с целью адаптации к конкретным условиям применения или к конкретным аппаратным средствам. В частности, при использовании ОС QUADRIX подключение нестандартных ВУ осуществляется намного проще, чем в других ОС, поскольку для написания драйверов можно использовать язык Си.

Во многих областях применения компьютеров на них возлагается решение задач, связанных с хранением больших объемов информации и обеспечением быстрого доступа к ней. Для решения такого рода задач чаще всего используются системы управления базами данных (СУБД). По этой причине в СП ИНТЕРКВАДРО была разработана СУБД реляционного типа QUADRES, работающая под управлением ОС QUADRIX. СУБД QUAD-

СЕМЕЙСТВО СОВМЕСТИМЫХ 32-РАЗРЯДНЫХ МИКРОПРОЦЕССОРОВ ФИРМЫ MOTOROLA

MC 68008 — 32-разрядный микропроцессор с 8-разрядной внешней шиной

MC 68000 — 32-разрядный микропроцессор с 16-разрядной внешней шиной

MC 68010 — 32-разрядный микропроцессор с 16-разрядной внешней шиной и аппаратной поддержкой виртуальной памяти

MC 68012 — 32-разрядный микропроцессор с 16-разрядной внешней шиной и аппаратной поддержкой расширенной виртуальной памяти

MC 68020 — 32-разрядный микропроцессор с 32-разрядной внешней шиной и аппаратной поддержкой виртуальной памяти

MC 68030 — 32-разрядный микропроцессор с 16-разрядной внешней шиной, аппаратной поддержкой виртуальной памяти и операций над полубайтами

РЕЗУЛЬТАТЫ СРАВНИТЕЛЬНОГО АНАЛИЗА ПРОИЗВОДИТЕЛЬНОСТИ СОВРЕМЕННЫХ МИКРОПРОЦЕССОРОВ

Таблица 1

Абсолютные времена выполнения тестовых программ, с

(Использовано динамическое ОЗУ с циклом 150 нс; N — без кэша, C — с кэшем)

Тип	f	E	F	H	I	K
80286	10	4.89	13.63	6.59	11.20	19.39
80386	16	3.57	5.16	3.63	6.86	6.20
68000	8	13.73	14.61	8.79	12.08	16.59
68020 N	16	8.02	5.55	3.84	5.65	4.78
68020 C	16	3.84	2.47	2.14	2.75	3.02
32032	10	12.52	13.07	6.21	8.57	13.07
32100 N	18	16.81	8.84	5.05	8.57	9.17
32100 C	18	6.75	4.29	2.74	3.63	4.45

Таблица 2

Абсолютные времена выполнения тестовых программ, с

(Использовано статическое ОЗУ с циклом 45 нс; N – без кэша, C – с кэшем)

Тип	f	E	F	H	I	K
80286	10	3.18	6.81	3.46	5.98	10.27
80386	16	1.92	2.53	1.53	3.18	3.68
68000	8	8.79	9.67	5.60	7.69	11.37
68020 N	16	3.74	2.74	1.87	2.69	2.64
68020 C	16	2.52	1.98	1.26	1.92	2.25
32032	10	11.04	10.05	4.83	6.59	11.65
32100 N	18	9.28	4.72	2.52	4.45	4.84
32100 C	18	6.04	3.07	1.81	2.97	3.46

Таблица 3

Средние значения времени выполнения, с, и дисперсии выборок по тестам, представленным в табл. 1 и 2

		Динамическое ОЗУ		Статическое ОЗУ	
Тип	f	M	D	M	D
80286	10	11.14	3.92	5.94	2.98
80386	16	5.08	1.79	2.57	1.29
68000	8	13.16	4.63	8.62	4.33
68020 N	16	5.57	1.96	2.74	1.38
68020 C	16	2.84	1.00	1.99	1.00
32032	10	10.69	3.76	8.83	4.44
32100 N	18	9.69	3.41	5.16	2.59
32100 C	18	4.37	1.54	3.47	1.74

E – поиск подстроки длиной 15 символов в строке длиной 120 символов, выполняемый 24 577 раз.

F – проверка, установка и сброс бита в строке длиной 128 бит, выполняемые 65 536 раз.

H – вставление пяти записей в пустой список, выполняемое 24 577 раз.

I – быстрая сортировка массива, состоящего из 100 записей, выполняемая 257 раз.

K – транспонирование битовой матрицы размером 7×7, выполняемое 20 481 раз.

f – тактовая частота микропроцессоров, МГц.

M – среднее значение времени выполнения тестов E, F, H, I, K, с.

D – дисперсия.

Тип – тип микропроцессора.

* По материалам статьи: Cooper T. C., et al. A Benchmark Comparison of 32-bit Microprocessors.// IEEE Micro. – August 1986. – p. 53 – 58.

RES совместима с получившей широкое распространение СУБД INGRES. Необходимость обеспечения совместимости СУБД QUADRES и INGRES обусловлена наличием значительного числа информационных систем, построенных на основе СУБД INGRES. Применение СУБД QUADRES обеспечит переход от использования автономных компьютеров к использованию распределенных многомашинных комплексов, построенных на базе компьютерных сетей.

Производимые фирмой Aniral UTEC супермикрокомпьютеры семейства UTEC 32, поставку, установку и сервисное обслуживание которых в СССР осуществляет СП ИНТЕРКВАДРО, могут быть использованы в самых различных областях. В настоящее время они успешно применяются в институтах и на предприятиях АН СССР, Госагропрома, Минавтопрома, Минвуза, Минлесбумпрома, Минморфлота, Минцветмета, Минчермета и Минхиммаша. Основными областями использования супермикрокомпьютеров семейства UTEC 32 являются:

- автоматизация учрежденческой деятельности (автономно или в качестве файл-сервера компьютерных сетей);
- автоматизация управления технологическими процессами;
- автоматизация научных (в том числе и экспериментальных) исследований;
- разработка программного обеспечения;
- подготовка текстовых документов;
- создание и эксплуатация информационных систем.

Хорошим примером, иллюстрирующим использование супермикрокомпьютеров семейства UTEC 32 для автоматизации экспериментальных исследований и управления технологическими процессами, может служить рабочая станция АСК-МИКРО, разработанная совместно Минвузом СССР и фирмой Aniral UTEC. Рабочая станция АСК-МИКРО представляет собой аппаратно-программный комплекс, предназначенный для автоматизации измерений и обработки их результатов, проведения контрольно-поверочных и наладочных работ, метрологических исследований, приемо-сдаточных и других видов испытаний. Рабочая станция АСК-МИКРО построена на основе супермикрокомпьютера семейства UTEC 32, работающего под управлением ОС QUADRIX, к которому помимо стандартных ВУ дополнительно подключены устройства отображения информации, модули связи с объектом и измерительные приборы. Подключение этих устройств осуществляется через интерфейс RS 232 или IEEE 488. Функционирование рабочей станции АСК-МИКРО поддерживается программным комплексом СМЭК, включающим в себя алгоритмический язык БАТЯ и пакет прикладных программ ИСАД. Программный комплекс СМЭК функционирует в среде ОС QUADRIX и обеспечивает возможность решения широкого класса задач в рассматриваемой области применения.

Если взглянуть на спектр задач, для решения которых уже применяются компьютеры, то легко заметить, что подавляющее большинство этих задач требует для своего решения работы не отдельных специалистов, а коллективов. В частности, это в полной мере относится к перечисленным выше областям применения супермикрокомпьютеров семейства UTEC 32. В этой ситуации неизмеримо возрастает роль информационных ресурсов и средств коммуникации. Имеющийся опыт создания и

использования компьютеризованных систем для решения подобных задач показывает, что необходимыми условиями их эффективности являются использование сетевых средств коммуникации, правильное распределение информации между локальными и централизованной базами данных и правильное распределение функций между компонентами системы. Необходимость использования (независимо от назначения компьютеризованной системы) идентичных средств для организации коммуникаций, локальных и централизованной баз данных вынуждает разработчиков такого рода систем базироваться на унифицированных аппаратно-программных средствах, обеспечивающих все необходимые виды сервиса. В качестве такого унифицированного средства СП ИНТЕРКВАДРО использует неоднородный многомашинный комплекс, построенный на базе локальной сети типа Ethernet. Особенностью такого многомашинного комплекса является ярко выраженная функциональная специализация входящих в него компьютеров. В частности, для организации централизованной базы данных используется файл-сервер на основе суперминикомпьютеров семейства UTEC 32, а в качестве рабочих мест в зависимости от решаемой задачи могут быть использованы либо обычные терминалы, либо ПК типа IBM PC/XT/AT, либо супермикрокомпьютеры семейства UTEC 32.

История создания ОС *UNIX*

Во второй половине 60-х гг. фирма Bell Laboratories была вовлечена в работы по проекту Multics, в которых принимали также участие фирма General Electric и Массачусетский технологический институт. В цели проекта Multics входило создание многопользовательской операционной системы разделения времени. В 1969 г. фирма Bell Laboratories вышла из проекта Multics и полностью прекратила все связанные с этим проектом работы. Наиболее активное участие в проекте Multics со стороны фирмы Bell Laboratories принимали К. Томпсон, Д. М. Ритчи, М. Д. МакИлрой, Дж. Ф. Осанна. Им не хотелось терять накопленные наработки, поэтому, несмотря на запрет руководства продолжать работы по созданию ОС, они начали искать различные способы использования полученных ими результатов. Например, ими рассматривались такие варианты, как разработка новой ОС для имевшегося в фирме Bell Laboratories компьютера GE-645 (для которого создавалась ОС Multics) и приобретение компьютера среднего класса (например, PDP-10 или Sigma-7) с целью разработки новой ОС для него. Параллельно с этим на имевшемся компьютере GE-645 осуществлялось моделирование файловой системы и схем управления памятью. Однако в силу ряда обстоятельств обе эти идеи были отвергнуты.

В это время К. Томпсон занимался разработкой программы Space Travel, моделировавшей движение основных небесных тел, входящих в состав солнечной системы. Каждый прогон этой программы на компьютере GE-645 обходился в 75 дол, что было непомерно дорого. Поэтому К. Томпсон принялся за поиски какого-либо другого компьютера, использование которого обходилось бы не так дорого. В ходе своих поисков он обнаружил практически никем не используемый мини-компьютер PDP-7 – предшественника широко известных мини-компьютеров семейства PDP-11. Тогда К. Томпсон начал перерабатывать свою программу Space Travel с тем, чтобы ее можно было использовать на мини-компьютере PDP-7. Однако оказалось, что переход к использованию мини-компьютера PDP-7 не так уж прост, поскольку имевшееся для него программное обеспечение состояло лишь из ассемблера и компоновщика. При этом необходимо было использовать перфоленту. Да и работать на этом мини-компьютере можно было лишь в монопольном режиме. Отсутствие программных средств, обеспечивающих удобство разработки, отладки и выполнения программ, натолкнуло К. Томпсона на мысль о создании соответствующих программных средств. Так на новом витке возникла идея разработки собственной операционной системы.

В 1969 г. появилась написанная на языке ассемблера однопользовательская ОС для мини-компьютера PDP-7, перенесенная через некоторое время на мини-компьютер PDP-9 (являвшийся также предшественником широко известного семейства мини-компьютеров PDP-11). Именно эта ОС и получила впоследствии название UNIX. Это название было предложено Б. У. Керниганом в 1970 г. и отражает противопоставление ОС Multics. Несмотря на такое противопоставление (простая и маленькая ОС UNIX – сложная и большая ОС Multics), ОС UNIX заимствовала определенные черты у ОС Multics. Влияние на ОС UNIX оказали также такие ОС, как CTSS и CMAS.

В 1970 г. по предложению Дж. Ф. Осанны фирма Bell Laboratories приобрела мини-компьютер PDP-11/20, который предполагалось использовать в проекте, связанном с разработкой системы, облегчающей создание документации. В 1971 г. ОС UNIX была перенесена на мини-компьютер PDP-11/20, где была использована в качестве основы для разработки системы, облегчающей создание документации. В июне 1972 г. появилась новая версия ОС

UNIX, также написанная на языке ассемблера и известная впоследствии как версия 2.

В это время К. Томпсон уже работал над созданием безтипового языка Би, строившегося на основе также безтипового языка Би-Си-Пи-Эл. Однако вскоре выяснилось, что без введения типов данных в разрабатываемый им язык программирования обойтись невозможно, да и реализация в виде интерпретатора не обеспечивала требуемого быстродействия. В результате был создан новый язык – Эн-Би. Наконец, занимавшийся вопросами генерации кода для транслятора с языка Эн-Би Д. М. Ритчи создал на его основе получивший впоследствии широчайшую известность язык Си.

В 1973 г. впервые появилась версия ОС UNIX, написанная на языке Си, известная впоследствии как версия 3. В мае 1975 г. появилась первая пригодная для режима промышленной эксплуатации версия ОС UNIX, известная в настоящее время как версия 6. В декабре 1977 г. ОС UNIX была перенесена на компьютер Interdata 8/32. Появившаяся в результате этого переноса версия ОС UNIX в настоящее время известна как версия 7. В 1978 г. ОС UNIX была перенесена на микрокомпьютеры семейства VAX-11. Эта версия ОС UNIX получила название 32V.

После создания ОС UNIX в течение длительного времени она была "широко известна лишь в узком кругу" специалистов по операционным системам, поскольку использовалась лишь в фирме Bell Laboratories и вычислительных центрах ряда американских университетов.

Затем последовал период многочисленных переносов ОС UNIX на компьютеры различных типов и создания ОС, подобных ОС UNIX. В истории создания и развития ОС UNIX особое место занимает 1980 г., когда ОС UNIX и подобные ей ОС начали активно использоваться, в том числе и на персональных компьютерах. Именно в 1980 г. ОС UNIX привлекла к себе внимание делового мира – основного пользователя персональных компьютеров.

В 1980 г. корпорация AT&T (в которую входит фирма Bell Laboratories) выпустила версию ОС UNIX, получившую название System III, а в январе 1983 г. – версию System V. В январе 1984 г. корпорация AT&T выпустила версию ОС UNIX, получившую название System V Release 2.0, претендующую на то, чтобы стать стандартом.

Разработка ОС UNIX была высоко оценена специалистами. В 1982 г. К. Томпсон и Д. М. Ритчи за создание ОС UNIX были удостоены "Премии за достижения", ежегодно присуждаемой редакцией журнала Electronics. При этом необходимо отметить, что впервые за 9 лет с момента ее учреждения эта премия была присуждена за разработку программного обеспечения, а не за разработку аппаратных средств. В 1983 г. за создание ОС UNIX К. Томпсон и Д. Ритчи были удостоены сразу двух премий Ассоциации специалистов по вычислительной технике: "Премии Тьюринга" и "Премии за разработку нового программного обеспечения".

Стандартизация ОС *UNIX*

В настоящее время предпринимаются серьезные усилия по стандартизации ОС UNIX и подобных ей операционных систем. Причиной этого является, с одной стороны, чрезвычайно широкое распространение ОС этого типа, а с другой стороны, появление все большего числа их неполностью совместимых друг с другом версий. В 1985 г. число таких версий перевалило за 70. Ситуация к тому же усугубляется необычайно быстрыми темпами развития ОС этого типа.

Наличие столь большого числа не совместимых друг с другом версий ОС этого типа вызывает все большую озабоченность среди пользователей, которые по вполне понятным причинам хотят иметь единую унифицированную операционную среду.

Начало мероприятия по стандартизации ОС UNIX, по-видимому, было положено в 1984 г., когда Ассоциация пользователей ОС UNIX /usr/group выпустила для своих членов стандарт на ОС UNIX. Вслед за этим весной 1985 г. корпорация AT&T опубликовала документ под названием System V Interface Definition (SVID), представляющий собой стандарт на ОС UNIX версии System V, в котором ощущается явное влияние стандарта, подготовленного за год до этого Ассоциацией /usr/group. В 1986 г. корпорация AT&T выпустила расширенное издание этого документа в трех томах.

Деятельность по стандартизации ОС UNIX, начатая Ассоциацией /usr/group, в настоящее время осуществляется рабочей группой P1003

Института инженеров по электротехнике и радиоэлектронике, разрабатывающей стандарт Posix на мобильную операционную систему. В рамках этой рабочей группы образованы две подгруппы: P1003.2, занимающаяся разработкой стандарта на интерфейс с ОС, и P1003.3, занимающаяся разработкой верификатора, обеспечивающего возможность проведения проверки ОС на их соответствие стандарту. В апреле 1986 г. рабочая группа P1003 подготовила документ под названием Trial Use Standard for a Portable Operating System, представляющий собой проект стандарта на мобильную ОС. Этот вариант проекта стандарта продолжает быстро совершенствоваться, о чем, в частности, свидетельствует появление 2 января 1987 г. его девятой редакции.

Большую работу по стандартизации ОС UNIX проводит основанная в конце 1984 г. Ассоциация производителей средств вычислительной техники и программного обеспечения X/OPEN. В момент своего создания Ассоциация X/OPEN была чисто европейской организацией, в которую входили следующие фирмы: Bull, ICL, Nixdorf, Olivetti, Philips, Siemens. Несколько позднее в состав Ассоциации X/OPEN вошел ряд американских фирм: DEC, Ericsson, Hewlett-Packard, Sperry (в настоящее время фирма Sperry вошла в состав корпорации Unisys). Впоследствии Ассоциация X/OPEN получила заявки на вступление в нее, по крайней мере, еще от 15 европейских, американских и японских фирм.

Весной 1985 г. Ассоциация X/OPEN издала документ под названием X/OPEN Portability Guide. К лету 1986 г. было продано уже свыше 7000 копий этого документа. Второе — расширенное и дополненное — издание этого документа вышло в 1987 г.

Существенным препятствием на пути стандартизации ОС UNIX и подобных ей ОС является отсутствие опыта по стандартизации операционных систем. К тому же накопленный к настоящему времени опыт разработки и использования стандартов в области программного обеспечения носит пока что скорее негативный характер.

Однако, несмотря на сложности формальной стандартизации, ОС UNIX уже стала стандартом de facto для промышленных и научных организаций.

Микропроцессор 80386 фирмы *Intel*

ЭРИК БЕНДЕР, КЕН ГРИНБЕРГ

Наделенный огромным потенциалом, микропроцессор 80386 уже шагает к зрелости. Благодаря конкуренции производителей операционных систем детище фирмы Intel вплотную приблизилось к реализации мультипрограммного и многопользовательского режима.

Добро пожаловать в новое поколение вычислительных систем! Хотя его появление и связано с применением микропроцессора 80386, пока что чувство уже виденного может вызвать скорее раздражение, чем удовлетворение. Пользователи ПК еще не успели получить удовлетворительный инструментарий для работы с микропроцессором 80286, как уже нахлынул поток AT-совместимых ПК, базирующихся на микропроцессоре 80386. Их технические возможности прекрасно рекламируются, но пока от них мало пользы.

С точки зрения перспективы, дела обстоят иначе. Если отбросить в сторону внешние впечатления, то микропроцессор 80386 готов показать себя уже сейчас. Однако тем пользователям, которые хотят использовать его потенциальные возможности, необходимо понимание некоторых архитектурных особенностей вычислительных систем на базе микропроцессора 80386.

Естественно, фирма Intel не вложила бы 100 млн дол в разработку микропроцессора 80386 только для того, чтобы создать группу быстродействующих ПК типа AT. По мере поступления на рынок систем на базе микропроцессора 80386 становится очевиднее, что при разработке микропроцессора 80386 фирма Intel предусмотрела (на время отсутствия ориентированной на микропроцессор 80386 операционной системы) возможность использования промежуточных работоспособных решений, способных дать еще больше возможностей тем, кому они нужны сейчас. Учитывая 6-миллиардную стоимость программного обеспечения, находящегося в пользовании корпораций, изготовитель процессора вряд ли мог поступить иначе.

Однако фирма Intel оставила на усмотрение предприимчивых производителей аппаратуры и программного обеспечения задачу максимально полного использования возможностей своего новейшего процессора. Хотя может пройти год или два, прежде чем появятся новые типовые приложения, ориентированные на микропроцессор 80386, его нынешние пользователи, несомненно, оценят возможность одновременного выполнения нескольких программ, разработанных для микропроцессора 8086.

Появление систем на базе микропроцессора 80386 могло показаться главным событием 1987 г. на рынке аппаратуры, однако гораздо важнее решение вопроса о создании нового системного программного обеспечения, которое должно предусматривать совместимость с DOS и одновременно мультипрограммный режим.

В марте 1987 г. фирмы IBM и Microsoft еще оставались в стороне от решения этого вопроса, поэтому понятно, почему микропроцессор 80386 находился на "детской" стадии своего развития и применялся в быстродействующих AT-совместимых ПК управляющими программами, хорошо использующими его быстродействие, и специализированными операционными системами. Однако это только начало: Рональд Фишер, президент фирмы Interactive Systems, поставляющей программное обеспечение для ОС UNIX, считает, что микропроцессор 80386 предоставляет качественно новые возможности. Он говорит: "Происходит смена караула. Персональный компьютер, основанный на микропроцессоре 80386, не является более ПК уровня IBM PC. По своим функциональным возможностям он соизмерим с мини- и большими ЭВМ".

Как выглядят первые ПК на базе микропроцессора 80386 и каковы последствия различий в их конструкции? Конкретнее, как может помочь вам система на базе микропроцессора 80386 сегодня и что она сможет сделать для вас завтра? Должны ли вы переждать некоторое время, пока эти машины немного "повзрослеют"? Или стоит "окунуться с головой" немедленно?

КАВАЛЬКАДА ВОЗМОЖНОСТЕЙ

Микропроцессор 80386 в конечном счете преобразует мир персональных компьютеров не столько за счет увеличения производительности, сколько за счет предоставления программам огромного объема адресуемой памяти. Огромный размер ОЗУ приведет к появлению новых классов приложений, многие из которых будут использовать возможность работы в фоновом режиме. Роберт Карр, создатель пакета программ Framework, считает, что появление микропроцессора 80386 будет способствовать созданию "модульных" приложений, которые будут предоставлять пользователю выбор из нескольких интерфейсов. Кроме того, появление микропроцессора 80386 должно привести к взрыву в области экспертных систем, издательского дела, сетевого программного обеспечения, САПР, моделирования, обработки транзакций, а также мультипликации в реальном времени.

Одним из предвестников эры микропроцессора 80386 является проект Migent's Emerald Bay – детище создателя СУБД dBASE Уэйна Ратлиффа. Ратлифф отлаживает "двигатель" (ядро программы) для 32-битовой системы, который будет способен работать с огромными многопользовательскими базами данных. Микропроцессор 80386 позволит ему поддерживать набор стандартных интерфейсов, что будет способствовать подлинной интеграции данных.

Хотя подобные приложения еще не появились, ПК, основанные на микропроцессоре 80386, уже появляются и занимают свое место на все более функционально разграниченном рынке. На наиболее прозаическом конце находятся AT-совместимые ПК, предназначенные для тех, кто желает (и может) быстро адаптироваться к новым условиям и возможностям. На выставке COMDEX в ноябре 1986 г. было огромное число таких ПК, более двух дюжин производителей – от Zenith и Compaq до Mitsui и PC's Limited – демонстрировали свою продукцию. А сменные процессорные платы с микропроцессором 80386 фирм Intel, Quadram и др. подтвердили, что не нужно сбрасывать со счетов уже установленные ПК AT и XT.

Более специализированными являются созданные на базе микропроцессора 80386 рабочие станции для технических приложений, когда нужна высококачественная графика или большая внешняя память. Эту тенденцию олицетворяет ПК HummingBoard LISP – совместный продукт фирм Gold Hill и AI Architects. Юджин Уонг, вице-президент фирмы Gold Hill по маркетингу, говорит: "Микропроцессор 80386 является последним достижением в этой области". Сообщество специалистов по искусственному интеллекту соглашается с такой точкой зрения: поставщики трансляторов языка Лисп фирмы Lucid и Franz, а также поставщик транслятора языка Пролог, фирма Arity были одними из первых производителей программного обеспечения, которые стали ориентироваться на микропроцессор 80386.

Большинство рабочих станций на базе микропроцессора 80386, работающих в области графики и искусственного интеллекта, будут специализированными: они будут иметь большие экраны, огромные диски и очень высокоскоростные связи между шиной и периферийными устройствами.

Фирмы Corvus, Novell, Kaypro и TeleVideo одними из первых выпустили основанные на микропроцессоре 80386 серверы локальных сетей. Все они совмещают быстродействие микропроцессора 80386 с очень большими и быстрыми жесткими дисками. Сервер фирмы Novell (согласно рекламе) поддерживает до 1000 рабочих станций, выдерживает перебои в электросети и снабжен новой версией сетевого программного обеспечения Netware. Конечно, как замечает глава фирмы 3Com Уильям Краузе, производительность сетей не измеряется исключительно в миллионах операций в секунду: "Суперпроцессор еще не гарантирует суперсервера; сервер должен оптимизировать движение данных, а не их обработку".

Вопрос о том, будут ли основанные на микропроцессоре 80386 ПК процветать в качестве многопользовательских систем, остается открытым, хотя ответ на него упорно пытается найти целая группа производителей программного обеспечения. По неофициальным сообщениям известная компания Altos разработала систему, способную поддерживать до 40 пользователей. При этом подобные многопользовательские ПК могут быть только началом. Ланс Хэнше, вице-президент фирмы Phoenix Technologies, подумывает о создании экспериментальной многопроцессорной системы на базе микропроцессора 80386, которая будет иметь до шести процессоров на одном шасси. Rexon, еще одна фирма, выпускающая многопользовательские ПК, подкрепила свою линию систем, основанных на микропроцессоре 80286, возможностью добавления нескольких сопроцессоров 80386.

Выступая уже сейчас во множестве ролей, микропроцессор 80386 имеет практически гарантированное будущее. Президент фирмы Ashton-Tate Эд Эсбер открыто говорит: "Теперь мы должны использовать разрыв между микропроцессорами 8088/8086 и 80386, а не разрыв между микропроцессорами 80286 и 80386.

ПРОЦЕССОР ДЛЯ НОВЫХ ДЕЛ

"Очень трудно описать микропроцессор 80386, поскольку он приносит слишком много нового в мир персональных компьютеров", – говорит председатель Совета директоров фирмы Microsoft Билл Гейтс.

Ведущий 32-разрядный микропроцессор фирмы Intel способен выполнять более 3 млн команд в секунду, имея непосредственный доступ к 4 Гбайт физического адресного пространства и 64 Тбайт (более 70 триллионов байт) виртуального адресного пространства. Конвейерная архитектура процессора позволяет иметь очень быстрый доступ к памяти, в то время как реализованная на плате система управления памятью защищает задачи друг от друга, создавая возможность независимого сосуществования многих операционных систем. Микропроцессор 80386 может одновременно выполнять программы, написанные для микропроцессоров 8086, 80286 и 80386 (см. вставку "Интригующая память").

В целом микропроцессор 80386 подобен процессору большой ЭВМ, размещенному на 1 см^2 кремниевой пластины. Можно возразить, что он не предоставляет достаточных возможностей высокоскоростного ввода-вывода, чтобы быть отнесенным к этой категории, но при наличии вспомогательных аппаратных средств он успешно справляется с этой работой (обзор возможностей микропроцессора 80386 приведен в февральском номере журнала PC World за 1986 г. в статье "80386: The Megabyte Manager").

Микропроцессор 80386 более сбалансирован и удобен, чем предыдущие микропроцессоры фирмы Intel. Джерард Попек, президент фирмы Locus Computing, производящей программное обеспечение для ОС UNIX, говорит: "Микропроцессор 8086 имеет ограниченные возможности, а микропроцессор 80286 обладает рядом дефектов и не совместим с другими микропроцессорами в защищенном режиме. В микропроцессоре 80386 эти ошибки микропроцессора 80286 исправлены, и он является удобным и быстрым средством, на которое можно ориентировать программы. Он имеет все достоинства процессоров серии 68000, а кроме того, существует масса приложений, которые могут на нем выполняться".

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ, ОРИЕНТИРОВАННОЕ НА ОПРЕДЕЛЕННЫЙ РЕЖИМ РАБОТЫ ПРОЦЕССОРА

Микропроцессор 80386 может работать в двух режимах: реальном и защищенном. В реальном режиме процессор имитирует микропроцессор 8088/8086, выполняя соответствующие программы с большей скоростью; однако при этом он работает в однопрограммном режиме и его адресное пространство также ограничено объемом 1 Мбайт. За некоторыми исключениями создатели нынешних AT-совместимых ПК, реализованных на базе микропроцессора 80386, придерживаются этой линии.

Защищенный режим (формально – защищенный режим виртуальной адресации) раскрывает полные возможности микропроцессора 80386. В этом режиме процессор может работать со всем своим адресным пространством и поддерживать "виртуальные 8086-машины", работающие одновременно, в то время как изощренные аппаратные средства управления памятью обеспечивают необходимую поддержку. По мнению Дэна Крелле, ответственного в фирме Intel за продажу микропроцессора 80386, "задачи защищены друг от друга самой архитектурой процессора. Все возможности предоставляются всем задачам".

Несмотря на это, существуют некоторые ограничения. Хотя в виртуальном режиме микропроцессор 80386 способен одновременно выполнять несколько программ, ориентированных на микропроцессор 8086, при этом возникают проблемы с программами, использующими специфические элементы аппаратного обеспечения ЭВМ, например экран. В этом случае операционная система,

ориентированная на микропроцессор 80386, должна перехватывать аппаратные прерывания микропроцессора 8086, что заметно снижает производительность.

Операционная система (или некоторый посредник между операционной системой и ПК) может обеспечить нормальную работу приложений, ориентированных на DOS, с помощью программного обеспечения, называемого виртуальным монитором. Виртуальный монитор вмешивается, когда программа делает системный вызов или обращается к процедурам ввода-вывода и перехватывает эти вызовы, если это необходимо. Несколько программ могут одновременно выводить информацию на то, что, как они думают, является экраном (виртуальный экран); операционная система или посредник в таком случае решают, что же действительно изображается на экране. Приложения, ориентированные на однопользовательский и однопрограммный режим работы, могут, таким образом, выполняться в многопользовательской и мультипрограммной среде.

Программы, ориентированные на защищенный режим работы микропроцессора 80286, также могут выполняться на микропроцессоре 80386 при соблюдении следующих условий: либо операционная система ориентирована на микропроцессор 80286, либо она ориентирована на микропроцессор 80386, но способна перехватывать (и поддерживать) системные вызовы программ, разработанных для микропроцессора 80286. Чувство неудобства, вызываемое этими особенностями микропроцессора 80286, которое возникает у разработчиков программного обеспечения, частично объясняет малое число программ для микропроцессора 80286. К тому же ходят слухи об аппаратной ошибке в ранних версиях микро-

Интригующая память

ЭРИК БЕНДЕР

Системы, базирующиеся на микропроцессоре 80386, появляются в момент, когда происходит переход на новые аппаратные компоненты. Один из них – блок оперативной памяти, где 1-Мбитовые микросхемы начинают заменять 256-Кбитовые. Чем позже производитель системы на базе микропроцессора 80386 выйдет на рынок, тем более вероятно, что его машина будет иметь 1-Мбитовые микросхемы.

Вице-президент фирмы Phoenix Technologies Лэнс Хэнше считает: "Для систем, которые выйдут на рынок в первом квартале 1987 г., использование 1-Мбитовых микросхем представляет определенный конструкционный риск". Однако, поскольку 1-Мбитовые микросхемы появляются в больших количествах, а цена на них становится ниже, чем на 256-Кбитовые, выбор становится очевидным.

Более трудной проблемой для проектировщиков является выбор архитектуры памяти. Дэн Крелле из фирмы Intel предлагает следующую классификацию типов архитектуры памяти: традиционная, использующая расслоенную память, использующая странично-организованную память и реализующая кэш-память. Традиционный подход хорошо изучен и сравнительно дешев, но не дает системе никаких выгод в скорости.

Расслоение позволяет использовать конвейерную архитектуру микропроцессора 80386 и осуществлять различные операции (выборку, декодирование, выполнение, управление памятью и доступ к шине) одновременно. "Увеличение производительности с помощью расслоения достигается путем динамического переключения – каждый последующий доступ идет к альтернативному банку памяти", – объясняет Крелле. Расслоение позволяет скрывать такты на шине от процессора; для доступа к памяти могут потребоваться, например, три такта, но процессор будет считать, что прошло только два.

Для 32-битовых операций такой подход требует, чтобы "ширина" памяти была не менее 64 бит. Для системы, построенной на 256-Кбитовых микросхемах, это требует памяти не менее 2 Мбайт (поскольку вам потребуется 64 микросхемы), а система на 1-Мбитовых микросхемах должна соответственно иметь не менее 8 Мбайт памяти. Вице-президент фирмы Compaq Гэри Стимак говорит: "Подобная организация памяти неестественна".

Фирма Compaq остановилась на микросхемах со странично-организованной памятью, которые появились как раз во время разработки машины Deskpro 386 и обеспечивают высокую скорость доступа. Во время последовательных операций в пределах одной страницы такой памяти адрес строки остается постоянным, а адрес столбца модифицируется. В результате обычное время доступа 100 нс сокращается вдвое. Система управления памятью, реализованная фирмой Compaq, имеет страницы размером 2 Кбайт; обычно около 60% обращений к памяти делается в страничном режиме и в среднем получается 0,8 цикла ожидания на одно обращение к памяти.

Использование кэш-памяти, кроме того, означает сохранение наиболее часто используемых программ и данных в очень высокоскоростной (и соответственно дорогой) памяти. Кэш-память представляет собой просто буфер между стандартной оперативной памятью и системой. Крелле замечает, что существуют различные схемы организации кэш-памяти, наиболее быстрые из которых дают 0,3 цикла ожидания на одно обращение.

Исследования фирмы Compaq опровергают мнение, что экзотические архитектуры памяти обеспечивают более высокую производительность; подключение подобной памяти к шине сложнее, чем подключение обычной памяти, но на стандартных PC-ориентированных программах они обеспечивают сравнимую производительность.

процессора 80386, которая не позволяет работать на них программному обеспечению, ориентированному на защищенный режим микропроцессора 80286.

Несмотря на обещания фирмы Intel о сохранении совместимости микропроцессора 80386 с его 16-разрядными предшественниками, он в некотором смысле "посеял смуту" в процессе "естественного" развития операционных систем для ПК. Джим Джонсон, один из управляющих Отделом по усовершенствованию ПК фирмы Intel, говорит: "Микропроцессор 80386 обладает чертами, требующими новой операционной системы".

Что же это за операционная система? В мире персональных компьютеров этот вопрос является предметом шуток: в настоящий момент нет операционной системы, одобренной фирмой IBM, которая могла бы обеспечить плавный переход для существующих приложений, ориентированных на DOS, а фирма Microsoft вообще намекает, что такой переход может оказаться в принципе невозможным.

КАКАЯ ОПЕРАЦИОННАЯ СИСТЕМА?

Учитывая отсутствие серьезной альтернативы у разработчиков программного обеспечения, DOS версий 3.xx можно считать доминирующей операционной системой для систем на базе микропроцессора 80386, по крайней мере на ближайшее время.

В то время как фирма Microsoft всячески рекламирует свой пакет Windows как предвестника DOS будущего, она не дает никакой детальной информации относительно путей, ведущих в это будущее.

Помимо этого, Адриан Кинг, директор фирмы Microsoft, ответственный за разработку операционных систем, признает только то, что фирма продолжает работу по использованию возможностей как микропроцессора 80286, так и микропроцессора 80386. Это можно интерпретировать как обязательство обеспечить "должный мультипрограммный режим", убрать ограничение в 640 Кбайт оперативной памяти для будущих приложений и перенести некоторые функции Windows в DOS.

Под управлением DOS 3.xx системы на базе микропроцессора 80386 работают в два-три раза производительнее, чем соответствующие ПК на микропроцессоре 80286, благодаря большей тактовой частоте, более быстрому доступу к памяти и одновременной обработке 32 бит. Версия DOS 4.00, не получившая "благословения" фирмы IBM и распространенная в основном в Европе, является однопользовательской мультипрограммной системой. Кинг говорит: "Версия DOS 4.00, ориентированная в основном на задачи коммуникации между ЭВМ и локальные сети, не является в полном смысле слова новой многоцелевой".

Версия DOS 5.00, так называемая усовершенствованная DOS или DOS для защищенного режима работы, как предполагается, даст приложениям возможность использовать все 16 Мбайт непосредственно адресуемой памяти микропроцессора 80286. Еще позднее DOS 6.00 откроет широкие возможности доступа к памяти, которыми обладает микропроцессор 80386 (рис. 1 и 2). Президент фирмы Phoenix Нил Колвин считает, что различие версий DOS несет в себе угрозу несовместимости: "Как DOS 5.00, так и DOS 6.00 представляют собой совершенно новую среду, отличную от среды DOS. Будут ли поддержаны все три?".

Другую альтернативу представляет операционная система Xenix System V/386 фирмы Microsoft. "Наш пер-

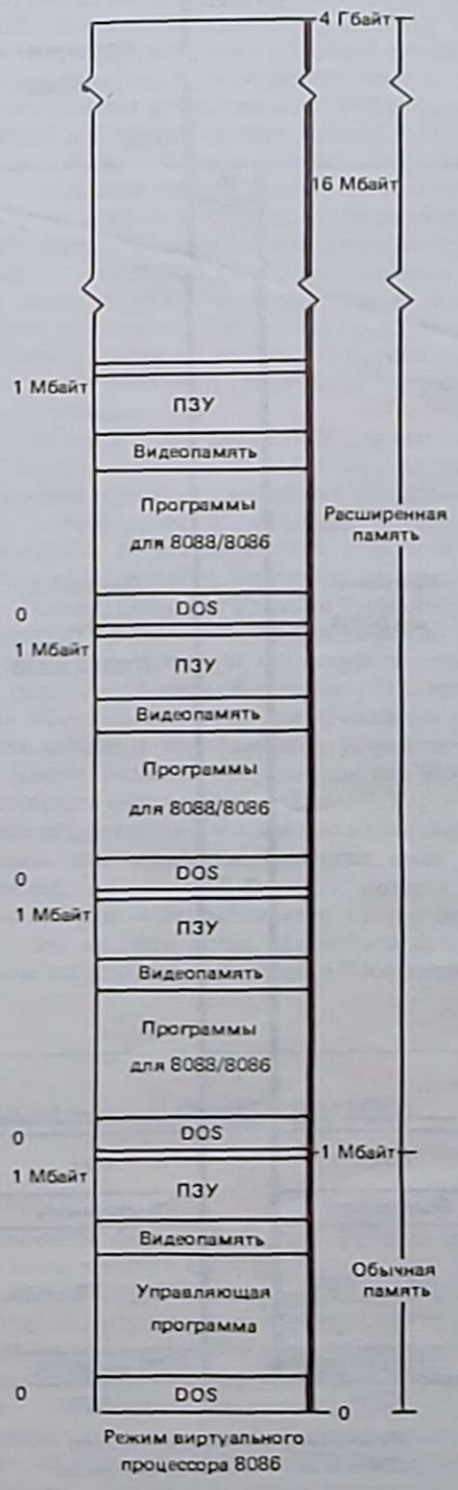
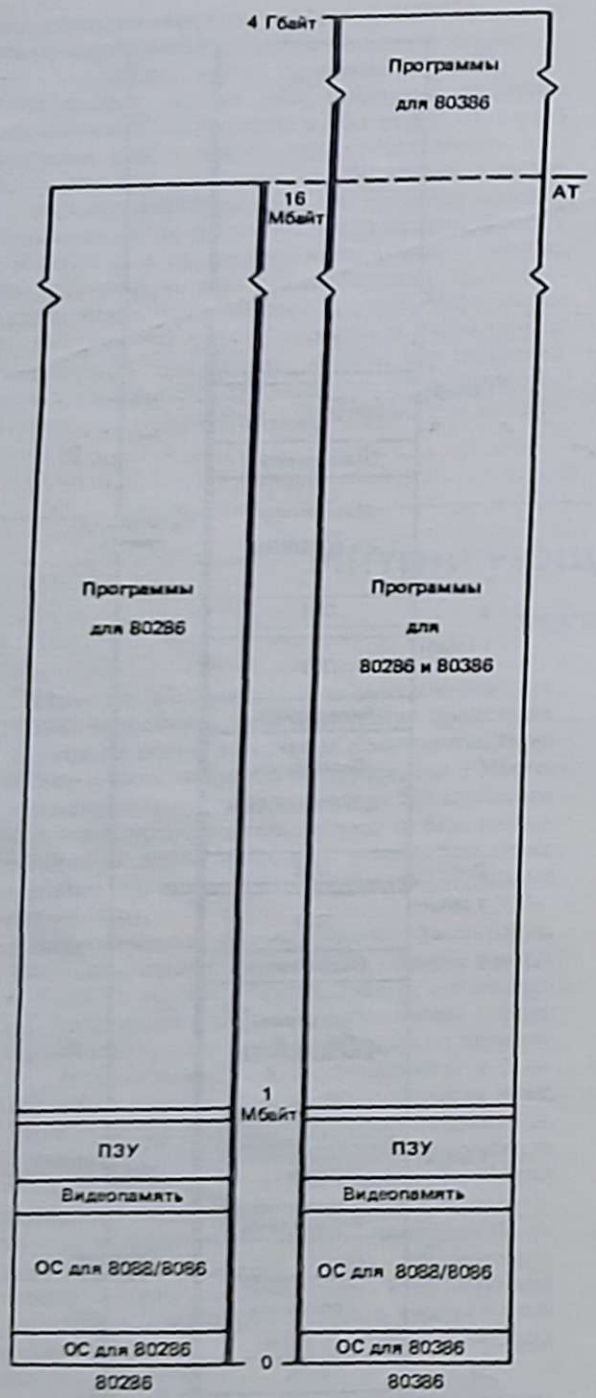


Рис. 1. При работе микропроцессора 80386 в режиме виртуального процессора 8086 каждое ориентированное на процессор 8086 приложение получит 1 Мбайт в расширенной памяти. Управляющая программа разделяет область памяти, имеющей наименьшие адреса, с программами для процессора 8086



Модели работы микропроцессоров 80286 и 80386 в защищенном режиме

Рис. 2. Операционная система DOS, работающая в защищенном режиме, сделает для микропроцессора 80386 доступными 16 Мбайт адресного пространства микропроцессора 80286, но не позволит воспользоваться огромными ресурсами оперативной памяти самого микропроцессора 80386. Обратите внимание на схожесть двух процессоров в адресации 16 Мбайт оперативной памяти с наименьшими адресами

воначальный выбор полностью на стороне ОС Xenix, – говорит Кинг. – Когда дело касается использования возможностей микропроцессора 80386, то очевидно, что Xenix технологически более совершенна, чем DOS". Кинг замечает, что ОС Xenix отлично использует линейную адресацию микропроцессора 80386, которая значительно удобнее для многопользовательской операционной системы, чем сегментная адресация 80286.

ОБЕЩАНИЯ, ОБЕЩАНИЯ

Поскольку основной поставщик операционных систем все еще не создал операционной системы, ориентированной на микропроцессор 80286, а тем более на микропроцессор 80386, сумятица сохраняется. По крайней мере в ближайшее время для новых компаний существует возможность вступить в игру. Действительно, поскольку большое рабочее пространство микропроцессора 80386 позволяет сосуществовать нескольким операционным системам, опытные компании, которые ранее не производили программных продуктов для DOS, рассматривают микропроцессор 80386 как тропинку в прибыльный мир персональных компьютеров.

Рекламируя "управляющие программы", поставщики программного обеспечения для ОС UNIX кооперируются с компаниями, производящими программное обеспечение для DOS, поставщики расширений операционных систем выпускают программные продукты, ориентированные на микропроцессор 80386, а некоторые фирмы даже начинают с нуля, рассчитывая, что судьба может улыбнуться им, пока Microsoft трудится над созданием будущей DOS. Сама фирма Microsoft серьезно изучает один такой гибрид DOS и UNIX (см. вставку "Управляющие программы действуют").

Состояние неопределенности в связи с задержкой появления новой операционной системы фирмы Microsoft еще не дает полной картины происходящего. Существует единое мнение, что с появлением микропроцессора 80386 индустрия ПК и ее потребители достигли водораздела; чтобы его пересечь, необходима только соответствующая операционная система. Рон Фишер из фирмы Interactive System говорит: "Потенциал микропроцессора 80386 привел к переоценке стратегии развития операционных систем. Когда-то не было сомнений в том, что средой разработки будет DOS. Сейчас это значительно менее очевидно".

Interactive System и подобные ей фирмы действуют во многом аналогично производителям аппаратного обеспечения, которые стали производить сверхбыстрые AT: и те, и другие подчеркивают безопасность предлагаемого ими подхода и возможность немедленного получения решений существующих проблем. И те, и другие утверждают, что их системы, когда они появятся, будут полностью совместимы с DOS 3.xx, мало или почти совсем не повлияют на производительность, будут способны выполнять существующие программы, ориентированные на микропроцессор 8086, и (через посредство виртуальных мониторов) будут способны работать в мультипрограммном режиме.

ТРУДНЫЙ ВЫБОР

Несмотря на отсутствие в настоящий момент четких представлений относительно будущей операционной системы, нетерпеливые поставщики аппаратного обеспечения рвутся вперед с ПК на базе микропроцессора 80386.

В то время как относительно немногие фирмы заняты проектами, основанными на использовании нескольких микропроцессоров 80386, и прочей экзотикой, производители типичных систем на базе микропроцессора 80386 сосредоточивают свои усилия на проблеме скорости. Центральной проблемой становится повышение тактовой частоты микропроцессора.

Хэнше из фирмы Phoenix говорит: "Люди пытаются создать машины с тактовой частотой 20 МГц, а наиболее напористые – 24 МГц". Однако проблемы с поддерживающими микросхемами возникают на много меньших скоростях. Гэри Стимак, вице-президент фирмы Compaq, ответственный за технологию разработок, считает, что частота 16 МГц уже поставила типовое аппаратное обеспечение на грань его возможностей; следующий шаг ведет в область аппаратной логики, используемой в основном большими ЭВМ и военной электроникой.

Параллельно с системами на базе микропроцессора 80386 появляется огромное число сменных плат, обещающих обработку на сверхбольших скоростях. Они подразделяются на платы, заменяющие процессор (соединенные кабелем с гнездом, освободившимся в результате удаления микропроцессора 80286), и сопроцессоры. Большинство основанных на микропроцессоре 80386 плат работает только с ПК AT; лишь фирма Quadram поступила иначе, выпустив модель для ПК XT. Большинство плат имеет большое ОЗУ с 32-битовым доступом; но некоторые производители, подобно фирме Intel, компенсируют использование более медленного и дешевого ОЗУ использованием кэш-памяти (описание кэш-памяти см. на вставке "Интригующая память"). В целом объем продажи таких сменных плат может превзойти объем продажи их предшественников, основанных на микропроцессоре 80286.

Ввиду отсутствия стандарта фирмы IBM первые производители применили разные подходы к шинам. Фирма Compaq скомбинировала свою 32-битовую шину памяти для доступа к высокоскоростному ОЗУ с 8- и 16-битовыми шинами для остальной периферии. Фирма Zenith, напротив, расширила до 32 бит шесть из восьми разъемов своего ПК, но все шесть допускают установку 8- и 16-битовых плат; в соответствии с традицией фирма Zenith даже разделила процессор и системную память. В основном же поставщики ограничиваются 32-битовым разъемом для ОЗУ и не трогают других каналов связи.

Президент фирмы Compaq Род Кэнион настаивает на том, что в настоящее время "единственный компонент, для которого требуется скорость 32-битовой шины, – это ОЗУ". Доступ к диску лимитируется быстродействием дискового контроллера, а с требованиями коммуникации легко справляются существующие 8-битовые шины. Хотя графические программы могли бы хорошо использовать 32-битовый доступ к внешним устройствам, неизбежное появление графических сопроцессоров умеряет эту необходимость.

Не дождавшись шины фирмы IBM для микропроцессора 80386, фирма Phoenix весной 1986 г. организовала неформальную группу для выработки отраслевого стандарта 32-битовой шины; это начинание вылилось в создание Комитета по стандартизации в области усовершенствованной технологии персональных компьютеров (PCET – Personal Computer Extended Technology), который выдвинул предложения, претендующие на роль будущих стандартов Института инженеров по электротехнике и радиоэлектронике. В октябре PCET предложил стандарт AT-совместимой шины с 32-битовым расширением для памяти и периферийных устройств, что, по словам Хэнше, является достаточным для разработчиков аппаратного обеспечения.

На ближайшее время сумятица почти наверняка будет преобладать над согласием. Выпуская свою систему Premium/286, фирма AST Research, которая создала целую индустрию плат расширения вокруг шины ПК IBM PC, без лишнего шума предложила собственное 32-битовое расширение 16-битовой AT-совместимой шины ПК.

Управляющие программы действуют

КЭН ГРИНБЕРГ

Появление микропроцессора 80386 породило новые операционные системы, управляющие программы и другие попытки покушения на традиционную монополию фирмы Microsoft.

Управляющая программа просто выполняет необходимые действия, чтобы приложения, ориентирующиеся на процессор 8086, мирно сосуществовали на нескольких виртуальных машинах. Когда-нибудь каждая операционная система, ориентированная на микропроцессор 80386, будет включать управляющий программный компонент. Пока же предлагаемые новые подходы различаются как по своим требованиям к памяти машины, так и по своей зрелости.

Ниже следует небольшой обзор предлагаемых (и еще разрабатываемых) продуктов такого рода.

VM/386. В то время как рынок средств защиты программного обеспечения сокращается, опытные программисты фирмы Softguard Systems ищут новые возможности в области систем на базе микропроцессора 80386. Предлагаемый фирмой продукт VM/386, созданный по образцу VM/370 фирмы IBM, по словам представителей компании, "создавался с ясным намерением предоставить среду для работы различных операционных систем". Управляющая программа фирмы Softguard, которую сама фирма окрестила "гостиницей для операционных систем", размещается между микропроцессором 80386 и несколькими виртуальными 8086-машинами (рис. А).

В среде VM/386 приоритетная задача контролирует монитор и клавиатуру, фоновые задачи выводят информацию на виртуальные мониторы, что позволяет программам типа 1-2-3, которые выводят информацию непосредственно на экран, работать в мультипрограммной среде.

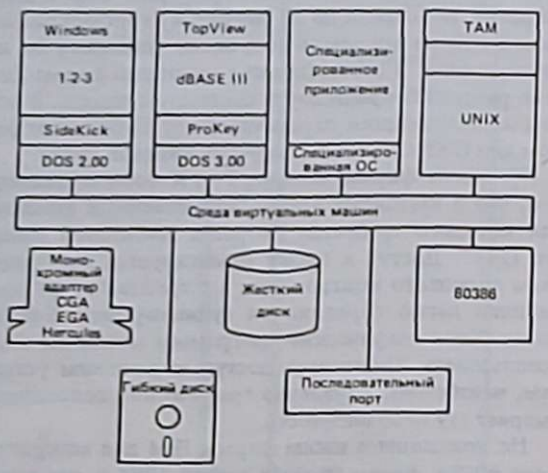


Рис. А. Работая под управлением программы VM/386 фирмы Softguard Systems, каждая виртуальная машина имеет собственную операционную систему и текущую задачу. Каждая действует так, как будто она монопольно владеет процессором

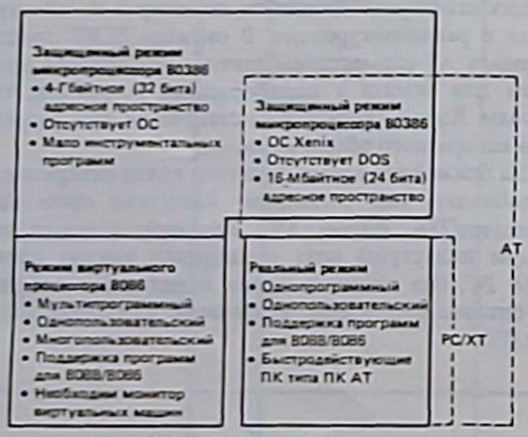


Рис. Б. В этом графическом представлении режимов работы микропроцессора 80386 фирмы Softguard Systems выделяются реальный и защищенный режимы; последний, в свою очередь, делится на режим виртуального процессора 8086, защищенный режим микропроцессора 80286 и собственно защищенный режим микропроцессора 80386

Система VM/386 рассматривает микропроцессор 80386 как простое 4-Гбайтное адресное пространство; карта памяти простирается через всю область виртуальной памяти. Выделяются области для приложений, ориентированных на процессор 8086, расширенной памяти (обоих основных типов), коммутации банков для процедур, ориентированных на работу с адаптером EGA приложений, изначально ориентированных на 32-разрядный процессор, и MVM/386 (8-пользовательский вариант операционной системы). Кроме того, целый гигабайт зарезервирован для межмашинной связи, что дает пользователям возможность "резать и клеить" данные между приложениями, выполняющимися на различных виртуальных машинах (рис. Б).

VP/ix. Результат совместной разработки фирм Interactive Systems и Phoenix Technologies, система VP/ix базируется на версии 3 ОС UNIX System V. Рон Фишер, главный управляющий фирмы Interactive Systems, называет VP/ix "мостом", который будет существовать, пока существуют ориентированные на процессор 8086 приложения, которые он поддерживает.

Система VP/ix – это то, что получается, когда объединяется опыт фирмы Phoenix в области эмуляции DOS с версией UNIX фирмы Interactive. Получающаяся система позволяет многим ориентирующимся на DOS приложениям выполняться в многопользовательской и мультипрограммной операционной среде ОС UNIX или Xenix 386. Фирма Interactive увеличила на 50 Кбайт ядро ОС UNIX и разработала интерфейс "виртуального терминала", в среде которого вы можете переходить от одной рабочей станции к другой простым нажатием клавиши. Система VP/ix также позволяет совместное использование данных программами, ориентированными на ОС UNIX, и программами, ориентированными на DOS.

Хотя сама система VP/ix почти не замедляет работы приложений, для работы вам требуется загрузить DOS, ядро ОС UNIX, затем VP/ix плюс сами программы и данные. Не удивительно, что для своей работы VP/ix требует наличия не менее 2 Мбайт оперативной памяти.

Фирма Interactive обхаживает поклонников ОС UNIX, работающих на мини-ЭВМ и в инженерных областях. Интересно, что это начинание получило одобрение фирмы Microsoft, которая попросила фирму Phoenix создать версию, работающую под управлением ОС Xenix.

Merge 386. Фирма Locus Computing базировалась на более ранней разработке SimulTask, которая позволяла загружать DOS на ЭВМ 6300 Plus фирмы AT&T как задачу в среде ОС UNIX. В отличие от своего ориентированного на микропроцессор 80286 предшественника, система Merge 386 способна одновременно выполнять ряд приложений, ориентированных на DOS, и не требует специального аппаратного обеспечения. Президент фирмы Locus Джерард Попек говорит: "Merge 386 выглядит и действует точно так же, как DOS. Однако этот продукт обладает дополнительными возможностями, так как он опирается на коммуникационные возможности ОС UNIX".

В сущности, Merge 386 перехватывает системные вызовы приложений, ориентированных на DOS 3.xx, и переводит их в соответствующие системные вызовы ОС UNIX. Хотя кажется, что это должно снижать производительность микропроцессора 80386, фирма Locus утверждает, что верно как раз обратное. Попек говорит, что, поскольку Merge 386 использует собственную схему кэш-памяти, некоторые программы работают даже быстрее. Merge 386 увеличивает ядро ОС UNIX приблизительно на 70 Кбайт и требует столько же оперативной памяти, сколько VP/ix. Фирма Locus включила (и, по слухам, дополнила) в пакет AT-совместимые BIOS-процедуры фирмы Award Software.

PC-MOS/386. Первая созданная с нуля операционная система, ориентированная на микропроцессор 80386 и совместимая с DOS, является продуктом фирмы The Software Link (TSL), расположенной в Атланте и в основном занимающейся разделяемой обработкой данных. Президент фирмы Род Роук говорит: "Мы не собираемся разобрать DOS до винтика или воссоздать ее – мы просто хотим поддерживать то, что требуется от операционной системы разработчикам приложений". PC-MOS – единственная из описываемых здесь систем, которая действительно подменяет DOS, что, может быть, объясняет, почему фирме TSL пришлось пройти самый трудный путь.

Фирма встраивает средства управления памятью, защиты программ друг от друга, управления входными и выходными потоками на принтер и межпрограммной коммуникации в PC-MOS/386. Операционная система сконструирована так, чтобы перехватывать приложения, которые выводят информацию непосредственно в видеопамять, действуя как арбитр для пользователей, запускающих несколько программ, или пользователей, работающих на удаленных терминалах.

Когда фирма TSL "нанесет последние штрихи" на свой продукт, PC-MOS/386 будет позволять своим пользователям работать в мультипрограммном режиме как в режиме виртуального процессора 8086, так и в защищенном режиме микропроцессора 80386. Фирма готовит 1-, 5- и 25-пользовательские версии системы.

CTOS/VM. На некотором расстоянии от родовых, ориентированных на DOS вариантов находится собственная разработка фирмы Convergent Technologies CTOS/VM – многопользовательская ОС, базирующаяся на микропроцессоре 80386, впервые продемонстрированная осенью 1986 г. на ЭВМ 386 NGEN той же фирмы. Тем не менее фирма оставила достаточно места для DOS; в состав CTOS/VM входит компонент, называемый Context Manager, который позволяет пользователям перескакивать между несколькими (до десяти) виртуальными 8086-машинами, работающими параллельно с приложениями для CTOS. Операционная система CTOS/VM ориентирована исключительно на разработчиков и скорее всего будет работать на системах фирмы Convergent с торговой маркой Unisys.

Desqview 1.3. Фирма Quarterdeck Office Systems была первой на рынке управляющих программ с версией Desqview для машины Deskpro 386 фирмы Compaq. Версия 1.3 перекрывает все предыдущие, но не поддерживает виртуальных 8086-машин, если только на вашем столе не стоит Deskpro 386 фирмы Compaq или Access 386 фирмы ALR. В декабре 1986 г. фирма Quarterdeck начала продавать продукт с системой фирмы ALR и создала свою версию модуля управления расширенной памятью фирмы Compaq (CEMM).

Подобно драйверу CEMM, с которым она может общаться, программа Desqview использует аппаратные средства управления памятью микропроцессора 80386, чтобы преодолеть барьер в 640 Кбайт рабочего пространства. А поскольку Desqview поддерживает усовершенствованную спецификацию расширенной памяти фирмы AST, программный код каждого приложения, ориентированного на виртуальный процессор 8086, может находиться в памяти выше границы в 640 Кбайт. Версия 1.3 поддерживает девять одновременно работающих приложений, причем ограничение не является существенным и фирма Quarterdeck обещает устранить его в следующей версии продукта.

"Мы думаем, что расширенная память уменьшает существующие в режиме виртуального процессора 8086 ограничения по памяти и тем самым переопределяет этот режим", – говорит Гэри Поуп из фирмы Quarterdeck. Результатом, по его мнению, является возможность выполнения больших программ, предоставления им большей памяти и более быстрого выполнения их, чем в случае использования плат расширения памяти.

Фирма Quarterdeck настаивает, что продукты типа CEMM и Desqview преобразовали то, что было временным решением, в фактический стандарт. Поуп добавляет: "Нет нужды смотреть свысока на практические пошаговые решения – именно так и делаются дела".

КОГДА ОТРЕАГИРУЕТ ФИРМА IBM?

Хотя рынок ПК на базе микропроцессора 80386 обещает много сюрпризов, выход на сцену фирмы IBM должен стать наиболее драматичным событием. Пока фирма почти не давала информации о своих намерениях, и преобладающее в индустрии ПК мнение сводилось к тому, что ее продукты не появятся до второй половины 1987 г.

Хотя очевидно, что фирма IBM активно работает над системами на базе микропроцессора 80386, она хочет сохранить свои большие инвестиции в программное обеспечение для ПК IBM PC, не говоря уже о сохранении альтернатив (типа System/36), которые не являются конкурентоспособными по критерию производительность–цена.

Столь же ясно, что фирма IBM имела веские причины для подписания соглашения о совместных разработках с фирмой Intel – выгода заключается в компонентах, поддерживающих микропроцессор 80386 наряду с другими процессорами. Менее вероятны слухи о том, что фирма IBM собирается разработать собственную версию 32-битового процессора, поскольку неясно, какими выгодами можно было бы оправдать требуемые огромные капиталовложения.

Ходят также слухи, что фирма IBM собирается предложить своим клиентам из числа крупных корпораций

системы на базе микропроцессора 80386 с реализованными в микросхемах ПЗУ коммуникационными возможностями сетевой архитектуры (SNA – Systems Network Architecture). Очевидно, возможен и ряд других подходов, базирующихся на концепции собственной разработки.

Другие ожидают, что микропроцессор 80386 будет работать в качестве сопроцессора в существующих многопользовательских системах фирмы IBM. Майкл Гоулд, аналитик фирмы Yankee Group, считает, что для этой цели неплохо подходит относительно дешевая ЭВМ IBM 9370. Он говорит, что, работая под управлением операционной системы VM фирмы IBM, "ЭВМ IBM 9370 рассматривает наличие сопроцессора как еще одну задачу, так что подобный вариант может быть реализован без каких бы то ни было затруднений". Подобный вариант может привести к мощной комбинации большой ЭВМ и программного обеспечения ПК.

Хэнше предсказывает сближение систем на базе микропроцессора 80386 и системы System/36, что даст фирме IBM возможность использовать большое количество существующего программного обеспечения, разработанного для System/36. Он говорит: "Я думаю, что фирма IBM не будет в состоянии выдержать конкуренцию в области разработки программного обеспечения, если она не сможет воспользоваться уже имеющимися программами".

ЗАТРУДНИТЕЛЬНЫЙ ВЫБОР

Когда же, учитывая неясность ситуации, покупка ПК на базе микропроцессора 80386 будет иметь смысл – особенно для типичных коммерческих приложений?

Нынешние цены делают невыгодным приобретение появляющихся новых систем для нерегулярных работ по подготовке текстов и т. п., однако однопользовательские модели стоят немного дороже соответствующих AT-совместимых ПК. Действительно, на выставке COMDEX осенью 1986 г. демонстрировался целый ряд ПК из Юго-Восточной Азии, которые продавались по вполне доступным ценам. Платы, содержащие только микросхемы процессора 80386, стоили около 400 дол, хотя они, естественно, не обеспечивают всех дополнительных потребностей пользователя.

Несмотря на цену, системы на базе микропроцессора 80386 вряд ли являются расхожим товаром; процессор просто слишком нов, а его потенциал слишком велик, чтобы относить их к этой категории. Не удивительно поэтому, что ни пользователи, ни производители программного обеспечения не слишком торопятся с переходом на микропроцессор 80386. Адриан Кинг из фирмы Microsoft отмечает, что разработчики не склонны к тому, чтобы игнорировать большое число работающих ПК класса AT. "Микропроцессор 80286 будет стандартом крупных корпораций в течение как минимум двух–четырех лет", – говорит он. Учитывая, что приложения для ПК, которые будут в состоянии действительно использовать обе архитектуры (80286 и 80386), появятся только через год-два, некоторые поставщики программного обеспечения, вероятно, сразу начнут ориентироваться на микропроцессор 80386.

В более широкой перспективе некоторые рассматривают появление микропроцессора 80386 как событие, которое должно вызвать гигантские изменения в области программирования на ПК: переход от манипулирования битами к эре, которая обещает как высокую производительность, так и мирное сосуществование приложений, обеспечиваемое поддержкой одновременной работы виртуальных машин.

"Мы рассматриваем это как очень многообещающий фактор, – говорит Хэнше. – Виртуальная среда позволяет создавать ПК, не являющиеся ПК IBM PC по своей сути, но способные работать с программным обеспечением для них. Сейчас вы можете делать поистине странные машины, которые все еще могут оставаться совместимыми".

Совместимость с ПК IBM PC не является, однако, автоматической. "Многие теперь считают, что обеспечить совместимость легко, – отмечает Род Кэнион из фирмы Compaq – Мы думаем, что появление микропроцессора 80386 изменит это мнение. Слишком многие проблемы совместимости идут далеко за пределы BIOS ПЗУ и самого процессора".

"Работая с микропроцессором 80386, вы немедленно сталкиваетесь с всевозможными странными проблемами, связанными с временными соотношениями", – говорит Кинг из фирмы Microsoft. Одним из примеров являются схемы защиты программного обеспечения, которые не срабатывают при тактовой частоте 16 МГц. Фирма Compaq нашла выход, разработав достаточно хитрую систему BIOS, чтобы замедлять работу при выполнении проверок, осуществляемых схемами защиты. Однако не стоит полагаться на подобную программистскую изворотливость каждого поставщика систем. Дейв Спрингер из фирмы NDR Engineering, разработчик платы-ускорителя для машины 386 Turbo фирмы American Computer, говорит, что многие фирмы "копируют плату процессора, вставляя в нее более быстрый микропроцессор, и молятся за удачу".

Для изготовителей подлинно PC-совместимых машин на базе микропроцессора 80386 основным привлекательным фактором является просто долгое и многообещающее будущее подобных систем. Сейчас, когда наступает эра микропроцессора 80386, мечты производителей программного обеспечения простираются далеко.

Джеральд Попек из фирмы Locus говорит: "Программы смогут лучше использовать усовершенствованные возможности аппаратуры. Мультипликация в реальном времени обогатит пользовательский интерфейс". Президент фирмы Microsoft Кент Джонсон намекает, что готовятся еще более далеко идущие проекты. Он говорит: "Микропроцессор 80386 позволит объединить административные информационные системы с операционной средой персональных компьютеров. Базы данных в миллионы записей станут обычным явлением; графика и изображения будут храниться в базах данных. Локальные сети будут основываться на обработке транзакций и будут легко настраиваемыми. Подумайте, программы смогут поддерживать различные интерфейсы". Создатель пакета dBASE Уэйн Ратлифф превозносит начало подлинной интеграции "на уровне данных". И этот список можно продолжить.

Так что не исключено, что следующее поколение пользователей, может быть, вообще не будет доверять компьютерам с разрядностью менее 32.

Фирма *Intel* подтверждает наличие ошибки в микропроцессоре 80386

ФИРМА SCO ПРЕДЛАГАЕТ СПОСОБЫ ОБХОДА ОШИБКИ

МАРТИН МАРШАЛЛ *

В середине апреля фирма Intel подтвердила наличие ошибки в микропроцессоре 80386, работающем совместно с сопроцессором 80387 под управлением 32-битовой версии ОС UNIX.

Разработчик ОС Xenix фирма Santa Cruz Operation (SCO) детально описала ошибку, на которую она натолкнулась при попытке полного использования 32-битовых команд микропроцессора 80386, а также способы ее обхода. Ошибка, возникающая в случае использования 32-битовой адресации при работе микропроцессора 80386 в режиме страничной виртуальной памяти, не может произойти при работе программ, ориентированных на MS-DOS или OS/2, поскольку эти операционные системы используют процессор либо в реальном режиме, либо в защищенном режиме (режиме защиты памяти) микропроцессора 80286.

Согласно мнению представителей фирмы SCO, при работе ОС Xenix или другой реализации ОС UNIX, ориентированной на микропроцессор 80386, в случае наличия сопроцессора 80387 может произойти зависание системы. По словам одного из руководителей фирмы SCO Дейвида Бернстейна, это может произойти только в случае одновременного выполнения следующих четырех условий.

Первое из них состоит в том, что режим страничной виртуальной памяти должен быть активным и всегда выполняться при работе ОС Xenix. Второе заключается в том, что система должна использовать периферийное устройство, ориентированное на прямой доступ к памяти, такое как дисковод ГМД, сетевая плата, лентопротяжное устройство или (в случае модели 80 серии PS/2) жесткий диск. Третье условие предполагает наличие и использование сопроцессора 80387. Наконец, четвертое — микропроцессор 80386 должен находиться в состоянии ожидания. При выполнении всех этих условий система может находиться в состоянии ожидания неопределенно долгое время. Сопроцессор 80387 будет ожидать микропроцессор 80386, и наоборот.

По словам представителя фирмы Intel Урсулы Херрик, указанная ситуация происходит при работе с нынешними версиями B и C микропроцессора 80386 и будет исправлена в версии D, которая еще не поступила в продажу.

По мнению Бернстейна, простейшим выходом из ситуации является отказ от использования сопроцессора 80387. Если же вы все-таки хотите сохранить возможность использования сопроцессора, выходом может стать покупка специальной платы, размещаемой между микросхемой процессора и объединительной платой.

Такая плата включает программируемые матричные логические схемы, содержащие поставляемый фирмой Intel микрокод, позволяющий обходить указанную проблему. Одна версия такой платы продается фирмой Ironwood Electronics (Сент-Пол, шт. Миннесота) и стоит 145 дол. Другая версия, называется 386 Math Adaptor, продается фирмой Bell Technologies (Фримонт, шт. Калифорния) и стоит 195 дол. Различие между ними состоит в основном в том, что плата фирмы Ironwood Electronics намного меньше платы фирмы Bell Technologies и допускает добавление дочерних плат.

Фирма SCO советует пользователям не спешить с покупкой такой платы, а предварительно проверить, не установлена ли она уже в ПК. Некоторые недавно выпущенные ПК на базе микропроцессора 80386 включают такую плату.

Пользователи ОС UNIX, работающие с ПК на базе микропроцессора 80386, не использующие сопроцессора 80387, столкнулись с собственными проблемами, связанными с сигналом наличия сопроцессора на микросхеме микропроцессора 80386. По утверждению представителей фирмы SCO, эти проблемы не являются следствием какой-либо ошибки в микропроцессоре 80386, а связаны со способом обработки сигнала наличия сопроцессора некоторыми разработчиками систем. "Все они предполагали, что люди будут использовать эти ПК с операционной системой MS-DOS, которая получает информацию о конфигурации системы из КМОП-ОЗУ", — говорит руководитель фирмы SCO, ответственный за разработку ОС Xenix, Билл Бродерз. Некоторые из разработчиков просто не подключали этого сигнала к шине ПК AT, в то время как другие рассматривали его как не имеющий определенного значения и игнорировали.

Эта проблема возникает не тогда, когда пользователь имеет в составе системы сопроцессор 80387, а тогда, когда сопроцессор отсутствует, в то время как операционная система предполагает его наличие. Она в основном возникала на ПК фирмы Compaq с тактовой частотой 20 МГц, и именно эта фирма указала инженерам фирм SCO, Interactive и Microport способы решения проблемы. Фирма SCO ликвидировала эту проблему в версии 2.2.2 своей ОС Xenix 386. Фирма Interactive Systems (Санта-Моника, шт. Калифорния) планирует исключить возможность ее возникновения в следующей версии (1.06) своего продукта, а фирма Microport — в версии 3.0 своей операционной системы.

---

* INFO World, 1988, April 25, P. 5.

Некоторые сведения об OS/2

ЭРИК НОРР

Сборник "В мире персональных компьютеров" отвечает на основные вопросы об операционной системе, о которой все говорят, но которой мало кто пользовался.

Даже если вы никогда не приобретете OS/2, рано или поздно вы непременно столкнетесь с этой операционной системой. Мысленно вернитесь к тому моменту, когда вы впервые увидели корневой каталог DOS. Что вы предполагаете увидеть при загрузке OS/2? Не пропадете ли вы во враждебном окружении?
Пользователи DOS, успокойтесь. Когда начнет работу Presentation Manager (входящая в состав OS/2 программа взаимодействия с пользователем, которая аналогична программе Windows), только большие любители Windows сразу почувствуют себя как дома. Однако под этой оболочкой скрывается операционная система, знакомая вам не меньше, чем команда DIR. Может показаться, что средства OS/2 для поддержки многозадачного режима и работы с оперативной памятью довольно существенно отличаются от аналогичных средств DOS, но они выглядят и функционируют как более совершенные их собратья. Чтобы подготовить вас к первой встрече с OS/2 сборник "В мире персональных компьютеров" дает ответы на некоторые наиболее распространенные вопросы об этой операционной системе. Ответы относятся к OS/2 Standard Edition 1.00 фирмы IBM, поэтому некоторые пояснения (в частности, относящиеся к документации и установке системы) могут оказаться неприложимыми к версии фирмы Microsoft.

ПОХОЖИ ЛИ КОМАНДЫ OS/2 НА КОМАНДЫ DOS?

Команды OS/2 не просто похожи – они практически идентичны командам DOS и полностью соответствуют строгому синтаксису DOS. OS/2 воспринимает названия DIR, COPY, DEL, FORMAT, RENAME, BACKUP (список исключений из этого правила и список новых команд OS/2 приведены в разделе "Сведения о командах"). Создание, изменение и удаление каталогов по-прежнему осуществляются с помощью команд MD, CD и RD. Вы будете разочарованы, узнав, что длина имени файла все так же ограничена восьмью символами, а длина расширения – тремя.

КАКИЕ ТЕХНИЧЕСКИЕ СРЕДСТВА НЕОБХОДИМЫ ДЛЯ ЗАПУСКА OS/2?

Нужен жесткий диск и по меньшей мере 1,5 Мбайт оперативной памяти. Версия 1.00 OS/2 фирмы IBM поставляется на четырех 3½-дюймовых дисках емкостью 1,44 Мбайт каждый. На одном из этих дисков записана программа, предназначенная для установки операционной системы и обучения работе с ней. После установки OS/2 занимает около 2,5 Мбайт пространства на диске, на котором записано 112 файлов.

Поскольку для OS/2 требуется около 500 Кбайт оперативной памяти, то 1,5 Мбайт являются минимально возможным объемом памяти. Это справедливо и для рекомендованного фирмой IBM минимального объема памяти (2 Мбайт), необходимого для одновременного запуска программ DOS и OS/2. Наконец, основным достоинством OS/2 является возможность доступа к расширенной оперативной памяти объемом до 16 Мбайт, поэтому абсурдно ограничивать себя объемом памяти, лишь на 384 Кбайт превышающим объем, доступный в DOS. Фирма Microsoft считает, что для одновременной эксплуатации программ DOS и OS/2 требуется минимум 2,5 – 3 Мбайт оперативной памяти.

НАСКОЛЬКО СЛОЖНА УСТАНОВКА OS/2?

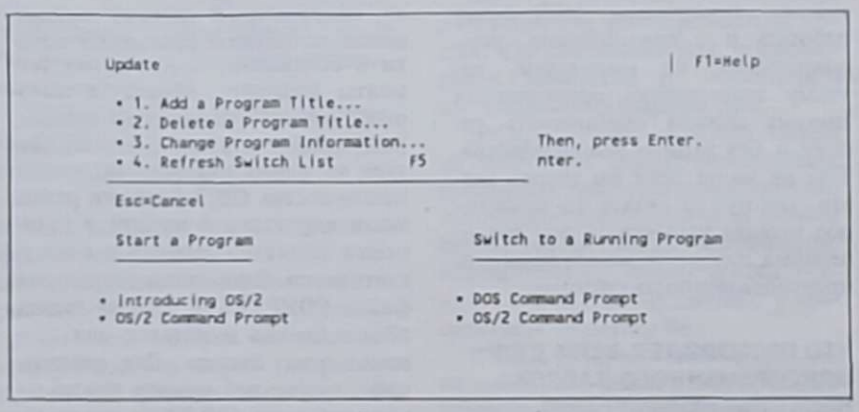
OS/2 установить проще, чем DOS. Программа для установки и обучения похожа на программу, записанную на диске Reference Diskette для PS/2 (см. статью "A Model for the '80s", опубликованную в декабрьском номере "PC World" за 1987 г.). Вы вставляете установочный диск в дисковод A, производите перезагрузку, и на экране появляется последовательность четких инструкций, дополненных меню и иллюстрациями. В общей сложности процедура установки системы занимает около получаса.

Одной из наиболее существенных функций, выполняемых программой установки, является автоматическое формирование файлов CONFIG.SYS и AUTOEXEC.BAT. От вас требуется лишь ответить на ряд запросов, и файлы будут сформированы. Если вы не знаете, как отвечать на запросы, то OS/2 сообщит значения, заданные по умолчанию. Это облегчает работу, поскольку число команд в файле CONFIG.SYS для OS/2 более чем вдвое больше по сравнению с DOS.

НУЖНО ЛИ ПЕРЕФОРМАТИРОВАТЬ ЖЕСТКИЙ ДИСК DOS ДЛЯ УСТАНОВКИ OS/2?

Если вы собираетесь на одном и том же компьютере запускать и DOS, и OS/2, то делать этого не нужно. Программа установки выдаст запрос, хотите ли вы пользо-

ваться программами DOS. Если да, то эта программа установит OS/2, оставив все, за исключением DOS, без изменений (старые файлы CONFIG.SYS и AUTOEXEC.BAT будут переименованы с расширением .BAK). Если нет, то в вашем распоряжении окажется высококачественный эквивалент утилиты FDISK, позволяющий перед переформатированием разбить жесткий диск на части. К сожалению, в OS/2 сохраняется принятое в DOS ограничение объема пространства на логических дисководах (32 Мбайт).



Program Selector позволяет загружать программы OS/2, не дожидаясь запроса ввода команды. С помощью функции Update можно добавлять, удалять или изменять имена программ в списке, приведенном под заголовком Start a Program.

ЧЕМ БЛОК СОВМЕСТИМОСТИ С DOS (DOS compatibility box) ОТЛИЧАЕТСЯ ОТ DOS?

Прежде всего блок совместимости с DOS — это достаточно громкий термин, обозначающий часть среды OS/2, зарезервированную для DOS и ее приложений. Эта часть среды может измениться при добавлении к системе долгожданной программы Presentation Manager. В настоящий момент блок совместимости вообще не локализован в системе как отдельный блок. Работа программ DOS в такой среде не отображается ни в окне, ни в меню. Работа программ просто выглядит похожей на работу этих программ под управлением DOS.

Основное различие между DOS и DOS под управлением OS/2 (и это различие очень огорчительно) состоит в том, что последняя может работать только с объемом оперативной памяти до 500 Кбайт. Выход из положения обеспечивается версией 4.0 стандарта Expanded Memory Specification фирм Lotus, Intel, Microsoft. Этот стандарт обеспечивает возможность запускать требующие большого объема памяти программы DOS, при этом дополнительная (сверх отведенных для режима DOS 500 Кбайт) память заимствуется из области оперативной памяти, предназначенной для приложений OS/2.

Есть и незначительные различия. В корневом каталоге DOS запрос ввода команды выглядит так: C:\>. Это соответствует случаю, когда в файле AUTOEXEC.BAT есть строка PROMPT=PG. Более очевидное различие состоит в том, что в верхней части экрана расположена надпись синими буквами по белому фону: DOS Ctrl+Esc=Program Selector–Type HELP=help (при работе с OS/2 вы увидите надпись белыми буквами по синему фону, начинающуюся с обозначения OS/2).

ЧТО В ПЕРВУЮ ОЧЕРЕДЬ ПОЯВЛЯЕТСЯ НА ЭКРАНЕ ПРИ ЗАГРУЗКЕ OS/2?

После пары сообщений и стандартной информации фирмы IBM на экране появляется информация программы Program Selector. По существу это основное меню операционной системы. Program Selector — это наиболее явное отличие OS/2 до DOS. (Это средство вам необходимо, поскольку OS/2 загромождает корневой каталог 50 файлами.) Program Selector разделяет экран на три области: Start a Program (запустить программу), Update (обновить) и Switch to a Running Program (переключиться на запущенную программу).

Под заголовком Start a Program приведен список всех доступных программ OS/2; под заголовком Update перечислены все функции, необходимые для редактирования этого списка. Для запуска программы OS/2 нужно добиться того, чтобы имя этой программы было подсвечено, и нажать клавишу Enter. Чтобы запустить программу DOS, нужно выбрать элемент DOS Command Prompt (запрос ввода команды DOS) из области экрана Switch to a Running Program и запустить программу обычным способом.

После запуска программы ее имя будет продублировано в области экрана Switch to a Running Program. Если при установке операционной системы было указано, что совместимость с DOS необходима, то элемент DOS Command Prompt всегда присутствует в этой области экрана. Если запущено несколько программ, то самый быстрый способ переключения с одной программы на другую состоит в одновременном нажатии клавиш Alt и Esc. При этом происходит циклическое переключение с одной программы на другую в той последовательности, в которой эти программы запускались.

Если из области экрана Switch to a Running Program выбрать элемент OS/2 Command Prompt (запрос ввода команды OS/2), то вы обнаружите, что он выглядит несколько необычно. Запрос ввода команды [C:\] и белая надпись на синем фоне, очевидно, указывают на то, что вы работаете в среде OS/2 (каталоги DOS и OS/2 выглядят совершенно одинаково). Чтобы возвратиться от запроса ввода команды OS/2 или DOS к работе с Program Selector, нужно одновременно нажать клавиши Ctrl и Esc.

СКОЛЬКО ПРОГРАММ МОЖНО ЗАПУСТИТЬ ОДНОВРЕМЕННО ПОД УПРАВЛЕНИЕМ OS/2?

OS/2 может обрабатывать до 12 программ одновременно, но только одну программу DOS. Программы OS/2 работают в фоновом режиме, а программы DOS приостанавливают работу при переходе в фоновый режим.

В OS/2 есть несколько новых операторов для файла CONFIG.SYS, позволяющих настраивать параметры обработки, объем памяти и приоритеты ввода-вывода при переключении на многозадачный режим

работы. Программы OS/2 будут работать и с приоритетами, установленными по умолчанию, поэтому теоретически операционная система должна обеспечивать работу и без вашего вмешательства. Тем не менее если вы твердо знаете, что нужно делать, то правильное задание параметров для параллельных процессов может повысить производительность системы.

ЧТО ПРОИЗОЙДЕТ, ЕСЛИ ДЛЯ ОДНОВРЕМЕННОГО ЗАПУСКА НЕСКОЛЬКИХ ПРОГРАММ НЕ ХВАТИТ ОПЕРАТИВНОЙ ПАМЯТИ?

Ничего страшного. Помимо многозадачного режима и обеспечения доступа к защищенному режиму наиболее важной особенностью OS/2 является возможность превышения границ оперативной памяти. Когда программа OS/2 выходит за пределы оперативной памяти, операционная система находит неиспользованные фрагменты памяти и объединяет их, создавая сегменты большего объема, в которых можно размещать данные. Если памяти и после этого не хватает, то в действие вступает входящая в состав OS/2 функция реализации виртуальной памяти и начинается подкачка сегментов с жесткого диска. С помощью операторов файла CONFIG.SYS можно задать объем блоков и каталог для хранения этих блоков. Для реализации виртуальной памяти нужно по меньшей мере 512 Кбайт пространства на диске, но при желании можно отвести для этого целый логический дисковод — под виртуальную память можно отхватить и несколько мегабайт, если вас волнует в первую очередь не объем оперативной памяти, а многозадачность. Естественно, что виртуальная память работает медленнее, чем обычная оперативная память.

ЕСТЬ ЛИ В OS/2 СРЕДСТВО, ЭКВИВАЛЕНТНОЕ УТИЛИТЕ VDISK.SYS?

Фактически в OS/2 сохранена утилита VDISK.SYS — удобное средство DOS для реализации псевдодиска в оперативной памяти. Но это еще не все! Наибольший выигрыш пользователи компьютеров PS/2 получат от использования замечательной утилиты, реализующей кэш для диска, вместе с драйвером для "мыши" фирмы IBM. Все также получат программу подкачки для печати (print spooler), которая устанавливается автоматически (вместе с собственным подкаталогом) из программы установки OS/2. Программы OS/2 будут по умолчанию обращаться к программе подкачки, можно обратиться к этой программе и в режиме DOS — для этого нужно одновременно нажать клавиши Ctrl, Alt и Print Screen.

Сведения о командах

Большинство пользователей будут удивлены, обнаружив лишь незначительные различия между работой с OS/2 и работой с DOS. Лишь небольшая группа команд DOS версии 3.30 не воспринимается OS/2 при работе в базовом режиме. Еще меньшее число команд не включено в набор команд, обрабатываемый OS/2 в режиме DOS. Большинство из них настолько редко используется, что средний пользователь никогда с ними не сталкивается. Наиболее важное отличие двух наборов команд составляет небольшая группа команд, специфических для OS/2.

Следующие команды DOS версии 3.30 не воспринимаются OS/2 в базовом режиме:

APPEND — включает в файл CONFIG.SYS DOS оператор PATH, который расширяет маршрут поиска за пределы текущего каталога. Команда DPATH, входящая в набор команд OS/2, выполняет примерно такую же функцию.

ASSIGN — позволяет переназначать имена дисководов в DOS. Наиболее часто эта команда используется для давно разработанных программ, которые обращались к дисководам A и B, а не к дисководу C. В программах OS/2 таких проблем не возникает.

BREAK (с опцией ON) — сообщает DOS о том, что нужно отслеживать все случаи нажатия клавиш Ctrl-Break (не только при работе с экраном, клавиатурой или печатающим устройством). В OS/2 нажатие этих клавиш отслеживается всегда.

COMMAND — служит для запуска командного процессора, эквивалентна команде CMD, входящей в набор команд OS/2.

GRAFTABL — задает страницу кода, которая должна использоваться при свопинге наборов символов. В OS/2 аналогичную функцию выполняет оператор CODEPAGE файла CONFIG.SYS.

JOIN —позволяет заменить имя дисковода именем каталога. В OS/2 это средство переназначения отсутствует, поскольку оно не годится для сетевых структур.

SUBST — позволяет заменить имя каталога именем дисковода. Эта команда используется в основном для давно разработанных программ, которые не работают с подкаталогами. Все программы OS/2 могут работать с подкаталогами.

Следующие команды DOS версии 3.30 не воспринимаются OS/2 ни в базовом режиме, ни в режиме DOS:

CTTY — позволяет переключаться с поддержки стандартных клавиатуры и монитора на поддержку любого другого устройства посимвольного ввода-вывода. От этой команды DOS пришлось отказаться, чтобы избежать конфликтов с программами OS/2 при назначении устройств.

FASTOPEN — копирует каталоги и адреса открытых файлов в оперативную память, что ускоряет поиск файлов DOS. В DOS эта команда была не обязательной (это было сделано для экономии оперативной памяти). Если оперативной памяти достаточно, то можно указанным образом повысить производительность. Благодаря тому, что в OS/2 избыток защищенной памяти, данная команда сделана внутренней.

GRAPHICS — служит для вывода изображения с адаптера цветной графики Color/Graphics Adapter (или совместимого с ним) на одно из пяти графических печатающих устройств фирмы IBM (или совместимое с ними). В OS/2 нет такого средства, поскольку предполагается, что соответствующие функции реализуются в прикладных программах.

NLSFUNC – подготавливает DOS к переключению страниц кода (к свопингу наборов символов). В OS/2 эта команда заменена оператором DEVINFO файла CONFIG.SYS.

SELECT – обычно используется для создания загружаемого диска DOS с файлами CONFIG.SYS и AUTOEXEC.BAT, которые служат для формирования конфигурации системы для конкретной страны. Эквивалентные функции выполняются программой установки OS/2 фирмы IBM.

SHARE – подготавливает DOS для реализации функций разделения файлов в сети и обеспечения сохранности данных в случае возникновения прерываний при чтении или записи данных на гибкий диск. В OS/2 эти функции являются внутренними.

Следующие команды OS/2 не входят в версию 3.30 DOS:

ANSI – позволяет переопределять клавиши, манипулировать курсором и изменять атрибуты цвета дисплея. Действие этой команды аналогично действию оператора DEVICE=ANSI.SYS файла CONFIG.SYS DOS.

CMD – запускает командный процессор OS/2.

CREATEDD – это название является сокращением от слов *create dump diskette* (создать диск дампа). Созданные таким образом диски служат для хранения содержимого оперативной памяти. Диски дампа помогают диагностировать неполадки в системе.

DETACH – отключает программу OS/2 от командного процессора; после этого запуск программы никак не отображается на экране. В OS/2 поддерживаются отдельные фрагменты видеопамяти для каждой прикладной программы, которых может быть до 16. Отключение программы, которая не выводит информацию на экран (например, программы реализации спулинга), предотвращает использование ненужных в данном случае фрагментов видеопамяти.

DPATH – почти идентична команде APPEND, входящей в набор команд DOS и дополняющей файл CONFIG.SYS оператором PATH.

EXIT – прерывает работу текущего командного процессора и осуществляет возврат к предыдущему командному процессору. Если предыдущего командного процессора не было, то происходит возврат в Program Selector.

HELP – реализует доступ к средствам поддержки пользователя OS/2. Ввод этой команды, пробела и уникального номера, которым снабжены почти все сообщения об ошибках, приводит к выводу на экран списка возможных причин возникновения ошибки и рекомендуемых действий.

PATCH – средство программиста для работы с фиксированным кодом. Предназначено для использования совместно с "заплатами", поставляемыми фирмой IBM.

SPOOL – перехватывает вывод на печать из разных прикладных программ; при этом гарантируется, что данные, поступающие из разных программ, не будут перемешаны. Выводимые на печать файлы хранятся в очереди, записанной в специальном подкаталоге.

