12•2001

http://radio-mir.com

Индексы: 48925, 45995

радиомир

Ваш компьютер

В НОВЫЙ ВЕК —
С НОВЫМ НАЗВАНИЕМ!

Выбран Министерством ... зования для пос... тысячу школ Р...

Профессор Хигги...
Английский без акцен...

КЛИПАРТ GETTY

ИСКУССТВО & НАРОДЫ

Magic Gooddy 2001

Microsoft ENCARTA deluxe 2002

3cd

ЛУЧШАЯ ЭНЦИКЛОПЕДИЯ ОБО ВСЕМ НА СВЕТЕ!

ПОДПИСКА — 2002!

Для жителей России и стран СНГ (кроме Беларуси):
индексы: 48925, 45995 (годовая),
каталог Агентства "Роспечать", стр. 325.

Для жителей Беларуси:
индексы: 48925, 489252 (ведомств.),
каталог РО "Белпочта" "Газеты и журналы Республики Беларусь", стр. 99.

библиотека Мошкова

LE PETIT ROBERT

Lingvo 7.0

COREL

5CD

Web-Spice Clipart

ВЕБ - мастеру профессионалу

радиомир 2002

Январь

Пн	Вт	Ср	Чт	Пт	Сб	Вс
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31			

Февраль

Пн	Вт	Ср	Чт	Пт	Сб	Вс
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28			

Март

Пн	Вт	Ср	Чт	Пт	Сб	Вс
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

Апрель

Пн	Вт	Ср	Чт	Пт	Сб	Вс
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30					

Май

Пн	Вт	Ср	Чт	Пт	Сб	Вс
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31		

Июнь

Пн	Вт	Ср	Чт	Пт	Сб	Вс
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

Июль

Пн	Вт	Ср	Чт	Пт	Сб	Вс
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

Август

Пн	Вт	Ср	Чт	Пт	Сб	Вс
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	

Сентябрь

Пн	Вт	Ср	Чт	Пт	Сб	Вс
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30						

Октябрь

Пн	Вт	Ср	Чт	Пт	Сб	Вс
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31			

Ноябрь

Пн	Вт	Ср	Чт	Пт	Сб	Вс
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	

Декабрь

Пн	Вт	Ср	Чт	Пт	Сб	Вс
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

Учредитель и издатель:
ООО "НТК ИНФОТЕХ"

Свидетельство о регистрации
ПИ №77-7399 от 19.02.2001

Главный редактор
Ольга СТРЫЖАНКОВА

Зам. гл. редактора
Иван БЕЛЬСКИЙ

Редколлегия:
Сергей ДРОЗДОВСКИЙ,
Виктор ЕРМОЛЕНКО,
Алексей КОНДРАШОВ,
Анатолий КОРБИТ,
Владимир КУЦЕНКО,
Елена ЛЕВИТМАН,
Николай МЕДВЕДЬ,
Геннадий ПЕЧЕНЬ,
Геннадий ШУЛЬГИН

Контактные телефоны:
(095) 105-99-89,
+(37517) 249-41-47

Компьютерная верстка —
Янина БЕЛЬСКАЯ

Техническая графика —
Наталья БЕЛЬСКАЯ,
Александр ГОРАЮТИН,
Мария КАЛАБУХОВА,
Игорь ШЕВКУН

Оформление обложки —
Наталья БЕЛЬСКАЯ,
Надежда БОГОМОЛОВА

Коммерческий отдел
(095) 139-75-77,
+(37517) 221-01-10
E-mail: rm-sales@radio-mir.com

Адреса для писем:
141406, РФ, г.Москва, Химки-6,
ул.Библиотечная, 18-84;

220095, РБ, г.Минск-95, а/я 199

E-mail: rl@radiopage.by
rm@radio-mir.com
WWW: http://radio-mir.com

Наши платежные реквизиты:
получатель : ООО "НТК ИНФОТЕХ",
ИНН 7703155561,
р/с 40702810100022120172
в АКБ "Межтопэнергобанк"
корр. счет 30101810900000000237
БИК 044585237
Адрес банка: 107078, г.Москва,
ул.Садовая-Черногрязская, 6

Матариалы для публикации принимаются в рукописном, печатном и электронном вариантах