START – предназначена для использования в файле STARTUP.CMD – принятом в OS/2 эквиваленте файла AUTOEXEC.BAT. Строка файла STARTUP.CMD, содержащая команду START и команду запуска программы OS/2, задает загрузку этой программы. Несколько таких строк обеспечивают автоматическую загрузку нескольких программ при запуске системы.

TRACE – обеспечивает отслеживание заданных событий (например, открытие файла) и запись результатов в буфер для диагностических целей.

TRACEFMT – считывает содержимое буфера трассировки и выводит его на экран. Можно переадресовать вывод содержимого буфера трассировки в файл или на печатающее устройство.

ИМЕЕТСЯ ЛИ ДОКУМЕНТАЦИЯ ПО OS/2, КОТОРАЯ БЫЛА БЫ ЛУЧШЕ ДОКУМЕНТАЦИИ ПО DOS?

Это, безусловно, 50-страничное руководство для пользователя (почти в три раза меньшее по объему аналогичного руководства по DOS версии 3.30), которое продублировано во встроенных в OS/2 средствах обучения пользователя, а также справочник, снабженный рисунками, иллюстрирующими синтаксис каждой команды. Лучше всего то, что теперь описание команд написано хорошим языком.

Однако наибольшее впечатление производят встроенные средства обучения пользователя. Если нажать функциональную клавишу F1 в любой момент работы с Program Selector, то на экран будет выведена соответствующая данному контексту вспомогательная информация. Более того, чтобы получить полное описание ошибки (в OS/2 сообщения об ошибках сопровождаются числовым кодом), нужно ввести с клавиатуры слово help, нажать клавишу пробела, ввести числовой код ошибки и нажать клавишу Enter. На экран будет выведен список возможных ошибок, допущенных вами или системой, а также четкие рекомендации о том, как следует поступать. Очень немногие программы располагают такими удобными средствами.

ЕСЛИ НЕ ЗАПУЩЕНА НИ ОДНА ИЗ ПРОГРАММ OS/2, ТО БУДУТ ЛИ ПРОГРАММЫ DOS РАБОТАТЬ МЕДЛЕННЕЕ, ЧЕМ ПОД УПРАВЛЕНИЕМ DOS?

Неформальная проверка показала, что скорость практически не изменяется. Это означает, что если нужной вам программы OS/2 у вас нет, а перейти в новую среду нельзя, то вы ничего не потеряете, переключившись в режим DOS (при условии, что нужная программа DOS работает с блоком совместимости).

ГДЕ ПРИОБРЕСТИ OS/2:

IBM
P. O. Box 1328-W
Boca Raton, FL 33429
305/998-2000

Цена: 295 дол.

Требования для установки: 1,5 Мбайт памяти, жесткий диск.

Зачем покупать *PS/2*?

ЭРИК БЕНДЕР

Ведущие специалисты фирмы IBM объясняют причины, по которым они считают линию ЭВМ Personal Systems/2 идеальной для будущего программного обеспечения.

Если линия Personal Systems/2 действительно представляет собой следующее поколение персональных ЭВМ, то как объяснить, что различные тесты на производительность, проведенные фирмой National Software Testing Laboratories и другими группами пользователей, все как один показывают, что новые машины фирмы IBM ничем не выделяются среди других ЭВМ?

Главный конструктор линии PS/2 Деннис Эндрюс считает, что оценка производительности его ЭВМ на сегодняшнем программном обеспечении несостоятельна, поскольку при этом игнорируются реальные потребности пользователей: "По мере усложнения приложений вам вовсе не важно, насколько быстро исполняется пакет программ Lotus сам по себе. В действительности вам требуется знать, сколько времени у вас отнимает реальная работа с пакетом Lotus, в процессе которой придется обмениваться файлами с другими приложениями, делать заметки в электронном календаре и выполнять фоновые программы".

По словам Эндрюса, директора лаборатории Отдела интерактивных систем фирмы IBM, находящейся в Бока-Рейтон, линия PS/2, в отличие от классической линии ЭВМ PC, изначально рассчитана на режим параллельного исполнения программ. Он добавляет, что не пройдет и нескольких лет, как новая микроканальная архитектура MCA (Micro Channel Architecture), положенная в основу PS/2, с блеском проявит себя в условиях "многопользовательской среды, содержащей сложные центральные ЭВМ, шлюзы и прочие компоненты, которые требуют наличия дополнительных процессоров или сложных подсистем".

ЭВМ PS/2 моделей 50 и 60 с MCA-шиной начали поставляться в апреле 1987 г., а в конце лета третьи фирмы стали выпускать для них дополнительные платы. Однако большинство выгод от применения новой архитектуры не проявится до 1988 г., в течение которого ожидается появление прикладных программ, рассчитанных на операционную систему Operating System/2 (OS/2).

Как указывает главный разработчик MCA-архитектуры Чет Хит, в настоящей мультипрограммной среде "каждая программа может иметь свой режим ввода-вывода, что влечет за собой увеличение общего числа операций, связанных с организацией ввода-вывода. Из-за этого, во-первых, конфигурации системы усложняются, во-вторых, число прерываний в 1 с возрастает. Мы должны справиться с обеими проблемами."

Этот возрастающий поток прерываний, инициируемых каждым ресурсом системы, создаст серьезные проблемы для шины IBM AT. Трудности возникли уже сейчас, даже при работе с такими простыми компонентами, как последовательные порты на системной плате.

Например, в настоящее время для исключения конкуренции двух последовательных портов за прерывания COM1 "приходится конструировать схемы разделения прерываний, которые достаточно дороги и имеют тенденцию создавать помехи. Все это значительно усложняет систему". Но в MCA-архитектуре все системные прерывания разделяются без дополнительных схем, что исключает добавочные электромагнитные помехи.

В настоящее время манипулирование DIP-переключателями для исключения конфликтов между адресами векторов прерываний, адресами областей оперативной памяти и уровнями прерываний становится настоящим бедствием. По словам Хита, "в мультипрограммной среде эта проблема усложняется на порядок. Поверьте, пользователям придется рвать на себе волосы. Мы должны с этим что-то сделать".

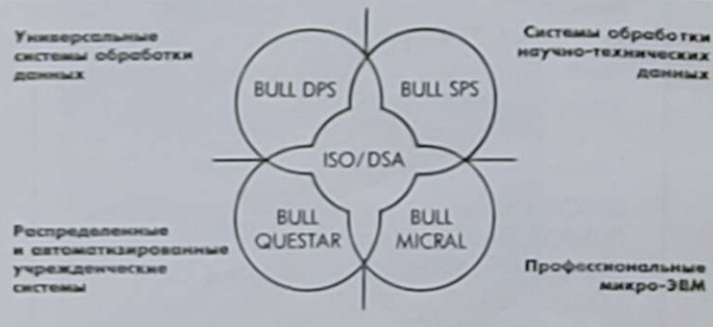
По словам создателей, новая архитектура шины PS/2 – это всего лишь фрагмент большого комплекса, включающего в себя аппаратуру, операционную систему и прикладные программы и предназначенного фирмой IBM для обеспечения мультипрограммирования в будущем. К числу обещанных выгод перехода на PS/2 относятся повышенная надежность, четкие спецификации солидной аппаратной и программной базы, а также ряд новых возможностей, например обеспечение параллельной работы нескольких процессоров. Наконец, новая архитектура создает потенциал для разработки более компактных конструкций и повышения производительности за счет традиционного увеличения тактовой частоты.

НОВЫЕ "ВИТКИ" ДЛЯ ДРАЙВЕРА ШИНЫ

Операционная система OS/2, одна из наиболее уязвимых основ линии PS/2, должна обеспечивать намного более развитый мультипрограммный режим по сравнению с тем, что имеется на IBM PC. При этом многие прикладные программы не только смогут исполняться параллельно и не мешая друг другу, но и получат возможность параллельного исполнения некоторых внутренних процедур, что ускорит их работу. На жаргоне разработчиков OS/2 программы образованы из одного или нескольких процессов, каждый из которых, в свою очередь, составлен из одного или нескольких "витков" (этим термином называется наименьшая самостоятельно исполняемая группа команд). "Витки" захватывают системные ресурсы для выполнения таких функций, как выдача на экран или доступ к жесткому диску. Таким образом, всю систему можно было бы назвать "многовитковой средой".

Поскольку все "витки" должны делить процессорное время и выдавать запросы на предоставление системных ресурсов, то выполняемые OS/2 операции становятся намного сложнее, чем это имеет место в "одновитковой" DOS. Хотя шина IBM AT и сможет обеспечить выполнение этих операций, но все-таки они больше рассчитаны на MCA-шину ЭВМ PS/2. *(Окончание на с.49)*

BULL
ДЕРЕВО КОММУНИКАЦИИ



Фирма BULL, занимающая одно из ведущих мест в мире в области разработки и изготовления информационно-вычислительных комплексов и автоматизированных учрежденческих систем, предлагает промышленным предприятиям и фирмам широкий выбор возможностей в четырех важнейших областях вычислительной техники.

В области универсальных систем обработки данных BULL выпускает широкий ассортимент малых, средних и больших систем, в котором самая высокая модель в 40 раз мощнее первой модели. Для всех вариантов ЭВМ имеется одна операционная система GCOS, что позволяет уменьшить затраты пользователя на программное обеспечение.

В области ЭВМ промышленного и научно-технического назначения фирма BULL предлагает серию усовершенствованных мощных мини и супермини-ЭВМ, предназначенных в основном для инженерных расчетов и решения научно-исследовательских задач.

Терминалы и рабочие станции многоцелевого назначения, выполненные с учетом требований современной эргономики, разработаны для распределенных информационно-вычислительных и автоматизированных учрежденческих систем.

В области профессиональных микро-ЭВМ фирма BULL предлагает серию высокопроизводительных машин индивидуального и коллективного пользования, совместимых с различными промышленными стандартами.

Архитектура сети ISO/DSA, соответствующая международным стандартам, обеспечивает коммуникацию всех этих систем в пределах однородных или смешанных сетей.

BULL. Дерево коммуникации.

Представительство в СССР : BULL S.A. ул. Конюшковская, 28 123242 МОСКВА - Телефон : 253 97 13 - Телекс : 413 574 BULL SU.

Bull

Победитель конкурса поставщиков программного обеспечения

интересуется программными продуктами, которые можно покупать или продавать в СССР

Нам хотелось бы расширить объем поставляемых программных продуктов за счет выпуска программного обеспечения для MS-DOS и создания новых каналов сбыта для наших пакетов XTreePro, XTree, XTreeNet и Hot.
Компания Executive System, Inc., дочерней фирмой которой мы являемся, специализируется в разработке программ для изготовителей технических средств. Мы заинтересованы в партнерах, занимающихся продажей BIOS, утилит и прикладных программ. Нашей задачей является также разработка специализированной операционной среды для компьютера Headstart фирмы Vendex — победителя конкурса журнала PC Magazine.

Нашими клиентами являются фирмы:

EPSON AMERICA, ARCO WESTERN DIGITAL CORPORATION, EMERALD SYSTEMS, TALLGRASS TECHNOLOGIES, VENDEX TECHNOLOGIES, SPERRY-UNIVAC . . .

Дополнительную информацию и ответы на вопросы вы можете получить по адресу:

XTREE COMPANY

XTree Company
4330 Santa Fe Road
San Luis Obispo, CA 93401
USA

Tel #: (805) 541-0604
Telex #: 910-2503-461
FAX #: (805) 541-8053

ОСНОВНЫЕ ХАРАКТЕРИСТИКИ

DataEase (Version 2.5)
Однопользовательская
Совместима с IBM PC, PC-XT, PC-AT, PS/2 и большинством клонов
- Wang Professional
- Victor
- Burroughs

Предполагаемая операционная система:
- PC-DOS или MS-DOS, Version 2.1 или старше
- ПЗУ 512 К как минимум
- 2 дисковода (рекомендуются жесткие диски)

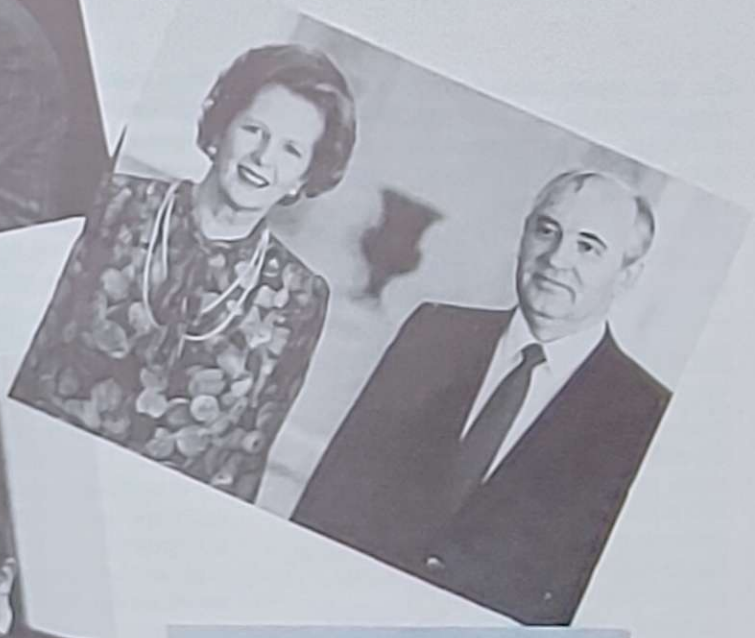
Цветной или монохромный дисплей
Полная поддержка принтеров
До 26 различных применений на один каталог
До 255 файлов на базу данных
До 255 сообщений на базу данных
До 100 активных связей
До 32 файлов, открытых одновременно
До 16 экранов на форму или на экран ввода данных
До 255 символов на поле
До 255 полей на запись
До 65.535 записей на файл
Индексация по B-дереву
Поиск (частичный или полный) в файле по имени или его части
Сортировка или группировка на любое число уровней
Тип поля — 99 стандартных типов и возможность издания своих типов (имена — до 60 символов)
До 255 индексов на файл
Большие текстовые поля с переносом слов

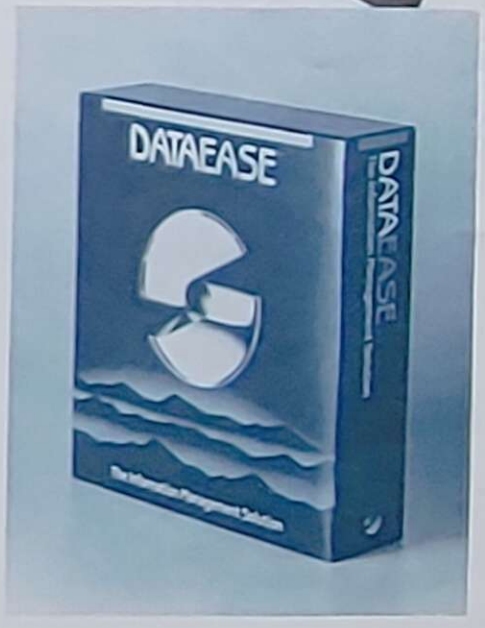
Форматы импорта:

DataEase®	Variable Length
Lotus	Fixed Length
dBase II & III	Mail Merge
DIF	

Форматы экспорта:

Lotus	Variable Length
DIF	Fixed Length
MultiMate	GrafTalk®
Mail Merge	





Ещё один международный лидер прибыл в СССР

Вы знаете, что значит быть лидером? В случае реляционной СУБД DataEase® это означает уникальное сочетание мощности и простоты использования. Реляционная СУБД DataEase, предназначенная для использования на персональных компьютерах фирмы IBM®, стала стандартным средством для хранения и манипулирования информацией в бизнесе, промышленности, сельском хозяйстве, образовании, медицине. Это великолепный инструмент с помощью которого пользователи и программисты могут создавать различные прикладные системы, например, для управления производством, для автоматизации учрежденческой деятельности, для ведения отчетности и управления поставками и для руководства реализацией проектов.

Реляционная СУБД DataEase так легка в использовании, что создавать с ее помощью прикладные системы могут даже те, кто не имеет предварительного опыта использования компьютеров. Реляционная СУБД DataEase так мощна, что опытные программисты с ее помощью могут легко и быстро создавать сложные прикладные системы. Поэтому реляционную СУБД DataEase используют такие фирмы, как IBM®, Ford®, Exxon®, Barclay's Bank®, GTE®, BMW® & Siemens®, Citibank®, United Airline® и Wang. Реляционная СУБД DataEase действительно является лидером среди СУБД. И этот лидер едет в СССР. За более подробной информацией обращайтесь по телексу 703972 или по телефаксу 203-372-4498 или по телефону 280-13-31 в московское представительство А/О "Видеотон", которое представляет наши интересы.

DataEase
INTERNATIONAL
Seven Cambridge Drive, Trumbull, CT 06611

МОЩНЫЙ И МНОГОКРАСОЧНЫЙ

Как ни странно, многим приходилось иметь дело с тусклыми экранами, на которых буквы и цифры изображаются одним цветом. При этом они считают, что это — вполне приемлемый режим работы. Однако вы, наверное, замечали, что жизнь полна красок, меняющихся картин картин и звуков.

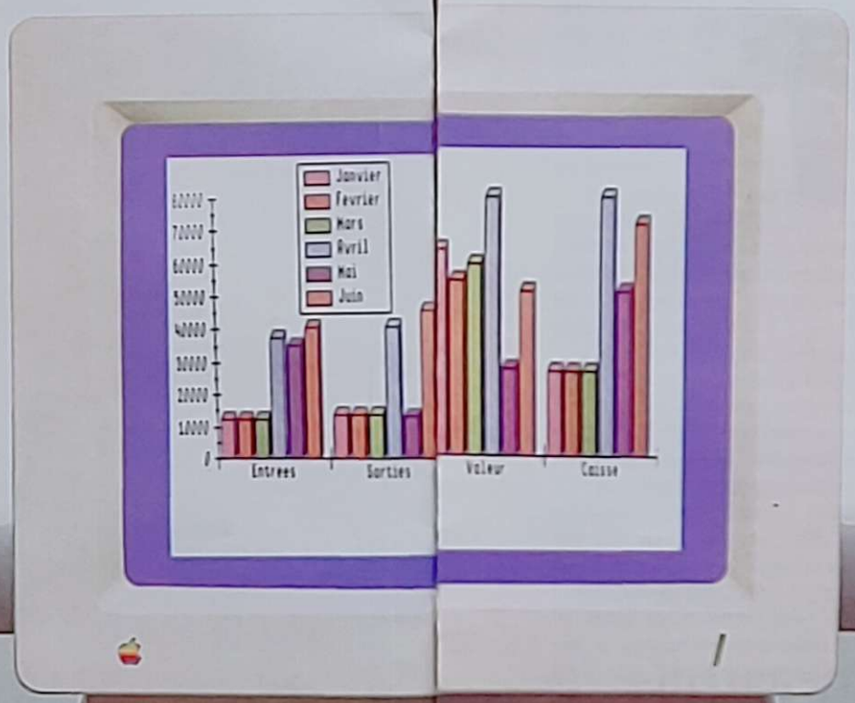
Мы серьезно работали над графикой, красками и звуком.

Разрешающая способность экрана компьютера AppleIIGS 640 X 200 точек для четырех цветов и 320 X 200 точек для 16 цветов из палитры в 4096 оттенков. Надеемся, что этого вполне достаточно. Можно вставлять рисунки в текст или вносить текст в предварительно составленное или ошифрованное изображение.

А если подключить звуковой сопроцессор ENSONIQ, способный генерировать 16 звуков разной высоты, то компьютер "заговорит".

Он также позволит вам увидеть все многообразие красок.

Сердцем Apple IIGS является 16-битовый микропроцессор 65C816, созданный на базе микропроцессора 65C02 компьютера Apple II. Компьютер Apple IIGS работает на двух тактовых частотах: 2,8 МГц в обычном режиме и 1 или 2,8 МГц в режиме эмуляции (если вы хотите использовать программы для компьютера Apple II). С помощью контроллера IWM к компьютеру Apple IIGS можно подключать любое число дисководов для 3,5-дюймовых дисков емкостью 800 Кбайт или 5,25-дюймовых дисков.



емкостью 140 Кбайт, а также подключать дисководы обоих типов одновременно, что позволит использовать самую большую в мире библиотеку программ. Очень приблизительное число таких программ — 16 000, и единственное, чего не может подсчитать компьютер Apple IIGS, в скольких случаях он будет вам полезен.

Память хотя и не имеет красок, но практически не ограничена.

Оперативная память компьютера Apple IIGS наращивается до 8 Мбайт.

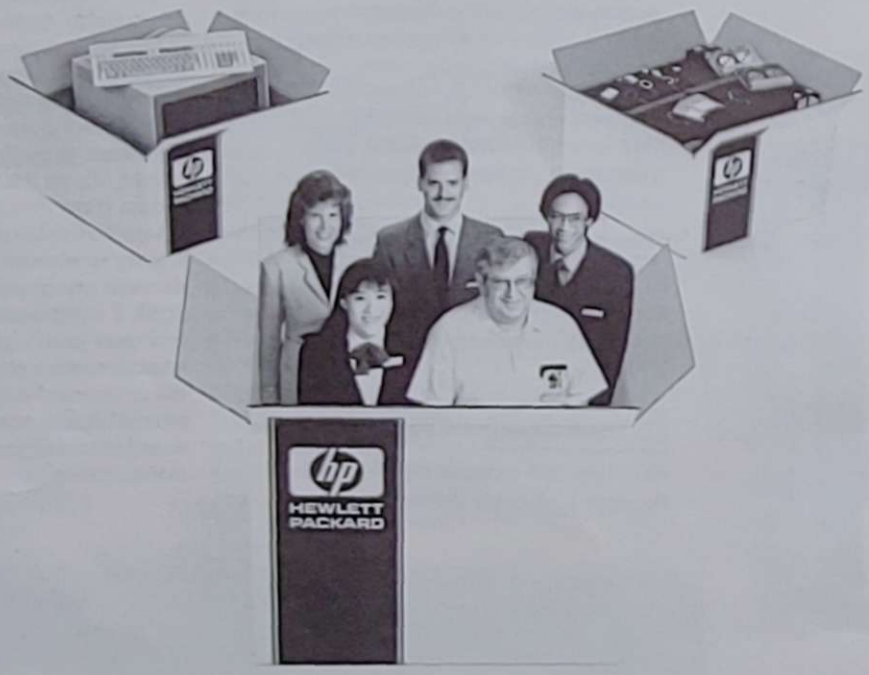
Для успеха необходима связь с внешним миром.

За 10 лет для компьютера Apple II были разработаны отличные интерфейсы с печатающими устройствами, модемами, жест дисками, которые с появление компьютера Apple IIGS будут развиваться и совершенствоват Компьютер Apple IIGS с успехо можно подключить к сети Apple Talk и с его помощью мо управлять устройством Laser Writer, что до последнего врем можно было делать только с помощью компьютера Macintos К нему можно также подключат жесткие диски через интерфейс SCSI. У компьютера Apple IIGS есть семь разъемов для подключения внешних устройст что позволяет через несколько интерфейсных плат подключать периферийные устройства любых типов.

Фирма Apple представляет

модель Apple IIG

HEWLETT-PACKARD
ПРЕДЛАГАЕТ ПРОГРАММУ ТРЁХГОДИЧНОГО БЕСПЛАТНОГО ОБСЛУЖИВАНИЯ



Держаться на плаву и преуспевать – не одно и то же

Держаться на плаву в современном мире конкуренции означает с помощью новой техники повышать производительность и качество. Однако, чтобы преуспевать, нужны еще опыт, знания и люди, заставляющие эту технику работать, – нужно обслуживание.

Фирма Hewlett-Packard не ограничивается гарантией новизны и высочайшего качества продукции.

С 1 апреля 1988 г. мы поставляем современные технические средства в соответствии с программой "Три года бесплатного обслуживания", чтобы вам было удобнее с ними работать. Вот уже более 15 лет мы с вами в Болгарии, ЧССР, ГДР, Венгрии, Польше, Румынии, СССР и Югославии.

Программа фирмы Hewlett-Packard "Три года бесплатного обслуживания" позволит вам не только держаться на плаву, но и преуспевать, а это – не одно и то же.

МЫ ВСЕГДА С ВАМИ HEWLETT PACKARD

Разработка и применение программных средств ПЭВМ в учебном процессе

И. А. БЕЛАЯ, Ю. К. ШТЕЙН

С 21 по 23 сентября 1988 г. на базе Симферопольского государственного университета проходил IV Всесоюзный семинар "Разработка и применение программных средств ПЭВМ в учебном процессе". Для участия в его работе было приглашено около 120 специалистов, представляющих АН СССР и АН союзных республик, Госкомитет СССР по народному образованию, АПН СССР, а также другие организации, хотя заявок было около пятисот. Это говорит о большом интересе к семинару, об актуальности рассматриваемых на нем вопросов.

Работало пять секций. Было сделано более 60 сообщений. К началу работы семинара сотрудниками ИПИ АН СССР был подготовлен сборник тезисов докладов "Материалы IV Всесоюзного семинара "Разработка и применение программных средств ПЭВМ в учебном процессе"."

Большой интерес вызвало обсуждение проблем создания систем учебной информатики. О наиболее актуальных из них мы попросили рассказать начальника отдела учебной информатики ИПИ АН СССР, кандидата физико-математических наук С. А. Христочевского.

"В области учебной информатики до спокойного положения еще далеко, – сказал он. – В первую очередь это связано с тем, что нет установившейся программы обучения основам информатики в школах и ПТУ (в технических вузах дело обстоит несколько лучше), поскольку до сих пор идет дискуссия о том, что делать: учить программированию или алгоритмическим языкам или нужен курс ликвидации компьютерной безграмотности.

Для использования ПЭВМ в качестве средства обучения еще не накоплено достаточного опыта, нет данных об эффективности использования ПЭВМ в сфере образования и отсутствуют обоснованные педагогические требования к ним. Положение усугубляется тем, что до сих пор в школах нет соответствующих задачам обучения ПЭВМ.

Несмотря на это практически всеми высказывается мнение, что уникальные качества ПЭВМ делают их независимыми в сфере образования.

Создаваемые программные и аппаратные средства должны быть частью системы ПЭВМ для народного хозяйства страны. Ранее мы чаще всего шли от производителя к пользователю, т. е. предлагали готовый продукт без учета специфических потребностей пользователей. Учитывая массовый характер компьютеризации (ведь речь идет о нескольких миллионах ПЭВМ для учебных целей), необходимо идти от пользователя, разрабатывая новые средства в соответствии с предполагаемым использованием.

Отсюда вытекает следующая программа действий:

изучение требований пользователя;

разработка концепции применения средств вычислительной техники в сфере образования;

технико-экономический анализ и прогноз;

разработка критериев аттестации программно-технических комплексов;

разработка требований к ПЭВМ, периферийным устройствам, программному обеспечению (системные, инструментальные средства);

выпуск компонентов системы учебной информатики.

При создании систем учебной информатики важно учитывать психолого-педагогические аспекты компьютерной технологии обучения".

При рассмотрении инструментальных средств для систем учебной информатики основное внимание было уделено диалоговой системе структурированного программирования, инвариантным инструментальным средствам разработки активных методов обучения, наполняемой обучающей системе, позволяющей учителю конструировать конкретные уроки, а также применению машинной графики в системах учебной информатики и некоторым другим разработкам.

Ряд докладов был посвящен разработке программной поддержки курса "Основы информатики и вычислительной техники", вопросам тиражирования и использования учебных программ.

В процессе обсуждения новых аспектов применения вычислительной техники в учебном процессе наибольший интерес вызвали сообщения об "обучающих средах", о разработке учебно-научных деловых игр на базе ПЭВМ и их использовании на уроках русского языка, обществоведения, физкультуры, о применении ПЭВМ в экспериментальном интегрированном курсе "Природа", о синтезе игровой и обучающей компонент в педагогических программных средствах, об использовании теории алгоритмизации обучения при составлении компьютерных обучающих программ по математике и ряд других сообщений.

О перспективных направлениях использования ПЭВМ в учебном процессе рассказал старший научный сотрудник НИИ ШОТСО АПН СССР кандидат педагогических наук И. В. Роберт: "В настоящее время применение ЭВМ в учебном процессе уже не может ограничиваться обучением программированию на одном или нескольких языках либо использованием одних только программных средств при обучении тому или иному общеобразовательному предмету.

Будущее, прежде всего, за качественно новыми средствами обучения, объединяющими в себе программы и технические устройства, имитирующие разнообразные промышленные механизмы и приспособления (учебные роботы, модели механизмов, управляемые с помощью ЭВМ). Такие средства обучения обеспечивают демонстрацию возможностей современных ЭВМ в сфере управления объектами реальной действительности, обучение составлению программ управления учебными роботами, профориентации учащихся.

*) "Моя мечта:
персональный компьютер – в каждый дом и на каждый рабочий стол. С момента создания "Майкрософт„ миссией фирмы было предоставить каждому возможность познакомиться с удобствами, которые дает пользование компьютером! Создав основу MC-ДОС для более чем 20-ти миллионов персональных компьютеров, мы обеспечили программы, превращающие компьютер в инструмент творчества, приобретения знаний и улучшения эффективности. Повсюду в мире.„

PC SOFTWARE: YESTERDAY, TODAY AND TOMORROW!

FROM THE MIRACLE FOR A FEW TO THE STANDARD FOR ALL.

How many Personal Computers are being used around the world? 500.000? 1 million? 10 million? Or even more?

The answer is: over 20 million PC's are already working around the globe. And every month 500.000 more are being put into work.

Just a few years ago computers were used only by experts or enthusiasts. A confusing number of different operating systems was offered, with only a few software packages available. Software developers did not know which operating system might have a future and would be worth it to develop respective applications. Accordingly, users suffered from lack of good software.

1981. THE INTRODUCTION OF IBM'S PC CHANGES THE ENTIRE INDUSTRY.

All this changed in 1981, when IBM introduced its Personal Computer. And

IBM's PC used Microsoft's MS-DOS operating system. Soon a fast growing number of hardware manufacturers licensed Microsoft's MS-DOS. Within no time a standard operating system was born: MS-DOS from Microsoft.

This standard was a solid platform for the development of a wide range of application software, thus laying the foundation for the PC's tremendous success. – Today more than 20 million PC's are running under MS-DOS, with compatability guaranteed and available only from Microsoft.

MS-DOS PUTS PC'S ON THE ROAD TO SUCCESS.

With the extensive use of application software it became obvious that similar interfaces for different application types were necessary to reduce learning time for a new product. In order to make users quickly familiar with their software a computer based training program was offered as part of the application as well as Microsoft's Mouse support. The Mouse support helps to reduce the necessary keystrokes up to 60%. All one needs to do is, to point at what should be marked or activated.

Microsoft has been spending quite some time to indentify users' needs and whishes. The results can be found in the entire application family: Microsoft Word, the word processor – Microsoft Multiplan, the spreadsheet or Microsoft Chart, the business graphic. Each of these products has the same interface, a computer based training program and a Mouse support. After all, application software should increase working efficiency without consuming too much learning time.

EASY TO USE AND A VAST VARIETY OF FEATURES: THE MICROSOFT APPLICATION FAMILY.

An easy use of Microsoft's application, however, is one side only. The other side, and just as important, is the huge amount of features Microsoft has to offer.

Microsoft Word, the word processor, includes for example, a document retrieval system and the ability to put graphics into the copy.

Microsoft Multiplan, the spreadsheet, offers the possibility to work with 8 different tables and macro-language.

And with Microsoft Chart, the business graphic, business or production figures can be shown with a 3D effect.

Data exchange is another important issue. Whether tables from the spreadsheet should be brought into business graphic or tables and/or graphics into the word processing system. With the Microsoft application family all this is quite simple.

TOMORROW'S APPLICATIONS: EASY TO USE AND TO LEARN THROUGH GRAPHIC SYMBOLS - AND MORE POWERFUL THAN EVER.

Tremendous improvements on the hardware side offer new opportunities for software. A graphical interface is just one example. And Microsoft has set a milestone for the future with the development of Windows, the graphical user interface for MS-DOS.

Naturally enough, a new hardware architecture is also a challenge for the operating system. Therefore once again a new operating system is being introduced. MS-OS/2. And once again it is from Microsoft.

MS-OS/2. THE OPERATING SYSTEM OF THE FUTURE. BY MICROSOFT.

Microsoft's MS-OS/2 is a multitasking operating system. It takes full advantage of the 286 processor and offers users up to 16 MB memory. MS-OS/2 has just started its career and practically all important hardware manufacturers have committed themselves already to use this new operating system. The first MS-OS/2 applications will probably be introduced at the end of 1988. And as for MS-DOS again Microsoft's Windows will be the graphical user interface and the platform for a new generation of application software.

The increasing hardware performance offers new and exciting possibilities for the creation of even more advanced software. Microsoft's founder and chairman, Bill Gates, once said „my vision is to see a PC on every desk". Today 20 million PC's are already installed and in 1989 the 30 million mark will probably be reached. So we are not that far away anymore to see this vision fulfilled. And if it comes to the future of software for Personal Computers, it comes to Microsoft.

Microsoft®

FUTURE OF SOFTWARE

COUPON

I would like to have further information about Microsoft's software.

My profession: _____

I am interested in: ☐ operating system ☐ wordprocessor ☐ spreadsheet ☐ business graphic ☐ database

Please send this coupon to: Microsoft Deutschland GmbH, Erdinger Landstraße 2, 8011 Aschheim-Dornach, West-Germany. PCW 2/8

Принципиально новым компонентом учебной деятельности становится работа со средствами пространственного ввода и манипулирования текстовой и графической информацией. К ним относятся, например, манипуляторы типа "мышь", планшеты, световое перо. Эти средства обучения позволяют демонстрировать возможности современных ЭВМ в сфере обработки информации и изучать сущность происходящих в ней явлений.

Другое перспективное направление – использование датчиков и устройств для измерения некоторых физических величин (например, светового потока, температуры, кислотности среды) совместно с устройствами, обеспечивающими ввод в ЭВМ, а также вывод аналоговых и дискретных сигналов, считаемых датчиками. С помощью таких устройств, подключаемых к ЭВМ, или оборудования на их базе становится возможным представление на экране различных физических закономерностей в виде графиков, динамически изменяющихся в зависимости от изменения входных параметров. Таким образом, учащиеся смогут создавать модели изучаемых процессов, "проигрывать" их поведение и развитие при различных условиях, прогнозировать развитие процессов и осуществлять с помощью ПЭВМ проверку достоверности прогнозов. Становится возможным на исследовательском уровне проводить лабораторные работы, демонстрационный лабораторный эксперимент по основам наук, изучать развитие процессов, протекающих в реальной жизни.

Реализация этих направлений позволит обучать самостоятельному проникновению в закономерности изучаемой науки, т. е. перевести процесс обучения с уровня "сообщение суммы знаний – усвоение суммы знаний" на уровень "исследовательский подход и прогнозирование"."

В докладах, представленных на секции "Интеллектуальные системы в обучении", были рассмотрены различные аспекты разработки обучающих программ с использованием методов и средств искусственного интеллекта. В настоящее время проводятся работы по созданию инструментальных средств экспертно-обучающих систем для ПЭВМ, методик построения базы знаний интеллектуальных обучающих систем, интеллектуальных систем решения задач по их постановкам.

Было отмечено, что не все рекомендации III Всесоюзного семинара, проходившего в марте 1987 г. в г. Пущино, выполнены. До сих пор не решен вопрос об авторских правах на программные средства, не получены предложения от участников семинара по архитектуре и базовому программному обеспечению учебных ПЭВМ.

Большой интерес вызвали доклады представителей ВНТК "Школа-1" АН СССР, НИИСчетмаш, Северо-Осетинского государственного университета, Марийского государственного университета, МИЭМ, МГУ, НИИВШ, ИПИ АН СССР, ИК АН УССР, Кишеневского государственного университета и многих других.

Участники семинара провели дискуссию за круглым столом по вопросу "Нужен ли Пролог в учебном процессе?"

Было обращено внимание на необходимость усиления фундаментальных исследований в области создания перспективных средств вычислительной техники, программного обеспечения и массового внедрения результатов этих исследований в систему образования всех уровней, на необходимость координации работ в области учебной информатики.

В итоговом документе семинара были сформулированы следующие рекомендации:

1. Просить Госкомитет СССР по народному образованию рассмотреть вопрос о первоочередных поставках инструментальных комплексов учебной вычислительной техники (КУВТ) MSX-2, УК НЦ и "Корвет" разработчикам программных средств и научным организациям, ведущим исследования в области учебной информатики.

2. Обратить внимание ГКВТИ СССР, МНТК "ПЭВМ", Министерства электронной промышленности СССР, Министерства радиопромышленности СССР и ряда других министерств на серьезное отставание в оснащении учебных заведений средствами вычислительной техники и низкое качество поставляемой техники, а также на недопустимость поставки КУВТ-86 на базе БК-0010 в учебные заведения, поскольку по своим технико-программным возможностям он не отвечает требованиям, предъявляемым к учебным КУВТ.

3. В связи с тем, что развитие перспективных систем учебной информатики сдерживается ограниченными возможностями выпускаемых в настоящее время учебных ПЭВМ, выдвинуть в качестве первоочередной задачи разработку новых перспективных моделей ПЭВМ.

4. Обратить внимание Госкомитета СССР по народному образованию на следующие первоочередные работы для финансирования их в рамках госзаказа:

- разработка разнообразных обучающих средств для пользователей-непрофессионалов в области вычислительной техники;
- стандартизация и унификация систем учебной информатики;
- разработка и внедрение "электронной доски";
- разработка коммуникационных средств с привлечением соответствующих министерств и ведомств;
- разработка перспективных средств новых информационных технологий (видеокомпьютерные системы, компьютерные лаборатории и т. п.).

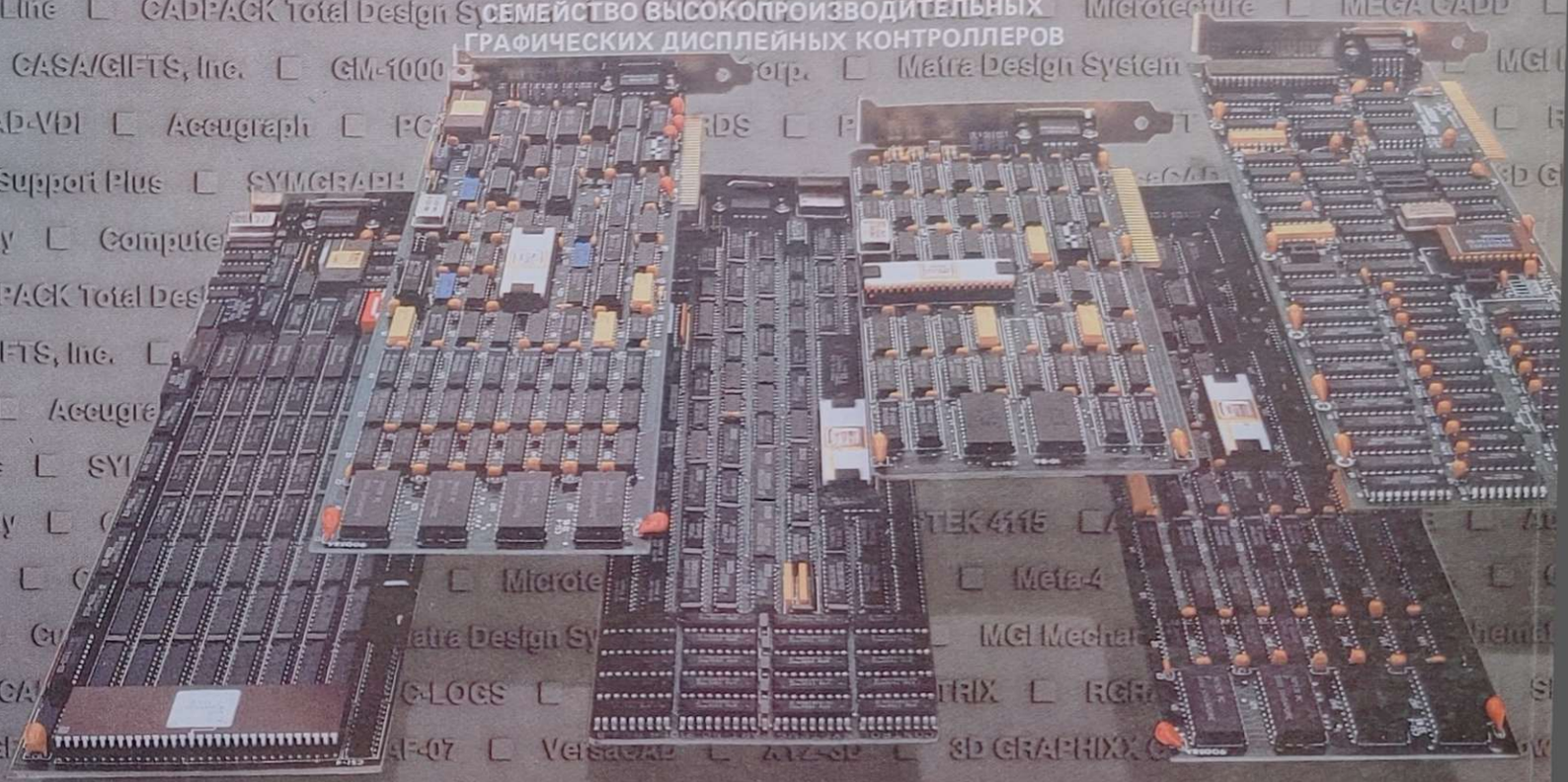
5. Предусмотреть разработку аппаратных и программных средств в серийно выпускаемых ПЭВМ для профессионального обучения в средних специальных учебных заведениях, ПТУ, системе повышения квалификации, а также для обеспечения учебного процесса в национальных школах (возможность работы на языках народов СССР).

6. Учитывая разобщенность исследований, проводимых различными организациями в области разработки и применения систем учебной информатики, создать межведомственный центр, решающий такие задачи:

- обобщение опыта создания и применения средств учебного назначения;
- критический анализ разрабатываемых и внедряемых средств учебной информатики;
- установление приоритетности разработки систем учебной информатики;
- выявление перспективных направлений использования новых информационных технологий в учебном процессе;
- координация работ, проводимых различными ведомствами;
- определение единой технической политики в области использования новых информационных технологий в образовании.

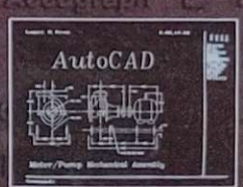
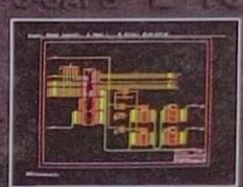
“Контрол Системс” представляет семейство “АРТИСТ”

СЕМЕЙСТВО ВЫСОКОПРОИЗВОДИТЕЛЬНЫХ ГРАФИЧЕСКИХ ДИСПЛЕЙНЫХ КОНТРОЛЛЕРОВ



ARTIST 10/16	ARTIST 1 PLUS	ARTIST 8	ARTIST	ARTIST 1	ARTIST 1/EGA
1024 X 1024	1024 X 1024	800 X 800	Monochrome	1024 X 768	1024 X 768
16 цветов	16 цветов	16 цветов	1024 X 768	16 цветов	16 цветов
1 или 2 экрана	2 экрана	1 или 2 экрана	4 тона	2 экрана	2 экрана
			2 экрана		

СВЫШЕ 200 ПАКЕТОВ ПРОГРАММ РАБОТАЕТ НА НАШИХ КОНТРОЛЛЕРАХ

Трехмерные тоновые изображения	САПР в строительстве	Художественная графика	САПР в машиностроении	САПР в электронике	Деловая графика
					
Пакет Personal Designer Software фирмы Computervision	Пакет CADVANCE фирмы CalComp	Пакет ARTBRUSH фирмы Control Systems	Пакет AutoCAD фирмы Autodesk	Пакет RGRAPH фирмы APTOS Systems	Пакет Speaker Support PLUS фирмы Meta-4, Inc.

ОБРАЩАЙТЕСЬ К НАМ ЗА ДОПОЛНИТЕЛЬНОЙ ИНФОРМАЦИЕЙ

Телефон: 612 631 7800
Телекс: 756601 CNTRLYS UD
Фототелеграф: 612 631 7802

Control Systems

2675 Patton Road
P.O. Box 64750
St. Paul, MN 55164

4 миллиона доказательств того, что фирма SPC является одним из ведущих разработчиков программного обеспечения

Сегодня более четырех миллионов пользователей во всем мире эксплуатируют программые продукты из широкого ассортимента предлагаемого фирмой Sofware Publishihg Corporation.

Есть и другие доказательства – наши пакеты: 1. Harvard Graphics – деловая графика. 2. Professional Write – обработка текста. 3. PFS First Choice – интегрированный пакет для начинающий пользователей. 4. Harvard Project Manager – управление проектами. 5. PFS: Professional File – управление файлами. 6. PFS: First Publisher – издательская система. 7. PFS Professional Plan – бухгалтерская система.

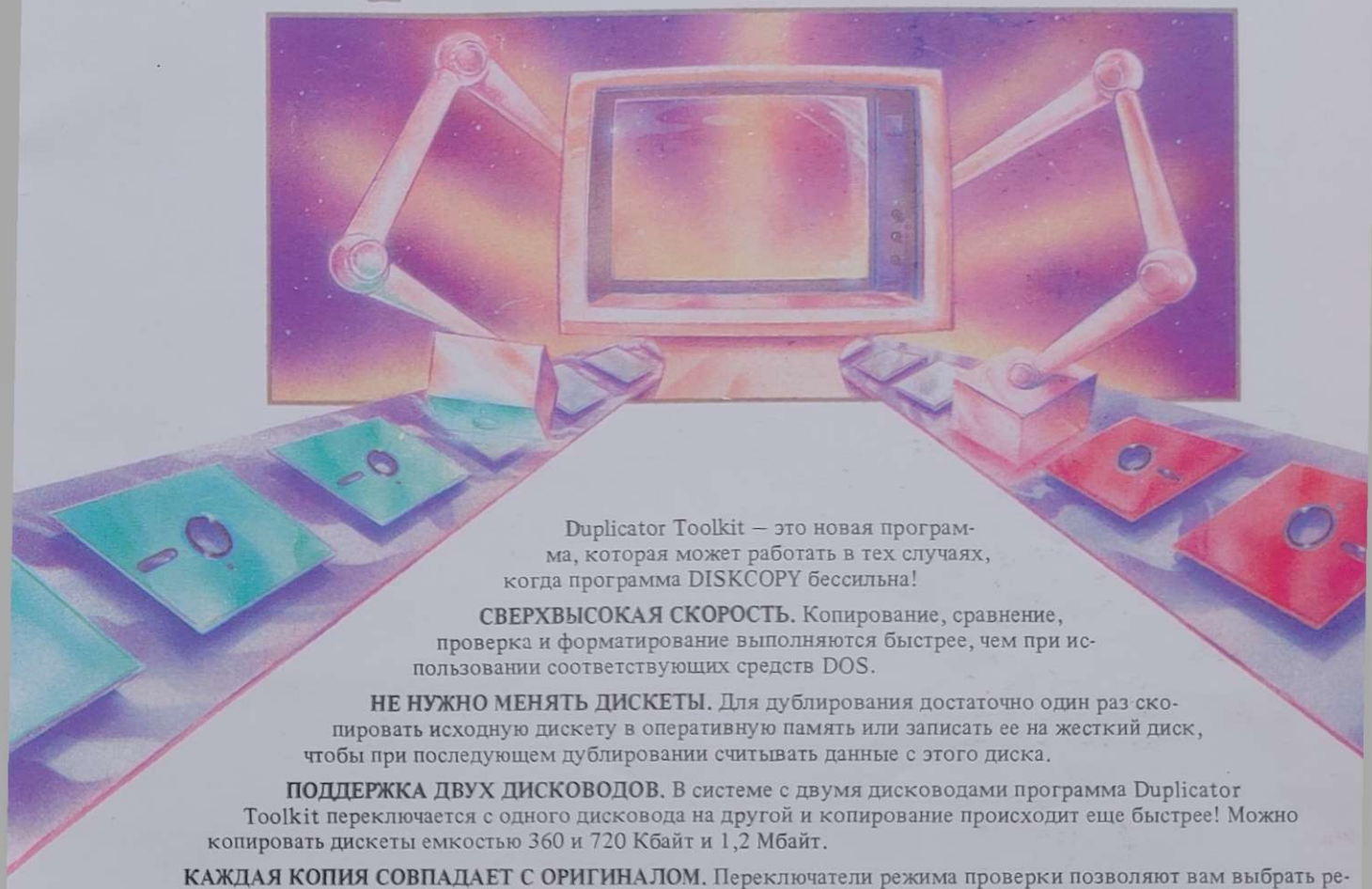
Пишите и телеграфируйте нам по приведенным ниже адресам, и, может быть, вы найдете новые доказательства того, что фирма SPC занимает лидирующее положение в мире программного обеспечения.

SPC SOFTWARE PUBLISHING CORPORATION

Telex: 650 268-5487 FAX: (415) 960-1918 1901 Landings Drive, P.O. Box 7210, Mountain View, CA USA 94039 (415) 962-8910

NEW!

Turn Your PC Into A Duplicating Machine!



Duplicator Toolkit – это новая программа, которая может работать в тех случаях, когда программа DISKCOPY бессильна!

СВЕРХВЫСОКАЯ СКОРОСТЬ. Копирование, сравнение, проверка и форматирование выполняются быстрее, чем при использовании соответствующих средств DOS.

НЕ НУЖНО МЕНЯТЬ ДИСКЕТЫ. Для дублирования достаточно один раз скопировать исходную дискету в оперативную память или записать ее на жесткий диск, чтобы при последующем дублировании считывать данные с этого диска.

ПОДДЕРЖКА ДВУХ ДИСКОВОДОВ. В системе с двумя дисководами программа Duplicator Toolkit переключается с одного дисковода на другой и копирование происходит еще быстрее! Можно копировать дискеты емкостью 360 и 720 Кбайт и 1,2 Мбайт.

КАЖДАЯ КОПИЯ СОВПАДАЕТ С ОРИГИНАЛОМ. Переключатели режима проверки позволяют вам выбрать режим, обеспечивающий абсолютную надежность копирования. Одновременно с копированием можно осуществлять ф матирование.

МОЖНО ЗАПИСЫВАТЬ И МЕТКИ! Можно генерировать и записывать метки для каждой дискеты. Можно задавать режим нумерации дискет в последовательном порядке.

ДЛЯ ПОЛУЧЕНИЯ КОПИЙ ПОЧТИ НЕ НУЖНО НАЖИМАТЬ НА КЛАВИШИ! Если вы не хотите читать даже сокращ ные инструкции, то эта программа предназначена специально для вас. На экран выводятся простые подсказки, которые помогут вам осуществить дублирование в считанные *секунды*.

Вы получите такие *возможности*, которые программа DISKCOPY предоставить не может. Заказывайте сегодня же!

МИНИМАЛЬНЫЕ ТРЕБОВАНИЯ К СИСТЕМЕ: ПК PC, XT или AT фирмы IBM (или совместимый с ним) и 256 Кбайт оперативной памяти. Операционная система PC DOS или MS DOS (версия 2.0 или старше). Рекомендуется жесткий диск. Программа не защищена от копирования.

COPY TECHNOLOGIES
14252 Culver Drive, Suite 323
Irvine, CA 92714

I've had it with DISKOPY! turn my PC into Duplicating Machine. ONLY $ 150.00

FOR INFORMATION OR ORDERS,
PLEASE CONTACT:

Wolfgang Fechner
Computer Periphere Systeme GMBH
Offenbacher Landstr. 70
6000 Frankfurt/M. 70
West Germany
49-69-6108013
Telex 412762

COPY TECHNOLOGIES
14252 Culver Drive, Suite 323
Irvine, CA 92714
(714) 975–1477

Если бы освоение космоса продвигалось со скоростью развития вычислительной техники, то из Вашего окна открывался бы такой вид:

В истории человечества ни одно техническое начинание не развивалось так быстро, как вычислительная техника за последние 40 лет. Для того чтобы шагать столь уверенной поступью, нужна своевременная и надежная информация.

Именно поэтому и появились мы. Мы – это фирма IDG Communications, Inc. – крупнейший в мире издатель газет и журналов по вычислительной технике. Более 14 000000 связанных с компьютерами людей во всех концах света полагаются на наши источники информации, чтобы оставаться на передовых позициях.

Если Вы предлагаете (или хотели бы предложить) программные продукты или услуги этим разбросанным по всему миру пользователям компьютеров, мы поможем Вам.

Мы поможем Вам найти поставщиков технических средств и программного обеспечения, которые занимаются обслуживанием средних и крупных организаций. Их интересы отражаются в 30 выпускаемых нами изданиях, которые выходят в Австрии, Австралии, Аргентине, Бразилии, Великобритании, Венесуэле, Венгрии, Греции, Дании, Израиле, Индии, Испании, Италии, Канаде, КНР, Южной Корее, Мексике, Нидерландах, Новой Зеландии, Норвегии, Саудовской Аравии, Финляндии, Франции, ФРГ, Чили, Швейцарии, Швеции, Юго-Восточной Азии и Японии.

Мы также поможем Вам связаться с пользователями персональных компьютеров: либо с помощью журналов широкого профиля по микрокомпьютерам, либо с помощью журналов, ориентированных на конкретные модели компьютеров. Для нужд таких пользователей мы выпускаем 32 издания в Австралии, Бразилии, Великобритании, Венгрии, Дании, Израиле, Индии, Мексике, Нидерландах, Норвегии, СССР, Финляндии, Франции, ФРГ, Чили и Швеции.

Выясните, насколько просто войти в контакт с нужной Вам аудиторией на этих рынках. Когда Вы пришлете программные продукты или сервисные средства, мы обеспечим одноразовую рекламу. Обращайтесь по адресу: Frank Cutitta, Managing Director, International Marketing Services, 375 Cochituate Road, Framingham, MA 01701-9171 (USA); телефон (508) 879-0700; телекс 951153; телефакс (508) 875-5981.

IDG
COMMUNICATIONS

INTERNATIONAL MARKETING SERVICES

Продукция фирмы БАСФ – системы хранения информации

Приступив в 1934 г. к промышленному изготовлению магнитных носителей информации, фирма БАСФ произвела своего рода технологическую революцию. На принципе магнитной записи, который поначалу использовался исключительно для целей звуковоспроизведения, основывается и по сегодняшний день накопление и хранение информации в области акустики, видео- и вычислительной техники. Для разработки и изготовления современных высококачественных магнитных носителей информации необходимо обладать глубочайшими знаниями и обширным „ноу-хау" в сфере химии и технологии производства.

Многолетние научные исследования и поиски позволили фирме БАСФ занять одно из ведущих мест в мире.

На протяжении многих лет фирма БАСФ является компетентным торговым партнером Советского Союза. Фирма располагает комплексными, испытанными технологиями в таких областях, как использование сырьевых и энергетических ресурсов, производство химикалиев, красителей и отделочных материалов, пластмасс, химических продуктов для сельского хозяйства, а также товаров народного потребления.

Главной задачей в нашей работе является выпуск продукции, ориентированной на запросы потребителей.
Сюда же относятся и носители информации для нужд вычислительной техники.

Продукция фирмы БАСФ – гарантия надежности

Необходимым условием создания долгосрочной системы архивирования данных является обеспечение их надежного и длительного хранения. Без выполнения этого условия не смогли бы существовать ни современная наука, ни экономика. Марка БАСФ служит залогом и гарантией высокого качества магнитных носителей информации. Колоссальные средства на исследования, которые ежегодно расходуются фирмой, высочайший технологический уровень производства – вот основа, определяющая технологическое и качественное преимущество продукции фирмы БАСФ.

От носителей информации к вычислительным системам

Производственная программа фирмы БАСФ в области вычислительной техники включает в себя также комплексные системы обработки данных.

В 1970 году фирма БАСФ в числе первых изготовителей вывела на рынок системы совместимых периферийных устройств.

Производственная программа постоянно обогащалась и в 1980 году была расширена до комплексных систем, изготовление которых ведется в Японии.

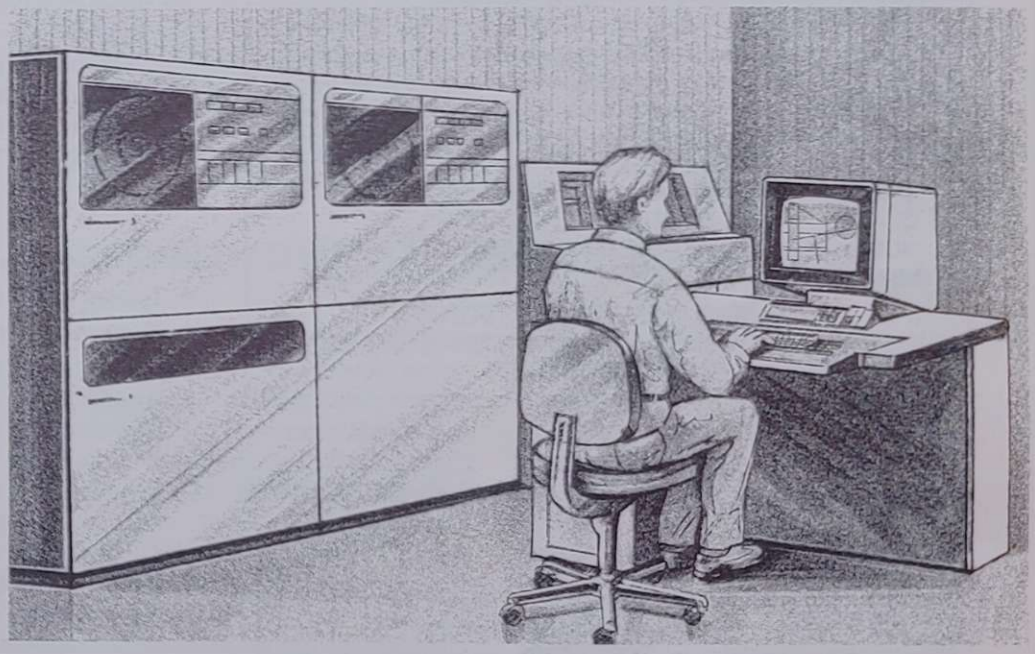
- Центральные процессоры – самые маленькие вычислительные устройства своего класса; необходимая площадь для установки – 1 м^2.

Они имеют все технические возможности расширения функций в соответствии с индивидуальными потребностями.

- Компактные устройства записи и воспроизведения данных на магнитной ленте; блок управления встроен в первое устройство; до трех уровней плотности записи в одной системе (800/1600/6250 бит/дюйм).
- Компактные и бесшумные высокоскоростные печатающие устройства; в каждой модели могут быть предусмотрены две скорости распечатки – 1500/2000 строк в минуту; прекрасное качество печати; используются стальные ленты; около 200 различных видов шрифтов.

В производственную программу входят также

- телепроцессорные системы для работы как в местном, так и в дистанционном режиме.



- Устройства для записи и воспроизведения данных на магнитных дисках; встроенные блоки дисков с неподвижными головками; емкость ЗУ составляет 317,5 Мбайт на 1 блок.

Наши системы обладают всеми преимуществами современной техники. Они занимают мало места и расходуют минимум энергии. Системы нечувствительны к колебаниям частоты и величины питающего напряжения. Испытанная продукция фирмы БАСФ гарантирует надежность в эксплуатации.

Внимание! Аппаратура фирмы БАСФ совместима не только с оборудованием, изготовленным по стандартам IBM, но и RIAD.

В программу поставок фирмы БАСФ входят гибкие магнитные диски, компьютерные магнитные ленты, информационные кассеты и кассетные запоминающие устройства, а также жесткие кассетные дисковые запоминающие устройства, пакеты магнитных дисков. Функциональная надежность этой продукции повышает эффективность вычислительных систем в целом.

Представительство фирмы БАСФ в СССР
103001 Москва
пер. Садовских 4, кв. 13
Тел.: 209-65-21/209-66-17
209-67-81/209-68-08
Телефакс 230-21-06
Телекс 413167

BASF

Для решения этой проблемы фирма IBM предлагает программируемый выбор режимов (Programmable Option Select): каждому адаптеру присваивается свой идентификатор, распознаваемый системой. Это позволяет разрешать возможные конфликты между адаптерами и отказаться от использования DIP-переключателей. Разработчик новой линии ЭВМ Эндрюс сказал, что фирма IBM предполагает включить в расширенную версию (Extended Edition) операционной системы OS/2 такие функции, которые еще более упростят установку систем (см. вставку "Эксплуатация операционной системы OS/2").

ПОДСТРОЙКА КОМПОНЕНТОВ В МСА-АРХИТЕКТУРЕ

В то время как добавочные платы для AT-совместимых ЭВМ функционируют только при тактовой частоте 8 МГц, МСА-архитектура позволяет платам работать с любой скоростью, не превышающей скорость работы микропроцессора. Кроме того, МСА-архитектура гарантирует по меньшей мере 16-битовый интерфейс. Хит утверждает: "Это гарантирует, что ваша система ввода-вывода BIOS может быть 16-битовой и вы сможете использовать ее гораздо эффективнее. Если в ваших приложениях требуется значительное число обменов данными, это может оказаться существенным фактором". Хотя большинство добавочных плат рассчитано на 8-битовые обмены данными, 32-битовая шина данных в PS/2 модели 80 не является бесполезной игрушкой. В будущем 32-битовые обмены в ЭВМ, выполненных на базе микропроцессора 80386, понадобятся не только оперативной памяти, но и интеллектуальным контроллерам. Хит приводит следующий пример: "Если у вашей системы нет 32-битового ввода-вывода, а вам требуется создать файл на интеллектуальном жестком диске с кэшем, то вас будет ограничивать не доступ к диску, а скорость обмена данными".

Среди других аппаратных новинок в PS/2 можно отметить наличие двух ПЗУ с системами BIOS: одно предназначено для работы в защищенном режиме микропроцессоров 80286 и 80386, а другое – для работы в реальном режиме. Это имеет важное значение, поскольку в OS/2 прикладные программы вместо инициации прерываний вызывают процедуры операционной системы и эти вызовы реализуются через систему ABIOS (Advanced BIOS), предназначенную для работы в защищенном режиме. Как и все другие программные компоненты операционной системы OS/2, система ABIOS состоит из реентерабельных процедур, которые могут отвлекаться на обслуживание более приоритетной программы, а затем продолжать работу с точки прерывания. Кроме того, входящий в состав ABIOS драйвер экрана обеспечивает возможность мультиэкранного режима работы. Наконец, ABIOS в несколько раз ускоряет довольно затруднительное переключение

Эксплуатация операционной системы *OS/2*

Большинство аппаратных возможностей PS/2 не будет использоваться до тех пор, пока не появится операционная система OS/2. В предвкушении того момента, когда новая операционная система приобретет товарный вид, ее потенциальные покупатели и разработчики программного обеспечения гадают о том, каким будет сценарий ее ежедневной эксплуатации? Уровень производительности при работе в новой операционной системе может оказаться недостаточным, и многих это беспокоит гораздо больше, чем необходимость иметь не менее 2 Мбайт оперативной памяти и резервировать большие объемы внешней дисковой памяти.

Обе фирмы, IBM и Microsoft, осторожно заявили, что при работе под управлением DOS в "блоке совместимости" прикладные программы будут исполняться медленнее, чем в своей естественной среде. Однако как поведут себя программы, специально написанные для OS/2?

"Производительность всегда оказывается ахиллесовой пятой новой операционной системы, [в особенности] если вы переходите от однопрограммной к мультипрограммной среде" – поясняет Джим Арчер, руководитель разработки OS/2 в лаборатории Отдела итерактивных систем фирмы IBM, расположенной в Бока-Рейтон. Он не приводит точных цифр, но говорит: "Я вполне доволен, что у нас не будет серьезных жалоб на производительность при работе в новой операционной системе на IBM AT" (наименее мощной ЭВМ, на которой может функционировать OS/2).

"Мы проявили известный максимализм при выборе целей, – заявляет Арчер. – Наша работа по повышению производительности начинается с совершенствования наименьших "атомов" системы и продолжается до уровня среды прикладных программ... Мы будем тесно сотрудничать с теми, кто захочет отладить систему, обеспечим их подходящим инструментарием и научим соответствующим приемам. Мы будем учиться сами, а вместе с нами будут учиться и производители программного обеспечения."

Другая серьезная проблема – "запрятывание" сложных процессов от конечного пользователя. По словам Арчера, в настоящее время для установки системы пользователю требуется достаточно высокий уровень знаний, в то время как хотелось бы свести процедуру установки OS/2 к выполнению одной команды INSTALL и не заниматься компоновкой системы из различных кусков.

Фирма IBM обещает предоставить пользователям полезные примеры выполнения процедуры установки, но не гарантирует, что тем пользователям, которые собираются работать в мультипрограммном режиме или использовать добавочные платы, не придется вносить изменения в файл CONFIG.SYS. Арчер по этому поводу поясняет: "Естественно, предполагается дать пользователям такую процедуру установки, которая будет удобнее редактирования файла CONFIG.SYS с помощью программы EDLIN". Он предупреждает, что при первом знакомстве с OS/2 не надо ожидать ничего магического. Но со временем сочетание функ-

ции расширенной версии OS/2 с программируемым выбором параметров в PS/2 обеспечит базу для комплексного решения задач.

В OS/2 будут включены возможности оперативных подсказок, а ее сообщения об ошибках будут достаточно ясными, но фирма IBM еще продолжает работу над несколькими существенными элементами интерфейса пользователя, например способами визуального представления файлов в ситуациях, когда данные распределены по нескольким ЭВМ. Одна из возможностей решения этой задачи состоит в применении подхода, апробированного в ЭВМ Macintosh, но Арчер указывает на следующее обстоятельство: "Миллионы пользователей научились пользоваться вложенными каталогами файлов. Надо ли пытаться их переучивать?" Другие затруднения связаны с тем, что все способы решения в OS/2 подобных задач должны быть согласованы со стандартами общего доступа пользователей (Common User Access), устанавливаемыми для архитектуры системных приложений (Systems Application Architecture), генеральной схемы для всей этой линии продукции фирмы IBM, которая еще не определена окончательно.

Как и любая мультипрограммная система, OS/2 имеет "регуляторы производительности", которые позволяют разработчикам прикладных программ или пользователям устанавливать своим программам определенные приоритеты. Эти возможности будут скрыты от большинства пользователей, хотя Арчер добавляет: "Конечно, вы захотите, чтобы такими возможностями обладали администраторы системы или квалифицированные пользователи".

Все существующие коммуникационные пакеты для DOS при переносе в OS/2 должны быть полностью переработаны. Резидентные прикладные программы придется разрабатывать заново, поскольку их эквиваленты в OS/2 должны функционировать на тех же правах, что и диск-резидентные прикладные программы, и разделять с последними системные ресурсы.

Особые проблемы возникают при исполнении в "блоке совместимости" программ, написанных для работы в DOS. Если с такой программой случится авария, то как поведут себя параллельно исполняемые приложения в OS/2? Разработчики программного обеспечения фирмы IBM отвечают, что некоторые модули OS/2 расположены в адресном пространстве DOS и поэтому защита прикладных программ в OS/2 не гарантируется. Если прикладная программа, работающая в "блоке совместимости", испортит самое себя, то с программами в OS/2 ничего не случится. В противном случае аварию может потерпеть вся система.

Пока производители программного обеспечения борются с новыми хитрыми деталями (в условиях, когда операционная система не до конца отлажена), главный вопрос состоит в том, все ли разработчики будут соблюдать правила работы с OS/2. Напомним, что в мире приложений DOS все производители программного обеспечения, включая фирму Microsoft, обходили правила работы с операционной системой. В мультипрограммной среде такая тактика приведет к хаосу.

"Если каждый пользователь захочет иметь [для своей программы] наивысший приоритет, то система перестанет быть управляемой", — говорит Арчер. Если каждый разработчик будет игнорировать некоторые спецификации или пытаться получить прямой доступ к аппаратным средствам, то проблемы возрастут в геометрической прогрессии, особенно с появлением графического интерфейса пользователя Presentation Manager.

Чтобы избежать этой ситуации, "мы включили в систему множество функций с таким расчетом, чтобы не было нужно ее обходить", заявляет Арчер. По этой же причине в OS/2 радикально ускорено (по сравнению с DOS) выполнение процедур вывода текста на дисплей.

Что это даст разработчикам? Прежде всего совместимость: их покупателям не придется жаловаться, что купленные программы не способны к параллельной работе с другими программами. Арчер также обратил внимание на то, что сегодняшние приложения, обладающие подобным "плохим поведением", приходится с большими затратами труда переделывать для каждой новой модели ЭВМ: "Гораздо более структурированная среда позволит упростить перенос на новые модели и позволит нам быстрее извлекать преимущества, заложенные в ее новых свойствах".

микропроцессора 80286 из защищенного режима в реальный.

Другим достоинством PS/2 является кэш для диска, который обслуживает прикладные программы как в режиме DOS, так и в режиме OS/2. Он был разработан вслед за MCA-архитектурой и дает возможность считывать за одно обращение к дисководу целую дорожку, а затем мгновенно передавать эти данные путем "взрывного" режима обмена, обеспечиваемого MCA-архитектурой, что с лихвой окупает накладные расходы на выполнение транзакций с кэшем. Согласно Хиту другой изюминкой PS/2 является продуманный алгоритм кэширования, который получен "в результате анализа реальных режимов работы и моделирования на ЭВМ".

Хит отмечает, что, если требуемые вам данные в 90 % случаев находятся в кэше, вы не почувствуете медлительности доступа к диску, который в противном случае стал бы бутылочным горлышком для модели 50. Оптимизация режимов доступа при использовании кэша минимизирует износ диска и тем самым повышает надежность системы.

В отличие от адаптера EGA (Extended Graphics Adapter), новый адаптер VGA (Video Graphics Array) может сообщать системе, в каком из графических или текстовых режимов он находится. Это упрощает управление экраном при переключении системы с одного приложения на другое. По словам Эндрюса, при работе с адаптером EGA (на IBM AT) операционной системе OS/2 приходится решать проблему переключения достаточно топорным способом. Насколько он помнит, способность адаптера VGA давать информацию о текущем режиме экрана была "одним из требований, переложенных на аппаратуру операционной системой".

Будут ли учитываться эти довольно тонкие отличия первыми приложениями OS/2 и не окажется ли,

что они будут исполняться на IBM AT заметно медленнее, чем на PS/2? "Думаю, что нет, если привести единицы измерения к общему знаменателю с учетом разницы в тактовой частоте и наличия кэша, – отвечает Эндрюс. – Программное сохранение содержимого видеорегистров или задержки переключения видеорежимов в первое время не будут серьезными препятствиями. Однако за этим надо следить."

АКЦЕНТ НА НАДЕЖНОСТЬ

Хит замечает: "Крайне трудно взять некую программу, загрузить ее в ЭВМ и получить от нее число, которое покажет, что система обладает низкой надежностью. Я утверждаю, что, оставляя в стороне средства восстановления после сбоев, надежность системы можно считать наиболее важным фактором".

Он добавляет: "Когда вы оцениваете производительность, учитывайте и другие факторы. Если я потерял прерывание и понадобилось 5 мин для отключения системы и ее повторного вызова, то сколько состояний ожидания пройдет за эти 5 мин?"

По словам Эндрюса, у новых машин фирмы IBM среднее время наработки на отказ в пять раз выше, чем у первых IBM PC, хотя он и не называл точных цифр. "Мы были лучшими в этой отрасли, но общее повышение надежности продукции – как нашей, так и других производителей, использующих эту [IBM PC] архитектуру, – не позволяет оставаться на месте".

Забота о надежности чувствуется во всех деталях конструкции PS/2. Фрэнк Кинг, вице-президент Отдела интерактивных систем, ответственный за разработку, отмечает широкое использование в целях снижения тепловыделения и уменьшения числа паек собственных интегральных микросхем, выполненных на базе КМОП-технологии.

Увеличение тактовой частоты и плотности расположения компонентов приводит к росту электромагнитных помех. Для снижения их уровня в PS/2 предусмотрены заземляющая панель, целиком накрывающая шасси, и заземление всех заглушек для неустановленных плат, а также устранены прямоугольные изгибы соединительных проводников, расположенных на системной плате. В конечном счете эти меры обеспечивают возможности дальнейшего повышения быстродействия системы.

Группа разработчиков PS/2 подчеркивает необходимость составления четко определенных спецификаций, связанных с другими стандартами фирмы IBM и дающих возможность третьим фирмам обеспечить совместимость их программных продуктов и дополнительных плат с ЭВМ PS/2. Хит замечает: "Важнее найти четкое и последовательное архитектурное решение, которое можно объяснить и для которого можно составить точные спецификации, чем приводить принципиальные схемы, типы модулей и листинги системы BIOS, заставляя пользователей догадываться, что бы это значило. У IBM PC не было таких спецификаций, и вам постоянно приходилось мучиться в догадках. Это затрудняет обеспечение совместимости, поскольку приходится переносить все, что было придумано для IBM PC".

КАВАЛЬКАДА ПРОЦЕССОРОВ

В то время как в архитектурах типа архитектуры IBM AT доступом к шине управляет только центральный процессор, в микроканальной MCA-архитектуре арбитрами одновременного доступа к шине кроме центрального процессора могут служить еще 15 других интеллектуальных устройств, называемых хозяевами шины. Хотя фирма IBM пока еще не предлагает подобных устройств, она уже публично продемонстрировала одновременную работу с шиной нескольких процессоров. По словам Хита, микропроцессор 80386 может работать в 16-битовом режиме и его можно встраивать в ЭВМ, имеющие 16-битовые интерфейсы, таким образом, что он будет работать либо в монопольном режиме, либо совместно с другими процессорами.

Он добавляет: "Мультипроцессный режим работы допускает множество сценариев. Наиболее вероятно, что он будет включать в себя гибкую организацию связей и использовать присущую хозяевам шины эффективность выполнения обменов данными". По оценке Хита, этот метод будет в два–четыре раза более эффективным, чем использование контроллеров прямого доступа к памяти (Direct Memory Access). (PS/2 имеет восемь каналов прямого доступа к памяти, так что устройства типа дисководов смогут общаться с памятью в обход центрального процессора.)

"Применяя для ввода-вывода выделенные процессоры, вы можете избежать ситуаций, когда один высокоприоритетный процесс блокирует выполнение другого процесса, – отмечает Хит. – Наконец-то, у меня развязаны руки и я могу сконструировать такую систему, в которой контролируются взаимные помехи и избыточные действия." К другим приложениям параллельных процессоров относятся манипуляции с большими объемами входных и выходных данных, например обмены информацией с центральными ЭВМ или – при работе в локальной сети – с другой персональной ЭВМ.

Джим Макклеллан, менеджер по стратегическому и системному планированию линии PS/2, считает, что мультипроцессный режим на PS/2 начнется с подключения интеллектуальных устройств, управляемых расширениями операционной системы DOS через соответствующие драйверы. Затем уже под управлением OS/2 появятся возможности более гибких подключений, хотя в первой версии они еще не будут в достаточной мере обеспечены. Макклеллан предсказывает, что скорее всего мы увидим "очень сложный вид разделения процессов с использованием подключенных к шине параллельных процессоров".

"Мы решили эту проблему [параллельного исполнения процессов] для ЭВМ ценой в 5 млн дол, – отмечает Макклеллан. – Теперь мы говорим о ее решении для такой ЭВМ, которую мы с вами были бы в состоянии купить и принести домой."

Гораздо ближе на пути к дому находятся специализированные процессоры, которые, например, обеспечивают телекоммуникацию или вывод графических образов. "Когда говорят о центральной ЭВМ локальной сети, то имеют в виду совершенно иные параметры, – считает Макклеллан: – Не стоит и думать, что центральная ЭВМ как раз и окажется тем ящиком, который вы захотите установить на своем столе, но технология коммуникаций примерно одинакова."

КТО СЛЕДУЮЩИЙ?

Фирма IBM защищает отдельные аспекты микроканальной MCA-архитектуры от конкурентов, пытаю-

щихся скопировать PS/2, не менее тщательно, чем некоторые виды других технологий в прошлом. Кинг разъясняет политику фирмы следующим образом: "Микроканальная MCA-архитектура опирается на сочетание конструкции, которая частично запатентована, системы BIOS, на которую фирма IBM заявила авторские права, а также масок и собственно микросхем. Мы не собираемся продавать лицензии на технологию изготовления этих четырех компонентов".

Однако значительная часть архитектуры PS/2 открыта для всех желающих, и фирма IBM освещает ее на семинарах для разработчиков из других фирм. "Если вас интересует, будут ли наши ЭВМ клонироваться, можете спросить об этом у тех, кто пытается это сделать, – замечает Хит. – Не хочу сказать, что это [технически] невозможно, но нам самим понадобилось очень много времени [на их разработку]."

"Мы могли бы поднять структуру и архитектуру IBM AT на более высокий уровень, но это дало бы лишь краткосрочное преимущество, – резюмирует Эндрюс. – Гораздо дальновиднее создать новую платформу. Мы могли бы сделать еще один детский шаг, но время не ждет."

"Коварное" предварительное сообщение о новых персональных компьютерах *PS/2* заставило покупателей колебаться

АЛИСА ЛАПЛАНТ

В связи с недавними рекламными заявлениями фирмы IBM, касающимися будущей продукции этой фирмы и включающими дюжину новых машин, выпускаемых только в этом году, агенты по снабжению перестали покупать дополнительные ПК PS/2 до тех пор, пока фирмой IBM не будет официально объявлено о продаже обещанных более мощных и дешевых машин. Агенты по снабжению, торговые агенты и обозреватели одинаково ожидают, что основные заявления фирма IBM сделает к первой годовщине объявления о своем ПК PS/2, и многие "сдерживают дыхание" до этого.

"Мы откладываем закупки новых машин, за исключением самых безотлагательных нужд, – говорит Эдвард Перриш, руководитель группы планирования и технологии разработки информационных систем фирмы Sun Oil в Радноре, шт. Пенсильвания. – Это (планы фирмы IBM) заставило нас очень серьезно задуматься, прежде чем вложить деньги в дополнительные ПК PS/2.

Сотрудник фирмы IBM Уильям Лоу кратко информировал обозревателей и журналистов о жесткой стратегии фирмы IBM в области маркетинга и цен на 1988 и 1989 гг.

Среди прочего Лоу сказал, что к концу 1988 г. первоначальный вариант системы, основанной на микропроцессоре 286, будет стоить столько же, сколько сегодня стоит модель 25, и что к концу 1989 г. наименее мощный вариант системы, основанный на микропроцессоре 386, будет продаваться по такой же цене.

Эд Гриффит, руководитель Информационного центра Первого банка шт. Алабама, также воздерживается от покупки ПК PS/2. "За последние два года фирма IBM давала большие обещания относительно первой половины апреля, – сказал Гриффит. – Я хочу подождать и посмотреть, что они будут анонсировать в этом году, прежде чем принимать какие-либо решения о покупке".

Защита интересов покупателя? Норм Девитт, сотрудник фирмы Dataquest, считает, что Лоу определенно рисковал, когда делал такие однозначные заявления о будущей продукции. Однако Девитт также подчеркнул, что фирма IBM заявила о своем намерении защитить интересы покупателей, вкладывающих средства в их продукцию PS/2.

Например, Лоу обещал, что когда фирма IBM усовершенствует модель 50, которую многие называют "покалеченной машиной", введя более емкий и быстрый жесткий диск, то интересы всех нынешних владельцев модели 50 будут защищены.

"Таким образом, нет причин откладывать покупку модели 50, – сказал Девитт. – Однако если вы подумывали о покупке модели 30, то, может быть, имеет смысл подождать, пока к концу года не появится вариант ПК на микропроцессоре 286.

Тем не менее многие специалисты признают, что они не в силах сдержать закупки ПК.

"Мы покупаем рабочие станции, поскольку они нам необходимы, – сказал Дейвид Мессинджер, директор фирмы Avon в Нью-Йорке. – Мы не можем позволить себе ждать".

Тем не менее Мессинджер отмечает, что фирма Avon приведена в замешательство политикой фирмы IBM в отношении ПК PS/2, что вызвало приостановку закупок этих машин в качестве рабочих станций со стандартной конфигурацией. Один из отделов фирмы Avon опирался именно на модель 50 как на стандартную, однако стратегия этой фирмы в отношении ПК, которую определяет Мессинджер, основывается на использо-

вании в качестве временного стандарта ПК Deskpro 286 фирмы Compaq.

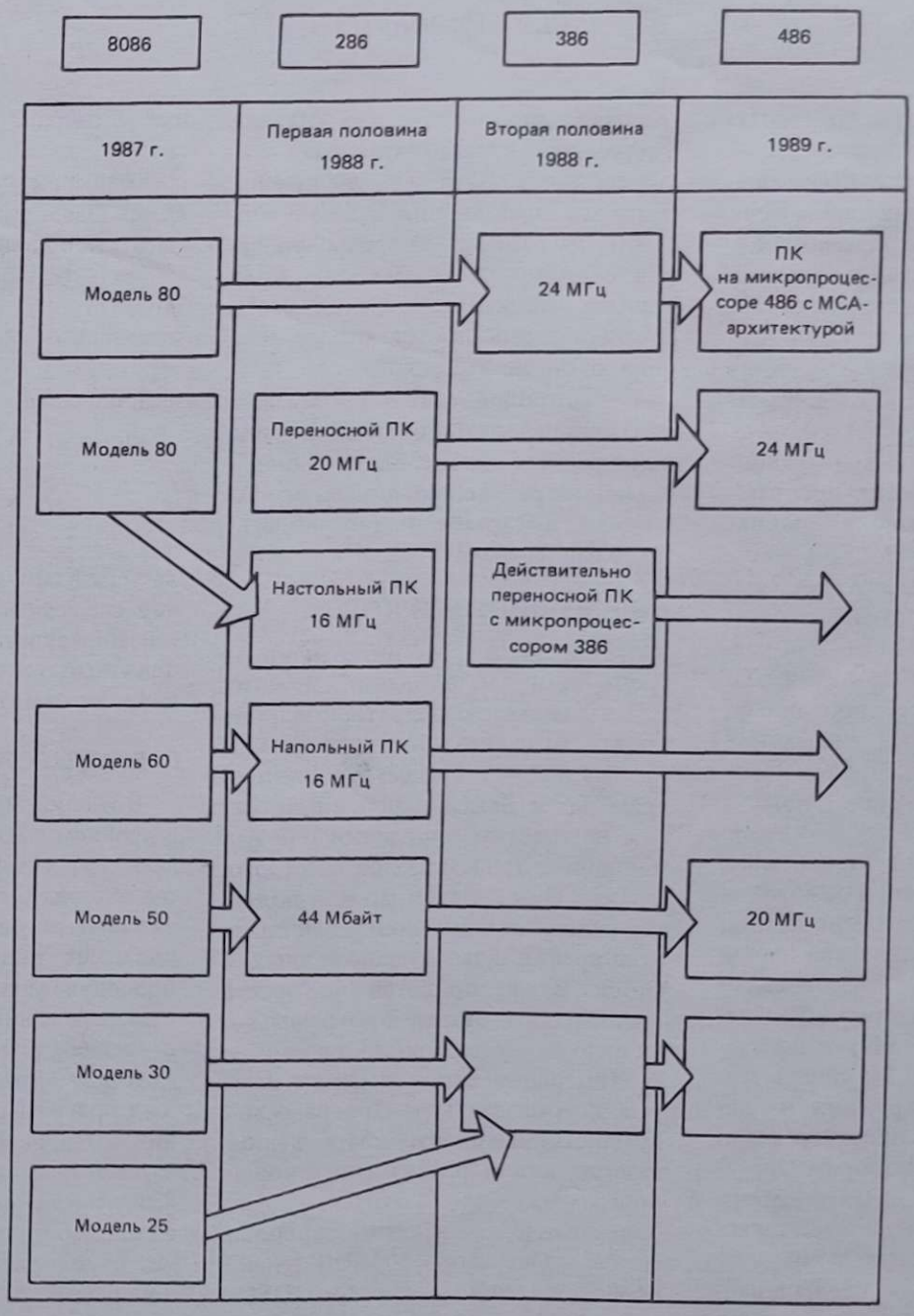
"К сожалению, ПК PS/2 доставили нам много хлопот", – сказал Мессинджер и добавил, что их цель – связать машины фирм DEC и IBM сетью Ethernet – была осложнена именно из-за ПК PS/2.

Другие пользователи не согласны с тем, что произошло что-то действительно заслуживающее внимания.

"Наши дела так приноровились к быстрым изменениям технологии, что последние заявления фирмы IBM не явились для нас новостью", – заявил Гэри Хоторн, вице-президент по маркетингу фирмы Pac Tel Infosystem.

Многие торговые агенты ты по снабжению заявили, чт мало уверенности в любых рительных сообщениях о бу машинах фирмы IBM. "Я н когда я увижу, – сказал Миллс, владелец магазина М в Берлингтоне, шт. Массачу Я не могу себе позволить бе иться о будущем".

Стратегия фирмы IBM по выпуску ПК PS/2



Фирма Dataquest предсказывает, что в объявления фирмы IBM на 1988 г. будут включены настольный ПК на микропроцессоре 386, улучшенная модель 50 и модели 25 и 30 на основе микропроцессора 286

Бейсик и его применение

Программирование на *Quick Basic*

ДЕЙВИД УИЛЛЬЯМЗ

В начале 1987 г. фирма Microsoft выпустила Quick Basic версии 3.0 – новейший вариант компилятора для языка программирования Бейсик. На программистов, пишущих программы на этом языке, большое впечатление своими скоростью и возможностями для отладки произвела предыдущая версия компилятора Quick Basic 2.0 той же фирмы. Теперь Quick Basic версии 3.0 предлагает дополнительные возможности и еще большую гибкость программирования и совместно с языком Turbo Basic фирмы Borland устанавливает новый стандарт для языка программирования Бейсик.

Новые средства Quick Basic расширяют его возможности для разработки более структурированных, понятных и легко переносимых программ. Алфавитно-цифровые метки, например, заменили номера строк, что облегчает отслеживание потока управления в программе. Процедуры расширили возможности модульного программирования, которое давно уже является неотъемлемым свойством языков высокого уровня. Операторы определения функции (DEF FN) и ветвления IF . . THEN . . ELSE теперь могут включать конструкции из нескольких строк, что облегчает работу со сложными функциями. Операторы цикла DO . . LOOP и выбора SELECT . . CASE увеличивают гибкость и вычислительную мощь.

Однако основным преимуществом Quick Basic является скорость вычислений. В отличие от интерпретирующих версий Бейсика, который преобразует операторы языка в машинный код при каждом выполнении программы, компилятор преобразует программу на Бейсике в машинный код только один раз. В результате компилированная программа выполняется в 5–10 раз быстрее, чем интерпретируемая.

Раньше вы платили за увеличение скорости выполнения менее удобными средствами программирования и отладки. Современные компиляторы, такие как Quick Basic, сочетают в себе достоинства и компиляторов, и интерпретаторов.

В настоящей статье рассказано, как использовать новые эффективные средства Quick Basic, и описаны некоторые возможности, не упомянутые в фирменном руководстве по этому языку.

БОЛЕЕ ПОНЯТНЫЙ КОД

Quick Basic – это интерактивная среда программирования, в которой вы можете создавать, компилировать и выполнять программы быстро и легко. Существующие программы на Бейсике почти или совсем не требуют переделки для выполнения, причем более быстрого, в среде Quick Basic. Но для полного использования новых мощных расширений языка, заложенных в Quick Basic, придется не просто адаптировать старые программы – их надо будет переписать заново.

Нумерация строк в Quick Basic уже не обязательна. Программы с нумерованными строками выполняются, как и раньше, но в новых программах предпочтительнее использование алфавитно-цифровых меток. Оператор GOSUB SORT более понятен, чем оператор GOSUB 1234.

Метки нужны, естественно, не для каждой строки, а только для тех, на которые есть ссылки из других строк программы. Quick Basic допускает смесь меток и номеров. Поставляемый диск с Quick Basic содержит программу, написанную на Бейсике, которая удаляет ненужные номера строк из старых программ.

Хотя при использовании языка Quick Basic уже нет необходимости писать длинные программные строки, для большей удобочитаемости одну строку можно разделить на несколько с помощью символа подчеркивания. Рассмотрим следующий оператор описания поля:

```
FIELD #1, 10 AS STKNUM, _
          20 AS ITEMNAME, _
           5 AS LOCATION, _
          10 AS QUANT
```

За каждым символом подчеркивания следует символ перевода строки. В результате получается более понятный текст и облегчается введение комментариев.

ПРЕИМУЩЕСТВО ПРОЦЕДУР

Пожалуй, наиболее мощным расширением в Quick Basic являются процедуры (subprograms). Процедуры сходны с подпрограммами, вызываемыми оператором GOSUB, но обладают рядом преимуществ. В процедуре можно описывать локальные переменные, что облегчает построение родовых подпрограмм, которые можно вызывать из разных программ, не опасаясь совпадения имен переменных. Процедуру можно поместить в библиотечный файл, и к ней можно будет обратиться из любой программы, что облегчит и ускорит разработку больших программ.

На рис. 1 показана программа, состоящая из основной программы и пяти процедур, которая демонстрирует формат процедуры. Атрибут STATIC в заголовке процедуры указывает, что процедура не рекурсивная, т. е. не может вызывать сама себя. Рекурсивные процедуры будут реализованы в последующих

версиях Quick Basic, а в версии 3.0 все процедуры должны описываться с этим атрибутом.

Поскольку переменные, описанные в процедуре, локальны в ней, то необходимы средства для передачи значений при вызове процедуры из основной программы. Одним из них является использование атрибута SHARED (разделяемый) в операторах DIM или COMMON. (Оператор COMMON должен быть расположен до исполняемых операторов.) Пример программы на рис. 1 показывает, как это делается. В начале основной программы размещены операторы DIM SHARED и COMMON SHARED. Все переменные, описанные в этих операторах, становятся доступны всем процедурам, причем в процедурах не надо вводить дополнительное описание COMMON. Это лучший способ для описания многих переменных, если они должны быть использованы во всех процедурах.

В процедуре DISPLAY с помощью оператора SHARED, похожего на атрибут SHARED, описываются разделяемые переменные, которые используются в этой процедуре. Такое описание можно использовать тогда, когда к разделяемым переменным обращаются только некоторые процедуры. Массив A( ) доступен процедуре DISPLAY, поскольку он доступен всем процедурам.

Переменные, которые не объявлены разделяемыми, являются локальными в главной программе или в той подпрограмме, где они описаны. Переменная I, которая описана в главной программе и в каждой процедуре, на самом деле представляет пять отдельных переменных.

Описание COMMON и SHARED применимо к программам и процедурам, контролируемым как один модуль. Их использование несколько отличается в процедурах, компилируемых независимо; эти особенности мы опишем чуть позже.

Значения в процедуру могут передаваться с помощью списка аргументов, что является преимуществом процедур по сравнению с традиционными подпрограммами Бейсика, вызываемыми оператором COSUB. Список аргументов позволяет многим программам с разными именами переменных вызывать одну процедуру. Аргументами (фактическими параметрами) в операторе вызова процедуры CALL могут

Рис. 1. Формат процедуры

```
'A sample program illustrating use of subprograms
      DIM B(10), C(10)
      DIM SHARED A(10)          'All subprograms can use the array A()
      COMMON SHARED N           ' and the variable N
      CLS
      CLEAR
      N=3
      CALL INIT
'Subprogram RAISE is used to operate on two different arrays
      CALL RAISE(B(),2)         'The entire array B() is passed
      CALL RAISE(C(),3)         'Likewise C()
      CALL AVERAGE(B(),I)       'I returns the average of B()
      CALL DISPLAY
      PRINT
      PRINT "The average is ";I
      CALL SUM(B(8),C(8))       'Individual array elements are passed
      END

SUB   SUM(X,Y) STATIC           'X and Y are local variables that
      PRINT X+Y                 ' take on the value of the variables
END   SUB                       ' passed by the CALL statement

SUB   INIT STATIC               'Operates on the array A()
      FOR I = 1 TO 10
          A(I) = I+N
      NEXT
END   SUB

SUB   RAISE(X(1),POWER%) STATIC          'X() assumes the identity of
      FOR I = LBOUND(A,1) TO UBOUND(A,1) ' the passed array
          X(I) = A(I)^POWER%
      NEXT
END   SUB

SUB   AVERAGE(X(1),AVE) STATIC           'Computes a single value
      FOR I = LBOUND(X,1) TO UBOUND(X,1) ' and returns it
          TOTAL = TOTAL + X(I)
      NEXT
      AVE = TOTAL/UBOUND(X,1)
END   SUB

SUB   DISPLAY STATIC
      SHARED B(),C()            'Gives access to B() and C()
      FOR I = 1 TO 10
      PRINT A(I),B(I),C(I)      'A() is made available by the
      NEXT                      ' COMMON statement
END   SUB
```

Конец ◄

Рис. 2. Компиляция процедур

```
' a module used to compile subprograms

      DIM A(10), B(10), C(10)             'These statements must match
      COMMON SHARED N, A(), B(), C()      ' those in main program

SUB   INIT STATIC
      FOR I = 1 TO 10
          A(I) = I+N
      NEXT
END   SUB

SUB   RAISE(X(1),POWER%) STATIC
      FOR I = LBOUND(A,1) TO UBOUND(A,1)
          X(I) = A(I)^POWER%
      NEXT
END   SUB

SUB   AVERAGE(X(1),AVE) STATIC
      FOR I = LBOUND(X,1) TO UBOUND(X,1)
          TOTAL = TOTAL + X(I)
      NEXT
      AVE = TOTAL/UBOUND(X,1)
END   SUB

SUB   DISPLAY STATIC
      FOR I = 1 TO 10
      PRINT A(I),B(I),C(I)
      NEXT
END   SUB
```

Конец ◄

Рис. 3. Определяемые функции

```
'The sample program converted to use defined functions
    DIM A(10), B(10), C(10)

    DEF FNINIT              'Initializes A() without passing a
    FOR I = 1 TO 10         ' value directly
        A(I) = I+N
```

Продолжение ►

быть: переменные, такие как A, A ( ) или A$; константы, такие как числа (145) или строки ("January"); выражения, такие как B$+"month" или STRING$ (40, " "). В руководстве по языку Quick Basic не описано использование выражений, но я не встретил никаких трудностей в их построении. Формальные аргументы, стоящие в скобках после имени процедуры, являются локальными переменными в данной процедуре. Соответствующие фактические аргументы в операторе CALL должны иметь такой же тип данных и располагаться в том же порядке.

Процедуры RAISE и AVERAGE иллюстрируют использование списка аргументов. В обе процедуры оператором CALL передаются массив и еще одна переменная. Опущенные индексы в скобках после имени массива в операторе CALL означают, что массив передается целиком. В заголовке процедуры число в скобках после имени массива указывает число измерений массива. Если необходимо передать в качестве аргумента только один элемент массива, то в операторе CALL надо указать имя массива с соответствующим индексом, как в вызове процедуры SUM.

Обратите внимание на то, что символ % в переменной POWER%, стоящей в описании процедуры, обозначает преобразование типа передаваемого оператором CALL аргумента в целый. Можно было бы использовать при вызове аргументы 2.0 и 3.0, а символ % в формальном параметре опустить.

Две процедуры демонстрируют использование функций LBOUND и UBOUND, с помощью которых можно определить размер массива. Аргументами этих функций являются имя массива и число измерений. Эти функции позволяют сделать процедуры еще более независимыми от вызывающих программ. В данном примере вы можете изменять размер трех массивов, не меняя процедур RAISE и AVERAGE. Функция LBOUND возвращает 0 или 1, если в программе есть оператор OPTION BASE 1. Функция UBOUND возвращает максимальный индекс массива.

Функции AVERAGE и SUM являются "самыми родовыми" в данном примере. Они полностью независимы от основной программы; вы можете перенести их в другую программу и использовать, не заботясь о именах переменных и размерах массивов. Из всех процедур данного примера только эти две можно помещать в библиотеку, не меняя ничего в основной программе.

Хотя процедуры – это прекрасный подарок программистам, они имеют некоторые ограничения. Метки, использованные в процедурах, не являются локальными и поэтому не могут иметь имена, совпадающие с именами меток в основной программе или в других подпрограммах. Из этого вытекает, что имена меток должны образовываться на основе имени процедуры, в которой они используются. Вы не можете войти в процедуру, применив операторы GOTO или GOSUB, и выйти из процедуры, применив оператор RETURN. Если вы используете эти операторы в процедуре, они должны ссылаться на метку или номер строки вне данной процедуры.

► Продолжение

```
  NEXT
  FNINIT=1              'Dummy statement
  END DEF

  DEF FNRAISE(X,M)      'Raises a single variable to a power
  FNRAISE = X^M
  END DEF

  DEF FNAVERAGE
  STATIC I,TOTAL        'Identifies two local variables
  FOR I = LBOUND(B,1) TO UBOUND(B,1)
      TOTAL = TOTAL + B(I)
  NEXT
  FNAVERAGE = TOTAL/UBOUND(B,1)       'The result
  END DEF

  DEF FNSUM(X,Y)
  FNSUM = X+Y
  END DEF

  CLS
  CLEAR
  N=3
  Z=FNINIT              'Initialize array A()
  FOR I = 1 TO 10
      B(I) = FNRAISE(A(I),2)
      C(I) = FNRAISE(A(I),3)
      PRINT A(I),B(I),C(I)
  NEXT
  PRINT
  PRINT "The average is ";FNAVERAGE
  PRINT FNSUM(B(8),C(8))
  END
```

Конец ◄

Рис. 4. Оператор IF . . . THEN . . . ELSE

```
IF X$ = "A" THEN
      B = 1                       'Condition 1
      G$ = "Message 1"
      GOTO MENU1
ELSEIF X$ = "B" THEN
      B = 2                       'Condition 2
      G$ = "Message 2"
      GOTO MENU2
ELSEIF X$ = "C" THEN
      B = 3                       'Condition 3
      G$ = "Message 3"
      GOTO MENU3
ELSE                              'None of the above conditions met
      PRINT "Non valid input"
      GOTO CMDIN
END     IF
```

Конец ◄

БИБЛИОТЕКИ ПОЛЬЗОВАТЕЛЯ

Библиотеки пользователя – это набор часто используемых подпрограмм, скомпилированных компилятором Quick Basic или написанных на языке ассемблера. Эти библиотеки – еще одно важное средство, предоставляемое Quick Basic. Они позволяют разрабатывать прикладные программы по частям, уменьшают время компиляции и облегчают использование уже готовых программ при разработке новых. Естественно, заносимые в библиотеку процедуры должны быть тщательно отлажены.

Процедуры Quick Basic идеальны для построения библиотек пользователя, но при этом надо соблюдать некоторые правила. В программе на рис. 1 библиотечными могут быть только процедуры SUM и AVERAGE. Поскольку остальные используют разделяемые переменные и общие области, то для занесения в библиотеку их надо преобразовать.

Если основная программа использует оператор COMMON, то такой же оператор следует использовать во всех процедурах, которые должны компилироваться раздельно. Все

операторы COMMON должны содержать атрибут SHARED. Если в операторе COMMON описан массив, то надо также дублировать оператор DIM.

Оператор SHARED (не атрибут!) не может использоваться в библиотечных процедурах. Следовательно, необходимо включить все переменные, описанные в операторах SHARED, в операторы COMMON.

На рис. 2 показан программный модуль, содержащий пять процедур с необходимыми заголовками. Оператор COMMON в основной программе должен соответствовать такому же оператору в программном модуле. Напомним, что операторы DIM и COMMON в отдельно транслируемом модуле будут не нужны, если этот модуль будет содержать только процедуры вроде SUM и AVERAGE.

Придерживайтесь этого формата, не дублируйте метки, имеющиеся в основной программе или в других процедурах, и процедуры и библиотеки не доставят вам никаких хлопот.

Для занесения ваших процедурных модулей в библиотеки вы должны сначала скомпилировать их с опцией BRUN.LIB и получить файл объектного кода (типа OBJ). Один объектный файл может содержать несколько процедур, как в программе на рис. 2. Далее, используя программу BUILDLIB, надо занести объектные модули в библиотечный файл. Команда должна выглядеть следующим образом:

```
BUILDLIB FILE1.OBJ FILE2.OBJ . . . ;
```

По этой команде создается файл с именем USERLIB.EXE. После того как библиотечный файл создан, в него нельзя добавлять и из него нельзя удалять отдельные процедуры. Если вы хотите изменить библиотеку, необходимо повторить формирование библиотеки с перечислением всех объектных файлов.

Для использования библиотеки надо загружать Quick Basic с параметром /L:

```
QB имя_файла/L
```

где имя_файла – имя файла программы на Бейсике.

ОПЕРАТОР ОПРЕДЕЛЕНИЯ ФУНКЦИИ

Quick Basic поддерживает традиционную форму однострочного определения функции и, кроме того, позволяет создавать функции, состоящие из многих строк. Функции теперь могут включать операторы IF . . . THEN, FOR . . . NEXT и другие сложные последовательности команд. Программа на рис. 3 выполняет те же действия, что и в предыдущих примерах, но вместо процедур используются функции. Обратите внимание, что функции должны быть определены до использования. Хотя этот пример и не может служить образцом хорошего стиля программирования, он показывает применение определяемых функций.

Значения передаются в функцию через список аргументов. Переменные в списке формальных аргументов определения функции являются локальными в ней. Вы можете определить другие локальные переменные оператором STATIC, как это сделано в функции FNAVERAGE. В качестве параметра можно передавать только элемент массива, но не весь массив. Возвращаемое функцией значение должно быть присвоено переменной, имеющей имя функции. Когда имя функции встречается в программе, возвращаемое функцией значение подставляется вместо имени. Функция может оперировать переменными, не описанными в списке аргументов, в том числе и массивами, и менять их значение. В нашем примере это делает функция FNINIT. Фактически эта функция не возвращает результата в традиционном смысле. Переменная Z в строке программы Z=FNINIT яаляется пустой, она нужна только для вызова функции FNINIT.

Когда надо использовать определяемые функции, процедуры и подпрограммы, вызываемые оператором GOSUB? Они имеют смысл, когда необходимо повторять некоторую последовательность кода несколько раз или использовать в разных программах одинаковые подпрограммы. Определяемые функции больше подходят для маленьких подпрограмм, которые вычисляют одно значение. Процедуры потребуются для построения независимых подпрограмм, которые можно занести в библиотеку пользователя.

Какую форму подпрограммы лучше использовать, если по логике работы программы годится любая? Я прогнал некоторые тесты для определения относительной скорости выполнения и занимаемой памяти для каждого из вариантов. Разница в скорости выполнения несущественна (механизм процедур немного быстрее других), зато по занимаемой памяти выигрывают подпрограммы, вызываемые оператором GOSUB; процедуры немного проигрывают. Однако я рекомендовал бы преимущественное использование механизма процедур, поскольку они предоставляют бóльшие возможности при работе с переменными.

РАСШИРЕННЫЙ ОПЕРАТОР ВЕТВЛЕНИЯ

Подобно определению функций оператор ветвления IF . . . THEN . . . . . ELSE в Quick Basic может включать выражения из нескольких строк. Эта возможность не столько увеличивает гибкость языка, сколько повышает удобочитаемость программ (см. пример на рис. 4).

Правила построения таких конструкций простые. Рассмотрим программный код на рис. 4. За ключевым словом THEN может следовать только комментарий. Ничего, кроме номера строки или метки, не может предшествовать другим ключевым словам, и блок должен заканчиваться сочетанием END IF. Блок кода может включать любые операторы, в том числе и другие операторы IF . . . THEN, но не может включать определения функций.

Хотя оператор DO . . . LOOP похож на оператор WHILE . . . WEND, эта форма цикла расширяет гибкость в проверке условия окончания цикла. Синтаксис цикла DO . . . . . LOOP может иметь одну из двух форм:

```
DO
. . .
LOOP(WHILE/UNTIL)
```

или

```
DO(WHILE/UNTIL)
. . .
LOOP
```

В первой форме цикл выполняется хотя бы один раз, даже если проверяемое условие ложно с самого начала. Данное условие может быть записано как с атрибутом WHILE (пока истинно), так и с атрибутом UNTIL (пока ложно). Это является еще одним преимуществом по сравнению с циклом WHILE . . . WEND.

Кроме того, введен дополнительный оператор принудительного выхода из цикла EXIT DO. Он обеспечивает безопасное средство для выхода из любого места тела цикла.

Модульное программирование

Рис. 5. Удаление из строки хвостовых пробелов

```
'This function strips spaces from the end of strings
    DEF FNA$(A$)=LEFT$(A$,INSTR(A$+"  ","  ")-1)
```

Конец ◄

Рис. 6. Две функции для обработки символов, вводимых с клавиатуры

```
'This function waits for keyboard input, then converts it to
'upper case
    DEF  FNINK$
         STATIC I$
    INK: I$=""
         I$=INKEY$
         IF I$="" THEN GOTO INK
         IF ASC(I$) > 96 AND ASC(I$) < 123 THEN _
              I$ = STRING$(1,ASC(I$)-32)
         FNINK$ = I$
    END  DEF
'This function inputs a line terminated by a CR and strips
'any spaces from the end
    DEF  FNLININ$
         STATIC A$
         LINE INPUT A$
         FNLININ$ = LEFT$(A$,INSTR(A$+"  ","  ")-1)
    END  DEF
```

Конец ◄

Рис. 7. Две процедуры для экранного редактирования текста

```
   defint i
   dim  g$(6), g%(6)
   g%(1)=15              'Set up field lengths
   g%(2)=20
   g%(3)=15
   g%(4)=20
   g%(5)=5
   g%(6)=8
   call prynt("First Name",10,10,0)         'Print field names
   call prynt("Last Name",11,10,0)
   call prynt("Street Address",12,10,0)
   call prynt("City, State",13,10,0)
   call prynt("Zip",14,10,0)
   call prynt("Remarks",15,10,0)
   for i=1 to 6                          'Establish field variables
       g$(i)=string$(g%(i)," ")
   next
   call editor(10,27,g$(),g%())
   call prynt("First Name",10,10,0)         'Print field names
   call prynt("Last Name",11,10,0)
   call prynt("Street Address",12,10,0)
   call prynt("City, State",13,10,0)
   call prynt("Zip",14,10,0)
   call prynt("Remarks",15,10,0)
   for i=1 to 6
       call prynt(g$(i),9+i,27,0)          'Display variables
   next
   call editor(10,27,g$(),g%())
'This subprogram combines the Locate and Print statements
'PR$ is the string to be printed
'R% is the row number, 1 - 25
'C% is the column number, 1 - 80
'X is 1 to turn cursor on, 0 to turn it off
SUB  PRYNT(PR$,R%,C%,X%)          STATIC
     LOCATE ,,0
     LOCATE R%,C%,0
     PRINT PR$;
     LOCATE R%,C%+LEN(PR$),X%
END  SUB
'This subprogram is a full screen editor
'ROW% and COLUMN% define the upper left corner of the window
'X$() is a string array containing the data
'X%() is an array containing the length of the fields in X$()
SUB  EDITOR  (ROW%,COLUMN%,X$(1),X%(1)) STATIC
     R=ROW% : C=COLUMN%
ED1: LOCATE R,C,1      'Place cursor
ED2: I$="" : I$=INKEY$ : IF I$="" THEN GOTO ED2
     K=ASC(I$)
     IF K>31 AND K<123 THEN              'Alphanumeric character
          MID$(X$(R-ROW%+1),C-COLUMN%+1,1)=I$
          PRINT I$;
          K=77
     END  IF
```

Продолжение ►

Новые средства Quick Basic способствуют модульному программированию. Модуль — это программная секция или подпрограмма, которую вы можете компилировать и отлаживать отдельно. Модуль может предназначаться только для одной программы или быть родовым и использоваться во многих различных программах.

Я написал несколько программных модулей на Quick Basic, которые вы можете использовать без изменений или модифицировать для решения ваших задач. Первые три модуля (рис. 5 и 6) — это определяемые функции, две из которых состоят из нескольких строк. Еще шесть модулей (рис. 7 – 9) представляют собой процедуры, включенные в некоторые простые программы для иллюстрации их работы.

Определяемые функции должны размещаться в начале использующей их программы. Вы можете включить эти функции в исходный файл вашей программы просто текстовым редактором. Процедуры для удобства использования нужно откомпилировать и занести в библиотеку.

ИСКЛЮЧЕНИЕ ПРОБЕЛОВ

Когда ваша программа сохраняет примененные в файле прямого доступа, используя оператор LSET, содержательная информация дополняется пробелами до длины поля, определенного в описании FIELD. Если вы начнете сравнивать эти данные со строками, введенными с клавиатуры, то эти дополнительные пробелы приведут к ошибкам сравнения. Программа на рис. 5 представляет определяемую функцию, которая удаляет ненужные пробелы, расположенные в конце строки. Эта функция ищет два расположенных подряд пробела и удаляет их, а также все последующие пробелы. Если существует только один избыточный пробел, функция вначале добавляет два пробела в конец строки.

В третьей строке процедуры на рис. 9 показано использование

этой функции. Вторая функция на рис. 6 использует этот алгоритм для удаления избыточных пробелов из строки, введенной с клавиатуры.

ОБРАБОТКА СИМВОЛОВ, ВВОДИМЫХ С КЛАВИАТУРЫ

На рис. 6 показаны две функции, состоящие из нескольких строк, назначение которых — обрабатывать символы, вводимые с клавиатуры. Первая функция ждет ввода одного символа, а получив его, преобразует строчную букву в прописную. Такое преобразование очень полезно для ввода данных под управлением меню, поскольку прикладной программе уже не придется разбираться, строчные или прописные буквы получены. Оператор STATIC описывает I$ как локальную переменную. Вторая функция на этом рисунке вводит строку с клавиатуры и удаляет из нее хвостовые пробелы.

ЭКРАННЫЙ РЕДАКТОР

Многие программы требуют от пользователя ввода данных в режиме заполнения некоторых формуляров. Оператор Бейсика INPUT не позволяет обеспечить необходимое удобство ввода. Большинство пользователей требуют возможностей экранного редактирования, подобных тем, которые обеспечивают текстовые процессоры. В программе на рис. 7 представлены две процедуры, которые предназначены для подобного рода применений.

Первая процедура, используя операторы LOCATE и PRINT, выполняет функцию печати в заданном месте экрана заголовков или других коротких строк. Курсор выключается и перемещается в заданное место. Печатается строка, курсор перемещается в конец этой строки и включается или выключается в зависимости от заданного значения аргумента.

Вторая процедура обеспечивает функции редактирования. Вызывающая ее программа должна установить два массива: строковый, содержащий введенные или отредактированные данные, и целых чисел, в котором определе-

► Продолжение

```
        IF K=0 THEN K=ASC(RIGHT$(I$,1))     'Extended character
        IF K=8 AND C>COLUMN% THEN           'Back space
             C=C-1
             LOCATE R,C,1
             K=83
        END  IF
        IF K=77 AND C<COLUMN%-1+X%(R-ROW%+1) THEN     'Right arrow
             C=C+1
             GOTO ED1
        END  IF
  Program Listing 3 (cont'd)
        IF K=75 AND C>COLUMN% THEN          'Left arrow
             C=C-1
             GOTO ED1
        END  IF
        IF K=79 THEN                        'End
             C=COLUMN%+X%(R-ROW%+1)-1
             GOTO ED1
        END IF
        IF K=71 THEN                        'Home
             C=COLUMN%
             GOTO ED1
        END IF
        IF K=80 AND R<ROW%+UBOUND(X$,1)-1 THEN          'Down arrow
             R=R+1
             C=COLUMN%
             GOTO ED1
        END  IF
        IF K=72 AND R>ROW% THEN             'Up arrow
             R=R-1
             C=COLUMN%
             GOTO ED1
        END  IF
        IF K=83 THEN                        'Del
             X$(R-ROW%+1)=LEFT$(X$(R-ROW%+1),C-COLUMN%)+ _
               MID$(X$(R-ROW%+1),C-COLUMN%+2,X%(R-ROW%+1)- _
               C+COLUMN%-1)+" "
             LOCATE R,COLUMN%,1
             PRINT X$(R-ROW%+1);
             GOTO ED1
        END  IF
        IF K=82 THEN                        'Ins
             X$(R-ROW%+1)=LEFT$(LEFT$(X$(R-ROW%+1),C-COLUMN%)+" "+ _
                  MID$(X$(R-ROW%+1),C-COLUMN%+1,COLUMN%),X%(R-ROW%+1))
             LOCATE R,COLUMN%,1
             PRINT X$(R-ROW%+1);
             GOTO ED1
        END  IF
        IF K <> 13 THEN GOTO ED1
        LOCATE ,,0
  END   SUB
```

Конец ◄

Рис. 8. Две процедуры для сортировки многомерных массивов и файлов прямого доступа

```
        dim a(1000,6)
        .
        .
        .
        call sort (a(),6)              'Perform sort on column 6
  'This subprogram will sort a two dimensional array
  'X() is the array
  'Y% is the column to sort on
  SUB   SORT (X(2),Y%) STATIC
        N=UBOUND(X,1) : Q=UBOUND(X,2)
        M=N
  SO1:  M=INT(M/2)                     'Find midpoint
        IF  M=0 THEN EXIT SUB          'Done
        K=N-M : J=1
  SO2:  I=J                            'Lower index
  SO3:  L=I+M                          'Upper index
        IF X(I,Y%)  > X(L,Y%) THEN GOTO SO5
  SO4:  J=J+1 : IF J > K THEN GOTO SO1
        GOTO SO2
  SO5:  FOR A=1 TO Q
        SWAP X(I,A),X(L,A)             'Exchange order
        NEXT
        I=I-M
        IF I < 1 THEN GOTO SO4
        GOTO SO3
  END   SUB
        .
        open "r",1,"sorts",30
        field 1, 10 as a$(1), 10 as a$(2), 10 as a$(3)
        field 1, 30 as b$
        .
        call sortfile (1,a$(2),b$,30)       'Perform sort on a$(2)
  'This subprogram will sort a random access file
  'X% is the number of the file
  'Y$ is the field variable to sort on
  'YY$ is a field variable encompassing the total field
  'F% is the length of the field
  SUB   SORTFILE  (X%,Y$,YY$,F%) STATIC
        N=LOF(X%)/F%
        M=N
  SF1:  M=INT(M/2)                     'Find midpoint
        IF  M=0 THEN EXIT SUB          'Done
        K=N-M : J=1
  SF2:  I=J                            'Lower index
```

Продолжение ►

► *Продолжение*

```
SF3: L=I+M                                  'Upper index
     GET X%,I
     I$=Y$ : II$=YY$
     GET X%,L
     IF I$ > Y$ THEN GOTO SF5
SF4: J=J+1 : IF J > K THEN GOTO SF1
     GOTO SF2
SF5: PUT X%,I                               'Exchange records
     LSET YY$=II$
     PUT X%,L
     I=I-M
     IF I < 1 THEN GOTO SF4
     GOTO SF3
END  SUB
```

Конец ◄

Рис. 9. Две процедуры для поиска данных в массивах и в файлах прямого доступа

```
     def fna$(a$)=left$(a$,instr(a$+"  ","  ")-1)
     dim a$(127,2)
     .
     .
     .
     input b$                                   'Input search key
     call arraysearch (a$(),b$,1,d%)            'Perform search
     if d% <> 0 then print a$(d%,1),a$(d%,2)  'Print the result
'This subprogram will search a two-dimensional array
'Calling program must define function FNA$
'X$() is the array
'Y$ is the search key
'M% is the column of the array to be searched
'N% is the index of the row where the match was found
'N% returns 0 if no match was found
SUB  ARRAYSEARCH (X$(2),Y$,M%,N%)  STATIC
     I = INT(UBOUND(X$,1)/2 + 1) : Y=I          'Set I to middle of list
SE1: A$=FNA$(X$(I,M%))
     IF A$ <> "" THEN GOTO SE3                  'Skip blank records
     Z=I
SE2: I=I-1
     IF I<1 THEN
          I=Z
          GOTO SE6
     ELSE
          A$= FNA$(X$(I,M%))
     END IF
     IF A$ = "" THEN GOTO SE2
     I=Z
SE3: IF INSTR(A$,Y$)<>0 THEN N%=I : EXIT SUB
     IF A$ > Y$ THEN GOTO SE5
     IF A$ < Y$ THEN GOTO SE6
SE5: IF Y/2 < 1 THEN N%=0 : EXIT SUB
     Y = INT(Y/2 + .5)                  'Cut interval in half
     I=I-Y : IF I<1 THEN I=1
     GOTO SE1
SE6: IF Y/2 < 1 THEN N%=0 : EXIT SUB
     Y = INT(Y/2 + .5)                  'Cut interval in half
     I=I+Y : IF I>UBOUND(X$,1) THEN I=UBOUND(X$,1)
     GOTO SE1
END  SUB
     .
     def fna$(a$)=left$(a$,instr(a$+"  ","  ")-1)
     open "r",1,"testfile",30
     field 1, 10 as a$, 10 as b$, 10 as c$
     .
     line input d$                              'Input search key
     call filesearch (1,b$,d$,30,k%)            'Perform search
     if k% = 0 then ....
     get 1,k%                                   'Read file
     print a$, b$, c$                           'Print the result
'This subprogram will search a random access file
'Calling program must define function FNA$
'X% is the file number
'X$ is the field variable to be searched
'Y$ is the search key
'F% is the length of the field
'N% is the record number where the match was found
'N% returns 0 if no match is found
SUB  FILESEARCH (X%,X$,Y$,F%,N%)   STATIC
     I = INT(LOF(X%)/F%/2 + 1) : Y=I            'Set I to middle of list
FS1: GET 1,I : A$=FNA$(X$)
     IF A$ <> "" THEN GOTO FS3                  'Skip blank records
     Z=I
FS2: I=I-1
     IF I<1 THEN
          I=Z
          GOTO FS6
     ELSE
          GET 1,I
          A$= FNA$(X$)
     END IF
     IF A$ = "" THEN GOTO FS2
     I=Z
FS3: IF INSTR(A$,Y$)<>0 THEN N%=LOC(X%) : EXIT SUB
     IF A$ > Y$ THEN GOTO FS5
     IF A$ < Y$ THEN GOTO FS6
FS5: IF Y/2 < 1 THEN N%=0 : EXIT SUB
     Y = INT(Y/2 + .5)                  'Cut interval in half
     I=I-Y : IF I<1 THEN I=1
     GOTO FS1
FS6: IF Y/2 < 1 THEN N%=0 : EXIT SUB
     Y = INT(Y/2 + .5)                  'Cut interval in half
     I=I+Y : IF I>LOF(X%)/F% THEN I=LOF(X%)/F%
     GOTO FS1
END  SUB
```

Конец ◄

ны длины полей. Редактор работает в окне, длина которого в строках равна числу элементов в строковом массиве, а ширина, разная для каждой строки, равна длине поля. Переменные ROW% и COLUMN% содержат координаты левого верхнего угла окна.

Клавиши со стрелками управляют движением курсора по окну. Нажатия клавиш "Стрелка вверх" и "Стрелка вниз" приводят к перемещению курсора в начало предыдущей и следующей строк соответственно, а нажатия клавиш "Home" и "End" – к перемещению курсора в начало или конец строки. При нажатии клавиши "Insert" открывается строка для ввода дополнительных символов, а при нажатии клавиши "Delete" стирается символ, под которым расположен курсор. Клавиша "Back-space" выполняет обычную для себя функцию – при ее нажатии стирается последний введенный символ. Нажатие клавиши "Enter" приводит к окончанию редактирования в данном окне.

Главная программа на рис. 7 иллюстрирует начальную установку массивов для ввода данных в пустой формуляр и для редактирования уже введенных данных. Первый раздел программы описывает размеры массивов и устанавливает длины полей. Вы можете обмениваться данными с файлом прямого доступа или хранить их в двумерном массиве в ОЗУ. Второй раздел программы выводит на дисплей имена полей и устанавливает переменные полей. В цикле FOR. . . NEXT в переменные заносятся пробелы. В третьем разделе программы показано редактирование уже введенных данных. Цикл FOR. . . NEXT выводит значения переменных на дисплей.

ПРОЦЕДУРЫ СОРТИРОВКИ

На рис. 8 показаны процедуры для сортировки массива и файла прямого доступа. Обе используют одинаковый алгоритм сортировки; различия между процедурами связаны с разными источниками данных. Два небольших программных фрагмента иллюст-

рируют применение этих процедур.

Процедура SORT сортирует данные в колонках двумерного массива. Используется числовой массив, но процедуру нетрудно изменить для сортировки массивов строк. Поскольку тип аргументов в описании процедуры должен соответствовать типу фактических аргументов при вызове процедуры, одна процедура не сможет сортировать данные разных типов.

Процедура SORTFILE сортирует поля в файле прямого доступа. Она рассматривает переменные как элементы массива, но это не обязательно. Обратите внимание, что в вызывающем эту процедуру программном коде стоят два оператора определения поля. Второе описание определяет всю запись файла как одно поле и позволяет обменивать данные разных записей. Поскольку процедура сравнивает записи файла только друг с другом, то для нее безразлично, какого типа данные содержит этот файл.

ПРОЦЕДУРЫ ПОИСКА

На рис. 9 представлены две процедуры поиска: одна в массивах и другая для поиска в файле. В обоих случаях исходные данные должны быть расположены в возрастающем порядке. Обе процедуры используют алгоритм двоичного поиска, который делит весь набор данных на уменьшающиеся половины, пока не будет найден требуемый ключ или показано его отсутствие. Обратите внимание на использование функции, удаляющей хвостовые пробелы из данных. Аргумент N% возвращает индекс или номер записи при успешном поиске или 0 при неудачном поиске.

Процедура поиска в массиве ARRAYSEARCH работает с массивами любого размера. В том виде, в каком она представлена, процедура поиска в файле FILESEARCH может работать только с файлами, содержащими строки. Чтобы она работала с числовыми файлами, необходимо вставить соответствующую функцию преобразования в строку с меткой FS1. Эта функция работает с любым числом переменных в поле и с полями любой длины.

Синтаксис его имеет следующую форму:

```
DO
. . .
IF X>Y THEN EXIT DO
. . .
LOOP UNTIL EOF(1)
```

Оператор EXIT FOR обеспечивает такие же возможности в теле оператора цикла FOR . . . NEXT.

ОПЕРАТОР ВЫБОРА SELECT CASE

Этот оператор похож на оператор ON GOSUB, но является более мощным. Он выполняет один из нескольких блоков кода в зависимости от значения некоторого выражения. Следующий пример показывает синтаксис оператора:

```
SELECT CASE X
      CASE 15
      (Statements)
      CASE 2 TO 5,7,8,10 TO 12
      (Statements)
      CASE IS <1
      (Statements)
      CASE ELSE
      (Statements)
END SELECT
```

Вместо переменной X может стоять любое выражение, в том числе и строковое. Когда значение этого выражения удовлетворяет условию, стоящему после ключевого слова CASE, выполняется соответствующий блок кода. Блок, стоящий после CASE ELSE, выполняется, если ни одно другое условие CASE не выполняется. Вы можете использовать его для обработки значений по умолчанию. После выполнения этого оператора управление передается на строку, следующую за ключевыми словами END SELECT.

ОПЦИИ КОМПИЛЯЦИИ

В системе Quick Basic вы можете компилировать программу для отладки прямо в оперативную память, не выходя из среды редактора. Этот метод почти не отличается от выполнения программы интерпретатором. Когда программа, наконец, заработает, загрузка системы Quick Basic для ее выполнения будет излишней. Откомпилируйте вашу программу с опцией EXE, и будет создан выполнимый файл типа .EXE.

Как уже ранее упоминалось, для компиляции процедур, заносимых в библиотеку, надо использовать опцию BRUN.LIB. Эта же опция необходима, если вы хотите откомпилировать модули для последующей сборки из них выполняемой программы стандартным компоновщиком MS-DOS. Это единственный способ создания больших (занимающих более 64 Кбайт) программ. Для этого необходимо, чтобы файл BRUN20.LIB находился в текущем каталоге во время компиляции и файл BRUN20.EXE (содержащий модули периода выполнения) находился в период выполнения в текущем каталоге или в каталоге, указанном командой PATH.

Если вы используете дискеты, то ваша прикладная программа и файл BRUN20.EXE должны быть на одной дискете, но файл BRUN20.EXE не обязательно должен быть на одной дискете с компилятором. Если вы работаете с твердым диском, то файл BRUN20.EXE может располагаться где угодно, если путь к нему указан командой PATH.

Компиляция с опцией BCOM20.LIB создает объектный файл, который после компоновки не требует для выполнения модуля BRUN20.EXE. Программа, откомпилированная таким образом, занимает больше места на диске, но выполняется немного быстрее. Файл BCOMP20.LIB во время компиляции должен находиться в текущем каталоге.

ДРУГИЕ УСОВЕРШЕНСТВОВАНИЯ

Стоит упомянуть еще о нескольких новых возможностях Quick Basic. Функции COMMAND$ и ENVIRON$ обеспечивают вашей программе доступ к командной строке, полученной операционной системой, и к таблице описания среды Бейсика соответственно. Оператор ENVIRON модифицирует параметры в таблице описания среды операционной системы DOS.

Quick Basic предоставляет три процедуры, которые, находясь в библиотеке пользователя, обеспечивают доступ к функциям DOS и BIOS. Процедуры INT86 и INT86X вырабатывают программные прерывания. Первая устанавливает только несегментные регистры, вторая устанавливает все регистры. Массив целых чисел служит для обмена значениями регистров с процедурой прерывания. Процедура PTR86 преобразует адрес большого массива чисел в адрес смещения в сегменте для использования с процедурой INT86X.

Quick Basic поддерживает привычные статические массивы, но, кроме того, обеспечивает и динамические массивы. Система распределяет память для статических массивов во время компиляции, а для динамических массивов во время выполнения. Статические массивы занимают чуть меньше места, чем динамические, и доступ к ним немного быстрее. Динамические массивы могут увеличиваться и уменьшаться в процессе выполнения программы и таким образом позволяют использовать память более экономно. Для задания динамического массива надо описать его с переменным размером:

```
DIM A(N)
```

Метакоманды, размещаемые в комментариях, описывают некоторые функции, выполняемые во время компиляции. В Quick Basic возможны три метакоманды: $INCLUDE, $STATIC и $DINAMIC. Команда $INCLUDE загружает с диска и вставляет вместо себя файл исходного кода. Это еще один способ модульного построения программ.

Команды $STATIC и $DINAMIC переключают статус массива, заданный по умолчанию. Например, массив, описание которого встречается после команды $STATIC, рассматривается как статический независимо от того, в какой форме задан оператор описания массива DIM.

ЗАКЛЮЧЕНИЕ

Quick Basic включает в себя множество новых средств, каждое из которых оправдывает цену пакета само по себе. Собранные вместе, они образуют отличную систему. Если вы еще не начали программировать, я советую вам купить пакет Quick Basic 3.0 и начать.

Управление окнами

Библиотека управления окнами Quick Windows, предназначенная для работы с системой программирования Quick Basic версии 4.0 или более ранних версий, поможет вам в создании окон, разнообразных меню, окон для справочной информации и формуляров для ввода данных. Для доступа к 60 функциям библиотеки Quick Windows и к ассемблерным функциям необходимо использовать оператор Бейсика CALL.

Когда вы открываете окно, Quick Windows автоматически сохраняет информацию, которую это окно стирает, при закрывании окна содержимое экрана восстанавливается. Вы можете управлять любой характеристикой окна, его размерами и атрибутами. Текст автоматически продвигается в виде свитка по мере заполнения окна, и каждое окно имеет свой собственный курсор. Перемещением курсора можно управлять как с клавиатуры, так и посредством "мыши", совместимой с пакетом управления "мышью" фирмы Microsoft.

Пакет Quick Windows поставляется за 79 дол без исходных текстов и за 99 дол с исходными текстами. Адрес для контактов: Software Interphase Inc., 5 Bradly St., Providence, RI 02908, 401–274–5465.

Усовершенствования *Quick Basic*

Quick Basic версии 4.0 фирмы Microsoft сочетает скорость выполнения с диалоговыми возможностями интерпретатора. Если вы остановите выполнение программы и внесете в нее изменения, то компилятор Quick Basic обработает эти изменения со скоростью 150 000 строк/мин при работе на персональном компьютере типа AT с рабочей частотой 8 МГц.

Кроме возросшей скорости выполнения язык Quick Basic обладает такими новыми возможностями, как определяемые пользователем структуры данных (записи), иерархическая организация текста программы, интеграция с отладчиком Codeview и совместимость с другими языками программирования фирмы Microsoft. Кроме того, возможны быстрая проверка синтаксиса, покомандное тестирование и отладка. Программа может состоять из многих модулей.

Предполагаемая цена 99 дол; расширения доступны пользователям версии 3.0 за 25 дол, пользователям более ранних версий за 35 дол.

Адрес для контактов: Microsoft Corp., 16011 N. E. 36th Way, Box 97017, Redmont, WA 98073, 206–882–8080.

Сравнение компиляторов Quick Basic *4.0 и* Turbo Basic *1.1 показывает, что хотя загрузка файла в память и компиляция с записью на диск компилятором* Quick Basic *выполняются медленнее, высокая скорость компиляции в оперативной памяти и средства отладки на уровне исходного текста при его использовании обеспечивают меньшие затраты времени на разработку программ.*

Работа с адаптером *EGA* на Бейсике

ДЖОН ВУЛФСКИЛЛ

Если бы проблемы совместимости занимали первые полосы газет, то однажды вы могли бы прочесть сообщение: "Гибель модема при превышении скорости работы на AT-подобном компьютере".

Проблемы совместимости, как и сенсации или несчастные случаи, нельзя предвидеть заранее. Обычно они возникают при столкновении старой и новой техники – побочном явлении прогресса. Хорошим примером сказанного является противоречие между адаптером EGA (Enhanced Graphics Adapter) и языком программирования Бейсик.

Стандарт EGA позволяет программистам, работающим с версией 3.2 языка GW-Basic фирмы Microsoft, создавать программы, использующие возможности графики высокого разрешения. Можно выводить изображения на экран размером 640X350 точек, используя 16 цветов из палитры в 64 цвета. Это гораздо лучше, чем 4 цвета и относительно низкая разрешающая способность адаптера CGA (Color Graphics Adapter). Однако при работе с адаптером EGA нельзя пользоваться некоторыми простыми приемами программирования, которые применялись для адаптера CGA. Одним из таких приемов является запись образа экрана на диск.

КАК ЭТО ДЕЛАЕТСЯ

Многие программирующие на Бейсике привыкли для записи содержимого буфера адаптера CGA на диск пользоваться оператором BSAVE. Это быстро и удобно, к тому же не требует больших затрат при программировании. Вот как это делается:

```
DEF SEG=&HB800
BSAVE"имяфайла.рас",0,&H400
DEF SEG
```

Восстановить образ экрана с диска столь же просто:

```
DEF SEG=&HB800
BLOAD"имяфайла.рас",0
DEF SEG
```

Хотя этот прием никогда не имел официального признания, он стал очень популярным и широко используется при работе с видеодисплеем. К сожалению, конструктивное решение адаптера EGA не позволяет для записи образа экрана в файл использовать операторы BLOAD и BSAVE.

Причины этого так же сложны, как и сам адаптер EGA. Организация его постоянной памяти обеспечивает широкие возможности, но очень сложна. Каждая из четырех страниц видеопамяти адаптера EGA разбита на непрерывные области памяти, называемые слоями. В каждом 16-Кбайтном слое содержатся данные о цвете точек экрана, относящиеся к одному из четырех бит цвета (интенсивность, красный, синий и зеленый), комбинируя которые можно составить любые 16 из 64 цветов.

Для записи содержимого видеопамяти адаптера EGA программа должна управлять компьютером примерно так же, как водитель управляет грузовиком: с помощью рук и ног. Одной "рукой" она должна манипулировать различными регистрами адаптера EGA, а в это время DOS и ПЗУ с BIOS другой "рукой" будут "загонять" информацию на диск. Чтобы обратиться к видеопамяти, не используя BIOS адаптера EGA, нужно непосредственно манипулировать регистрами адаптера EGA. Наиболее важными из них являются регистры Map Mask и Address Sequencer. Вместе они действуют как сдвиговый механизм для выбора отдельных слоев видеопамяти, к которым сможет обратиться ЦП.

При работе с адаптером EGA нельзя просто загрузить информацию в видеопамять и надеяться, что она будет выведена на экран: это можно делать с контроллером Motorola 6845 адаптера CGA. Однако существует способ, позволяющий включить соответствующие средства в программу на Бейсике и получить тем самым возможность использовать преимущества адаптера EGA.

РЕШЕНИЕ ПРОБЛЕМЫ

Программа 1 (см. распечатку) создает резидентную программу ADDEGA.COM. Ее нужно загрузить только один раз – в начале сеанса работы на компьютере. Программа ADDEGA.COM становится логическим расширением DOS и до перезагрузки системы будет выполнять функции, обеспечивающие запись и чтение содержимого видеопамяти адаптера EGA.

Чтобы использовать программу ADDEGA.COM совместно с вашими программами на Бейсике, введите и запустите программу 1, а затем выйдите из интерпретатора Бейсика. В ответ на запрос ввода команды DOS введите

```
ADDEGA
```

Программа 1. Программа ADDEGA.BAS создает резидентную программу ADDEGA.COM, которая записывает и восстанавливает содержимое видеопамяти адаптера EGA

```
10 ON ERROR GOTO 330
20 CLS:PRINT"CREATING ADDEGA.COM"
30 OPEN "ADDEGA.COM" FOR OUTPUT AS #1
40 FOR BYTE=1 TO 4
50   READ VALUE$
60   PRINT#1,CHR$(VAL("&H"+VALUE$));
70 NEXT BYTE
80 FOR BYTE=5 TO 1556
90   PRINT#1,CHR$(0);
100 NEXT BYTE
110 FOR BYTE=1557 TO 1802
120   READ VALUE$
130   PRINT#1,CHR$(VAL("&H"+VALUE$));
140 NEXT BYTE
150 CLOSE 1
160 LOCATE 1,1:PRINT"COMPLETE ..          ":END
170 DATA E9,11,06,31,FC,1E,B8,00,00,8E,C0,26,8B,36,A8,04
180 DATA 26,A1,AA,04,8E,D8,BF,38,01,B9,0E,00,07,F3,A5,0E
190 DATA 1F,8B,36,38,01,BF,54,01,0E,07,8E,D8,B9,C0,05,F3
200 DATA A4,0E,1F,8C,1E,3A,01,B8,54,01,A3,38,01,B8,00,00
210 DATA 8E,C0,A1,3A,01,26,A3,AA,04,B8,38,01,26,A3,A8,04
220 DATA BA,CA,08,CD,27,55,16,1E,0E,1F,BA,05,01,B9,00,00
230 DATA B4,3C,CD,21,72,2A,A3,36,01,83,03,BA,CE,03,B0,04
240 DATA EE,8A,C3,42,EE,53,8B,1E,36,01,1E,B8,00,A0,8E,D8
250 DATA BA,00,00,B4,40,B9,60,6D,CD,21,1F,5B,FE,CB,7D,DB
260 DATA B4,3E,8B,1E,36,01,CD,21,1F,17,5D,CB,55,16,1E,0E
270 DATA 1F,BA,05,01,B4,3D,B0,00,CD,21,72,E4,A3,36,01,8B
280 DATA 1E,36,01,B8,00,A0,8E,D8,BA,CE,03,B0,08,EE,B0,FF
290 DATA 42,EE,BA,CE,03,B0,05,EE,B0,00,42,EE,BA,CE,03,B0
300 DATA 03,EE,42,B0,00,EE,B1,03,8A,C4,03,B0,02,B4,01,D2
310 DATA E4,EE,8A,C4,42,EE,51,B4,3F,B9,60,6D,BA,00,00,CD
320 DATA 21,59,72,9C,FE,C9,7D,E0,EB,96
330 PRINT"BASIC ERROR";ERR;"IN LINE";ERL:STOP
```

Конец ◄

Программа 2. Эта программа связывает вашу программу на Бейсике с программой ADDEGA.COM

```
10 SCREEN 9:KEY OFF:COLOR 15,0:CLS
20 DIM PROMPTBUFFER%(3000)
30 'DISPLAY EGA SCREEN
40 LINE(0,13)-(639,337),4,BF
50 FOR J=1 TO 15
60   LOCATE J+4,20
70   PRINT STRING$(40,178);
80   COLOR J
90 NEXT J
100 GET(0,0)-(540,13),PROMPTBUFFER%
110 LOCATE 1,1:INPUT"[1] LOAD [2] SAVE";A$
120 IF A$<"1" OR A$>"2" THEN CLS:END
122 GOSUB 140:GOTO 100
130 ' ** ADDEGA.COM LINK SUBROUTINE **
140 LOCATE 1,1:PRINT STRING$(58,32);
150 LOCATE 1,1:INPUT"Pathname";EGAFILE$ ' GET PATHNAMME
160 PUT(0,0),PROMPTBUFFER%,PSET          ' RESTORE SCREEN
170 EGAFILE$=EGAFILE$+CHR$(0)            ' ADD ASCII ZERO
180 LOADEGA%=&H7AC:SAVEEGA%=&H765        ' DEFINE OFFSETS
190 DEF SEG=0                            ' EGA SAVE PTR
200 SAVEPTR%=PEEK(&H4AB)*256+PEEK(&H4AA)' FIND ADDEGA.COM
210 DEF SEG=SAVEPTR%                     ' CS TO ADDEGA.COM
220 FOR J=1 TO LEN(EGAFILE$)             ' TRANSFER PATHNAME
230 POKE &H104+J,ASC(MID$(EGAFILE$,J,1))' TO ADDEGA.COM
240 NEXT J                               ' MAX 49 BYTES
250 IF A$="1" THEN CLS:CALL LOADEGA%     ' LOAD EGA SCREEN
260 IF A$="2" THEN CALL SAVEEGA%         ' SAVE EGA SCREEN
270 RETURN
```

Конец ◄

Затем снова загрузите интерпретатор Бейсика и запустите программу 2 (см. распечатку).

В строках 10–100 задается переключение адаптера EGA в режим высокого разрешения (640X350 точек) и формируется тестовое изображение для загрузки и записи. Подпрограмму, начинающуюся в строке 140, можно включать в любую программу на Бейсике. Эта подпрограмма свяжет вашу программу с программой ADDEGA.COM. В строках 140–160 запрашивается полное имя файла (pathname), в который будет записан образ экрана. В строке 170 к заданному вами имени добавляется признак окончания полного имени файла CHR$(0). В строке 180 в указатель команд (IP) заносится значение смещения для входящих в состав программы ADDEGA.COM средств ввода-вывода с диска.

Ссылка на адрес, по которому интерпретатор Бейсика размещает программу ADDEGA.COM в памяти, обеспечивается указателем таблицы параметров BIOS адаптера EGA (в распечатке BIOS адаптера EGA фирмы IBM он обозначен Save_PTR). Если в вашей системе используется адаптер EGA, этот 4-байтовый вектор (всегда находящийся по адресу 0000:04A8) указывает на таблицу из семи 4-байтовых указателей, первый из которых является адресом 1472-байтовой таблицы, содержащей по 64 байта данных на каждый из 23 режимов работы адаптера EGA. В BIOS адаптера EGA информация используется для форматирования экрана и т. п.

Программа ADDEGA.COM пересылает эти семь указателей и таблицу параметров в свою собственную область данных, а затем, прежде чем разместить в памяти свои средства ввода-вывода, переопределяет указатель Save_PTR в соответствии со своим новым адресом.

В строке 200 выполняется чтение указателя Save_PTR (теперь это адрес сегмента с программой ADDEGA.COM) и к значению IP прибавляются смещения для переменных LOADEGA% и SAVEEGA%, заданные в строке 180. Таким образом происходит формирование полного межсегментного адреса вызова, что обеспечивает вашей программе прямой доступ к средствам ввода-вывода программы ADDEGA.COM.

Чтобы программа 2 могла работать с компилятором Turbo Basic фирмы Borland International, ее нужно скорректировать следующим образом: вместо операторов CALL интерпретатора подставить в строку 250 оператор CALL ABSOLUTE LOADEGA%, а в строку 260 – оператор CALL ABSOLUTE SAVEEGA%. После этого программу можно компилировать в обычном порядке. Если вы используете компилятор Quick Basic фирмы Microsoft, то замените указанные строки следующими:

```
CALL ABSOLUTE (LOADEGA%) и
CALL ABSOLUTE (SAVEEGA%)
```

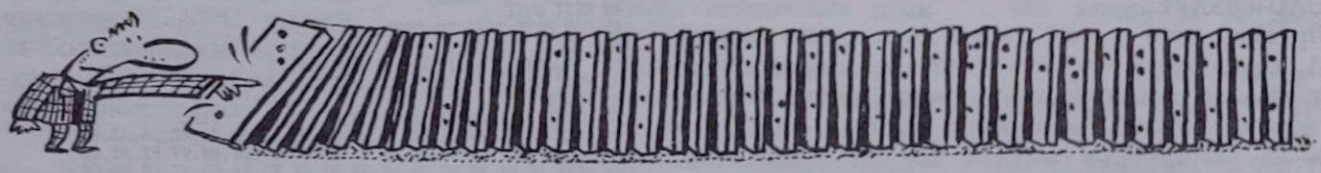
соответственно. Запишите файл с исходным кодом Бейсика (файл с расширением .BAS) на диск, затем вернитесь к запросу ввода команды DOS. Воспользуйтесь методом раздельной компиляции для получения выполняемого файла (файл с расширением .EXE) компилятора Quick Basic:

```
QB имяфайла.BAS /V/D/O;
LINK имяфайла.OBJ;
```

где имяфайла – то имя, которое вы выбрали для файла, содержащего программу 2.

Для использования программы ADDEGA.COM вам не нужно знать правила программирования адаптера EGA. Просто введите тексты приведенных здесь программ и получите удовольствие от работы с адаптером EGA.

Волшебные клавиши



Я написал программу на Бейсике, которая позволяет выполнить ряд программ или команд DOS, нажав одну клавишу. Эту программу можно использовать для изменения функций большинства клавиш вашего ПК, задания цветов для экрана и управления курсором.

Программа AUTOMAC.BAS (см. распечатку) создается пакетный файл AUTOMAC.BAT на заданном вами дисководе. Вы можете также включить файл AUTOMAC.BAT в файл AUTOEXEC.BAT, и тогда при включении компьютера будут автоматически переопределяться функции заданных клавиш. В отличие от стандартных пакетных файлов, файл AUTOMAC позволяет автоматически вводить данные в прикладную программу из отдельного файла, а не с клавиатуры, не используя при этом переназначения ввода в системе. Для обеспечения автоматического запуска большинства прикладных программ можно заранее сформировать данные, которые должны вводиться с клавиатуры.

В качестве примера работы файла AUTOMAC предположим, что вам надо поставить в соответствие комбинации клавиш Ctrl-A следующее макроопределение:

```
A:
COPY FILE1.DOC C:\ WP
COPY FILE2.DOC C:\ WP
BASIC AUTOMATE.BAS
PRINT\ PRN FILE1.DOC
DIR | SORT
C:
```

Когда вы нажмете комбинацию клавиш Ctrl-A, DOS скопирует файлы FILE1.DOC и FILE2.DOC на диск C, загрузит интерпретатор Бейсика, запустит программу AUTOMATE.BAS и выведет на экран отсортированный каталог диска A. Любым 16 клавишам (за исключением клавиш "," и ";", которые могут вызвать ошибки при интерпретации строковых переменных Бейсика) можно поставить в соответствие до 200 символов.

Если у вашего ПК цветной монитор, то можно изменить цвета на экране, вставив в макроопределение команду PROMPT:

```
PROMPT $e37m
PROMPT $e44m
PROMPT $e1m
PROMPT
```

Обязательно нажимайте клавишу Enter после окончания каждой строки. Это макроопределение дает указание драйверу экрана и клавиатуры, соответствующему стандарту ANSI, установить голубой цвет фона на экране, белый цвет символов и интенсивный белый цвет основного изображения.

Прежде чем обратиться к файлу AUTOMATE.BAS, убедитесь, что файл утилиты ANSI.SYS находится на системном диске, и добавьте в файл CONFIG.SYS следующую строку:

```
DEVICE=ANSI.SYS
```

Введите с клавиатуры программу AUTOMATE.BAS и задайте макроопределения или переопределите функции клавиш. Когда вы будете готовы записать файл на диск, нажмите Ctrl-C. В ответ на запрос ввода команды DOS введите

```
TYPE AUTOMAC.BAT
```

Чтобы файл AUTOMAC.BAT устанавливался при запуске стартового файла, добавьте приведенную выше строку в файл AUTOEXEC.BAT. Нажимая на клавишу Alt и одновременно набирая на клавиатуре три соответствующие цифры, можно задавать макроопределения для значений со 128 по 255 кода ASCII. Однако я не рекомендую задавать макроопределения для клавиш, которые вы используете для ввода команд. Функции клавиш остаются переопределенными до тех пор, пока вы либо не переопределите их заново, либо не перезагрузите систему.

К сожалению, не все прикладные программы обращаются к указанному драйверу клавиатуры и экрана. К таким программам относятся GW=Basic и некоторые редакторы текста.

Дж. Вулфскилл

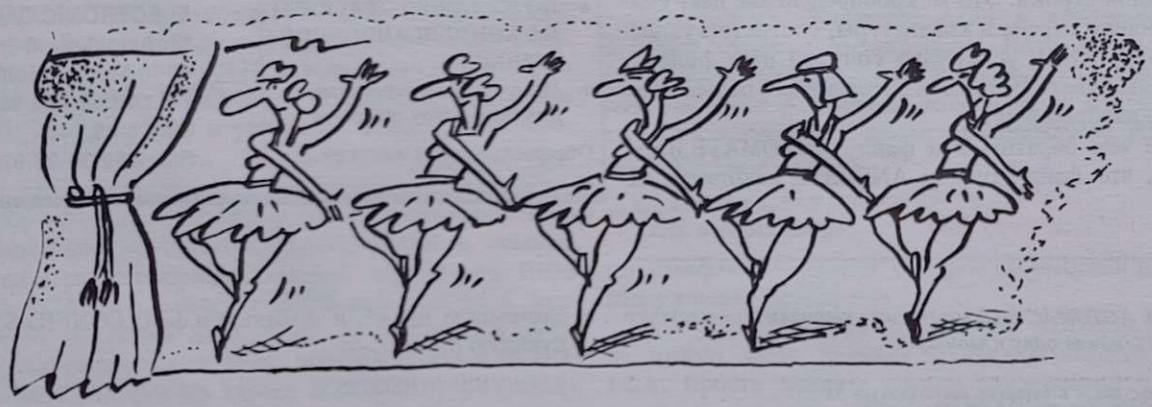
Программа AUTOMAC.BAS позволяет выполнить несколько команд DOS, нажав одну клавишу.

```
5 '*** AUTOMAC.BAS - A keyboard macro utility **
10 CLS:DIM K$(17):KEY OFF:CLS:CT=200
20 FOR J=1 TO 10:KEY J,"":NEXT
30 INPUT"Target [drive]/[pathname]";PATH$
40 CLS:I=I+1:PRINT"REMAINING: KEYS=";17-I;" CHARACTERS=";CT
50 PRINT:PRINT"[CTRL][C] to quit
60 IF I>16 THEN 300
70 PRINT:PRINT"Press key to assign/reassign ?";
80 X$=INKEY$:IF X$="" THEN 80
90 IF X$=CHR$(3) THEN 300
100 OLDKEY$=STR$(ASC(RIGHT$(X$,1)))+";"
110 OLDKEY$=RIGHT$(OLDKEY$,LEN(OLDKEY$)-1)
120 IF LEN(X$)=2 THEN OLDKEY$="0;"+OLDKEY$
130 PRINT ASC(X$);"[";X$;"]"
140 PRINT"[M]acro string or [K]ey reassignment ?"
150 X$=INKEY$:IF X$="" THEN 150
160 IF X$="M" OR X$="m" THEN GOSUB 340:GOTO 230
170 PRINT:PRINT"Press new key : ";
180 X$=INKEY$:IF X$="" THEN 180
190 NEWKEY$=STR$(ASC(RIGHT$(X$,1)))
200 NEWKEY$=RIGHT$(NEWKEY$,LEN(NEWKEY$)-1)
210 IF LEN(X$)=2 THEN NEWKEY$="0;"+NEWKEY$
220 PRINT ASC(X$);"[";X$;"]"
230 K$(I)=CHR$(27)+"["+OLDKEY$+NEWKEY$+"p"
240 CLS:PRINT"*** CURRENT ASSIGNMENTS ***"
250 FOR J=1 TO 16
260 IF K$(J)>"" THEN PRINT K$(J)
270 NEXT J
280 PRINT:PRINT"Press [ANY KEY] to continue..";
290 X$=INKEY$:IF X$<>"" THEN 40 ELSE 290
300 OPEN"O",1,PATH$+"AUTOMAC.BAT"
310 I=1:PRINT#1,"echo off"+CHR$(13)
320 IF I>16 OR K$(I)="" THEN PRINT#1,CHR$(26):CLOSE 1:CLS:SYSTEM

330 PRINT#1,K$(I):I=I+1:GOTO 320
340 PRINT"Type macro string    Press [TAB] to end":PRINT"? ";:JJ
    =0
350 X$=INKEY$:IF X$="" THEN 350
360 IF X$=CHR$(9) OR CT-JJ<1 THEN GOSUB 400:RETURN
370 IF X$=CHR$(8) THEN PRINT:NEWKEY$="":GOTO 340
380 NEWKEY$=NEWKEY$+X$:PRINT X$;:JJ=JJ+1
390 GOTO 350
400 NEWKEY$=CHR$(34)+NEWKEY$+CHR$(34):CT=CT-JJ:RETURN
```

Конец ◄

Горизонтальная прокрутка



Программа HSCROLL

```
10 SCREEN 0:COLOR 15,0
20 KEY OFF:CLS:DIM J$(14)
30 J$(1)="**********************************
                                          **
40 J$(2)="**                                **
50 J$(3)="**     PRESS ANY KEY TO BEGIN     **
60 J$(4)="**                                **
70 J$(5)="**********************************
80 FOR J=1 TO 39
90   READ A
100  SCROLL$=SCROLL$+CHR$(A)
110 NEXT J
120 ' ** PRINT TILE BLOCK **
130 FOR N=1 TO 14
140   LOCATE 24,25:PRINT J$(N)
150 NEXT N
160 ' ** WAIT FOR KEY **
170 X$=INKEY$
180  IF X$="" THEN 170 ELSE GOSUB 200:RUN
190 ' ** SCROLL DISPLAY LEFT **
200 SCROLL=VARPTR(SCROLL$)
210 SCROLL1=PEEK(SCROLL+2)*256+PEEK(SCROLL+1)
220 FOR J=1 TO 24+LEN(J$(1))
230   CALL SCROLL1
240 NEXT J
250 RETURN
260 ' **
270 DATA 30,6,180,6,176,8,181,8,177
280 DATA 8,182,24,178,8,183,7,205,16
290 DATA 184,8,184,142,192,142,216,198
300 DATA 2,8,191,8,8,185,203,7,243,165
310 DATA 7,31,203
```

Конец ◀

Программисты, работающие на машинном языке, используют для быстрой вертикальной прокрутки функциональные вызовы ROM BIOS или прокрутку с помощью аппаратных средств. Еще проще запрограммировать вертикальную прокрутку на Бейсике. Просто выведите текст в последнюю строку экрана, и Бейсик сам выполнит прокрутку. К сожалению, ни в Бейсике, ни в ROM BIOS нет средств для горизонтальной прокрутки. Чтобы восполнить этот пробел, я написал программу HSCROLL (см. распечатку).

Программа HSCROLL занимает 39 байт и позволяет в программах на Бейсике выполнять горизонтальную прокрутку. В строках 80–110 данные на машинном языке считываются в строковую переменную. В строках 200–250 осуществляется прокрутка на один символ влево при каждом вызове подпрограммы SCROLL1.

Программа HSCROLL стирает крайний левый символ во всех строках экрана, используя для этого программу вертикальной прокрутки ROM BIOS. Затем каждый символ текста и соотвествующий ему байт атрибута сдвигаются влево на одну колонку.

Дж. Вулфскилл

Разные стороны быстродействия

РОДЖЕР ОЛФОРД

Практически все серьезные пользователи компьютеров хотели бы работать с более быстрой системой. Более быстрый компьютер, более быстрый принтер, более быстрый модем... Пользователю нужно, чтобы все работало быстрее.

Эта одержимость скоростью не удивительна – взгляните на рекламу компьютеров: там прежде всего говорится о быстродействии и производительности. Даже обозреватели журналов заражены скоростной болезнью: при сравнении двух систем редко приводятся оценки, в которых отсутствует тест на скорость.

В этой статье я расскажу о возможностях повышения быстродействия. Вначале будут рассмотрены те блоки компьютера, которые влияют на его быстродействие. В следующих статьях я расскажу о том, как можно повысить производительность отдельных блоков, а также укажу подводные камни, на которые вы можете наткнуться, пытаясь работать слишком быстро. Вы узнаете, как можно стать настоящим и компетентным пользователем персонального компьютера.

МОЗГ КОМПЬЮТЕРА

Центральный процессор (ЦП) – мозг вашего компьютера. Он выполняет команды, записывает информацию в память и считывает ее оттуда, осуществляет доступ к внешним устройствам, таким как последовательный порт и контроллер диска. Поскольку вся обрабатываемая компьютером информация проходит через ЦП, он оказывает наибольшее влияние на производительность системы.

Производительность ЦП определяют четыре фактора: тактовая частота, пропускная способность шины данных, набор команд и эффективность микрокода. Хотя основной упор изготовители делают на тактовую частоту, не давайте вводить себя в заблуждение. Тактовая частота – важная характеристика, но и остальные факторы также существенно влияют на производительность ЦП.

Персональные компьютеры – это устройства последовательного действия. Поэтому для соблюдения очередности действий необходим синхронизирующий импульс – такт. Центральный процессор использует такт для последовательного выполнения шагов функций, включая доступ к ячейке памяти, доступ к периферийным устройствам и выполнение команд. За время каждого такта ЦП выполняет одну элементарную операцию. Для выполнения функции (например, доступа к ячейке памяти) требуется несколько тактов. Чем выше тактовая частота, тем быстрее ЦП выполняет элементарные операции и тем быстрее работает компьютер вообще.

Если число тактов для доступа к ячейке памяти остается постоянным при переходе от одной тактовой частоты к другой, то можно очень легко подсчитать изменение быстродействия. Предположим, что ЦП с тактовой частотой 8 МГЦ переключается на частоту 10 МГц (частота 8 МГц соответствует 8 млн тактов в 1 с, а 10 МГц – 10 млн тактов в 1 с). При работе на частоте 8 МГц ЦП на каждый такт требуется 125 нс, а при работе на частоте 10 МГц – только 100 нс. Скорость работы ЦП увеличится на 25% [(10–8)/8X100], а выполнение команд будет происходить за 80% [(100/125)X X100] требовавшегося раньше времени.

Увеличение тактовой частоты приводит к некоторым отрицательным эффектам. Центральные процессоры разрабатываются и испытываются для работы на максимально допустимых частотах. Когда ЦП превышает расчетную тактовую частоту, изготовитель ЦП уже не может гарантировать правильную его работу. Если испытания ЦП проводились тщательно и при строго выдержанном температурном режиме, то ЦП может работать безошибочно на слегка завышенной тактовой частоте. Но не стоит рисковать своими данными.

Другой эффект от увеличения тактовой частоты ЦП – сокращение времени обращения к памяти и внешним устройствам. Как и ЦП, память и периферийные устройства разработаны и испытаны в расчете на определенные времена доступа. Уменьшение времени доступа за счет увеличения тактовой частоты ЦП может привести к осложнениям.

ДРУГИЕ ХАРАКТЕРИСТИКИ ЦП

Ширина шины данных ЦП определяет количество бит двоичных разрядов информации, которые ЦП может принять или передать одновременно. Центральный процессор 8088 ПК IBM PC и XT имеет 8-битовую шину данных, а ЦП 80286 ПК AT имеет 16-битовую шину. В новых ЦП 80386, применяемых в ПК Compac Deskpro 386 и IBM Personal System 2 (PS/2) модели 80, используется 32-битовая шина.

Фирма Intel, разработчик этих ЦП, утверждает, что 16-битовый процессор 8086, использующийся в новом ПК модели 30 фирмы IBM, увеличивает производительность в среднем на 40% по сравнению с 8-битовым процессором 8088 при работе на одинаковой частоте. В остальном процессоры 8086 и 8088 идентичны. Фирмы Intel внесла некоторые изменения в ЦП 80286, улучшающие его работу. Хотя я сам этого и не наблюдал, но фирма Intel утверждает, что 32-битовая шина ЦП 80386 с точки зрения производительности значительно превосходит 16-битовую шину ЦП 80286.

Еще одним фактором, влияющим на производительность, является набор команд ЦП – функции, которые ЦП может выполнять вне оперативной памяти. Например, если в набор команд ЦП входит команда умножения, которая позволяет найти произведение двух 16-битовых чисел, скажем, за 20 тактов, то это дает программисту возможность не писать выполняющую умножение подпрограмму, которая работала бы в течение сотен тактов.

В наборе команд ЦП 80286 есть несколько команд, которых нет в наборе команд ЦП 8088; эти дополнительные команды предназначены, в частности, для повышения быстродействия ПК AT. Беда в том, что дополнительные команды позволяют ускорить работу только тогда, когда их используют. Большинство программ написано в рас-

чете на эксплуатацию на ПК PC, XT и AT, и поэтому в этих программах стараются не применять специфические команды ЦП 80286. От неиспользуемых команд примерно столько же пользы, сколько от несуществующих.

Другим фактором, влияющим на быстродействие ЦП, является эффективность микрокода. В большинстве ЦП, в том числе и во всех тех, о которых я рассказывал, для реализации набора команд используется микрокод. Выполнение каждой команды ЦП состоит в выполнении одной или нескольких микрокоманд. Вы можете заставить ЦП работать быстрее, изменив микрокод таким образом, чтобы для выполнения команд потребовалось меньшее число микрокоманд.

При разработке новой модели ЦП изготовители обычно оптимизируют микрокод, по крайней мере, для нескольких команд. Например, в ЦП 80286 повышена скорость выполнения микрокода для некоторых команд ЦП 8088.

Несколько лет назад фирма NEC Information Systems предложила свои ЦП серии V, совместимые по расположению контактов и по функциональным возможностям с ЦП Intel 8088 и 8086. Кроме того, фирма NEC расширила набор команд и усовершенствовала микрокод. В результате ЦП серии V гораздо производительнее, чем ЦП, совместимые с ЦП фирмы Intel. Последняя подала в суд на фирму NEC в связи с нарушением авторских прав на микрокод, но это дело судом еще не решено.

ЭТИ БЕСКОНЕЧНЫЕ СОСТОЯНИЯ ОЖИДАНИЯ

Хотя вы, наверное, уже слышали термин "состояние ожидания", большинство пользователей понастоящему не понимают, для чего нужны состояния ожидания и как они реализуются. Поскольку состояния ожидания могут очень сильно влиять на производительность системы, я объясню, что они собой представляют на примере состояний ожидания ЦП 8088.

Обычно ЦП 8088 требуется четыре такта для выполнения операций чтения из памяти или записи в нее. Если ЦП работает на стандартной частоте 4,77 МГц, каждый такт длится примерно 210 нс. Таким образом, цикл доступа к памяти составляет 840 нс. За это время надо выполнить операции проверки истинности записываемой или считываемой информации. Доступ к периферийным устройствам осуществляется аналогично.

Если цикл доступа к внешнему устройству или памяти превышает стандартное значение, то ЦП позволяет запросить одно или несколько состояний ожидания. Состояние ожидания – это дополнительный такт, добавляемый для получения доступа к периферийному устройству или памяти. Оно позволяет продлить цикл обращения настолько, насколько это необходимо для доступа. В ПК IBM PC автоматически добавляется одно состояние ожидания для доступа к любому периферийному устройству, так как обычно эти устройства имеют невысокие характеристики по времени доступа.

Состояния ожидания могут значительно снизить производительность системы. Добавление одного состояния ожидания к каждому циклу доступа к памяти ПК, т. е. использование пяти тактов для доступа к памяти вместо четырех, увеличивает время выполнения каждой операции чтения и записи на 25 %. Хотя ЦП не тратит все время на эти операции, все равно доля их значительна.

Вы должны знать о состояниях ожидания при приобретении ПК, многие из которых рекламируются как модели "турбо", работающие на двух скоростях – 4,77 и 8 МГц. Такие системы весьма привлекательны: 8 МГц – это на 68% больше, чем стандартная тактовая частота ПК. Но в рекламе не сказано, что во многих таких ПК (если не в большинстве) к каждому циклу обращения к памяти при работе в режиме "турбо" на частоте 8 МГц добавляется одно состояние ожидания. Таким образом, вместо ожидаемого 68%-ного ускорения покупатели таких систем получат, по всей вероятности, лишь 45 – 50%-ное улучшение. Хотя изготовители правы, когда добавляют одно состояние ожидания при работе на высокой частоте (о причинах этого будет рассказано позже), подобная реклама вводит в заблуждение.

ТИПЫ ПАМЯТИ

На производительность ПК влияет также тип памяти. Большинство ПК в настоящее время имеет по меньшей мере 256 Кбайт оперативной памяти (ОП), реализованной на нескольких микросхемах. Микросхемы памяти бывают двух типов – для статической и динамической памяти. В статической памяти хранение информации осуществляется с помощью триггеров. Информация в триггерах хранится до тех пор, пока на них не поступит новая информация либо пока не будет выключено питание.

В динамической памяти, напротив, используют встроенные конденсаторы для хранения двоичных данных. Данные хранятся в виде зарядов. Так как емкости этих конденсаторов малы, то хранимые в них заряды имеют тенденцию к быстрому исчезновению – обычно в течение нескольких миллисекунд. Поэтому, чтобы информация не была потеряна, нужно периодически ее восстанавливать. В системах с динамической ОП периодическое восстановление информации выполняется самим компьютером.

Как и следовало ожидать, у каждого типа памяти есть достоинства и недостатки. В микросхемах статической памяти требуется больше места для хранения информации. Следовательно, при заданной площади кристалла динамическая память более эффективна и экономична, что определяет меньшую стоимость хранения одного бита. Однако статическую память не надо восстанавливать. Пока компьютер включен, можно записывать информацию в статическую ОП и затем "забывать" о ее существовании до тех пор, пока она вновь не понадобится.

Для обеспечения низких цен в ПК IBM PC и практически во всех совместимых с ним ПК используются микросхемы динамической памяти. Для восстановления памяти в ПК IBM PC и XT периодически осуществляется цикл прямого доступа к памяти. Эта операция отвлекает ЦП от непосредственной деятельности и снижает производительность системы примерно на 8%. В ПК IBM AT и в совместимых с ним ПК есть специальные схемы, которые добавляют состояния ожидания при осуществлении доступа к памяти. Эти состояния ожидания обеспечивают дополнительное время, необходимое для восстановления памяти, но при этом снижается производительность системы.

Не поддавайтесь на рекламу ПК AT "без состояний ожидания": для ЦП 80286 такие утверждения, вооб-

ще говоря, нереальны. Хотя подсистема памяти и может быть спроектирована таким образом, что состояния ожидания для каждого обращения к памяти будут не нужны, но неизбежны конфликты при попытке ЦП получить доступ к памяти во время цикла восстановления. В этих случаях система вынуждена добавлять необходимые состояния ожидания для завершения восстановления памяти, прежде чем открыть доступ к ней для ЦП. Такой подход дает невысокий уровень потерь – порядка 5 – 10%, но потери все же есть.

По меньшей мере в одной совместимой с ПК AT системе – ПК марки 386 фирмы PC Limited – используется статическая, а не динамическая память. Использование статической памяти – единственный известный мне способ, действительно позволяющий работать без состояний ожидания.

Другой подход, обычно применяемый для больших ЭВМ, но начинающий уже проникать в мир микрокомпьютеров, – применение высокоскоростной кэш-памяти. В системах использующих этот метод, основной памятью является обычная динамическая ОП или в некоторых случаях более медленная статическая ОП, но в дополнение к основной ОП в этих системах есть высокоскоростная память небольшого объема, называемая кэш-памятью и работающая без состояний ожидания. По мере необходимости компьютер переносит программы или сегменты программ в кэш-память, где они выполняются без задержек.

ЗАДЕРЖКИ, ВНОСИМЫЕ ДИСКОВОДАМИ

Когда ПК обращается к гибкому диску, то сначала он должен включить двигатель дисковода и разогнать его до нужной скорости. Затем с помощью шагового двигателя головка чтения-записи перемещается к нужной дорожке. При подходе к требуемому месту до начала операции записи или чтения необходимо сделать паузу для позиционирования головки. Затем нужно дождаться момента, когда к головке переместится нужный сектор, после чего данные передаются с дисковода на ЦП со скоростью, определяемой характеристиками дисковода. Для жестких дисков используется аналогичная процедура доступа к информации.

Временем доступа называется среднее время, требующееся устройству для достижения нужной дорожки. Чем меньше время доступа, тем быстрее работает устройство. Для стандартных жестких дисков емкостью 10 и 20 Мбайт время доступа составляет примерно 65 мс. Высокоскоростные устройства обеспечивают время доступа 28 мс. Существуют еще более быстрые дисководы, но их цена очень высока.

Жесткие диски небольшой емкости обычно работают медленнее. Дисководы на некоторых переносных ПК, а также съемные дисководы имеют время доступа около 120 мс.

Способ обмена информацией между ЦП и дисководом сильно сказывается на производительности последнего. В большинстве систем ЦП записывает и считывает данные непосредственно с дискового контроллера. В некоторых высокопроизводительных системах, таких как новые ПК PS/2 фирмы IBM, используется произвольный доступ к памяти для передачи данных между ОП и контроллером. При этом ЦП не используется, что обеспечивает гораздо более высокую скорость обмена информацией, чем в случае участия ЦП в обработке данных.

ЗАВИСИМОСТЬ ОТ ПРИНТЕРА

На производительность ПК может влиять и принтер. Особенно важны три характеристики принтера: скорость печати, скорость продвижения бумаги и скорость передачи информации от ПК к принтеру.

Скорость печати – наиболее очевидный фактор. Чем быстрее принтер печатает символы, тем меньше времени вам надо ждать окончания печати. Подобной характеристикой является и скорость продвижения бумаги. Принтер тратит значительное время, продвигая бумагу и не печатая при этом ни одного символа (например, при переходе на начало нового листа). В связи с этим некоторые изготовители оборудования добавляют возможность быстрого продвижения бумаги, что позволяет принтерам быстро пропустить пустые места.

Третьим фактором, определяющим быстродействие принтера, является скорость передачи символов. Если символы передаются на принтер последовательно, то даже при максимально возможной для ПК скорости передачи 9600 бод они будут переданы значительно медленнее, чем через более быстрый параллельный интерфейс.

ВИДЕОИНТЕРФЕЙС

Последний фактор, влияющий на производительность ПК, – это видеоинтерфейс. В отличие от доступа к ОП, к видеопамяти возможен двойной доступ: ЦП и блоков видеоинтерфейса. Так как два устройства должны одновременно обращаться к одной и той же области памяти, в состав видеоадаптера входит специальный блок, предотвращающий конфликты при одновременном обращении устройств к памяти.

Видеоблоки в период кадровой развертки (во время вывода изображения на экран) должны иметь постоянный доступ к памяти. Центральный процессор может получить доступ к этой памяти на время обратного хода кадровой развертки – промежутка времени, в течение которого луч электронной пушки монитора переводится в левый верхний угол экрана.

Тот факт, что видеоблоки большую часть времени заняты кадровой разверткой, серьезно ограничивает возможность ЦП записывать информацию в видеопамять. Адаптер CGA (Color Graphics Adapter) фирмы IBM хорошо иллюстрирует эту проблему. При выполнении прокрутки видеоблоки отключаются, что позволяет ЦП осуществлять доступ к памяти. Когда ЦП заканчивает прокрутку строки, видеоблоки подключаются вновь. В результате на экране возникает раздражающее мерцание – типичный признак работы адаптера CGA.

Конструкция адаптера EGA (Enhanced Graphics Adapter) обеспечивает более широкий доступ ЦП к видеопамяти. Об этом свидетельствуют более высокая скорость работы адаптера EGA и отсутствие мерцания при прокрутке изображения на экране. Новый видеоадаптер VGA фирмы IBM, устанавливаемый на ПК PS/2 этой фирмы, имеет меньшее время доступа к видеопамяти, чем адаптер EGA.

Теперь, когда вы знакомы с теми блоками вашего ПК, которые влияют на его производительность, можно начать поиск увеличения быстродействия. В следующих статьях я дам несколько советов и рассмотрю опасности чрезмерного увлечения повышением быстродействия.

Повышение скорости выполнения программ на Бейсике

ГРЕГ ПЕРРИ

Язык программирования Бейсик достаточно прост. Вы можете повысить скорость выполнения программ на этом наиболее популярном языке, если будете использовать некоторые очевидные приемы, которыми зачастую пренебрегают.

Вы вряд ли вызовете возражения, заявляя, что Бейсик стал самым массовым языком программирования. Но мало кто согласится с утверждением, что такой успех Бейсика объясняется исключительно его высокими качествами и не зависит от программно-аппаратной реализации.

Но и в этом есть свой резон. Бейсик представляет собой удивительно удачную попытку предоставить многим пользователям многие возможности. Этот язык содержит в себе такие изощренные средства, которые отсутствуют и в некоторых "полномерных" языках. Хотя Бейсик неэлегантен и не имеет четкой структуры, тем не менее он легок в использовании. Основу его составляют простые глаголы английского языка, такие как PRINT и INPUT, однако он позволяет описывать и сложные операции, включая создание и корректировку последовательных файлов и файлов прямого доступа.

Скорее всего вы, как и большинство программирующих на Бейсике, стремитесь написать программу как можно быстрее и при этом пренебрегаете скоростью ее выполнения. Но и ветеранам Бейсика, и начинающим программистам могут пригодиться методы ускорения выполнения программ на Бейсике.

Конечно, компьютеры не понимают ни Бейсика, ни других языков программирования. Словарь компьютера ограничен машинными командами, составленными из нулей и единиц. Но поскольку люди сами разрабатывают машины и устанавливают правила общения с ними, то нет никакой необходимости сводить команды к битам и байтам. Поэтому языки высокого уровня используют команды, которые любой смертный может понять и сохранить в виде текстовых файлов. Одной команде языка высокого уровня соответствует целый набор элементарных машинных команд. Интерпретаторы и компиляторы играют роль переводчиков наших команд в понятные компьютеру указания.

КАК РАБОТАЕТ ИНТЕРПРЕТАТОР

Интерпретируемый Бейсик – это язык, позволяющий получать результаты очень быстро. Команды выполняются без компиляции и компоновки, а программа, коль скоро она загружена, начинает выполняться сразу. Это происходит потому, что интерпретируемая программа на Бейсике транслируется в процессе выполнения. Интерпретатор читает строку программы, преобразует ее в машинные команды и выполняет их. Затем то же самое интерпретатор проделывает со следующей (логически) строкой программы, даже если он эту строку уже обрабатывал (например, в теле цикла). В отличие от этого, компилятор читает и преобразует программу на Бейсике целиком и создает на диске новый файл, называемый файлом объектного кода. Этот файл затем объединяется с другими компилированными программными модулями специальной программой – компоновщиком. Даже если других компилированных модулей вообще нет, все равно необходимо выполнить компоновку для создания третьего файла на диске – выполнимого файла (типа .EXE).

Время, затрачиваемое на компиляцию, компенсируется более быстрым выполнением: интерпретируемая программа, в отличие от компилируемой, сразу готова к исполнению, и ее можно легко и быстро отредактировать, но выполняется она медленнее.

ЗАМЕДЛЯЮЩИЕ КОММЕНТАРИИ

В большинстве реализаций Бейсика (включая и Бейсик для персональных компьютеров) используются интерпретаторы, а не компиляторы. Хотя интерпретируемая программа на Бейсике может быть откомпилирована, компилятором в основном пользуются только профессиональные программисты. Начинающие программисты предпочитают интерпретатор, поскольку интерпретируемые программы легко исправить и запустить заново. Такая интерактивная среда выполнения удобна для экспериментов. За это приходится платить скоростью выполнения: компилированная программа на несколько порядков быстрее, чем интерпретируемая.

Если программа на рис. 1 будет компилироваться, то вся она (кроме комментариев REM в строках 5 и 15) преобразуется в объектный код и сохраняется на диске. (Компилятор просматривает строки с комментариями только один раз и в дальнейшем их не компилирует и не использует.) Затем объектный код компонуется в файл типа .EXE.

В отличие от этого, интерпретатор, обрабатывая программу, транслирует и выполняет каждую строку индивидуально. Даже не нужные для выполнения строки с комментариями обрабатываются (распознаются) каждый раз, когда выполнение программы проходит через эти строки. Программа на рис. 2 требует 500 интерпретаций комментариев, расположенных внутри цикла. Поэтому не приходится удивляться, что интерпретатор Бейсика работает медленно. Но если вы познали достоинства и недостатки интерпретаторов, то можно использовать это знание для оптимиза-

ции интерпретируемых Бейсик-программ согласно следующим рекомендациям.

Удаляйте из программы комментарии.

Описывайте все переменные до их использования операторами DEF.

Имена переменных делайте короткими.

Используйте переменные вместо числовых или строковых констант.

Операторы DATA размещайте в начале программы.

Часто используемые подпрограммы помещайте в начале программы.

Помещайте в одну строку программы как можно больше операторов.

Удаляйте все избыточные пробелы.

Используйте краткую форму записи операторов.

Очевидно, исключение всех комментариев из программы ускорит ее выполнение. Однако хорошее документирование тоже необходимо. Поэтому вместо полного отказа от комментариев вы можете удалить их только из конечной версии программы. Документация должна разрабатываться после отладки программы, а не заранее, и снабженный продуманными комментариями вариант программы будет служить хорошим описанием. Короче говоря, при разработке программ на Бейсике используйте комментарии свободно, а когда программа будет закончена и отлажена, скопируйте ее (с другим именем) и удалите все комментарии.

О ПОЛЬЗЕ ОПИСАНИЙ

Поиски путей повышения эффективности приводят к требованию явного описания с помощью оператора DEF всех используемых переменных как целых (одинарной или двойной точности) или как строковых переменных в начале программы. Для подпрограммы, обрабатывающей целые числа, отсутствие такого описания приводит к определению числовой переменной по умолчанию как целой переменной одинарной точности. Например, каждое выполнение цикла FOR . . . . . NEXT при этом приводит к тому, что под индексную переменную (такую, как I в цикле FOR I = 1 TO 10) интерпретатором отводится в памяти место, необходимое для шести десятичных цифр, и в это место пересылается начальное значение. И на все это тратится процессорное время.

```
5 REM *** Program to print the
6 REM *** numbers 1 through 10
10 I=1
15 REM --- Print the number
20 PRINT I
30 IF I>=10 THEN 60
40 I=I+1
50 GOTO 15
60 END
```

Рис. 1. Компилятор Бейсика читает и транслирует программу целиком, операторы REM обрабатываются один раз. Интерпретатор же читает и транслирует программу построчно, поэтому операторы внутри цикла, такие как в строке 15, обрабатываются многократно

```
5 REM *** Interpreting REMarks slows
6 REM ***  down BASIC programs
10 DEFINT I
20 FOR I=1 TO 100
25 REM --- This REMark takes up time
26 REM --- So does this one
27 REM arkable, isn't it?
30 PRINT I
35 REM --- We are almost done
36 REM --- with the REMarks
40 NEXT I
50 END
```

Рис. 2. Интерпретация комментариев в этой программе замедляет печать чисел

Некоторые программисты предпочитают описывать тип переменных при их использовании явно вместо того, чтобы полагаться на глобальное описание оператором DEF. Вы можете, например, объявить переменную I целой, поставив после нее символ % при каждом появлении этой переменной в программе. Хотя такое описание и оставляет точность переменной одинарной, но каждая переменная I% требует затрат времени на интерпретацию. Выполнение программы ускорится, если описывать переменную I оператором DEFINT I.

Даже самые изощренные программисты чаще всего описывают строковые переменные как H$ вместо того, чтобы объявлять переменную H оператором DEFSTR H. Как и описание целых переменных в виде I%, описание H$ собирает свою дань в форме замедления времени выполнения.

Кроме того, хотя Бейсик для IBM PC допускает имена переменных длиной до 40 символов, более короткие имена приводят к более эффективной интерпретации. Обратной стороной краткости, очевидно, является снижение информативности программ.

Каждый раз, когда в вашей программе встречается числовая константа, интерпретатору добавляется работа. Он анализирует ASCII-представление каждой цифры, знака и десятичной точки, преобразует константы в специальный внутренний формат Бейсика. Каждый раз, когда интерпретатор встречает численную константу, все начинается сначала.

Вы можете обойтись без этой карусели, если присвоите значение константы некоторой переменной и в дальнейшем будете ее использовать. Коль скоро число преобразовано и размещено в памяти, интерпретатору достаточно заглянуть в соответствующую таблицу – и значение тут как тут. Примером служат программы на рис. 3; вторая

программа демонстрирует, что число 3.14159 лучше представить в виде переменной PI.

Такое же ускорение можно получить, если убрать из текста программы строковые константы. Присвойте их один раз переменным – программа станет короче и будет быстрее интерпретироваться.

ПОРЯДОК ОПЕРАТОРОВ

Размещение операторов в программе самым тесным образом связано с эффективностью ее выполнения. Каждый раз, когда в программу необходимо ввести операторы READ или DATA, размещайте их как можно ближе к началу программы. При каждом выполнении оператора READ интерпретатор просматривает программу с самого начала в поисках следующего неиспользованного оператора DATA, даже если искомый оператор DATA стоит сразу за оператором READ.

Аналогичным образом, выполняя оператор GOSUB, интерпретатор Бейсика переходит к началу программы и просматривает строку за строкой в поисках нужного номера. Интерпретатор Бейсика никак не может выйти на нужную строку сразу. Следовательно, часто используемые подпрограммы должны размещаться раньше остальных. Таким образом вы практически без потерь уменьшите работу интерпретатора при выполнении перехода на подпрограмму. Хотя этот прием и очевиден, о нем часто забывают.