Требования к графическим материалам рекламного характера в электронном виде: CorelDRAW 6.0, 7.0 все шрифты в кривых, bitmaps 300 dpi; TIFF, 300 dpi; CMYK в сопровождении печатной копии

За достоверность рекламной и другой публикуемой информации несут ответственность рекламодатели и авторы. Мнение редакции не всегда совпадает с мнениями авторов

Адрес редакции: 141406, г.Москва, Химки-6, ул.Библиотечная, 18-84, тел. (095) 105-99-89

Подписано к печати 2.11.2001 г.
Формат 60 x 84 1/8.
Печать офсетная. 6 печ. л.
Цена свободная

Отпечатано в типографии ЗАО "Красногорская типография"

Заказ 3468. Тираж 1700

радиомир
Ваш компьютер

Ежемесячный массовый журнал.
№12/2001. Издается с сентября 1995 г.

ЧИТАЙТЕ В НОМЕРЕ:

Дорогие друзья!

Начав новый век с нового названия, мы сохранили все старые традиции. Авторы лучших публикаций 2001 г. премируются бесплатной подпиской на 2002г. В список призеров вошли:

М.Долинский (г.Гомель),
Е.Илясов (г.Балашов),
И.Рощин (г.Москва),
С.Рюмик (г.Чернигов).

Желаем и нашим авторам, и нашим читателям творческих успехов и реализации всех задумок!

Н.ЛУТКОВСКАЯ, В.ЛУТКОВСКИЙ,
E-mail: lutkovski@rfe.bsu.unibel.by

ИСТОРИЯ И ГЕОГРАФИЯ КВАНТОВЫХ КОМПЬЮТЕРОВ

Идея использования квантовой теории для построения принципиально нового компьютера имеет свою двадцатилетнюю историю. Сегодня над его созданием работают известные специалисты многих университетов и компьютерных фирм в Америке, Европе, Азии и Австралии.

В 1981 г. Джон Уилер (John A. Wheeler) — физик-теоретик университета Техаса в Аустине, известный своими работами по черным дырам — устроил прием. Все гости были молодыми физиками с общими интересами в области основ теории компьютеров — темы, которая по убеждению Уилера должна была становиться все более и более важной в последующие годы. Обсуждались проблемы, от которых зависело будущее компьютерной индустрии.

В каком направлении будут развиваться микрочипы? Скольких вычислений в секунду можно будет достичь в конечном счете, какое количество тепла при этом будет выделяться, и сможет ли кремний выдержать непрерывный разогрев? За помощью компьютерщики обращались к теории вычислительных систем, разработанной в 1930-х годах пионером в этой области — Аланом Тьюрингом (Alan Turing).

Во время этой дискуссии научного сотрудника Оксфордского университета Дэвида Дойча (David Deutsch) осенила идея. Он понял, что работа машины Тьюринга основана на законах Ньютона, а не на более фундаментальных и универсальных законах квантовой механики.

Имея в виду предельные возможности компьютеров, Дойч сказал: "Я уже вижу, что использование законов квантовой механики даст другой ответ". Он начал работать над статьей, которая сейчас считается классической в этой области. Статья была опубликована в 1985 г. В ней было описано, как компьютер может работать с использованием правил квантовой механики, и почему такой компьютер принципиально отличается от обычных компьютеров [1].

Практически одновременно была опубликована статья известного американского физика-теоретика Ричарда Фейнмана (Richard Feynman) о тесной взаимосвязи квантовой механики и теории информации [2].

Революция, которую начал Дойч, достигла глобального масштаба 15 лет спустя. Квантовые компьютеры сейчас уже считаются не чем-то экзотическим, а могучим будущим компьютерной индустрии. Сегодня ведутся дискуссии уже не о том, станут ли они реальностью когда-нибудь, а о том, как скоро это произойдет. Привлекательность квантовых компьютеров объясняется не их мощью, хотя они несомненно будут мощнее современных. Их главный козырь, можно сказать, убийственная сила, в том, что они могут решить задачи, которые принципиально невозможно решить на обычных компьютерах. Потенциал их таков, что ведущие компьютерные фирмы щедро финансируют программы исследований в этой области. В списке лидеров — IBM, Hewlett-Packard, Lucent Technologies, AT&T и Microsoft. В Нью-Йорке даже возникла компания MagiQ Technologies, которая надеется делать деньги на интвллектуальной собственности в этой области [3].