Форма представления операторов Бейсика также влияет на скорость выполнения. Вы можете объединить несколько операторов Бейсика в одной строке программы (которая вмещает до 255 символов), разделяя их двоеточием. На рис. 4, например, первая программа состоит из семи строк; та же программа (вторая) умещается в двух строках. Второй вариант выполняется так же, как и первый, и дает те же результаты, но делает это быстрее – просто потому, что операторы объединены. Интерпретатору не приходится тратить время на распознавание номера строки для каждого оператора.

Конечно, программу из двух строк несколько труднее понять, чем исходную (поэтому такое уплотнение текста не должно быть самоцелью), но исходную программу из семи строк также можно оставить для описания. Обратите внимание и на то, что из второго варианта программы удалены все избыточные пробелы.

Естественно, все пробелы удалить нельзя. Пробелы (или другие разделители) необходимы для отделения друг от друга команд и имен переменных. Например, интерпретатор не распознает в наборе букв PRINTI оператор PRINT I. Нельзя также удалять пробелы из строковых констант. Однако оператор присваивания можно записать как A=B+C, а можно как A = B + C. Удаление избыточных пробелов может показаться тривиальным, но интерпретатор обрабатывает пробелы так же, как и все остальные символы. Поэтому чем меньше пробелов, тем лучше.

```
5 REM *** Interpreting numeric constants
6 REM ***  slows down BASIC programs
10 DEFINT I
20 FOR I=1 TO 100
30 PRINT 3.14159*I
40 NEXT I
50 END

5 REM *** Using a variable in place of a
6 REM ***  constant speeds up BASIC programs
10 DEFINT I
20 PI=3.14159
30 FOR I=1 TO 100
40 PRINT PI*I
50 NEXT I
60 END
```

Рис. 3. Интерпретация числовых констант, таких как 3.14159 в строке 30 первой программы, замедляет ее выполнение. Замена числовой константы переменной PI в строке 40 второй программы ускоряет ее выполнение

```
5 REM *** This program is big and thus slow
10 CLS:PRINT "Hello."
20 PRINT
30 PRINT "Who are you";
40 INPUT N$
50 PRINT "Well, "N$", I'm pleased to meet you."
60 END

5 REM *** This program is more compact and thus faster
10 CLS:PRINT "Hello.":PRINT:PRINT "Who are you";:INPUT N$:PRINT "Well, "N$", I'm
 pleased to meet you.":END
```

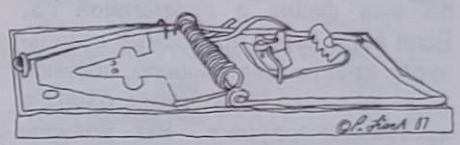
Рис. 4. В первой программе для каждого оператора отведена отдельная строка; программа понятнее, но выполняется медленее. Во второй программе несколько операторов, разделенных двоеточиями, объединены в одну строку, которая вмещает до 255 символов. Такая программа выполняется быстрее

ЧЕМ КОРОЧЕ, ТЕМ ЛУЧШЕ

Программисты, работающие с Бейсиком, обычно избегают использования избыточных слов и команд. Большинство опускают команду LET в операторе присваивания, поскольку скорее всего забыли о существовании этого ключевого слова. Нечего и говорить, что LET A=B занимает больше места и выполняется медленнее, чем оператор A=B. Но существуют и другие операторы Бейсика, которые можно сокращать. Оператор OPEN O, 1, имя_файла в два раза короче оператора OPEN имя_файла FOR OUTPUT AS #1 LEN=128, хотя делает то же самое.

Перечисленные выше приемы — это только первые шаги в направлении экономного и эффективного программирования. Программы, написанные по этим правилам, будут выглядеть не очень элегантно, и понять их будет трудновато. Но единственная трудность, возникающая при этом, заключается в необходимости иметь две версии программы — одну рабочую и одну хорошо документированную. И да будет вам программирование в радость.

Ставьте в программах ловушки ошибок



Сохраните эту удобную программу, она предохранит ваши программы на Бейсике от аварийного завершения в самый неподходящий момент

ДЕЙВИД ЛИТХОЗЕР

Все хорошие программы имеют ловушки ошибок, которые предохраняют их от аварийного завершения и возвращения к интерпретатору Бейсика каждый раз, когда возникает ошибка. Оператор On Error Goto (перейти по ошибке) в начале программы отсылает компьютер в раздел программного кода, который идентифицирует ошибку и предпринимает соответствующие действия.

К сожалению, написание программного кода ловушки ошибок может быть сложным и долгим процессом, так как необходимо обдумать все возможные неполадки и решить, как с ними поступать. Для экономии времени я написал программу-ловушку ошибок, которая фиксирует 99 % внешних ошибок при выполнении программы, — ошибок, связанных с факторами, внешними по отношению к программе, такими как переполнение диска или неготовность принтера. Вы можете добавить ее к любой программе на Бейсике с небольшими изменениями или без них.

ЧЕЛОВЕКУ СВОЙСТВЕННО ОШИБАТЬСЯ

Аварийное завершение программы может вызываться ошибками трех типов. Моя ловушка ошибок фиксирует те из них, которые могут оказаться наиболее разрушительными. Она не защищает от ошибок при программировании, таких как использование оператора Goto, отсылающего к несуществующей строке, несогласованные операторы For и Next и синтаксические ошибки. Она также игнорирует внутренние ошибки, возникающие при выполнении программы, которые обычно связаны с неправильным вводом данных пользователем (типичные примеры — деление на нуль, выход индекса за пределы массива). Вы можете избежать таких ошибок с помощью операторов, строго определяющих допустимые пределы вводимых данных. В ловушках ошибок такого типа нет необходимости.

Внешние ошибки при выполнении программ являются наиболее неприятными, так как обычно они возникают после того, как вы затратили время на ввод данных или выполнение программы. Все ваши усилия пропадут при аварийном завершении программы. Представьте, как неприятно, например, играя в приключенческую игру, потерять ее только из-за того, что диск переполнился, когда вы пытались выполнить команду сохранения Save.

Чтобы использовать программу-ловушку ошибок, наберите ее и запишите как файл в коде ASCII с помощью следующей команды:

```
SAVE "ERRTRAP",A
```

Написав какую-то свою программу, вы можете подсоединить к ней программу-ловушку ошибок с помощью команды

```
MERGE "ERRTRAP"
```

Для работы программы-ловушки ошибок ERRTRAP.BAS требуется интерпретатор GW-Basic или A-Basic.

Программа-ловушка ошибок предохраняет программу, написанную на Бейсике, от аварийных завершений, вызванных такими внешними причинами, как "Устройство не готово", "Файл не найден", "Диск с дефектом", "Переполнение диска", "Плохое имя файла", "Диск защищен от записи", "Диск не готов".

КАК РАБОТАЕТ ЛОВУШКА

Чтобы познакомиться с различными внешними ошибками при выполнении программы и знать, чего ожидать, когда программа столкнется с ними, вы должны иметь представление об операторах, входящих в программу-ловушку. Я кратко опишу, как работает каждый из них.

Строка 1. Активизация ловушки переходом на строку 65000, с которой начинаются операторы ловушки. Так как обычно в большинстве программ строка 1 или строки с номерами больше 65000 отсутствуют, то вы можете добавить эти операторы без перенумерования. Однако если ваша программа содержит оператор очистки Clear, то вы должны перенумеровать строку с оператором On Error Goto так, чтобы она стояла после оператора Clear, так как последний прекращает действие ловушки ошибок.

Строка 65000. Фиксация ошибки, возникающей при попытке использовать периферийное устройство, которое не готово. Может быть не включен принтер, или потерян сигнал при связи через модем, или компьютер на другом конце канала связи может не отзываться на вызов. Оператор строки 65000 печатает сообщение "Устройство не готово" и подсказывает пользователю, что надо включить устройство и нажать клавишу "Ввод", чтобы возобновить выполнение программы с того места, где произошел выход из программы. Ошибка 24 (таймаут устройства) означает, что интерпретатор Бейсика не получил сигнала от устройства ввода-вывода в течение определенного интервала времени. Ошибка 25 (неисправность устройства) означает, что что-то не в порядке с принтером или интерфейсом. Ошибка 27 (отсутствие бумаги) означает буквально отсутствие бумаги, но также может указывать и на другие неполадки в принтере.

Строка 65002. Фиксация ошибок, которые возникают, когда компьютер пытается загрузить несуществующий файл. Печатается сообщение: "Файл не существует. Новое имя файла?". Вы можете ввести новое имя файла, и программа попытается вновь загрузить файл.

В строке 65002 предполагается, что первоначальная программа содержит запрошенное у пользователя имя файла в переменной F$. Если вы в своей программе используете другую переменную, поставьте ее на место F$ в строке 65002 в программе-ловушке. Вы должны для имен файлов по всей программе использовать одну и ту же переменную.

Можно также предоставить пользователю возможность увидеть список существующих на диске файлов перед запросом о новом имени файла. Замените оператор Input в строке 65002 на следующие команды:

```
PRINT "Файл не найден. Файл на диске:":
FILES:INPUT "Новое имя файла"; F$
```

Добавление этих операторов имеет некоторые недостатки. Если на диске нет никаких файлов, то программа столкнется с ошибкой "Файл не найден" при выполнении программы-ловушки и завершится аварийно. Кроме того, команда Files покажет на экране все файлы, существующие на диске, а не только файлы с данными. Все расширения, добавляемые вашей программой к имени файла, тоже будут воспроизведены на экране, что может смутить некоторых пользователей. Вывод списка файлов может в какой-то степени запутать картину на экране, однако при вводе нового имени файла удобно видеть список существующих файлов и это удобство окупает недостатки.

Строка 65004. Фиксация ошибки "Диск с дефектом", которая обычно встречается при попытке компьютера прочитать информацию или сохранить ее на неотформатированном диске или на диске с плохим каталогом. Пользователь может заменить диск и нажать клавишу "Ввод" для повторного выполнения операции чтения или записи.

Строка 65006. Печатается сообщение "Переполнение диска" при попытке пользователя сохранить информацию на диске, когда он полностью заполнен. Это позволяет получить время для замены диска и нажатия клавиш "Ввод" для продолжения операции. Обычно исходная программа не повторяет команды сохранения Save, поэтому пользователь должен ввести ее заново. (В некоторых ситуациях интерпретатор Бейсика не может справиться с этой ошибкой и программа завершается аварийно. Однако это не мешает оставить строку 65006, так как она все-таки увеличивает степень защиты от ошибок.)

Строка 65008. Фиксация ошибок, которые встречаются при попытке открыть файл с неправильным именем. Обычно программа запрашивает у пользователя информацию об имени файла, а пользователь может ввести запрещенный символ, такой как звездочка. В подобном случае пользователю предлагается ввести новое имя файла, после чего делается новая попытка открыть файл.

Как и в строке 65002, в строке 65008 предполагается, что имя файла запоминается в переменной F$. Если в вашей программе используется другая переменная, вы должны заменить F$ в этой строке на новую переменную. Так как ошибка 64 обычно возникает при попытке сохранить новый файл, а не при попытке загрузить старый, то нет необходимости включать команду Files в строку 65008.

Строка 65010. Фиксация ошибок, возникающих при попытке сохранить информацию на защищенном от записи диске. Это дает пользователю шанс заменить диск и нажать клавишу "Ввод" для повторного исполнения операции.

Наберите текст программы ERRTRAP.BAS и подсоедините ее к любой программе на языке Бейсик, чтобы избежать аварийных завершений из-за внешних ошибок

```
1 ON ERROR GOTO 65000
65000 IF ERR=24 OR ERR=25 OR ERR=27 THEN INPUT "Device not ready. Fix an
      d press ENTER.",DUMMY$:RESUME
65002 IF ERR=53 THEN INPUT "File not found. New file name";F$:RESUME
65004 IF ERR=57 OR ERR=72 THEN INPUT "Bad disk. Replace and press ENTER.
      ",DUMMY$:RESUME
65006 IF ERR=61 OR ERR=67 THEN INPUT "Disk full. Replace and press ENTER
      .",DUMMY$:RESUME
65008 IF ERR=64 THEN INPUT "Bad file name. New file name";F$:RESUME
65010 IF ERR=70 THEN INPUT "Disk write protected. Change and press ENTER
      .",DUMMY$:RESUME
65012 IF ERR=71 THEN INPUT "Disk not ready. Fix and press ENTER.",DUMMY$
      :RESUME
65014 ON ERROR GOTO 0
```

Конец

Строка 65012. Фиксация ошибки "Диск не готов" при попытке пользователя открыть файл для ввода или вывода информации. Может оказаться не закрытой крышка дисковода или в устройстве может отсутствовать диск. Строка 65012, как и другие, позволяет пользователю исправить ситуацию и затем нажать клавишу "Ввод" для повторного выполнения первоначальной операции.

Программа ERRTRAP.BAS обрабатывает все ошибки, которые могут встретиться в программе, которую вы уже отладили, устранив ошибки программирования и внутренние ошибки, возникающие во время исполнения. Если же все-таки встречается ошибка последнего типа, то оператор строки 65014 отменяет выполнение программы-ловушки и заканчивает выполнение программы с выдачей сообщения об ошибке.

ЗАКЛЮЧИТЕЛЬНЫЕ СООБРАЖЕНИЯ

Если вам захочется выводить сообщения об ошибках в определенном месте экрана, вставьте команду позиционирования Locate перед каждым оператором Input, распечатывающим сообщение об ошибке.

Вы также можете поместить курсор в начало сообщения об ошибке и напечатать строку пробелов, чтобы удалить сообщение с экрана после нажатия клавиши "Ввод". Чтобы сделать это, вставьте операторы Locate и Print Tab непосредственно перед оператором Resume в каждой строке ловушки для ошибок.

Чтобы сделать оба указанных выше изменения, замените строку 65012 на следующую:

```
65012 IF ERR = 71 THEN LOCATE
25,1:INPUT "Диск не готов. Установите диск и нажмите клавишу
"Ввод".",DUMMY$:LOCATE
25,1:PRINT TAB(39):RESUME
```

Вы можете повысить надежность работы ловушки при случайно возникающем переполнении диска, модифицировав оператор Resume и добавив оператор Close. Например, после изменения строки 650006 на следующую:

```
65006 IF ERR = 61 OR ERR = 67
THEN INPUT "Переполнение диска.
Замените диск и нажмите клавишу
"Ввод".",DUMMY$:CLOSE:RESUME
150
```

при переполнении диска будет происходить возврат к строке 150 исходной программы, содержащей оператор Open, при выполнении которого была сделана первоначальная попытка открыть файл. Помните, что такие изменения делают программу ERRTRAP.BAS менее транспортабельной.

И последнее. В некоторых программах требуются дополнительные ловушки ошибок. Например, программы, в которых используется связь через модемы, должны осуществлять контроль ошибок четности и других ошибок, присущих связи по телефонным каналам. Создание обобщенной ловушки ошибок – задача слишком сложная. Программа же ERRTRAP.BAS даже при отмеченных ограничениях поможет повысить надежность ваших программ ценой лишь минимальных дополнительных хлопот.

Quick Basic и Turbo Basic

Пример, приведенный в июньском номере журнала PC World, показывает, что время, затраченное при использовании компиляторов Quick Basic 4.0 и Turbo Basic 1.1, соответственно равно: на загрузку файла 10.4 и 1.0, на компиляцию 3.0 и 20.1, на выполнение программы 21.6 и 18.9 с; память, требуемая для выполнения программы, – 115 293 и 92 677 байтов. Для этого же примера компилятор Quick Basic 3.0 обеспечил размер программы 69 865 байтов, а скорость выполнения выше (хотя и не намного), чем компилятор версии 4.0.

Если вы задали время, мы запишем ваш файл

```
1 GOTO 5
2 T$=TIME$:HOUR$=MID$(T$,1,2):MINUTE$=MID$(T$,4,2)
3 FILENAME$="PROG"+HOUR$+MINUTE$
4 SAVE FILENAME$
5 REM
6 REM Start program here
```

Программа, опубликованная в июльском номере журнала "PC World" за 1987 г., натолкнула меня на мысль написать аналогичную программу, которая мне очень пригодилась.

Предложенный Д.Кингсли способ сохранения программ на Бейсике позволяет сохранить на диске только последнюю версию программы. Для формирования уникального имени файла я использовал время дня, поэтому при каждой записи программы на диск создается новый файл.

Введите текст программы PROG.BAS (см. распечатку) и запишите его в файл в коде ASCII. Вставляйте эту программу во все разрабатываемые вами программы, заменяя обозначение PROG в третьей строке 4-символьным именем соответствующей программы. Если потом запускать программы как обычно – задавая команду RUN, то оператор GOTO в строке 1 позволит обойти добавленные строки. Когда вам понадобится записать вариант программы перед ее проверкой, введите команду RUN2, и выполнение начнется со строки 2. В строке 2 значение текущего времени присваивается переменной T$, а текущие значения часов и минут – переменным HOUR$ и MINUTE$; в сроке 3 обе эти переменные включаются в формируемое имя файла FILENAME$ (например, запись программы в 2 час. 53 мин. дня приведет к созданию файла PROG1453.BAS). В строке 4 производится запись на диск текущей версии программы (если хотите записать версию в коде ASCII, то используйте команду SAVE FILENAME$,A). Записанные ранее версии остаются в неприкосновенности.

Д. Уинфри

Переназначение вывода

Вопрос. Я программирую на языке Бейсик и почти всегда вывожу выдаваемые программой данные либо на экран с помощью оператора PRINT, либо на печатающее устройство с помощью оператора LPRINT. Однако иногда мне бы хотелось направлять эти данные прямо в файл. Возможно ли это? Если да, то могу ли я выводить данные одновременно и в файл, и на экран?

Ответ. Задав при загрузке интерпретатора Бейсика имя выходного файла, вы можете так переназначить вывод, что выдаваемые программой на Бейсике данные будут поступать и в файл на диске, и на экран. Например, приведенная ниже команда дает указание DOS загрузить интерпретатор Бейсика, а затем загрузить и выполнить программу MYPROG.BAS. Символ переназначения (>) заставит интерпретатор Бейсика после загрузки направлять выводимые на экран данные в файл MYFILE.ASC:

```
BASIC MYPROG>MYFILE.ASC
```

Задав два символа переназначения, вы можете указать, что выводимые данные следует добавлять к уже существующему файлу:

```
BASIC MYPROG>>MY-
FILE.ASC
```

Как и в предыдущем примере выводимые данные будут дублироваться на экране.

Подавление ошибок

Подпрограмма, приведенная на распечатке, во время выполнения программы, создающей различные файлы данных, проверяет, сколько свободного пространства осталось на диске. Она позволяет продолжить выполнение программы в случае возникновения ошибок, сопровождаемых такими сообщениями: Disk full (Диск заполнен), Disk not ready (Диск не готов), Disk write protect (Диск защищен от записи). Эта подпрограмма с помощью оператора LOCATE, обеспечивающего соответствующую обработку сообщений об ошибках, помогает избежать потери ценных данных и сохранить изображение на экране в неприкосновенности. В строке 50 анализируется запись с наибольшим номером, и до попытки записать файл определяется, достаточно ли на диске свободного места для записи всего файла.

Дж. Володзко

```
10 ON ERROR GOTO 200
20 OPEN "R",#1,"DUMMY.DAT",20
30 FIELD #1, 20 AS A$
40 LSET A$=MK$(0)
50 PUT#1,100
60 CLOSE #1
70 PRINT"Test okay. Space available for new file."
80 KILL"DUMMY.DAT"
90 GOTO 300    ' branch to main code
100 END
190 ' Error handling routine
200 NUM=ERR
210 ON ERROR GOTO 0
220 IF NUM=61 THEN PRINT"No space, delete unused files.":
    RESUME 80
230 IF NUM=71 THEN PRINT"Drive not ready.":RESUME 80
240 IF NUM=70 THEN PRINT"Remove write-protect tab.":RESUM
    E 80
250 ERROR NUM    ' let Basic handle anything else
300              ' start main code here...
```

Конец

Обращение к *DOS* из программы на Бейсике

Вопрос. Как я могу выполнить команду DOS из программы на Бейсике?

Ответ. Команда SHELL языка Бейсик позволяет вам временно приостановить работу программы на Бейсике для выполнения команд DOS. Вы можете запускать файлы с расширениями .COM, .EXE, .BAT, выполнять команды DOS (такие, как DIR, TYPE и COPY), а когда закончите эти действия, можете вернуться в программу на Бейсике.

В том месте программы на Бейсике, где вы хотите обратиться к средствам DOS, добавьте, например, такую команду:

```
100 SHELL
```

После выполнения функций DOS для возврата в программу на Бейсике введите

```
EXIT
```

Вы можете создать пакетный файл для автоматического выполнения описанной процедуры. Например, приведенный ниже пакетный файл копирует все файлы с расширением .BAS из корневого каталога на дисководе A в подкаталог на дисководе C, а затем возвращает управление программе на Бейсике. Для создания этого пакетного файла в ответ на запрос ввода команды DOS введите

```
COPY CON: COPYFILE.BAT

COPY A:\*.BAS C:\BASIC
```

Закончив ввод, нажмите клавишу F6, а затем — клавишу Enter. Для обращения к этому пакетному файлу из программы на Бейсике вставьте в нее, например, такую строку:

```
100 SHELL "COPYFILE.BAT"
```

Знакомство с аппаратными средствами

Основные принципы работы модема

РОДЖЕР ОЛФОРД

Модемы настолько широко распространились среди пользователей персональных компьютеров, что, как и сами компьютеры, стали чем-то само собой разумеющимся. Так же, как компьютеры, модемы пережили запутанный и полный событий эволюционный процесс, включающий в себя разработку множества стандартов и внесение многочисленных технологических усовершенствований. Современные модемы, несмотря на их сравнительно низкую стоимость, обеспечивают такой уровень пропускной способности, надежности, интеграции, стандартизации и скорости передачи, который был недоступен для более дорогостоящих модемов всего несколько лет назад. Сегодня я освещу некоторые вопросы, связанные с терминологией, технологией, стандартизацией и основными принципами работы модемов.

ЧТО ТАКОЕ МОДЕМ?

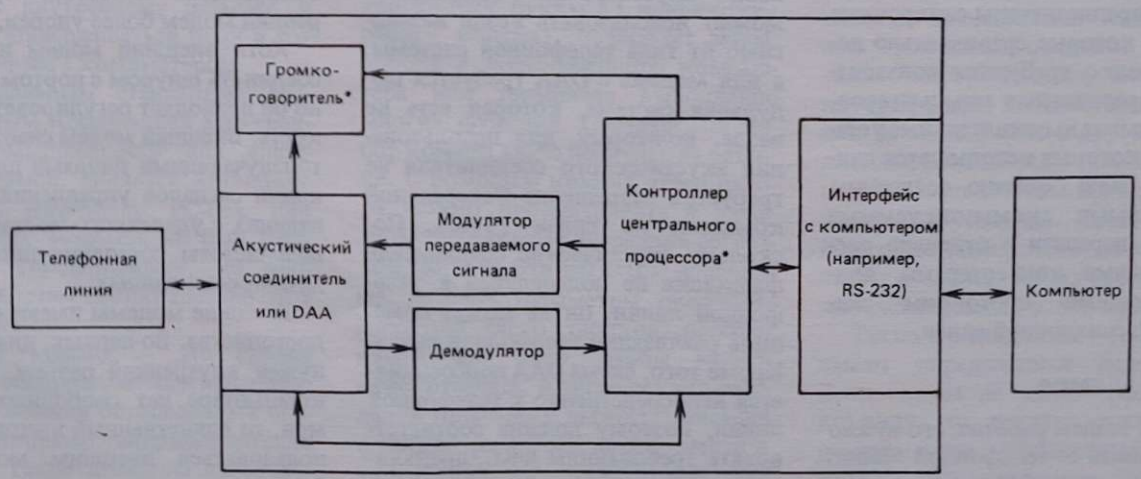
Слово *модем* является сокращением, образованным от слов *модулятор/демодулятор*. Модем – это периферийное устройство, преобразующее цифровые сигналы, поступающие с компьютера, в частотно-модулированные сигналы, которые гораздо проще передавать на большие расстояния – чаще всего по телефонным сетям. Кроме того, модем осуществляет прием модулированных сигналов, сгенерированных другим модемом, и преобразование (демодуляцию) их в цифровые сигналы, которые могут распознаваться и обрабатываться компьютером.

Связи внутри компьютеров реализуются на основе представления различных цифровых значений разными уровнями потенциала. Поскольку телефонные сети не рассчитаны на реализацию связи с помощью сигналов с разными уровнями потенциала, модем преобразует уровни потенциала в звуковые частоты, которые могут обрабатываться телефонными системами. Аналогично звуковые частоты он преобразует в уровни потенциала, которые могут обрабатываться принимающим компьютером (см. рисунок).

Модем обычно работает в одном из трех режимов: дуплексном, полудуплексном и эхоплексном. В дуплексном режиме соединенные между собой модемы могут одновременно передавать информацию друг другу, поэтому суммарная пропускная способность в два раза больше скорости передачи одного модема. Все наиболее распространенные модемы, осуществляющие передачу со скоростью 300, 1200 или 2400 бод, работают в дуплексном режиме.

В полудуплексном режиме в каждый момент только один из модемов может осуществлять передачу, другой модем должен дожидаться своей очереди. Некоторые полудуплексные модемы, например модем Bell 202 со скоростью передачи 1200 бод, поддерживают и низкоскоростной обратный канал, который позволяет принимающему модему передавать небольшой объем информации (обычно это информа-

Структурная схема модема



* Поставляется за отдельную плату

о состоянии) на передающий дем, реализуя тем самым псевдо-плексный режим работы моде-в.

Эхоплексный режим – это один видов полудуплексного режима, гда принимающий модем переда- обратно на передающий модем о-отображение всех поступающих него символов.

Во многих сегодняшних моде-ах есть переключатель для уста-овки полудуплексного режима. акие модемы работают не совсем том полудуплексном режиме, ко-орый был определен выше. Уста-овка переключателя приводит к ому, что модем осуществляет эхо-тображение каждого принятого имвола обратно на компьютер или а терминал. Для этого к некото-ым работающим в неавтономном режиме компьютерам нужно под-ключать терминалы, поскольку на сам работающий в неавтономном режиме компьютер эхо-отображе-ние вывести нельзя.

Модемы бывают как с синхрон-ной, так и с асинхронной связью. Большинство модемов, включая стандартные, совместимые с моде-мами Хейза модемы со скоростя-ми передачи 1200 и 2400 бод, явля-ются асинхронными. Для модемов этого типа не требуется синхрони-зация работы двух систем, как для синхронных модемов. Асинхрон-ные модемы подсоединяются к стан-дартному порту асинхронной связи компьютера и, как правило, пере-дают каждый символ отдельно. Исключением являются модемы, использующие протоколы коррек-ции ошибок.

В некоторых больших ЭВМ, в частности во многих больших ЭВМ фирмы IBM, используется синхрон-ная связь. Именно для этих прило-жений и предназначены синхронные модемы, которые сравнительно до-роги и редко требуются пользова-телям персональных компьютеров. Даже большие вычислительные уста-новки, в которых используется син-хронная связь, обычно соединены асинхронными коммутируемыми линиями передачи с отдельно рас-положенными конвертерами, под-держивающими протоколы син-хронной/асинхронной связи.

ТИПЫ МОДЕМОВ

Чтобы модем работал, его нужно подсоединить к телефонной линии. Обычно это осуществляется одним из двух способов – с помощью акустического соединения или пря-мого включения.

Акустическое соединение пред-полагает использование специально-го блока – акустического соедини-теля, в состав которого входят ре-зиновые чашки, надеваемые на труб-ку обычного телефона. (Вы долж-ны пользоваться телефоном со стан-дартной трубкой.) Акустический соединитель усиливает сгенериро-ванный модемом сигнал звуковой частоты до слышимого уровня. За-тем этот сигнал улавливается мик-рофоном телефонной трубки и пере-дается по телефонной линии на удаленный модем. Аналогично акус-тический соединитель принимает слышимый сигнал из телефонной трубки и преобразует его до уровня, воспринимаемого модемом.

Поскольку в акустических сое-динителях для передачи и приема сигналов звуковой частоты исполь-зуется телефонная трубка, они чув-ствительны и к внешним шумам. Степень влияния этих шумов зави-сит от скорости передачи. При ско-рости до 300 бод влияние шумов незначительно, но, как правило, при скорости 1200 бод и выше это влия-ние становится существенным.

Прямое включение реализуется с помощью специального оборудова-ния прямого доступа к данным (DAA – direct-access arrangement). Оно обеспечивает модему прямой доступ к телефонной сети. Прямое включение не восприимчиво к внеш-ним шумам и обеспечивает более высокий уровень сохранности дан-ных, чем при акустическом соеди-нении.

В свою очередь, акустическое со-единение имеет два преимущества перед прямым включением. Во-пер-вых, акустический соединитель можно использовать везде незави-симо от типа телефонной системы, а для модема с DAA требуется мо-дульная система, которая есть не везде. Во-вторых, для использова-ния акустического соединителя не требуется разрешение Федеральной комиссии по связи (ФКС). По-скольку акустический соединитель физически не подсоединен к теле-фонной линии, он не может поме-шать реализации телефонной связи. Кроме того, схема DAA подсоединя-ется непосредственно к телефонной линии, поэтому должна соответст-вовать требованиям ФКС, предъяв-ляемым к телефонной связи.

Интегральные схемы существен-но упростили конструкцию моде-мов. Чтобы сформировать модем с прямой связью, для большинства модемов нужны кристалл модема, развязывающий трансформатор и еще кое-какие мелочи. В таких мо-демах легко реализовать требова-ния ФКС к прямому включению в телефонную линию. Поскольку для современных модемов с прямой связью не нужны дополнительные усилители и соединительные чашки акустических соединителей, такие модемы дешевле.

Большинство модемов сегодня – это модемы с прямой связью. Они исключительно надежны и экономи-чески выгодны. Раньше такие моде-мы редко конструировались в од-ном корпусе со схемой DAA, теперь же они почти всегда монтируются либо на плате персонального ком-пьютера, либо в одном корпусе с ним. Большинство функций схемы DAA реализуется на кристалле мо-дема.

СРАВНЕНИЕ ВНЕШНИХ И ВНУТРЕННИХ УСТРОЙСТВ

Модемы изготавливаются либо в виде встроенных устройств, либо как отдельные устройства. Внутрен-ние модемы вставляются в разъем платы компьютера, а внешние моде-мы устанавливаются рядом с ком-пьютером и присоединяются к нему кабелем через порт RS-232.

Внутренний модем не мешает на столе и расположен далеко от шну-ров питания. Он немного более эко-номичен, чем внешний модем, и для его работы не нужны соединитель-ные шнуры. В отличие от внешнего модема, для подключения внутрен-него модема не нужен интерфейс RS-232. По всей вероятности, для большинства пользователей внут-ренний модем более удобен.

Хотя внешний модем и нужно соединять шнуром с портом RS-232, но он позволяет регулировать гром-кость. Внешний модем снабжен све-тоизлучающими диодами для инди-кации сигналов управления, посту-пающих с удаленного модема, несу-щей частоты, состояния линий пере-дачи-приема данных.

Внешние модемы имеют еще два достоинства. Во-первых, для них не нужен внутренний разъем. Если в компьютере нет свободных разъе-мов, то единственный выход – вос-пользоваться внешним модемом. Во-вторых, некоторые компьютеры

работают слишком быстро, чтобы можно было гарантировать надежную работу стандартных устройств, подключенных к внутренним разъемам. Например, в компьютере Campaq Deskpro 386 часто возникают неполадки при подключении плат сторонних изготовителей. Если у вас быстродействующий компьютер, то лучше выбрать внешний модем. Внешние модемы используются также и для портативных компьютеров, в которых есть порт RS-232, но нет внутреннего модема.

СТАНДАРТИЗАЦИЯ

Сейчас "стандартными" называются модемы, разработанные с учетом официальных и фактических стандартов. Однако для различных компонентов конструкции модема нужны разные стандарты. Существуют четыре разновидности стандартов: на интерфейс с компьютером, на набор команд, на протокол связи, на протокол коррекции ошибок. Во многих широко распространенных сегодня модемах протокол коррекции ошибок отсутствует, однако он есть в большинстве модемов с высокой скоростью передачи.

Для внешних модемов интерфейс с компьютером обычно осуществляется через порт RS-232. Для внутренних модемов интерфейс с компьютером осуществляется через разъем. Для реализации интерфейса недостаточно просто вставить модем в разъем. Модем должен иметь адрес в стандартном диапазоне адресов, и должен обеспечиваться стандартный доступ к стандартным регистрам с помощью заданных адресов портов. Необходим стандарт на интерфейс модема с компьютером, всеми модемами, подключаемыми к внутреннему разъему, и должен поддерживаться один и тот же интерфейс с компьютером.

Официальным стандартом, разработанным Ассоциацией электронной промышленности для внешних модемов, является интерфейс RS-232-C. Интерфейс PC-bus для внутренних модемов не был официально стандартизован, но промышленностью принят именно этот интерфейс.

Набор команд – это группа команд, воспринимаемых модемом, для выполнения определенных операций, например генерации сигнала вызова. Такие команды используются в пакетах программ связи для реализации дополнительных возможностей и повышения эффективности работы.

Самые первые модемы были "немыми": они выполняли только основные функции модема и не воспринимали команд, поступающих с подключенного к ним компьютера. Современные модемы выполняют полный "репертуар" команд. Стандарт для набора команд современного модема был разработан фирмой Hayes Microcomputer Products. Никакой модем не выдержит конкуренции на сегодняшнем рынке персональных компьютеров, если он не соответствует этому стандарту (несовместим с модемами Хейза).

Набор команд Хейза, который называется также AT-набором команд (поскольку буквы AT инициируют работу модема), позволяет запрашивать с подключенного компьютера или терминала выполнение многих функций. Стандартный набор команд Хейза приведен на вставке. С помощью команд этого набора можно просматривать и изменять содержимое внутренних регистров модема. Эти регистры позволяют задавать параметры операций, например число звонков, необходимое для того, чтобы модем начал отвечать, и число секунд ожидания тонального вызова. Описание функций регистров можно найти в инструкции по эксплуатации модема.

Способность эмулировать этот набор команд и определяет совместимость с модемами Хейза. Для обеспечения полной совместимости модем должен не только поддерживать полный набор команд Хейза, но и отвечать сообщениями, которые с точностью до буквы соответствуют сообщениям, выдаваемым модемом Smartmodem Хейза. Это позволяет пакетам программ связи, поддерживающим такой набор команд (а большинство пакетов именно его и поддерживает), правильно интерпретировать ответы модема.

Как и все стандарты на вычислительную технику, стандарты на модемы соблюдать точно иногда бывает очень трудно. В некоторых случаях модемы считаются совместимыми с модемами Хейза, если они выполняют основное подмножество набора команд Хейза, а некоторые редко используемые функции не выполняют. В других случаях модемы выполняют дополнительные команды, чтобы привлечь внимание покупателей. Даже фирма Hayes Microcomputer Products предлагает в своих самых новых модемах расширенный набор команд.

Такие отклонения от стандартов имеют определенные недостатки. Если модем не может распознать команду, то пакеты связи, поддерживающие стандартный набор команд Хейза, работают неправильно и пользователю может не удастся

Набор команд модема Хейза

A: Ответить на поступающий вызов.
C: Включить и выключить сигнал несущей частоты.
D: Набрать номер телефона.
E: Отобразить на экране символы, пересылаемые на модем.
F: Переключить модем с полудуплексного в дуплексный режим работы.
H: Снять или повесить телефонную трубку.
I: Запросить код идентификации или контрольную сумму.
M: Включить и выключить громкоговоритель в зависимости от различных условий.
O: Подключить модем к линии.
Q: Задать признак "посылать или не посылать код результата".
S: Установить переменные регистра.
V: Послать коды результата в виде цифр или полных слов.
X: Использовать базовый или расширенный набор кодов результата.
Z: Привести модем в исходное состояние.

выполнить важную функцию. Кроме того, поскольку большинство пользователей модемов целиком полагаются на программы связи, дополнительные команды, выполняемые модемом, могут оказаться недоступными для пользователя, который к тому же мог за эти средства внести дополнительную плату.

Какова вероятность того, что действующий ныне стандарт на набор команд будет изменен? Большинство пакетов программ связи (в том числе распространенный пакет Хейза Smartcom II), как правило, поддерживает стандартный набор команд Хейза. Для расширенного набора команд нужны более сложные пакеты программ связи. Пользователь должен знать, какой набор команд выполняется данным модемом, чтобы предотвратить попытку выполнения нераспознаваемой команды. Поставщики программ связи будут до тех пор препятствовать изменению существующего стандарта, пока новый стандарт для набора команд не завоюет рынок.

В качестве основного правила я рекомендую заказывать модемы, которые выполняют полный набор команд Хейза, и не платить лишних денег за расширения набора команд. Понятно, что существуют и исключения из этого правила, но, прежде чем платить, узнайте, в чем суть этих исключений.

ПРОТОКОЛ СВЯЗИ

Протокол связи определяет, каким образом поступающий с модема сигнал передается по телефонной линии. Фирма Bell Laboratories и Европейский комитет по стандартизации при МККТТ разработали соответствующие стандарты. Для почти устаревших модемов со скоростью передачи 300 бод основным является стандарт 103 фирмы Bell Laboratories. Для модемов на 1200 бод установлен стандарт 212A этой фирмы, а для модемов на 2400 бод – стандарт V.22-бис. Для модемов на 4800 и 9600 (высокоскоростной) бод стандартов нет. Хотя для таких модемов предназначаются стандарты V.29 и V.32 МККТТ, эти стандарты распространены пока не очень широко.

Протокол связи между модемами хорошо воспринимается пользователями, поскольку его действие им вполне понятно. Конечно, вы должны убедиться, что на вашем модеме используется тот же протокол, что и на модеме или модемах, с которыми Вы хотите установить связь. Если вы следуете стандарту для модемов на 1200 и 2400 бод, совместимых с модемами Хейза, то можете не беспокоиться. Риск начинается при скорости выше 2400 бод.

ПРОТОКОЛ КОРРЕКЦИИ ОШИБОК

Хотя для модемов со скоростью передачи 2400 бод и ниже это нетипично, но для нормальной работы высокоскоростных модемов нужен протокол коррекции ошибок. Уже разработано несколько таких протоколов, которыми предусматриваются сообщения о выявленных ошибках с принимающего модема на передающий. Передающий модем после приема таких сообщений повторяет передачу поврежденной информации. В большинстве случаев на высокоскоростных модемах реализовано средство автоматического ввода в действие "запасного варианта", который состоит в том, что если возможности телефонной линии ограничены, то уменьшается скорость передачи.

Как уже говорилось, стандарты на высокоскоростные модемы еще не установлены и протоколы коррекции ошибок для модемов разных изготовителей отличаются друг от друга. Почти невероятно найти два высокоскоростных модема разных изготовителей, между которыми можно установить связь. Даже в модемах, в которых используется один и тот же протокол связи, например новейший протокол V.29, как правило, используются разные протоколы коррекции ошибок.

БУДУЩЕЕ МОДЕМОВ

Чем шире распространяются модемы, тем существеннее снижается их стоимость. Хотя совместимые с модемами Хейза модемы на 1200 бод стали основным промышленным стандартом, постепенно усиливаются позиции аналогичных модемов на 2400 бод. В прошлом году цены на модемы со скоростью передачи 2400 бод резко упали, теперь эти модемы лишь немного дороже модемов на 1200 бод, но в два раза производительнее.

Хотя в модемах на 1200 бод трудно ошибиться, но новым покупателям модемов следует внимательно ознакомиться с модемами на 2400 бод, поскольку они станут следующим основным стандартом. До тех пор, пока не будут установлены твердые стандарты, избегайте использования модемов со скоростью передачи выше 2400 бод, за исключением тех случаев, когда необходима высокая производительность и такие модемы могут быть установлены во всех связываемых пунктах.

Лидер серии *JET*

Хотя цены на лазерные принтеры очень быстро снижаются, большинство пользователей по-прежнему думают, что их можно применять только в распределенных системах обработки данных. Однако благодаря фирме Hewlett-Packard (HP) вам больше не понадобится становиться в очередь, чтобы получить качественную копию текста или рисунка. Струйный принтер фирмы HP с плотностью печати 300 точек/дюйм предоставляет те же возможности, что и лазерный принтер, но стоит лишь 995 дол – полцены самого дешевого лазерного принтера.

В первую очередь эта цена бьет по матричным принтерам с головками на 18 или 24 иголки, которые стоят от 700 до 1100 дол. Приблизительно столько же стоящий 80-позиционный принтер модели DeskJet может печатать в режиме быстрой печати (draft) с той же скоростью, что и матричные принтеры (240 знаков/с). В режиме высококачественной печати (letter quality) принтер DeskJet работает быстрее матричных принтеров (120 против 80 знаков/с) и с более высокой плотностью печати (300×300 против 180×360 точек/дюйм). Он работает достаточно тихо (44 дБ) и имеет небольшие габариты (17,3×14,8 дюйма). Все это, по всей вероятности, приведет к дальнейшему распространению безударной технологии печати.

Усовершенствованная запатентованная съемная печатающая головка фир-

мы HP позволяет принтеру DeskJet печатать на обычной бумаге. В отличие от других принтеров фирмы HP (моделей ThinkJet и QuietJet), имеющих 12 сопел, у принтера DeskJet 50 сопел, и все они снабжены небольшими насадками, предотвращающими их засорение. Съемная печатающая головка принтера DeskJet стоит 18,95 дол и может напечатать в режиме быстрой печати до 1100 страниц (в 2–3 раза больше, чем головка принтера ThinkJet), а в режиме высококачественной печати – до 450 страниц.

Принтер DeskJet поставляется со шрифтами Courier, Courier Bold, Courier Compressed. Существует 12 дополнительных кассет со шрифтами. В каждой кассете по меньшей мере 4 шрифта, различающихся по типу, размеру, стилю. Кассета со шрифтом стоит от 75 до 125 дол. Загружаемый в память шрифт с универсальным набором символов продается за 95 дол.

Несмотря на то, что принтер DeckJet может печатать целую страницу в графическом режиме, не используя памяти, для загружаемых шрифтов нужна дополнительная память, которая поставляется в различных объемах по 150 дол за кассету на 128 Кбайт памяти. Кассета для имитации работы принтера Epson FX-80 стоит 75 дол. Принтер DeskJet имеет внутренний буфер емкостью 16 Кбайт, параллельный (типа Centronics) и последовательный (типа RS-232C) интерфейсы. В комплект принтера входит также устройство для автоматической подачи бумаги, для которого подходят бумага формата А4 и (при загрузке вручную) почтовые конверты 40.

Совместимость с *CP/M*

Второй способ заключается в замене ЦП 8088 на микропроцессор V20 фирмы NEC, который кроме набора команд ЦП 8088, обычно используемых на вашем ПК, может выполнять и команды ЦП 8080. Управляющая программа системы MS-DOS будет имитировать работу CP/M и для выполнения прикладных программ CP/M реализовывать набор команд ЦП 8080 на ЦП V20.

Мы рекомендуем две платы, которые легко установить и эксплуатировать: RUN CP/M, поставляемую фирмой Micro Interfaces (6824 N.W. 169th St., Miami, FL 33015, 305-623-9262; цена 239 дол) и базирующуюся на ЦП V20, и Blue Thunder – укороченную плату, выпускаемую фирмой Decimation (2065 Martin Ave., Santa Clara, CA 95050, 408-980-1678; цена 249 дол), в которой в качестве сопроцессора используется микропроцессор Z80.

Вопрос. Как можно использовать программное обеспечение CP/M на моем компьютере, совместимом с ПК фирмы IBM? Мне нужно такое средство, которое можно было бы легко установить и эксплуатировать. Я не программист и пока не хочу им становиться.

Ответ. Для программ, написанных в расчете на выполнение под управлением CP/M, требуется компьютер с ЦП 8080 фирмы Intel или Z80 фирмы Zilog. К сожалению, ЦП 8088 вашего компьютера не воспринимает системы команд ни ЦП 8088, ни ЦП Z80.

Первый способ добиться совместимости с CP/M – это установить дополнительную плату с сопроцессором, предназначенным для работы с CP/M. Программы CP/M будут храниться в памяти этого сопроцессора и выполняться на его ЦП Z80. Если возникнет необходимость в услугах операционной системы (например, для чтения файла или вывода текста на экран), то сопроцессор передаст управление специальной программе MS-DOS, которая выполнит необходимую операцию, обратившись к соответствующим средствам MS-DOS. Поскольку файлы хранятся в формате MS-DOS на дисках MS-DOS, для работы с ними не нужна операционная система CP/M.

Проблемы, возникающие при использовании дисководов большой емкости

ДЖОН ВУЛФСКИЛЛ

Если вы работаете в организации, где одни и те же файлы используются на ПК моделей PC, XT и AT, то вам, наверное, уже приходилось мучиться при переносе файлов с дисковода большой емкости (1,2 Мбайт) на дисковод с двойной плотностью записи (360 Кбайт) и обратно. Тот факт, что дисководы на 1,2 Мбайт не совместимы с дисководами на 360 Кбайт (как и используемые на этих дисководах диски), давно доказан, но недостаточно хорошо осознан.

Можно ли, учитывая различия между этими дисководами, а также между соответствующими им магнитными носителями, гарантировать надежную передачу файлов с AT на XT, не используя модема и не подключая к системе дополнительного дисковода на 360 Кбайт? Можно, но для этого существуют определенные правила (в основном неписаные), и если вы не будете неукоснительно следовать этим правилам, то велика вероятность, что ваши файлы будут повреждены. Ниже я приведу эти правила, но, прежде чем искушать судьбу, рискуя важными данными, стоит узнать, почему эти правила существуют.

НЕПАРНАЯ ПАРА

По внешнему виду дисководы на 360 Кбайт и на 1,2 Мбайт выглядят, как близнецы, но по внутреннему устройству — они лишь дальние родственники. Диск в дисководе большой емкости вращается быстрее, чем в дисководе с двойной плотностью записи (360 против 300 об/мин), информация на контроллер гибкого диска передается с большей скоростью (500 000 против 250 000 бит/с), и для обеспечения высокой плотности записи используется диск со специальным покрытием. Узкая головка чтения-записи дисковода большой емкости создает магнитное поле вдвое меньшей напряженности, чем для стандартного дисковода на 360 Кбайт. Поэтому при форматировании диска в дисководе на 1,2 Мбайт можно записать 80 узких дорожек на ту поверхность, на которую в стандартном дисководе записывается 40 дорожек.

На поверхность диска большой емкости нанесено специальное покрытие из окиси железа с добавками кобальта, повышающее коэрцитивность. Благодаря этому образуется полезная поверхность, на которой можно разместить более 1 Мбайт информации — в три раза больше, чем на стандартном диске с двойной плотностью записи. Чтобы проникнуть сквозь специальное покрытие и прочитать данные с диска на дисководе большой емкости, требуется электрический ток большей силы, чем при записи данных на дисководе с двойной плотностью записи.

ПЕРЕДАЧИ ИНФОРМАЦИИ С AT НА XT

В руководстве по PC-DOS говорится нечто о несоответствии между форматами дисков, но по этому поводу не предлагается ни объяснений, ни возможных решений проблемы несовместимости. Ниже приводится выдержка с той страницы руководства, где описывается команда FORMAT:

"Для форматирования односторонних и двусторонних дискет на дисководе большой емкости пользуйтесь параметром /4. Этот параметр предназначен для того, чтобы обеспечивать возможность использования односторонних и двусторонних дискет на дисководах большой емкости. Однако дискеты, форматированные с параметром /4, могут ненадежно читаться или записываться на одно- или двустороннем 5 ¼-дюймовом дисководе".

С помощью параметра /4 контроллеру дисковода для гибких дисков ПК AT дается указание форматировать диск на 360 Кбайт, пропуская каждую вторую дорожку. В результате получается 40-дорожечный диск более широкими, чем обычно, расстояниями между соседними дорожками.

На первый взгляд использование параметра /4 может показаться хорошим способом обеспечить на стандартном дисководе чтение диска, полученного на дисководе большой емкости. И действительно, на большинстве стандартных дисководов можно читать диски, форматированные указанным способом. Неприятности все же возникают, когда при вращении диска не удается центрировать узкие дорожки, полученные на дисководе большой емкости, относительно более широких головок чтения-записи дисковода на 360 Кбайт. Если это происходит, то на стандартном дисководе удается прочитать лишь часть содержимого дорожки диска.

В результате операция чтения выполняется ненадежно и прерывается либо из-за ошибки при чтении диска, либо из-за ошибки в данных. Как это ни смешно, но при чтении такого диска на правильно настроенном дисководе могут возникнуть ошибки, а при чтении на дисководе со слегка смещенными головками все может пройти нормально.

Очевидно, что параметр /4 предназначался для того, чтобы на дисководе большой емкости можно было использовать имеющиеся у вас диски емкостью 360 Кбайт. Однако он не рассчитан на подготовку дисков для дисководов на 360 Кбайт. Тем не менее на таких дисководах, как правило, читается диск, форматированный с параметром /4, и он будет читаться до тех пор, пока существенно не будет меняться положение головки. Однако следует помнить, что на стандартных дисководах надежная работа дисков, форматированных с параметром /4, не гарантируется.

ПЕРЕДАЧА ИНФОРМАЦИИ С XT НА AT

Передать файлы в противоположном направлении проще. Когда контроллер дисковода для гибких дисков ПК AT распознает, что поступающие (с дисковода для гибких дисков) данные были записаны не

с большой плотностью, он автоматически переключается на более низкую, соответствующую диску емкостью 360 Кбайт скорость передачи информации. (Диск продолжает вращаться со скоростью 360 об/мин.)

На дисководе большой емкости можно читать данные с диска емкостью 360 Кбайт, а также копировать с него файлы. Но если вы попытаетесь на дисководе большой емкости записать файл на стандартный диск, то ваши игры на этом закончатся. Узкие дорожки, записываемые дисководом большой емкости, не будут полностью перекрывать старые дорожки, записанные на стандартном дисководе. Когда потом вы попытаетесь прочитать эти дорожки на стандартном дисководе, то широкая головка чтения-записи захватит и старую, и новую информацию. И вновь результатом будет сообщение об ошибке при чтении с диска или об ошибке в данных.

Недавно один из наших штатных редакторов по незнанию вставил диск, форматированный на стандартном дисководе, в дисковод большой емкости, отредактировал записанный на этом диске файл и записал новую версию файла на старое место. Некоторое время спустя другой редактор попытался прочитать обновленный файл на стандартном дисководе другого компьютера.

Текст состоял из ошибочной информации и фрагментов файлов, которые располагались на диске по соседству с отредактированным файлом. Урок: контроллер дисковода для гибких дисков ПК AT не проверяет, соответствует ли вставленный вами магнитный носитель данному дисководу. Если контроллеру удается прочитать с диска таблицу расположения файлов (FAT – file allocation table), то он производит запись на любой вставленный в дисковод диск, в том числе и на диск емкостью 360 Кбайт.

Лучший способ избежать подобных неприятностей – точно определить тип диска, прежде чем производить на него запись на дисководе большой емкости. Диск большой емкости можно отличить по цвету покрытия, даже если на нем нет ярлыка изготовителя. Покрытие такого диска имеет характерный темно-серый оттенок. Покрытие диска емкостью 360 Кбайт – коричневое. Для более точной идентификации взгляните, есть ли на диске ободок вокруг центрального отверстия. Изготовители не делают таких ободков на дисках большой емкости, поскольку они могут помешать перемещению головок в дисководах на 1,2 Мбайт.

Если вы скряга или если неожиданно обнаружилось, что под рукой нет диска нужной емкости, то можете попытаться форматировать диск емкостью 360 Кбайт на дисководе ПК AT. На некоторых дисководах ПК AT это можно сделать. При форматировании будет отбраковано много секторов на самых внутренних дорожках (как правило, на дорожках 70–79), но зато вы получите вполне пригодный к употреблению диск емкостью 1 Мбайт. Хотя такой способ кажется вполне надежным, я считаю его лишь побочным продуктом моих исследований. Он не одобряется изготовителями компьютеров и не прошел достаточной проверки временем.

НАСТАЛО ВРЕМЯ СФОРМУЛИРОВАТЬ ПРАВИЛА

Ниже кратко изложены правила переноса файлов с дисководов большой емкости (1,2 Мбайт) на дисководы с двойной плотностью записи (360 Кбайт).

1. На стандартных дисководах нельзя читать информацию с дисков большой емкости и записывать информацию на эти диски.

2. С помощью параметра /4 можно форматировать диск емкостью 360 Кбайт на дисководе большой емкости для использования на этом или другом дисководе большой емкости. В DOS этот диск форматируется, как диск емкостью 360 Кбайт.

3. На стандартном дисководе, как правило, можно прочитать диск емкостью 360 Кбайт, форматированный с параметром /4 на дисководе большой емкости.

4. После того как на диск емкостью 360 Кбайт была произведена запись на дисководе большой емкости, этот диск уже не будет надежно работать на стандартном дисководе.

5. Если вы случайно нарушили правило 4, то диск можно будет читать на дисководе большой емкости.

Наилучший способ исключить неожиданности при переносе данных – заказать дополнительный дисковод на 360 Кбайт и подключить его к ПК AT. Если вы не испытываете восторга от мысли вложить деньги в дополнительный дисковод и если вам к тому же не хочется запоминать правила, то есть еще один выход. Программа, которая называется CPYAT2PC, позволяет форматировать диски емкостью 360 Кбайт на дисководах большой емкости. При копировании файлов эта программа за счет точной установки головки дисковода может записывать на диск дорожки двойной ширины. Эта программа стоит 79 дол и поставляется фирмой Microbridge Computer International Inc., 655 Sky Way, San Carlos, CA 94070, 415-593-8777.

Программирование на машинном уровне

Введение в язык ассемблера

ХАРДИН БРОДЕРЗ

Не нужно быть профессионалом, чтобы писать программы на языке ассемблера. Программа Debug позволит вам освоить ассемблер в рекордно короткий срок.

Первым языком программирования был язык ассемблера. Нули и единицы "естественного" машинного языка на языке ассемблера заменяют символами, похожими на английские слова. Когда вы пишете программу на языке ассемблера, то обращаетесь непосредственно к центральному процессору (ЦП), системе ввода-вывода BIOS и операционной системе (ОС) MS-DOS, минуя прикладные программы. Пользуясь языком ассемблера, вы начинаете понимать, что же на самом деле происходит внутри ЭВМ.

У языка ассемблера есть и другие преимущества, которые хорошо известны профессиональным программистам: это быстрый язык – он примерно в 100 раз быстрее Бейсика; ассемблер использует память эффективнее, чем языки высокого уровня, и обычно является наилучшим языком для управления устройствами ввода-вывода, в особенности экраном дисплея. Основной недостаток языка ассемблера – высокая, иногда недопустимо высокая трудоемкость программирования компенсируется при его сочетании с языком высокого уровня. Так, лучшее программное обеспечение совмещает программы, написанные на языке высокого уровня, с подпрограммами, написанными на языке ассемблера.

Как правило, для серьезного программирования на языке ассемблера необходимо приобрести транслятор, обеспечивающий перевод команд в двоичный код, воспринимаемый ЭВМ. К счастью, программа Debug.COM, поставляемая в составе ОС MS-DOS, дает возможность получать доступ к программированию на машинном языке и языке ассемблера, не покупая транслятора. Ниже вы узнаете, как вызвать программу Debug, и познакомитесь с тем, как ЭВМ хранит данные в числовом виде. После этого мы рассмотрим несколько коротких программ, демонстрирующих некоторые концепции, которые применимы и при работе с настоящими трансляторами языка ассемблера.

ЗНАКОМСТВО С ПРОГРАММОЙ DEBUG

Сначала установите на дисковод A диск с ОС MS-DOS и наберите команду DEBUG. Через одну-две секунды на следующей строке появится дефис. Этот дефис является приглашением к вводу программы Debug и указывает на то, что Debug ожидает от вас команды. Команды можно вводить как в нижнем, так и в верхнем регистрах, после набора каждой команды следует нажать клавишу возврата каретки.

Наберите букву r и нажмите клавишу возврата каретки. На экране появятся три строки. Содержание появившегося изображения объясняется на схеме начального вывода программы Debug. Вид этого изображения одинаков для всех компьютеров, хотя некоторые буквы и числа могут варьироваться. (При повторении действий, выполняемых в программах, приведенных в этой статье, вам надо набирать только те символы, которые расположены за приглашением к вводу и адресами ячеек памяти [например, 16BA:0138]. Все остальное – это комментарии или отображение того, что вы должны увидеть на экране.)

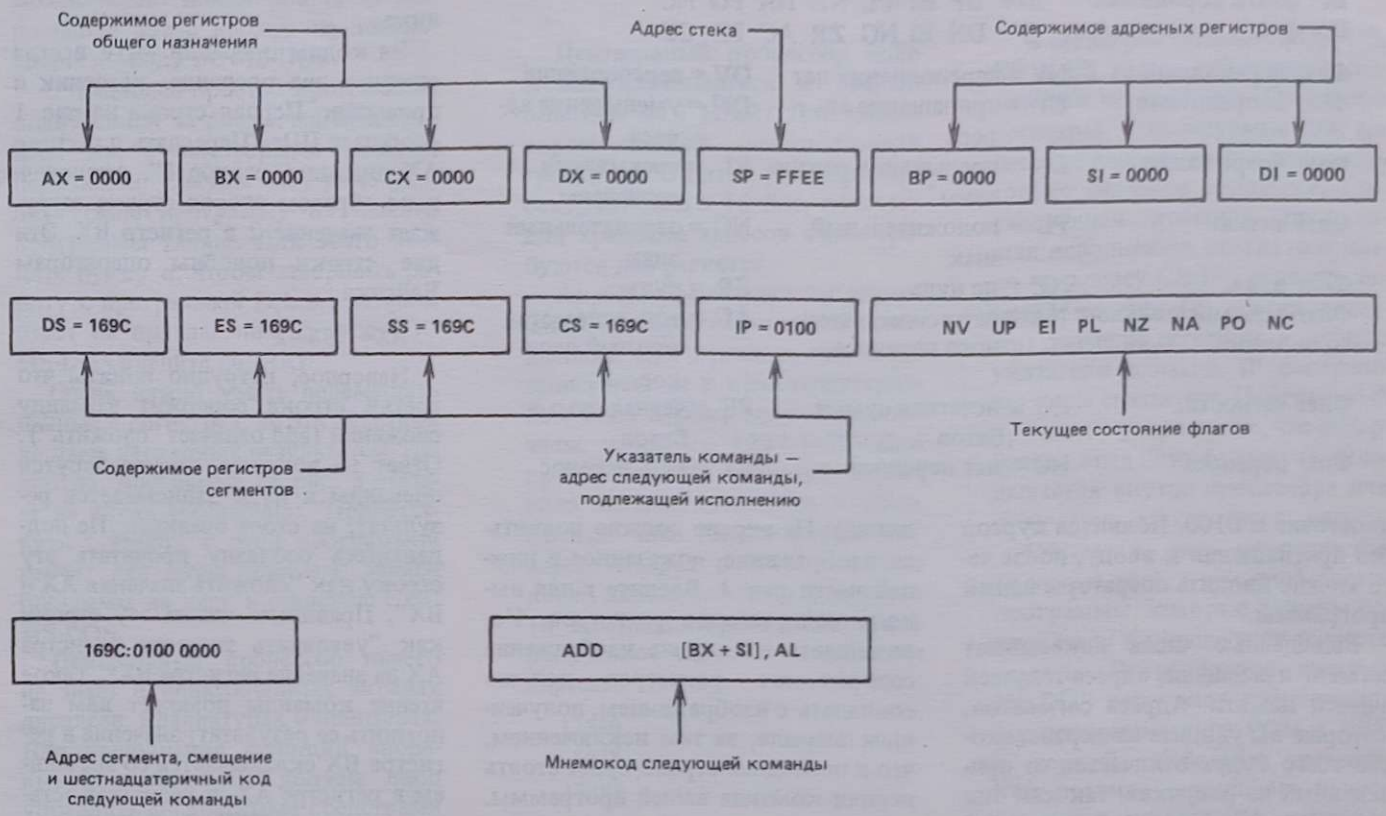
Введя r, вы предложили программе Debug показать текущее содержимое регистров ЦП. Регистры – это ячейки памяти, находящиеся в ЦП (в отличие от ячеек ОЗУ). Некоторые регистры можно использовать почти для любых целей, другие – имеют специальное назначение (если вы не знаете, как устроена адресация памяти, то ознакомьтесь с текстом на вставке "Структура памяти").

Центральный процессор тех ЭВМ, которые работают под управлением ОС MS-DOS, имеет 14 встроенных регистров. Каждый регистр содержит 16 бит, или одно слово. Первые четыре регистра (AX, BX, CX и DX) называются регистрами общего назначения, и мы достаточно свободны в способе их использования. Кроме того, каждый из этих регистров может быть использован в качестве двух 8-битовых (или 2-байтовых, что одно и то же) регистров. Старший байт регистра AX именуется AH, младший – AL. От вас зависит, в каких случаях использовать эти части памяти в качестве 16-битовых регистров, а в каких – в качестве 8-битовых.

Начальный вывод программы Debug показывает, что содержимое всех регистров общего назначения равно 0000. Такое же значение содержится и в трех регистрах указателей: BP, SI и DI. Регистр указателя базы BP используется в основном в качестве метки, помогающей обрабатывать сложные структуры данных, называемые кадрами стека.

Регистры индекса источника SI и индекса приемника DI используются в основном для пересылки содержимого больших блоков памя-

Схема начального вывода программы Debug



ти, именуемых на языке ассемблера строками, независимо от того, содержат они текстовые данные или нет.

Последний регистр в первом ряду вывода программы Debug — это указатель стека SP. Стек представляет собой структуру данных в памяти, имеющую множество назначений. При каждом вызове подпрограммы в стек заносится адрес возврата. Программисты могут использовать стек для временного сохранения содержимого регистров, а иногда и для передачи значений из одной подпрограммы в другую.

Первые четыре регистра второго ряда — это регистры сегментов. Память ЭВМ, работающих под управлением ОС MS-DOS, устроена так, что для адресации ячеек памяти необходимы регистры двух видов: регистры сегментов указывают на большие блоки памяти, а в одном из других регистров содержится адрес внутри этого блока.

Пятый регистр во втором ряду — это указатель команд IP. Этот регистр всегда содержит адрес следующей команды, подлежащей исполнению, подобно тому как в Бейсике всегда хранится номер следующей строки, подлежащей интерпретации и выполнению.

Содержимое всех 13 числовых регистров приведено в шестнадцатеричном виде. При программировании на языке ассемблера редко пользуются десятичными числами, в основном числа записывают в двоичном (по основанию 2) или в шестнадцатеричном (по основанию 16) виде.

В конце второй строки начального вывода программы Debug показаны восемь двухбуквенных кодов, отображающих состояние значащих битов регистра флагов состояния (см. таблицу флагов состояния). Для фиксации состояний этих флагов в ЦП имеется специальный 16-битовый регистр. Состояния флагов изменяются в результате выполнения многих команд языка ассемблера и могут быть проверены для того, чтобы установить, следует ли передать управление новой группе команд.

Все проверки выполнения условий на языке ассемблера производятся на основании состояний соответствующих флагов. Например, флаг нуля может поменять значение с NZ на RZ; это показывает, что результат выполнения арифметической операции равен нулю. Не все биты регистра являются значащими.

РЕЖИМ ВЫВОДА НА ЭКРАН

Пока что вы только наблюдали, но не программировали. Перед тем как написать простую программу, вам необходимо познакомиться еще с одной командой программы Debug. Если вы наберете на клавиатуре букву r, то программа Debug покажет вам начальное состояние регистров. Если же вы наберете букву r и укажете за ней имя регистра, то программа Debug покажет содержимое этого регистра и позволит вам занести в него новое значение.

Например, если вы наберете

```
rax
```

то программа Debug выведет на экран AX 0000 в первой строке и двоеточие во второй. Если вы введете 1111, то программа Debug заменит значение регистра AX на 1111H (в данной статье мы используем суффикс H для обозначения шестнадцатеричной записи числа). Помните, что для ввода и вывода чисел программа Debug использует шестнадцатеричную систему счисления.

Для набора команды на языке ассемблера введите букву a. В ответ на это программа Debug выведет на экран четырехзначное число,

Флаги состояния

Все флаги сброшены:	NV UP DI PL NZ NA PO NC	
Все флаги установлены:	OV DN EI NG ZR AC PE CY	
Флаг переполнения:	NV = переполнения нет	OV = переполнение
Флаг направления:	UP = приращение адреса	DN = уменьшение адреса
Флаг прерывания	DI = прерывания отключены	EI = прерывания включены
Флаг знака:	PL = положительный знак	NG = отрицательный знак
Флаг нуля:	NZ = не нуль	ZR = нуль
Флаг вспомогательного переноса:	NA = нет вспомогательного переноса	AC = есть вспомогательный перенос
Флаг четности:	PO = нечетная сумма битов	PE = четная сумма битов
Флаг переноса:	NC = нет переноса	CY = есть перенос

двоеточие и 0100. Появится курсор без приглашения к вводу, после чего можно вводить операторы вашей программы.

Выведенные числа показывают сегмент и смещение адреса текущей ячейки памяти. Адреса сегментов, которые вы увидите на экране, скорее всего будут отличаться от приведенных на рисунках, так как они зависят от используемой версии ОС MS-DOS, наличия резидентных программ, виртуальных дисков или программ буферизации печати. Число, стоящее за двоеточием, называется смещением и должно быть равно 0100. Все программы типа .COM начинаются с относительного адреса 100H, а программа Debug может создавать только программы типа .COM. Если смещение не равно 0100, то нажмите клавишу возврата каретки и наберите

```
a100
```

Теперь размещение команд языка ассемблера начнется с адреса 0100H.

Наша первая программа будет чрезвычайно простой. Наберите три строки операторов, показанные на рис. 1. Чтобы поля операторов располагались в столбик, используйте клавишу табуляции; после набора каждой строки следует нажимать на клавишу возврата каретки. Нажмите клавишу возврата каретки еще раз: вы должны снова получить на экране дефис – приглашение к вводу программы Debug. Для проверки выполненной вами работы введите команду

```
u100 l7
```

По этой команде Debug "ретранслирует" содержимое семи байтов с начальным адресом 100H (буква l – это сокращение слова length – длина). На экране должно появиться изображение, показанное в нижней части рис. 1. Введите r для вывода содержимого регистров. Появившееся на экране изображение содержимого регистров должно совпадать с изображением, полученным вначале, за тем исключением, что в последней строке будет стоять первая команда вашей программы.

ОСНОВНЫЕ ПОНЯТИЯ

Каждая строка программы на языке ассемблера, написанной с помощью программы Debug, состоит из двух частей. Первая часть строки всегда содержит имя команды, состоящее из двух, трех или четырех букв. Эти имена называют мнемокодами, так как они соответствуют в точности одной команде ЦП и их проще запомнить, чем последовательность двоичных чисел. Иногда их называют кодами операций, так как они представляют операции центрального процессора.

За кодом операции следуют один или два (или не следует ни одного) операнда. Число операндов зависит от конкретных команды и типа используемых ею данных. В конце строки можно поставить точку с запятой и написать комментарий.

Первым в программе идет код операции MOV, получивший свое имя от слова move (переслать). Это один из наиболее часто употребляемых мнемокодов, встречающихся в программах на языке ассемблера. Он используется для пересылки данных в регистры, в память, из регистра в регистр, из регистра в память, и наоборот. Строго говоря, название команды некорректно, так как она только *копирует* данные из одного места в другое и подобно оператору Бейсика LET не изменяет значения операнда-источника.

За кодом операции MOV всегда следуют два операнда: источник и приемник. Первая строка на рис. 1 сообщает ЦП: "Переслать в регистр AX значение, равное 1". Аналогичным образом вторая строка загружает значение 2 в регистр BX. Эти две строки подобны операторам Бейсика

```
AX=1:BX=2
```

Наверное, нетрудно понять, что третья строка содержит команду сложения (add означает "сложить"). Ответ на вопросы, откуда берутся операнды и куда записывается результат, не столь очевиден. Не поддавайтесь соблазну прочитать эту строку как "сложить значения AX и BX". Правильно читать эту строку как "увеличить значение регистра AX на значение регистра BX". Такое чтение команды поможет вам запомнить ее результат: значение в регистре BX складывается со значением в регистре AX, и результат остается в регистре AX. Эта строка аналогична оператору Бейсика

```
AX=AX+BX
```

Одно из достоинств программы Debug заключается в том, что она позволяет шаг за шагом проследить за выполнением составленной программы. Введите три раза букву t (первую букву имени команды трассировки Trace) и проследите за содержимым регистров AX и BX. Команда Trace указывает программе Debug, что следует выполнить команду вашей программы и снова выдать содержимое регистров. Это позволит вам наблюдать загрузку значений в регистры и занесение конечного результата в регистр AX. На вашем экране должно появиться изображение, похожее на то, что показано на рис. 2.

ЗАПИСЬ НА ДИСК И ИСПОЛНЕНИЕ ПРОГРАММЫ

Хотя наша первая программа и не дает интересных результатов, она позволяет понять элементы изображения содержимого регистров с помощью программы Debug, трансляции и ретрансляции программы на языке ассемблера, трассировки программы, демонстрирующей ход ее исполнения. Чтобы с помощью программы Debug создать свою программу, вам нужно научиться

записывать ее на диск так, чтобы можно было инициировать ее выполнение после выхода на приглашение к вводу ОС MS-DOS.

С помощью простой программы, приведенной на рис. 3, мы познакомим вас с некоторыми новыми понятиями. Сначала вам надо удалить свою программу из памяти программы Debug. Для этого введите букву q, чтобы завершить работу с программой Debug; затем в ответ на приглашение к вводу ОС MS-DOS введите DEBUG.

Очень важно уметь перемещать данные в регистрах, но эти действия не дают наглядных результатов. Напротив, программа на рис. 3 выдает на экран сообщение: "Приветствую!". Работа с этой программой научит вас двум приемам: записи программ на диск и общению с ОС MS-DOS.

Центральный процессор ничего не знает о подключенных экранах дисплеев, клавиатурах и принтерах. Для выполнения функций ввода-вывода программа должна либо управлять аппаратными средствами компьютера непосредственно, что является сложной и трудоемкой задачей, либо запросить помощь у ОС MS-DOS или системы BIOS.

Для составления собственных программ вам потребуются список процедур ОС MS-DOS и описание способа их использования. Эту информацию можно найти в любом справочном руководстве по ЭВМ, работающей под управлением ОС MS-DOS. Процедуры ОС MS-DOS не зависят от модели используемой вами ЭВМ, а процедуры системы BIOS почти одинаковы для всех ЭВМ, совместимых с IBM PC/XT/AT.

Первая строка программы помещает в регистр AH (старший байт регистра AX) значение, равное 09H. В дальнейшем для вывода строки на экран программа обращается к ОС MS-DOS. Во всех случаях обращения к служебным функциям ОС MS-DOS в регистр AH заносится ее номер, в остальные регистры заносится необходимая системе информация, после чего вызывается функция ОС MS-DOS. В нашем случае мы используем служебную функцию 9 ОС MS-DOS (изображение строки).

Команда второй строки загружает в регистр DX число 120H. Для вывода строки ОС MS-DOS должна знать ее местонахождение в памяти, поэтому при вызове служебной

Структура памяти

Центральный процессор 8088 может адресоваться к памяти объемом до 1 Мбайт. Для записи адреса каждой ячейки памяти требуется 20 бит, а регистры ЦП содержат всего 16 бит, поэтому для хранения адресов ячеек требуются два регистра.

Адреса ячеек задаются двумя числами: адресом сегмента и смещением. Как правило, эти числа записываются в шестнадцатеричном виде и разделяются двоеточием, например 1234:5678. Для преобразования значений адреса вида сегмент:смещение в абсолютный или "исполнительный" адрес следует дописать нуль в конец адреса сегмента и сложить (в шестнадцатеричной системе) полученное число со смещением адреса:

```
12340
 5678
------
179B8
```

Такой метод адресации позволяет записывать адреса ячеек многими способами. Например, адрес одной и той же ячейки можно записать как 2000:8000, 2800:0000, 2400:4000 и 2125:6DB0.

В регистрах сегментов CS, DS, SS и ES всегда содержится сегментная часть адреса. Остальные регистры, используемые для адресации памяти, могут содержать только значение смещения. Так, следующая команда, подлежащая исполнению, всегда находится по адресу CS:IP; регистр сегмента команд CS всегда содержит значение сегмента, а регистр указателя команд IP содержит значение смещения. Центральный процессор устроен так, что содержимое этих двух регистров складывается внутри процессора для получения абсолютного адреса.

Считается, что некорректно (а кроме того, и сложно) писать программы, которые должны исполняться в определенном месте памяти. Операционная система MS-DOS загружает практически все программы типа .COM и .EXE в первый свободный сегмент. Как правило, все адреса в программах типа .COM задаются смещениями относительно адреса начала программы. Фактическое расположение программы в памяти зависит от объема памяти, занимаемой ОС MS-DOS, драйверами устройств, резидентными утилитами и т. д.

Рис. 1. Введите три строки команд программы, которые расположены на рисунке ближе к центру. Эта простая программа показывает, как использовать общеупотребительные команды MOV и ADD языка ассемблера

```
Программа в том виде, как Вы ее набрали

A>debug
-a100
4B17:0100 mov     ax,1
4B17:0103 mov     bx,2
4B17:0106 add     ax,bx
4B17:0108

Программа после ретрансляции отладчиком DEBUG

-u100 17
4B17:0100 B80100          MOV     AX,0001
4B17:0103 BB0200          MOV     BX,0002
4B17:0106 01D8            ADD     AX,BX
```

Конец ◄

Рис. 2. Введя команду Trace программы Debug, вы увидите, как каждая строка программы, приведенной на рис. 1, изменяет содержимое регистра центрального процессора

```
-r
AX=0000  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=4B17  ES=4B17  SS=4B17  CS=4B17  IP=0100   NV UP EI PL NZ NA PO NC
4B17:0100 B80100          MOV     AX,0001
-t

AX=0001  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=4B17  ES=4B17  SS=4B17  CS=4B17  IP=0103   NV UP EI PL NZ NA PO NC
4B17:0103 BB0200          MOV     BX,0002
-t

AX=0001  BX=0002  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=4B17  ES=4B17  SS=4B17  CS=4B17  IP=0106   NV UP EI PL NZ NA PO NC
4B17:0106 01D8            ADD     AX,BX
-t

AX=0003  BX=0002  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=4B17  ES=4B17  SS=4B17  CS=4B17  IP=0108   NV UP EI PL NZ NA PE NC
4B17:0108 0A00            OR      AL,[BX+SI]                   DS:0002=00
-q
```

Конец ◄

Рис. 3. Эта программа демонстрирует сохранение программы на языке ассемблера на диске и вызов ОС MS-DOS

```
A>debug
-a
661D:0100 mov    ah,9
661D:0102 mov    dx,120
661D:0105 int    21
661D:0107 int    20
661D:0109
-e 120 'Приветствую!$'
-d 120 l 20
661D:0120  BF E0 A8 A2 A5 E2 E1 E2-A2 E3 EE 21 24 20 64 78   Приветствую!$ dx
661D:0130  2C 31 32 30 0D 0D 0A 36-36 31 44 3A 30 31 30 35   ,120...661D:0105
-rcx
CX 0000
:2d
-n program2.com
-w
Writing 002D bytes
-q
A>
```

Конец ◄

Рис. 4. Проведите трассировку для того, чтобы увидеть, как каждая строка программы, приведенной на рис. 3, изменяет содержимое регистров центрального процессора

```
A>debug
-n program2.com
-l
-r
AX=0000  BX=0000  CX=002D  DX=0000  SP=FFFE  BP=0000  SI=0000  DI=0000
DS=6635  ES=6635  SS=6635  CS=6635  IP=0100   NV UP EI PL NZ NA PO NC
6635:0100 B409          MOV     AH,09
-t

AX=0900  BX=0000  CX=002D  DX=0000  SP=FFFE  BP=0000  SI=0000  DI=0000
DS=6635  ES=6635  SS=6635  CS=6635  IP=0102   NV UP EI PL NZ NA PO NC
6635:0102 BA2001        MOV     DX,0120
-t

AX=0900  BX=0000  CX=002D  DX=0120  SP=FFFE  BP=0000  SI=0000  DI=0000
DS=6635  ES=6635  SS=6635  CS=6635  IP=0105   NV UP EI PL NZ NA PO NC
6635:0105 CD21          INT     21
-g 107
Приветствую!
AX=0924  BX=0000  CX=002D  DX=0120  SP=FFFE  BP=0000  SI=0000  DI=0000
DS=6635  ES=6635  SS=6635  CS=6635  IP=0107   NV UP EI PL NZ NA PO NC
6635:0107 CD20          INT     20
-q

Program terminated normally
-q
```

Конец ◄

Рис. 5. Три приведенные на рисунке программы используют разные команды языка ассемблера для организации одного и того же цикла. Все три программы используют цикл для того, чтобы напечатать строку 20 раз

```
A>debug
-a 100
661D:0100 jmp    130          ;Перепрыгнуть через текст сообщения
661D:0102
-e 102 "Любое Ваше сообщение" 0D 0A "$"
-a 130
661D:0130 mov    dx,102       ;DX = адрес строки
661D:0133 mov    cx,14        ;CX = счетчик цикла (14h = 20)
661D:0136 mov    ah,09        ;Выбор функции DOS: изобразить строку
661D:0138 int    21           ;Вызвать функцию DOS
661D:013A loop   138          ;Повторить вызов 20 раз
661D:013C int    20           ;Затем завершить программу
661D:013E
-rcx
CX 0000
:3e
-n prog3a.com
-w
Writing 003E bytes
-a 130
661D:0130 mov    dx,102       ;DX = адрес строки
661D:0133 mov    bx,14        ;На этот раз счетчик цикла - BX
661D:0136 mov    ah,09        ;Выбор функции DOS: изобразить строку
661D:0138 int    21           ;Вызвать функцию DOS
661D:013A dec    bx           ;Уменьшить счетчик цикла
661D:013B jnz    138          ;Повторять цикл до BX = 0
661D:013D int    20           ;Завершить программу
661D:013F
-rcx
CX 003E
:3f
-n prog3b.com
-w
Writing 003F bytes
-a 130
661D:0130 mov    dx,102       ;DX = адрес строки
661D:0133 mov    cx,14        ;Счетчиком цикла снова служит CX
661D:0136 mov    ah,09        ;Выбор функции DOS: изобразить строку
661D:0138 int    21           ;Вызвать функцию DOS
661D:013A dec    cx           ;Уменьшить счетчик цикла
661D:013B jcxz   13f          ;Выйти из цикла, если CX = 0
661D:013D jmp    138          ;Иначе перейти к началу цикла
661D:013F int    20           ;Завершить программу
661D:0141
-rcx
CX 003F
:41
-n prog3c.com
-w
Writing 0041 bytes
-q
```

Конец ◄

функции 9 вы должны поместить адрес начала строки в регистры DS и DX. Так как в регистре DS уже находится необходимая информация об используемом программой участке памяти, то вам остается установить значение регистра DX. Значение 120H превышает адрес последней команды программы, следовательно, строку с сообщением удобно поместить по адресу 120H.

Команда третьей строки вызывает ОС MS-DOS и передает ей ваш запрос. Мнемокод INT – это сокращение от английского слова interrupt – прервать; этот термин отражает способность ЦП прервать работу, реагируя на внешние события. Каждый раз, когда вы нажимаете на клавишу, аппаратные средства клавиатуры прерывают работу ЦП, который приостанавливает такую работу, принимает код нажатой вами клавиши и сохраняет его в буфере клавиатуры. Кроме того, примерно 18 раз за 1 с работа ЦП прерывается таймером для изменения счетчика времени ОС MS-DOS.

Прерывания возникают настолько часто, что ЦП необходимо иметь эффективный способ их обработки. Так, каждый представитель семейства микропроцессоров 8088 содержит в ОЗУ таблицу адресов программ обработки прерываний. В ЭВМ, совместимых с IBM PC и XT, предусмотрена обработка девяти типов аппаратных прерываний, и ЦП несложно приостановить выполняемую работу, сохранить значения требуемых регистров, найти адрес соответствующей процедуры обработки прерываний и перейти к ее исполнению. После того как обработка прерывания закончена, дается команда, возвращающая управление прерванной программе, у которой обычно нет способа узнать, имело ли место прерывание.

Хотя этот процесс и выглядит сложным, на самом деле он выполняется быстро и эффективно. Важно уяснить, что аппаратные средства, вызывающие прерывание, не знают адреса своей обрабатывающей процедуры. Они только должны сообщить ЦП номер своего прерывания. Это позволяет разработчикам ОС MS-DOS и системы ввода-вывода BIOS помещать служебные процедуры в любой участок памяти, требуется только указать в таблице прерываний адреса этих процедур.

В таблице прерываний достаточно места для 256 прерываний с номерами от 0 до 0FFH. Из них жестко закреплены только 16: девять для прерываний от аппаратных средств, семь для прерываний, задающих режим работы самого ЦП. Некоторые из остальных номеров зарезервированы для нужд ОС MS-DOS. Например, большинство служебных функций ОС MS-DOS вызывается командой INT 21. Так как для вызова служебной процедуры ОС MS-DOS вместо указания ее адреса используется прерывание, то у вас нет необходимости запоминать адреса точек входа процедур ОС MS-DOS. Таким образом, создатели ОС MS-DOS могут в разных версиях по-разному размещать в памяти эти процедуры, и при этом у программистов не возникает необходимости переписывать все свои программы.

В последней строке на рис. 3 оператор INT 20 используется для завершения программы и передачи управления ОС MS-DOS. Это наилучший способ завершения программ типа .COM, но если вы пишете на языке ассемблера программу типа .EXE, то лучше вместо этого использовать служебную функцию 4CH ОС MS-DOS.

После трансляции первых четырех строк программа готова к исполнению, но строка с сообщением еще не помещена в память. Команда ввода данных Enter программы Debug (вызываемая набором буквы e) позволяет вам занести строку байтов непосредственно в память. Команда

```
e 120
```

сообщает Debug, что необходимо поместить следующую информацию в память, начиная с адреса 120H. Последним в тексте строки должен стоять символ доллара, который указывает процедуре изображения строк ОС MS-DOS конец строки.

В следующей строке нашей программы записана команда дампа, вызываемая вводом буквы d. По этой команде на экране дисплея изображается содержимое блока памяти. Команда

```
d120 L20
```

сообщает Debug, что вы хотите посмотреть содержимое 32 (20H) байт памяти. Программа Debug изображает как шестнадцатеричные значения этих байтов, так и соответст-

Рис. 6. С помощью программы Debug мы создаем программу на языке ассемблера, которая использует функции ОС MS-DOS и BIOS для установки новых атрибутов изображения

```
Схема программы:

100: Выдать приглашение к вводу
     Считать символ с клавиатуры
        Вызвать процедуру преобразования
        Если флаг переноса установлен, начать заново
           иначе сохранить значение символа в регистре BH
     Считать символ с клавиатуры
        Вызвать процедуру преобразования
        Если флаг переноса установлен, начать заново
           иначе скомбинировать значение символа со значением BH
     Считать символ с клавиатуры
        Сравнить с возвратом каретки
        Если не совпал с ним, начать заново
     Вызвать видеофункцию BIOS для очистки экрана
     Завершить программу

150: Процедура преобразования -- преобразовать символ, содержащийся
                                 в регистре AL, в двоичное число:
        Если символ меньше '0', перейти к "выходу по ошибке"
        Если символ меньше или равен '9', перейти к "установке значения"
        Преобразовать символ в одноименный на верхнем регистре
        Если символ меньше 'A', перейти к "выходу по ошибке"
        Если символ больше 'F', перейти к "выходу по ошибке"
        Добавить 9 к значению символа

170: Установка значения:
        Обнулить четыре старшие бита значения символа
        Сбросить флаг переноса
        Возвратиться в программу

180: Выход по ошибке:
        Установить флаг переноса
        Возвратиться в программу

190: Текст приглашения к вводу
```

```
A>debug
-a 100
4B17:0100 mov    dx,190         ;DX = адрес строки
4B17:0103 mov    ah,9           ;Выбор функции DOS: изобразить строку
4B17:0105 int    21             ;Вызвать функцию DOS
4B17:0107 mov    ah,1           ;Выбор функции DOS: считать символ с клавиатуры
4B17:0109 int    21             ;Вызвать функцию DOS
4B17:010B call   150            ;Преобразовать символ
4B17:010E jc     100            ;Если обнаружена ошибка, начать заново
4B17:0110 mov    cl,4           ;Число сдвигаемых битов
4B17:0112 shl    al,cl          ;Сдвинуть в старший полубайт
4B17:0114 mov    bh,al          ;Сохранить результат
4B17:0116 int    21             ;Считать другой символ
4B17:0118 call   150            ;Преобразовать его
4B17:011B jc     100            ;Если обнаружена ошибка, начать заново
4B17:011D add    bh,al          ;Иначе скомбинировать с первым значением
4B17:011F int    21             ;Считать еще один символ
4B17:0121 cmp    al,0d          ;Возврат каретки?
4B17:0123 jne    100            ;Нет - начать заново
4B17:0125 mov    al,0           ;Иначе прокрутить вверх окно
4B17:0127 mov    cx,0           ;0,0 - координаты левого верхнего угла
4B17:012A mov    dx,184f        ;18h,4fh = 24,79 -- правого верхнего угла
4B17:012D mov    ah,6           ;Выбор функции BIOS: прокрутить окно вверх
4B17:012F int    10             ;Вызвать видеофункцию BIOS
4B17:0131 mov    dx,0           ;0,0 - координаты левого верхнего угла
4B17:0134 mov    bh,0           ;Выбрать страницу 0
4B17:0136 mov    ah,2           ;Выбор функции BIOS: переместить курсор
4B17:0138 int    10             ;Вызвать видеофункцию BIOS
4B17:013A int    20             ;Возвратиться в DOS
4B17:013C
-a 150
4B17:0150 cmp    al,30          ;Символ < '0'?
4B17:0152 jb     180            ;Да - пометить как ошибочный
4B17:0154 cmp    al,39          ;Символ < '9'?
4B17:0156 jbe    170            ;Да - получить его значение
4B17:0158 and    al,df          ;Иначе преобразовать в символ верхнего регистра
4B17:015A cmp    al,41          ;Символ < 'A'?
4B17:015C jb     180            ;Да - пометить как ошибочный
4B17:015E cmp    al,46          ;Символ > 'F'?
4B17:0160 ja     180            ;Да - пометить как ошибочный
4B17:0162 add    al,9           ;Иначе добавить смещение
4B17:0164 jmp    170            ; и вычислить значение символа
4B17:0166
-a 170
4B17:0170 and    al,0f          ;Обнулить 4 старших бита
4B17:0172 clc                   ;Обнулить признак ошибки
4B17:0173 ret                   ; и возвратиться в программу
4B17:0174
-a 180
4B17:0180 stc                   ;Установить признак ошибки
4B17:0181 ret                   ; и возвратиться в программу
4B17:0182
-e 190 0d 0a "Задайте атрибут экрана (две шестнадцатеричные цифры) > $"
-d 190 l 40
4B17:0190  0D 0A B7 A0 A4 A0 A9 E2-A5 20 A0 E2 E0 A8 A1 E3   ..Задайте атрибу
4B17:01A0  E2 20 ED AA E0 A0 AD A0-20 28 A4 A2 A5 20 E8 A5   т экрана (две ше
4B17:01B0  E1 E2 AD A0 A4 E6 A0 E2-A5 E0 A8 E7 AD E8 A5 20   стнадцатеричные
4B17:01C0  E6 A8 E4 E0 EB 29 20 3E-20 24 64 0D 0A 00 06 39   цифры) > $d....9
-rcx
CX 0000
:ca
-n screen.com
-w
Writing 00CA bytes
-q
```

Конец ◀

вующие им символы в системе ASCII. Команда дампа позволяет проверить правильность заполнения строки и ее адрес.

Для записи вашей программы на диск программа Debug должна знать ее имя и длину. Напомним, что все программы типа .COM начинаются с адреса 100H. Последний байт нашей программы имеет адрес 12CH. Следовательно, программа имеет длину 2DH байт. Для определения длины программы при записи ее на диск Debug использует значение, находящееся в регистре CX, так что в регистр CX следует занести 2DH.

Следующая строка содержит команду объявления имени программы, вызываемую вводом буквы n (от английского слова Name), в нашем случае имени Program 2.COM. Наконец, команда записи, вызываемая вводом буквы w (от слова Write), сообщает программе Debug, что данную программу следует записать на диск. В ответ программа Debug должна сообщить количество записанных байтов. После этого надо закончить работу с программой Debug и возвратиться в ОС MS-DOS. Чтобы убедиться, что длина программы в действительности равна 4D байт (2DH в шестнадцатеричном представлении), наберите команду DIR. Для вызова вашей программы введите PROGRAM2 в ответ на приглашение к вводу, полученное от ОС MS-DOS.