Одна из главных причин, побуждающих вкладывать деньги в разработку квантовых компьютеров — это страх перед возможностью взлома с помощью этих компьютеров секретных кодов с легкостью, недоступной для других компьютеров. Сигнал тревоги впервые прозвучал в 1994 г., когда Питер Шор (Peter Shor) из Белловских лабораторий фирмы AT&T в Нью-Джерси показал, что квантовые компьютеры справляются с факторизацией чисел (представлением их в виде произведения простых чисел) значительно быстрее, чем обычные компьютеры [4, 5]. Факторизация больших чисел на обычных компьютерах настолько сложна, что она используется программистами для защиты информации. С появлением квантовых компьютеров этот метод устаревает. Как только первый квантовый компьютер средних размеров заработает, правительственные и оборонные ведомства вынуждены будут допустить, что многие из их кодов уже не являются секретными. Понятно, что они приложат все силы для того, чтобы выяснить, на что же способны квантовые компьютеры. Поэтому различные национальные лаборатории приступили к программам основатвльных исследований, в частности, в Американском национальном институте стандартов и технологий в Боулдере (шт.Колорадо), Лос-Аламосской национальной лаборатории в Нью-Мехико, Британском агентстве исследований и оценивания в Малверне.

Ожидается, что информационная емкость квантовых компьютеров достигнет фантастических размеров, причем поиск информации в базах данных с их помощью значительно ускоряется [6].

Квантовый компьютер обещает стать мощным инструментом шпионажа. Над раскрытием тайн квантовой физики сейчас упорно работают ученые, пытаясь понять суть и механизмы управления квантовой информацией. Квантовые компьютеры становятся крошечными лабораториями, в которых ученые могут проверить различные квантово-механические теории со значительно более высокой точностью, чем прежде. В этом направлении наиболее сильные команды работают в Оксфордском и Инсбрукском университетах. Менее многочисленные группы работают в Массачусетском технологическом институте, Калифорнийском технологическом институте, Мичиганском университете, а также в университетах Австралии. Кроме того, многие известные ученые ведут исследования индивидуально в США, Европе и Израиле. Япония стартовала с опозданием, но она прикладывает значительные усилия, чтобы нагнать лидеров.

Трудно найти другое научное направление, которое развивалось бы столь же быстро и привлекало бы столько звезд первой величины. Первый ежегодный международный симпозиум по приложениям квантовой физики (QAS-2001), состоявшийся 1...3 июля 2001 г. в Мичиганском университете, служит ярким примером. Симпозиум открылся докладом Дэвида Дойча, представляющего Центр квантовых вычислений Оксфордского университета. По его мнению, один из главных факторов успешного развития нового направления в ближайшие десятилетия определяется ответом на вопрос: "примут ли те, кто работает в данной области, квантовую теорию всерьез как описание реальности, или нет" [7].

Заметим, что специалисты по цифровой технике твердо знают, что триггер может находиться в одном из двух состояний — нулевом или единичном. Мы располагаем одним битом информации, если знаем — в каком именно. В мире квантовых компьютеров все иначе. Квантовый бит, называемый кью-битом, допускает вероятность того, что он может одновременно находиться в смешанных состояниях. Эта ситуация называется перемешиванием или перепутыванием (entanglement) состояний. Наиболее близкая аналогия — прохождение фотона через два отверстия, так как однозначно указать его траекторию невозможно. Неудивительно, что наиболее быстро новые идеи восприняли специалисты в области квантовой оптики. Они смогли реализовать уникальную возможность манипулирования изолиро-

ванными атомами и фотонами и научились управлять их взаимодействиями на одноквантовом уровне без потери когерентности. Логические вентили квантовых компьютеров можно создавать на различных физических принципах, причем использование изолированных атомов и ионов считается одним из наиболее эффективных способов.

Петер Цоллер (Peter Zoller) из Института теоретической физики университета Инсбрука известен как ведущий специалист в этом направлении. Он выступил с докладом "Реализации квантово-вычислительных и коммуникационных систем на основе квантовых оптических систем". Для создания квантовых процессоров предлагается использовать технику лазерного охлаждения атомов или ионов и управления ими посредством внешних электромагнитных полей. Таким образом, можно реализовать одно- и двух-кьюбитные логические вентили. Физическим механизмом получения смешанного двух-кьюбитного состояния может быть, например, связывание кьюбитов в коллективные моды (фононные или фотонные), контролируемое взаимодействие двух твл, например, сверххолодное столкновение атомов, или использование больших дипольных моментов ридберговых уровней для получения очень сильной связи между нейтральными атомами (так называемый быстрый вентиль).

В последнем случае можно манипулировать мезоскопическими атомными ансамблями вместо отдельных атомов для накопления и управления кьюбитами. Также П.Цоллер выдвинул новые идеи по реализации квантовой коммуникации с использованием генерирования сильно смешанных сжатых спиновых состояний на основе конденсата Бозе—Эйнштейна.

Известный специалист в области квантовой оптики Бахаа Е.А.Салех (Bahaa E.A. Saleh) из Бостонского университета выступил с работой "Квантовое распределенное формирование изображений". Квантовые компьютеры открывают самые широкие возможности для передачи и формирования изображений, принципиально неотличимых от оригинала. Другими словами, виртуальную реальность в "квантовом интернете" будет невозможно отличить от оригинала.

Большие надежды при создании квантовых компьютеров ученые возлагают и на явление сверхпроводимости.

Фил М. Плацмен (Phil Platzman) из Lucent Bell Labs, в докладе "Квантовый компьютинг с электронами, плавающими над жидким гелием" предложил идею аналогового квантового компьютера (АКК), способного решать задачи, недоступные никаким классическим цифровым компьютерам. Главная проблема — в выборе подходящей физической системы, которая допускала бы легкое манипулирование кьюбитами, их легкое считывание и почти полную изоляцию системы от остальной Вселенной. Ф.М.Плацмен предлагает в качестве такой системы использовать ансамбль из $1<N<10^9$ электронов, захваченных над пленкой жидкого гелия при температуре, близкой к абсолютному нулю ($T \leq 10^{-2}$ K). Такие электроны представляют собой почти идеально чистый вакуум, нигде во Вселенной более не встречающийся, когда даже самое слабое взаимодействие с внешним миром контролируется посредством теплового поля на поверхности жидкого сверхтекучего гелия. По мнению Плацмена, принципиально новые процессоры могут быть созданы уже в ближайшее время.

В работе симпозиума принимал участие Брайен Джозефсон (Brian Josephson)— Нобелевский лауреат по физике 1973 г. за открытие эффекта туннельной сверпхроводимости, названного его именем. Его работа называлась "Отношения, перемешивание состояний и PSI". Он считает, что биосистемы формируют отношения селективно, в том числе на смысловой основе. Джозефсон высказал гипотезу, что такие системы учатся управлять некоторыми формами квантового перемешивания состояний, т.е. контролировать, какие связанные состояния могут появиться, и как они будут себя вести. При определенных условиях такие состояния могут действовать как своеобразные информационные центры, с которыми способны взаимодействовать другие биологические агенты. Не исключено, что на этом пути будет получено естественное объяснение многих паранормальных явлений.

С докладами на симпозуме также выступил Генри Стэпп (Henry Stapp) из Национальной лаборатории Лоуренса Беркли Калифорнийского университета и другие выдающиеся ученые.

Мы убеждаемся, что география исследований в этой области приняла глобальный характер и охватила практически все страны. Практически одновременно с симпозиумом QAS-2001, в Минске проходила 17-я международная конференция по когерентной и нелинейной оптике (ICONO-2001), на которой были представлены работы более трехсот ученых. Многие из докладов на этой конференции в той или иной степени были связаны с теорией или экспериментальными исследованиями, проводимыми в области квантовой информатики. Среди них, несомненно, следует упомянуть работы, выполненные в Инсбрукском, Боннском, Бостонском университетах, а также в Московском государственном университете. Лекция о квантовых компьютерах, прочитанная во время этой конференции профессором МГУ В.Задковым, вызвала огромный интерес.

Уровень работ, представленных от Института молекулярной и атомной физики национальной Академии наук Беларуси, вполне соответствовал уровню тех, кто уже добился значительных успехов в этой области. Можно надеяться, что к активным исследованиям в этой многообещающей обпасти в ближайшее время подключатся многие специалисты, и в историю квантовых компьютеров будет вписано немало новых страниц.