На рис. 4 показано, как загрузить вашу программу обратно в программу Debug, чтобы выполнить ее по шагам. Сначала вы должны объявить имя программы с помощью команды Name, а затем загрузить ее командой Load (вызываемой вводом буквы l). Обратите внимание на то, что регистр CX содержит значение 2DH – длину программы.

Выполнение первых двух команд можно проследить с помощью команды Trace, но не пытайтесь проделать трассировку двух вызовов прерываний, это приведет к тому, что вам придется наблюдать выполнение служебной процедуры ОС MS-DOS, что может занять полчаса, а то и больше. Изучение работы служебных процедур ОС MS-DOS требует часов и даже дней кропотливого анализа.

Для исполнения прерываний ОС MS-DOS используйте команду Go (вызываемую вводом буквы g). Первая команда Go содержит адрес следующей команды нашей программы. Используя такой формат команды Go, вы сообщаете программе Debug, что, достигнув этого адреса, следует остановиться и снова вывести содержимое регистров (вторая команда Go не содержит адреса). После этого программа Debug выдает сообщение, что программа завершилась успешно.

Может быть, вам захочется поупражняться в написании программ, выводящих на экран сообщения. Добавив в конец строки байты 0DH и 0AH (проследите за тем, чтобы они стояли вне кавычек), вы добьетесь перехода на начало следующей строки. При работе с Бейсиком вы, вероятно, экспериментировали с оператором PRINT, точно так же при знакомстве с программой Debug вы можете поупражняться в выводе строк с помощью ОС MS-DOS.

ОРГАНИЗАЦИЯ ЦИКЛА

В программировании почти каждая задача имеет более одного решения. На рис. 5 приведены три способа организации цикла, в котором 20 раз выводится одна и та же строка.

Программа начинается с команды перехода JMP по адресу 130H. Так как эта команда занимает два байта, то в начале программы остается достаточно места для строки длиной до 42 символов, а также символов возврата каретки, пропуска строки и признака конца. Вместо приведенной на рис. 5 строки можно набрать любое сообщение, содержащее не более 42 символов.

За JMP идет команда ввода данных Enter, которая несколько отличается от примененной в предыдущей программе. Приведенные на рис. 5 программы выводят сообщение на экран 20 раз, каждый раз с новой строки. Следует сообщить MS-DOS, что к строке следует присоединить символы возврата каретки и пропуска строки. В системе ASCII возврату каретки соответствует код 13 (0DH), а пропуску строки – код 10 (0AH). По команде Enter программа Debug вводит текст сообщения, байт 0DH, байт 0AH и символ доллара, которым согласно правилам ОС MS-DOS должна оканчиваться строка.

Начальные строки рис. 5 используются во всех трех циклах программы. Команда

```
a130
```

сообщает программе Debug, что трансляцию следует начать с адреса 130H. Напомним, что именно этот адрес указан в команде JMP в начале программы.

В первом цикле в регистр DX загружается адрес строки, в регистр AH загружается номер служебной функции ОС MS-DOS. Как и ранее, вызов функции вывода строки осуществляется командой прерывания INT 21. Следующая команда программы, loop 138, использует специальный механизм организации цикла, встроенный в ЦП. Команда loop сообщает ЦП, что следует уменьшить значение регистра CX на 1; если после этого значение CX не равно нулю, то перейти по адресу, указанному в команде; если оно равно нулю, то перейти к выполнению следующей команды. Другими словами, команда loop программы Debug аналогична оператору Бейсика Next.

Вторая версия программы использует иную технику организации цикла. На этот раз счетчик цикла загружается в регистр BX (с равным успехом можно использовать регистры CX, SI, DI или BP).

В конце цикла команда

```
dec bx
```

указывает ЦП, что значение регистра BX следует уменьшить на 1. Цикл должен продолжаться до тех пор, пока значение регистра BX не станет равным 0. В этот момент программа устанавливает флаг нуля.

Команда

```
jnz 138
```

указывает ЦП, что следует перейти по адресу 138H, если только не установлен флаг нуля. Так как флаг нуля будет установлен только тогда, когда значение регистра BX станет равным нулю, то цикл выполнится равно 20 раз. Эти две строки аналогичны операторам Бейсика

```
BX=BX-1:IF BX<>0 THEN
GOTO 138
```

К достоинствам такой структуры цикла можно отнести то, что ее можно вложить в цикл, управляемый регистром CX.

В последней версии цикла используются команды другого типа. Счетчик цикла опять загружается в регистр CX, но на этот раз значение счетчика уменьшается в конце цикла. Команда

```
jcxz 13f
```

сообщает ЦП, что следует перейти по адресу 13FH, если регистр CX содержит 0; в противном случае выполнить следующую команду. Следующая строка содержит команду безусловного перехода на начало цикла.

Три строки, контролирующие окончание цикла в третьем примере, похожи на операторы Бейсика

```
CX=CX-1:IF CX=0 THEN GOTO 13F
ELSE GOTO 138
```

Команда JCXZ (перейти, если значение регистра CX равно нулю) часто используется при организации циклов сложной структуры для проверки условия завершения цикла внутри его тела.

После того как вы оттранслировали программы, убедитесь, что они дают правильные результаты: исполните их после выхода на приглашение к вводу ОС MS-DOS. Если вы обнаружите ошибки, то выполните трассировку программ (помните, что при трассировке для исполнения команды INT следует воспользоваться командой Go).

Возможно, вас разочарует скорость выполнения этих трех программ. Считается, что программы, написанные на языке ассемблера, должны выполняться быстро, а эти программы выполняются не быстрее команды Бейсика Print. Это происходит потому, что ОС MS-DOS выводит строки посимвольно, проверяя после ввода каждого символа состояние клавиатуры, выясняя, не нажали ли вы клавишу Ctrl-Break или Ctrl-C для остановки программы. Большинство коммерческих программ не использует для вывода функции дисплея ОС MS-DOS из-за их медленного выполнения.

Управление экраном

Управлять экраном дисплея можно, изменяя значения битов байта, задающего режим изображения каждого символа. Он называется байтом атрибута изображения и расположен в памяти непосредственно перед байтом, содержащим код символа. Оба эти байта хранятся в особом участке ОЗУ, отведенном для операций с экраном.

Программа на рис. 6 запрашивает ввод значения байта атрибута в шестнадцатеричном виде. Если вы используете цветной монитор, то в приведенной ниже таблице вы найдете трехбитовый код цветов символа и фона. Поместите коды цветов в соответствующие места в байте, содержимое которого показано на рисунке "Атрибуты изображения".

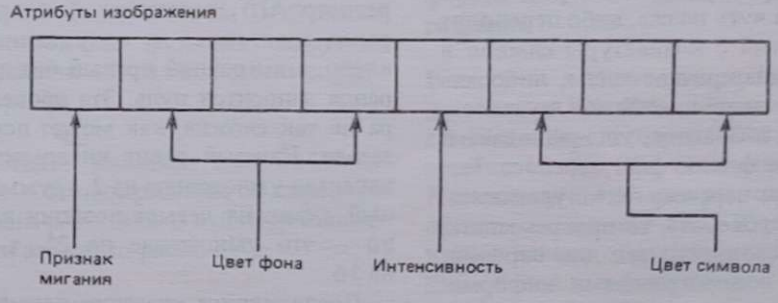
Для установки мигания или повышенной интенсивности свечения соответствующий бит должен быть равен 1, а для отмены этих режимов – 0. После установки значений битов значение байта следует перевести в шестнадцатеричную форму.

Для адаптеров монохроматического графического дисплея значение 000 соответствует черному цвету, а 111 – белому. Большинство других значений соответствует серому цвету различной интенсивности. Если значение кода цвета для фона равно 100, то символы будут выводиться подчеркнутыми.

При работе с адаптерами цветных мониторов возможны следующие цвета:

Черный	– 000
Синий	– 001
Зеленый	– 010
Голубой	– 011
Красный	– 100
Пурпурный	– 101
Коричневый	– 110
Светло-серый	– 111

Установив бит интенсивности свечения, мы получим вместо черного цвета темно-серый, вместо коричневого – желтый, а вместо светло-серого – белый.



ПОЛЕЗНО ЗНАТЬ . . .

Программа, приведенная на рис. 6, значительно длиннее предыдущих программ, однако у вас не должно возникнуть особых затруднений при ее разборе. Она использует несколько новых служебных функций ОС MS-DOS и одну функцию системы BIOS.

Эта программа запрашивает у вас две шестнадцатеричные цифры и интерпретирует их как атрибуты экрана. После этого она стирает содержимое экрана, устанавливает новый атрибут и возвращает управление ОС MS-DOS. Эти атрибуты действительны до тех пор, пока другая программа их не изменит. Чтобы научиться работать с битами байта атрибутов и переводить результат в шестнадцатеричную систему, вам следует ознакомиться с текстом вставки "Управление экраном".

При написании подобных программ с помощью программы Debug сначала следует наметить структуру программы и адреса ее разделов. Поскольку необходимо знать адреса команд еще не написанной программы, то вы должны оценить длину каждого раздела. Хотя при этом и останутся неиспользованные участки памяти, программа установки атрибутов экрана займет на диске всего 193 байт, что меньше минимального количества 1024 байт, отводимого ОС MS-DOS для дисковых файлов. Таким образом, вам не следует беспокоиться об экономии байтов.

В верхней части рис. 6 описана структура программы. Программа выводит на экран приглашение к вводу, ожидает ввода пользователем двух шестнадцатеричных цифр и символа возврата каретки, очищает экран, устанавливает требуемые атрибуты и на этом заканчивает свою работу. Программа разбита на блоки, а каждая ее строка снабжена комментарием, так что вы можете

проследить шаг за шагом за выполняемыми программой действиями.

Первая команда программы расположена по адресу 100H. Эта команда вызывает служебную функцию 9 ОС MS-DOS для вывода на экран дисплея приглашения к вводу. Затем для ввода символа с клавиатуры и отображения его на экране программа вызывает служебную функцию 1 ОС MS-DOS (начальный адрес этой команды — 107H). Пользователь видит на экране вводимые символы, но не может воспользоваться клавишей возврата и редактировать их.

Служебная функция 1 помещает введенный с клавиатуры символ в регистр AL. Затем, используя команду Call с адресом 10BH, программа передает управление подпрограмме. Команда Call программы Debug во многом напоминает оператор Бейсика Gosub, при выполнении этой команды текущий адрес программы записывается в стек и управление передается подпрограмме. После того как подпрограмма заканчивает работу, она возвращает управление в основную программу с помощью команды RET, аналогичной оператору Бейсика Return.

Подпрограмма, которая будет описана чуть позже, либо переводит введенный с клавиатуры символ в шестнадцатеричное число, либо, если этот символ не лежит в установленном диапазоне, устанавливает в регистре флагов флаг переноса. Так как флаг переноса легко устанавливать и проверять, то программисты часто используют его для передачи между подпрограммами информации об успешном или ошибочном завершении выполнения. В строке, следующей за командой Call, стоит команда JC (передать управление, если установлен флаг переноса).

Как правило, значения флагов проверяются для того, чтобы в зависимости от состояния одного или нескольких битов регистра флагов передать управление определенным разделам программы. Эти условные переходы похожи на оператор Бейсика IF . . . THEN GOTO. В языке ассемблера около 30 видов команд условного перехода. Многие из имен этих команд являются синонимами, поэтому не волнуйтесь, если при трассировке программы вам покажется, что программа Debug изменила команду перехода.

Если флаг переноса не установлен, то это означает, что подпрограмма перевода символов кода ASCII завершилась успешно и в регистр AL занесено шестнадцатеричное значение от 00H до 0FH, зависящее от нажатой клавиши. Так как это первая из двух шестнадцатеричных цифр, вводимых пользователем, то в действительности нам нужно значение от 00H до 0F0H, т. е. программа должна сдвинуть полученное значение из нижней половины байта в верхнюю.

Этот сдвиг можно выполнить двумя способами: либо умножить значение байта на 10H, либо сдвинуть каждый бит на четыре позиции влево. Второй метод быстрее и проще.

Команда с адресом 110H помещает число сдвигаемых битов (4) в регистр CL. В следующей строке команда

```
shl al,cl
```

сообщает ЦП, что необходимо сдвинуть влево значение регистра AL на число позиций, равное значению регистра CL. На каждом шаге этой процедуры текущее значение флага переноса заменяется на значение крайнего левого бита операнда (которым в нашем случае является регистр AL), остальные биты операнда сдвигаются на одну позицию влево, а в крайний правый бит операнда заносится нуль. Эта процедура не так сложна, как может показаться. Каждый сдвиг влево эквивалентен умножению на 2, а суммарный сдвиг на четыре позиции влево — это умножение на 2^4, т. е. на 16.

После сдвига значение регистра AL загружается в регистр BH. Причина, по которой выбран именно этот регистр, будет объяснена позже.

Ни одна из выполненных до сих пор команд не изменила начального значения регистра AH, равного 1. Это позволяет программе запросить ввод следующего символа с клавиатуры с помощью еще одного вызова INT 21. Для обработки символа программа снова вызывает подпрограмму преобразования символа и проверяет успешность ее завершения по флагу переноса.

Если подпрограмма преобразования символа не выдает сообщения об ошибке, то новое значение AL (от 00H до 0FH) складывается со значением, находящимся в BH. И наконец, третий символ клавиатуры, который должен быть символом возврата каретки, вводится командой вызова служебной функции 1 ОС MS-DOS. Эта команда имеет адрес 11FH. После этого программа использует команду CMP с адресом 121H для того, чтобы сравнить значение AL с кодом возврата каретки. В следующей строке еще одна команда условного перехода JNE (перейти, если не равно) передает управление в начало программы при условии, что пользователь не ввел символа возврата каретки.

Если программа дошла до адреса 125H, то это означает, что пользователь ввел две шестнадцатеричные цифры и символ возврата каретки, программа перевела эти цифры в двоичное представление и записала их в регистр BH. Теперь настало время стереть содержимое экрана и установить новые атрибуты.

В ОС MS-DOS нет служебных функций, стирающих содержимое экрана, устанавливающих новые атрибуты или положение курсора. Если вы используете драйвер консоли ANSI.SYS ОС MS-DOS, то можно набрать специальные последовательности символов, инициирующих такие действия, но они, как правило, выполняются медленно и не работают без установки файла ANSI.SYS. На всех ЭВМ, совместимых с IBM PC/XT/AT, команда INT 10 вызывает одну из процедур системы BIOS, хранящихся в ПЗУ и управляющих экраном, изменяющих режимы, положение курсора, выводящих на экран символы псевдографики и т. д.

Вызываемая по команде INT 10 служебная функция 6 прокручивает на заданное число строк любой активный участок экрана (точно так же служебная функция 7 прокручивает участок вниз). Для вызова этой функции следует поместить ее номер (6) в регистр AH, загрузить число строк, на которое требуется выполнить прокрутку, в регистр AL, занести в регистры CX и DX координаты левого верхнего и правого нижнего углов прокручиваемого окна и поместить в регистр BH устанавливаемые атрибуты. Так как наша программа уже занесла значения атрибутов в регистр BH, то это действие мы опускаем.

По команде с адресом 125H программа помещает нуль в регистр AL, чтобы указать на то, что окно должно быть очищено целиком.

Команда загрузки нуля в регистр CX (адрес 127H) эквивалентна команде загрузки нулей в регистры CH и CL, это указывает системе BIOS, что верхний левый угол окна находится в нулевом столбце нулевой строки (номера строк и столбцов экрана всегда начинаются с нуля, а не с единицы). Поместив в регистры DH и DL значения, равные 18H (24 в десятичном виде) и 4FH (79 в десятичном виде) соответственно, вы укажете, что нижний угол окна находится в 79-м столбце 24-й строки. Вместо двух команд, заносящих эти значения, в программе используется одна команда

```
mov dx,184F
```

Наконец, программа помещает в регистр AH номер служебной функции и вызывает прерывание командой INT 10.

Вызванная этой командой служебная функция системы BIOS стирает содержимое экрана, используя для этого символы пробелов и новый байт атрибута в регистре BH. Однако она не перемещает курсора в левый верхний угол экрана; вы еще не завершили ассемблерный эквивалент команды очистки экрана CLS языка Бейсик.

Положением курсора управляет служебная функция 2 системы BIOS. Для ее вызова программа должна поместить в регистры DH и DL значения, указывающие положение курсора, записать номер страницы экрана в регистр BH и номер служебной функции (2) в регистр AH. Программа заносит в регистр DX нулевое значение, так как мы хотим, чтобы курсор находился в левом верхнем углу. Если только предыдущая программа не изменила состояния видеопамяти, то текущее значение номера страницы экрана равно нулю и это значение программа на рис. 6 заносит в регистр BH. После этого она снова вызывает прерывание INT 10 для того, чтобы перевести курсор в начало экрана. Последним действием мы возвращаем управление ОС MS-DOS с помощью команды INT 20.

ПРЕОБРАЗОВАНИЕ ЧИСЛА

Оставшаяся часть программы в основном занята преобразованием набранного на клавиатуре символа в шестнадцатеричную цифру. Например, если пользователь нажимает клавишу 5, то в регистр AL будет занесено значение 35H, которое в

Словарь терминов

Адрес – число, определяющее положение в памяти (ОЗУ или ПЗУ) байта или слова. В ЭВМ, работающих под управлением ОС MS-DOS, каждый адрес может быть представлен единственным образом 20-битовым числом. Всего имеется 1 048 576 (1 М) адресов.

Байт – восемь бит, сгруппированных в логическое целое. Обычно информация хранится в байтах, команды машинного кода имеют длину один байт и больше.

Бит – наименьшая единица информации. Представлен в ЭВМ напряжениями +5 и 0 В, которым часто ставят в соответствие значения 1 и 0. Это слово является сокращением от английского binary digit – двоичная цифра.

Дисковая операционная система (DOS) – комплекс процедур, позволяющий программистам организовывать обмен данными с дисками, клавиатурой, экраном и другими аппаратными средствами ЭВМ. На DOS также возложены функции загрузки, исполнения и завершения программ.

Кбайт – килобайт, или 1024 байт. К – это сокращение от *кило*, что означает 1000. В действительности это число равно 2^{10}.

Математический сопроцессор – процессор, сконструированный для быстрого выполнения операций над числами с плавающей запятой. В качестве математического сопроцессора в ЭВМ, совместимых с IBM PC/XT, часто используется сопроцессор Intel 8087, в то время как в ЭВМ класса IBM AT обычно используется сопроцессор 80287.

Мбайт – мегабайт, или 1 048 576 байт. Приставка *мега* означает 1 000 000. На самом деле мегабайт содержит 2^{20} байт.

ОЗУ – оперативное запоминающее устройство. Центральный процессор может изменять содержимое ОЗУ. Как правило, этот термин относится к памяти, отведенной для ОС MS-DOS и программ. ЭВМ, работающие под управлением MS-DOS, имеют в зависимости от конфигурации максимальный объем памяти ОЗУ 640 или 704 Кбайт.

Параграф – 16 байт. В ОС MS-DOS сегменты памяти поделены на параграфы.

Полубайт – половина байта, или четыре бита, объединенные в логическое целое.

ПЗУ – постоянное запоминающее устройство. В ПЗУ содержится неизменяемый программный код, "прожженный" в нем на заводе. Центральный процессор может выполнять программы, находящиеся в ПЗУ, но не может их изменять. В ПЗУ содержатся программы начальной загрузки, процедур системы BIOS и, в ЭВМ фирмы IBM, часть интерпретатора языка Бейсик.

Регистр – ячейка памяти, находящаяся в ЦП. Каждый регистр семейства процессоров 8088 содержит 16 бит, или одно слово. К каждому из четырех регистров общего назначения (AX, BX, CX и DX) можно обращаться как к двум 8-битовых регистрам.

Сегмент – старшие 16 бит 20-битового адреса ячейки памяти.

Слово – 16 бит, или 2 байта.

Смещение – младшие 4 бита 20-битового адреса.

ЦП – центральный процессор, основная микросхема персональной ЭВМ. В большинстве ЭВМ, совместимых с IBM PC/XT, используются процессоры 8088 или 8086 фирмы Intel. В ЭВМ Tandy 2000 и некоторых других используется процессор 80186, а в ЭВМ типа AT – процессор 80286.

системе ASCII соответствует символу 5. Для того чтобы перевести 35H в 05H, необходима подпрограмма, которая должна также проверять, что пользователь ввел допустимую шестнадцатеричную цифру.

Подпрограмма, выполняющая эти действия, начинается с адреса 150H. Она открывается серией проверок и условных переходов. Если содержимое регистра AL меньше 30H, то это означает, что оно не является допустимой шестнадцатеричной цифрой. Если же оно лежит в пределах от 30H до 39H включительно, то соответствует десятичной цифре и в этом случае может быть переведено непосредственно в двоичную цифру. Команда JB (перейти, если меньше) с адресом 152H означает "передать управление, если при последней проверке левый операнд оказался меньше правого". Команда JBE, стоящая двумя строками ниже, означает "перейти, если значение оказалось меньшим или равным".

Если код набранного на клавиатуре символа не лежит между значениями 0 и 9, то он может соответствовать букве алфавита, которая могла быть набрана как на верхнем, так и на нижнем регистрах. Если вы посмотрите на таблицу символов системы ASCII, то увидите, что коды букв верхнего и нижнего регистров отличаются значением одного бита: пятый бит равен 1 в нижнем регистре и 0 – в верхнем. Команда с адресом 158H

```
and al,df
```

использует логическую операцию И для принудительного обнуления пятого бита. Аналогичный прием применяется в Бейсике, где для этого часто используют выражения типа

```
CH$=CHR$(ASC(CH$) AND &HDF)
```

После этого программа выполняет еще два теста для проверки того, что символ клавиатуры, код которого содержится в регистре AL, действительно лежит между символами A и F. Если это так, то, перед тем как передать управление по адресу 170H, программа увеличивает значение кода символа на 9. Смысл этого трюка заключается в следующем. В коде ASCII символам A–F соответствуют значения 41H–45H. Прибавляя к ним 9, мы переводим эти значения в диапазон 4AH–4FH и получаем правильное значение второй цифры.

Если символ является допустимым, то в регистре AL теперь находится значение, лежащее в диапазоне 30H–39H или 4AH–4FH. Нам необходимо получить значение в диапазоне 00H–0FH, для этого достаточно обнулить первую половину байта, что и выполняет команда And, расположенная по адресу 170H. После этого команда CLC обнуляет флаг переноса, а команда RET возвращает управление основной программе.

Если пользователь ввел недопустимый символ, то программа передаст управление по адресу 180H, где командой STC (установить флаг переноса) устанавливается флаг переноса, а затем управление возвращается основной программе. Все, что нам осталось сделать, это поместить в память нужное приглашение к вводу с помощью команды ввода данных программы Debug, вывести на экран содержимое блока памяти для того, чтобы выяснить длину программы, а затем сохранить программу на диске. (Так как приглашение к вводу начинается с символов возврата каретки и пропуска строки, то оно всегда выводится с новой строки, даже если пользователь допустит ошибку и программа начнет выполняться сначала.)

После сохранения программы вам наверняка захочется вернуться в ОС MS-DOS и исполнить эту программу. Если в результате исполнения программы вы обнаружите ошибки, то снова вызовите программу Debug, загрузите свою программу и проведите ее трассировку (помните, что нельзя трассировать вызовы прерываний). Отладка и трассировка – необходимые элементы процесса написания программ на языке ассемблера, так как почти невозможно избежать ошибок в первом варианте текста программы.

ЧТО ДАЛЬШЕ?

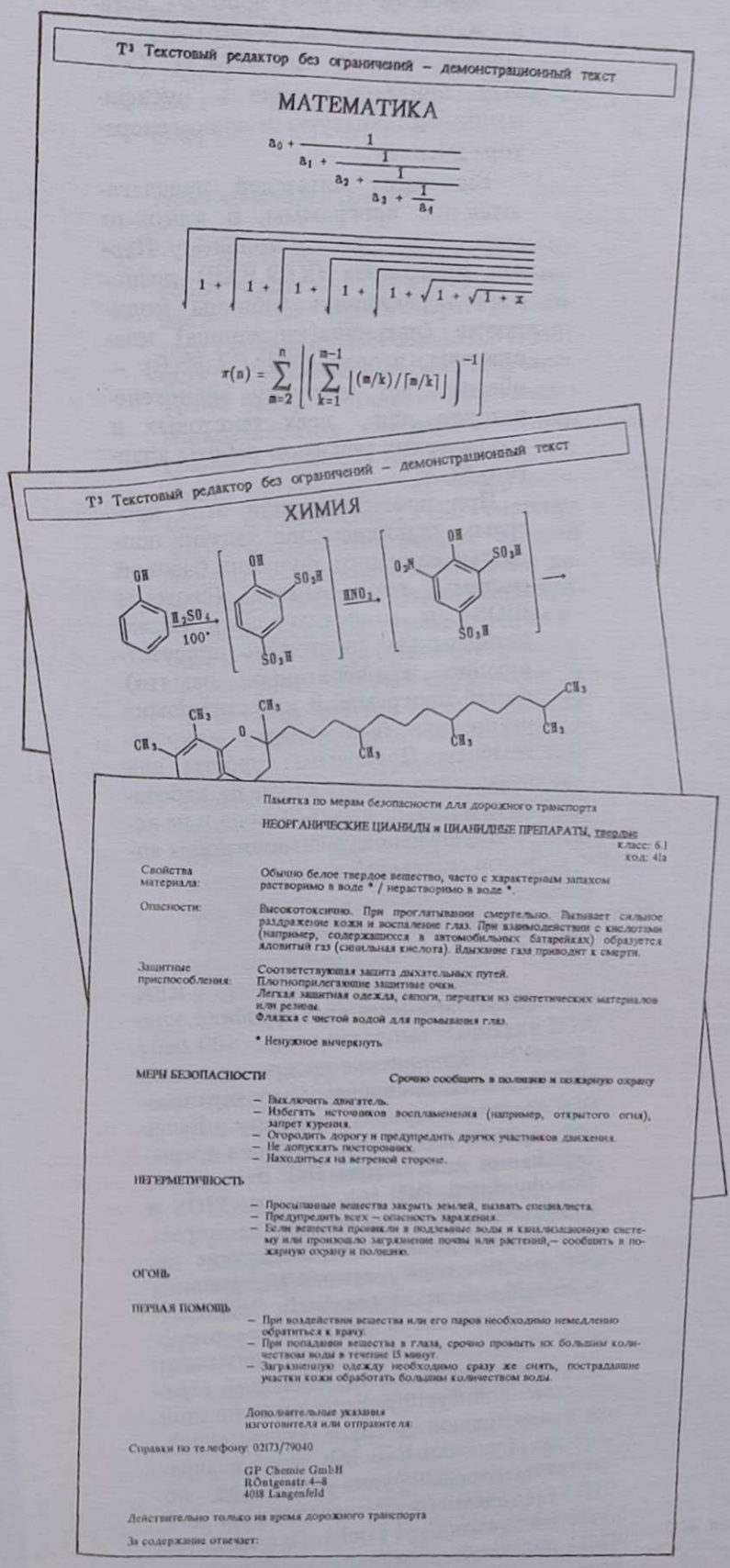
Если вам понравилось это краткое введение в язык ассемблера, то вам, наверное, захочется попробовать свои силы в составлении более сложных программ и изучить полную систему команд ЦП. Для этого вам потребуется документация по служебным функциям ОС MS-DOS и прерываниям системы BIOS, а также полный список команд центрального процессора. Существует много хороших книг по языку ассемблера, где можно найти и то, и другое.

Скорее всего вас перестанут удовлетворять возможности программы Debug.COM, и вам захочется пользоваться более мощным транслятором и лучшими средствами трассировки. Стандартным транслятором, который к тому же чаще всего используется в журнальных статьях, является макроассемблер MASM фирмы Microsoft. Последние версии макроассемблера MASM содержат программу Symdeb, которая является большим шагом вперед по сравнению с программой Debug.COM.

Лучше всего учиться, упражняясь с короткими программами. Вы обнаружите, что программу на языке ассемблера, содержащую хорошие комментарии, читать не сложнее, чем программу той же длины на языке высокого уровня. Программируя на языке ассемблера, вы будете лучше понимать устройство ЭВМ, что сделает вас более искусным программистом независимо от того, какой язык высокого уровня вы используете.

КОМПЬЮТЕРНАЯ ОБРАБОТКА ТЕКСТОВ НЕ ЗНАЕТ ГРАНИЦ

Универсальная программа обработки текстов



С использованием персонального компьютера программа позволяет обрабатывать самые разнообразные тексты; письма, корреспонденцию, доклады, договоры, формуляры, научно-техническую информацию, журнальные материалы…

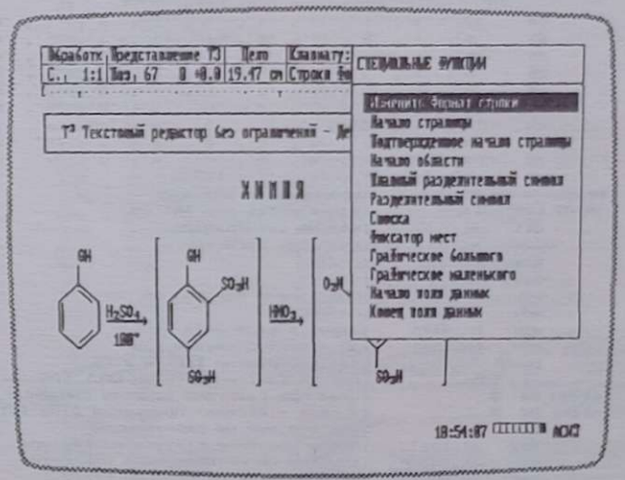
Программа и документация на русском языке.

Техника меню-опросов существенно упрощает работу с программой.

Программа содержит в себе вспомогательную информацию для пользователя, которая может вызываться в случае необходимости.

Форма распечатанных текстов в точности соответствует представлению на дисплее (What you see, is what you get).

Предусмотрен целый ряд функций, повышающих эффективность работы: постоянные тексты в верхней и нижней частях страницы, нумерация страниц, сноски, индексы, блочные операции типа копирования, стирания, изменения алфавитных наборов знаков, подчеркивания и т.д.



Представление на экране дисплея

Система контроля орфографии русского языка находится в стадии подготовки.

Многочисленные функции компоновки текстов: пропорциональный шрифт в блочном наборе, многоколонное печатание, автоматическая верстка страниц, впечатывание графиков.

Значительное количество русских алфавитных наборов знаков различной формы и размеров, а также специальные знаки всех восточноевропейских языков.

Программа рассчитана на персональные компьютеры, совместимые с системой фирмы IBM; решение достигнуто исключительно методами программного обеспечения, не требуется специального согласования персонального компьютера или печатающего устройства. Для использования программы необходимо только ОЗУ на 640 кбайт, магнитное ПЗУ, плата графики и печатающее устройство.

По запросу высылается подробная информация на русском языке.

ADA

ADA Computer und Peripherie GmbH
Dohrweg 63, 4050 Mönchengladbach 1,
Telefon: 02161/60060
Telex: 8529148 rfi
Telefax: 02161/60074

WestLB
Krasnopresnenskaya Nab 12, Fl. 12
123610 Moskau

Программы "русификации" клавиатуры и видеоадаптера ПК

А.В.КОЗЛОВ

Текст программы KBR

```
; KBR.ASM - перекодировщик клавиатуры.
; Выберем в качестве переключателя латиницы/кириллицы клавишу
; Ctrl. Переключатель срабатывает, если два клавиатурных
; прерывания подряд были вызваны: первое - нажатием, а второе -
; отпусканием клавиши Ctrl. Если включен режим кириллицы, то
; ASCII - код, возвращаемый прерыванием 16 должен быть
; транслирован по таблицам (отдельная таблица для Caps Lock).
; Следует обратить  особое внимание на то, что в случае, когда
; BIOS возвращает позиционный код (SCAN), равный нулю (функции 0
; и 1 прерывания 16), трансляция не нужна.
CODE SEGMENT
     ASSUME CS:CODE
my      db  0      ; 1 означает, что предыдущее прерывание
                   ; было вызвано нажатием клавиши CTRL
Cyrillic db 0      ; переключатель режима
                   ; 0 - латиница, FF - кириллица
ORDTBL label byte ; таблица перекодировки для режима "кириллица"
      db    32,  '1', 'Э', '3', '4', '5', '7', 'э', '9', '0'
      db   '8', '+', 'б', '-', 'ю', '/', ')', '!', '"', '№'
      db   'Й', ':', ',', '.', ';', '(', 'Ж', 'ж', 'Б', '='
      db   'Ю', '?', '2', 'Ф', 'И', 'С', 'В', 'У', 'А', 'П'
      db   'Р', 'Ш', 'О', 'Л', 'Д', 'Ь', 'Т', 'Щ', 'З', 'Й'
      db   'К', 'Ы', 'Е', 'Г', 'М', 'Ц', 'Ч', 'Н', 'Я', 'Х'
      db   '\', 'ъ', '6', '_', '`', 'ф', 'и', 'с', 'в', 'у'
      db   'а', 'п', 'р', 'ш', 'о', 'л', 'д', 'ь', 'т', 'щ'
      db   'з', 'й', 'к', 'ы', 'е', 'г', 'м', 'ц', 'ч', 'н'
      db   'я', 'х', '|', 'ъ', '~', 127
CPSTBL label byte ; таблица перекодировки для режима
                  ; "кириллица" Caps Lock.
      db    32,  '1', 'э', '3', '4', '5', '7', 'Э', '9', '0'
      db   '8', '+', 'Б', '-', 'Ю', '/', ')', '!', '"', '№'
      db   'Й', ':', ',', '.', ';', '(', 'Ж', 'Ж', 'б', '='
      db   'ю', '?', '2', 'Ф', 'И', 'С', 'В', 'У', 'А', 'П'
      db   'Р', 'Ш', 'О', 'Л', 'Д', 'Ь', 'Т', 'Щ', 'З', 'Й'
      db   'К', 'Ы', 'Е', 'Г', 'М', 'Ц', 'Ч', 'Н', 'Я', 'Х'
      db   '\', 'ъ', '6', '_', '`', 'Ф', 'И', 'С', 'В', 'У'
      db   'а', 'п', 'р', 'ш', 'о', 'л', 'д', 'ь', 'т', 'щ'
      db   'з', 'й', 'к', 'ы', 'е', 'г', 'м', 'ц', 'ч', 'н'
      db   'я', 'х', '|', 'ъ', '~', 127
presence dw 4376h       ;код присутствия
INT09 LABEL BYTE        ;точка входа для INT 09
      cli               ;код не реентрантен
      push  ax          ;сохраним регистры
      push  bx          ;--//--
      push  ds          ;--//--
      xor   bx,bx       ;подготовим сегментный регистр
      mov   ds,bx       ;--//--
      mov   bx,417h     ;адрес в BIOS DATA AREA
      mov   ah,[bx]     ;ah = статус клавиатуры
      pushf             ;для инструкции IRET в BIOS
      DB  9Ah           ;первый байт инструкции CALL FAR
old9o DW  0             ;сюда при загрузке занесем смещение
old9s DW  0             ;а сюда - сегмент программы BIOS INT 9
      cli               ;код все еще не реентрантен
      mov   al,[bx]     ;новое значение статуса клавиатуры
      and   ax,0404h    ;очистим не интересующие нас биты
      cmp   ax,0004h    ;было ли нажатие клавиши Ctrl?
      jnz   fut1        ;нет, см. ниже
      mov   cs:my,1     ;да, запомним нажатие Ctrl
      jmp   bye         ;переход на возврат из прерывания
fut1: cmp   ax,0400h    ;было ли отпускание клавиши Ctrl?
      jnz   fut2        ;нет, см. ниже
      cmp   cs:my,1     ;было ли предыдущее прерывание
                        ;вызвано нажатием Ctrl?
      jnz   fut2        ;нет, см. ниже
      not   cs:Cyrillic;переключим режим
fut2: mov   cs:my,0     ;забудем про предыдущее нажатие
bye:  pop   ds          ;восстановим регистры
      pop   bx          ;--//--
      pop   ax          ;--//--
      iret              ;вернемся из прерывания
INT16 LABEL BYTE        ;точка входа для INT 16h
      or    ah,ah       ;функция 0?
      jz    func0       ;выполним функцию 0
      cmp   ah,01       ;функция 1?
      jz    func1       ;выполним функцию 1
      db    0EAh        ;первый байт инструкции JMP FAR
old16 label dword       ;метка двойного слова
old16o dw   0           ;сюда поместим смещение,
old16s dw   0           ;а сюда сегмент программы BIOS INT 16h
func0:                  ;функция 0 - чтение кодов клавиши
      pushf             ;для инструкции IRET в BIOS
      call  cs:[old16]  ;вызов BIOS INT 16h
      cmp   cs:Cyrillic,0 ;латиница?
      jz    bye16       ;да, латиница, см.ниже
      call  translate   ;нет, кириллица, выполним трансляцию
bye16:iret              ;вернемся из прерывания
func1:                  ;функция 1 - проверка нажатия клавиши
      pushf             ;для инструкции IRET в BIOS
      call  cs:[old16]  ;вызов BIOS INT 16h
      pushf             ;сохраним флаги
      cmp   cs:Cyrillic,0 ;латиница?
      jz    by116       ;да, латиница см.ниже
      call  translate   ;нет, кириллица, выполним трансляцию
by116:                  ;
      popf              ;восстановим флаги
      retf  2           ;вернемся из прерывания
```

Продолжение ▶

Одной из первых проблем, встающих перед новоиспеченными пользователями ПК типа IBM PC (XT, AT), является проблема "русификации" клавиатуры и знакогенератора дисплея.

Вниманию читателей предлагаются две программы, в какой-то мере решающие эту проблему. Первая программа (KBR.EXE) позволяет переключать таблицы кодировки (латиница/кириллица) клавиатуры, вторая (EGAGA.EXE) – обеспечивает загрузку знакогенераторов для всех текстовых и графических режимов работы адаптера EGA.

При проектировании этих программ ставились две задачи: полностью сохранить функции базовой системы ввода-вывода (системы BIOS) и минимизировать объем резидентных (постоянно присутствующих в оперативной памяти) частей программ. В качестве языка реализации был выбран макроассемблер. Программы просты для понимания, компактны, не работают с портами ввода-вывода и не используют недокументированных адресов системной памяти.

ПРОГРАММА KBR

Резидентная программа KBR (KeyBoard Russian – русская клавиатура) занимает около 600 байт ОЗУ. Выполнение программы начинается с проверки, предотвращающей повторную загрузку. Далее происходят чтение векторов прерывания клавиатуры INT 9 и адреса функций INT 16h системы BIOS и замена этих векторов на адреса подпрограмм KBR. В качестве переключателя латиница/кириллица выбрана клавиша Ctrl. Однократное нажатие и немедленное отпускание этой клавиши вызывают инвертирование управляющей переменной Cyrillic. Если значение этой переменной не нуль (кириллица), то функции INT 16h перед возвратом транслируют ASCII-код по таблицам (отдельная таблица для состояния Caps Lock). При трансляции необходимо оставлять без изме-

нения ASCII-коды всех не лежащих в поле пишущей машинки клавиш и, кроме того, желательно обнулять позиционный код всех транслированных клавиш. Такая трансляция позволяет наиболее безболезненно вводить старшие ASCII-коды с клавиатуры, поскольку эквивалентна стандартному способу ввода десятичного эквивалента ASCII-кода в цифровом поле (Alt-Keypad).

ПРОГРАММА EGAGA

Резидентная программа EGAGA (EGA Generator Auxiliary – вспомогательный знакогенератор адаптера EGA) занимает около 6,8 Кбайт ОЗУ. Из этого объема большую часть (6144 байт) составляет собственно знакогенератор. Общая схема программы похожа на приведенную выше. Подпрограмма обслуживания функций INT 10h системы BIOS анализирует запрос и, если запрос не относился к смене режима или к функциям знакогенератора, немедленно передает управление в ROM BIOS. При смене режима управление также передается в ROM BIOS, однако не инструкцией JMP, а инструкцией CALL, затем в зависимости от режима загружается необходимый знакогенератор (размер матрицы знака 8X8 или 8X14). Запросы к знакогенератору анализируются, и там, где это необходимо, вместо адреса фиксированного в ПЗУ шрифта подставляется адрес соответствующего шрифта в программе EGAGA.

Чисто программная замена знакогенератора в видеоплате EGA имеет следующие преимущества перед изменением прошивки ПЗУ: во-первых, неквалифицированное вмешательство в EGA BIOS может плохо закончиться (как для микросхем, так и для монитора); во-вторых, желательно сохранить оригинальный знакогенератор для программ, использующих математические символы и буквы национальных алфавитов.

Для уменьшения задержки из-за необходимого анализа кода функции применяется техника самомодифицирующегося кода. Вместо занесения оригинального вектора прерывания в переменную для инструкции косвенного перехода (JMP [переменная]) использована инструкция прямого перехода (JMP <адрес>), формируемая в коде программы в момент загрузки.

► *Продолжение*

```
translate proc near     ;(флаги сохранены)
      push  bx          ;сохраним регистры
      push  ds          ;--//--
      xor   bx,bx       ;подготовим сегментный регистр
      mov   ds,bx       ;--//--
      mov   bx,417h     ;адрес в BIOS DATA AREA
      test byte ptr [bx],40h ;режим Caps Lock?
      jnz   capslock    ;да, см. ниже
      lea   bx,ORDTBL   ;нет, используем первую таблицу
      jmp   dotrans     ;переход на трансляцию
capslock:               ;
      lea   bx,CPSTBL   ;альтернативная таблица для Caps Lock
dotrans:                ;
      or    ah,ah       ;SCAN присутствует?
      jz    notrans     ;нет, трансляция не нужна
      cmp   ah,35h      ;поле пишущей машинки?
      ja    notrans     ;нет, трансляция не нужна
      cmp   al,127      ;код > 127?
      ja    notrans     ;да, трансляция не нужна
      cmp   al,32       ;код < 32?
      jbe   notrans     ;да, трансляция не нужна
      add   bl,al       ;вычислим адрес байта
      adc   bh,0        ;откорректируем адрес
      sub   bx,32       ;пропустим младшие коды
      mov   al,cs:[bx]  ;новый код
      xor   ah,ah       ;очистим SCAN
notrans:                ;
      pop   ds          ;восстановим регистры
      pop   bx          ;--//--
      ret               ;возврат из процедуры
translate endp
INSTALL:
      mov   ax,3509h          ;прочитаем адрес INT 09
      int   21h               ;--//--
      mov   ax,es:[bx-2]      ;проверим присутствие
      cmp   ax,cs:presence    ;программы в памяти
      jz    already           ;уже загружено, выход
      mov   cs:old9o,bx       ;модифицируем код
      mov   cs:old9s,es       ;--//--
      mov   ax,3516h          ;прочитаем адрес INT 16
      int   21h               ;--//--
      mov   cs:old16o,bx      ;модифицируем код
      mov   cs:old16s,es      ;--//--
      mov   ax,cs             ;подготовим сегменты
      mov   ds,ax             ;--//--
      cli                     ;критический участок
      lea   dx,INT09          ;адрес подпрограммы
      mov   ax,2509h          ;функция установки вектора
      int   21h               ;установим вектор INT 9
      lea   dx,INT16          ;адрес подпрограммы
      mov   ax,2516h          ;функция установки вектора
      int   21h               ;установим вектор INT 16
      sti                     ;конец критического участка
      lea   dx,INSTALL        ;вычислим размер резидентной части
      mov   cl,4              ;--//--
      shr   dx,cl             ;--//--
      add   dx,20             ;добавим размер PSP и еще 160 байт
      mov   ax,3100h          ;завершим работу,
      int   21h               ;оставив резидентную часть в памяти
already:
      push  cs                ;подготовим сегменты
      pop   ds                ;--//--
      lea   dx,loaded         ;адрес сообщения об ошибке
      mov   ax,0900h          ;функция вывода строки
      int   21h               ;выведем сообщение об ошибке
      mov   ax,4C01h          ;и завершим работу
      int   21h               ;с кодом ошибки, равным 1
loaded db   'Программа KBR уже загружена',10,13,'$'
CODE  ENDS
END   INSTALL                                        Конец ◄
```

Текст программы EGAGA

```
;EGAGA.ASM - загрузчик кириллических знакогенераторов EGA.
;Матрицы знакогенераторов 8x8 и 8x14 содержатся в файлах
;FONT8X8.INC и FONT8X14.INC. Загрузка знакогенераторов
;производится при смене режима (функция 0). Особого внимания
;требуют функции знакогенератора (функции 11h) - вместо адреса
;шрифта в ПЗУ необходимо подставлять адрес шрифта в программе.
pushrs MACRO                    ;;сохранить регистры,
       push   ax                ;;используемые для загрузки
       push   bx                ;;шрифта
       push   cx                ;;
       push   dx                ;;
       push   es                ;;
       push   bp                ;;
       ENDM
poprs  MACRO                    ;;восстановить регистры,
       push   bp                ;;используемые для загрузки
       push   es                ;;шрифта
       push   dx                ;;
       push   cx                ;;
       push   bx                ;;
       push   ax                ;;
       ENDM
 CODE SEGMENT WORD
        ASSUME CS:CODE
 clstoggle db  0                ;флаг очистки экрана
           even
 presence dw  1234h             ;индикатор присутствия
 myint10 LABEL BYTE             ;точка входа функций INT 10h
         or      ah,ah          ;функция смены режима?
         jz      change_mode    ;да, изменяем режим
         cmp     ah,11h         ;функции знакогенератора?
         jz      chargen        ;да, см. ниже.
 jmprom: db      0EAh           ;первый байт инструкции JMP
 old10   label   dword          ;метка двойного слова
```

Продолжение

► *Продолжение*

```
oldoffs dw      ?         ;сюда при загрузке занесем смещение,
oldseg  dw      ?         ;а сюда сегмент программы BIOS INT 10
change_mode:
        mov     cs:clstoggle,00h ;обнулим переключатель
        test    al,80h           ;очистить экран?
        jz      c1               ;да, очищаем экран
        mov     cs:clstoggle,0FFh;установим переключатель
        and     al,7Fh           ;очистим старший бит
c1:     cmp     al,3             ;проверка текстовых режимов 0..3
        jbe     sett8x14         ;да, установить шрифт 8x14
        cmp     al,7             ;монохромный режим?
        jz      sett8x14         ;да, установить шрифт 8x14
        cmp     al,0Eh           ;режим 8x8?
        jbe     setg8x8          ;установить граф. шрифт 8x8
        cmp     al,10h           ;расширенные режимы EGA?
        jbe     setg8x14         ;установить граф. шрифт 8x14
        test    cs:clstoggle,0FFh;проверим переключатель
        jz      c2               ;очистка, см. ниже
        or      al,80h           ;без очистки
c2:     jmp     jmprom           ;вызов BIOS
chargen:
        cmp     al,30h           ;запросы информации EGA info?
        jz      info             ;да, см. ниже
        cmp     al,02            ;загрузка шрифта из ПЗУ?
        jnz     rr               ;нет, см ниже
        jmp     seta8x8          ;загрузить шрифт
rr:     cmp     al,23h           ;необходимо сменить шрифт?
        jz      nextchck         ;да, проверяем дальше
        cmp     al,12h           ;загрузить шрифт 8x8 из ПЗУ?
        jnz     jmprom           ;нет, вызов INT 10
        jmp     sett8x8          ;загрузим шрифт текст. режима
nextchck:
        cmp     bl,3             ;загрузка шрифта из ПЗУ?
        jnz     jmprom           ;нет, вызов BIOS
        jmp     seta8x8          ;да, загрузим шрифт из программ
info:   cmp     bh,2             ;адрес шрифта 8x14 из ПЗУ?
        je      give8x14         ;см. ниже
        cmp     bh,3             ;адрес шрифта 8X8 из ПЗУ?
        je      give8x8          ;см.ниже
        cmp     bh,4             ;адрес верхней половины
                                 ;таблицы шрифта 8x8 из ПЗУ?
        je      give8x8top       ;см.ниже
        jmp     jmprom           ;вызов BIOS
sett8x14:                        ;загрузка текстового шрифта 8x14
        test    cs:clstoggle,0FFh;проверим флаг очистки
        jz      s1               ;очистка нужна
        or      al,80h           ;выставим бит "без очистки"
s1:     pushf                    ;для инструкции IRET
        call    cs:[old10]       ;вызов BIOS INT 10
        pushrs                   ;сохраним регистры
        mov     ax,1100h         ;загрузка шрифта
        push    cs               ;подготовим сегмент
        pop     es               ;--//--
        lea     bp,font8x14      ;адрес шрифта 8x14
        mov     cx,256           ;вся таблица
        mov     dx,0             ;начальный символ
        mov     bx,0E00h         ;14 байт/символ в блок 0
        pushf                    ;для инструкции IRET
        call    cs:[old10]       ;вызов BIOS INT 10
        poprs                    ;восстановим регистры
        iret                     ;возврат из прерывания
setg8x14:                        ;загрузка граф. шрифта 8x14
        jmp     short mysetg     ;см. ниже
setg8x8:                         ;загрузка граф. шрифта 8x8
        jmp     short mysetg8    ;см. ниже
give8x14:                        ;эти фрагменты вместо
        pushf                    ; адреса знакогенератора ПЗУ
        call    cs:[old10]       ; подставляют адрес знако-
        push    cs               ; генератора в программе EGAGA
        pop     es               ;
        lea     bp,font8x14      ;адрес шрифта 8x14
        iret                     ;
give8x8:                         ;
        pushf                    ;
        call    cs:[old10]       ;
        push    cs               ;
        pop     es               ;
        lea     bp,font8x8       ;адрес шрифта 8x8
        iret                     ;
give8x8top:                      ;
        pushf                    ;
        call    cs:[old10]       ;
        push    cs               ;
        pop     es               ;
        lea     bp,font8x8       ;адрес шрифта 8x8
        add     bp,128*8         ;плюс половина размера таблицы
        iret                     ;
mysetg:                          ;загрузка граф. шрифта 8x14
        pushf                    ;
        call    cs:[old10]       ;
        pushrs                   ;
        mov     ax,1121h         ;загрузка графического шрифта
        push    cs               ;
        pop     es               ;
        lea     bp,font8x14      ;адрес шрифта 8x14
        mov     cx,14            ;14 байт/символ
        mov     bl,25            ;25 строк
        pushf                    ;
        call    cs:[old10]       ;
        poprs                    ;
        iret                     ;
mysetg8:                         ;загрузка граф. шрифта 8x8
        pushf                    ;
        call    cs:[old10]       ;
        pushrs                   ;
        mov     ax,1121h         ;загрузка графического шрифта
        push    cs               ;
        pop     es               ;
```

Продолжение ►

► *Продолжение*

```
        lea     bp,font8x8       ;адрес шрифта 8x8
        mov     cx,8             ;8 байт/символ
        mov     bl,25            ;25 строк
        pushf                    ;
        call    cs:[old10]       ;
        poprs                    ;
        iret                     ;
seta8x8:                         ;загрузка графического шрифта
        pushrs                   ; 8x8 для 43-строчного режима
        mov     ax,1121h         ;загрузка графического шрифта
        push    cs               ;
        pop     es               ;
        lea     bp,font8x8       ;
        mov     cx,8             ;8 байт/символ
        mov     bx,3             ;43 строки
        pushf                    ;
        call    cs:[old10]       ;
        poprs                    ;
        iret                     ;
sett8x8:                         ;загрузка текстового шрифта 8x8
        pushrs                   ; в т.ч. для 43-строчного режима
        mov     ax,1110h         ;загрузка текстового шрифта
        push    cs               ;
        pop     es               ;
        lea     bp,font8x8       ;
        mov     cx,100h          ;
        mov     bx,800h          ;основной шрифт 8x8
        mov     dx,0             ;
        pushf                    ;
        call    cs:[old10]       ;
        poprs                    ;
        iret                     ;
font8x8 LABEL BYTE               ;
INCLUDE FONT8X8.INC  ;файл содержит матрицы знакогенератора 8x8
font8x14 LABEL BYTE              ;
INCLUDE FONT8X14.INC ;файл содержит матрицы знакогенератора 8x14
install:                         ;
        mov     ax,3510h         ;прочитаем вектор INT 10
        int     21h              ;--//--
        mov     ax,es:[bx-2]     ;проверим присутствие
        cmp     ax,presence      ;--//--
        jz      already          ;уже загружено, выход
        mov     cs:oldoffs,bx    ;модифицируем код
        mov     cs:oldseg,es     ;--//--
        push    cs               ;подготовим сегмент
        pop     ds               ;--//--
        lea     dx,myint10       ;и смещение
        mov     ax,2510h         ;установим вектор INT 10h
        int     21h              ;--//--
        lea     dx,font8x14      ;установим адрес шрифта 8x14
        mov     ax,2544h         ;для вектора INT 44h
        int     21h              ;--//--
        lea     dx,font8x8       ;установим адрес старшей
        add     dx,128*8         ;половины таблицы шрифта 8x8
        mov     ax,251Fh         ;для вектора 1Fh
        int     21h              ;--//--
        mov     ax,03h           ;очистим экран
        int     10h              ;в текстовом режиме
        lea     dx,install       ;определим размер резидентной части
        mov     cl,4             ;--//--
        shr     dx,cl            ;--//--
        add     dx,11h           ;добавим размер PSP
        mov     ax,3100h         ;завершим работу, оставив
        int     21h              ;резидентную часть
already:                         ;
        push    cs               ;подготовить сегмент
        pop     ds               ;сообщения об ошибке
        lea     dx,loaded        ;адрес сообщения
        mov     ax,0900h         ;выведем сообщение об ошибке
        int     21h              ;--//--
        mov     ax,4C01h         ;закончим работу
        int     21h              ;с кодом ошибки 1
loaded  db      'Программа EGAGA уже загружена',10,13,'$'
CODE ENDS
END install
```

Конец ◄

Рекомендуется включение вызовов этих программ в командный файл начальной загрузки системы AUTOEXEC.BAT, поскольку обе программы (и EGAGA, и KBR) должны загружаться до загрузки остальных резидентных программ. Ввиду небольшого объема, а также указанного порядка загрузки программы не имеют средств освобождения занимаемой ими памяти и не могут быть удалены иначе, чем перезагрузкой системы. Двухлетний опыт использования этих программ подтверждает правильность принятого подхода.

Коммуникация между программами

ХАРДИН БРОДЕРЗ

Учитесь писать программы на языке ассемблера, умеющие обмениваться сообщениями.

Эта статья открывается проблемой, с которой мне пришлось столкнуться при написании прикладной программы. Данная программа должна была иметь доступ к описанию конфигурации системы, задаваемому пользователем и включающему, например, команды инициализации принтера, тип экрана, имена выбранного дисковода и области файлов и т. д. Мне хотелось хранить информацию в дисковом файле, но где?

Следовало учесть и другие факторы. Пользователю может понадобиться вызывать одну и ту же программу, находясь в различных областях файлов, и при этом в каждой из них сохранять различные наборы файлов данных. Мне бы не хотелось требовать от него создания множества копий инициализирующего файла. Кроме того, не хотелось задавать и конкретное имя файла в конкретной области файлов на конкретном дисководе.

Ответ пришел после решения нескольких других проблем программирования. Как сделать резидентную программу, которая сообщает свое местоположение другим программам? Как передать результаты исполнения одной программы следующей за ней? Или, в более общей постановке вопроса, как обеспечить коммуникацию между программами?

Говоря о *коммуникации между программами*, мы имеем в виду как краткосрочную (в течение одного сеанса работы за ЭВМ), так и долгосрочную коммуникацию (когда результаты ожидают извлечения дни и недели). При этом мы получим программы на языке ассемблера, которые повторяют некоторые из функций, автоматически реализуемых в некоторых языках программирования высокого уровня, таких как Quick Basic и Си.

КАНАЛЫ КОММУНИКАЦИИ

Программы могут общаться друг с другом по меньшей мере шестью способами. Мы рассмотрим четыре из них, а остальные два заслуживают краткого упоминания.

В первом, пожалуй, наиболее распространенном способе коммуникации используется запись данных в файлы, которые могут быть прочитаны другими программами. Менее известный способ состоит в том, что программа оставляет в таблице прерываний ЭВМ указатели на область памяти, занимаемую данными. Этот прием широко используется процедурами системы BIOS при обслуживании различных периферийных устройств.

Чтобы понять способы коммуникации между программами, надо иметь представление о том, как операционная система MS-DOS загружает, исполняет и завершает программы. Начнем с процесса загрузки.

НАЧАЛЬНАЯ ЗАГРУЗКА ОПЕРАЦИОННОЙ СИСТЕМЫ

Выполняя процедуру начальной загрузки, ЭВМ считывает с диска операционную систему MS-DOS, затем файл CONFIG.SYS (если таковой имеется) и загружает программный интерфейс пользователя, называемый также оболочкой. Обычно такой оболочкой служит программа COMMAND.COM, но при желании можно указать в файле CONFIG.SYS имя другой оболочки.

Вся коммуникация пользователя с операционной системой MS-DOS, включая исполнение программ и командных файлов, чтение каталогов областей файлов и манипулирование файлами, в действительности исполняется командами, обрабатываемыми содержимым файла COMMAND.COM. Пользователь никогда не общается непосредственно с операционной системой. Программа COMMAND.COM генерирует знакомое приглашение к вводу A>, ожидает ввода команд пользователя, считывает и исполняет командные файлы и обращается к операционной системе MS-DOS для вызова и исполнения программых файлов.

Программа COMMAND.COM очень похожа на обычную программу типа .COM. За исключением необычного способа загрузки в память, она ведет себя подобно любой другой программе, и ее возможности доступны другим программам типа .EXE или .COM. Естественно считать программу COMMAND.COM частью операционной системы MS-DOS, но полезнее рассматривать ее как наиболее популярную из всех написанных прикладных программ. Руководство для пользователя по операционной системе MS-DOS в действительности является справочником по функциям программы COMMAND.COM и других служебных программ, а не по собственно операционной системе.

Если программа COMMAND.COM может загружать и исполнять программы, передавать им информацию и интерпретировать результаты их исполнения, то наши программы могут делать то же самое. В терминах руководств по MS-DOS любая программа, вызывающая другую, называется *материнской*, а вызванная программа – *дочерней*. Большинство прикладных программ является дочерними для программы COMMAND.COM, которая является материнской для любой другой программы и единственной, не имеющей материнской программы.

Когда пользователь (или командный файл) предлагает программе COMMAND.COM вызвать программу, то COMMAND.COM интерпретирует этот запрос, находит файл и вызывает служебную процедуру операционной системы MS-DOS с номером 4BH (где суффикс H означает шестнадцатеричную запись) для загрузки и исполнения этой программы. В ответ MS-DOS обращается к своей системе управления памятью, которая создает заголовок программы, называемый *префиксом сегмента программы* (ПСП, в оригинале PSP – program segment prefix), определяет, является ли программа файлом типа .EXE или .COM, загружает программу в память, обрабатывает ссылки на сегменты (только для файлов типа .EXE) и передает управление первой команде программы.

Операционная система MS-DOS всегда инициирует регистры DS и ES как указатели на сегментную

НЕ ЗАПУТАЙТЕСЬ В ВЕРСИЯХ МАКРОАССЕМБЛЕРА MASM

В последней версии макроассемблера MASM фирмы Microsoft (MASM 5.0) предусмотрена сокращенная запись для определения сегментов и групп сегментов в программе. В этой статье будет использована именно эта запись. Если у вас нет макроассемблера MASM версии 5.0 или более поздней, то вам будет нетрудно раскрыть сокращенную запись, используя приведенную ниже процедуру. Вначале объясним новые форматы, а затем покажем строки, выполняющие эквивалентную функцию в более старых версиях макроассемблера MASM.

Псевдооператор .MODEL SMALL означает, что используется малая модель распределения памяти (менее 64 Кбайт для команд и 64 Кбайт для данных). Кроме того, он задает имена для сегментов команд, стека и данных, группу сегментов с именем DGROUP и соответствующие операторы ASSUME. Замените этот псевдооператор на следующие строки:

```
_TEXT  SEGMENT BYTE PUBLIC 'CODE'
_TEXT  ENDS

_DATA  SEGMENT WORD PUBLIC 'DATA'
_DATA  ENDS

CONST  SEGMENT WORD PUBLIC 'CONST'
CONST  ENDS

_BSS  SEGMENT WORD PUBLIC 'BSS'
_BSS  ENDS

STACK  SEGMENT PARA STACK 'STACK'
STACK  ENDS

DGROUP  GROUP CONST,_BSS,_DATA, STACK

ASSUME  CS: _TEXT,DS:DGROUP,SS:DGROUP,ES:DGROUP
```

Псевдооператор .STACK создает сегмент стека и занимает для него 1 Кбайт памяти по умолчанию. Объем занимаемой памяти можно изменить, задав его параметром в псевдооператоре .STACK. Например, псевдооператор

```
.STACK 100h
```

создает стек размером в 100h (256) байт. Этот псевдооператор заменяется на следующие строки:

```
STACK  SEGMENT
        db  400h dup (?)
STACK  ENDS
```

Псевдооператор .DATA открывает сегмент данных. Сегмент завершается автоматически, как только будет открыт другой сегмент (в программах настоящей статьи это происходит при указании псевдооператора .CODE). Он заменяется на следующие строки:

```
_DATA  SEGMENT
 (определения данных)
_DATA  ENDS
```

Псевдооператор .CODE открывает сегмент команд, который завершается автоматически, как только будет открыт другой сегмент. Он заменяется на следующие строки:

```
_TEXT  SEGMENT
 (команды программы)
_TEXT  ENDS
```

Команда mov ax, @data загружает в регистр AX сегментную часть адреса сегмента данных. Она заменяется на команду

```
mov ax, seg _DATA
```

Новые операторы определения сегментов не только сокращают набираемый текст программы, но еще и обеспечивают совместимость процедур на языке ассемблера с программами на языках высокого уровня, откомпилированными с помощью Microsoft C, Quick C, Quick Basic 4.0, Fortran и MS-Pascal.

часть адреса ПСП программы (ПСП содержит 256 байт информации, которая доступна как операционной системе MS-DOS, так и прикладной программе). Информация заносится в ПСП как операционной системой MS-DOS, так и вызывающей программой, которой обычно является COMMAND.COM. Просматривая свой ПСП, программа может получить информацию от своей материнской программы.

ХВОСТ КОМАНДЫ

После того как вы набрали команду в ответ на приглашение A>, программа COMMAND.COM разбивает ее на отдельные поля. Первое набранное слово рассматривается как имя файла, и COMMAND.COM ищет файл с тем же именем и расширением .EXE, .COM или .BAT, чтобы вызвать его для исполнения. Если вы включили в командную строку символы переадресации ввода-вывода (>, >>, < или |), то COMMAND.COM вначале выполняет переадресацию, а уж затем вызывает вашу программу. Вся остальная часть текста команды передается программе без изменения. Эта дополнительная информация называется *хвостом команды* и может содержать все, что требуется программе, например имя файла или дополнительные параметры. Если программа не рассчитывает получить эту информацию, то она будет проигнорирована.

Программа извлекает хвост команды из ПСП, начиная с относительного адреса 80H. Хвост закодирован в формате LSTRING, названном так потому, что первый байт указывает длину остальной части строки (Length + STRING). Если этот первый байт содержит нуль, то командный хвост пуст. Поскольку последний адрес ПСП равен 0FFH, то максимальная длина хвоста команды равна 7FH (127) байт. (Правда, программа COMMAND.COM не воспримет команду такой длины.)

Показанная на рис. 1 программа демонстрирует один из способов получения хвоста команды. Как и в остальных программах этой статьи, в ней использована сокращенная запись определения сегментов, используемая в макроассемблере MASM 5.0 фирмы Microsoft. Если у вас более старая версия макроассемблера MASM, то см. вставку "Не запутайтесь в версиях макро-

ассемблера MASM", где объясняется смысл сокращенной записи.

Эта программа прежде всего проверяет байт длины хвоста, извлекаемый из ПСП. Если он нулевой, то программа сообщает, что у команды нет хвоста, и завершает свою работу. В противном случае она изображает на экране хвост команды и сообщает об успешном выполнении работы.

Основная цель данной программы – показать, какую именно часть командной строки программа COMMAND.COM передает вашей программе. После трансляции программы попробуйте исполнить ее, задавая различные варианты переадресации ввода-вывода и различные виды текста. Например, поочередно наберите следующие пять строк:

```
CMDLINE This is a test
CMDLINE This >is a test
CMDLINE This /is /a /test
CMDLINE This is <a test
CMDLINE This |is a test
```

После того как будет введена первая строка, вы должны получить на экране сообщение "This is a test". После ввода второй строки будет открыт файл с именем IS и в него будет записана строка "This a test" (содержимое этого файла можно изобразить на экране с помощью команды type). После ввода третьей строки, как и первой, на экране будет изображен хвост команды. Ввод четвертой строки вообще не повлечет за собой исполнения программы, поскольку программа COMMAND.COM не сможет открыть файл с именем A в качестве входа для программы CMDLINE. За вводом пятой строки последует сообщение об ошибке, так как программа COMMAND.COM попытается переадресовать вывод программы CMDLINE несуществующей программе IS.

Хотя из программы на рис. 1 этого не видно, но COMMAND.COM еще и выделяет первые два слова хвоста команды, разбирает их как если бы они означали имена файлов, а затем помещает результаты в подготовленные для последующего открытия блоки управления файлами (БУФ, в оригинале FCB – file control block), расположенные со смещениями 5CH и 6CH от начала ПСП. Кроме того, буквы, которыми написаны эти первые два слова хвоста команды, преобразуются из строч-

Рис. 1. Демонстрация одного из способов чтения из программы на языке ассемблера командной строки операционной системы MS-DOS

```
comment !
   Эта программа показывает, как считать командную строку
   из программы на языке ассемблера. Для компиляции используйте
   MASM 5.0. Информацию о преобразовании директив задания
   сегментов для работы с предыдущими версиями MASM см. на
   вставке

   Сохранить как CMDLINE.ASM
   Компиляция:   MASM CMDLINE;
                 LINK CMDLINE;
       !

stdout  equ     1
stderr  equ     2

        .MODEL SMALL
        .STACK

        .DATA
error$  db      10,13,'Вывод сообщений об ошибках на stdout'
lerror  equ     $-error$
nolin$  db      10,13,'Командная строка пуста'
lnolin  equ     $-nolin$
okay$   db      10,13,'Программа выполнена успешно'
lokay   equ     $-okay$

        .CODE
start:
        mov     ax,@data                ;Установить DS
        mov     ds,ax                   ;  DS==> область данных
        mov     bx,80h                  ;Смещение байта, содержащего
                                        ; длину строки
        test    byte ptr es:[bx],-1     ;Что-нибудь есть?
        jz      no_line                 ;Нет -- сообщение о пустой строке

        mov     dx,81h                  ;DX==> командная строка
        mov     cl,es:[bx]              ;Загрузить длину строки в cx
        mov     ch,0                    ;Должна быть меньше 256
        push    ds                      ;Сохранить DS
        push    es                      ;Загрузить сегмент ПСП
        pop     ds                      ; в DS
        mov     bx,stdout               ;Номер стандартного устройства вывода
        mov     ah,40h                  ;Задать вывод в файл/на устройство
        int     21h                     ;Вызвать функцию DOS
        pop     ds                      ;Восстановить указатель данных
        jc      bad_write               ;Перейти при ошибке (флаг переноса CY
                                        ; установлен)
        cmp     al,cl                   ;Все байты выведены?
        je      success                 ;Да -- перейти

bad_write:
        lea     dx,error$               ;DS:DX==> сообщение об ошибке
        mov     cx,lerror               ;CX = длина сообщения
        xlat    bx                      ;Извлечь соответствующую цифру
        mov     [di],al                 ;Сохранить ее
        inc     di                      ;Указать на следующий пробел
        mov     al,ah                   ;Восстановить преобразуемый байт
        and     al,0fh                  ;Выделить младшую половину байта
        xlat    bx                      ;Извлечь соответствующую цифру
        mov     [di],al                 ;Сохранить ее
        ret

        end     start
```

Конец ◄

Рис. 2. Демонстрация способа извлечения ключей из командной строки операционной системы MS-DOS (например, /P)

```
comment !
   Эта программа показывает один из способов выделения однобуквенных
   ключей из командной строки. Она использует переменную размером в
   слово для сохранения состояния 16 ключей от /A до /P. Программа
   рассматривает каждый ключ как флаг.

   Сохранить как SWITCH.ASM
   Компиляция:   MASM SWITCH;
                 LINK SWITCH;
       !

stdout  equ     1

        .MODEL SMALL
        .STACK

        .DATA

swtch   dw      0                       ;Дайте другое значение, если
                                        ; по умолчанию некоторые ключи
                                        ; должны быть "включены"
error$  db      10,13,'Обнаружен ошибочный ключ'
lerror  equ     $-error$

rpt$    db      10,13,'Текущий символ-признак ключа = '
swtchch db      ?                       ;Место для символа-признака ключа
        db      10,13,'Установлены следующие ключи:
aswtch  db      'ABCDEFGHIJKLMNOP'
lrpt    equ     $-rpt$

        .CODE

start:  mov     ax,@data                ;Установить DS
        mov     ds,ax                   ;  DS==> область данных
        cld                             ;Установить флаг направления
                                        ; обработки строк
        mov     ax,3700h                ;Задать чтение текущего признака ключа
        int     21h                     ;Вызвать функцию DOS
```

Продолжение ►

► *Продолжение*

```
        mov     swtchch,dl              ;Сохранить признак ключа
        mov     al,dl                   ; и поместить в AL для поиска
        mov     cl,es:[80h]             ;Извлечь длину командной строки
        sub     ch,ch                   ;CX = длина командной строки
        test    cx,cx                   ;Строка не пуста?
        jz      report                  ;Пуста -- перейти
        mov     di,81h                  ;Начало командной строки
        mov     bx,swtch                ;Слово состояния ключей

inloop: repne   scasb                   ;Найти очередной ключ
        jcxz    report                  ;Если ключей больше нет, выйти
        push    cx                      ;Иначе сохранить CX
        push    ax                      ; и признак ключа
        mov     al,es:[di]              ;Извлечь ключ
        and     al,0dfh                 ;Преобразовать в прописную букву
        jmp     short err_write         ;Изобразить сообщение

no_line:
        lea     dx,nolin$               ;DS:DX==> сообщение об ошибке
        mov     cx,lnolin               ;CX = длина сообщения
        jmp     short err_write         ;Изобразить сообщение

success:
        lea     dx,okay$                ;DS:DX==> сообщение об успехе
        mov     cx,lokay                ;CX = длина сообщения

err_write:
        mov     bx,stderr               ;Номер стандартного устройства вывода
                                        ; сообщений об ошибках
        mov     ah,40h                  ;Задать вывод в файл/на устройство
        int     21h                     ;Вызвать функцию DOS
        mov     ax,4c00h                ;Выйти -- ошибок нет
        int     21h                     ;Вернуться в DOS

        end     start
```

Конец ◄

Рис. 3. Программа считывает содержимое среды, т. е. области памяти, которая создается программой COMMAND.COM для каждой вызываемой ею программы

```
comment !
   Эта программа показывает, как читать записи, содержащиеся в
   среде. При работе с MS-DOS версии 3.0 или более поздней версии
   она дополнительно изображает свое имя.

   Сохранить как ENVIRON.ASM
   Компиляция:   MASM ENVIRON;
                 LINK ENVIRON;
!

stdout  equ     1

        .MODEL SMALL
        .STACK

        .CODE
start:  cld                             ;Установить флаг направления
                                        ; обработки строк
        mov     ds,es:[2ch]             ;Установить DS, ES ==>
        mov     es,es:[2ch]             ; среда
        mov     di,0                    ;Индекс данных, указываемых ES
        mov     cx,7fffh                ;Максимальный размер среды
        mov     bx,stdout               ;Номер стандартного устройства вывода
lp:     test    byte ptr es:[di],-1     ;Конец списка?
        jz      cmd_nam                 ;Да -- перейти
        sub     al,al                   ;AL = 0
        mov     dx,di                   ;Поместить в DX адрес начала
        repne   scasb                   ;Сравнить с нулем
        push    cx                      ;Сохранить счетчик
        mov     cx,di                   ;Взять адрес нулевого байта + 1
        sub     cx,dx                   ;CX = длина строки
        mov     ah,40h                  ;Задать вывод в файл/на устройство
        int     21h                     ;Вызвать функцию DOS
        mov     ah,02h                  ;Изобразить один символ
        mov     dl,10                   ; прогона строки,
        int     21h
        mov     ah,02h                  ; а также символ
        mov     dl,13                   ; возврата к началу строки
        int     21h
        pop     cx                      ;Восстановить счетчик байтов
        jmp     lp                      ;Взять следующую строку
cmd_nam:
        mov     ah,30h                  ;Задать чтение номера версии
        int     21h                     ;Вызвать функцию DOS
        cmp     al,3                    ;Версия 3.x?
        jb      exit                    ;Нет -- перейти
        inc     di                      ;Иначе ES:DI==> 0001 флаг
        cmp     word ptr es:[di],1      ;Здесь имя программы?
        jne     exit                    ;Нет -- перейти
        add     di,2                    ;DI==> имя программы
        mov     dx,di                   ;Сохранить начальный адрес
        sub     al,al                   ;AL = 0
        mov     cx,-1                   ;Повторять цикл, пока не обнаружен
                                        ; конец
        sub     al,'A'-1                ;Преобразовать в число 1-16
        cmp     al,16                   ;Находится в диапазоне?
        ja      error                   ;Нет -- выйти из цикла
        mov     cl,al                   ;Сохранить число в CL
        sub     ax,ax                   ;AX = 0
        stc                             ;Установить флаг переноса
        rcl     ax,cl                   ;Поместить флаг в AX
```

Продолжение ►

ных в прописные, после чего данные слова помещаются в ПСП со смещениями 5DH и 6DH. Знание этого факта иногда позволяет упростить разбор хвоста команды.

ЧТЕНИЕ КЛЮЧЕЙ

Многие программы рассчитывают обнаружить в хвосте команды перечень ключей наподобие /P и /W, которые используются в команде листинга каталога файлов DIR. Программа на рис. 2 показывает один из путей чтения и разбора таких ключей.

Эта программа имитирует ситуацию, когда допускается задавать до 16 ключей с именами от /A до /P. (Вы можете инициировать переменную размером в слово так, чтобы по умолчанию определенные ключи были в положении "включен".) Например, если вы вызовете эту программу с помощью команды

```
SWITCH /A /B
```

то она сообщит, что ключи A и B "включены". Вначале программа запрашивает у операционной системы MS-DOS текущий символ-признак ключа. В некоторых версиях MS-DOS 2.x можно изменить этот символ (по умолчанию – косая черта /), указав соответствующую команду в файле CONFIG.SYS. Эта редко используемая возможность позволяет приблизить внешний облик MS-DOS к облику операционной системы UNIX, заменив признак ключа на дефис. К несчастью, эта процедура документирована не во всех версиях MS-DOS, отсутствует в версиях 3.x и обрабатывается не всеми служебными программами. То же самое относится и к служебной процедуре с номером 37H, которая может запросить или переустановить символ-признак ключа. Она никогда не упоминалась в руководствах по PC-DOS, хотя и описана в некоторых технических руководствах по MS-DOS.

Следовательно, манипулирование символом-признаком ключа достаточно проблематично. Некоторые программы игнорируют его. Если вы хотите поэкспериментировать, измените этот символ, для чего надо загрузить его новое значение в регистр DL, поместить значение 3701H в регистр AX и вызвать прерывание 21H (командой INT 21H). Однако если вы готовите программу для распространения, то она должна оставлять этот символ

неизменным, но на всякий случай считывать его значение перед разбором хвоста команды, который может содержать ключи.

Остальная часть программы на рис. 2 достаточно бесхитростна. Программа находит каждое появление признака ключа в хвосте команды, считывает следующий символ и использует его для того, чтобы установить или сбросить соответствующий бит в переменной swtch. Затем она просматривает биты переменной swtch. Обнаружив, что бит равен 1, она сообщает, что соответствующий ему ключ "включен".

СРЕДА ПРОГРАММЫ

Программа COMMAND.COM производит манипуляции в области памяти, называемой *средой* и содержащей информацию, которой может воспользоваться любая программа, включая COMMAND.COM. Каждый элемент среды представляет собой слово (текста), обычно записанное строчными буквами, за которым следуют знак равенства и дополнительный текст. Элементы записаны в формате ASCIIZ (текст в ASCII-кодах, заканчивающийся нулевым байтом), а признаком конца списка элементов служит еще один нулевой байт.

Среда содержит, по крайней мере, одну запись: COMSPEC=маршрут к копии программы COMMAND.COM или другой оболочки, вызываемой при начальной загрузке операционной системы. В операционной системе MS-DOS версии 2.x размер среды фиксирован и равен 127 байт; в версиях 3.x размер среды по умолчанию равен 160 байт, но его можно увеличить командой SHELL=, включенной в файл CONFIG.SYS, или запуском другой копии программы COMMAND.COM с ключом /E:. Синтаксис этой команды изменяется от одной частной версии MS-DOS 3.x к другой. Имеются служебные программы, которые увеличивают размер среды в версиях MS-DOS 2.x; пожалуй, из них наиболее удобна программа SETENV, поставляемая с несколькими компиляторами языков высокого уровня фирмы Microsoft. Максимальный размер среды равен 32 Кбайт во всех версиях MS-DOS.

Кроме записи COMSPEC= программа COMMAND.COM заносит в среду альтернативные маршруты поиска программных файлов. Что-

► *Продолжение*

```
        xor     bx,ax           ;Заменить значение выбранного бита
                                ; на противоположное
        mov     swtch,bx        ;Сохранить результат
        pop     ax              ;Восстановить признак ключа
        pop     cx              ; и счетчик
        jmp     inloop          ;Перейти к поиску следующего ключа

error:  lea     dx,error$       ;DS:DX==> сообщение об ошибке
        mov     cx,lerror       ;CX = длина сообщения
        jmp     short exit      ;Перейти

report:
        lea     si,aswtch       ;DS:SI==> ASCII-коды ключей
        clc                     ;Обнулить флаг переноса
        mov     cl,16           ;Число ключей
outlp:  rcr     bx,1            ;Проверить один бит
        jc      out_2           ;Перейти, если ключ "включен"
        mov     byte ptr [si],  ;Иначе заменить букву ключа на пробел
out_2:  inc     si              ;Указать на следующую букву
        loop    outlp           ;Выполнить цикл 16 раз

        lea     dx,rpt$         ;DS:DX==> итоговое сообщение
        mov     cx,lrpt         ;Длина в CX
exit:   mov     bx,stdout       ;Номер стандартного устройства вывода
        mov     ah,40h          ;Задать вывод в файл/на устройство
        int     21h             ;Вызвать функцию DOS
        mov     ax,4c00h        ;Выйти -- ошибок нет
        int     21h             ;Вернуться в DOS

        end     start
```

Конец ◄

Рис. 4. Программа на языке ассемблера считывает содержимое области межпрограммной коммуникации размером 16 байт. Через эту область одна программа может передать данные другой программе

```
comment :
   Эта программа считывает и изображает на экране текущее содержимое
   области межпрограммной коммуникации.

   Сохранить как READIAC.ASM
   Компиляция:   MASM READIAC;
                 LINK READIAC;
        :

stdout  equ     1

        .MODEL SMALL
        .STACK

        .DATA
hexasc  db      '0123456789ABCDEF'
title$  db      10,13,'Содержимое области межпрограммной коммуникации:'
ltitle  equ     $-title$
byte$   db      '  h',10,13
lbyte   equ     $-byte$

        .CODE
start:  mov     ax,@data
        mov     ds,ax           ;Инициировать регистр сегмента данных
        lea     dx,title$       ;DS:DX==> выходная строка
        mov     cx,ltitle       ;Загрузить в счетчик длину строки
        mov     bx,stdout       ;Номер стандартного устройства вывода

        mov     ax,0040h        ;Сегмент информации в ПЗУ BIOS
        mov     es,ax           ;Инициировать ES
        mov     si,0f0h         ;Начало ОМК
        mov     cx,10h          ;Число байтов в ОМК

lp:     mov     al,es:[si]      ;Считать байт
        push    cx              ;Сохранить счетчик
        push    bx              ; и номер устройства вывода
        call    convert         ;Преобразовать байт
        pop     bx              ;Восстановить номер устройства вывода
        lea     dx,byte$        ;DS:DX==> выходная строка
        mov     cx,lbyte        ;Загрузить длину строки в счетчик
        mov     ah,40h          ;Задать вывод в файл/на устройство
        int     21h             ;Вызвать функцию DOS
        pop     cx              ;Восстановить счетчик
        inc     si              ;Указать на следующий байт
        loop    lp              ;Обработать в цикле все байты
        mov     ax,4c00h        ;Выйти -- ошибок нет
        int     21h             ;Вернуться в DOS

convert:
        lea     di,byte$        ;DI==> рабочая область
        lea     bx,hexasc       ;BX==> преобразующая строка
        mov     ah,al           ;Сохранить копию байта
        mov     cl,4            ;Задать число перемещаемых битов
        shr     al,cl           ;Выделить старшую половину байта
        repne   scasb           ;Найти конец данных
        mov     cx,di           ;Получить адрес конца
        sub     cx,dx           ;CX = длина
        mov     bx,stdout       ;Номер стандартного устройства вывода
        mov     ah,40h          ;Задать вывод в файл/на устройство
        int     21h             ;Вызвать функцию DOS
exit:   mov     ax,4c00h        ;Выйти -- ошибок нет
        int     21h             ;Вернуться в DOS

        end     start
```

Конец ◄

бы указать новые или изменить уже записанные маршруты, можно воспользоваться командой path. Вы можете создать в среде новые записи командой set. Например, если вы наберете команду

```
SET abc=xyz
```

а затем наберете команду set без параметров, чтобы посмотреть содержимое среды, то последней записью окажется "ABC=xyz". Команда set, исполняемая программой COMMAND.COM, всегда записывает прописными буквами то, что стоит слева от знака равенства, и оставляет правую часть без изменения.

Двухбайтовая запись со смещением 2CH от начала ПСП содержит значение сегмента для копии среды, созданной для вашей программы. Оригинал среды управляется только программой COMMAND.COM, и только эта программа должна вносить в нее изменения. Каждая дочерняя программа получает копию среды материнской программы или любой другой среды, которую может создать материнская программа. Ни одна дочерняя программа не может изменить среду своей материнской программы, а также свою копию среды, если среда материнской программы изменена.

Многие прикладные программы используют среду для установки своих собственных параметров. Например, программа LINK, поставляемая вместе с операционной системой MS-DOS, просматривает среду в поисках "переменной" LIB=, чтобы определить, в каких каталогах файлов находятся имена библиотечных файлов. Используемая мною программа обработки текстов при каждом вызове считывает 10 различных записей среды для установки различных режимов исполнения.

"Переменные" среды можно использовать в командных файлах, окаймляя имя "переменной" знаками %. Если в среду внесена "переменная" с помощью указанного выше примера команды set, то любое вхождение %ABC% в командном файле будет заменено на xyz. Командные файлы, используемые для установки макроассемблера MASM 5.0, могут служить хорошими примерами использования подобных возможностей.

Из рис. 3 видно, каким образом программа может прочитать свою копию среды. Вначале она считывает из своего ПСП сегментную часть адреса среды, а затем изображает каждую запись среды, пока не обнаружит двух завершающих нулевых байтов. Как показывает эта программа, в операционной системе MS-DOS 3.x полное имя маршрута для исполняемой программы находится непосредственно за концом среды. Этим можно воспользоваться, чтобы выяснить, не переименовал ли пользователь программу, и чтобы определить, на каком дисководе и в какой области файлов находится программа.

ОБЩИЙ ПУЛ БАЙТОВ

Программа, приведенная на рис. 4, показывает последнее место, из которого программа может черпать информацию: 16 байтов, начинающихся с адреса 0040:00F0 (или 0:4F0), зарезервированы как область межпрограммной коммуникации (ОМК, в оригинале IAC – inter-application communication). Любая программа может записать предназначенные для другой программы данные в эту область. Более того, если только первая программа непосредственно не вызвала вторую, ОМК нельзя защитить от изменения любой промежуточной программой.

Показанная на рис. 4 программа читает и изображает на экране содержимое ОМК. Каждый байт ОМК выводится в форме двузначного шестнадцатеричного числа.

То, что лишь немногие прикладные программы пользуются ОМК, отчасти связано с опасениями, что другие программы могут изменить помещенную в ОМК информацию. Однако ОМК может быть достаточно полезна для передачи адреса от одной программы к другой.

Например, у меня есть несколько процедур на машинном языке, которые мне хотелось бы вызывать при работе с интерпретирующим Бейсиком. По-моему, обычные методы доступа к ним: запоминание их в строковых переменных Бейсика или в начале незанятой памяти – не являются удовлетворительными. Вместо этого я помещаю эту процедуру в короткую программу, которая загружает указатель в ОМК. В таком случае любая программа на Бейсике может воспользоваться командой PEEK, чтобы найти эти процедуры и использовать их содержимое для определения внешнего сегмента Бейсика. Затем можно выполнять вызовы процедур, задавая соответствующее смещение в данном сегменте.

Для исполнения приведенных в тексте программ требуются операционная система MS-DOS 2.0 или более поздней версии и макроассемблер MASM фирмы Microsoft.

DOS-интегратор

ОБОЛОЧКА- СПРАВОЧНИК ДЛЯ КОМАНД DOS

И.А.НАСТЕНКО

Среди операционных систем для 16-разрядных компьютеров наибольшее распространение получила система MS-DOS (далее в тексте просто DOS), разработанная фирмой Microsoft для ПК типа IBM PC/XT/AT и с успехом используемая на всех подобных машинах. Ориентироваться в командах операционной системы и правильно их использовать – необходимое условие успешной работы на персональном компьютере.

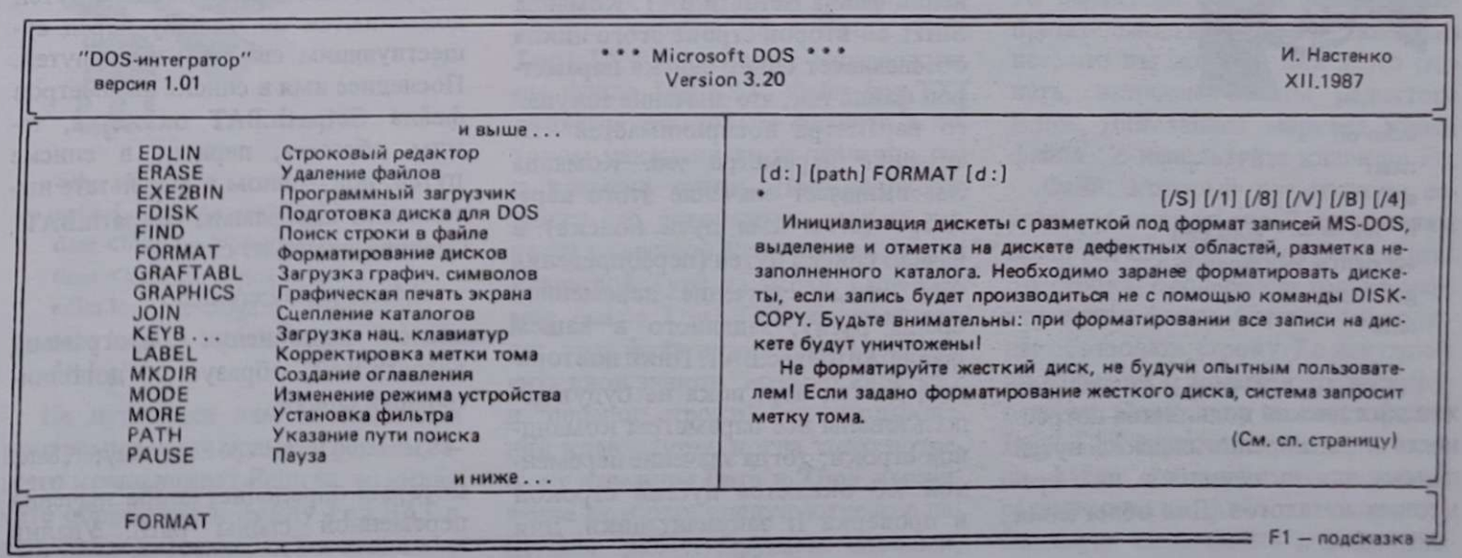
DOS-интегратор позволит вам увидеть на экране все команды DOS. Вы можете получить в компактной форме достаточно полную информацию о той или иной команде, а набрав в командной строке параметры и ключи команды, отправить ее на выполнение. Это позволяет считать DOS-интегратор обучающей программой для новичка и удобным справочником по редко используемым командам для опытного пользователя.

Как показано на рисунке, экран DOS-интегратора имеет два окна и командную строку в нижней части. Левое окно DOS-интегратора служит для просмотра перечня команд DOS с краткими комментариями. Используя клавиши перемещения курсора вверх и вниз, а также Home, End, PgUp, PgDn, можно перемещать подсветку к имени любой из 54 команд DOS. При этом в правом, справочном, окне будут получены информация о типе, формате и назначении команды, описания ключей, а также рекомендации по использованию команды и примеры. Если справочник по команде занимает несколько страниц, вы можете его "листать", используя клавиши "+" и "–". Имеется режим "быстрого поиска" (вызывается нажатием клавиши F3), позволяющий находить нужную команду по первым буквам ее имени.

В нижней, командной, строке DOS-интегратора при выборе команды высвечивается ее имя. Можно набрать параметры и ключи команды и нажатием клавиши Enter отправить ее на выполнение. При этом экран дисплея будет очищен, и на него будет выведен результат действия команды или стандартное диагностическое сообщение. Нажатие любой клавиши после выполнения команды возвратит вас в DOS-интегратор, который хранит до 18 ранее выполненных команд. Используя клавишу F7 для ввода предыдущей команды или клавишу F8 для ввода последующей команды, можно отредактировать одну из ранее выполненных команд и отправить ее на выполнение.

В DOS-интеграторе предусмотрена трехстраничная подсказка, где описаны назначение клавиш DOS-интегратора, правила чтения команд и специальные команды пакетных файлов и файла CONFIG.SYS.

При реализации программы не было проблем с разработкой встроенного редактора или отладчика – эти средства имеются среди утилит DOS и фигурируют в списке команд. Кроме того, имеется возможность очистить командную строку (нажатием клавиши F5), вписать имя любого выполнимого файла и, таким образом, вызвать большую



часть прикладного программного обеспечения непосредственно из DOS-интегратора (если, конечно, эти программы не имеют общих с интегратором точек прерывания).

DOS-интегратор подготовлен и откомпилирован в системе Turbo-Basic и содержит около 1700 операторов. Выполнимый файл занимает около 130 Кбайт (справочники по всем командам загружаются в память одновременно, что позволяет их быстро просматривать). Все полезные функции выполняются непосредственно DOS-утилитами, обращение к которым реализовано с помощью команды shell. DOS-интегратор — это оболочка для команд DOS, посредник между пользователем и командами операционной системы. При разработке DOS-интегратора хорошим примером такого рода программ служил Norton-Inegrator Питера Нортона.

DOS-интегратор предназначен для работы на ПК PC/XT/AT и им подобных с графическим адаптером Hercules, CGA или EGA при объеме памяти не менее 256 Кбайт. Справочники DOS-интегратора содержат информацию, соответствующую командам DOS версии 3.2.

Разрабатываемая мною следующая версия DOS-интегратора будет автоматически настраиваться на версию DOS, установленную на компьютере. Появится также возможность непосредственного обращения к накопителю выполненных команд для их оформления в пакетные файлы или присоединения к меню.

Избранные пакетные файлы

ДЕЙВ РАУЭЛЛ

Поиск путей, ведущих к автоматизации выполнения утомительных или часто повторяющихся заданий с помощью таких скудных средств, как программирование пакетных файлов, может оказаться хорошей проверкой изобретательности. Это занятие напоминает постройку сарая с использованием лишь ограниченного набора инструментов: необходимы тщательное обдумывание и специальные приемы. Тем не менее цель достижима.

Возможно, наилучшим способом постижения искусства создания пакетных файлов является изучение успешного опыта других программистов. Приведенные ниже пакетные файлы представляют творчество читателей журнала PC Resource и публикуются вместе с пояснениями, помогающими практическому использованию пакетных файлов и содержащими замечания о наиболее интересных приемах программирования и возможных вариантах текстов пакетных файлов. Эти файлы можно использовать и без изменения, однако я уверен, что вы найдете собственные варианты публикуемых пакетных файлов.

Setpath.BAT

ЗАДАНИЕ ПУТИ ПОИСКА НА ДИСКЕ

```
echo off
:start
shift
if "%0" = = "" goto end
set path = %0;%path%
goto start
:end
```

Время от времени у владельцев жестких дисков появляется потребность в расширении списков путей поиска для включения в них временных каталогов. Для облегчения решения этой задачи и предназначен пакетный файл Setpath.BAT, с помощью которого можно добавить один или несколько путей к имеющемуся у вас списку путей. Этот файл демонстрирует также, как задать программный цикл, выполняемый однократно для каждого из параметров, вводимых при выполнении файла Setpath.BAT. Команда Shift во второй строке этого цикла обеспечивает сдвиг списка параметров файла так, что значение текущего параметра воспринимается как значение параметра %0. Команда Set включает значение этого параметра (т. е. имя пути поиска) в начало списка путей (переопределяя тем самым значение переменной среды path), заданного в вашем файле Autoexec.BAT. Цикл повторяется до тех пор, пока не будут использованы все параметры командной строки; тогда значение переменной %0 окажется пустой строкой и проверка If закончит цикл. Для использования файла Setpath.BAT вызовите эту программу, добавив список имен путей, разделенных пробелами. Например:

```
setpath c:\text c\letter
```

Знаки точка с запятой, необходимые для разделения имен в команде Path, добавляются при выполнении пакетного файла. Имена путей добавляются по одному перед существующим списком имен путей. Последнее имя в списке параметров файла Setpath.BAT окажется, таким образом, первым в списке путей, полученном в результате выполнения программы Setpath.BAT. Следовательно, из списка

```
c:\dos;c:\utility;c:\batch
```

после выполнения программы Setpath.BAT образуется дополнительный список

```
c:\letter;c:\text;c:\dos;c:\utility;c:\batch
```

который определяет новое значение переменной среды path. Удалив

командуEcho Off в первой строке, можно проследить за процессом выполнения цикла, задав при выполнении файла Setpath.BAT в качестве параметров несколько имен путей.

Когда вы в ответ на приглашение операционной системы DOS вводите имя программы, то DOS ищет ее сначала в текущем каталоге, а затем в других каталогах в том порядке, в каком имена этих каталогов перечислены в списке имен путей. Таким образом, часто используемые подкаталоги лучше располагать в начале этого списка. Вы можете изменить файл Setpath.BAT так, чтобы новые имена путей добавлялись в конец списка, заменив строку с командой Set на строку

```
set path = %path%; %0
```

Можно также ввести сразу несколько путей, разделив их имена знаками точка с запятой (без пробела) так, чтобы весь список воспринимался как значение одного параметра:

```
setpath c:\text;c:\letter
```

Хотя использование в пакетных файлах таких переменных среды DOS, как path, не отражено в документации на версии DOS 2.x, этот прием пригоден для DOS 2.x так же, как и для DOS 3.x. При использовании в пакетных файлах переменная path должна быть заключена между знаками процента (%path%). ■

Log.BAT

УЧЕТ ВРЕМЕНИ РАБОТЫ КОМПЬЮТЕРА

```
echo off
echo Logged on at: >log3.txt
date <log1.txt >>log3.txt
time <log3.txt >>log3.txt
edlin log3.txt <log2.txt >nul
copy log.txt + log3.txt >nul
del log3.*
```

Не нужна ли вам экономичная программа учета времени работы вашего компьютера? Если да, то включите обращение к файлу Log.BAT в ваш файл Autoexec.BAT и запишите на диск, с которого загружается DOS, файлы Log.TXT, Log1.TXT и Log2.TXT и программу Edlin (из набора утилит DOS). После этого время каждого включения компьютера будет записываться на диск.

Эта необычная пакетная программа демонстрирует возможности операционной системы DOS для переназначения ввода-вывода. Для некоторых программ, особенно для программ DOS, вы обычно используете символ "больше" (>), чтобы выходные данные программы, предназначенные для вывода на экран, записать в некоторый файл. Если вы вместо одного знака "больше" напишите два таких знака (>>), то DOS все данные, предназначенные для вывода на экран, допишет в уже существующий файл, а не создаст нового файла для этих данных. Аналогично при вводе данных можно сэкономить на времени ввода с клавиатуры с помощью знака переназначения ввода "меньше" (<), подставив в качестве ожидаемого с клавиатуры текста ответа содержимое некоторого файла.

Рассмотрим, как создается каждый из информационных файлов, а затем выясним, как они объединяются при выполнении программы Log.BAT. Первые два файла, Log.TXT и Log1.TXT, вначале содержат лишь по два байта – символы перевода каретки и перевода строки. Для создания этих файлов введите

```
copy con log.txt
```

При появлении курсора на следующей строке нажмите клавишу "возврат каретки" (Enter), в результате чего курсор перейдет на следующую строку, а в файл запишутся требуемые два символа. Затем для завершения работы нажмите клавиши F6 и Enter. Файл Log1.TXT образуется копированием файла Log.TXT. Файл Log.TXT является текстовым файлом, в котором накапливаются значения дат и времени суток. При необходимости его содержимое можно вывести командой Print на печать или командой Type на экран. Содержимое файла Log1.TXT не изменяется; этот файл нужен, чтобы обеспечить ввод знаков "возврат каретки" и "перевод строки" без использования клавиатуры, когда этого потребуют команды Date и Time, выполнение которых предусмотрено в пакетном файле. Другими словами, файл Log1.TXT используется вместо клавиши Enter. С помощью переназначения вывода файл Log.BAT добавляет выходные данные команд Date и Time во временный файл с именем Log3.TXT.

После выполнения действий, указанных в первых четырех строках файла Log.BAT, временный файл данных Log3.TXT будет содержать строку "Logged on at:" (записанную в файл переназначением вывода команды Echo во второй строке) и выходные данные команд Date и Time.
Например:

```
Logged on it:
Current date is Thu 02-04-1988
Enter new date (mm-dd-yy):
Current time is 14:58:58.09
Enter new time:
```

В пятой строке вызывается Edlin – текстовый редактор DOS, который очищает файл Log3.TXT. Edlin сокращает две строки, содержащие данные о дате и времени, и удаляет две ненужные строки. Полученный файл содержит пустую строку в конце файла и следующий текст:

```
Logged on at:
Date: Thu 02-04-1988
Time: 14:58:58.09
```

В команде Copy в шестой строке используется знак "плюс" (+) для пополнения содержимого файла Log.TXT информацией о дате и времени суток. Последняя строка обеспечивает удаление временных файлов Log3.TXT и Log3.BAK (созданных редактором Edlin).

Файл Log2.TXT содержит такие же символы, которые вы вводили бы с клавиатуры при редактировании временного файла с помощью редактора Edlin. Создание этого файла требует использования самого редактора Edlin и немалой изобретательности. На рис. 1 показано все, что вы должны для этого сделать, включая ответы редактора Edlin. Для записи маркера конца файла ^Z используйте клавишу F6.

Файл, который вы должны создать, показан на рис. 2, однако вам придется изрядно потрудиться, чтобы строка 7 содержала только маркер конца файла. Сначала вы должны образовать строку 7 с ненужной информацией, а затем отредактировать ее. Поскольку в текст файла Log2.TXT включены символы конца файла, то после его создания редактировать файл невозможно, большую часть файла невозможно

Рис. 1

```
c> edlin log2.txt
New file
*i
     1:*2,2rCurrent date is^ZDate:
     2:*4,4rCurrent time is^ZTime:
     3:*5d
     4:*3d
     5:*#i
     6:*
     7:*You must type this line.
     8:*^Z
*7
     7:*You must type this line.
     7:*^Z
*#i
     8:*e
     9:*^Z
*e

C>
```

даже увидеть на экране. Если в процессе создания файла вы сделаете ошибку, всю работу придется повторить с самого начала.

Чтобы во время выполнения программы Log.BAT ненужные сообщения программ Copy и Edlin не выводились на экран, используется переназначение вывода этих программ на фиктивное устройство (>nul). Несуществующее устройство "заглатывает" выводимые данные как "черная дыра".

Рис. 2

```
1: 2,2rCurrent date is^ZDate:
2: 4,4rCurrent time is^ZTime:
3: 5d
4: 3d
5: #i
6:
7: ^Z
8: e
```

Если для пакетных файлов на вашем диске выделен специальный подкаталог, то поместите туда же файлы Log.TXT, Log1.TXT и Log2.TXT. Можно, конечно, выделить для них и отдельный подкаталог, а затем скорректировать файл Log.TXT так, чтобы все обращения к этим файлам включали соответствующее имя пути. Например, вторая строка файла Log.BAT могла бы иметь следующий вид:

```
echo Logged on at: c:\log\log.txt
```

Если существует реальная потребность слежения за временем, то необходимо также записывать время выключения компьютера. Для этого нужно сделать копию файла Log.BAT под именем Logoff.BAT, а затем отредактировать Logoff.BAT так, чтобы вторая строка имела следующий вид:

```
echo Logged off at: >log3.txt
```

Не забудьте выполнить программу Logoff.BAT перед выключением своего компьютера. Если обычно в конце сеанса работы вы делаете копии обновленных файлов для обеспечения их сохранности, то этот процесс вы могли бы включить в процесс выполнения файла Logoff.BAT. ■

В помощь начинающим

Требования к программному обеспечению, необходимому для написания пакетных файлов, весьма умеренны. Все, что вам нужно, – это любой текстовый редактор, с помощью которого можно подготовить текстовые файлы в формате стандарта ASCII. Достаточно, чтобы строки каждого такого файла, выведенного на экран командой Type, заканчивались там, где должны. Для записи файла на диск в формате стандарта ASCII в некоторых текстовых редакторах, например в редакторе Wordstar, для этой цели существует специальная команда. Я обычно пользуюсь резидентной программой Sidekick, так как она позволяет ввести текст пакетного файла, быстро вернуться в DOS для проверки его выполнения, а затем мгновенно вернуть текст файла на экран для редактирования. Система DOS поставляется с текстовым редактором Edlin, с которым также удобно работать, если число строк пакетного файла не превышает 20.

Для небольших, содержащих всего четыре или пять строк, пакетных файлов не нужен и текстовый редактор. Вместо него достаточно команды Copy. При таком способе создания пакетного файла текст вводится с консоли (CON), а выходным файлом является пакетный файл. Фактически каждая строка, вводимая вами с клавиатуры, прочитывается командой Copy и копируется в выходной файл.

Можно проиллюстрировать описанный выше способ на примере создания пакетного файла H.BAT, выполнение которого обеспечивает возврат в корневой каталог независимо от того, какой дисковод и какой каталог были установлены в качестве текущих до вызова программы H.BAT. Для создания файла H.BAT в ответ на приглашение DOS введите с клавиатуры следующую строку:

```
copy con h.bat
```

Затем введите следующие четыре строки, не забывая в конце каждой строки нажимать клавишу "возврат каретки" (Enter):

```
echo off
c:
cd c:\
cls
```

Для исправления ошибок вы можете использовать клавишу "забой" (backspace), но только для вводимой в данный момент строки. После завершения ввода последней строки нажмите клавишу Ctrl-Z или функциональную клавишу F6, после чего нужно нажать еще и клавишу Enter, чтобы отметить конец файла символами ^Z. Ввод последних символов сигнализирует системе DOS, что вы закончили ввод текста. Ответом DOS является сообщение "1 File(s) copied", за которым следует появление символа приглашения DOS. Несмотря на простоту создания пакетных файлов с помощью команды Copy CON, файлы с числом строк более пяти лучше создавать с помощью текстового редактора Edlin или другого текстового редактора, позволяющего проводить изменения в тексте после создания файла.

DATEX.BAT

ЗАПОМИНАНИЕ ДАТЫ


```
echo off
date <date1.txt >nul
date
date <date2.txt >date1.txt
debug date1.txt <date3.txt >nul
```

При изучении программы Log.BAT я нашел решение одной волновавшей меня проблемы: как исключить необходимость повторного ввода даты, если в течение текущего дня она уже вводилась.

Если у вашего ПК отсутствует батарейное питание встроенных часов, то использование файла DATEX.BAT сэкономит вам время при вводе даты с клавиатуры. При перезагрузке DOS или повторном включении компьютера в течение дня вместо ввода даты просто нажмите клавишу "возврат каретки". Однако вводить время все же придется.

Для файла DATEX.BAT применены те же приемы, что и для файла Log.BAT, только для редактирования текста файла вместо редактора Edlin использована программа Debug. Нужны три файла: файл Date1.TXT хранит текущую дату, файл Date2.TXT содержит только символы "возврат каретки" и "перевод строки", а файл Date3.TXT с помощью переназначения ввода обеспечивает инструкции для программы Debug. Файлы Date1.TXT и Date2.TXT создаются точно так же, как и файлы Log.TXT и Log1.TXT в предыдущей программе. Файл Date3.TXT содержит следующие команды программы Debug:

```
rcx
0c
w114
q
```

Для использования файла DATEX.BAT замените в вашем файле Autoexec.BAT команду Date строкой

```
command/c datex
```

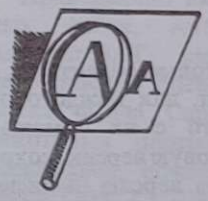
Вы не можете вызвать пакетный файл из другого пакетного файла и ожидать, что управление будет возвращено в вызывающий файл, если не примените показанный прием загрузки временной второй копии файла Command.COM. Однако если вы работаете с DOS 3.3, то можете воспользоваться новой командой Call, которая обеспечивает требуемый возврат управления автоматически:

```
call datex
```

Так же, как и для файла Log.BAT, если для трех файлов с данными используется отдельный подкаталог, удостоверьтесь, что в файле DATEX.BAT указываются полные имена путей. Необходимо также обеспечить доступ к программе Debug. ■

Menu.BAT

ПОМОЩЬ УТОМЛЕННЫМ ГЛАЗАМ



AUTOEXEC.BAT

```
prompt $e[0;62;"MODE 40";13p
prompt $e[0;65;"CLS";13p
prompt $e[0;66;"MODE 80";13p
prompt $e[7m
prompt
cls
h.bat
```

H.BAT

```
echo off
mode 40
cls
type a.txt
```

A.TXT

```
WELCOME
Would you like MS-DOS? Press M, then
<<ENTER>>
Rather have BASIC? Press B, then
<<ENTER>>
```

M.BAT

```
echo off
cls
type m.txt
```

M.TXT

```
In MS-DOS two displays
may be selected:
To select 40-COLUMN, press 4 then
<<ENTER>>
To select 80-COLUMN, press 8 then
<<ENTER>>
If you wish to continue
without changing, press F7
```

4.BAT

```
echo off
mode 40
cls
```

8.BAT

```
echo off
mode 80
cls
```

B.BAT

```
cls
basic k.bas
```

Наиболее интересным проектом, связанным с использованием пакетных файлов, является создание меню для системы с жестким диском. При выполнении файла Autoexec.BAT после необходимых установочных операций делается обращение к пакетному файлу (назовем его Menu.BAT), в котором командой Type на экран выводится содержимое текстового файла, представляющее собой список ваших задач, как список альтернатив для выбора из меню.

Каждая из перечисленных задач выполняется с помощью соответствующего пакетного файла. Например, если для вызова интерпретатора языка Бейсик указывается строка меню, обозначенная буквой A, то пакетный файл с именем A.BAT обеспечивает смену текущего каталога и вызов интерпретатора. В последней строке каждого такого пакетного файла, загружающего программу, содержится вызов файла Menu.BAT, опять выводящего на экран меню. Например, для обращения к интерпретатору языка Бейсик с клавиатуры просто вводится

буква A в ответ на приглашение DOS; после завершения работы интерпретатора на экране снова появляется меню.

Приведенная здесь программа меню является оригинальным вариантом описанной выше идеи. Представление меню в 40-колоночном режиме работы видеоадаптера облегчает восприятие изображения на экране для человека с ослабленным зрением. При выполнении этой программы некоторые функциональные клавиши переопределяются так, что можно быстро переключаться с работы в 80-колоночном режиме на работу в 40-колоночном режиме и обратно.

Для переопределения трех функциональных клавиш с помощью драйвера ANSI.SYS в строках 1–3 файла Autoexec.BAT используются команда Prompt и escape-последовательность. (Проверьте, обеспечивается ли загрузка драйвера ANSI.SYS в файле CONFIG.SYS.) Четвертая (предпоследняя) команда Promt устанавливает обращенный режим работы видеосистемы.

После завершения программы Autoexec.BAT нажатием клавиши F4 можно перейти в 40-колоночный режим, а с помощью клавиши F8 можно восстановить 80-колоночный режим видеосистемы. Нажатием клавиши F7 экран очищается. Чтобы сделать возможным изменение ширины экрана, необходимо обеспечить доступ к команде Mode системы DOS. Вызов программы меню из файла Autoexec.BAT обеспечивается обращением к пакетному файла H.BAT. Меню предлагает выбор из двух возможных вариантов: оставаться в системе DOS или вызвать интерпретатор языка Бейсик. При вводе с клавиатуры буквы M начинает выполняться файл M.BAT и соответствующая программа предлагает вам ввести цифру 4 для установки 40-колоночного режима или цифру 8 для 80-колоночного режима видеосистемы. Такой выбор определяет выполнение файла 4.BAT или 8.BAT. Нажмите клавиши F4 или F8 приводит к таким же результатам. Если вы выбрали работу с интерпретатором языка Бейсик (введя букву B в ответ на приглашение меню), из соответствующего пакетного файла вызывается небольшая программа на языке Бейсик (ее текст не приводится), которая устанавливает 40-колоночный режим и переопределяет несколько функциональных клавиш с помощью оператора Key. Например, нажатие клавиши F10 возвращает управление системе DOS. ■

NOBAK.BAT

УДАЛЕНИЕ ДУБЛИРУЮЩИХ ФАЙЛОВ

```
echo off
if exist *.bak goto delete
echo No BAK files to delete.
goto end
:delete
dir *.bak
echo If you DON'T want to delete the
echo listed file(s), press Ctrl-C, otherwise
pause
del *.bak
echo** File(s) deleted **
:end
```

Многие текстовые редакторы, в том числе Edlin, для каждого отредактированного с их помощью файла создают новую версию, сохраняя предыдущую версию на диске. После долгой работы с большим числом файлов эти дублирующие файлы быстро заполняют свободное пространство диска. Программа NOBAK.BAT служит для удаления дублирующих файлов из текущего каталога; во время работы сначала выдается список этих файлов, чтобы вы точно знали, что вы удаляете.

Команда Pause в строке 9 предоставляет вам возможность изменить свое намерение после просмотра списка файлов каталога. Для прекращения процесса достаточно нажать Ctrl-C. Если ваш редактор использует расширения имен дублирующих файлов, отличные от BAK, сделайте соответствующие изменения в тексте файла NOBAK.BAT.

Вариант файла NOBAK.BAT, который я назвал DL.BAT, позволяет использовать знаки группирования. Так как DL.BAT предупреждает о файлах, которые он собирается удалить, я использую его вместо команды DOS Delete (DEL) каждый раз, когда мне необходимо удалить сразу несколько файлов. Для получения файла DL.BAT подставьте %1 вместо *.BAK везде, где этот набор символов встречается. В частности, обращение вида

```
dl *.bak
```

обеспечивает тот же результат, что и вызов файла NOBAK.BAT. ■

WP.BAT

ЗАПОМИНАНИЕ ПОСЛЕДНЕГО ФАЙЛА

```
echo off
c:
cd \perfwrit
if "" = = "%1" goto noparm
set doc = %1
:noparm
wp %doc%
cd \
cls
```

В разделе "Советы по организации файловой системы на жестком диске" приведен стандартный загрузчик прикладной программы Wordpro.BAT. Файл WP.BAT является вариантом такого загрузчика, который запоминает имя последнего файла, с которым вы работали. При вызове файла WP.BAT задается имя файла данных как параметр командной строки:

```
wp program
```

При выполнении программы WP.BAT это имя становится значением переменной среды %doc%. Если вы продолжаете работу с этим же файлом после перерыва, нет необходимости при вызове программы указывать ее имя, достаточно ввести только WP. В этом случае условие проверки в строке 4 удовлетворяется и управление передается на метку :noparm. Имя вызываемой программы добавляется автоматически программой WP.BAT. Очевидно, при перезагрузке DOS или после выключения компьютера значение переменной %dos% оказывается неопределенным. ■

Об эффективности системы без жесткого диска

При загрузке файлов в оперативную память или поиске файлов в дисковой памяти гибкие диски проигрывают в эффективности жестким дискам. Работа с пакетными файлами подчеркивает этот недостаток гибких дисков. Это связано с тем, что DOS, выполняя команды одной строки пакетного файла, обращается к диску за следующей строкой этого файла.

В системе, имеющей только дисководы для гибких дисков, можно повысить скорость выполнения задач и получить дополнительные удобства, записав часто используемые программы DOS и пакетные файлы на псевдодиск. Псевдодиск — это резидентная программа или драйвер устройства, использующие область оперативной памяти для моделирования этого устройства. Псевдодиски работают быстрее, чем жесткие диски, однако содержащиеся в них данные пропадают при выключении питания компьютера.

Большинство псевдодисков, такие как Vdisk.SYS, поставляемые вместе с DOS 3.x, загружается как драйвер устройства, т. е. вы должны поместить соответствующее указание в файл CONFIG.SYS. Например, для псевдодиска Vdisk.SYS в файл CONFIG.SYS нужно добавить строку

```
device=vdisk.sys
```

Поскольку для псевдодиска используются ограниченные ресурсы — оперативная память, то рекомендуется ограничить размер псевдодиска, чтобы хранить на нем только часто используемые программы. Если возможно, то используйте минимальные размеры сектора и кластера так, чтобы небольшие по размеру пакетные файлы не занимали слишком больших областей оперативной памяти. Размер кластера, измеряемый числом содержащихся в нем секторов диска, является минимальной единицей объема памяти, отводимой для файла. Например, файлу длиной всего 1 байт отводится целый кластер дисковой памяти.

Для использования псевдодиска укажите его имя (например, c:) в команде Path файла Autoexec.BAT. Команды и пакетные файлы, записанные на псевдодиск, будут выполняться практически мгновенно. Если вы работаете с системой DOS 3.x, то можете записать на псевдодиск также Command.COM, и после этого вы никогда не увидите на экране сообщения, требующего вставить гибкий диск с файлом Command.COM в дисковод A.

Приведенные ниже два пакетных файла, Autoexec.BAT и Autoram.BAT, выполняют все необходимые операции по установке файла Command.COM и некоторых программ DOS на псевдодиск, объявленный в системе DOS 3.x как диск C.

Целью передачи управления программой Autoexec.BAT программе Autoram.BAT является возможно более раннее получение последовательности команд загрузки на быстром псевдодиске.

Файл Autoram.BAT следует рассматривать как выполняемое с большей скоростью расширение файла Autoexec.BAT, поэтому в файл Autoram.BAT можно добавлять все команды, которые обычно включаются в файл Autoexec.BAT. Любые другие команды, определяющие конфигурацию системы и начальные установки, следует включать после строки 5. Команда Prompt задает перед знаком приглашения DOS текущий путь поиска, затем выполняется переход на дисковод A и добавляется приглашение ввести дату и время, так как они не появляются автоматически при использовании файла Autoexec.BAT.

В качестве первого этапа подготовьте в файле Autoexec.BAT следующий текст:

```
echo off
copy autoram.bat c:
c:
autoram
```

Затем создайте файл Autoram.BAT:

```
echo off
copy a:command.com
set comspec=c:\command.com
copy a:\boot\*.*
path c:\
prompt $p$g
a:
date
time
```

Программа Autoexec.BAT копирует второй пакетный файл, Autoram.BAT, на псевдодиск и выполняет его. Далее программа Autoram.BAT копирует файл Command.COM с дисковода A; затем командой Set сообщается системе DOS, что Command.COM находится на дисководе C, а не на дисководе A, как обычно. Для упрощения копирования программ DOS и пакетных файлов на псевдодиск при создании файла Autoram.BAT предполагается, что на дисководе A они хранятся на гибком диске в каталоге по имени Boot, откуда загружается система DOS. Команда Path в файле Autoram.BAT устанавливает путь поиска, указывающий на псевдодиск.

Вывод сообщений из пакетных файлов

Включение в пакетные файлы команд для вывода сообщений как для пользователя, так и для внутреннего документирования является искусством. Первый прием, которому обучается большинство программистов, состоит во включении в пакетный файл в качестве первой строки команды Echo Off, которая запрещает вывод на экран, обеспечивая более быстрое выполнение пакетного файла. При необходимости вывода сообщения на экран по-прежнему можно пользоваться командой Echo. Потребность включения режима вывода на экран командой Echo On возникает лишь в редкие моменты, например при вывода управляющих кодов стандарта ANSI в команде Prompt. Полезно также отложить включение команды Echo Off до тех пор, пока не будет отлажена программа пакетного файла.

Следующая команда, которую осваивают программисты, служит для включения в пакетный файл комментариев и начинается с ключевого слова Rem. Если задан режим Echo On, воспользоваться им для отображения на экран сообщений и инструкций можно, однако вместе с ними будут выводиться на экран и остальные строки пакетного файла. Поскольку комментарии не выводятся на экран при установленном режиме Echo Off, то команды Rem больше подходят для комментирования больших пакетных файлов, поясняя их работу. Комментарии могут вставляться в любое место пакетного файла и начинаются с символов rem, например:

```
rem This batch file requires the Mode command.
```

Хотя во время выполнения пакетного файла DOS игнорирует комментарии, она все же читает их по одному в каждый момент времени, что замедляет выполнение программы, особенно в системе с дисководами для гибких дисков. Чтобы помочь системе DOS быстро пропускать комментарии, можно применять два приема. Первый прием состоит во введении комментариев не с помощью символов rem, а символом двоеточия. Фактически такой прием превращает строку комментария в метку. Нужно лишь убедиться, что ни одна из команд перехода Go to не использует в качестве метки перехода первое слово в каком-либо комментарии.

Второй прием заключается в предварении последовательности строк с комментариями командой Go to, которая обеспечивает обход блока комментариев. При использовании этого приема нет необходимости в употреблении символов rem или двоеточия. Оба приема обеспечивают примерно одинаковый выигрыш в скорости. Например, компьютеру Blue Chip PC (эквивалент IBM PC с тактовой частотой 4,77 МГц) требуется 18 с для чтения 20 комментариев и только 5 с для их обхода. На высокопроизводительном компьютере Mitsubishi AT (10 МГц) нужно соответственно 1 и 0,5 с для выполнения тех же операций.

Для улучшения представления результатов и работы пакетного файла существует другой метод. Улучшению вида сообщений способствует добавление пустых строк. В DOS 2.x вывод пустой строки обеспечивает включение в пакетный файл команды Echo с двумя пробелами после слова Echo. Однако этот прием не пригоден для DOS 3.x. В этом случае можно после слова Echo включить пробел и число нуль или 255. Число 255 можно ввести, нажав клавишу Alt и набрав на цифровой клавиатуре 255. Нуль можно ввести просто нажатием функциональной клавиши F7, после чего на экране появятся знаки ^@. Хотя нулю и числу 255 соответствуют коды ASCII, они невидимы на экране.

Если вы хотите вывести на экран сразу несколько пустых строк, лучше воспользоваться командой Type для вывода содержимого заранее подготовленного текстового файла. Это оказывается быстрее, чем вывод каждой строки отдельно с помощью команды Echo. При использовании команды Type необходимо помнить о существовании дополнительного текстового файла и его местонахождении. Избавиться от ошибок в этом случае может помочь выбор имен файлов, показывающий связь между ними.

Прием, который, вероятно, привлекает наибольшее внимание, помогает воспринимать сообщение не зрительно, а на слух – звуковой сигнал. Звуковой сигнал обеспечивается стандартным кодом ASCII 7 (^G), который можно получить нажатием клавиши G при нажатой заранее клавише Ctrl. Этот символ можно получить, работая с редактором Edlin или с командой Copy CON, описанными в разделе "В помощь начинающим". Для получения звукового сигнала поместите в пакетный файл строку

```
echo ^G
```

Советы по организации файловой системы на жестком диске

Если ваша система имеет жесткий диск, вы уже, вероятно, организовали на нем подкаталоги и распределили по ним свои файлы. Такая организация облегчает прослеживание местонахождения файлов. С помощью пакетных файлов можно также упростить выполнение программ.

Прежде всего вам понадобится разделить прикладные программы и соответствующие им файлы данных по разным подкаталогам. Если какая-либо прикладная программа создает много файлов данных, можно подумать о том, как их распределить по нескольким подкаталогам следующего уровня внутри подкаталога, выделенного для файлов данного приложения. В любом случае приходится искать компромиссное решение, выбирая между длинными списками подкаталогов и слишком большим числом уровней подкаталогов: в обоих случаях трудно разобраться в организации программ и данных на диске.

Необходимо, чтобы можно было вызывать определенные программы DOS и утилиты независимо от того, какой каталог объявлен текущим. Поместите все такие программы DOS в каталог с именем DOS, а утилиты — в каталог с именем Utility. Затем, используя команду Path и полные имена путей поиска после команды Path, включите в файл Autoexec.BAT такую строку:

```
path c:\dos;c:\utility
```

Тогда программы DOS и утилиты можно вызывать из любого каталога. Точно так же, используя пакетные файлы, можно обеспечить доступ из любого каталога к любой прикладной программе. Сначала создайте подкаталог Batch, в котором будут содержаться все ваши пакетные файлы. Затем добавьте имя подкаталога в список подкаталога в команде Path:

```
path c:\dos;c:\utility;c:\batch
```

Для каждой прикладной программы напишите пакетный файл наподобие приведенного ниже для программы Wordpro.EXE, находящейся в подкаталоге Wordpro:

```
echo off
c:
cd \wordpro
wordpro %1 %2 %3 %4
cd \
```

Вызов программ в этом случае становится полностью автоматизированным. После ввода имени пакетного файла в ответ на приглашение DOS система DOS выбирает в качестве текущих соответствующий дисковод и подкаталог и вызывает программу на выполнение. После завершения программы управление возвращается в пакетный файл и осуществляется обратный переход в корневой каталог. Имея файлы такого типа в подкаталоге Batch и указывая имя этого подкаталога (c:\batch) в команде Path, можно выполнять любые прикладные программы из любого подкаталога. Кроме того, собрав все пакетные файлы в одном месте, можно облегчить ведение каталога.

Я предпочитаю именно этот способ, а не создание длинного списка каталогов прикладных программ в команде Path. Система DOS не должна вести поиск программы в подкаталогах этого списка, когда вы рискуете загрузить не ту программу, если существуют две программы с одинаковыми именами в разных подкаталогах.

Blank.BAT

ГАШЕНИЕ ЭКРАНА ДО НАЖАТИЯ ЛЮБОЙ КЛАВИШИ

```
prompt $e[0;30m$e[2J
pause
prompt $e[0m
echo off
prompt $d It is now
$t$h$h$h$h$h$h$_$p$g
cls
```

Этот необычный пакетный файл гасит ваш экран и оставляет его в таком состоянии до тех пор, пока не будет нажата какая-либо клавиша. Файл Blank.BAT поможет вам продлить жизнь монитора, оставляя погашенным экран, если в это время компьютер не используется.

На примере файла Blank.BAT демонстрируется также использование управляющих кодов стандарта ANSI для вывода на экран и команды Prompt. Эта программа не будет работать, если вы не загрузите драйвера клавиатуры и экрана ANSI.SYS, включив в файл CONFIG.SYS строку

```
device-ansi.sys
```

Команда Prompt обычно определяет, в каком виде на экране выводятся символы приглашения DOS. Файл Blank.BAT показывает другое применение этой команды: посылку управляющих кодов стандарта ANSI дисплею. Первой строкой файл Blank.BAT посылает escape-последовательность $e[0;30m$e[2J на экран, а драйвер ANSI.SYS перехватывает эту последовательность и интерпретирует ее. Нуль задает обычные атрибуты черно-белого экрана; 30m задает цвет текста (черный), фактически гася экран, а символы 2J обеспечивают очистку экрана.

Команда Pause приостанавливает выполнение пакетного файла до тех пор, пока не будет нажата какая-либо клавиша. Если клавиша будет нажата, то будет выполнена другая команда Prompt и экран вернется

к своему нормальному состоянию. Третья команда Prompt обеспечит включение в набор символов подсказки DOS сведений о дате, времени и текущем каталоге.

В последовательности символов стандарта ANSI символы верхнего и нижнего регистров различаются. Например, вы должны будете использовать букву m нижнего регистра и букву J – верхнего. Команда Echo Off в четвертой строке и команда cls очистки экрана в последней строке предотвращают повторный вывод на экран символов подсказки после завершения выполнения пакетного файла. Если бы команда Echo Off была встречена раньше, то символы подсказки не были бы выведены, а управляющие символы стандарта ANSI не были бы посланы.

С помощью текстовых редакторов, таких как Edlin или Sidekick, которые могут включать в текст управляющие символы, можно направить управляющие коды драйверу ANSI.SYS командой Echo. Например, последовательность

```
echo ^[[0;30m^[[ 2J
```

эквивалентна первой строке файла Blank.BAT. ■

Команды *DOS* и пакетные файлы

Если вы часто используете команды DOS с конкретными параметрами, возможно, лучше оформить обращения к ним в виде пакетных файлов. То же самое можно сказать и о специальных вариантах команд DOS, которые используются не настолько часто, чтобы помнить их наизусть. В дополнение к файлу H.BAT из раздела "В помощь начинающим" я приведу несколько небольших пакетных файлов, которые могут вам пригодиться.

CAT.BAT

Пакетный файл CAT.BAT вы можете использовать вместо команды DIR, когда необходимо получить список большого числа файлов какого-нибудь каталога. Мне кажется, что удобно сначала очистить экран, чтобы список файлов уместился на экране целиком без сдвига за пределы экрана. Этот пакетный файл содержит всего три строки:

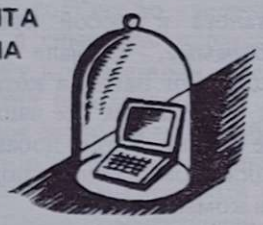
```
echo off
cls
dir %1 /w
```

Если вам хочется получить информацию о величине файла и дате создания, то используйте другой вариант файла CAT.BAT:

```
echo off
cls
dir %1 /p
```

Список файлов каталога будет выводиться постранично.

ЗАЩИТА ФАЙЛА



Файлы Lock.BAT и Unlock.BAT предназначены для защиты файлов. В этих программах для установки и снятия защиты используется команда ATTRIB, имеющаяся в DOS 3.0.

Файл Lock.BAT устанавливает атрибут файла "только для чтения", что предотвращает модификацию любого файла или списка файлов, указанного в качестве параметра в командной строке. Например, командой Lock *.* можно защитить все файлы подкаталога от модификации или удаления. Файл Lock.BAT содержит следующий текст:

```
echo off
if " "= ="%1 goto message
attrib +r %1
goto end
:message
echo No file(s) specified
:end
```

Файл Unlock.BAT снимает защиту с файла, защищенного с помощью файла Lock.BAT, и содержит следующий текст:

```
echo off
if " "= ="%1 goto message
attrib -r %1
goto end
:message
echo No file(s) specified
:end
```

Чтобы сократить число нажатий клавиш, можно заменить имена пакетных файлов на L.BAT и UL.BAT.

ВЫВОД ТЕКСТОВЫХ ФАЙЛОВ НА ЭКРАН

Когда вам необходимо просмотреть на экране большой текстовый файл, выводимый командой Type, было бы хорошо выводить на экран за один прием не более страницы текста, а не нажимать клавиши Ctrl-S или Pause, чтобы остановить сдвиг текста за пределы экрана. С этой целью и написан файл Read.BAT, содержащий следующие строки:

```
echo off
cls
type %1 |more
```

В файле Read.BAT используется программа DOS More.COM, которая является фильтром выводимых командой Type данных, останавливая вывод после заполнения экрана до тех пор, пока не будет нажата какая-либо клавиша. Программа More.COM должна находиться в текущем каталоге диска; в системе с жестким диском местонахождение программы должно быть указано в команде Path, задающей пути поиска в подкаталогах.

Go.BAT

ПЕРЕХОД В УКАЗАННЫЙ ПОДКАТАЛОГ И ВЫПОЛНЕНИЕ ПРОГРАММЫ

```
echo off
cd \
:loop
if "%2" = = "" goto done
cd %1
shift
goto loop
:done
if exist %1.com goto run
if exist %1.exe goto run
if exist %1.bat goto run
cd %1
if exist %1.com goto run
if exist %1.exe goto run
if exist %1.bat goto run
shift
:run
%1
```

Пакетный файл Go.BAT является хорошей заменой команды DOS "перейти в другой каталог" (CHDIR или CD). Выполняя функцию перехода в другой каталог, файл Go.BAT не требует использования обратной наклонной черты, расположенной в неудобном для пальцев месте. Более существенна способность этого пакетного файла вызвать программу на выполнение, если вы укажете имя каталога и имя программы как параметры командной строки. Например, команда

```
go plan sheet
```

обеспечит переход в каталог Plan и вызов программы Sheet.EXE, содержащейся в этом каталоге.

Программа Go.BAT "понимает", что если программа и подкаталог имеют одинаковые имена, то достаточно указать это имя только один раз. Например, если в подкаталоге с именем 123 содержится файл 123.EXE, то команда

```
go 123
```

обеспечит переход в каталог 123 и вызов программы 123.

В цикле в начале файла выполняется последовательный сдвиг списка параметров (по одному на каждой итерации). В процессе выполнения постоянно проверяется число оставшихся параметров; если остались два параметра, то выполняется переход в каталог, указанный первым параметром. Как только останется один параметр, Go.BAT ищет в текущем каталоге программу с именем, заданным этим параметром, и с расширением имени .COM, .EXE или .BAT. Если такая программа существует, то она вызывается на выполнение. Если такой программы нет в корневом каталоге, то в Go.BAT предполагается, что параметр %1 является именем подкаталога и осуществляется переход в этот подкаталог. Затем в этом подкаталоге ищется программа с тем же именем, и если она существует, то выполняется.

Последняя инструкция Shift присваивает переменной %0 значение переменной %1, так что значение переменной %1 становится пустым, если управление не передается последней строке командой Goto Run. Это гарантирует, что файл Go.BAT не станет выполнять программу подкаталога на некотором более глубоком уровне, указанном в списке путей поиска команды Path.

Если вы хотите перейти в некоторый подкаталог без автоматического вызова программы, добавьте в качестве последнего параметра точку. Вызов go пакетного файла Go.BAT без параметров обеспечит просто переход в корневой каталог. ■

Ввод данных из пакетных файлов

Вопрос. Можно ли добавить в пакетный файл такую команду, которая бы автоматически вводила Y (Yes – да) в ответ на запрос, выдаваемый программой, запущенной из этого пакетного файла? Сейчас мне приходится сначала вводить с клавиатуры название пакетного файла, а потом – Y для продолжения выполнения программы. Хорошо бы, чтобы для продолжения выполнения программы мне не нужно было нажимать на клавиши.

Ответ. Так как вам нужно ответить только один раз, для запуска программы без ввода Y можно использовать следующий пакетный файл:

```
echo off
echo Y>yes
program<yes
del yes
```

В этом пакетном файле слово *program* замените названием вашей программы и необходимыми для нее параметрами, задаваемыми из командной строки. Вторая строка файла служит для создания небольшого файла с именем yes, в котором содержатся символы Y и "возврат каретки". Символ ">" служит для записи данных, выдаваемых командой echo, в файл. В третьей строке пакетного файла задается запуск вашей программы, причем с помощью символа "<" ввод с клавиатуры заменяется вводом из файла yes, а раньше вы сами вводили Y и нажимали клавишу Enter. Команда, стоящая в последней строке пакетного файла, стирает файл yes.

Этот же способ можно использовать, если при запуске программы из пакетного файла нужно ответить на несколько запросов. Однако в этом случае нужно создать постоянный текстовый файл, содержащий необходимые ответы. Запомните, что символ возврата каретки, задающий конец строки текста, равнозначен нажатию клавиши Enter. Учтите также, что текстовый файл должен находиться в одном каталоге с пакетным файлом, чтобы при обращении к пакетному файлу текстовый файл был доступен для DOS.

Например, если с помощью редактора текста вы создадите состоящий из двух строк файл INTABLE.TXT в коде ASCII:

```
D0:0 LFF
Q
```

а затем – пакетный файл INTABLE.BAT:

```
echo off
debug<intable.txt
```

то вы получите пакетный файл, который запускает программу Debug и выводит на экран первые 256 байт памяти компьютера, а затем возвращает управление DOS. Если в процессе работы вам нужно быстро просматривать таблицу векторов прерываний компьютера, то приведенный выше пакетный файл вам пригодится.

Системы управления базами данных

Использование языка *SQL* обеспечивает возможность объединения баз данных

СКОТТ МЕЙС

В соответствии с одним из прогнозов около 81 % фирм из числа входящих в список 500 крупнейших предполагают, что им потребуется доступ к данным более чем одного компьютера.

В последнее время во множестве публикаций отмечается, что язык SQL (Structured Query Language – структурированный язык запросов) становится стандартом "де факто".

Недавнее заявление фирмы Ashton-Tate о том, что преемником системы управления базой данных (СУБД) Dbase IV станет СУБД, построенная на основе использования языка SQL, подлило масла в огонь споров, ведущихся разработчиками СУБД для ПК о пригодности и достаточности этого языка для обеспечения совместимости и целостности СУБД.

Консультант фирмы Codd & Date Consulting Group Фабиан Паскаль считает, что "это слабый стандарт, но все же он бесконечно лучше того, с чем приходилось иметь дело до его появления". Паскаль убежден, что язык SQL должен в будущем стать обязательным компонентом любой СУБД.

Разработчики традиционных СУБД для ПК считают такие рекомендации преждевременными. По их мнению, язык SQL можно рассматривать в лучшем случае как дополнительную возможность, но ни в коем случае как средство, ориентированное на конечных пользователей. Сторонники использования языка SQL считают, что поскольку в отделах административных информационных систем ряда фирм уже начался переход к использованию языка SQL, то любая СУБД, не включающая в себя этого языка, окажется "за бортом".

НЕОБХОДИМОСТЬ ИСПОЛЬЗОВАНИЯ ЯЗЫКА SQL

Необходимость использования средства, аналогичного языку SQL, абсолютно очевидна. В соответствии с одним из прогнозов около 81 % фирм из числа входящих в список 500 крупнейших предполагают, что им потребуется доступ к данным более чем одного компьютера. Этот прогноз является итогом обработки результатов анкетирования 100 руководителей отделов административных информационных систем, отвечающих за выбор СУБД. Исследования, проведенные фирмой International Data, показали, что 84 % фирм предполагают, что им потребуется доступ к более чем одной СУБД или файловой системе.

Это захватило врасплох фирму Ashton-Tate, специализирующуюся в основном на создании однопользовательских СУБД. Объявленное ею на этот год начало поставок непомерно раздутой СУБД Dbase IV возможно будет перенесено на более поздний срок, что, в частности, является следствием добавления в эту СУБД большого числа новых команд, часть из которых реализует конструкции языка SQL.

Недавно фирма Ashton-Tate объявила о своем намерении отказаться от используемой ею идеологии построения СУБД (не менявшейся с момента выпуска СУБД Dbase II в 1981 г.) и перейти к использованию разработанной фирмой Interbase Software СУБД, ориентированной на язык SQL.

По мнению Роя Фоука, вице-президента фирмы Ashton-Tate, удобство работы конечных пользователей с новой СУБД будет обеспечено использованием разнообразных надстроек и инструментальных средств типа генераторов отчетов, языков запросов "а-ля" QBE (Query by Example – запрос по образцу), средств разработки входных и выходных форм и т. д. Все эти надстройки и инструментальные средства будут переписаны для их использования с новой СУБД фирм Ashton-Tate и Interbase Software в ОС OS/2.

Сценарий взаимодействия пользователей с системами, построенными на основе СУБД Dbase III Plus, может усложниться в результате перехода к использованию новой СУБД. И хотя ожидается, что существующие системы смогут быть использованы с СУБД Dbase IV, переход к новой СУБД, по мнению Фоука, может потребовать значительной переработки существующих систем с тем, чтобы воспользоваться предоставляемыми языком SQL возможностями.

"Мы делаем все, от нас зависящее, чтобы все системы, разработанные для наших СУБД, начиная от СУБД Dbase II, сохранили свою работоспособность", – заявил Фоук. По его словам, при переходе к использованию новой СУБД пользователи смогут в ряде случаев (например, при генерации отчетов) применять уже знакомые им команды, однако, чтобы воспользоваться такими

новыми возможностями, как обработка транзакций, сохранение базы данных и ее восстановление, им придется применять примитивы языка SQL.

ВЛИЯНИЕ НА ПОЛЬЗОВАТЕЛЕЙ СУБД

Это откровение может оказаться либо благословением, либо проклятием для пользователей СУБД Dbase, число которых в настоящее время по самым скромным оценкам достигает 2,5 млн, что позволяет считать СУБД Dbase самой распространенной СУБД для ПК.

Фирма Ashton-Tate не одинока в своих усилиях по добавлению возможности использования языка SQL в существующую СУБД. В частности, фирма Microrim предусмотрела возможность использования языка SQL в последней версии своей СУБД Rbase, предназначенной для использования с ОС MS DOS и OS/2.

Президент фирмы Microrim Дейвид Халл, представляя новый продукт, заявил, что "предлагаемый вариант языка SQL не на 100 % соответствует стандарту Американского национального института стандартов на язык SQL, а представляет собой его достаточно большое подмножество, расширенное рядом средств, навеянных СУБД DB2". Фирма Borland International планирует реализовать язык SQL для СУБД Paradox.

По мнению президента фирмы Wallsoft, занимающейся разработкой инструментария для пользователей СУБД Dbase, Мартина Райнхарта, язык SQL может быть реализован аппаратно в виде модулей, вставляемых в ПК типа PC/AT или ПК семейства PS/2, что позволит использовать их в качестве файл-серверов, не уступающих файл-серверам, построенным на основе современных мини-компьютеров, а, быть может, даже превосходящих последние.

"Однако написание программ на языке SQL для выполнения достаточно сложных действий – весьма трудоемкий процесс, да и сами программы получаются достаточно громоздкими и неуклюжими, – считает Райнхарт. – Язык SQL является единственным языком, который после целого месяца его интенсивного использования не позволил мне ощутить, что я овладел им в достаточной степени."

Для того чтобы не заставлять программистов изучать язык SQL, фирма Ashton-Tate включила в СУБД Dbase IV специальные средства, обеспечивающие преобразование программ, написанных на традиционном для СУБД Dbase языке, в программы на языке SQL. Одна из этих программ разработана фирмой Sybase. В настоящее время фирмы Ashton-Tate, Sybase и Microsoft разрабатывают программу SQL Server, которая представляет собой предназначенную для использования с ОС OS/2 версию программы SQL Dataserver, разработанную фирмой Sybase для использования с ОС UNIX. Эта программа будет распространяться фирмой Ashton-Tate.

Поставка первой версии СУБД Dbase IV в соответствии с планами фирмы Ashton-Tate должна начаться с 31 июля 1988 г., опережая на несколько месяцев поставку программы SQL Server, которая предположительно должна начаться во второй половине 1988 г. Одновременно с началом поставки программы SQL Server фирма Ashton-Tate планирует начать поставку следующей версии СУБД Dbase IV – версии 1.1, функционирование которой будет поддерживаться программой SQL Server. Переход пользователей к этой версии СУБД Dbase IV должен быть "безболезненным".

Старший аналитик фирмы Tom Rettig Associates Том Реттиг считает, что первая версия СУБД Dbase IV позволит программистам освоить новый язык Dbase SQL. Программы, написанные на этом языке, будут транслироваться в программы, написанные на традиционном для СУБД Dbase языке. Хотя фирма Ashton-Tate еще не сообщила об используемых ею методах, Реттиг подозревает, что слова о том, что функционирование версии 1.1 СУБД Dbase IV будет поддерживаться программой SQL Server, означают, что операторы языка SQL будут непосредственно выполняться программой SQL Server без их предварительной трансляции в операторы традиционного для СУБД Dbase языка.

Абсолютно неясно также, будут ли в дальнейшем фирмы Ashton-Tate и Interbase Software поставлять средства, обеспечивающие трансляцию программ, написанных на традиционном для СУБД Dbase языке, в программы, написанные на языке SQL. От этого зависит решение вопроса о том, должны ли программисты изучать язык SQL. Для СУБД Dbase IV такие средства объявлены.

По мнению Реттига, "большинство программистов не будут использовать язык SQL, а останутся верными традиционному для СУБД Dbase языку".

"Чтобы ощутить преимущества, предоставляемые языком SQL, пользователи могут преобразовывать (вручную) те или иные фрагменты своих программ, – считает Фоук. – И я думаю, что многие пользователи пойдут по этому пути, хотя он и не является обязательным для всех."

ИСПОЛЬЗОВАНИЕ ЯЗЫКА

Между тем среди экспертов продолжаются споры о том, как использовать язык SQL – в качестве основного языка или в качестве добавочного средства, надстраиваемого над уже существующими СУБД.

Реттиг предсказывает, что многие разработчики последуют примеру фирмы Ashton-Tate, встроившей язык SQL в свою СУБД Dbase IV, и надстроят предлагаемые ими СУБД средствами, позволяющими использовать язык SQL.

Сторонниками использования языка SQL в качестве основного являются главным образом те, у кого за плечами имеется опыт использования мини-, средних и больших ЭВМ, поскольку для машин этих классов язык SQL был принят в качестве основного еще несколько лет назад вследствие шумного успеха СУБД DB2 и SQL/DS. Среди тех, кто высказывает намерения перенести на ПК те средства, которые реализованы на основе использования языка SQL на больших ЭВМ, можно назвать фирму IBM, планирующую выпустить расширенную версию ОС OS/2 (OS/2 Extended Edition) и фирму Information Builders, разработавшую СУБД Focus. Фирма Oracle, кстати, создала себе имя, объявив о выпуске СУБД с языком SQL до того, как это сделала фирма IBM.

Боб Уильямс, руководитель отдела маркетинга фирмы Relational Technology, разработавшей СУБД Ingres, в которой обеспечивается возможность использования языка SQL на ПК, убежден, что "надстроить новый интерфейс с пользователем над сложной системой управления данными, создание которой заняло восемь лет, намного проще, чем разработать аналогичную систему для ПК, ориентируясь на уже готовый интерфейс с пользователем".

По мнению Паскаля, применение языка SQL представляет собой попытку справиться с потерями времени, возникающими при использовании нереляционных СУБД.

"Некоторые разработчики СУБД надстраивают язык SQL над нереляционными СУБД, не обеспечивающими возможности работы с множествами. В этих случаях операторы языка SQL транслируются в операторы языка более низкого уровня, обеспечивающие возможность работы с записями. В результате этого возникает ряд проблем, связанных с необходимостью обеспечения достаточно высокой производительности", – считает Паскаль. По его мнению, помимо недостаточно высокой производительности СУБД, основанные не на использовании языка SQL, оказываются потенциально небезопасными при их эксплуатации в распределенных системах, в то время как "в случае применения СУБД, основанной на использовании языка SQL, пользователь оказывается не в состоянии нарушить целостность базы данных. СУБД третьего поколения, над которыми реализована надстройка, позволяющая использовать язык SQL, дает возможность обойти механизм защиты и нарушить целостность базы данных".

ПРОТИВНИКИ ЯЗЫКА

Некоторые специалисты по СУБД, однако, считают, что язык SQL не так уж хорош, как про него говорят.

"Требование необходимости использования языка SQL придумано журналистами", – считает консультант по СУБД Dbase Адам Грин. В качестве примера, иллюстрирующего его правоту, он приводит многочисленных пользователей, успешно применяющих СУБД Dbase, которые еще не имеют возможности использовать язык SQL.

"Сегодня покупатель не просит дать ему язык SQL, он просит дать ему СУБД, – говорит Грин. – А завтра покупатель будет клясться, что ему необходима СУБД с языком SQL."

"Фирма Ashton-Tate доказала, что ее СУБД и язык – самые лучшие", – убежден Грин.

Проект фирм *Microsoft* и *Ashton-Tate* по созданию пакета для обслуживания баз данных*

Фирмы Ashton-Tate и Microsoft приняли неожиданное решение сотрудничать в разработке программного обеспечения для обслуживания баз данных в локальных сетях на основе OS/2. Разрабатываемый этими фирмами пакет SQL/Server, в котором используются команды языка SQL, будет поддерживать работу распределенной базы данных, обеспечивающей разделение заданий между рабочей станцией с DOS или OS/2 и обслуживающим пакетом, работающим на мощном компьютере с микропроцессором 80386. Этот предназначенный для поддержки транзакций и выполнения других внутренних функций пакет поступит в продажу во второй половине года. Фирма Microsoft будет поставлять его организациям, а фирма Ashton-Tate обеспечит его розничную продажу по цене от 1500 до 3000 дол. Хотя другие поставщики баз данных не будут допущены к распространению этого программного обеспечения, создание ими сервисных пакетов для рабочих станций будет приветствоваться. Фирмы Borland, Symantec, Information Bilders и Blyth уже приступили к подобным разработкам. По утверждению фирмы Ashton-Tate, когда появится очередная версия системы dBase, в ее вариантах как для DOS, так и для OS/2 будет использоваться пакет SQL/Server, что обеспечит доступ к нему уже существующим прикладным программам системы dBase. Обе фирмы также отмечают, что пакет SQL/Server не полностью совместим с расширенной версией OS/2 фирмы IBM, в состав которой входят функции для управления базой данных, использующие язык SQL, но не входит рассматриваемый высокопроизво́дительный пакет. Поставщики сетей фирмы 3Com и Novell одобрили концепции пакета SQL/Server, но о своих планах пока не сообщают. Рассмотренные принципы будут прорабатываться фирмой Sybase, которая работает над этим проектом совместно с фирмой Microsoft с 1986 г.

*PC World, 1988, №3, P.73.

Система *dBase* становится дружественной

То ли виновато ее происхождение от больших ЭВМ, то ли размах амбиций, но система dBase из всего многообразия компьютерных программ долгое время считалась наименее пригодной для эксплуатации на ПК. В то же время она была одной из наиболее популярных систем благодаря прикладным программам многих разработчиков, которые помогли пользователям понять смысл загадочной подсказки, выдаваемой системой dBase в виде точки.

Сейчас любимое детище фирмы Ashton-Tate имеет свои собственные средства взаимодействия с пользователем. Система dBase IV – это хорошо разработанная система управления базой данных, которая имеет более широкие функциональные возможности, чем ранние версии, и гораздо проще их в эксплуатации. Около 250 новых или усовершенствованных команд сделали и без того мощный язык системы dBase еще мощнее, а добавление структурированного языка запросов и нового непроцедурного интерфейса подготовили предпосылки для обмена файлами между dBase IV и базами данных как на больших ЭВМ, так и на множестве ПК.

Новый программный интерфейс Control Center заменяет меню Assist, которое использовалось в системе dBase III Plus. Этот интерфейс больше всего похож на систему меню интегрированной программы Framework фирмы Ashton-Tate, и его с помощью программных средств формирования меню легко приспособить к потребностям пользователя. В нескольких окнах приводятся списки всех доступных баз данных, запросов, программ, сообщений и файлов. Выберите один файл, и он, а также все относящиеся к нему файлы перейдут в активную область соответствующих им окон. Можно быстро переключиться с табличного формата данных (просмотр) на представление данных в виде форм (редактирование) и обратно. Если вы не хотите пользоваться меню, то можно вызвать команду, нажав клавишу Alt и клавишу, соответствующую первой букве команды.

Ветераны ПК все еще гордятся умением вводить команды в командной строке, но система dBase IV создана для максимального облегчения процесса программирования. Она располагает средством "запрос по образцу", которое позволяет выбирать различные поля и операции для поиска по всем файлам. Система автоматически формирует код завершения поиска. Для тех же целей предназначен и генератор программ системы dBase IV, который позволяет обобщенные программные функции ставить в соответствие элементам меню и таким образом автоматически формировать меню. В состав системы dBase IV входит также генератор отчетов WYSIWYG.

Система dBase IV генерирует псевдокомпилированный код, поэтому программы, созданные с ее помощью, работают в среднем в девять раз быстрее, чем при интерпретации в более ранних версиях системы. Хотя с ее помощью нельзя создавать независимо выполняемые файлы типа .EXE, но псевдокомпилятор позволяет разработчикам формировать оптимизированные по времени выполнения объектные файлы, которые могут использоваться в программе или в загрузочном модуле, поставляемом вместе с исходной версией системы.

В системе dBase IV учтен опыт ранних версий. Новая версия полностью совместима с базами данных, программами и т. п., созданными с помощью предыдущих версий. Все разработки, выполненные с помощью предыдущих версий dBase, применимы в dBase IV, но преимущества новых возможностей системы при этом не используются. В новой версии можно использовать без изменений вновь написанные или даже уже готовые прикладные программы системы dBase для локальных сетей.

Будут поставляться версии системы dBase IV и для DOS, и для OS/2 1.0. Для эксплуатации системы необходимы 640 Кбайт памяти и жесткий диск. Цена в настоящее время еще не определена.

Применение СУБД *DataFlex* для автоматизации управленческого труда

Б. К. НОРКИН, С. К. СОМОВ

ВВЕДЕНИЕ

Наряду с популярной системой управления базами данных (СУБД) dBase III Plus фирмы Ashton-Tate широкое распространение получили и некоторые другие реляционные СУБД для IBM PC/XT/AT-совместимых ПК, например DataFlex фирмы Data Access. Обе эти СУБД имеют много общего и в принципе с равным успехом могут быть применены для решения многих задач автоматизации управленческого труда. Выбор той или иной СУБД во многом зависит от вкусов разработчиков, предшествующего опыта программирования и других субъективных факторов, оказывающих, однако, непосредственное влияние на сроки разработки и качество полученного программного продукта. В настоящей статье дается краткое представление о СУБД DataFlex как возможной альтернативе dBase III Plus и другим аналогичным СУБД.

Свойства СУБД DataFlex будут иллюстрироваться на примере автоматизированной системы контроля исполнения документов (АСКИД), разработанной нами для ПК Labtam-3004. У нас имеется опыт использования этой СУБД и на IBM XT-совместимых ПК; ее работа опробована также на ПК Искра-1030 и ЕС 1841.

Назначение и функции АСКИД

Контроль исполнения документов – это одна из самых популярных задач автоматизации управленческого труда. Поэтому система АСКИД является удачным примером, на котором можно проиллюстрировать возможности СУБД DataFlex при решении задач этого класса. Подобные системы разрабатывались для различных типов ЭВМ, включая мини-ЭВМ и ЭВМ серии ЕС. Их реализации отличаются и по конкретному назначению, и по функциям, и по объему хранимой информации, и по технологии работы пользователей. Наша реализация системы АСКИД по назначению и функциям довольно близка к системе "АСУ-Коллегия", разработанной для большой ЭВМ.

Принцип построения АСКИД и подобных ей систем достаточно прост: в систему вводится краткая информация о документе (приказе, письме и т. п.) с регламентированным сроком исполнения. По мере приближения срока исполнителю документа выдаются напоминания. Как только документ исполнен, в систему вводится докладная о его исполнении. Поддерживаемая таким образом база данных может использоваться для подготовки разного рода оперативных сводок и периодических отчетов, форма которых во многом зависит от потребностей конкретного учреждения.

На самом деле при реализации АСКИД появляется много нюансов, существенно усложняющих описанную выше схему. Например, один документ может иметь много исполнителей, докладная может "закрывать" только часть документа и т. д. Эти нюансы нередко зависят от сложившейся в учреждении практики делопроизводства.

ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ СУБД DataFlex ВЕРСИИ 2.2

Структура: реляционная база данных с индексируемыми файлами.

Требования к аппаратуре: раздел оперативной памяти объемом 150 Кбайт для подсистемы исполнения приложений, дисплей с адресуемым курсором и двумя уровнями интенсивности свечения, 1 Мбайт памяти на магнитном диске.

Операционная среда: MS-DOS 2.x и 3.x, Advanced Netware, Concurrent CP/M и Concurrent DOS, VAX VMS, UNIX System V.

Количество файлов: до 250.

Число записей в файле: до 16,7 млн.

Число полей в записи: до 255.

Число индексов в файле: до 10.

Число полей в составном индексе: до 6.

Длина составного индекса: до 255 байт.

Размер файла: до 2 Гбайт.

Характеристики СУБД *DataFlex*

Технические характеристики СУБД DataFlex довольно типичны для программных продуктов этого класса (см. вставку). Ее можно подразделить на пять подсистем: разработки приложений, исполнения приложений, выполнения запросов, меню, эксплуатации базы данных.

ПОДСИСТЕМА РАЗРАБОТКИ ПРИЛОЖЕНИЙ

Подсистема разработки приложений предназначена для определения структуры базы данных, создания прикладных программ и включает в себя редактор, вместо которого можно (и лучше!) использовать любой другой редактор текстовых файлов, компилятор языка DataFlex и несколько утилит, которые позволяют определять структуру файлов базы данных и генерировать "заготовки" будущих программ. Этих утилит вполне достаточно для автоматизации телефонного справочника или библиографического указателя, однако уже такая система, как АСКИД, требует программирования на языке DataFlex.

Логическая структура базы данных системы АСКИД представлена на рис. 1. Основными файлами АСКИД являются картотеки документов и писем; файлы докладных играют подчиненную роль, а классификаторы служат для перевода хранящихся в основных и вспомогательных файлах числовых значений (индекса исполнителя, индекса корреспондента и т. д.) в соответствующий им текст (фамилия исполнителя, фамилия корреспондента и т. д.).

На рис. 2 приведен пример структуры файла картотеки документов системы АСКИД, выданный с помощью одной из утилит DataFlex. На этом рисунке представлен весь спектр допустимых типов полей записи: число (до 14 знаков слева и до 4 знаков справа от десятичной точки), ASCII (строка текста длиной до 255 символов), дата (возможен один из трех форматов даты: европейский, американский, военный американский). На нем показано наличие у файла составных индексов; поле 0 соответствует номеру записи и включается в конец индекса в том случае, если комбинация значений предшествующих полей не обеспечивает уникального значения индекса. Так как одно и то же поле может входить в несколько индексов, то для таких полей указывается "главный индекс", т. е. тот индекс, который будет использован при поиске по значению данного поля. В столбце "связь" значения "файл" и "поле" указывают номер связанного файла и его связующего поля.

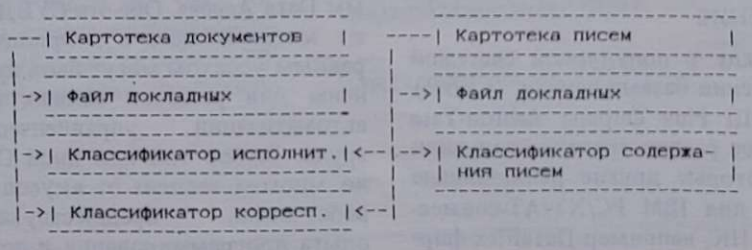


Рис. 1. Структура базы данных системы АСКИД

Языковые средства DataFlex довольно оригинальны. К наиболее приятным особенностям языка DataFlex можно отнести следующие.

1. Наличие в языке конструкций "экранная форма" и "окно" (переменный элемент экранной формы). На рис. 3 показан упрощенный вариант одной из программ АСКИД, содержащий две экранные формы с именами HEADER и ESC_SCREEN. Каждому окну экранной формы присваивается имя, получаемое присоединением номера окна (окна нумеруются слева направо и сверху вниз) к имени экранной формы. Например, окно "индекс корреспондента" будет иметь имя HEADER.13, а окно для ввода ответа в экранной форме ESC_SCREEN – имя ESC_SCREEN.1. Тип данных, соответствующий окну, определяется по виду разметки окна, например:

_____	ASCII-строка длиной 11 символов,
_____.__	число с шестью знаками до точки и двумя после,
_____.	шестизначное целое число,
_/_/_	дата.

Если данная экранная форма активна (т. е. изображена на экране), то результаты всех операций над содержимым ее окон немедленно отображаются на экране.

2. Большое число разнообразных логических и арифметических операций. Например, имеется операция сложения даты и целого числа. В качестве операндов операций могут использоваться как переменные прикладной программы и окна экранных форм, так и поля записей файлов базы данных.

3. СУБД DataFlex допускает многопользовательский режим работы в операционной среде Concurrent DOS (Concurrent CP/M) фирмы Digital Research и в среде локальной сети Advanced Netware фирмы Novell. Для обеспечения многопользовательского режима в языке DataFlex имеется несколько специальных операндов (LOCK, UNLOCK, REREAD).

4. Наличие мощных операторов ввода-вывода, позволяющих контролировать вводимые значения и управлять форматами вводимой и выводимой информации. Примером может служить макрокоманда ENTER, которая дает возможность

```
/HEADER
                КАРТОЧКА КОНТРОЛИРУЕМОГО ДОКУМЕНТА
-------------------------------------------------------------------
| Входящий номер  Корреспондент    Инд.исп.  Исполнитель          |
| <_.|____.|_.>  _________________  <__.>  _____________________  |
|                 _________________  __.   _____________________  |
|                 _________________  __.   _____________________  |
|             Индекс корресп. <__.>  __.   _____________________  |
|                                                                 |
| N док.       Дата док.   Срок исполнения  Дата получения       |
|<______>     <__/__/__>     <__/__/__>      <__/__/__>          |
|Содержание   ____________________________________________________ |
|документа    ____________________________________________________ |
|             ____________________________________________________ |
|             ____________________________________________________ |
|             ____________________________________________________ |
|Резолюция    ____________________________________________________ |
|             ____________________________________________________ |
|Дата резолюции __/__/__  Автор резолюции _____________________   |
-------------------------------------------------------------------
/ESC_SCREEN
Подтвердите завершение работы (д/н): _
/*
PAGE HEADER                    // Изобразить на экране форму HEADER
ENTER KARTDOK                  // Ввести данные в файл KARTDOK
   ENTRY KARTDOK.RAZDKART HEADER.1  (RANGE=1,256) //Ввести компоненты
   ENTRY KARTDOK.NDOK     HEADER.2  (RANGE=1,3600)//входящего
   ENTRY KARTDOK.GODREG   HEADER.3  (RANGE=0,99)  //номера
                  . . . . . . . . . .
   ENTRY KARTDOK.DATREZ   HEADER.27 //Ввести дату резолюции
   ENTRY KARTDOK.AWTREZ   HEADER.28 //Ввести автора резолюции
ENTEREND                           // Конец тела макрокоманды
                                   //ENTER
//
KEYPROC KEY.ESCAPE                 // Заголовок клавишной процедуры
    CLEARFORM ESC_SCREEN           // Очистить окно для ввода
                                   // подтверждения
    PAGE ESC_SCREEN                // Изобразить на экране форму
                                   // ESC_SCREEN
    ACCEPT ESC_SCREEN.1  (CHECK="д|н")
    IF ESC_SCREEN.1 EQ "д" ABORT //При подтверждении завершить работу
    ENTAGAIN                       //В противном случае возвратиться в
                                   // то окно формы HEADER, при вводе в
                                   // которое была нажата клавиша ESC
    RETURN                         // Конец тела клавишной процедуры
```

Рис. 2. Пример структуры файла картотеки документов системы АСКИД

```
      Листинг определения файла номер 8
*************************************************************
      Корневое имя        = C:KARTDOK
      Отображаемое имя    = Kartoteka
      DATAFLEX имя файла  = KARTDOK
*************************************************************
     Длина записи  = 768   (факт.= 730)
     Макс. кол-во записей  = 3600 (факт.= 2500)
     Освобожд. память испол.
     Мультитерм. RS-READ включ.
*************************************************************
Номер Сдвиг Длина  Тип    Дес.  Глав.  Связь
поля                      тчк   инд.   файл  поле
----  ----- -----  -----  ---   ----   ----  ----
   1     1    25 ASCII           0      0     0  KORR1
   2    26     2 Число    0      2      0     0  INDISP1
   3    28    25 ASCII           0      0     0  ISP1
   4    53     2 Число    0      1      0     0  RAZDKART
   5    55     3 Число    0      1      0     0  NDOK
   6    58     1 Число    0      1      0     0  GODREG
   7    59     3 Дата             3      0     0  SROKISP
   . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
  35   706     3 Дата             7      0     0  DATPOL
  36   709     3 Дата             0      0     0  DATREZ
  37   711    30 ASCII            0      0     0  AWTREZ
-------------------------------------------------------------
Инд.  1: поля-сегменты : <4> <6> <5>
Инд.  2: поля-сегменты : <2> <0>
Инд.  3: поля-сегменты : <7> <0>
   . . . . . . . . . . . . . . . . .
Инд.  7: поля-сегменты : <35> <0>
```

Рис. 3. Упрощенный пример одной из программ АСКИД

а) Макроопределение

```
#COMMAND  m$e                    // Заголовок макроопределения
                                 // m$e - имя макрокоманды.
#IF !0>0                         // Если имеются параметры,
  entry !1                       // подставить команду ENTRY
                                 // с первым параметром,
  m$e !2 !3 !4 !5 !6 !7 !8 !9  // и продолжить для ос-
                                 // тальных параметров.
#ENDIF                           // Конец блока условной
                                 // трансляции.
#ENDCOMMAND                      // Конец макроопределения.
```

б) Макрокоманда

```
m$e KARTDOK.RAZDKART KARTDOK.NDOC KARTDOK.DATREZ
```

в) Макрорасширение

```
entry KARTDOK.RAZDKART
entry KARTDOK.NDOC
entry KARTDOK.DATREZ
```

Рис. 4. Пример макроопределения для команды ENTRY: а — макроопределение, используемое для сокращенной записи последовательности команд ENTRY, б — использование макрокоманды m$e в тексте программы, в — макрорасширение — последовательность команд ENTRY для каждого из параметров макрокоманды

автоматически определить набор "клавишных" процедур для основных операций над записями: поиск записи, поиск предыдущей или следующей записи, сохранение и удаление записи, печать содержимого экрана, очистка экрана и буфера записей, а также редактирование полей записи. Кроме того, с помощью команд ENTRY, используемых в теле макрокоманды ENTER, вы можете устанавливать соответствие между окнами экранных форм и полями записей файлов. Все операции по редактированию полей выполняются в окнах, а сброс информации в буфер записи выполняется только перед выполнением операции сохранения ее содержимого. Непосредственно перед сохранением записи надо закрыть доступ к файлу другим пользователям (при работе в многопользовательском режиме), повторно считать активную запись в буфер, сбросить в него содержимое измененных окон, сохранить запись и затем открыть файл для коллективного доступа. Наличие такой процедуры сохранения практически снимает проблемы обеспечения многопользовательского режима, но накладывает некоторые ограничения на программирование обработки полей записи. Важно помнить, что при использовании макрокоманды ENTER все изменения следует выполнять над окнами экранных форм, являющихся буфером для редактирования. Фрагмент программы, работающей с экранными формами и использующей макрокоманду ENTER, приведен на рис. 3.

Параметрами команды ENTRY являются поле записи и окно, которое соответствует этому полю, а также режим обработки ввода в это окно. Второй параметр может быть опущен, если вас устраивает проход по окнам в порядке их естественной нумерации. В этом случае перед первой командой ENTRY указывается команда AUTOPAGE. Третий параметр также не обязателен и используется для проверки допустимости ввода в это окно, указания режима поиска в связанных файлах и т. д.

5. Наличие "клавишных процедур", дающих возможность управлять обработкой функциональных и специальных клавиш в зависимости от состояния диалога пользователя и системы.

Процедура KEY.ESCAPE, приведенная на рис. 3, обеспечивает необходимую пользователю реакцию на нажатие клавиши ESCape при вводе в любое окно экранной формы.

6. Наличие развитых средств программирования подсказок.

7. Наличие встроенного макрогенератора, позволяющего создавать собственные расширения языка, учитывающие специфику разрабатываемых программ. Вы можете использовать макроопределения локально, а можете включить их в стандартную макробиблиотеку СУБД DataFlex. На рис. 4 приведен пример рекурсивного макроопределения.

ПОДСИСТЕМА ИСПОЛНЕНИЯ ПРИЛОЖЕНИЙ

Подсистема исполнения приложений предназначена для исполнения откомпилированных прикладных программ, написанных на языке DataFlex, и обеспечивает их взаимодействие с программными и информационными ресурсами СУБД, а также с внешними устройствами ПК (клавиатурой, экраном дисплея, печатающим устройством и т. д.).

Компилятор СУБД DataFlex, входящий в состав подсистемы разработки приложений, генерирует псевдокод, который интерпретируется подсистемой выполнения приложений. В этом отношении СУБД DataFlex аналогична многим компилирующим версиям языка Бейсик и по производительности находится примерно на том же уровне.

Производительность подсистемы исполнения приложений является узким местом СУБД DataFlex, особенно в случае, если необходимо проводить относительно сложные вычисления. По сведениям фирмы DataAccess в СУБД DataFlex версии 2.3 подсистема исполнения приложения переписана на языке Си, что ускоряет исполнение программ в среднем на 25 %. Кроме того, большим неудобством является отсутствие возможности раздельной компиляции, тем более, что скорость компиляции довольно невелика: например, на ПК Labtam-3004 компиляция 600-строчной программы для СУБД DataFlex версии 2.1 занимает около 10 мин. Скорость компиляции на ПК IBM XT ненамного выше.

Тем не менее использование СУБД DataFlex для реализации АСКИД позволило достичь вполне удовлетворительных временных характеристик. Например, добавление новой записи к файлу контролируемых документов, структура которого показана на рис. 2, занимает около 1,5 с. Переиндексирование этого файла по одному индексу производится со скоростью около 6 записей в 1 с, а переиндексирование по всем семи индексам — со скоростью 1 запись в 1 с. Последовательный просмотр всего файла производится со скоростью 12–15 записей в 1 с.

ПОДСИСТЕМА ВЫПОЛНЕНИЯ ЗАПРОСОВ

Подсистема выполнения запросов позволяет осуществлять поиск информации в файлах базы данных и выдавать простые отчеты. Для формулировки запросов в СУБД

DataFlex имеется язык запросов QUERY. Информация, полученная при ответе на запрос, может быть выведена на экран дисплея, распечатана на принтере или записана в файл на магнитном диске. По сформулированному запросу может быть сгенерирован, а затем и откомпилирован текст программы, реализующей этот запрос, которую в дальнейшем можно использовать, как и все другие прикладные программы, а также в качестве "заготовки" для программирования более сложного отчета.

```
НОМЕР МЕНЮ : 2        НАЗВАНИЕ:      = А С К И Д =
                      МЕНЮ     : Работа с картотекой документов

                      МЕНЮ ВОЗВРАТА   1 (по "RETURN")

   ПОЗИЦИЯ В МЕНЮ                     ДЕЙСТВИЕ        ПАРОЛЬ
1. Регистрация документов             CHAIN PROGR1
2. Поиск, корректура документов       CHAIN PROGR2
3. Печать карточек документов         CHAIN PROGR3
4. Регистрация докладных документов   CHAIN PROGR4
5. Поиск, корректура докладных        CHAIN PROGR5
6. Справки, сводки по документам      MENU 3
7. Меню системных утилит              MENU 4          SYS
8.
9.
```

Рис. 5. Пример описания меню в СУБД DataFlex

ПОДСИСТЕМА МЕНЮ

Подсистема меню позволяет с помощью утилиты MENUDEF создавать внешний уровень интерфейса пользователя и обеспечивает вызов утилит СУБД и откомпилированных прикладных программ пользователя. На рис. 5 представлено описание меню средствами СУБД DataFlex.

Утилита MENUDEF позволяет в каждом меню определять до девяти различных действий, которые могут заключаться как в переходе к другому меню (например, MENU 3 на рис. 5), так и в вызове прикладной или системной программы (например, оператор CHAIN PROGR1 на рис. 5 обеспечит исполнение программы PROGR1). Каждый из выборов, предложенных в меню, может быть защищен паролем. Например, на рис. 5 показано, что переход к меню системных утилит защищен паролем SYS.

ПОДСИСТЕМА ЭКСПЛУАТАЦИИ БАЗЫ ДАННЫХ

Подсистема эксплуатации СУБД DataFlex предоставляет довольно мощные средства для поддержки целостности информации, хранящейся в базе данных. В эту подсистему входят следующие утилиты: FILEDEF – определение и модификация структуры файлов данных и их индексов; REINDEX – реиндексирование файлов данных по одному или одновременно по нескольким индексам; READ – загрузка информации в базу данных из текстовых файлов, что обеспечивает возможность импорта/экспорта данных из других приложений и обмена данными между СУБД DataFlex и другими СУБД.

Заключение

Существенной, хотя и субъективной характеристикой СУБД является скорость разработки прикладных программ ее средствами. В этой связи мы можем отметить, что разработка системы АСКИД средствами СУБД DataFlex заняла приблизительно 4 человеко-месяца. Сюда включены освоение языковых средств и установка системы пользователю, обучение и ввод в опытную эксплуатацию. Опыт следующих разработок показал, что трудозатраты в 3–4 человеко-месяца являются характерными для разработки систем подобного уровня сложности средствами СУБД DataFlex.

Наряду с описанными выше достоинствами СУБД DataFlex имеет ряд недостатков. В первую очередь к ним следует отнести то, что компилятор DataFlex создает псевдокод, который интерпретируется подсистемой исполнения приложений. Это затрудняет включение в программу, написанную на языке DataFlex, обрабатывающих процедур, написанных на других языках программирования. Далее, в СУБД DataFlex отсутствует механизм передачи параметров; среди типов данных отсутствуют массивы. Остальные недостатки СУБД DataFlex, например недостаточно полная документация, носят менее принципиальный характер.

Тем не менее при известных навыках программирования на языке DataFlex эти недостатки не мешают успешной реализации широкого класса задач автоматизации управленческого труда. Кроме того, СУБД DataFlex активно развивается и в ее новых версиях острота недостатков постепенно сглаживается. Система АСКИД разработана с помощью СУБД DataFlex версии 2.1, в последующих наших разработках используется версия 2.2. Недавно появилось сообщение о выпуске версии 2.3; в ней существенно улучшена производительность подсистем разработки и исполнения приложений. Кроме того, эта версия обладает встроенными средствами графического вывода данных.

В целом СУБД DataFlex можно рассматривать как хорошее инструментальное средство для решения задач автоматизации управленческого труда в подразделении или относительно небольшом учреждении, где достаточно ограничиться несколькими терминалами или локальной сетью, обслуживающей несколько рабочих станций.

Быстрый компилятор

Dbfast – это компилятор языка базы данных Dbase III Plus, который обеспечивает меньшее время компиляции и более компактный код. Минимальный размер файла типа .EXE составляет 1 Кбайт, типичные программы занимают от 5 до 10 Кбайт. Этот компилятор обрабатывает стандартные команды, функции и файлы, включая индексные, базы данных Dbase III Plus. Возможно использование его в локальных сетях. Цена пакета Dbfast 149 дол.

Адрес для контактов: Dbfast Inc., 1420 N.W. Gilman Blvd., Issaquah, WA 98027-5399, 206-392-0368.

Персональные компьютеры в учебной лаборатории по физике

В.Н. СУШКИН

Вычислительная техника находит широкое применение как в теоретической, так и в экспериментальной физике. Поэтому не удивительно, что компьютеры быстро и эффективно вошли в методику преподавания физики. Здесь, как и при обучении другим предметам, возможно использование компьютера в качестве средства автоматизированного контроля знаний или носителя автоматизированного обучающего курса. Однако в физике применение компьютеров значительно расширяется приложением к автоматизированной обработке результатов эксперимента и возможностью численного моделирования физических процессов. Появление персональных компьютеров резко расширило возможности применения вычислительной техники в обучении. Наличие цветного графического дисплея и простота организации диалога позволяют строить на персональных компьютерах учебные имитационные компьютерные модели. При функционировании компьютерной модели машина воспроизводит некоторый физический процесс или эксперимент, ход которого отображается на графическом дисплее. Учащийся управляет экспериментом путем нажатия определенных клавиш.

Обучение с применением имитационных моделей удачно вписывается в работу общего практикума по физике. Обычно каждый учащийся один-два раза в семестр выполняет работу не в лаборатории, а в дисплейном классе. До занятия учащемуся выдается описание, в котором приведена необходимая теория, предложена методика управления экспериментом, сформулировано задание. Полученные в процессе работы на ПК данные обрабатываются учащимися самостоятельно, а затем проверяются преподавателем.

При выборе тематики имитационного моделирования отбирались задачи, удовлетворяющие таким основным требованиям, как ясность физического содержания и использование материала, который трудно иллюстрировать в лабораторном практикуме. При составлении программ учитывались обычные требования к педагогическому программному продукту.

В настоящее время создан пакет программ по моделированию физического эксперимента, содержащий следующие разделы: молекулярно-кинетическая теория, электростатика, магнетизм, колебания и волны. Каждая имитационная модель сконструирована и управляется по единой схеме. Большая часть экрана выделена для отображения информации о работе модели. Одновременно на дисплее представлено меню, в котором кратко сформулированы допустимые в данный момент действия учащегося. Управление процессом осуществляется с помощью передвижения курсора и пробела. Особое внимание уделено защите моделей от несанкционированных действий, ввода недопустимых значений или возникновения нештатных ситуаций. Модели реализованы на ПК класса MSX фирмы Yamaha. Большая часть программ написана на языке Турбо-Паскаль, некоторые – на Бейсике. Каждая модель занимает 15 – 20 Кбайт памяти. Остановимся кратко на моделях каждого из разделов.

МОЛЕКУЛЯРНО-КИНЕТИЧЕСКАЯ ТЕОРИЯ

Моделирование процессов в газе в соответствии с уравнением Ван-дер-Ваальса и нахождение его критических параметров. На экране изображен цилиндр с газом, закрытый подвижным поршнем. С помощью измерительных приборов определяются объем, температура и давление газа. Объем и температуру можно изменять независимо друг от друга. При изменении объема на экран также выводятся графики изотерм. Перед учащимся ставится задача определения на основании данных эксперимента константы Ван-дер-Ваальса и критических параметров газа.

Изучение функций распределения молекул газа по скоростям. В сосуде имеется газ, находящийся в термодинамическом равновесии при некоторой температуре T. К сосуду присоединен прибор, который поочередно измеряет скорости поступающих в него молекул. В ПК результаты измерений такого прибора моделируются с помощью датчика случайных чисел, распределение которых соответствует закону Максвелла. На экран дисплея выводится гистограмма скоростей, а также распределение Максвелла, к которому в пределе стремится гистограмма. Одновременно проводится статистическая обработка требуемых реализаций, в результате которой определяются температура газа и доверительный интервал ее определения при заданной надежности. Перед учащимся ставится задача накопления статистических данных, необходимых для определения температуры с заданной точностью.

Релаксация функции распределения газа. В этой работе учащийся может расположить на экране ПК до десяти молекул газа, а также задать направления их скоростей. Затем ПК моделирует движение молекул газа, при этом столкновения между молекулами рассматриваются как соударения абсолютно упругих шаров. Движение шаров отображается на дисплее в реальном масштабе времени. Так же, как и в предыдущей работе, строится гистограмма распределения скоростей молекул. Кроме того, на экран можно вывести текущее распределение молекул по объему. В процессе выполнения работы видно, что любое начальное распределение релаксирует к равномерному распределению по координатам и максвелловскому — по скоростям. Перед учащимся ставится задача определения характерного времени релаксации. На рис. 1 приведены изображаемые на экране ПК шары-молекулы, график распределения Максвелла и гистограмма конкретной реализации распределения по скоростям, а также краткое меню возможных действий.

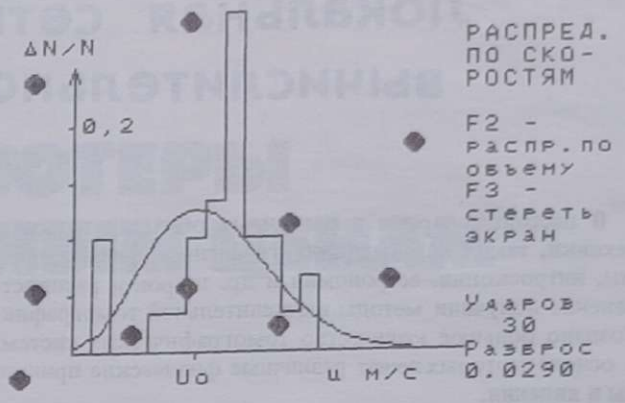


Рис. 1

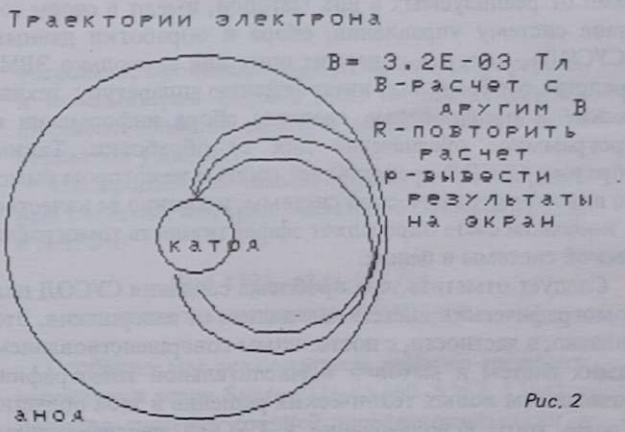


Рис. 2

ЭЛЕКТРИЧЕСТВО И МАГНЕТИЗМ

Некоторые работы посвящены расчету силовых линий электростатического и магнитного полей. Для электростатического поля строятся также эквипотенциальные поверхности. Имеется возможность расчета линий поля точечных зарядов, длинных нитей, прямых токов, соосных витков с током. Расчет производится путем решения дифференциального уравнения силовой линии методом Рунге–Кутта. Проекции векторов напряженности поля на оси координат определяются с помощью принципа суперпозиции. Поля точечного заряда, длинной нити или прямого тока изучаются в курсе общей физики, поле витка с током более сложно и не выражается через элементарные функции. В программе использовано выражение поля витка через полные эллиптические интегралы. В указанных работах учащийся имеет возможность с помощью клавиш управления курсором формировать систему, создающую поле, и выбирать точки, через которые пройдет силовая линия или эквипотенциаль.