Литература

1.Deutsch D. Quantum theory, the Church-Turing principle and the universal quantum computer. — Proceedings of the Royal Society of London, 1985, Vol. A400, p.97-117.

2. Feynman R. Relation between quantum mechanics and information theory. — Optics News, 1985, Vol.11. P.11-20.

3. Mullins J. The Topsy Turvy World of Quantum Computing. — IEEE Spectrum, 2001, № 2. P.42-49.

4. Shor P.W. Algorithm for quantum computation: Descrete logarithm and factoring algorithm. — Proc. of the 25th Anual ASM Symposium on Theory of Computing, ASM Press, New York, 1993, P.11-20.

5. Shor P.W. Polynomial-time algorithm for prime factorization and discrete logarithms on a quantum computer. — SIAM Journal on Computing, 1997, Vol.26, P.1484-1509.

6. Grover L.K. A fast quantum mechanical algorythm for database search. — In: Proceedings of the 28th Annual ACM Symposium on the Theory of Computation, 1996, P.212-219.

7. Quantum Application Simposium 2001: www.erim.org/qas2001. (Перевод С.Санько. "На пороге квантовой эры". — Компьютерные вести, 2001, № 39, С.13).

А.ГРИНЧУК,
г.Минск, РИВШ БГУ.
E-mail: gav@nihe.unibel.by

Работа в системе Mathematica 3.0/4.0.
Решение алгебраических уравнений

Для решения огромного количества типов алгебраических уравнений Mathematica предлагает буквально несколько команд. Однако в этих командах "спрятаны" просто энциклопедические знания о методах решения. Основная и достаточно универсальная команда — **Solve** (дословный перевод — решить). При этом необходимо помнить о том, что в записи уравнений и ответов применяются логические операторы. Например, вместо знака равенства "=" используется знак логического равенства "==". Так, для решения квадратного уравнения необходимо выполнить команду **Solve**[$ax^2 + bx + c == 0, x$], где после запятой указано, относительно какой переменной решается уравнение. Возвращаемый ответ имеет вид:

$$\left\{\left\{x \to (-b+\sqrt{b^2-4ac})\,/\,(2a)\right\}, \left\{x \to (-b-\sqrt{b^2-4ac})\,/\,(2a)\right\}\right\}.$$

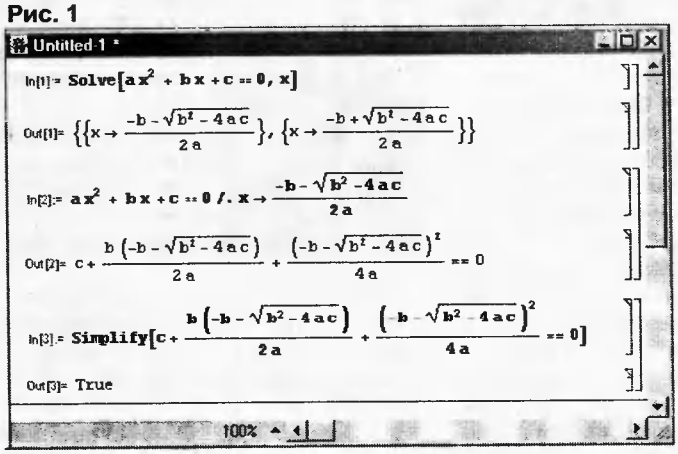
Т.е. в ответе указываются правила подстановки. Переменной x при этом никакое новое значение не присваивается.

Никакая программа не может быть застрахована от ошибок. Решение можно проверить, например, подстановкой ответа в исходное выражение. Сделать это можно разными способами. Во-первых, можно скопировать правило подстановки из ответа и применить его к уравнению:

$$ax^2 + bx + c == 0 /. x \to (-b+\sqrt{b^2-4ac})\,/\,(2a).$$

Возможно, придется применить команды упрощения выражений (Simplify, FullSimplify). В уравнении стоит знак логического равенства, поэтому после выполнения всех манипуляций и в случае правильного решения вы получите ответ **True**, а если что-то не так, то, соответственно — **False** (рис.1).

Рис. 1



То же самое можно проделать и в "автоматическом" режиме. Например, переменной можно присвоить значение "уравнение" — $eqn = ax^2 + bx + c == 0$, аналогичную операцию проделаем и с решением — sol=**Solve**[eqn, x]. Если решений, как в рассматриваемом случае, несколько, то извлекать их из общего списка можно указывая номер в двойных квадратных скобках — sol[[1]]. Подстановка решений в уравнение выглядит следующим образом: eqn/.sol[[1]], eqn/.sol[[2]].

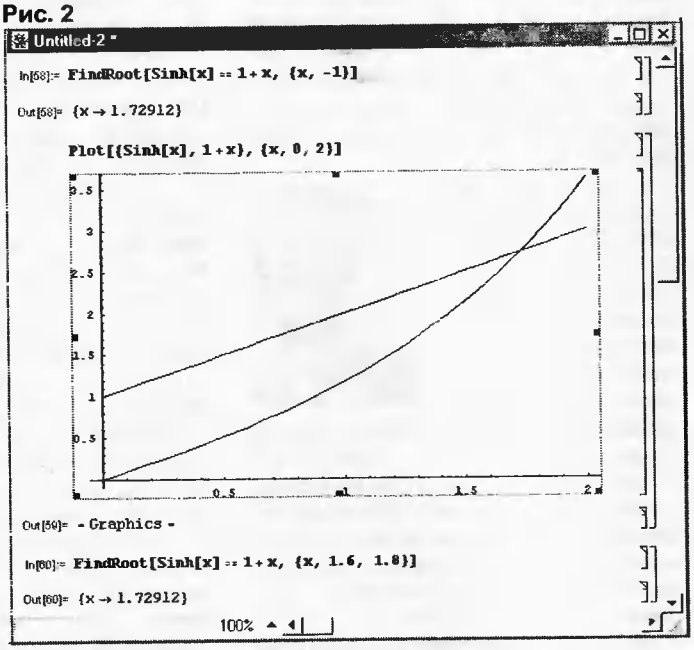
Mathematica справляется с уравнениями третьего и четвертого порядков. Для уравнений высших порядков, как известно, общих формул для решения не существует. Посмотрим, как система обращается с уравнением пятой степени: **Solve**[$x^5 + 5x + 1 == 0, x$], возвращается ответ: {{x → Root[1 + 5 #1 + #1^5 &, 1]}, {x → Root[1 + 5 #1 + #1^5 &, 2]}, {x → Root[1 + 5 #1 + #1^5 &, 3]}, {x → Root[1 + 5 #1 + #1^5 &, 4]}, {x → Root[1 + 5 #1 + #1^5 &, 5]}}.

Выражение "Root[1 + 5 #1 + #1^5 &, 1]" расшифровывается следующим образом — первый корень уравнения $1 + 5x + x^5$. Для Mathematica это такое же точное аналитическое выражение, как и $\sqrt{2}$ — его можно возводить в степень, подставлять в исходное уравнение и т.д. Здесь необходимо сказать несколько слов о нумерации корней решения — сначала в порядке возрастания идут действительные корни, затем — комплексные (располагаются в порядке возрастания их действительных частей). Принципы сортировки корней с одинаковыми действительными частями видны из следующего примера:

{{x → 1 - I}, {x → 1 + I}, {x → 1 - 2I}, {x → 1 + 2I}},

т.е. идет сортировка по возрастанию модуля мнимых частей. Алгоритмы, заложенные в Mathematica, позволяют избежать потери корней полиномов, но, естественно, список выражений вида Root[1 + 5 #1 + #1^5 &, 1] обычно ни о чем не говорит. Здесь на выручку приходит команда преобразования в формат с плавающей точкой. Так, **N[Solve**[$x^5 + 5x + 1 == 0, x$]] приводит к более понятному списку решений {{x→-0.199936}, {x→-1.0045 - 1.06095I}, {x→-1.0045 + 1.06095I}, {x→1.10447 - 1.05983I}, {x→1.10447 + 1.05983I}}.