Интересна работа по расчету траекторий электрона в поле магнетрона. Упрощенно магнетрон можно представить как вакуумный диод цилиндрической формы, помещенный в однородное магнитное поле, направленное вдоль оси диода. Компьютер рассчитывает траекторию электрона путем решения дифференциального уравнения движения методом Рунге–Кутта без учета пространственного заряда электронного пучка. При расчете учитываются значения и направления тепловых скоростей. В процессе моделирования учащийся может варьировать размеры системы, напряженности электрического и магнитного полей, а также значения и направления тепловых скоростей. На рис. 2 изображены три траектории, различающиеся направлением тепловой скорости электрона, магнитное поле близко к полю отсечки. Приведено также краткое меню возможных действий.

Кроме названных имеются работы по определению сопротивления резистора и ЭДС источника по вольт-амперным характеристикам линейных цепей, моделированию нелинейной вольт-амперной характеристики вольтовой дуги.

Работы по электричеству и магнетизму предворены автоматизированным коллоквиумом. Они могут быть использованы как в вузе, так и в средней школе.

КОЛЕБАНИЯ И ВОЛНЫ

Указанный цикл находится в стадии формирования. На сегодняшний день он содержит две работы.

Колебания ангармонического маятника. В работе задаются положение и начальная скорость математического маятника. В процессе решения дифференциальных уравнений движения методом Рунге–Кутта получаются зависимости скорости и угловой координаты от времени, которые и выводятся на экран в графическом и, по команде учащегося, в цифровом видах. Возможен расчет движения маятника при его отклонении на любые углы, в том числе превышающие 360°. Проводится также разложение зависимости угла от времени в ряд Фурье. В данной работе учащемуся предлагается исследовать зависимость периода и ангармонизма колебаний от амплитуды.

Интерференция встречных волн. На экране дисплея демонстрируются исходная бегущая волна, три ее отражения от передней и задней стенок при различных коэффициентах отражения и расстояниях между стенками, а также суммарная волна. На экране представлены амплитуды каждой из волн. В работе демонстрируются такие эффекты, как возникновение стоячих волн, резонанс. Видна зависимость формы резонансной кривой от добротности системы.

Эксплуатация пакета в течение более полутора лет подтвердила плодотворность предложенной методики. За первые 10–15 мин учащийся полностью осваивал диалог с ПК и в оставшееся время с возрастающим интересом решал предложенные ему задачи. Практика показала устойчивость программ в эксплуатации и их эргономичность.

Локальная сеть ПК в задачах вычислительной томографии

Б. И. ШНЕЙДЕРМАН

В настоящее время в различных областях науки и техники, таких как медицина, геофизика, физика плазмы, интроскопия, астрономия и др. широкое распространение получили методы вычислительной томографии. Создано большое количество томографических систем, в основе которых лежат различные физические принципы и явления.

Все используемые томографические системы, независимо от реализуемых в них методов, имеют в своем составе систему управления, сбора и обработки данных (СУСОД), которая содержит одну или несколько ЭВМ, средства отображения, интерфейсную аппаратуру, технические и программные средства сбора информации и программное обеспечение для ее обработки. Таким образом, СУСОД представляет собой в некотором смысле ядро томографической системы, и именно ее качество в конечном счете определяет эффективность томографической системы в целом.

Следует отметить, что проблема создания СУСОД для томографических систем еще далека от завершения. Это связано, в частности, с постоянным совершенствованием самих систем и методов вычислительной томографии, появлением новых технических решений в этой области. Кроме того, использование в СУСОД перспективных средств вычислительной техники приводит к новым постановкам задач в этой области и требует соответствующих изменений в организации информационных потоков и вычислительного процесса в томографических системах.

Рентгеновская вычислительная томография стала одним из самых мощных средств диагностики в медицине. Современные рентгеновские компьютерные томографы дают изображение самого высокого качества. Если в конце 70-х годов пространственное разрешение 0,35 + + 0,5 мм$^{-1}$ (число пар линий на 1 мм) считалось достижением, то сейчас типичным для большинства коммерческих систем является разрешение 1 – 1,5 мм$^{-1}$. Томограммы, получаемые на современных рентгеновских компьютерных томографах, практически свободны от специфических помех (артефактов), и в некоторых из них изображение реконструируется в процессе сканирования. Высокие требования к надежности и ремонтопригодности обеспечиваются созданием комплекса программ, реализующих функции контроля, тестирования и диагностики. Ниже сформулированы основные требования, предъявляемые к СУСОД для рентгеновских томографических систем:

1. Вычислительный комплекс должен обладать производительностью, достаточной для удовлетворения требований к времени получения изображения (томограммы).

2. С целью обеспечения необходимого быстродействия СУСОД должна содержать операционную систему, максимально "настроенную" на специфику конкретной задачи.

3. В состав СУСОД должны входить аппаратные средства, обеспечивающие сбор информации с детекторов рентгеновского излучения, представление полученной информации в цифровом виде, ее обработку (реконструкцию изображения), а также средства, осуществляющие взаимодействие комплексов и устройств системы.

4. СУСОД должна быть достаточно надежной и удовлетворять требованиям ремонтопригодности, а также иметь необходимые габариты, массу и энергопотребление; кроме того, желательно отсутствие принудительной вентиляции.

5. СУСОД должна предоставлять необходимый сервис врачу-оператору, включая максимально упрощенный диалог с пользователем, архивирование томограмм, банк данных, а следовательно, наличие системы управления специализированной базой данных, предназначенной для выдачи необходимых справок врачу-оператору.

В начале 80-х годов СУСОД томографических систем были построены на основе одной универсальной ЭВМ и необходимых устройств сопряжения. Позднее ведущие фирмы, производящие вычислительные томографы, такие как Siemens (ФРГ), General Electric (США), CGR (Франция), перешли на более совершенную схему, реализующую функции реконструкции изображения с помощью спецпроцессора, разработанного для решения задач вычислительной томографии. В настоящее время эти фирмы серийно выпускают рентгеновские компьютерные томографы, построенные на основе управляющей ЭВМ и спецпроцессора, реализующих математические алгоритмы реконструкции (такие, как предобработка, БПФ или свертка, обратное проецирование). По такой же схеме построены и отечественные рентгеновские компьютерные томографы. Перспективные модели компьютерных томографов, например ТСТ-900S фирмы Toshiba (Япония), строятся на базе СУСОД, содержащей набор конвейерно работающих микропроцессоров с частичным распараллеливанием вычислений, которая обеспечивает время получения одной томограммы порядка 1 с.

Микропрограммная и схемная реализации основных операций реконструкции в структурах со спецпроцессорами позволяют создавать томографические системы, обеспечивающие синтез высококачественных томограмм примерно за 5–30 с.

Современные компьютерные томографы позволяют производить сканирование со скоростью порядка одного слоя в 1 с. Причем объем данных, собираемых за это время, может достичь 1 млн 16–20-разрядных слов, т. е. 2–2,5 Мбайт. Такое количество данных, организованных в 10^3 проекциях по 10^3 отсчетов в каждой из них, позволяет синтезировать томограммы высокого качества с матрицей цифрового изображения 512×512 элементов. При этом выходные данные составляют примерно 0,25 млн 12–16-разрядных слов, или 0,5 Мбайт. В процессе синтеза томографического изображения необходимо произвести $5 \cdot 10^9$ операций для калибровки восстановления (реконструкции) и коррекции данных. Для полного использования измеренных данных и уменьшения искажений, вызванных конечной разрядностью слова ЭВМ, необходимо проводить вычисления с числами с

КОМПЬЮТЕРЫ ПО СНИЖЕННЫМ ЦЕНАМ

С 1976 г. на Западе расширилась торговля подержанными компьютерами. В настоящее время ее оборот превысил 2 млрд. марок ФРГ в год. В основном это касается компьютеров фирмы IBM (свыше 80 %), так как:

- *IBM определяет рынок и имеет самый большой объем продаж*
- *IBM гарантирует специальное обслуживание*
- *Возможна покупка компьютера в США или Японии и продажа его в Европе или в другом месте и наоборот (этот вид услуг уже реализован)*
- *Программные средства, выпускаемые фирмой IBM, – это стандарты, на которые ориентируются все другие производители*
- *Большинство специалистов в области вычислительной техники используют модели фирмы IBM или хотя бы знают эти машины и их программные средства*

Занимаясь с 1978 г. торговлей в США и Европе подержанными компьютерами, наша фирма приобрела большой опыт. Мы поставляем подержанные компьютеры также в Югославию и Венгрию, обеспечиваем Иран системами фирмы IBM, частично перестраивая устройства ввода-вывода (клавиатуру, экран, печатающее устройство) для работы на персидском языке (фарси). Мы можем также адаптировать системы для работы на русском языке (одна такая версия уже демонстрировалась Торговому представительству СССР в ФРГ). Мы являемся самой крупной в Европе специализированной фирмой по подключению печатающих устройств, дисковых ЗУ и других устройств, произведенных различными фирмами, к компьютерам фирмы IBM. На эту услугу существует большой спрос, и мы достигли в этом огромных успехов.

Мы продаем не только новые, но и подержанные системы, которые эксплуатировались от 2 до 5 лет. Это

IBM 4331, 4341, /34, /36
и соответствующая периферия.

Особенно мы рекомендуем широко распространенную в мире System/36 (5360, 5362, 5364) в качестве системы средней производительности, а также в качестве подсистемы больших систем. К настоящему времени продано 250 000 систем IBM/36^{x}, признанных простыми, удобными и эффективными. Для этих систем создано очень много прикладных программ, позволяющих решать всевозможные задачи. Модели System/36 очень просты для обучения, компактны и удобны.

Мы предлагаем пользователям в СССР покупку подержанных систем фирмы IBM, так как:

- *Подержанные системы стоят в 2,5–5 раз дешевле новых*
- *Подержанные системы имеют ту же самую производительность, что и новые*
- *Вместо одной новой можно купить три бывшие в употреблении системы с той же самой производительностью*
- *Износ является минимальным, подержанные компьютеры так же долговечны*
- *Низкая цена позволяет значительно уменьшить капиталовложения*

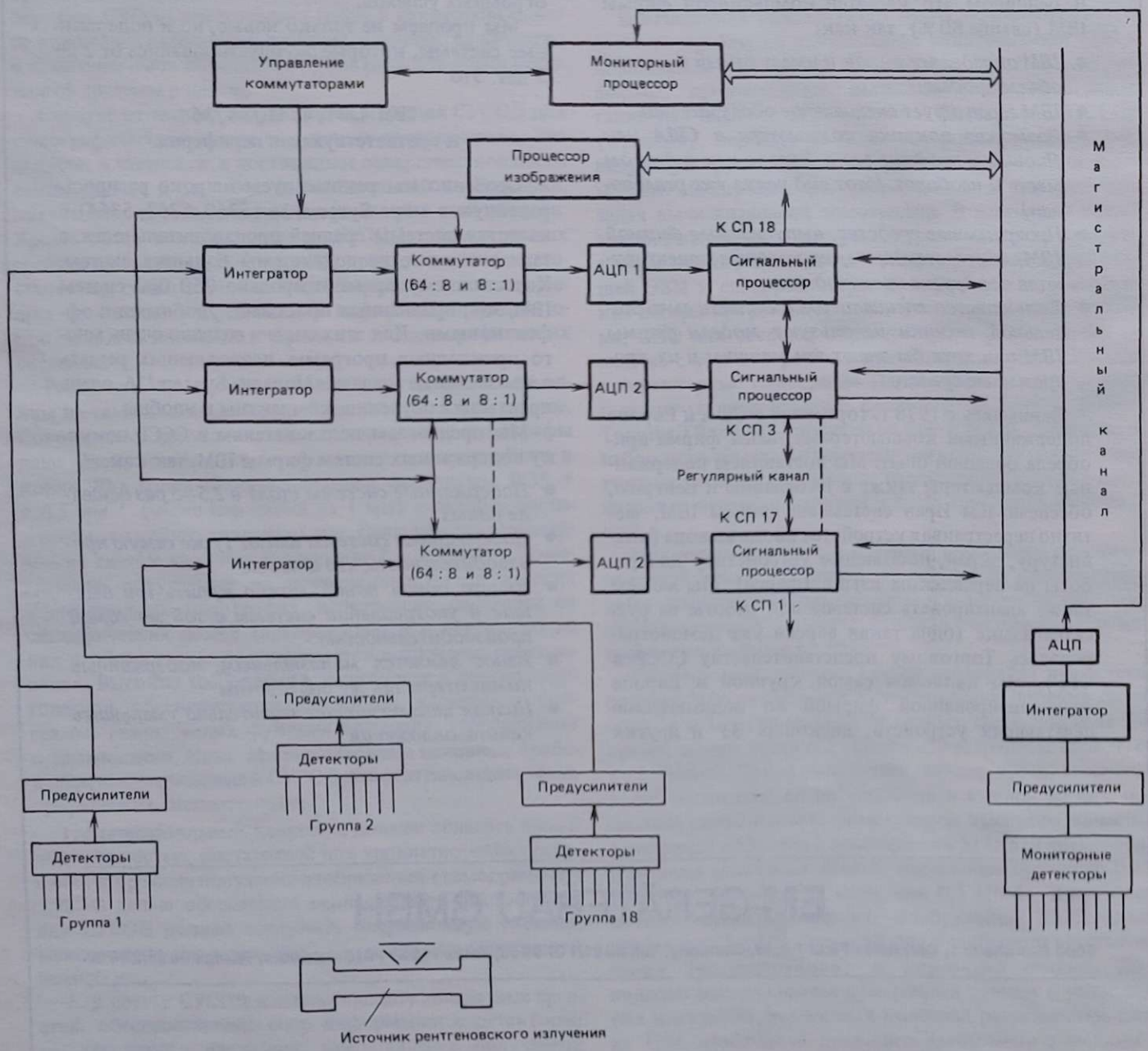
EM-GERÄTEBAU GMBH

6800 Mannheim 1, Carl-Reiss-Platz 7-9, W.-Germany, Tel. 06 21/1 07 96 55, Telex 176 211 815 data com, Telefax 06 21/2 04 37

плавающей точкой или с целыми числами на 32-разрядной сетке. Таким образом, типичное количество операций при получении одной томограммы составляет примерно $5 \cdot 10^9$ Мфлоп. Необходимость обеспечения высокой пропускной способности компьютерного томографа требует провести все необходимые вычисления при получении томограммы примерно за 5 с, т. е. вычислительные средства должны обеспечивать быстродействие примерно 10^3 Мфлоп/с. Такое быстродействие в сочетании с необходимостью использования относительно дешевой вычислительной техники ставит перед разработчиками вычислительно-отображающих комплексов существенные технико-экономические проблемы. Частично трудности могут быть уменьшены при использовании так называемого рутинного режима томографического исследования, при котором собирается полный объем данных, но для реконструкции изображения и его вывода на экран дисплея в виде уменьшенной матрицы размером 256X256 элементов используется примерно четверть собранных данных. В этом случае необходимо произвести примерно 1 млн операций и можно обойтись вычислительными средствами с быстродействием примерно 100 Мфлоп/с.

В современных коммерческих системах такая сложная техническая задача решается с помощью разрабатываемых для этой цели специальных вычислителей. Такие вычислители имеют архитектуру конвейерного типа и позволяют осуществлять три основных этапа синтеза томограмм – предварительную обработку (калибровку и коррекцию данных), фильтрацию (свертку или БПФ), обратное проецирование. Скорость обработки достигает примерно 100–200 проекций/с. Увеличение скорости наталкивается на существенные трудности. Тенденция к уменьшению времени сканирования до долей секунды и получению трехмерных изображений в реальном времени вынуждает разработчиков вычислительно-отображающих комплексов для компьютерных томографов искать другие архитектурные решения.

Структура томографических данных позволяет использовать существенное распараллеливание процесса обработки. Использование для параллельной обработки соответствующих коммерческих спецпроцессоров (на-



пример, таких, как ПС 2000) экономически невыгодно из-за их очень высокой стоимости.

Современные серийные компьютерные томографы, обеспечивая высокое качество изображения, тем не менее имеют один существенный недостаток: высокую стоимость, не всегда доступную для широкого круга возможных пользователей.

В настоящей статье описан другой подход к рассмотренной проблеме, позволяющий существенно снизить стоимость компьютерного томографа. Этот подход основан на распараллеливании входных данных, декомпозиции исходной вычислительной задачи (задачи реконструкции) и построении полного изображения по результатам декомпозиции. Техническая реализация этого подхода основана на использовании в составе вычислительной среды локальной сети ПК.

Суть предлагаемого решения состоит в следующем. Пусть группа из M детекторов разбита на K подгрупп по L детекторов в каждой (M = KL), и $F_h^V(\xi_1^V, \xi_2^V, \ldots, \xi_l^V)$ – результат реконструкции по данным $(\xi_1^V, \xi_2^V, \ldots, \xi_l^V)$, полученным по V-й группе детекторов (в V-м процессоре) с номерами 1, 2, . . . , l в h-м (h = 1, 2, . . . , N) элементе матрицы изображения. Обычно F_h – это яркость одного элемента матрицы изображения на экране устройства визуализации (полутонового дисплея). Изображение, построенное по данным от всех M детекторов, представляет собой линейную комбинацию

$$F_h = \Sigma a_V^h F_h(\xi_1, \xi_2, \ldots, \xi_l),$$

где a_V^h – коэффициент, учитывающий взаимное расположение элемента матрицы изображения и детекторов, входящих в группу с номером V, V = 1, 2, . . . , K.

Приведенная выше схема декомпозиции вычислительного процесса реконструкции позволяет распределить все вычисления, необходимые для получения изображений, между отдельными процессорами в соответствии с предложенной схемой аппаратно-программной реализации СУСОД томографической системы. Ядро этой схемы представляет собой вычислительную сеть, включающую в свой состав сигнальные процессоры для сбора данных, их обработки и синтеза отдельных изображений (вычисление $F_h(\xi)$), процессор для работы с результирующей томограммой (процессор изображения), а также мониторный процессор, осуществляющий функции как управления томографической системой в целом, так и контроля отдельных аппаратных средств. Все процессоры объединены с помощью магистрального канала в локальную вычислительную сеть. В качестве такого канала используется относительно быстродействующая сеть, построенная, например, по принципу сети Ethernet. Такая структура сети обеспечивает высокую гибкость процесса, позволяет перестроить общую схему преобразования информации, адаптируясь к реальным вычислительным мощностям имеющихся процессоров, снижая или увеличивая число детекторов, подключаемых к сигнальному процессору.

Каждый сигнальный процессор (СП) в предложенной схеме предназначен для получения матрицы изображения по информации, поступающей от группы из K детекторов. Так, при применении данной схемы для разрабатываемого рентгеновского медицинского компьютерного томографа из соображения сбалансированности технических средств сбора данных и вычислительных мощностей значение K было выбрано равным 18 при L = 64. Это позволило использовать в качестве сигнальных про-

цессоров ПК типа IBM PC/AT, реализующие необходимую обработку информации (20 с для 64 попарно объединенных каналов) за допустимое время. Результатом работы каждого сигнального процессора является изображение (полученное по данным от группы детекторов), передаваемое затем тем или иным способом в процессор изображения. Для типового варианта объем данных, передаваемых каждым сигнальным процессором, создает нагрузку от 0,5 Мбайт в 1 с на локальную сеть при размере матрицы изображения 50×10^4 элементов.

Матрицы изображения, передаваемые в центральный процессор или по конвейеру в остальные сигнальные процессоры, служат основой для синтеза томограммы в соответствии с приведенной выше формулой.

Предложенная схема СУСОД томографической системы позволяет строить вычислительно-отображающий комплекс на основе однородной вычислительной многопроцессорной среды. Рассмотренные решения были опробованы на экспериментальном образце, характеристики которого оказались близки к расчетным.

Приведенная схема оказывается более эффективной и дешевой, чем упомянутые выше (СУСОД, построенная с использованием спецпроцессоров и управляющей ЭВМ, а также СУСОД на базе универсальной ЭВМ), как с точки зрения ремонтопригодности и удобства эксплуатации, так и по надежности, поскольку отказ одного из сигнальных процессоров приводит лишь к некоторому ухудшению качества результирующего изображения (естественно, для томографической системы с неподвижными детекторами).

При конвейерном синтезе полной томограммы сигнальные процессоры пронумерованы от 1 до K, при этом V-й сигнальный процессор получает промежуточную матV-м процессоре вычисляется матрица изображения

$$S_h^V = F_h^{V-1} + F_h^V,$$

которая передается в $(V + 1)$-й процессор и т. д. Таким образом обрабатываются данные от всех детекторов (электронных каналов).

При такой организации взаимодействия удобно иметь в V-м $(V > 1)$ сигнальном процессоре память для хранения двух матриц: одной, полученной от $(V - 1)$-го процессора, и второй, полученной от собственных электронных каналов (детекторов). Хотя суммарный объем пересылок информации для обеих схем одинаков, радиальная схема является существенно несимметричной с точки зрения информационного обмена: центральный процессор осуществляет прием K матриц изображения, в то время как каждый из сигнальных процессоров осуществляет передачу только одной матрицы изображения. В конвейерной схеме каждый процессор осуществляет один прием и одну передачу матрицы изображения (кроме 1-го сигнального процессора и процессора изорицу изображения от $(V - 1)$-го процессора, складывает ее с матрицей, построенной по собственным данным, и передает результирующую матрицу в $(V + 1)$-й сигнальный процессор. Таким образом, в последнем K-м сигнальном процессоре будет построена результирующая матрица изображения, которая и передается в процессор изображения, а из него через дисплейную систему на полутоновый монитор. Пусть F_h^{V-1} – элементы промежуточной матрицы изображения, полученной V-м процессором от $(V - 1)$-го процессора. В этом случае в бражения, которые осуществляют только передачу и только прием соответственно матрицы изображения).

Важно отметить, что распараллеливание всего процесса обработки информации в компьютерном томографе на большое число сигнальных процессоров позволяет снизить требования к быстродействию отдельных процессоров даже при более высокой скорости обработки данных во всей системе по сравнению с конвейерными томографическими спецпроцессорами, а это, в свою очередь, позволяет использовать в качестве сигнальных процессоров микропроцессорные комплекты, в том числе транспьютеры. Низкая стоимость элементной базы, необходимой для распараллеливания, позволяет решать проблему построения СУСОД более экономичными путями по сравнению с обеспечиваемыми конвейерными спецпроцессорами, получая при этом существенный выигрыш в гибкости, адаптируемости и вычислительной мощности системы в целом.

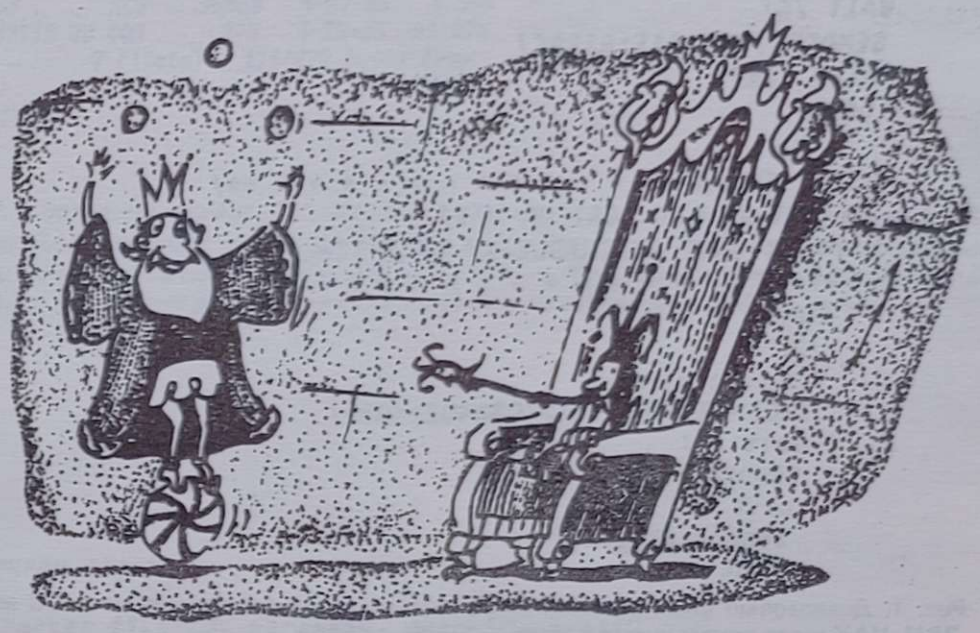
В ближайшей перспективе можно ожидать, что функции сбора данных, управления и контроля томографических систем будут полностью осуществляться с помощью большого числа децентрализованных микропроцессоров (микропроцессорных контроллеров), а для обработки информации (реконструкции) будут применяться спецпроцессор либо предложенные в настоящей статье параллельные процессоры, объединенные в локальную сеть. При этом каждый процессор должен обрабатывать информацию, поступающую от своей группы детекторов.

Для еще большего увеличения быстродействия томографических систем, а также для решения задач построения трехмерных изображений целесообразно оснащать каждый процессор своим "ускорителем" (обратным проектором).

Программная система для подключения персональных компьютеров к ЭВМ типа VAX

ТОМАС МАДРОН, ДЕЙВИД МОЛЬТА

Предприятиям и организациям, озабоченным поиском путей соединения персональных компьютеров с мини-ЭВМ фирмы DEC, фирма Datability Software Systems предлагает средства удаленного доступа с возможностями, значительно превосходящими простую эмуляцию терминала.



Владыка–раб – вот отношение, которое лучше всего описывает широко распространенную ситуацию, когда большие вычислительные системы играют роль главной ЭВМ (ГВМ) для персональных компьютеров (ПК). При этом ПК подвергается "программной лоботомии", так что может показаться глупым терминалом и склоняет голову перед богатством возможностей большой и мини-ЭВМ. Хотя лучшие терминальные эмуляторы и обеспечивают довольно мощные средства взаимодействия с центральной вычислительной системой, но даже операции, которые должны быть простыми, например пересылка файлов, нередко становятся труднодоступными.

К счастью, наметилась перспектива выхода из этого мрачного положения благодаря новому урожаю программных пакетов связи, способствующих перемене ролей. Эти программы превращают ГВМ в некое периферийное устройство, позволяющее ПК захватывать свой кусок из сотен мегабайтов дисковой памяти или даже запускать прикладные программы на ГВМ с помощью стандартных команд ОС DOS (см. статью о большой ЭВМ в качестве периферийного устройства "The Mainframe as Peripheral" в июньском номере журнала "PC World" за 1986 г.).

Большие вычислительные системы предлагают не только роскошь обильного дискового пространства, но также бдительно действующие службы резервного копирования и обеспечения безопасности и сохранности данных. В результате данные ПК, записанные на ГВМ, с большой вероятностью окажутся там и в тот момент, когда вам понадобятся. И это весьма отличается от положения, которое бытует на некоторых ПК.

Средства удаленного доступа (Remote Access Facility – RAF) фирмы Datability – это система связи, разработанная для совместного использования с широко распространенным семейством ЭВМ фирмы Digital Equipment Corporation (DEC). Используя диски большой ЭВМ, система RAF принимает и сохраняет файлы ПК в формате виртуального диска, который пользователями ПК воспринимается как совместимый с ОС DOS.

Система RAF и ей подобные системы разрабатываются с целью обеспечить персонал предприятий и организаций, обслуживающий информационные системы, столь необходимым ему инструментарием, имеющимся на ГВМ. Как правило, эти программы не ориентируются на какое-либо конкретное приложение, а предоставляют обобщенные каналы связи для соединения микро-ЭВМ с большими или мини-ЭВМ. Обычно они работают совместно с операционной системой, такой как VMS на ЭВМ VAX фирмы DEC, и обеспечивают канал связи между программным пакетом на ГВМ и сопутствующей ему программой, работающей на ПК.

Обобщенные каналы связи облегчают также решение другой проблемы, стоящей перед многими предприятиями и сопряженной с необходимостью, чтобы персонал для решения определенной задачи был способен работать попеременно на ПК, мини-ЭВМ и большой ЭВМ. Большинство сотрудников далеки от таких фокусов; в идеале они должны работать с одним, стандартным, интерфейсом.

ПЯТЬ ФУНКЦИЙ СИСТЕМЫ

Система RAF предоставляет средства пересылки файлов между ПК и мини-ЭВМ и манипуляции файлами, позволяет выполнять прикладные программы на ЭВМ VAX в операционной среде, которая выглядит, как ОС DOS, и обеспечивает использование ПК в качестве внешнего интерфейсного процессора для программных модулей на VAX, собранных в задачи, слишком большие для непосредственного выполнения на ПК.

Точнее говоря, RAF позволяет ПК выполнять пять различных видов взаимодействия между микро- и мини-ЭВМ: автоматическое установление контакта и вхождение в ОС удаленной ЭВМ VAX; непосредственный доступ к файлам VAX; выполнение на VAX самостоятельных про-

```
LOGIN: !remote monitor
TIMER 10
SEND /<13>/
WAIT /#/
CALL:  SENDcr /LOC 1/
WAIT /#/ /UNKNOWN COMMAND/
IF 2 GOTO CALL
ECHO:  SENDCR /ECHO OFF/
WAIT /#/
SENDCR /CALL DEC/
WAIT /CALL COMPLETED/
IF TIMEOUT GOTO E1
SEND /<13>/
WAIT /Username:/
SENDCR /USER/
WAIT /Password:/
SENDCR /PWD/
WAIT /$/
SENDCR /RUN RAF:RAFPC/
WAIT /RAF/
DONE

E1:  PRINT /* * * * * * * * * * * * * **/
PRINT /  LAN PORT ON VAX/
PRINT /NOT AVAILABLE AT THIS TIME/
PRINT /* * * * * * * * * * * * * * * */
DONE

LOGOUT: send
/<03><03><03><03><03><03><03>/
wait /$/
SENDCR /LOG/
wait /#/
sendcr /loc 17/
DONE
```

Рис. 1. Диалоговый файл системы RAF, осуществляющий вхождение в ОС на ЭВМ VAX через локальную сеть фирмы Sytek

граммм; удаленный вызов подпрограмм на VAX; доступ к нескольким ЭВМ VAX, объединенным в кластер.

Поскольку пользователи время от времени должны взаимодействовать с ЭВМ VAX напрямую, система RAF обеспечивает также режим полной эмуляции терминала VT100 фирмы DEC. При этом возможно оперативное переключение с терминального режима на режим работы прикладной программы на ПК и обратно. Связь системы RAF с ЭВМ VAX осуществляется через стандартный последовательный интерфейс RS-232C со скоростью до 19 200 бит/с или через локальную сеть типа Ethernet (к моменту написания данной статьи версия для локальной сети еще не была проверена).

УСТАНОВКА СИСТЕМЫ

Установка системы RAF – относительно простой двухэтапный процесс, занимающий около получаса. На первом этапе система RAF устанавливается на ГВМ VAX с помощью стандартной утилиты установки ОС VMS, называемой VMSINSTALL.

На ГВМ должна работать ОС VMS версии 4.0 или более поздней версии и должно быть свободно не менее 5000 512-байтовых блоков; 1600 блоков выделяются постоянно. Система RAF размещается в общем каталоге, где она доступна любому пользователю. Она также может быть установлена на многоузловом кластере из ЭВМ VAX. Если пользователь намеревается осуществлять доступ к системе RAF, он должен освободить в своем корневом каталоге минимум 2000 блоков.

При работе с утилитой установки система RAF запрашивает имя каталога, в котором она должна размещаться постоянно (если необходимо, она создает новый каталог). Вслед за этим система напоминает вам, что необходимо включить команды запуска системы RAF в стартовый командный файл операционной системы, чтобы гарантировать общесистемный доступ к системе RAF.

Следующий этап заключается в выполнении на ПК пакетного файла INSTALL для определения исходного содержимого диска, определения конфигурации порта связи и задания скорости передачи. Программа установки создает также файл конфигурации CONFIG.SYS, содержащий необходимые системе RAF драйверы устройств.

Чтобы активизировать систему RAF, вы должны перезапустить ОС DOS. Операционная система прочитает файл конфигурации и назначит системе RAF устройства от D: до S: в качестве виртуальных устройств на ГВМ VAX (если в файле конфигурации присутствует команда электронного диска, то первым устройством для системы RAF будет E:). Размер виртуального устройства ограничивается только полным объемом дискового пространства, выделенного для вашего учетного номера; потенциально это сотни мегабайтов.

Система RAF защищена от копирования, однако ключевого диска не требуется. Схема защиты позволяет вам установить программу дважды. К счастью, вы можете установить систему RAF на жестком дике и затем удалить ее для переноса на другую ЭВМ столько раз, сколько пожелаете.

НАЧАЛО РАБОТЫ С ЭВМ VAX

Войти в ОС на ЭВМ VAX вы можете автоматически с помощью диалогового файла (или файла сценария) системы RAF, содержащего последовательность управляющих операторов, которые осуществляют доступ к ГВМ при минимальном вмешательстве пользователя. Сложность такого файла варьируется в зависимости от типа прикладной системы, с которой вы работаете, и от нужной вам степени автоматизма. На рис.1 показан листинг диалогового файла, написанного для асинхронной последовательной связи с помощью широкополосной локальной сети LocalNet 20 фирмы Sytek.

Диалоговый файл может выполняться с помощью утилит LOGIN.COM и LOGOUT.COM системы RAF, относящихся к частям файла сценария, помеченным метками LOGIN: и LOGOUT: соответственно. Каждый такой сегмент должен заканчиваться оператором DONE. Система RAF отвечает на подсказки VAX "login" (вхождение) и "password" (пароль) и затем выполняет на VAX программу RAFPC, устанавливая тем самым сервер системы RAF на ЭВМ VAX.

После завершения выполнения диалогового файла вы имеете доступ к устройствам от D: до S:. С помощью команды изменения каталога CD ОС DOS вы можете изменить ка-

талог, ассоциированный с конкретным удаленным устройством системы RAF. Косая черта (/) перед командой не обязательна.

Например, из корневого каталога ПК вы можете выдать команду CD D:ПОЛЬЗОВАТЕЛЬ[УЧЕТНЫЙ_НОМЕР_1] и команду CD E: ПОЛЬЗОВАТЕЛЬ[УЧЕТНЫЙ_НОМЕР_2] (где ПОЛЬЗОВАТЕЛЬ [УЧЕТНЫЙ_НОМЕР_1] и ПОЛЬЗОВАТЕЛЬ [УЧЕТНЫЙ_НОМЕР_2] — это каталоги на ЭВМ VAX). И тогда в таких командах ОС DOS, как TYPE и COPY (распечатать на экране и скопировать), нужно будет ссылаться только на устройство D: или E:. Для распечатки, например, текстового файла из удаленного каталога вы можете выдать команду TYPE E:XYZ.TXT.

Чтобы скопировать файл с ЭВМ VAX на ПК, вы можете выдать команду COPY E:XYZ.TXT C:\TEXTFILE\XYZ.TXT. Эта команда поместит файл XYZ.TXT в DOS-каталог \TEXTFILE на устройстве C: –жестком диске на ПК.

Как только определено виртуальное устройство системы RAF, прикладные программы на ПК могут осуществлять доступ к файлам на ЭВМ VAX и выполнять запись непосредственно в VAX-каталог. Вы можете записать файлы) например, системы Wordstar), просто сохранив их на устройстве D: или E:. С помощью такой утилиты, как SideKick, вы можете непосредственно редактировать текстовые файлы, расположенные на ЭВМ VAX. Преобразование файлов, порожденных на ПК, и файлов в коде ASCII, порожденных на VAX, выполнять не нужно. Конечно, программы, работа которых зависит от аппаратуры (подобные Norton Utilities), на виртуальных устройствах системы RAF могут оказаться бесполезными.

РАБОТА – ПРЯМО КАК НА VAX

Пользователи ПК могут осуществлять доступ к прикладным программам на VAX, как если бы это были локальные программы на ПК. Например, нет проблемы в том, чтобы выполнить инструкцию типа VTX, вызывающую на ПК систему видеотекса с ЭВМ VAX.

Для выполнения этой задачи система RAF предлагает два простых метода. Прежде всего утилита REMOTE системы RAF может создать небольшой командный файл .COM, который подставит нужную команду ЭВМ VAX. Формат команды REMO-

```
C:\RAF>vdir


Directory $10$DRA0:[SY12]

COM1.DAT;1          LOGIN.COM;1         PC.DIR;1            RAF22280F5E01.DAT;1
RAF2248130201.DAT;1 UP.XCD;1            WRITE_OUT.COM;7     WRITE_OUT.COM;6
WRITE_OUT.COM;4

Total of 9 files.
C:\RAF>dir d:

 Volume in drive D is $10$DRA0:
 Directory of  D:\

COM1     DAT       20   5-27-86   4:14p
LOGIN    COM     1436   5-22-86   1:18p
PC           <DIR>      9-02-86   5:01p
RAF22280 DAT   153600   9-21-86  12:27p
RAF22481 DAT   153600   9-21-86  12:01p
UP       XCD    95868   4-27-86   4:56a
WRITE_OU COM      878   5-26-86  11:57a
       7 File(s)   1114878 bytes free

C:\RAF>-
```

Рис. 2. Состояние экрана ПК. Когда на ПК выполняется удаленная команда VDIR, система RAF посылает на VAX команду DIR; результаты показаны в верхней части экрана. Когда вводится команда DIR D:, система RAF преобразует ее в команду DIR для VAX и порождает распечатку DOS-каталога, показанную в нижней части

```
program rtest
c
character file*20, param*50, prog*50, sys*50
c
c  Initialize the program for RAF
c
call raffor
c
c  check for a RAF link
c
10 call rafok(iflag)
c
c  if the RAF link is not active, then login using the
c  'conversation' file 'lan.cnv'.
c
if (iflag .ne. 1) then
file = 'c:\raf\lan.cnv'
param = ' '
call raflgi (file,iflag,param)
endif
c
c  execute a VAX command (in this case, DIR)
c
sys = 'DIR '
call rafrun (iflag, prog, sys)
if (iflag .ne. 1) then
write (*,'('' Remote program/command ended in Error'')')
go to 20
endif
c
write (*,'('' VAX application finished'')')
c
c  End this program's link with RAF
c
20 call rafend
end
```

Рис. 3. Фортран-программа на ПК, которая использует подпрограммы из библиотеки RAFLIB.LIB и выполняет удаленную программу на ЭВМ VAX

ТЕ прост: REMOTE имя_файла "VAX-команда".

Чтобы система видеотекса воспринималась как локальная прикладная система, вам нужно только выдать команду REMOTE VTX "VTX".

Если полученный в результате файл VTX.COM размещается в каталоге с полным именем файла, определенным в ОС DOS, то выполнение команды VTX из любого каталога ОС DOS приведет к появлению на экране ПК программы системы видеотекса с ЭВМ VAX.

На рис.2 показано состояние экрана дисплея ПК, которое иллюстрирует использование команды REMOTE. Файл VDIR.COM был создан с помощью команды REMOTE VDIR "DIR" системы RAF. Когда на ПК выполняется команда VDIR, система RAF посылает на VAX команду DIR. В другом варианте вы можете ввести команду DIR D:. Система RAF преобразует ее в команду DIR ЭВМ VAX и распечатает на экране каталог в формате ОС DOS, как показано в нижней части рис.2.

БИБЛИОТЕЧНЫЕ СРЕДСТВА

Система RAF предоставляет набор подпрограмм, собранных в библиотеку RAFLIB.LIB. Эти подпрограммы позволяют специально написанным программам на ПК осуществлять доступ к средствам и возможностям, имеющимся на ЭВМ VAX, куда входят различные языки и компиляторы, включая Фортран, Паскаль, деловой Бейсик, компилятор Бейсика фирмы Microsoft, компилятор Бейсика фирмы IBM и компилятор Си фирм Lifeboat Associates и Lattice.

Эти подпрограммы могут, например, проверять состояние канала связи, чтобы убедиться, активна ли система RAF, и если нет, то могут войти в ОС на ЭВМ VAX, используя предварительно определенный диалоговый файл (см. рис.3, где показан листинг программы на Фортране).

При более оживленной буферизации обменов между ПК и VAX система RAF может вызывать подпрограммы на VAX прямо из прикладных программ, прогоняемых на ПК. Эта возможность особенно важна для обобщенных каналов связи, поскольку доступ к многим администраторам баз данных на больших вычислительных системах происходит через вызовы подпрограмм, которые выдаются из программ, выполняющих функции пользовательского интерфейса. Этот подход существенно, отличается от таких пакетов на ПК, как dBase III, в котором база данных строится и доступна с помощью некоторого прикладного языка.

Используя возможности системы RAF, программисты предприятия могут написать такие программы, которые реализуют дружественный интерфейс с большими базами данных. Аналогично решение вычислительных задач, выходящих за пределы возможностей ПК, может быть перепоручено ЭВМ VAX, причем получаемые результаты будут возвращаться прикладной программе, работающей на ПК. Система RAF обеспечивает, например, средства, позволяющие ПК-программе на Фортране вызывать VAX-подпрограмму на Фортране (обе программы должны быть скомпилированы до запуска в работу системы RAF).

Процесс построения такой программы облегчается удобной в использовании, управляемой меню утилитой системы RAF под названием RAFSL. Она позволяет вам сообщить системе RAF, какие и где имеются удаленные и местные подпрограммы, после чего утилита создает местную подпрограмму с тем же именем, что и удаленная подпрограмма, и порождает код, необходимый для осуществления связи с VAX.

Эта утилита может также создать окончательную версию VAX-подпрограммы в файле типа .EXE, запросив имя подпрограммы и имя библиотеки, в которой она располагается. В результате получается законченная программа, которая частично работает на ПК, частично – на VAX и выполняет то, чего желает от нее программист.

НЕДОСТАТКИ

Хотя очевидных ошибок система RAF не обнаруживает, некоторые аномалии в ее работе все же должны быть исправлены.

Например система RAF завладевает всеми именами устройств от последнего физического устройства до устройства S:. При работе с DOS версии 3.10 и последующих версий это обстоятельство значительно снижает полезность команды SUBSTitute (подставить, заменить) этой ОС, когда полные имена файлов определяются с использованием виртуальных дисков на жестком диске ПК. Столь же неудобно такое положение и на сети из ПК.

Назначение специальных клавиш в качестве имен устройств делает возможным ограниченное использование дополнительных имен устройств, но большинству пользователей это помогает мало. Более продуктивный подход состоял бы в том, чтобы дать пользователям возможность ограничивать число виртуальных устройств, отдаваемых системе RAF (как объявлено фирмой, выпуск 1.7 системы RAF, подготовленный уже в момент сдачи статьи в печать, частично решает эту проблему, позволяя пользователю выделять системе RAF 8 или 16 виртуальных устройств).

Кроме того, в системе RAF нет легкого способа изменять во время работы скорость передачи или порт связи. Чтобы произвести такие изменения, приходится возвращаться к программе INSTALL, хотя идеальной была бы возможность выполнять изменения из терминального эмулятора (изменения могут управляться диалоговым файлом или, по данным фирмы для выпуска 1.7, с помощью подпрограммы из библиотеки RAFLIB. LIB).

Если при вызове удаленной программы или команды, построенной с помощью процедуры REMOTE, канал связи системы RAF неактивен, восстановление системы может оказаться неудобным. Система RAF в течение нескольких минут может "пребывать в рассеянности", прежде чем заметит, что канал связи неактивен, и попросит вас сделать аварийное завершение или новую попытку. В некоторых случаях теряется подсказка ОС DOS, что приводит к необходимости перезагрузки операционной системы.

Фирма предполагает исправить процедуру REMOTE, чтобы она сначала проверяла состояние канала связи и затем обязательно возвращалась к DOS. Хотя от этих проблем и можно уйти с помощью некоторых программистских ухищрений, все-таки это снижает гибкость системы RAF.

Документация системы RAF невелика по объему и понятна, однако подобно руководствам других фирм местами проработана недостаточно, особенно когда объясняется, как строить прикладные системы, в которых первичная программа на ПК вызывает подпрограммы на VAX. Еще пример. Хотя система RAF и поддерживает Фортран версии 3 фирмы Microsoft, документация об этом умалчивает. Так же не оттенена воз-

можность использования альтернативных имен для некоторых подпрограмм из библиотеки RAFLIB.LIB, позволяющая приспосабливаться к изменениям соглашений о поименовании. Фирма сообщает, что в документации на систему RAF выпуска 1.7 все эти неясности устранены.

ПОСЛЕДСТВИЯ МЕДЛЕННОЙ РАБОТЫ АППАРАТУРЫ СВЯЗИ

Как и любая программная система, рассматривающая большую или мини-ЭВМ как файловый сервер для ПК, асинхронная установка системы RAF не очень-то подходит для задач, в которых скорость играет первостепенную роль. Например, при работе в режиме эмулятора терминала на выделенной сети типовая скорость передачи в линии связи для асинхронных устройств равна 9600 бит/с. Пересылка с помощью системы RAF на VAX двоичного файла размером 100 Кбайт занимает более 3 мин. Вы говорите, что можете прожить с такой скоростью? А как насчет резервного копирования жесткого диска в 20 Мбайт? Можете ли вы выделить на это 10 ч?

Если вы намереваетесь оставить ПК без присмотра для выполнения резервного копирования глубокой ночью, то это может у вас получиться . Однако в большой организации выполнение подобного копирования оптом может перегрузить как ГВМ, так и систему связи.

Собственный дуплексный протокол системы RAF с исправлением ошибок и алгоритмами сжатия данных несколько повышает скорость передачи. Правда, копирование файлов при скорости в линии связи 9600 бит/с проигрывает по сравнению с выполнением локальной команды COPY операционной системы DOS, однако RAF демонстрирует значительное превосходство над другими протоколами пересылки файлов с исправлением ошибок. Так, пересылка на VAX текстового файла размером 16 Кбайт при использовании системы Kermit занимает 60 с, копирование того же файла системой RAF – только 20 с.

Очевидное долговременное решение проблемы состоит в переходе к применению более быстродействующей аппаратуры связи, напрямую соединяющей большую вычислительную систему с высокоскоростной локальной сетью наподобие Ethernet, которую фирма DEC считает предпочтительной для соединения своих машин VAX с периферийными ЭВМ. Поддержка этой сети включена в выпуск 1.7 системы RAF. Когда в ПК вставлена плата для этой сети, пересылка файлов с VAX на ПК может превосходить по скорости обмен ПК с собственным гибким диском. Фирма Datability объявила, что система RAF пересылает файлы через сеть Ethernet со скоростью 100 Кбайт/с.

СТАБИЛЬНОСТЬ И ПОДДЕРЖКА

Если оставить в стороне упомянутые оговорки, то система RAF предстанет как мощное и хорошо разработанное средство связи ПК с VAX. Несмотря на разнообразие особенностей, система легка в использовании. По сделанным оценкам она одинаково хорошо работает с ПК типа XT и их аналогами, а также на совместимом с ПК типа AT ПК Business Pro фирмы Texas Instruments.

Инженеры, обслуживающие бесплатную линию связи для обеспечения технической поддержки фирмы Datability, при ответах на вопросы демонстрируют глубокое знание дела. И хотя процесс формирования цен на программные пакеты для связи микро- и мини-ЭВМ обнаруживает тенденцию к усложнению (из-за влияния производителей как ПК, так и VAX), полная стоимость системы RAF представляется конкурентоспособной.

Система RAF и аналогичные программные продукты обещают пользователям ПК простой и гибкий интерфейс с большими системами. По мере все большего распространения распределенной обработки программные системы будут предлагать все более развитые средства для интеграции ПК и больших ЭВМ.

ГДЕ КУПИТЬ RAF:

Datability Software Systems, Inc.
322 Eighth Ave.
New York, NY 10001.
800/342-5377

Цены. Копия RAF для одного ПК 395 дол. Лицензия на партию копий для 50 ПК 15 000 дол; дополнительные копии по 125 дол за каждую. Копия для MicroVAX I 2500 дол. Копия для VAX 8800 25 000 дол. Копия для DECsystem-20 10 500 дол.

Требования для установки: 256 Кбайт памяти, ОС DOS версии 2.0 или более поздних версий.

Копия защищена.

Способ описания данных в системе автоматизации эксперимента

А. Г. КОГАН, Р. В. КОСТИН, С. А. ПЛАТОНОВ

В настоящее время одной из важных областей применения ПК является автоматизация научных исследований и, в частности, автоматизация экспериментов. Особенностью систем автоматизации эксперимента на ПК является использование в таких системах существенно различных аппаратных устройств сбора данных, а также наличие большого числа программных средств обработки и анализа экспериментальной информации. Таким образом, становится очевидной необходимость стандартизации записи экспериментальных данных, осуществляемой системами автоматизации эксперимента на базе ПК, с целью обеспечения "совместимости по данным" при использовании различного математического обеспечения обработки и представления данных на ПК.

Основной проблемой при попытке такой стандартизации является обеспечение независимости процедур обработки данных от условий эксперимента (например, от количества экспериментальных данных, порядка их записи в файл, определяемого порядком съема данных, и т. д.). Вместе с тем слишком жесткий стандарт на запись данных в БД может вынудить экспериментатора к перекодировке полученных данных в требуемый (и, возможно, неестественный для данного эксперимента) формат.

Возможное решение – допустить различные формы записи файла экспериментальных данных, поместив информацию о том, как записан конкретный файл, в специальный дескриптор, добавляемый к каждому файлу экспериментальных данных. Такой дескриптор, названный нами ST-дескриптором, используется в настоящее время в ряде систем автоматизации эксперимента на базе PC/XT/AT-совместимых ПК в ИРЭ АН СССР. Он позволяет описывать файлы данных, представляющие собой массивы произвольной размерности, и налагает минимум ограничений на структуру самих данных, что позволило создать дескрипторы для уже существующих файлов экспериментальных данных без их модификации.

СТРУКТУРА ST-ДЕСКРИПТОРА

В процессе измерений переход от одного измерения к другому соответствует изменению какого-либо параметра эксперимента (например, повторное измерение производится по прошествии некоторого времени или при изменении положения измерителя и т. п.). Таким образом, определена функциональная зависимость измеряемых физических величин от параметров эксперимента. Файл экспериментальных данных – это последовательность измеренных значений. ST-дескриптор позволяет описывать разнообразные последовательности такого вида. Он содержит информацию о параметрах эксперимента (координатах), об измеряемых величинах (функциях), их областях определения и о порядке перебора функций и координат в процессе измерения, или, что то же самое, о порядке расположения измеренных значений в файле данных.

S-часть дескриптора содержит описание диапазонов изменения всех выделенных в эксперименте координат и функций. Значения координат и функций в дескрипторе представлены целыми числами (см. таблицу), что является естественным, поскольку координаты принимают лишь конечное число значений, а функции преобразуются к целочисленным значениям в процессе измерения (например, при аналого-цифровом преобразовании). Однако при интерпретации экспериментальных данных бывает необходимо преобразовывать эти целочисленные значения в истинные (имеющие физический смысл). В S-часть дескриптора включено описание таких преобразований.

Поля ST-дескриптора

Имя поля	Число байт	Пояснение	Тип поля
NCOORD	2	Число координат	Целый
		Для каждой координаты	
CNAME	16	Имя	Символьный
MINC	4	Минимальное значение	Целый
MAXC	4	Максимальное значение	То же
FNC	2	Номер преобразования	"
PC	32	Параметры преобразования	Зависит от поля FNC
NFUNCT	2	Число функций	Целый
		Для каждой функции	
FNAME	16	Имя	Символьный
MINF	4	Минимальное значение	Целый
MAXF	4	Максимальное значение	То же
FNF	2	Номер преобразования	"
PF	32	Параметры преобразования	Зависит от поля FNF
NBITF	1	Число бит для представления функции	Целый
STF	1	Форма записи	То же
TREE	Перем.	Описание дерева	Строка

T-часть описывает, от каких координат зависит каждая из измеряемых функций и в каком порядке они перебираются при записи измеренных значений в файл. Эта информация может быть представлена в виде дерева следующего вида:

1. Каждый лист помечен именем функции. При этом все листья помечены различными именами.

2. Каждый узел, не являющийся листом или корнем, помечен именем одной из координат.

3. Множество меток, встречающихся на пути от корня к листу с меткой f, совпадает с множеством координат, от которых зависит функция f.

Порядок записи значений функций в файле будет определяться последовательностью, порождаемой при обходе дерева по следующей рекурсивной процедуре: для каждого узла фиксируется минимальное значение соответствующей координаты и посещается самое левое поддерево, затем следующее вправо, и т. д., после чего координата увеличивается на единицу, и процесс повторяется до тех пор, пока она не достигнет максимального значения. Каждое посещение листа добавляет к порождаемой последовательности значение соответствующей функции при текущих значениях координат, т. е. дерево обходится в обратном порядке. T-часть представляет собой описание такого дерева в виде строки символов, состоящей из меток узлов и скобок.

Поля, входящие в ST-дескриптор, их длины и форматы записи представлены в таблице. Все символьные переменные записываются в коде ASCII. Отрицательные числа представляются в двоично-дополнительном коде. Поясним смысл некоторых полей.

FNF. Нами в настоящее время используются линейное, квадратичное, экспоненциальное, логарифмическое преобразования и некоторые другие. Этот набор существенно зависит от области экспериментальных исследований, поэтому мы не решаемся навязать какую бы то ни было стандартизацию номеров преобразований. Отметим лишь, что запись нуля в это поле обозначает отсутствие какого бы то ни было преобразования.

STF. Если число разрядов для представления функции не кратно 8, то считается, что ее значения записаны в поле минимальной длины, содержащем достаточное для их записи число байтов. Форма записи определяется нулевым (младшим) битом STF и будет

правой: 0000xxxxxxxxxxxx , если этот бит равен 0,

и

левой : xxxxxxxxxxxx0000 , если он равен 1.

Старший бит — Младший бит

Первый бит переменной STF определяет, являются ли значения функции знаковыми (бит 1 = 1) или беззнаковыми (бит 1 = 0).

TREE. Строка пишется сразу после описания координат и функций не содержит пробелов и вместо имен координат и функций содержит их порядковые номера (в соответствии с последовательностью их описания в S-части). Эти номера пишутся в ASCII-коде по байту на номер. Порядок номеров такой: 1 2 3 . . . 9 A B C D E . . . Z.

ПРОГРАММНАЯ ПОДДЕРЖКА ST-ДЕСКРИПТОРОВ

ST-дескриптор позволяет определить, в каком месте файла экспериментальных данных расположено значение произвольной функции, измеренное при заданных значениях координат. Для этого, однако, требуются некоторые вычисления. Поэтому целесообразно иметь стандартные процедуры, выполняющие эти вычисления. С этой целью нами был реализован пакет подпрограмм для PC/XT/AT-совместимого ПК, облегчающий работу с ST-дескриптором при программировании на языке Си. На рисунке приведен текст файла ST.H, содержашего прототипы этих подпрограмм. Опишем те из них, реализация которых нам показалась целесообразной в первую очередь.

```
typedef int node; /* тип узел */
#define NULLNODE (node)-1 /* несуществующий узел */
  /* тип описание координаты */
typedef struct \
       { unsigned char name[16]; long min,max; \
           int f; unsigned char k[16];} chdr;
  /* тип описание функции */
typedef struct \
       { unsigned char name[16]; long min,max,f; \
         int f; unsigned char k[32],b,t;} fhdr;

void st_begin(); /* начать работу */
void st_build(chdr* coords,int ncoor,
              fhdr* funct ,int nfunc,
              void* tree  ,int treelength,
              int* err);
void st_get(char filename[], int *err);
void st_put(char filename[], int *err);

int st_nc(),st_nf(),st_nt();
chdr* st_ac(); fhdr* st_af(); unsigned char * st_at();
long st_offset(node), st_step(node); int st_card(int c);
node st_root(), st_lson(node), st_father(node), st_next(node),
     st_nodef(int fnum), st_nodec(int fnum, int cnum);
int st_cnum(char * name), st_fnum(char * name);
```

Подпрограмма st_get вводит ST-дескриптор из заданного файла, проверяя его корректность. Подпрограмма st_build позволяет построить дескриптор. Ее входными параметрами являются массив с описанием координат, массив с описанием функций, строка с описанием дерева. В результате получается ST-дескриптор либо сообщается об ошибке, если его не удалось построить.

Остальные подпрограммы и функции сообщают информацию о введенном дескрипторе, представляя ее в виде, удобном для последующей обработки файла данных, описанного этим дескриптором. Функции st_nc, st_nf, st_nt возвращают число координат, число функций и длину дерева соответственно. Функции st_ac, st_af, st_at возвращают начальный адрес массива описателей координат, функций и адрес строки с деревом, что позволяет работать с данными структурами непосредственно. Реализованы обычные функции работы с деревом, обеспечивающие движение по нему (корень, отец, левый сын, правый сосед и т. п.). Большинство из них возвращает результат типа node – указатель на узел.

Функция st_nodef возвращает узел, помеченный функцией с данным номером, а функция st_nodec возвращает узел, помеченный координатой с данным номером, находящийся на пути от корня к листу, помеченному функцией с данным номером.

Следующие две функции возвращают информацию о части записи экспериментальных данных, соответствующей поддереву, определенному этим узлом. Функция st_offset возвращает целое число – номер байта, с которого начинается соответствующая часть данных. Функция st_step возвращает количество байтов, добавляемое к дереву порождающей процедурой при изменении соответствующей координаты на 1. Эти две функции необходимы при построении выборки значений функции из файла экспериментальных данных. Функции st_fnum и st_cnum возвращают номера функций и координат с заданными именами (и 0, если таковых нет).

Итак, основное преимущество ST-дескриптора – его гибкость, позволяющая снабдить таким дескриптором уже существующие файлы экспериментальных данных. С этой целью нами разработан редактор ST-дескриптора, позволяющий ввести все поля в необходимом формате, проверить корректность полученного дескриптора и записать его в файл.

Автоматизированное место экспериментатора на базе персонального компьютера *IBM PC*

А. Л. АЛЕКСАНДРОВ, А. А. ЖУКОВ, С. А. ПЛАТОНОВ, М. Л. ХИТРОВ

Перед пользователями часто возникает задача создания измерительных и управляющих систем на основе ПК, совместимых с ПК IBM PC. Несмотря на то, что существует большое число устройств, позволяющих осуществлять ввод и вывод аналоговой и цифровой информации в ПК, разработчик, как правило, испытывает трудности с приобретением подобных изделий. Стоимость этих изделий, даже при относительно невысоких их параметрах, сравнима со стоимостью ПК.

Предлагаемое устройство позволяет превратить ПК IBM PC/XT в контрольно-измерительную станцию, предназначенную для сбора данных, управления физическими экспериментами или технологическими процессами. Схема выполнена на базе отечественных комплектующих изделий. Конструктивное исполнение изделия соответствует принятому для плат расширения ПК IBM PC/XT, устройство размещено на одной плате, располагающейся внутри корпуса машины. Питание платы осуществляется от источников питания ПК.

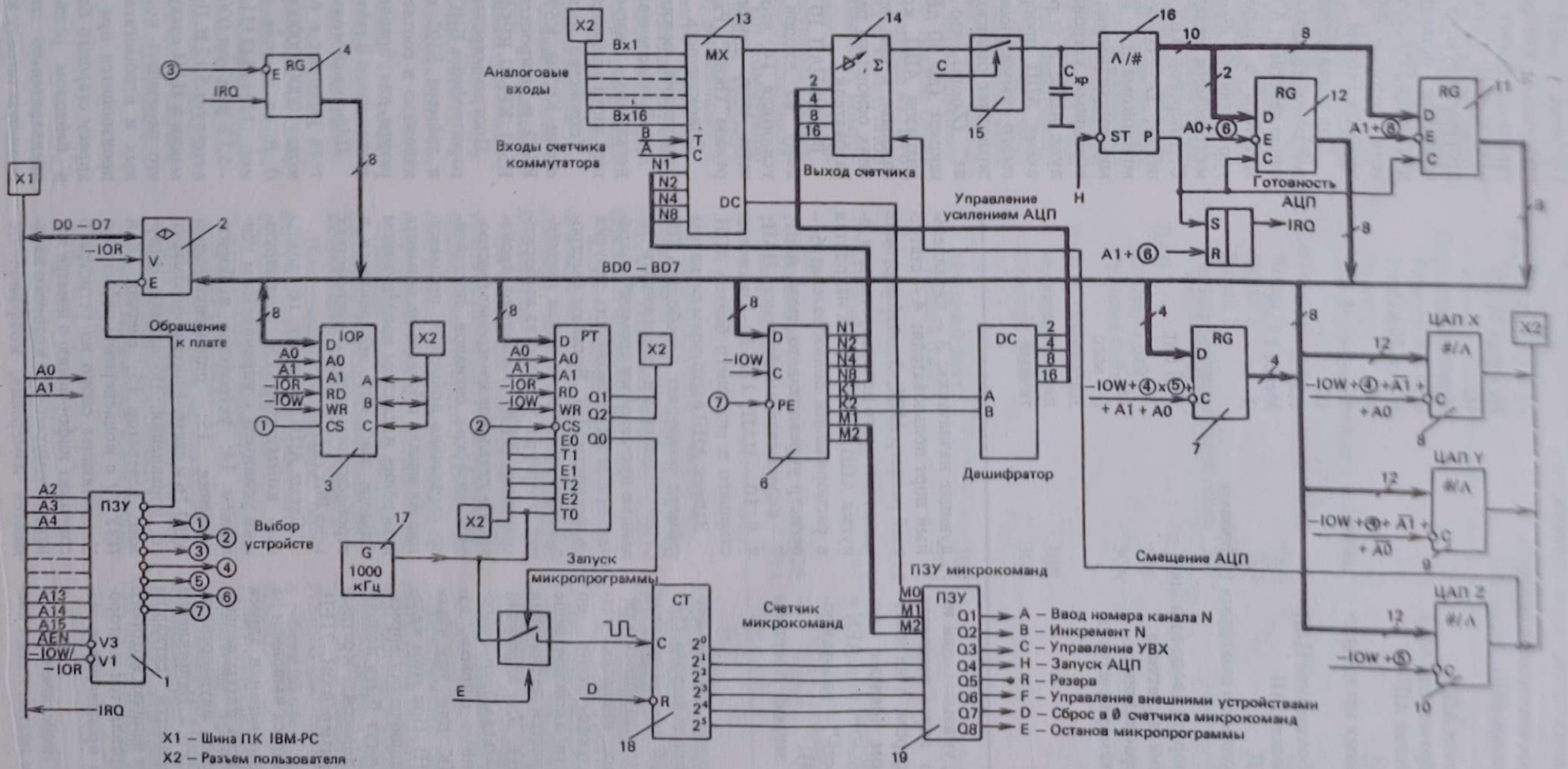
Функциональная схема платы приведена на рисунке. Обмен информацией между устройствами, расположенными на плате, и процессором производится через внутреннюю шину платы, которая связана с шиной ПК через двунаправленный буферный формирователь. Такой способ связи процессора и устройства выбран как более быстрый, чем связь через стандартные интерфейсы ввода-вывода (RS-232, Centronics). Кроме того, последние часто бывают заняты различными штатными устройствами ПК. Для обмена используются следующие сигналы шины ПК (подробное описание шины приведено в руководстве "IBM PC Hardware Reference Library", May 1984):

D0–D7 – 8-разрядная шина данных;

A0–A15 – часть 20-разрядной шины адреса;

–IOW – строб записи информации, следующей из процессора во внешнее устройство (при –IOW = = "низкий" – запись информации);

–IOR – строб чтения информации из внешнего устройства в процессор (при –IOR = "низкий" – чтение информации);


X1
X2
RG
IRQ
D0 – D7
–IOR
V
E
Обращение к плате
A0
A1
IOP
D
A0
A1
RD
WR
CS
A
B
C
–IOW
ПЗУ
A2
A3
A4
A13
A14
A15
AEN
–IOW/
–IOR
V3
V1
Выбор устройства
IRQ
G
1000 кГц
17
PT
Q1
Q2
Q0
E0
T1
E1
T2
E2
T0
Запуск микропрограммы
E
D
R
CT
C
18
Счетчик микрокоманд
ПЗУ микрокоманд
M0
M1
M2
19
Q1
Q2
Q3
Q4
Q5
Q6
Q7
Q8
A – Ввод номера канала N
B – Инкремент N
C – Управление УВХ
H – Запуск АЦП
R – Резерв
F – Управление внешними устройствами
D – Сброс в 0 счетчика микрокоманд
E – Останов микропрограммы
Аналоговые входы
Вх1
Вх16
Входы счетчика коммутатора
B
A
T
C
N1
N2
N4
N8
MX
DC
13
14
Выход счетчика
Σ
C
15
Управление усилением АЦП
H
Cхр
Л/#
ST
P
16
10
8
2
RG
12
11
A0+⑥
A1+⑥
Готовность АЦП
S
R
IRQ
A1+⑥
BD0 – BD7
N1
N2
N4
N8
K1
K2
M1
M2
DC
Дешифратор
–IOW+④x⑤+
+A1+A0
Смещение АЦП
ЦАП X
ЦАП Y
ЦАП Z
Л/#
–IOW+④+A1+
+A0
–IOW+⑤+A1+
+A0
–IOW+⑥
X1 – Шина ПК IBM-PC
X2 – Разъем пользователя

ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ УСТРОЙСТВА

Напряжение входных аналоговых сигналов	+5 – 0 – –5 В
Динамический диапазон АЦП	Не менее 80 дБ
Разрядность АЦП	10
Время преобразования АЦП	Менее 30 мкс
Погрешность преобразования АЦП	0,25 %
Входное сопротивление АЦП	Более 30 кОм
Число каналов АЦП	16
Напряжение выходных аналоговых сигналов	+5 – 0 – –5 В
Разрядность ЦАП	12
Погрешность преобразования ЦАП	0,25 %
Время преобразования ЦАП	Менее 15 мкс
Число каналов ЦАП	3
Совместимость входных и выходных цифровых сигналов	ТТЛ
Число цифровых портов ввода-вывода	3X8 бит
Число каналов таймера-счетчика	3
Разрядность таймера-счетчика	16
Быстродействие таймера счетчика	0,5 мкс
Тип счета	Двоичный, двоично-десятичный

AEN – сигнал, указывающий на то, что адрес выдается процессором, а не контроллером прямого доступа в память (при AEN = = "низкий" – работа процессора);

IRQ – сигнал запроса прерывания.

Обращение к плате производится как к порту ввода-вывода (I/O port). Спецификой адресации этих портов является то, что адреса в некоторых штатных внешних устройствах декодируются только до адреса 3FFH, хотя адресное поле устройств простирается до значения FFFFH (64 Кбайт). Для адресации к устройствам платы выбраны адреса, рекомендованные фирмами-изготовителями ПК для макетных плат (300H–31FH). При этом желательна полная дешифрация адреса, так как тогда могут быть использованы поля адресов, кратные 300H–31FH, например 700H–71FH, B10H–B1FH и т. д.

Дешифратор адреса выполнен на ПЗУ 1. При распознавании адреса одного из устройств платы дешифратор активизирует двунаправленный шинный буфер 2. Направление передачи определяется сигналом –IOR, только при наличии которого информация направляется в сторону процессора и производится чтение.

Устройства, подключенные к внутренней шине платы, имеют следующие назначения: 3 – параллельный порт пользователя; 4 – статусный регистр АЦП; 5 – таймер, один канал которого используется для запуска АЦП, а два других находятся в распоряжении пользователя; 6 – регистр управляющего слова АЦП; 7 – вспомогательный регистр ЦАП; 8, 9, 10 – ЦАП; 11 и 12 – регистры старшего и младшего байтов АЦП.

Запуск АЦП выходным сигналом таймера позволяет освободить ПК от необходимости отслеживать заданные промежутки времени, сохраняя при этом возможность запуска по командам ПК. Большое быстродействие таймера и малая временная апертура устройства выборки-хранения (УВХ) позволяют производить детальное измерение быстропериодических сигналов, постепенно наращивая временной промежуток от начала процесса до момента измерения в каждом последующем периоде так, как это делается в стробоскопических устройствах типа Box car.

Схема АЦП содержит: 16-канальный коммутатор 13; суммирующий усилитель с управляемым усилением 14; устройство выборки-хранения 15; однокристальный АЦП 16 и схему микропрограммного управления. Последняя содержит: генератор 17; счетчик 18 и ПЗУ 19 с конвейерным регистром. Управляющее слово из регистра 6 содержит информацию о номере канала коммутатора, с которого начинаются измерения, коэффициенте усиления усилителя и типе микропрограммы, которая выбирается установкой соответствующего начального адреса. Три микропрограммы программируются изготовителем, а четвертая оставляется резервной для специфических нужд пользователя, которые могут появиться в процессе работы.

Под управлением разрядов слова микрокоманды выполняются следующие действия: A – установка коммутатора в положение, заданное управляющим словом; B – инкремент номера канала коммутатора; C – перевод УВХ в состояние хранения; D – сброс в нуль счетчика микрокоманд; E – окончание микропрограммы; F – управление внешними устройствами; H – запуск АЦП; R – резерв. Наличие в схеме АЦП программно управляемого суммирующего усилителя 14 позволяет реализовать режим "Лупа" (Zoom), что значительно расширяет (до 80 дБ) динамический диапазон АЦП, который является важным, а при некоторых измерениях основным параметром преобразователя.

Выход ЦАП 10 подключен к одному из входов суммирующего усилителя 14, что позволяет смещать нуль усилителя, организуя режим "Окно" (Panorama).

Совокупность программного управления режимами "Окно" и "Лупа" обеспечивает возможность детального просмотра интересующей части входного сигнала.

Устройство выполнено на БИС серий КР580, К591, К556, К1113, К572 и микросхемах серий К555, К561, КР140, КР544 и КР1100.

Программирование параллельного интерфейса БИС типа КР580ВВ55 и таймера БИС типа КР580ВИ53 изложено в соответствующей литературе, для управления АЦП используется канал 0 таймера.

Цифроаналоговые преобразователи работают в смещенном коде: код 100000000000 соответствует 0 В на выходе преобразователя, код 111111111111 соответствует –5,12 В, а код 000000000000 соответствует 5,12 В. При записи информации в ЦАП следует иметь в виду, что перезапись кода из шины данных и вспомогательного регистра производится при записи кода по адресу старшего байта. При работе в байтовом режиме необходимо предварительно записать нули в младшую тетраду кода ЦАП. Такой

режим дает возможность работать быстрее, но с пониженной точностью.

Управляющее слово АЦП имеет следующий формат: старшие четыре бита представляют номер канала коммутатора; следующие два бита устанавливают коэффициент усиления управляемого усилителя, так что коду 00 соответствует усиление 1, коду 01 – 2, коду 10 – 4, коду 11 – 8. Младшие два бита управляющего слова определяют режим работы АЦП. В режиме 0 (код 00) устанавливается канал, номер которого записан в управляющем слове, после чего АЦП ожидает запуска; в этом режиме для переключения канала необходимо перезаписать его номер в регистр управляющего слова. В режиме 1 (код 01) коммутатор включает тот канал, номер которого записан в регистр управляющего слова, а после измерения и считывания данных номер канала инкрементируется, АЦП запускается, и так до тех пор, пока не будут измерены и считаны данные из 16-го канала. После этого коммутатор возвращается к исходному номеру канала (записанному в управляющем слове). Следующий запуск АЦП приводит к повторению цикла. В режиме 2 (код 10) коммутатор включает канал, номер которого соответствует коду, записанному в регистр управляющего слова, производится запуск АЦП, данные считываются, номер канала инкрементируется, система ждет следующей команды запуска, и так до тех пор, пока не будут считаны данные измерения из 16-го канала. Режим 3 зарезервирован для определения пользователем и может применяться, например, при быстром вводе данных с прямым доступом в память в случае применения АЦП типа К1108ПВ1.

При необходимости микропрограммы, выполняющие все операций, требующиеся для процесса измерения и обмена, могут быть изменены заменой микросхемы ПЗУ, которая установлена на панельку.

Аналого-цифровой преобразователь имеет три режима запуска:

1. Режим деления частоты, обеспечивающий периодические запуски АЦП с программно установленными интервалами.

Адреса устройств платы

Устройство	Адрес (шестнадцатеричный)	Операция
Параллельный порт:		
Регистр А	300	Запись-чтение
Регистр В	301	”
Регистр С	302	”
Регистр управляющего слова	303	Запись
Счетчик-таймер:		
Канал 0	304	Запись-чтение
Канал 1	305	”
Канал 2	306	”
Регистр управляющего слова	307	Запись
ЦАП X:		
Младший байт	308	”
Старший байт	30A	”
ЦАП Y:		
Младший байт	308	”
Старший байт	30B	”
ЦАП Z:		
Младший байт	308	”
Старший байт	30C	”
Регистр режима АЦП	310	”
Статусный регистр АЦП	314	Чтение
АЦП:		
Младший байт	318	”
Старший байт	31A	”

2. Режим ждущего запуска, обеспечивающий запуск АЦП через программно заданное время после поступления внешнего сигнала.

3. Запуск по команде ПК, обеспечивающий запуск АЦП через программно заданное время после команды процессора.

Об окончании измерения можно узнать опросом статусного регистра, седьмой бит которого устанавливается в 1, или по запросу на прерывание, формируемому устройством.

Порядок чтения информации из регистров АЦП произвольный. Возможен также байтовый режим.

Программное обеспечение предлагаемого устройства предусматривает следующие функции:

тестирование аппаратной части во всех режимах,

сбор данных с использованием АЦП во всех режимах (выбор соответствующего режима обеспечивается записью управляющего слова в регистр режима по адресу 30FH),

прием и передачу данных с использованием параллельного порта во всех режимах его работы,

выдачу программируемых временных итервалов и деление частоты с использованием таймера,

вывод аналоговых сигналов с использованием ЦАП.

Программное оебспечение может быть реализовано в виде диалоговой системы и пакета подпрограмм. Подпрограммы ориентированы на язык Си и реализуют непрерывные режимы работы (основная задача) и работу по прерываниям (фоновая задача).

Авторам представляется важным обеспечение возможности работы предлагаемого устройства в широко используемой в задачах сбора и обработки экспериментальных данных программной системе ASYST.

Диалоговая программа расчета открытых оптических резонаторов бегущей волны

А.В. БУРОВ, Е.Ф.ИЩЕНКО

В устройствах и приборах квантовой электроники широко применяются открытые оптические резонаторы бегущей волны, предназначенные для возбуждения встречных квазинезависимых волн. При проектировании таких резонаторов необходимо соотносить характеристики возбуждаемых бегущих волн с параметрами резонатора: числом, взаимным расположением и оптической силой элементов, образующих резонатор, а также характером неоднородности заполняющих резонатор оптических сред.

В зависимости от конкретного назначения возможны различные критерии качества резонатора, но прежде всего необходимо решить вопрос об устойчивости резонатора. В случае положительного ответа представляют интерес пространственные характеристики собственных гауссовских пучков резонатора. Для ряда приложений существенна степень частотного вырождения собственных мод резонатора. Кроме того, резонатор должен быть устойчивым к различного рода ошибкам реализации.

На основе известной методики расчета открытых оптических резонаторов нами разработана диалоговая программа, предназначенная для исследования трехзеркальных резонаторов бегущей волны. Входными данными являются параметры конфигурации резонатора: радиусы кривизны зеркал, длины плеч, углы падения осевого луча на зеркала. Вводятся также параметры оптически неоднородной среды, помещенной в первом плече резонатора: коэффициент оптической неоднородности, показатель преломления на оси, длина, поперечный размер, расстояние до первого зеркала.

Выполнение программы начинается с формирования на экране ПК двумерного изображения резонатора, изложения правил работы с программой и объявления требуемых исходных данных. После этого осуществляются их ввод и контроль, а в случае ошибки ввода или неправильного задания исходных данных появляется сообщение о необходимости повторить ввод. Затем вычисляется инвариант лучевой матрицы полного обхода резонатора в сагиттальной и меридиональной плоскостях и производится контроль устойчивости резонатора. После этого вычисляются конфокальные параметры в этих плоскостях, место расположения перетяжки гауссовского пучка резонатора в первом плече, размер пятна основной моды в выбранных сечении и плече резонатора, радиус кривизны волнового фронта и степень частотного вырождения собственных мод резонатора.

Программа позволяет исследовать влияние различных разъюстировок зеркал резонатора на деформацию осевого контура. Расчет эпюр смещения осевого луча при разъюстировке выполняется методом осевого контура, причем в программе предусмотрен вывод эпюр как на экран ПК, так и на печать.

При переходе к вычислению каждого следующего параметра резонатора на экране ПК появляется вопрос о необходимости его расчета, и в случае утвердительного ответа вычисление проводится, а в случае отрицательного — осуществляется переход к вычислению следующего. Имеется защита от неправильных ответов на вопросы.

Различные модификации программы занимают объем памяти от 10 до 18 Кбайт. Язык программирования — Бейсик или Паскаль. Блочная структура программы позволяет при незначительной модификации использовать ее для расчета четырехзеркальных резонаторов бегущей волны.

Экспертные системы и искусственный интеллект

Экспертная психодиагностическая система

А. Р. КУЛМАГАМБЕТОВ, Л. Т. ЯМПОЛЬСКИЙ

В настоящее время при организации массовых психологических исследований [1] все чаще используются ЭВМ. Еще совсем недавно это наблюдалось только на этапе обработки собранных данных. Сегодня картина существенно изменилась. ЭВМ вторглась не только в сферу обработки, но также в процессы обследования и интерпретации результатов тестирования. Это стало возможным благодаря разработке специальных компьютерных систем для психодиагностики [2 – 4]. Описываемая в статье компьютерная психодиагностическая система (ПДС) отличается тремя особенностями:

1. Применение микроЭВМ делает ее удобной для массового использования и тиражирования.

2. В ней выделяются две подсистемы: инструментальная и тестирующая, каждая из которых выполняет свои специфические функции: первая является банком тестов и предназначена для автоматизации процессов загрузки, формирования и накопления тестов, а вторая является инструментом тестирования и предназначена для автоматизации процессов психодиагностического обследования, обработки и интерпретации результатов тестирования.

3. Наличие средств описания и логической обработки знаний психолога-эксперта позволяет подойти к автоматизации процессов рассуждения психолога при анализе и интерпретации результатов психологического обследования. Это наиболее принципиальная особенность данной системы, сильно расширяющая возможности ее практического использования.

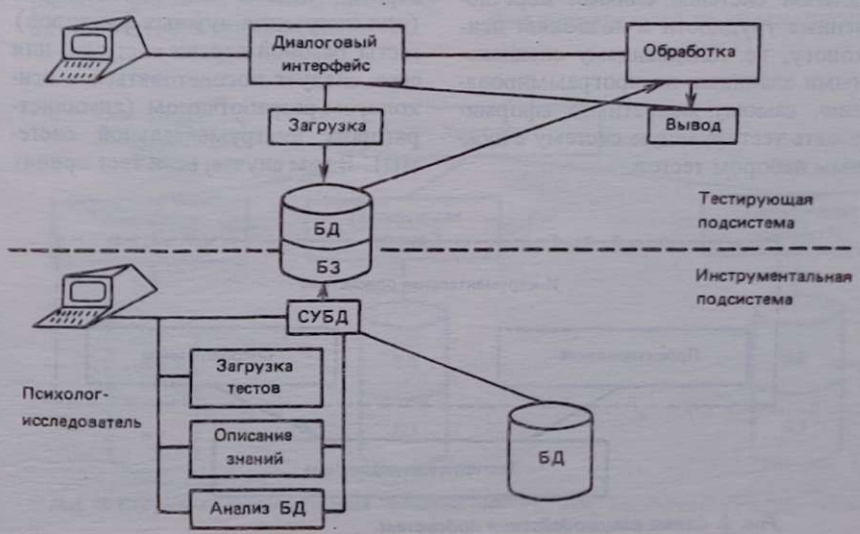
Чаще всего результаты тестирования нужны не сами по себе, а для прогноза поведения и успешной деятельности. С этой целью разрабатываются специальные количественные модели прогноза. Построение таких моделей – трудоемкое и длительное дело. Обычно до создания пригодной для практического использования модели прогноза, предназначенной для решения конкретной задачи, проходит несколько лет. Поэтому завершенные разработки, как правило, отстают от запросов практики.

Наличие в ПДС средств описания и поддержки знаний позволяет использовать теоретические знания и опыт психологов, решающих конкретные прикладные задачи, для выдачи индивидуальных прогнозов и рекомендаций, не дожидаясь завершения работ над построением количественных моделей. Если знания психологов в данной прикладной области достаточно богаты, система позволяет давать рекомендации, не уступающие по своей значимости тем, которые получаются на основе количественных моделей.

Функциональная структура ПДС

Описываемая система компьютерной психодиагностики является системой многоцелевого назначения и включает в себя две подсистемы: инструментальную и тестирующую (рис. 1). Инструментальная подсистема предназначена для проектирования и загрузки тестирующей подсистемы, а тестирующая – для автоматизации психодиагностических обследований. Каждая из подсистем предполагает своего пользователя: инструментальная – психолога-администратора системы, тестирующая – психолога-экспериментатора. Для каждого из пользователей в системе организован свой интерфейс с собственной программной поддержкой.

Рис. 1. Структура ПДС: БД – база данных, БЗ – база знаний



ИНСТРУМЕНТАЛЬНАЯ ПОДСИСТЕМА

Инструментальная подсистема ориентирована на построение специализированных психодиагностических систем для решения конкретных прикладных задач. Необходимость разработки инструментальной подсистемы связана с тем, что различные прикладные задачи требуют применения различных психодиагностических тестов. Поэтому для каждого нового исследования следует либо создавать новое математическое обеспечение, либо иметь специальную инструментальную систему, которая содержит банк тестов и снабжена дополнительными средствами проектирования тестирующих систем. Использование первого пути ограничено по следующим причинам:

а) разработка для каждого нового исследования специального математического обеспечения требует содержания штата программистов, а также времени для написания и отладки программ;

б) понятийный аппарат программистов значительно отличается от понятийного аппарата психологов, вследствие чего каждый раз прилагается много дополнительных усилий на выработку языка, понятного обеим группам специалистов (программистам и психологам);

в) процесс разработки новых тестов неотделим от использования уже существующих и непременно должен содержать средства накопления, хранения, просмотра и обработки результатов тестирования, что весьма сложно обеспечить для разрозненного множества тестов.

Создание специальной инструментальной системы снимает перечисленные трудности и позволяет психологу, не обладающему специальными знаниями по программированию, самому оперативно сформировать тестирующую систему с нужным набором тестов.

Обобщенная схема взаимодействия подсистем изображена на рис. 2.

Необходимость разделения инструментальной подсистемы на две части вызвана их различными функциональными возможностями и назначением.

Блок проектирования подсистемы содержит большой набор различных тестов и методик, заранее загруженных в инструментальную подсистему. Все множество факторов, измеряемых с помощью данных тестов, является тем системным набором факторов, из которых психолог-заказчик совместно с психологом-экспертом выбирает факторы, существенные для решения предлагаемой психологом-заказчиком прикладной задачи. К основным функциям блока проектирования относятся:

1) формирование теста, заключающееся в его описании, загрузке и модификации, а также описании условий и ограничений на применение загружаемого теста;

2) проектирование тестирующей подсистемы, включающее: формулировку прикладной задачи; выбор из системного набора необходимых факторов и формирование "батарей" тестов; подготовку и описание методики проведения тестирования; задание демографических показателей; описание знаний психолога, сформулированных в виде утверждений (продукций) [5 – 7] по факторным оценкам и их сочетаниям; описание модели отбора испытуемых.

Формирование теста. Прежде чем готовить тест к загрузке в инструментальную подсистему, нужно проверить, нельзя ли использовать (для получения нужных факторов) тесты базовой версии системы, для чего следует посоветоваться с психологом-разработчиком (администратором инструментальной системы). В том случае, если тест принят к загрузке, психолог должен подготовить тест в виде пронумерованного набора заданий W и списка измеряемых характеристик отношений: R_1 – "фактор", R_2 – "коэффициент", R_3 – "класс", R_4 – "текст".

Здесь отношение R_1 есть

$$R_1 (N_W, X, G, Z, T_W),$$

где N_W – множество номеров заданий (например, номеров вопросов), X – множество факторов, G – множество весовых коэффициентов, Z – варианты ответов, T_W – время подготовки ответа. Пример отношения R_1:

N_W	X	G	Z	T_W
14	1	0,045	1	0
7	1	–0,03	1	0
...	...	...	...	...
29	7	0,052	1	20

Отношение R_2 есть

$$R_2 (K, X),$$

где K – множество центрирующих констант, X – множество факторов.

Отношение R_3 есть

$$R_3 (N_1, N_2, \ldots, N_9, X),$$

где $N_1, N_2, \ldots, N_9$ – границы классов, так как измерение факторов может проводиться по 1, 2, ..., 10-балльной шкале.

Отношение R_4 есть

$$R_4 (U, T, X, O),$$

где U – указатель типа текста (текст для психолога или испытуемого), T – множество тестовых заготовок, O – оценка фактора, X – множество факторов.

Кроме того, система позволяет собирать об испытуемых демографические, медицинские или другие характеристики. Список этих характеристик формируется в отношении R_5, например

$$R_5 (A_1, A_2, \ldots, A_6),$$

где A_1 – номер испытуемого, A_2 – фамилия, A_3 – год рождения, A_4 – пол, A_5 – род занятий, A_6 – образование.

Все отношения R_1, R_2, R_3, R_4, R_5 описываются на языке описания данных СУБД и вводятся в базу данных средствами специально раз-

Рис. 2. Схема взаимодействия подсистем

работанной СУБД реляционного типа, использование которой позволяет:

сократить объем программного обеспечения, необходимого для проектирования ПДС;

осуществлять изменение и дополнение всех вводимых в БД отношений (таблиц);

включать в программное обеспечение инструментальной подсистемы дополнительные программы обработки и анализа БД.

Рассмотрим основные этапы процедуры загрузки тестов.

1. Ввод заданий W. Все задания вводятся независимо и в произвольном порядке. Длина текста одного задания не должна превышать 12 строк на экране дисплея с длиной строки 80 символов.

2. Ввод отношений R_1, R_2, R_3, R_4, R_5. При необходимости в отношениях могут быть заменены и дополнены новые столбцы. Порядок строк в отношениях несуществен.

3. Ввод текстовых заготовок T, соответствующих различным значениям факторов X в БЗ.

Формирование теста заканчивается подготовкой инструкций по работе с тестом для испытуемого и психолога. В инструкции для психолога указываются особенности создания теста, условия его использования и ограничения применения.

Ввод заданий, текстовых заготовок и инструкций осуществляется с помощью текстового редактора системы.

Проектирование тестирующей подсистемы. Этот этап начинается с постановки прикладной задачи, т. е. психолог, решающий данную задачу, обращается к инструментальной подсистеме с целью создать систему, автоматизирующую процесс решения поставленной задачи. Психолог должен также понимать, какие психологические факторы существенны для решения данной задачи. В случае, если он затрудняется это сделать, он должен полностью положиться на знания психолога-администратора инструментальной подсистемы. В процессе проектирования система предлагает психологу базовый список факторов, которые система может измерять. Психолог отмечает на экране дисплея существенные факторы и называет демографические показатели, необходимые для регистрации в БД. Далее психолог выделяет (и именует) психологические типы испытуемых, требуемые для решения поставленной прикладной задачи, после чего описывает эти состояния в виде утверждений (продукций) на множестве значений факторов X и множестве демографических, медицинских и других показателей личности $A_1, A_2, \ldots, A_k$. Описание и ввод в БЗ подготовленных утверждений ведутся на специально разработанном языке описания продукций, а для описания методики тестирования предлагается также язык описания методик. Этот язык позволяет в процессе тестирования личности в зависимости от ответов и вычисляемых характеристик испытуемого управлять предъявлением испытуемому заданий, что, в свою очередь, ведет к существенному сокращению времени тестирования.

Блок сопровождения включает в себя программы обработки накопленных в БД результатов тестирования и совершенствования БЗ тестирующей системы. Перечислим основные функции данного блока.

1. Просмотр накопленных тестирующей подсистемой данных. При этом обеспечиваются визуальный просмотр и распечатка ответов и социально-демографических данных для любой заданной группы испытуемых.

2. Повторная обработка и распечатка результатов тестирования отдельных испытуемых и групп испытуемых.

3. Выборка и перезапись в рабочую область любой части БД для проведения статистической обработки результатов тестирования.

4. Проверка утверждений БЗ на полноту, т. е. все заданные текстовые заготовки (константы) и измеряемые факторы должны использоваться в утверждениях БЗ.

5. Проверка утверждений БЗ на непротиворечивость:

по одним и тем же оценкам факторов не должны выдаваться различные текстовые заключения,

в любом текстовом заключении не должно быть взаимопротиворечивых предложений.