Рис. 2

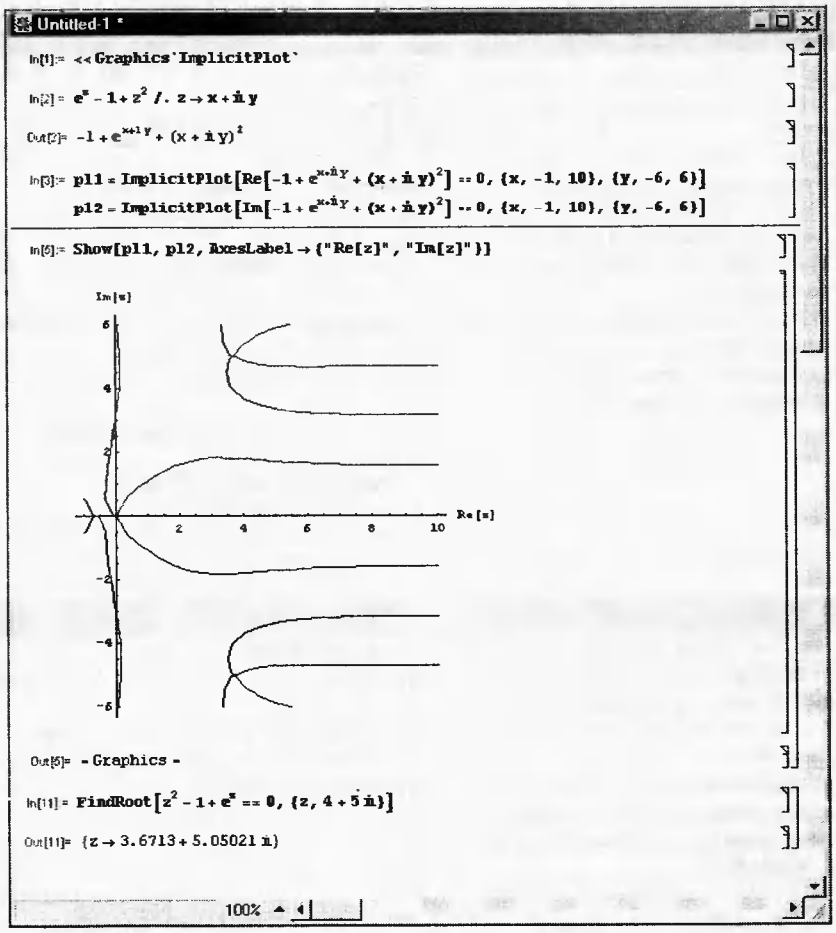


Решение систем уравнений осуществляется при помощи той же команды **Solve**. Только списки уравнений и переменных берутся в фигурные скобки: **Solve[**$\{x^2 + y^2 == 1, x + y == 1\}, \{x, y\}$**]**. Альтернативный вариант задания системы уравнений использует логическое И (&& в обозначениях Mathematica) — **Solve[**$x^2 + y^2 == 1$ && $x + y == 1, \{x, y\}$**]**. Для решения систем предусмотрена дополнительная команда исключения переменных **Eliminate**. Пример ее использования: **Eliminate[**$\{ax^2 + bxy + cy^2 == 0, x + y == 1\}, x$**]**, при этом из системы исключается переменная x, и возвращается уравнение

$$(-a + 2ay - by - ay^2 + by^2 == cy^2).$$

Если же Mathematica не справляется с решением уравнения, то просто возвращается входное выражение. Это служит сигналом для применения численных методов. Одна из команд — **NSolve** — эквивалентна комбинации **N** и **Solve**, она эффективна только при поиске корней полиномов. Значительно чаще используется команда **FindRoot** (найти корень). Здесь можно указать как начальное приближение **FindRoot[***Sinh[x]==1+x, {x, -1}***]**, так и интервал, на котором будет осуществляться поиск решения **FindRoot[***Sinh[x]==1+x, {x, -1, 2}***]**. И начальное значение, и интервал могут быть не только действительными, но и комплексными. Конечно, определяющее значение здесь имеет удачный выбор начального приближения. Удобнее всего решить эту проблему, используя графические средства. Например, можно построить графики правой и левой частей уравнения *Plot[{Sinh[x], 1+x},{x,0,2}]*, а затем найти точку пересечения (рис.2).

Рис. 3



Графический поиск комплексных корней — несколько более сложная процедура. Пожалуй, удобнее всего воспользоваться функцией **ImplicitPlot**, которая предназначена для построения графиков функций, заданных в неявном виде *f(x, y)=0*.