6. Выборка, просмотр и корректировка утверждений БЗ по различным условиям.

Вместе с тем блок сопровождения позволяет проводить сравнительный анализ результатов прогноза (например, профессиональной деятельности) по архивным данным, записанным в БД, и результатов последующей деятельности испытуемых, что позволяет уточнять модель классификации (например, профотбора), представленной в БЗ тестирующей подсистемы.

ТЕСТИРУЮЩАЯ ПОДСИСТЕМА

Предназначена для автоматизации психологического тестирования испытуемых в соответствии со сформированной программой эксперимента. Предполагается, что тестирующая подсистема ориентируется на решение конкретной прикладной задачи и включает "батареи" тестов, спроектированных на инструментальной подсистеме (рис. 3). Сформированные "батареи" тестов позволяют измерить факторы, выделенные психологом и необходимые для решения поставленной задачи. Структура тестирующей подсистемы представлена на рис. 4. Рассмотрим назначение каждого из блоков подсистемы.

Диалоговый интерфейс. Содержит три основные программы:

1. Предъявление инструкций. Для испытуемого на экране дисплея

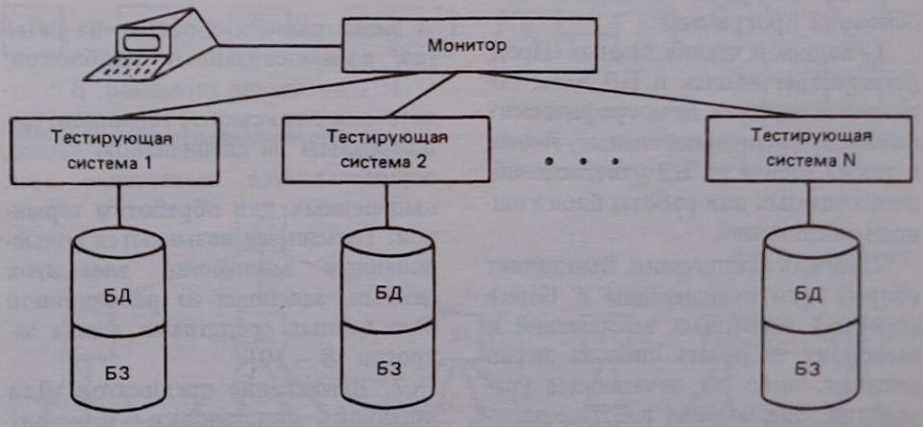


Рис. 3. Структура формирования "батарей" тестов ПДС

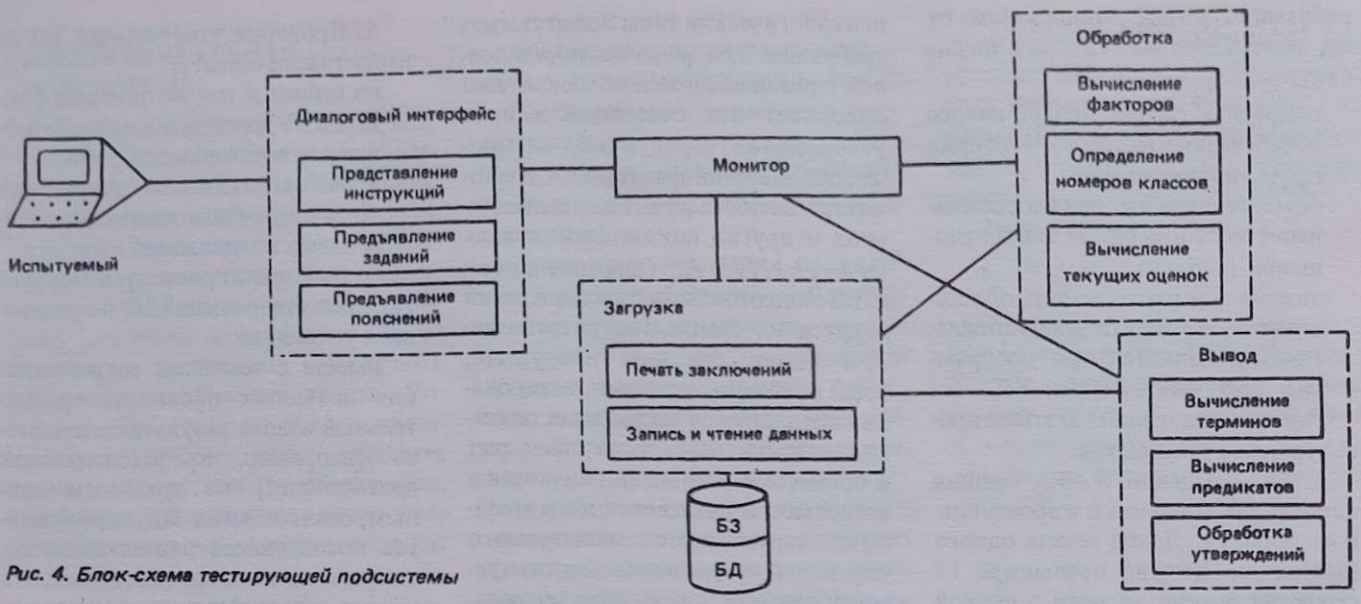


Рис. 4. Блок-схема тестирующей подсистемы

печатаются указания, которыми необходимо руководствоваться при выполнении заданий очередного теста.

2. Предъявление заданий. В зависимости от типа предъявляемого теста задания могут предлагаться в структурированном текстовом или графическом виде. Структурированный текст требует определенного расположения слов задания, предъявляемого на экране, с таким расчетом, чтобы оно облегчало восприятие и понимание выполняемого задания и в большей степени соответствовало поставленной цели. Объем предъявляемого задания ограничен 12X80 символами (12 строк экрана по 80 символов в каждой строке). Кроме того, в зависимости от типа текста программа поддерживает различные варианты ввода ответа испытуемого.

3. Предъявление пояснений. Эта информация используется в качестве постоянной подсказки испытуемому о вариантах ответа и возможных действиях.

Блок загрузки. Содержит две основные программы:

1. Запись и чтение данных. Предусматривает запись в БД даты обследования, всех демографических показателей, ответов испытуемого, а также чтение из БЗ утверждений, необходимых для работы блока вывода заключений.

2. Печать заключения. Выполняет сборку (по выведенным в блоке адресам) текстовых заключений и вывод их на печать либо на экран дисплея, либо на печатающее устройство. Заключения в БД не записываются, поскольку для любого испытуемого по хранящимся в БД ответам можно заново получить необходимое заключение.

Блок обработки. Содержит две программы:

1. Вычисление значений факторов. При загрузке теста порядок вычисления соответствующих тесту факторов анализируется и "погружается" в разработанную и описанную в инструментальной подсистеме схему вычисления факторов, а необходимые вычислительные константы и тексты заносятся в отношения БД R_1, R_2 и R_3.

2. Вычисление текущих оценок. В системе описана вычислительная сеть, позволяющая для любого теста определить достаточно большое количество вычисляемых легко интерпретируемых показателей – таких, как: среднее время ответа на задание, вероятность ошибочного ответа, среднее время ответа на задания в первой и второй половинах теста, процент правильных ответов, время обдумывания которых превышает заданный порог, и т. д.

Блок вывода. Состоит из четырех взаимосвязанных подблоков:

1. Вычисление терминов. В соответствии с описанием терминов, выполненным на специальном языке, осуществляется вычисление всех выделенных для обработки терминов. Терминами называются поименованные множества элементов данных, заданных на реляционной базе данных средствами языка запросов [8 – 10].

2. Вычисление предикатов. Для заданной подстановки констант вместо переменных (интерпретация переменных) осуществляется вычисление предъявляемого множества базовых предикатов P_1, P_2 ... ..., P_n, входящих в утверждения (продукции) [5, 6]. Все истинные предикаты отмечаются и передаются как результат работы программы.

3. Обработка утверждений. В данной программе обрабатывается все предъявляемое множество утверждений (модель) в соответствии с заданными правилами вывода. В результате печатается полученное на основе утверждений текстовое заключение.

4. Управление выводом. Программа осуществляет общее управление всеми предыдущими программами по всей БЗ.

База данных представляет собой стандартную БД реляционного типа, в которой хранятся все описанные в инструментальной подсистеме отношения.

База знаний. Представляет собой множество утверждений, сформулированных над множеством атрибутов и терминов, определенных в БД.

Тестирующая подсистема (см. рис. 4) представляет собой независимый самостоятельный программный продукт. Первоначально пустая, после загрузки она ориентируется на решение конкретной прикладной задачи. Если эта задача типовая, то данная тестирующая подсистема может тиражироваться. Сведения об использовании системы должны сообщаться разработчику (администратору инструментальной подсистемы) для централизованной обработки БД, накопленных тестирующими подсистемами.

Представление знаний и управление ими

Разработанная ПДС представляет собой вычислительную систему, основанную на знаниях, поэтому самыми существенными являются вопросы представления знаний и управления процессом их использования.

В качестве языка описания знаний используется ограниченный язык, ориентированный на формализованную запись утверждений психолога об измеряемых индивидуально-психологических особенностях, о проявлениях этих особенностей в поведении, их влиянии на успешность деятельности и т. п. Утверждения записываются в виде продукций, формулируемых на множестве предикатов P:

$$P_1 \wedge P_2 \wedge \ldots \wedge P_n \rightarrow P_k,$$

где P_k, $P_i \in P$, $1 \leqslant i \leqslant n$, P – множество одно- и двуместных предикатов, заданных на множестве атрибутов A, включающем измеряемые психологические параметры, социально-демографические показатели, производные оценки и т. д. Предикатом будем называть выражение вида

$(A_\alpha \theta a)$ или $(A_\alpha \theta A_\beta)$, где
$A_\alpha, A_\beta \in A$, $a \in A_\alpha$ и
$\theta \in \{<, >, \leqslant, \geqslant, =, \neq\}$,

например

(психическая неуравновешенность $\geqslant 3$) $\wedge$ (интроверсия $\geqslant 8$) $\rightarrow$ (текст=24)

Здесь утверждается, что если у испытуемого оценка по фактору "психологическая неуравновешенность" меньше или равна 3 и интроверсия больше или равна 8, то "Главными отличительными особенностями испытуемого являются спокойствие, уравновешенность и постоянство. Настроение его всегда ровное и устойчивое, вид невозмутимый и спокойный, мимика и жесты однообразны, голос тусклый и невыразительный, речь и движения замедленные. Прежде чем что-либо сделать, он тщательно обдумывает и планирует свои поступки, а приняв какое-либо решение, выполняет его последовательно и методично. Перемен и неожиданностей не любит . . . ".

Все текстовые заготовки (константы) представляют собой предложения на естественном языке, заранее сформулированные психологом-экспертом.

Для организации вывода все утверждения F разбиваются на непересекающиеся подмножества $F = < F_1, F_2, \ldots, F_n >$, где F_1 – множество утверждений, левая часть которых состоит из одного предиката (называемого простым), а в правой части утверждения указывается имя (номер), присваиваемый данному предикату. Например, в утверждении $(A_8 \leqslant 3) \rightarrow P_{30}$ A_8 – фактор интроверсии, P_{30} – наименование "экстраверсия" (в системе используется номер 30).

Множество образовано утверждениями, в левой части которых записываются конъюнкции простых предикатов из P, а в правой части указывается наименование (номер) термина $I_1, I_2, \ldots, I_k$, который считается актуальным (активным) при выполнении всех предикатов (условий) левой части утверждения, например

$$(A_3 > A_1) \wedge (A_3 > A_2) \wedge (A_1 > > A_2) \wedge (A_3 \geqslant 7) \wedge (A_1 \leqslant 5) \rightarrow \rightarrow (I = I_1),$$

где $\wedge$ – символ конъюнкции (логическое И), A_1 – индивидуалистическая ориентация, A_2 – коллективистская ориентация, A_3 – деловая ориентация, I_1 – терминологическая константа "деловой".

Множества $F_3, F_4, \ldots, F_n$ состоят из утверждений, образованных конъюнкциями различных сочетаний предикатов, определенных на множествах из A и множестве терминов I. Подмножества $F_1, F_2, \ldots$ $\ldots, F_n$ могут образовывать различные структуры обработки (рис. 5,6).

Создаваемая структура обработки зависит от особенностей утверждений и правил их обработки, которые делятся на две независимые системы правил: интерпретационные правила, определяющие подготовку текстовых заключений, и разделяющие правила, определяющие классификацию, т. е. отнесение испытуемого к тому или иному классу (профессиональной пригодности, прогностической оценки и т. д.).

В результате тестирования система вычисляет оценки по всем измеряемым факторам, результаты измерений передаются в систему логической обработки утверждений, где по заданному набору правил осуществляется интерпретация факторных оценок и формируется текстовое заключение, в котором дается развернутое описание личности испытуемого.

Реализация тестирующих систем

В настоящее время с помощью инструментальной ПДС разработаны три тестирующие системы различного назначения.

ТЕСТИРУЮЩАЯ СИСТЕМА ПДТ-25

Система загружена мультивариативным психодиагностическим тестом (ПДТ), позволяющим проводить измерения от 1 до 25 факторов личности [11].

Отношение между этими факторами личности показано на рис. 7 в

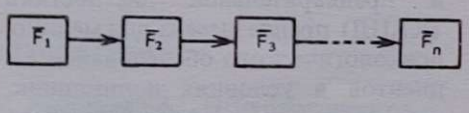


Рис. 5. Линейная структура обработки

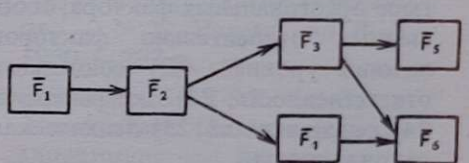


Рис. 6. Параллельная структура обработки

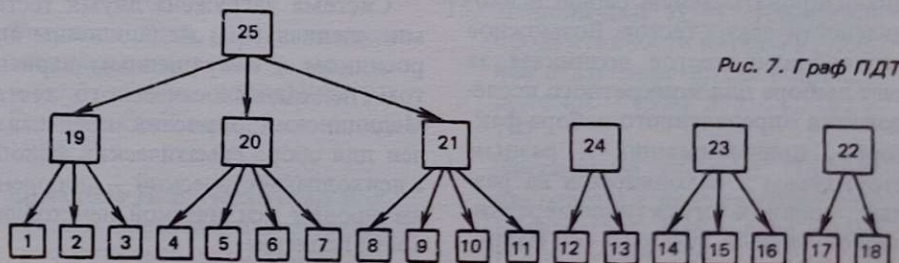


Рис. 7. Граф ПДТ

виде графа. Граф имеет четырехуровневую структуру. На каждом уровне расположено определенное число вершин. Вершины соответствуют личностным признакам, измеряемым с помощью вопросов теста, или факторам, обобщающим эти признаки. Чем выше уровень, тем выше степень обобщенности. Каждая вершина более низкого уровня соединяется только с одной вершиной более высокого уровня. Вертикальные связи между вершинами показаны стрелками.

Первый уровень соответствует 257 базисным заданиям теста. Ответы на них являются исходной информацией об индивидуально-психологических особенностях личности. Все факторы более высокого уровня выступают как обобщения разного уровня относительно этой информации.

Второй уровень содержит 11 факторов, измеряющих напряженность психических процессов: 1) раздражительная слабость, 2) тревожность, 3) ипохондрия, 4) фобии, 5) подозрительность, 6) паранойяльность, 7) шизоидность, 8) интрапсихическая неупорядоченность, 9) гипотимия, 10) конфликты узкого круга, 11) конформность.

Третий уровень содержит 10 факторов. Три из них (19 – 21) обобщают информацию от 11 факторов второго уровня, т. е. тоже характеризуют психическую напряженность, а семь других измеряют другие особенности личности: 12) совестливость, 13) расторможенность, 14) общая активность, 15) робость, 16) общительность, 17) эстетическая впечатлительность, 18) женственность, 19) невротизм, 20) психотизм, 21) депрессия.

Четвертый уровень содержит четыре ортогональных фактора, обобщенных относительно факторов нижних уровней: 22) социальная ответственность, 23) экстраверсия, 24) сенситивность, 25) психическая напряженность.

На базе приведенной модели инструментальная система может инициализировать целую серию психодиагностических тестов. Возможное разнообразие тестов возникает за счет выбора для конкретного исследования определенного набора факторов, принадлежащих к разным структурам и находящихся на разных уровнях структурно-иерархической модели (см. рис. 6). Например, задачи исследования могут потребовать измерения факторов, принадлежащих одной из четырех структур: социализация, экстраверсия, сенситивность и психическая напряженность (вершины 22–25), – или факторов одного уровня. Для проектирования конкретного варианта теста психологу достаточно перечислить номера выбранных им факторов. Инструментальная система сама отберет из БД и БЗ соответствующие вопросы, отношения и текстовые заготовки. Сформированный на инструментальной системе опросник предъявляется испытуемому в диалоговом режиме. Отвечая на вопросы теста, испытуемый нажимает клавиши "+", "–" и "*" на пульте микроЭВМ. Система анализирует полученные ответы, обрабатывает и выдает заключение. Заключение состоит из двух фрагментов: графического и вербального. На графике изображается профиль характеристик личности испытуемого с указанием оценок по каждому из факторов личности. Вербальное заключение представляет собой вспомогательную словесную интерпретацию профиля характеристик личности. Данная машинная интерпретация является предварительной и предназначается только для специалиста–психолога или психиатра, работающего с испытуемым.

Все ответы испытуемых записываются в БД, которая в дальнейшем используется для совершенствования теста и интерпретационной или прогностической модели.

ТЕСТИРУЮЩАЯ СИСТЕМА КОМПЬЮТЕРНОГО АНАМНЕЗА И ПРЕДВАРИТЕЛЬНОЙ ДИАГНОСТИКИ

Система компьютерного анамнеза и предварительной диагностики (КАПД) предназначена для медико-психологического обследования пациентов в условиях поликлиник, первичного и профилактического осмотров, массовой диспансеризации населения.

Система загружена двумя тестами: специальным медицинским опросником и сокращенным вариантом психодиагностического теста. Медицинский опросник предназначен для сбора соматических жалоб, а психодиагностический – для оценки уровня психической неустойчивости пациента.

Взаимодействие пациента с КАПД происходит следующим образом. Вначале система задает пациенту вопросы из соматического опросника, переводит ответы в симптомы, систематизирует их и печатает заключение. Если оказывается, что соматические жалобы пациента плохо структурируются или излишне обильны, ему предлагается ответить на вопросы психодиагностического теста.

Система КАПД анализирует получаемые ответы и выдает заключение для врача и пациента. В заключении для пациента указывается, к каким из 16 врачей-специалистов (терапевт, кардиолог, психиатр и др.) ему следует обратиться. Порядок, в котором система перечисляет специалистов, соответствует рекомендуемой последовательности их посещения. Заключение для врача печатается на бумаге, где указываются жалобы пациента, объединенные по следующим группам: глаза и уши, кожа, мочеполовая система, нервная система, сердечно-сосудистая система и др.

В зависимости от условий применения в системе можно управлять глубиной опроса и тактикой предъявления вопросов.

ТЕСТИРУЮЩАЯ СИСТЕМА "МОТИВАЦИЯ – СПОСОБНОСТИ – СВОЙСТВА"

Система "мотивация – способности – свойства" (МСС) предназначена для автоматизации профессионального психологического отбора руководителей. Психодиагностическое обследование кандидатов на руководящую работу должно быть разносторонним, затрагивающим мотивационную сферу, способности и свойства личности. Поэтому в систему МСС были загружены четыре теста. Для измерения мотивации в МСС включена "Ориентировочная анкета" чехословацких психологов В. Смекалы и Н. Кучеры [13]. Эта анкета является модификацией известного теста Б. Баса и измеряет относительную степень выраженности трех мотивов: индивидуалистического, коллективистского и делового.

Для оценки общих способностей использованы два теста: на аналогию [12, 14] и логико-комбинаторный [11, 15]. Оба теста хорошо себя зарекомендовали при решении задач профотбора.

Индивидуально-психологические особенности личности исследуются в МСС с помощью четырехфакторного варианта ПДТ [11, 12]. Он позволяет измерить четыре ортогональных личностных фактора: социальную ответственность, экстраверсию, сенситивность и психическую напряженность.

В связи с тем что психологическое обследование с целью отбора всегда сталкивается с установочным поведением, стремлением испытуемого показать себя в лучшем, более выгодном свете, в МСС включена шкала искренности, измеряющая отношение испытуемого к обследованию.

Несмотря на довольно большое количество тестов, длительность обследования в МСС не превышает 30 мин. По результатам обследования печатается заключение, содержащее графическую и вербальную части. Графическая часть представляет собой профиль характеристик личности, а вербальная – его интерпретацию. Вербальное заключение состоит из двух частей: оценки профессиональной пригодности кандидата и развернутого описания особенностей личности.

Собираемый МСС материал организуется в БД и может быть изучен и статистически обработан на инструментальной подсистеме.

Особенности ПДС

Описанная в статье компьютерная система, разработанная на языке Бейсик и предназначенная для автоматизации психологического обследования, использует знания психологов как для получения ответа на конкретный вопрос, возникающий при решении прикладной задачи (выбор руководителя, постановка диагноза, прогноз успешности профессиональной деятельности и т. д.), так и для написания развернутой вербальной интерпретации полученных результатов. Этим описанная система отличается от классических экспертных систем [16]. В них специалист может вызвать и просмотреть цепочку вывода для того, чтобы убедиться в логической правильности решения компьютера, но не может получить развернутого вербального объяснения решения. Однако большинство специалистов считают, что логическая правильность вывода не является гарантией разумности решения. Даже в тех случаях, когда решение экспертной системы совпадало с мнением специалиста, их разочаровывал путь решения: промежуточные выводы представлялись ужасающе глупыми и неестественными. Представление о разумности рассуждений компьютера не возникало. Поэтому появилось новое осмысление проблемы: разумность определяется не только качеством результата, но и способом его получения.

В прикладных задачах (геологических, медицинских, психологических и т. д.) специалисты отвергают решение как непонятное, если ему дается только логическое обоснование. Решение принимается человеком как обоснованное и истинное не тогда, когда оно удовлетворяет всем формальным критериям истинности, т. е. получено в соответствии с правилами логики, а тогда, когда оно правдоподобно, т. е. удовлетворяет общим представлениям специалиста и получено в соответствии с правилами содержательного обоснования, принятыми в данной науке. При этом не всякое истинное решение оказывается правдоподобным, а правдоподобное – истинным. Введение дополнительной вербальной интерпретации позволяет устранить данное противоречие, поэтому ее разумно включить в программное обеспечение экспертных консультативных систем различного назначения. Это приведет к развитию интерпретирующих экспертных систем, в которых контакт специалиста и экспертной системы может быть более полным.

ЛИТЕРАТУРА

1. **Анастази А.** Психологическое тестирование. – М.: Педагогика, 1982, в 2-х т., т. 1 – 320 с., т. 2 – 336 с.
2. **Гаврилов Т. А.** Представление знаний в экспертной диагностической системе АВТАМТЕСТ// Техническая кибернетика. – 1984, № 5. – С. 168 – 175.
3. **Ермакова И. В.** Некоторые подходы и перспективы развития автоматизированной диагностики и прогнозирования за рубежом//Вопросы психологии. – 1986, № 4. – С. 170 – 176.
4. **Butcher J. N.** (Ed.) Computerized Psychological Assessment. – N Y: Baste Books, 1987. – 536 p.
5. **Элти Дж., Кумбс М.** Экспертные системы: концепции и примеры: Пер. с англ. – М.: Финансы и статистика, 1987. –191 с.
6. **Хейес-Рот Ф., Уотерман Д., Ленат Д.** Построение экспертных систем: Пер. с англ. – М.: – Мир, 1987. – 430 с.
7. **Lenat D. R.** Knowledge-based System in Artificial Intelligence. – N Y: McGraw-Hill, 1982.
8. **Кулмагамбетов А. Р.** Расширение средств информационного описания объектов реального мира в реляционной СУБД// Логико-алгебраические модели представления знаний в экономических, технических и организационных системах. – Ашхабад: Ылым, 1983. – С. 30–32.
9. **Дрибас В. П.** Реляционные модели баз данных. – Минск: БГУ, 1982. – 198 с.
10. **Ульман Дж.** Основы систем баз данных: Пер с англ. – М.: Финансы и статистика, 1983. – 337 с.
11. **Мельников В. М., Ямпольский Л. Т.** Введение в экспериментальную психологию личности – М.: Просвещение, 1985. – 319 с.
12. **Мельников В. М., Ямпольский Л. Т.** Психология. – М.: Физкультура и спорт. – 1987. – С. 265–311.
13. **Buss B. M.** Leadership, Psychology and Organizational Behavior. – N Y: Harper and Row, 1960. – 297 p.
14. **Amthauer R.** T-S-T. Inteligenz-Structur-Test. Handanweizug die Durchfurung und Ausmertung. – Gottingen: Verlag fur Psychologie, 1955. – 435 p.
15. **Ямпольский Л. Т.** Измерение продуктивности интеллектуальной деятельности//Вопросы психологии. – 1984, № 5. – С. 142 – 147.
16. **Экспертные** системы. Принципы работы и примеры/Под ред. Р. Форсайта: Пер. с англ. – М.: Радио и связь, 1987. – 222 с.

Собирательная модель интеллекта

П. В. БУХ-ВИНЕР

В последнее время в области искусственного интеллекта (ИИ) наблюдаются многочисленные попытки объединить автоматическое обучение и целеполагание в единую систему. Для чего? Чтобы свести подготовку к работе сложных электронных систем только к воспитанию и образованию на доступном человеку-воспитателю концептуальном уровне, чтобы автомат надежно понимал воспитателя. Дальнейшее развитие ИИ должно состоять в разработке методов воспитания аппаратно готовых, на алгоритмически недисциплинированных систем. Наиболее же "воспитанные" автоматы должны массово воспроизводиться для решения практических задач.

Для построения ИИ представляется необходимым алгоритм сбора информации в реальном времени при одновременной ее классификации с целью продления существования (удовлетворения потребностей) этого алгоритма. Данная статья содержит разработку и описание такого алгоритма на основе ретроспективного анализа направлений создания ИИ.

Существуют два крайних взгляда, два учения о создании ИИ: коннективизм и символизм. Оба эти учения, реализуя вечное философское противоречие "душа–тело", развиваются параллельно, постепенно проникая друг в друга и обогащая наше общее представление об интеллекте [Д. Массаро, 1986].

КОННЕКТИВИЗМ

Коннективизм вырос из разработок в области перцептронов [Ф. Розенблатт, 1958; М. Минский, С. Пейперт, 1968] и первоначально стоял в стороне от ЭВМ. Перцептрон создавался как информационная модель нейронной сети в терминах кибернетики [К. Шеннон, 1948]. Неоконнективисты [Дж. Фельдман, 1985] строят такие модели на сетях микропроцессоров. Коннективизм объясняет интеллект как процесс, возникающий при передаче информации. Методом коннективизма является численное моделирование распространения активности по сети большого числа простейших пороговых элементов со случайными связями (машина Больцмана). Это физический, аппаратный подход к интеллекту. Устройства такого типа работают параллельно и имеют две различимые фазы – запись и чтение паттернов с ожидаемой погрешностью.

В фазе записи модифицируются активности (веса) связей сети, осуществляя хранение записываемого паттерна. Запись другого паттерна приводит к возрастанию (подкреплению) активности в совпадающих с предыдущим паттерном местах. В фазе чтения необходимо предъявить на вход сети паттерн-запрос – наиболее активные связи сети пришлют на выход наиболее близкий паттерн из запомненных [А. Фролов, И. Муравьев, 1987].

Таким образом, без явного использования структуры когнитивных процессов распознавания и обучения автоматически решается задача постановки в соответствие – важная задача психологии восприятия.

На первый взгляд, практическое применение идей коннективизма осложняется только недостатком числа элементов и связей аппаратуры или времени пересчета численной модели в компьютерах. Однако имеется более важный недостаток – случайность, вероятность процессов внутри таких устройств, протекающих (в идеале) непрерывно. Игнорирование структурности восприятия создает проблемы применения ответов таких систем. Чтобы получить гарантию правильного поведения такой машины, необходимо с ней сложно экспериментировать, воспитывая ее и привыкая к ней, самому экспериментатору [Л. Шастри, 1987].

СИМВОЛИЗМ

Символизм [А. Ньюэлл, Г. А. Саймон, 1956] трактует интеллект как целенаправленную обработку информации (манипуляцию символами). Это ментальная трактовка интеллекта. Методом этого подхода традиционно является логическое программирование компьютера (машина Тьюринга). Фундаментальным достижением этого подхода считают уточнение понятия алгоритма как структурно-процессорной сущности, необходимой для работы физической системы [Д. Гелернтер, 1987].

Символьный подход позволил структурировать когнитивные процессы в сетях параллельной обработки информации, определяя цели этих процессов. Разработка символьных систем принятия решения показывает значительное преобладание в качестве цели решения только контроля. Однако человеческое рассуждение имеет более объемные цели. Практически не существует вычислительной модели человеческого рассуждения, и необходимо ее создать [А. Дутта, 1987].

В чем заключается символизм системы обработки символов? – В реификации (цитировании, овеществлении). Интеллект в символистском смысле есть внутреннее выражение, "имеющее в виду" что-либо при использовании этого выражения. Цитирование объекта реального мира, представленного в перцепции образом объекта, состоит в наличии внутри системы ИИ символа (например, слова), связанного с этим образом. Цитаты необходимо последовательно попарно сопоставлять, имея целью бинарно ответить на вопрос: "Есть или нет?". Этот ответ, правильный или ошибочный относительно воспринимающей системы, и устанавливает связь между ментальным символом и образом в перцепции [Д. Перлис, 1987].

Распространение активности по сетям программируемых процессоров – коннективистская концепция в рамках символизма. Одна школа уверена, что распространяться должна символьная информация (распространение меток), другая – что используются числовые веса и активация распространяется более аналоговым способом (параллельная распределенная обработка, сеть микросвойств) [Дж. Хендлер, 1987].

Исследователи же последовательной активности, занятые планированием поведения роботов, имеют дело с проблемой правильного распределения ресурса по шагам к цели, делающим возможными более позд-

ние шаги, заканчивающиеся вознаграждением из среды [С. Уилсон, 1987].

Есть и другое применение планирования последовательного поведения с использованием распространения активности – перемещение в пространстве и времени. Реальное время – вот главная причина введения параллельного процессирования в символьные модели интеллекта. Движущийся в среде автономный робот должен планировать свой проход в реальном времени. В первой фазе планирования прохода робота воспринимаемое им пространство квантуется на узлы, описывающие объекты, и ребра, задающие проходимость этих узлов. Каждый узел заносится в свой процессор на сети параллельных процессоров в SIMD-машине. Начальное и заключительное положения робота соответствуют двум узлам сети. Во второй фазе планирования система запускает SIMD-сеть, разрешая параллельный обмен сообщениями по сети. Первое пришедшее в целевой узел сообщение и считается наилучшим. Вид сообщений позволяет восстановить путь к предплановому положению робота, так как каждый узел хранит стек сообщений. Робот передвигается, и все повторяется [М. Слэк, П. Миллер, 1987].

СТРУКТУРА, ВРЕМЯ, ЦЕЛЬ

Алгоритм в компьютере реализуется работой аппаратуры и математического обеспечения. Иначе говоря, описание алгоритма можно задать организацией процессора, памяти, программы или данных. Структура процессора и его работа задают самый твердый, закостеневший алгоритм компьютера. Данные – это преимущественно заготовки частей будущих алгоритмов, т.е. самые изменчивые структуры. Тексты программ есть всего лишь символы алгоритмов при передаче от одного компьютера (человека) к другому. Программирование – процесс передачи алгоритма из интеллекта программиста в компьютер.

Коннективизм вскрыл информационный процесс – распространение активности. Процессы в коннективистских моделях непрерывные (зацикленные) и делятся на фазы для описания их действия. Символизм занимается целенаправленной манипуляцией информационно-процессорными сущностями – алгоритмами. Но описание процесса есть структура во времени. Поэтому для создания ИИ необходимо учесть три категории: структуру, время и цель.

Именно структура позволяет принять решение, произвести выбор, отследить цель. Однако в сложных системах структуры меняются в реальном времени. Комбинаторный взрыв баз данных, экспертных систем – слишком хорошо знакомая программистам ситуация. Что должно служить целью во времени? Стратегически – неуменьшение срока существования процесса; тактически – неубывание ресурса (времени, памяти, энергии). Но если основное – неубывание, то необходимо организовать накопление и его баланс. Вот мы и подошли к статистике.

СОБИРАТЕЛЬ И ПОТРЕБИТЕЛЬ

На заре компьютерной эры была разработана концептуальная модель восприятия, в которой распознавание образов осуществлялось криками демонов в Пандемониуме [О. Селфридж, 1959]. Отдельный демон в Пандемониуме способен сопоставить предъявление с одному ему известным символом и закричать настолько сильно, насколько предъявление соответствует его символу. Психологи использовали эту модель при создании когнитивной психологии, трактующей интеллект как совокупность процессов в мозге [У. Нейссер, 1967]. Были разработаны последовательная и параллельная схемы Пандемониума [О. Селфридж, У. Нейссер, 1960]. Нужно отметить, что Пандемониум собирает в себе и распознавание образов (коннективизм), и манипуляцию символами (символизм).

Ниже изложено развитие концепции Пандемониума с уточнением его структурных, временны́х и целевых показателей. Разбирается статистическое затвердевание копий внешних относительно Пандемониума процессов с образованием интенциальной адаптированной структуры.

Рассмотрим пример собирателя книг. Пусть имеются собиратель и шкаф с книгами. Чем выше от пола полка, тем более любимые собирателем книги она содержит. Не имеет значения ни толщина книги, ни ширина книжной полки, ни число полок. Однако емкость шкафа ограничена.

Когда собиратель получает извне новую книгу и ее прочитывает, он может оценить ее и поставить на соответствующую этой оценке полку. Если на полке нет места, то собиратель выбирает наименее значимую книгу и переставляет ее на низшую по сравнению с данной полку. Этот процесс может закончиться выкидыванием самой ненужной книги из-за ограниченной емкости шкафа. Если вновь поступившая книга совпадает с какой-либо из уже имеющихся в шкафу, то ценность эталона в шкафу растет в глазах собирателя. Однако если следующее поступление не совпадает с данным эталоном, то ценность эталона падает.

Кроме того, собиратель время от времени пересматривает и переоценивает содержимое всего шкафа, сортируя при этом книги по убыванию ценности и выбрасывая "мусор". Однако такая сортировка требует веских оснований и больших затрат ресурса. Поэтому она исполняется редко, но постоянно, определяя качество коллекции в глазах собирателя. Сортировка может быть прервана без возобновления внешними и внутренними относительно собирателя причинами.

Далее, получая удовлетворение от собирательства, собиратель при каждом поступлении интересной ему книги хочет сообщить об этом другому такому же собирателю. При этом используется язык собирателя, показывающий ценность нового поступления номером полки в шкафу и состояние собирателя в момент приобретения. Сообщая эту новость, он надеется на ответное сообщение, которое может привести к новому нужному поступлению, расширяющему и углубляющему коллекцию, и приводящее к удовлетворению. Сообщение это тем сильнее, чем чаще попадалась собирателю данная книга, однако не может превышать определенный порог.

Может так случиться, что на определенном этапе собиратель полностью удовлетворит свои потребности, но сохранит желание общения. Если и дальше подкреплять этого собирателя, то основной смысл его знаний настолько закостенеет, что независимо от нового поступления его оценка будет одинаковой (и это может

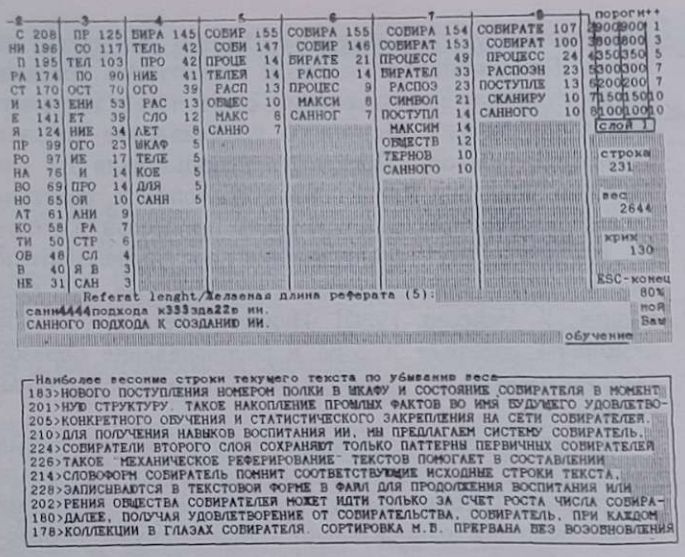


Рис. 1. Первый слой. Окончен первый проход воспитания СОБИРАТЕЛЯ текстом данной статьи. Внизу показана таблица пронумерованных по смещению в исходном тексте и отсортированных по убыванию весов строк

Рис. 2. Второй слой после первого прохода. Внизу виден пример покрытия характерной строки этой статьи для паттерна "СОБИРАТЕ" (справа вверху)

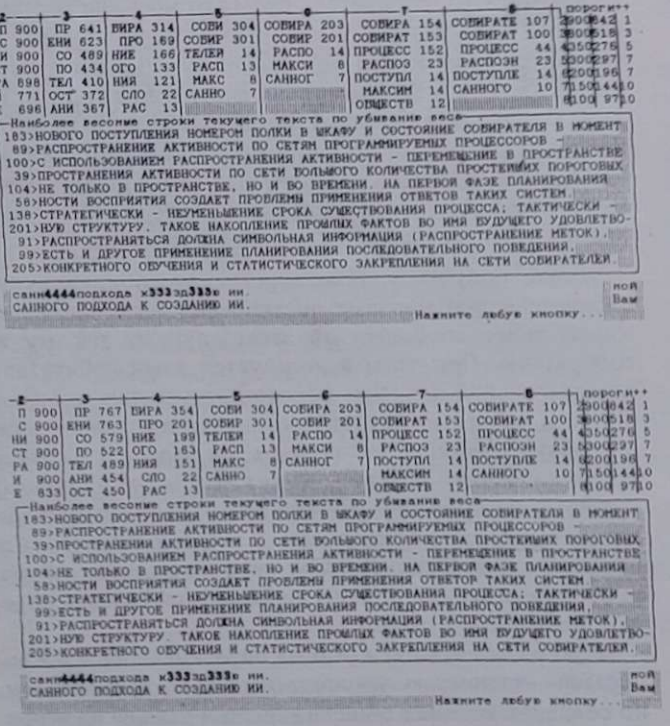


Рис. 3. Первый слой. Пятый и шестой проходы. Видны затвердевшие паттерны. Покрываемые строки стабилизируются

послужить роковой причиной беспомощности в новой важной информационной ситуации).

И наконец, могут иметься собиратели сообщений первичных собирателей любой иерархии, трактующие по принципу книжного шкафа крики сенсориума потребителям. Потребители преобразуют крики в действия.

Общество собирателей может быть представлено как один собиратель с одним книжным шкафом. При этом "закостеневшие" шкафы дадут основу более сложно устроенного шкафа, т.е. структурированная информация даст аппаратную структуру. Такое накопление прошлых фактов во имя будущего удовлетворения общества собирателей может идти только за счет роста числа собирателей. Чем их меньше, тем короче жизнь данного общества собирателей. Эта выработка интенциальной структуры идет автоматически, только за счет конкретного обучения и статистического закрепления на сети собирателей.

ПРАКТИЧЕСКАЯ РЕАЛИЗАЦИЯ

Для получения навыков воспитания ИИ предлагается система СОБИРАТЕЛЬ, написанная для ПК типа IBM PC на языке QuickBasic 4.0.

СОБИРАТЕЛЬ набирает статистику паттернов любого предъявляемого ему текста, заранее записанного в файл. За один такт СОБИРАТЕЛЬ сканирует одну строку текста. Для заученных словоформ СОБИРАТЕЛЬ помнит соответствующие исходные строки текста, облегчая процедуру воспитания.

Структурно СОБИРАТЕЛЬ состоит из двух слоев. В каждом слое содержится по семь собирателей, каждый из которых снабжен регистром состояния и магазином для сбора коллекции. Регистр состояния хранит количество доступного в данный момент каждому собирателю ресурса. Количество коллекционируемых паттернов с соответствующими им счетчиками частости не превышает 30. Доступна регулировка порогов затвердевания каждого собирателя.

Первый слой — сенсорный. Каждый собиратель первого слоя сканирует текущую строку текста в двух фазах: распознавании и обучении. Если паттерн распознан, формируется крик, доступный всем остальным собирателям, а этот собиратель получает подкрепление и заканчивает работу. Если нет распознавания, запускается фаза обучения с расходом ресурса данного собирателя. Собиратели второго слоя сохраняют только паттерны первичных собирателей с максимальной частостью (локальные максимумы).

На рис. 1 показана копия с экрана карты собирателей первого слоя во время работы (фаза обучения). Слева направо по возрастанию длины паттерна и сверху вниз по убыванию частости расположены коллекции собирателей. Снизу в строке "Ваш" показана предъявляемая строка. Над ней, в строке "мой", показано покрытие данной строки после текущего обучения всех семи собирателей первого слоя (цифры указывают распознавшего собирателя). Здесь же справа показаны процент покрытия, крик последнего распознававшего собирателя, вес всей распознанной строки и ее смещение в исходном тексте. Справа вверху показаны пороги затвердевания и шаги приращения частостей соответствующих собирателей.

На рис. 2 показан второй слой. Пользователь имеет возможность произвольно просматривать контекст каждого паттерна (его строку, процент покрытия и вес по второму слою). Чем больше предъявляется текст, тем больше веса статистически значимых паттернов. Они затвердевают (рис. 3).

Приведенные рисунки иллюстрируют воспитание СОБИРАТЕЛЯ текстом данной статьи. Такое "механическое реферирование" текстов помогает в составлении различных словарей и классификаторов баз данных. Затвердевшие таблицы прямо применимы к кодированию текстов. Результаты обучения записываются в текстовой форме в файл для продолжения воспитания или прикладного использования.

Планируется дальнейшее совершенствование этого пакета в русле описанного подхода к созданию ИИ.

ЛИТЕРАТУРА

1. Фролов А. А., Муравьев И. П. Нейронные модели ассоциативной памяти. – М.: Наука, 1987.
2. Гелернтер Д. Современное программирование//В мире науки. – 1987. – № 12. – С. 37–45.
3. Dutta A. The Explicit Support of Human Reasoning in Decision Support Systems// Adv. Comput. – 1987. – Vol.26. – P.1–45.
4. Feldman J. A. Connectionist Models and Their Applications: Introduction//Cognitive Science. – 1985. – №9. – P.1–2.
5. Hendler J. A. Marker-Passing and Microfeatures //IJCAI 87: Proc. 10rd Int. Joint Conf. Artif. Intell., Milan. – Los Altos, Calif., 1987. – Vol.1. – P.151–154.
6. Massaro D. W. The Computer as a Metaphor for Psychological Inquiry: Considerations and Recommendations//Behavior Research Methods, Instruments, & Computers. – 1986. – Vol.18, №2. – P.73–92.
7. Minsky M., Papert S. Perceptrons. Cambridge, MA:MIT Press, 1968.
8. Newell A., Simon H. A. The Logic Theory Machine//IRE Transactions on Information Theory. – 1956. – Vol.2, №3. – P.61–79.
9. Perlis D. How Can a Program Mean? IJCAI 87: Proc. 10rd Int. Joint Conf. Artif. Intell., Milan. – Los Altos, Calif., 1987. – Vol.2. – P.163–166.
10. Rosenblatt F. The Perceptron:A Probabilistic Model for Information Storage and Organization in the Brain//Psychological Review. – 1958. – №65. – P.386–407.
11. Selfridge O. G. Pandemonium: A Paradigm for Learning in Mechanization of Learning Processes. – London: Her Majesty's Stationary Office, 1959.
12. Selfridge O., Neisser U. Pattern Recognition by Machine//Scientific American. – 1960. – №203(2). – P.60–68.
13. Neisser U. Cognitive Psychology. – NY, Appleton, 1967.
14. Shannon C. E. The Mathematical Theory of Communication// Bell System Technical J. – 1948. – №27. – P.379–423, 623–658.
15. Shastri L. Massive Parallelism in Artificial Intelligence//Appl. Opt. – 1987. – Vol.26. – №10. – P.1829–1844.
16. Slack M. G., Miller P. D. Path Planning through Time and Space in Dynamic Domains//IJCAI 87:Proc. 10rd Int. Joint Conf. Artif. Intell., Milan. – Los Altos, Calif., 1987. – Vol.2. – P.1067–1070.
17. Wilson S. W. Hierarchical Credit Allocation in a Classifier System//IJCAI 87: Proc. 10rd Int. Joint Conf. Artif. Intell., Milan.– Los Altos, Calif., 1987. – Vol.1. – P.217–220.

Будущее Пролога

РОБЕРТ ЛЬЮИС

Если вы ждете появления таких экспертных систем, которые окажут вам существенную помощь в принятии решений, познакомьтесь с языком Пролог, который пока не оценен по достоинству, но его час пробьет в недалеком будущем.

Намерение дать точное определение искусственному интеллекту (ИИ) напоминает попытку покойного Верховного судьи Поттера Стьюарта охарактеризовать непристойность. Стьюарт в шутку сказал, что он не смог бы определить это понятие, однако добавил: "Я отличу непристойность, когда я ее увижу".

Хотя имеющее смысл определение ИИ может ускользать от сторонников точных формулировок, сообщество пользователей персональных компьютеров знает, что искать, и в его распоряжении появляется все больше средств для достижения этого. Специалисты в области информатики указали на способности машин моделировать человеческие рассуждения, обучение и общение как на характерные черты ИИ. После периода, который некоторые наблюдатели называют периодом многообещающих стартов и невыполненных обещаний, достижения в области ИИ уверенно шагнули из лабораторий в общий поток приложений персональных компьютеров (ПК).

Терминология ИИ забросала ныне профессиональный жаргон, принятый в области ПК, такими терминами, как *инженерия знаний, Лисп, эксперт в прикладной области, Институт вычислительной техники нового поколения (ICOT)* и *синтаксический анализатор естественного языка.* Искусственный интеллект является кормушкой для знатоков промышленности, делающих бизнес на предсказаниях. "Присматривайтесь к технологии искусственного интеллекта, чтобы заработать миллиарды к 90-м годам", – предсказывает один аналитик. "Вы уже сейчас можете запрограммировать свой компьютер для ведения диалога на естественном языке", – провозглашает другой ученый муж. Между тем рынок ПК приветствует семейство аппаратных средств, предназначенных для ИИ, например таких, как плата HummingBoard для ПК с микропроцессором 80386 фирмы Golg Hill, позволяющая программировать на языке Лисп, реализованном специально для этого микропроцессора.

Однако, возможно, наиболее впечатляющим знамением времени является неожиданно бурный рост числа выполненных на ПК реализаций Пролога – языка искусственного интеллекта. Фирма Borland International финансировала разработки компилятора Turbo-Prolog, который появился в продаже в апреле 1985 г. Эта, быть может, немного необычная реализация Пролога стоимостью всего 99 дол обес-

печивает многооконный интерфейс и графические возможности. Фирмы Automata Design и Chalcedony Software также предлагают недорогие высокомощные версии Пролога.

Предлагая терминологию для ИИ, язык типа Пролога является естественным средством реализации экспертных систем (которые перерабатывают правила и факты в человеческой манере), интерфейса на естественном языке, управления данными, моделирования и других приложений, используемых при распознавании образов и логическом выводе.

Хотя пользователь ПК может работать с инструментарием, необходимым для написания программ с использованием концепций ИИ, не нужно полагать, что из каждого ПК AT должна родиться экспертная система. Ценность Пролога – не в широкой доступности для массового пользователя, а в его возможностях как сложного и тонкого инструмента для разработки высокоспециализированных систем ИИ.

Основополагающие работы по Прологу для ПК ведутся такими фирмами, как Lotus Development и Expert Systems International. Фирма IBM вышла на рынок с системой VM-Prolog – реализацией Пролога для больших вычислительных машин. По словам одного промышленного аналитика Б. Блу, создается ощущение, что "малые системы искусственного интеллекта" отстают на 3–4 года.

СТРУКТУРА ЯЗЫКА ИИ

Языки ИИ резко отличаются от традиционных языков программирования высокого уровня: они скорее *описательны*, чем *процедурны*. Работая с декларативным языком, программист определяет "проблемную область" (т.е. параметры задачи), а не описывает алгоритм (упорядоченное множество инструкций для решения задачи). Языки, используемые в ИИ, в основе своей декларативны: они манипулируют символами, а не числами.

Работая с символами, программист должен описывать объекты качественно, а не количественно. Символьные языки поддерживают известные отношения, такие как "=" и ">", и программы, написанные на языках ИИ, часто содержат вычислительные процедуры. При этом программист ограничен символьными преобразованиями и может попасть в затруднительное положение, если оказывается, что в базу знаний необходимо поместить такое понятие, как "приблизительно". Неожиданно программист попадает в беспорядочный, многословный, почти субъективный мир. Для того чтобы программисты (а также и пользователи) могли с этим справиться, языки ИИ должны обладать специальными возможностями.

Декларативное программирование требует способности представления данных в базе знаний программы с использованием различных типов данных. Язык символьной обработки должен быть диалоговым, он должен как обеспечивать возможность поиска по образцу, так и обладать адекватными средствами для построения структур, основанных на знаниях. Он должен уметь соединять процедуры и знания. И наиболее важным представляется то, что он должен быть рекурсивным. Операция, в которой содержится вызов самой себя, является рекурсивной. В отличие от цикла GOTO языка Бейсик, который описывает некоторое число итераций, функции в Прологе могут вызывать сами себя; в Прологе такой цикл контролируется некоторой логической структурой, останавливающей рекурсию.

КОНКУРЕНЦИЯ ЯЗЫКОВ ИИ

Пролог, созданный как средство *лог*ического *про*граммирования (*pro*gramming in *log*ic), и Лисп – сокращение, образованное от словосочетания "List processing", являются основными языками ИИ. Лисп, разработанный в конце 50-х годов в Массачусетском технологическом институте, – долгожитель среди языков программирования и уступает лишь почтенному Фортрану. Пролог был разработан в Марсельском университете в начале 70-х годов и стал наиболее распространенным в Европе языком ИИ.

Хотя различия между любыми двумя языками программирования проявляются в нескольких областях, но то, что относится к происхождению Лиспа и Пролога, проливает некоторый свет на расхождения в них. В США, где возраст Лиспа перевалил за десять лет, исследователи ИИ имеют доступ к огромному объему памяти и вычислительной мощности больших ЭВМ. Лисп как серьезный потребитель вычислительных мощностей породил целое семейство мало похожих систем, предназначенных для выполнения программ, написанных на языке Лисп, и оптимизации самого языка. В будущем возможно появление реализаций Лиспа в виде плат-сопроцессоров и микросхем.

Так как вычислительная мощность и память в Европе стоят больше, исследователи разработали здесь Пролог, который во всех отношениях более пригоден для реализации на ПК. В дополнение к тому, что Пролог менее требователен к памяти, чем Лисп, он, по крайней мере внешне, еще и легче изучается. Но несмотря на популярность Пролога в Европе и его общепризнанный статус языка ИИ, выбранного пользователями ПК, Лисп остается главенствующим, особенно в США.

В 1980 г. японцы выбрали Пролог в качестве языка для своего Национального проекта развития суперЭВМ пятого поколения. Такой выбор, по-видимому, означает, что Пролог в некоторой степени превосходит все остальные языки. Поэтому ведущие разработчики в области ИИ в США взглянули на Пролог более серьезно. Несмотря на общий интерес к Лиспу и Прологу, большинство программ для ПК из области ИИ пишется на Си, Паскале, языке ассемблера и даже Форте. Intelligent Assistant – интерфейс на естественном языке для СУБД Symantec's Q&A – был написан на Си и языке ассемблера, затем приукрашен специализированными программами на Лиспе. M.1 – пакет инструментальных средств для разработки экспертных систем фирмы Teknowledge – первоначально был разработан на Прологе, но позднее был перетранслирован на Си.

ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ НА РАСПУТЬЕ

Хотя Пролог и Лисп основаны на весьма разных теоретических предпосылках, оба языка можно использовать для решения одних и тех же задач. Они оба обладают специфическими преимуществами в решении задач ИИ.

Лисп является удачно разработанным средством обработки списков; он позволяет и данные, и процедуры представлять общей списковой структурой, которая также мо-

жет представлять декларативные и процедурные элементы программы. Списки широко используются для представления характеристик проблемных областей ИИ, потому что данные программ ИИ, в отличие от данных в других приложениях, должны позволять представление неупорядоченных объектов: языка, семантических нюансов и символов, а не чисел – и выражать логические отношения между ними, которые по математическим стандартам являются нечеткими. Списковая структура может эффективно представлять разнообразные типы данных. И так как программа ИИ может изменить или реорганизовать свою управляющую структуру в процессе работы, возможность представлять списками как управление, так и данные является бесспорным преимуществом.

Пролог разделяет с Лиспом эти способности, но имеет заметно другую структуру: это скорее язык отношений, чем списков. Программы на Прологе не похожи на программы, написанные на других языках (рис. 1). Простая программа на Прологе состоит из базы знаний, скомплектованной из двух частей: правил и фактов. Правила манипулируют фактами, которые хранятся в базе знаний (см. вставку "Как программировать на Прологе").

ОБРАЩЕНИЕ К ЭКСПЕРТАМ

Практической реализацией ИИ явились экспертные системы. Разработанные частными фирмами, вышедшие из университетских лабораторий и покровительствуемые фирмами по разработке программного обеспечения, экспертные системы оказывают содействие в проектировании, производстве, планировании и диагностике (см. статью "Experts on Call" в сентябрьском номере журнала PC World за 1985 г.).

Пол Хармон, соавтор книги Expert Systems (John Wiley and Sons, New York, 1985) и издатель газеты Expert Systems Strategies, насчитал около 150 действующих коммерческих экспертных систем, подавляющее большинство которых было написано на Лиспе. По другим данным их насчитывается более 300.

Согласно подсчету Хармона до настоящего времени было создано лишь небольшое число экспертных систем, реализованных на Прологе. Они включают программу загрузки самолетов AALPS, разработанную военными фирмами и фирмой Quintus Computer Systems, программу Beacon, разработанную фирмой Burroughs для помощи в комплектовании аппаратуры больших вычислительных машин этой фирмы, консультативный пакет программ Component Impact Analysis System для операторов атомных электростанций, разработанный на языках Си и Quintus-Prolog Аргоннской национальной лабораторией, и программу Train Travel Advisor фирмы Thomas Cook Travel, которая помогает агентам бюро путешествий в планировании маршрутов (см. вставку "Пролог на практике").

Экспертные системы различаются размерами, определяемыми числом правил или тем, что известно как "эффект телефонного звонка": если вы советуетесь с экспертом по телефону и решаете свою задачу в течение 20 мин, значит, вы раз-

Рис. 1. Простая база знаний на Прологе – описание дома Хозяина

```
дом (красный)

владеет (хозяин) :-
        дом (красный).

двор ((дом (красный)), большой,
        плавательный_бассейн).


спальня (владеет (хозяин), 4).
кухня (владеет (хозяин), плита,
        холодильник).
столовая (владеет (хозяин),
        стол (дубовый), стул, 8,
        столовый_прибор, 8).
гараж (владеет (хозяин), машина,
        верстак, инструменты).
гостиная (владеет (хозяин), телевизор,
        видео, стерео, бильярд,
        бар (спиннинг, 3)).
живут (владеет (хозяин), хозяин,
        жена (гвен), дочь (бенни),
        дочь (мегги), сын (ленни)).

большой_дом (X) :-
        спальня (владеет (X), 4).

хорошо_жить (X) :-
        двор (_,_, плавательный_бассейн),
        гостиная (владеет (X),_,_,_,_,
                бар (спиннинг, 3)),
        гараж (владеет (X),_, машина,_,_,_).

вызывать_зависть (X) :-
        большой_дом (X),
        хорошо_жить (X).
```

Комментарии

Факт: красный дом.

Правило: хозяин владеет красным домом.

Факт: у красного дома есть большой двор с плавательным бассейном.

В доме у хозяина четыре спальни.
Далее следует группа фактов равной сложности. Число аргументов, которые появляются внутри круглых скобок, произвольно, хотя в этой простой программе владение определяется только одним аргументом.

Правила также могут быть простыми или сложными. Правило с предикатом "большой_дом (X)" является простым, а с предикатом "вызывать_зависть" – сложным:
оно построено из двух других правил – "большой_дом" и "хорошо_жить".
Подстановка "_" на место переменной или объекта предписывает программе просматривать только названные объекты или переменные. Это упрощает программирование и исполнение.
Значение X, возвращаемое во всех примерах, есть хозяин.

Как программировать на Прологе

Программирование на Прологе предполагает неортодоксальный подход к решению задач. При этом программист описывает скорее постановку задачи, а не четкий способ ее решения.

Основной единицей данных в базе знаний Пролога является факт, который соответствует записи в традиционной базе данных. Ограничения фактов в Прологе произвольны и устанавливаются программистом.

Основной формой записи факта является следующая:
предикат (аргумент 1, аргумент 2, . . . , аргумент x). Факт может быть сложным, например:
крикет_площадка (1-е_место_старта, 2-е_место_старта, краткая_остановка, 3-е_место_старта). Или он может быть простым:
ошибка (краткая_остановка).

Предикат фиксирует, что отношение существует, а аргументы обозначают объекты или значения, которые входят в отношение. Аргументы внутри скобок разделяются запятыми. Порядок аргументов произволен, хотя могут появляться определенные соглашения, как это принято в большинстве языков. (Заметим, что в Прологе требуется, чтобы аргументы и предикаты были записаны без пробелов; поэтому слово "1-е" должно соединяться со словом "место" с помощью знака "–".)

Таким образом, факты фиксируют характеристики объектов и их связи. Чтобы построить базу знаний об избранных домах вашего квартала, вы можете описать их по цветам:

дом (красный)
дом (белый)
дом (бело-зеленый)

Эти сведения могут отметить качественные различия между домами определением отношения "быть домом" в терминах цветов.

Для представления более сложной информации относительно домов на вашей улице вы могли бы указать владельцев. Вы могли бы начать сообщать ПК, кто владеет домом:

владеет (хозяин,дом)

Если бы вы хотели быть еще более конкретным, то могли бы констатировать:

владеет (хозяин,дом (красный))

Это последнее выражение демонстрирует способность Пролога трактовать отношение между фактами как единый объект, когда это отношение используется в качестве аргумента. Информация может иметь любую глубину вложенности; единственным ограничением является память системы.

Как только ваша база знаний загружена, вы можете задавать вопросы, используя возможности поиска в Прологе. Вы запрашиваете базу знаний, является ли Хозяин владельцем дома:

владеет (хозяин,дом)?

Компьютер отвечает "да", подтверждая, что вы уже поместили этот факт в базу знаний. Пролог демонстрирует свою способность сопоставления с образцом.

Формулировка запросов является прекрасным упражнением, но она быстро наскучивает. Чтобы повысить уровень общения с вашей базой знаний, вы можете объявить правила для данных. Правило выглядит как пример, в котором факт или группа фактов определяют другой факт. Чтобы спросить, например, каких цветов имеются дома, требуется другая форма:

дом_цвета (Цвет) :–дом (Цвет).

Прописная буква Ц в слове "Цвет" позволяет Прологу понимать Цвет как переменную. (В отличие от переменных в Бейсике и других языках, переменные в Прологе локальны, т.е. имеют значение только в выражениях, в которые они входят.) Компьютер выдаст:

Цвет = красный

Если вы спрашиваете, каких других цветов имеются дома, то, следуя тому же принципу, вам объявят, что Цвет равен белому и бело-зеленому. Это правило Пролога (иногда рассматриваемое как правило "ЕСЛИ . . . ТО") имеет форму, показанную на рис. А.

Правило интерпретируется следующим образом: "Если хвост (правая часть выражения) или условие "ЕСЛИ" удовлетворяются, то имеет место событие, стоящее после "ТО". Правила могут строиться из очень сложных хвостовых структур. Голова (левая часть выражения) заработает, когда удовлетворятся условия, т.е. когда все утверждения из хвоста будут найдены в базе знаний. Таким образом, возбуждением головы путем удовлетворения условий из хвоста программист может преобразовать задачу, научить базу знаний новым фактам или известить пользователя о том, что возможности программы уже исчерпаны и наступило время добавить что-либо еще.

В нашем примере программа выполняется следующим обра-

голова	если	хвост
дом_цвет (Цвет)	:–	дом (Цвет)

Рис. А

зом. Сначала она находит символ :- (символ "ЕСЛИ"). Этот символ, по существу, говорит: "Пролог, у тебя есть цель. Цель состоит в определении того, находятся ли в базе знаний факт или факты, расположенные в правой части". Программа исследует базу знаний, проверяя это. Если соответствие находится, то голова возбуждается и вы получаете сообщение. Когда Пролог ищет соответствие, он использует стратегию поиска, известную под названием поиск с возвратом.

Поиск с возвратом начинается с первого справа от символа "ЕСЛИ" выражения. Каждый факт в базе знаний проверяется до тех пор, пока не будет установлено соответствие. Если первый факт соответствует, то он маркируется и программа переходит к следующему справа утверждению. Поиск продолжается, пока для всех фактов из хвоста соответствие не будет установлено. Если для данного утверждения соответствие не находится, то поиск возвращается к предыдущему, уже установленному соответствию и подбирает другое.

Это преобразование целесообразно, не случайно; выполнение программы управляется моделью, которая определяется правилом (хвостовыми условиями). Именно в результате определения соответствия добавляется новая информация, и программа обучается.

Все коды программы на Прологе отличаются своим внешним видом и ведут себя по-разному во время выполнения.

Р. Л.

говариваете с тем, кто является лишь посредственным экспертом. Число правил, конечно же, не может быть точной мерой сложности программы; 1000 однострочных правил могут занимать такую же область, как и 200 многострочных.

Большая часть сегодняшних экспертных систем громоздки, дорогостоящи и приспособлены для работы на больших ЭВМ или Лисп-машинах. Хармон предполагает, что рабочие станции с операционной системой UNIX, позволяющие выполнять программы на языке Лисп, уже вытесняют некоторые Лисп-машины – тенденция, в соответствии с которой реализации Лиспа в виде микросхемы и ПК с микропроцессором 80386 вполне пригодны для дальнейшего повышения производительности.

Согласно этой тенденции Пролог сейчас "поднимает голову", чтобы бросить вызов гегемонии Лиспа, как следует из слов распространителей Пролога. Кажется, по крайней мере внешне, что это утверждение имеет под собой основу. Пролог предлагает различные реальные выгоды для развития экспертных систем.

ДОСТОИНСТВА ПРОЛОГА

Из вставки "Как программировать на Прологе" видно, что предложения Пролога – превосходное изобретение для представления знаний и записи правил: два главных требования для большинства экспертных систем. Например, запись правила на Прологе очень похожа на основной оператор "ЕСЛИ... ...ТО". В дополнение к этому факты в Прологе переводятся в отношение объект-свойство-значение (О-С-З). Структура отношения О-С-З является подмножеством семантической сети – мощного средства представления знаний в ИИ.

Отношение О-С-З строится по объекту, на который ссылается предикат, описывающий свойство или атрибут этого объекта (рис. 2). Атрибут, в свою очередь, находится под влиянием значения. Легко видеть, что это разумно. В отличие от факта, атрибутом является глагол "иметь" или "являться". В совокупности атрибуты создают множество отношений, многие из которых подразумеваются; если объект имеет голень, колено и бедро и все они соединены, то между ними предполагается связь. Это и есть семантическая сеть, состоящая из троек О-С-З.

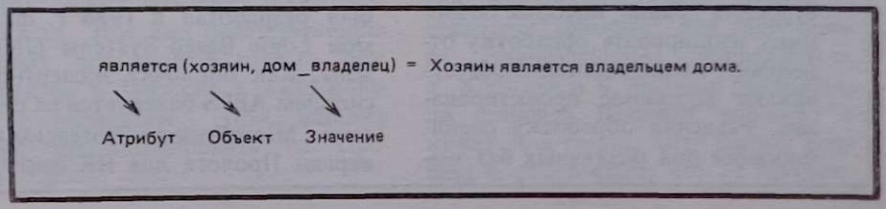
Хотя Пролог и предполагает некоторые выгоды по сравнению с Лиспом, по крайней мере на системах ПК, состояние современной индустрии ИИ таково, что экспертные системы независимо от реализации не могут еще стать универсальным средством. В основном экспертные системы, реализованные на ПК, ограничены узкоспецифическими применениями, и чем более специфична задача, тем лучше работает система.

Многие из этих средств могут рассматриваться как оболочки или подмножества языка, от которого они произошли. Оболочки – суть простые системы представления фреймов, которые обеспечивают требования "консультативной парадигмы" – обычно предсказание или диагноз.

Система ESP Advisor, разработанная фирмой Expert Systems International (ESI), является одной такой оболочкой, основанной на Прологе. Реализованная в 1984 г. система EPS Advisor была написана на версии Пролог-1 и недавно реализована на более мощном Прологе-2. Президент фирмы ESI Анджело Колокури утверждает, что развитие конкурентоспособной промышленной версии Пролога сдерживалось отсутствием соответствующих мощных программных средств – того, что сейчас обеспечивает Пролог-2.

Колокури говорит: "В новом Прологе модульная среда програм-

Рис. 2. Эта диаграмма объект-свойство-значение иллюстрирует семантическую сеть Пролога – способ группировки фактов и атрибутов, которая строит отношения и представляет знания в экспертной системе.



Пролог на практике

Хотя область экспертных систем не изобилует коммерческими приложениями Пролога, она оказывается весьма благодатной почвой для разработчиков. Большие экспертные системы, написанные на Прологе, работают в фирмах Boeing Aircraft, SRI International и Аргоннской национальной лаборатории.

В штаб-квартирах фирмы SRI International (Менло-Парк, шт. Калифорния) Пролог исправно служит в экспертных системах, моделирующих технические разработки, рыночную ситуацию в нефтехимической промышленности. Пролог является средством реализации интерфейса с существующей иерархической моделью, запрограммированной на языке Си. Полученная в результате система моделирует рыночную ситуацию при появлении новой продукции.

В экспертной системе фирмы SRI Пролог анализирует большую базу данных промышленных предприятий, продукции и технологии, а также управляет основанной на правилах системой логического вывода, которая руководит принятием обобщенных решений в торговом мире. Согласно мнению Я. Сайеда, аналитика из фирмы SRI, Пролог хорош, "когда мы не знаем, сколь жесткими будут ограничения. Пролог дает возможность формулировать те условия, которые позволят варьировать поиск решения".

В фирме Boeing Пролог является "сердцем" экспертной системы CASE (Connector Assembly Specification Expert), помогающей выбирать подходящее оборудование и материалы для сборки электрических соединителей. Система тщательно просматривает десятки тысяч страниц, чтобы выделить спецификации, которые управляют этой сборкой. Пролог "заработал себе на жизнь" своими простыми средствами построения правил, которые позволяют изолировать обработку отдельных спецификаций и обеспечивают поэтапное проектирование. Разделяя обработку спецификаций для различных баз знаний, проектировщик имеет возможность упростить процесс написания и отладки программ. Более того, интерактивные возможности Пролога позволяют пользователям находить правильные спецификации, осуществляя запрос к программе.

Вскоре может быть предложена программа на Прологе в качестве руководства для операторов атомных электростанций. Создаваемая система реализует то, что было разработано филиалом Аргоннской национальной лаборатории Argonne West (Аргонн, шт. Иллинойс). Эта программа состоит из системы поддержки решений, которая реализует сенсорное чтение и обеспечивает вывод на графический дисплей, изображение на котором демонстрирует операторам положения вентилей и переключателей в соответствии с требованиями производства. Как и в системе фирмы SRI, возможность Пролога работать с компонентами системы, написанными на языке Си, обеспечивает здесь большие преимущества.

Все три упомянутые системы слишком претенциозны для обычных ПК, так как требуют много аппаратуры. Все они были разработаны на языке Quintus-Prolog, продаваемом фирмой Quintus Computer Systems (Маунтин-Вью, шт. Калифорния). До сих пор применение языка Quintus-Prolog ограничено ЭВМ VAX фирмы DEC, Sun фирмы Microsystems и RT фирмы IBM. Представитель фирмы Quintus Дебби Стафф говорит, что фирма рассматривает возможность переноса своего Пролога на компьютеры на основе микропроцессора 80386.

Если вы ищете набор средств для реализации и последующего использования экспертной системы, вы можете рассмотреть пакет программ APES (Augmented Prolog for Expert Systems). Он был разработан в 1984 г. фирмой Logic Based Systems (Лондон). Как оболочка экспертной системы APES базируется на системе MicroProlog Professional, версии Пролога для ПК фирмы Logic Programming Associates, также расположенной в Лондоне.

По словам Антони Ковальского, президента фирмы Programming Logic Systems – американского распространителя пакета APES, его применения простираются от лечения диабета до проектирования автодорожных подпорных стенок. "Написанные на Прологе прикладные программы, реализующие выражения, основанные на правилах (включенных в APES), во много раз меньше, чем эквивалентные программы, написанные на процедурных языках", – говорит он.

Особенности пакета APES – это встроенные в систему средства, включающие транслятор с естественного языка, что позволяет программисту вводить правила и факты, используя предложения, похожие на английские. Простое правило для программы медицинской диагностики может быть записано следующим образом:

Пациенту [переменная] советуют принять лекарство [переменная], если у пациента имеется симптом [переменная] и это лекарство подавляет этот симптом и препарат не причинит вреда пациенту.

Другими особенностями пакета APES являются:

Система запросов пользователя. Эта элементарная обучающаяся система запрашивает у вас информацию, недостающую программе. Если для решения задачи программе нужны факт или утверждение, APES запрашивает информацию и использует ее в своем решении.

Средства достаточного обоснования. Когда APES достигает заключения, вы можете попросить программу восстановить и прокрутить в обратном порядке шаги, которые привели к данному заключению.

Ответвление. Программа может обращаться к файлу вне APES, используя его как средство для генерации вложенных объяснений конкретного события.

Р. Л.

мирования обеспечивает многооконный интерфейс и графические возможности". Располагая графическими возможностями Пролога-2, программист может создавать логические отношения, что интерпретатором Пролога будет прочитано и преобразовано в форму правил, основанных на знаниях. Компилируемый Пролог-2, как сообщается, превосходит в скорости интерпретируемый Пролог-1 в 20 раз. Фирма Arity, отделение фирмы Lotus Advanced Development Group, предложила Arity/Prolog – программную среду, содержащую все необходимые средства для разработки программ на Прологе. Кроме того, среди разработок фирмы Arity имеются пакет для разработки экспертных систем и средства для создания экранных форм.

На повестке дня фирмы Arity стоит расширение оболочек экспертных систем, основанных на Прологе, для промышленного использования. Фирма поговаривает о том, что называют "удобная профессиональная программная среда для конечных пользователей любого уровня". Между тем появление Arity/Prolog было встречено с большим одобрением, что позволило фирме Arity утроить свои деловые планы в работе с оболочками.

Фирмы Arity и ESI свои планы целиком связывают с Прологом. В этом отношении они решительно отличаются от фирмы Teknowledge – другого разработчика экспертных систем, которая сосредоточила свои силы на радикально другом направлении. M.1 – пакет экспертной системы этой фирмы, первоначально написанной на Прологе-1, – был переписан на Си. Фирма Teknowledge пытается достичь своей цели по максимизации возможностей для интеграции программного обеспечения. Перевод на Си системы M.1 позволил ускорить ее работу более чем в 4 раза по сравнению с ее предшественницей, написанный на Прологе-1.

ОБЩЕНИЕ НА ЕСТЕСТВЕННОМ ЯЗЫКЕ

Общение с ПК на естественном языке напоминает процесс получения прямого ответа от политика. На существующем уровне развития и доступности технологии ИИ ее возможности для решения проблемы понимания естественного языка еще не ясны. Трансляторы с естественного языка используются в нескольких базах данных, но вопрос об их надежности и гибкости весьма спорен. Пролог может в конечном счете изменить это положение.

Система Intelligent Assistant, разработанная фирмой Symantec, является базирующимся на ПК интерфейсом, который позволяет пользователям посылать команды в базу данных Q & A на упрощенном английском языке. Гэри Хендрикс, разработчик системы, при реализации интерфейса на языках Си, Лисп и языке ассемблера опирался на использование так называемой грамматики с фазовой структурой. По словам Хендрикса, Пролог базируется на другой теоретической схеме, которая имеет свои преимущества.

При формулировании задачи написания программы, понимающей естественный язык, Пролог позволяет использовать логические конструкции. Согласно Хнедриксу "вы пытаетесь доказать теорему, которая гласит, что строка слов, составляющих входное предложение, действительно является предложением".

Фернандо Перейра, специалист фирмы SRI International, принимавший участие в разработке первого компилятора Пролога, является лидером в применении Пролога в исследованиях, связанных с естественным языком. "Пролог особенно хорошо подходит для обработки запросов к базам данных", – говорит Перейра.

Согласно Перейре сформулировать проблему запросов к базе данных на естественном языке – это значит разработать множество правил, которые разбивают предложение на части и затем по смыслу этих частей определяют смысл всего предложения. Само предложение не определяет однозначно соединения этих частей. Структура упомянутых правил похожа на предложение Пролога "ЕСЛИ . . . ТО".

"При избытке правил, – говорит Перейра, – по-видимому, нет другого простого пути их общего согласования, за исключением того, чтобы опробовать несколько альтернатив. Этот вид недетерминированных применений правил осуществлен в Прологе в виде поиска с возвратом". (См. вставку "Как программировать на Прологе".)

Пролог обладает дополнительными преимуществами, которые позволяют сосуществовать в одной системе средствам обработки информации синтаксической (организующей), семантической (смысловой) и прагматической (реальный мир). Вспомните, что критерии для языков ИИ включают способность распознавать различные типы данных и смешивать данные и процедуры.

Интерфейс, подобный Intelligent Assistant, требует отдельных устройств для каждого вида обработки. В системе с естественным языком, построенной на базе алгоритма унификации Пролога, а именно поиска с возвратом, вы можете, как говорит Хендрикс, "бросить все в один котел и заставить алгоритм унификации разобраться во всем этом для вас".

Означает ли это, что вскоре вы будете иметь возможность использовать универсальный интерфейс на естественном языке, выполненный на Прологе, для вашей любимой программы обработки табличных данных? В обозримом будущем – нет. Интерфейсы на естественном языке в настоящее время ограничены узкоспециализированными применениями. Проектирование универсальных систем потребовало бы от программистов внести ясность в запутанный мир естественного языка, который содержит все возможные логические системы.

ПОПУЛЯРНЫЙ ПРОЛОГ

Фирма Borland International представила свой Turbo-Prolog как компилятор широкого использования, и президент фирмы Филипп Кан проявил намерение расшевелить специалистов в области ИИ. Некоторые полагают, что Кан пытается воспроизвести стратегию разработки компилятора Turbo-Pascal, привлекая новых пользователей и существенным образом переориентируясь на новый язык. Полное описание инструментальных средств компилятора Turbo-Prolog и их применения находится в стадии разработки.

Известность фирмы Borland позволяет ей поднимать вопрос о стандарте Пролога. Два профессора Эдинбургского университета У. Клоксин и К. Меллиш разработали стандарт Пролога де-факто в конце 70-х годов. Хотя фирма Borland и утверждает, что ее продук-

ция является "широким надмножеством" Эдинбургского стандарта, Turbo-Prolog не полностью удовлетворяет условиям, сформулированным Клоксином и Меллишем. Однако другие недорогие внедрения Пролога являются сходными со стандартом Клоксина и Меллиша: фирма Chalcedony Software продает "вылизанную" версию интерпретатора Prolog i менее чем за 100 дол. Фирма Automata Design предлагает несколько версий своего Пролога по ценам, изменяющимся от 29 до 200 дол. Фирма Arity даже переманивает пользователей у фирмы Borland, призывая их перейти к более традиционной среде Пролога и предлагая 50-долларовую скидку.

Для того чтобы Пролог смог действительно занять центральное место, он должен избавиться от репутации дублера Лиспа. Стивен Харди из фирмы Teknowledge, который возглавляет группу разработчиков версии М.1 Пролога, предполагает, что некоторые из ранних преимуществ языка утрачены. Так как стоимость оборудования уменьшилась, рассуждает Харди, "необходимость работать в условиях ограничений, существовавших в оборудовании 1970-х гг., снизила ценность Пролога как среды для разработчика экспертных систем".

Харди указал на две другие возможные проблемы, связанные с Прологом как языком решения прикладных задач. Прежде всего это сложность управления поиском с возвратом в случае больших задач. Кроме того, Харди описывает Пролог как язык, имеющий два пути развития. Прикладное или процедурное программирование на Прологе более сложно, чем работа в декларативном режиме. Фернандо Перейра сказал по этому поводу следующее: "Изучение Пролога подобно восхождению на пологую гору. Вы достигаете ее вершины и вдруг обнаруживаете огромную скалу. Вам нужно карабкаться на скалу и даже еще выше, если вы желаете достичь чего-либо серьезного в разработке искусственного интеллекта".

ПОНИМАНИЕ ПРОБЛЕМЫ — ПУТЬ К РЕШЕНИЮ

Гэри Хендрикс из фирмы Symantec рассматривает Пролог как язык с уникальными возможностями построения "моделей количественных фактов", что успешно демонстрируется приведенным примером с домовладельцем (см. вставку "Как программировать на Прологе"). В сущности, все, что может быть описано повествовательным предложением, может быть представлено и обработано на Прологе.

Программисты, работающие на Прологе, должны быть в некотором смысле более сведущими, чем разработчики, использующие другие языки, так как декларативное описание задач может быть весьма неточным. Анджело Колокури из фирмы ESI говорит: "Пролог требует от программиста умения четкой постановки задачи". Фернандо Перейра соглашается: "Если вы знакомы с программированием на Прологе, то вы работаете на очень высоком уровне выражения отношений".

"Я не думаю, что задачи в искусственном интеллекте должны решаться обязательно на языке А или языке В, — продолжает Перейра. — Мы не испытываем недостатка в языках программирования — языки могут лишь помочь выразить то, что мы знаем, если мы поняли это".

В МИРЕ ПЕРСОНАЛЬНЫХ КОМПЬЮТЕРОВ. ВЫПУСК 2

Составители Б. В. Кузьмин, И. В. Емелин
Ответственная за выпуск О. В. Толкачева
Редактор О. В. Толкачева
Технический редактор Т. Н. Зыкина
Художественный редактор А. С. Широков
Оформление обложки А. М. Ефремова
Корректоры Л. А. Буданцева, Г. Г. Казакова

Адрес редакции:

101000, Москва, ул. Кирова, 40
Издательство "Радио и связь"
Редакция сборника "В мире персональных компьютеров"

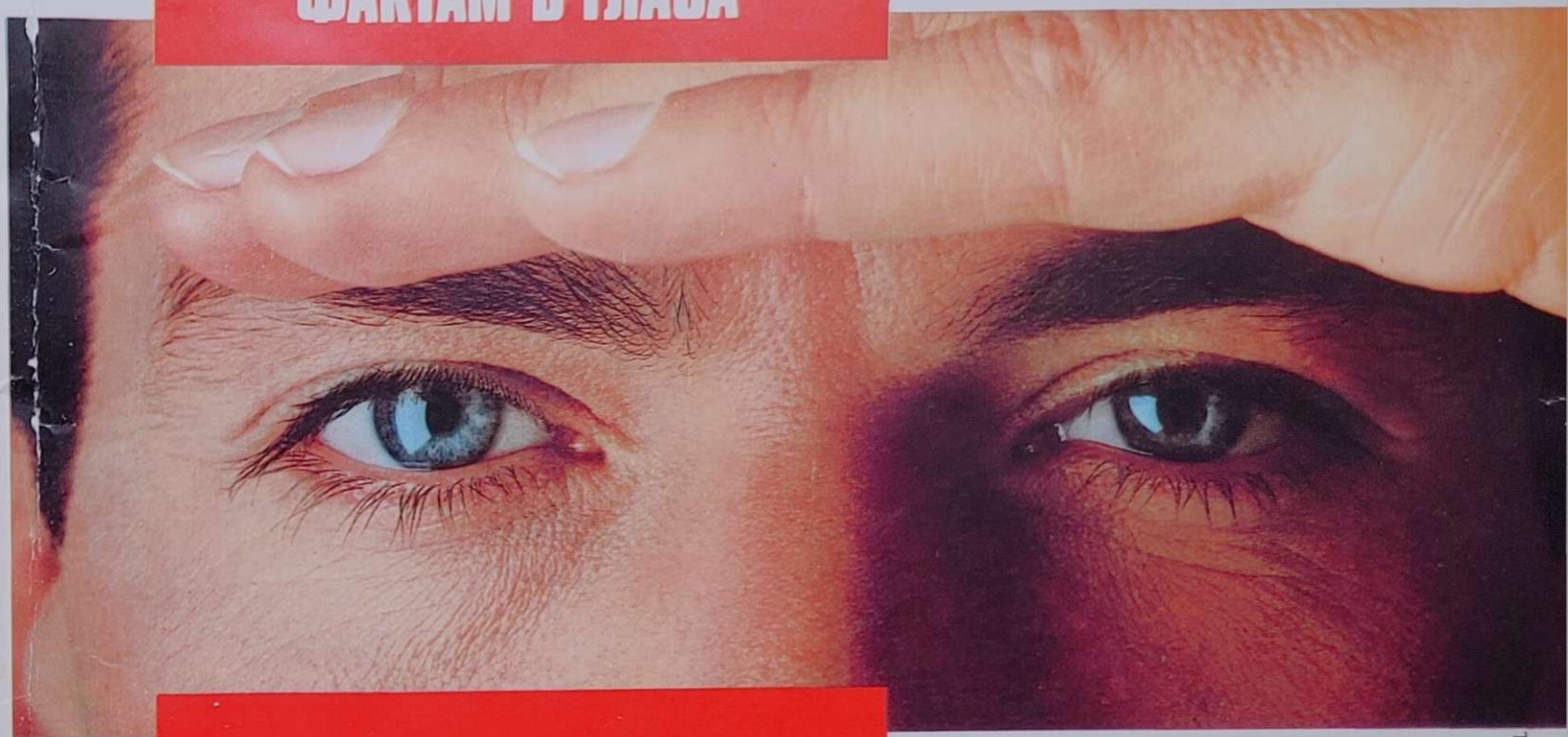
Телекс 411665 Radio SU

Подписано в печать с оригинала-макета 14.12.88. Т-21860. Формат 60x90/8. Бумага писчая № 1. Гарнитура "Пресс-роман". Печать офсетная. Усл. печ. л. 20,0. Усл. кр.-отт. 30,0. Уч.-изд. л. 21,48 (включая 2,38 уч.-изд. л. вкл.). Тираж 50 000 экз. Изд. № 22887. Заказ № 12 Цена 2 р. 50 к.

Издательство "Радио и связь". 101000, Москва, Почтамт, а/я 693

Московская типография № 6 Союзполиграфпрома при Государственном комитете СССР по делам издательств, полиграфии и книжной торговли. 109088, Москва, Ж-88, Южнопортовая ул., 24.

ПОСМОТРИМ ФАКТАМ В ГЛАЗА



CHATEL

КОГДА ВЫ МОЖЕТЕ НАРАЩИВАТЬ КОНФИГУРАЦИЮ СОВМЕСТИМЫХ КОМПЬЮТЕРОВ ОТ 2 ДО 256 РАБОЧИХ МЕСТ, ПЕРЕД ВАМИ РАСПАХНУТЫ ЛЮБЫЕ ГОРИЗОНТЫ.

Знаете ли вы, какая ошибка при выборе вычислительной техники обходится дороже всего? Выбор компьютера с жестко заданной конфигурацией . . .

Ваше предприятие развивается, его деятельность расширяется и потребности растут. В один прекрасный день потребуется наивысшая производительность.

Если вы сразу выбрали систему IN2, значит вы приняли решение, обеспечивающее наибольшую простоту, наивысшее быстродействие и наименьшие затраты.

IN2 — один из немногих компьютеров, позволяющих расширять конфигурацию от 2 до 256 рабочих мест.

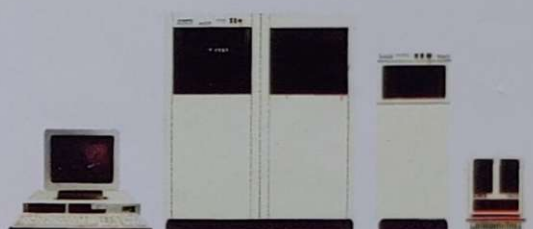
Для получения новой конфигурации компьютера IN2 требуется лишь добавить блоки центрального процессора, расширить объем памяти на диске и установить нужное число рабочих мест с соответствующими периферийными устройствами.

При этом вам не потребуется заменять программу IN-PICK. Поскольку на нее приходится большая часть стоимости программного обеспечения, экономия очевидна. Особенностью модели IN2 является совместимость с микрокомпьютерами семейства LEANORD.

Вы заботитесь о рентабельности, компьютер IN2 отвечает за совместимость. Ваши цели совпадают.

Компьютер IN2 был представлен на стенде фирмы COMEF S. A. на выставке "Наука-88" в Москве с 22 ноября по 1 декабря

В СССР фирму GROUPE INTERTECHNIQUE представляет фирма COMEF S. A.: 117049, Москва, ул. Мытная, д. 1, комн. 24; тел. 230-20-22, 230-20-33, 230-20-44.



IN2

GROUPE INTERTECHNIQUE

57, rue Pierre Curie - B.P. 63 - 78373 Plaisir Cedex
Tél. (1) 34.81.93.00 - Télex : 699302

L'INFORMATIQUE DES REALITES.

Ни один компьютер не удовлетворит всех Ваших потребностей, но одна компания может это

В любой отрасли существует компания, выделяющаяся низкой ценой и высоким качеством производимой продукции. Задача потребителя – как можно быстрее найти такую компанию.

AST – лидер в области вычислительной техники. Репутация этой компании, выпускающей высококачественные, современные, совместимые с изделиями других изготовителей средства вычислительной техники, многие годы безупречна. Сегодня название компании AST появляется на панелях ПК, и мы информируем об этом Вас.

AST Premium 286

AST Premium/386

AST Premium Workstation

Например, ПК AST Premium ™/286. Всего лишь через пять месяцев после начала его поставок мы вышли на третье место по объему продаж ПК, совместимых с IBM PC/AT, оставаясь позади лишь двух компаний – IBM® и Compaq®.

Специалисты единодушно отмечают полную совместимость, расширяемую архитектуру и огромные возможности наших изделий.

А теперь мы предлагаем два новейших ПК семейства AST.

	AST Premium/386	AST Premium/286	AST Premium Workstation
Микропроцессор	80386	80286	80286
Тактовая частота, МГц	20*	10/8/6	10/6
Число тактов ожидания	0–1	0	1
Емкость памяти (стандартно), Мбайт	До 2	1	1
Емкость памяти (расширение), Мбайт	13	13	4
Видеоадаптер	По выбору	EGA/HGC (Для большинства моделей)	Модель типа EEGA/EGA/HGC
Число разъемов дополнительных плат	7**	7***	2
Емкость НГМД, Мбайт	40/90/150/320	20/40/70	40
Тип НГМД	5¼" или 3½"	5¼" или 3½"	5¼" или 3½"

* Три значения тактовой частоты, выбираемые программно.
** Один 32-битовый для расширения памяти; три AT-совместимых 16-битовых типа SMARTslots ™ и один 16-битовый AT-совместимый; два 8-битовых в стандартных моделях.
*** Один 8-битовый, шесть 8/16-битовых, включая два разъема типа FASTslots™ и четыре свободных разъема в стандартных моделях.

Модель AST Premium/386 – первый ПК, обеспечивающий высокую производительность, эквивалентную производительности микрокомпьютеров. В то же время он позволяет не отказываться от уже приобретенных аппаратных и программных средств ПК. А рабочая станция AST Premium Workstation™ является идеальным средством для автоматизации учрежденческой деятельности с огромными возможностями и высокой производительностью при сохранении габаритов ПК.

Все ПК семейства AST Premium имеют высокую производительность и позволяют использовать существующие и будущие операционные системы. Семейство AST Premium – отличное средство для решения любых задач.

Компания AST, представительства которой имеются во многих странах, предлагает семейство современных высоконадежных ПК для решения любых задач.

За более подробной информацией обращайтесь по телефону (714) 756-4879 или телефаксу (714) 756-7657 или пришлите заполненный купон.

AST® RESEARCH INC.

Please send me more information on the following:

☐ Premium/386 ☐ Premium/286 ☐ Premium Workstation

Name _____

Title _____

Company _____

Address _____

City Country _____

Phone _____

AST Research, Inc., 2121 Alton Ave., Irvine, CA 92714-4992, U.S.A.
Attn: International Sales