Для примера найдем один из комплексных корней уравнения $Exp[z] - 1 + z^2$. Во-первых, необходимо загрузить дополнительный пакет, содержащий интересующую нас команду: <<Graphics`ImplicitPlot`. Далее перейдем к комплексной переменной: $Exp[z] - 1 + z^2/.z \to x + iy$. Выражение $Exp[z] - 1 + z^2$ обращается в ноль, когда равны нулю его действительная (**ImplicitPlot[***$Re[Exp[x+iy] - 1 + (x+iy)^2] == 0, \{x, -1, 10\}, \{y, -6, 6\}$***]**) и мнимая (**ImplicitPlot[***$Im[Exp[x+iy] - 1 + (x+iy)^2] == 0, \{x, -1, 10\}, \{y, -6, 6\}$***]**) части. Точки пересечения и определяют графически комплексные корни, после чего можно воспользоваться командой **FindRoot** (рис.3).

Последний пример — еще одна иллюстрация того, что эффективность Mathematica определяется не только набором входящих в нее команд, но и высокой степенью их интеграции, графические и математические возможности дополняют и усиливают друг друга. В заключение приведем таблицу основных команд, относящихся к работе с графикой и решению уравнений, а в следующей статье ознакомимся с методами решения дифференциальных уравнений.

Действие	Реализация
Двумерная графика	
1) Построение графика функции	**Plot[**Sin[x], {x, 0, 2*Pi}**]**
2) Построение графика функции с обрамлением	**Plot[**Sin[x], {x, 0, 2*Pi}, Frame → True**]**
3) Построение графика функции с подписями к осям	**Plot[**Sin[x], {x, 0, 2*Pi}, **AxesLabe** → {"x value", "Sin[x]"}**]**
4) Построение графика функции с обрамлением, координатной сеткой, заданием отношения высоты к ширине графика, цвета графика	**Plot[**Sin[x], {x, 0, 2*Pi}, **Frame** → True, **GridLines** → Automatic, **AspectRatio** → 1/GoldenRatio, **PlotStyle** → {RGBColor[1, 0, 0]}**]**
5) Построение графиков нескольких функций	**Plot[**{Sin[x], Sin[2*x], Sin[3*x]}, {x, 0, 2*Pi}**]** или **Plot[**{Sin[x], Sin[2*x], Sin[3*x]}, {x, 0, 2*Pi}, **PlotStyle** → {{RGBColor[1, 0, 0]}, {RGBColor[0, 1, 0]}, {RGBColor[0, 0, 1]}}**]** — с указанием опций
6) Вызов списка использованных опций	**Plot[**Sin[x], {x, 0, 2*Pi}**]** Options[%]
7) Построение графика функции, заданной параметрически	**ParametricPlot[**{Cos[5*t], Sin[3*t]}, {t, 0, 2*Pi}, **AspectRatio** → Automatic**]**
8) Построение графика функции в полярных координатах	Вызвать пакет расширений **<<Graphics`Graphics`** **PolarPlot[**Sin[t]^2, {t, -(Pi/2), (3*Pi)/2}**]**

Действие	Реализация
Трехмерная графика	
1) Построение трехмерного графика	**Plot3D[**Sin[x y], {x, 0, 4}, {y, 0, 4}**]**
2) Вызов списка использованных опций	**Options[%]**
3) Построение трехмерного графика с дополнительными опциями: число точек 40*40, без ячеек, с лицевыми линиями, с подписями к осям	**Plot3D[**Sin[x*y], {x, 0, 4}, {y, 0, 4}, **PlotPoints** → 40, **Mesh** → False, **FaceGrids** → All, **AxesLabel** → {"Length", "Width", "Height"}**]**
4) Построение графика кривой, заданной параметрически	**ParametricPlot3D[**{Cos[5*t], Sin[3*t], Sin[t]}, {t, 0, 2*Pi}**]**
5) Построение поверхности, заданной параметрически	**ParametricPlot3D[**{Cos[s]*Sin[t], Sin[s]*Sin[t], Cos[t]}, {t, 0, Pi}, {s, 0, 2*Pi}**]**
6) Изменение точки обзора для трехмерного графика	1. Набрать **Plot3D[Sin[**x + y], {x, 0, Pi}, {y, 0, Pi}, **]**. 2. Установить курсор перед закрывающей квадратной скобкой. 3. **Input / 3D ViewPointSelector**. 4. Используя линейки прокрутки (справа и внизу), изменить точку обзора. 5. Нажать **Paste** в диалоговом окне. 6. Выполнить команду рисования
Анимация	
1) Вызов пакета анимационной графики	**<<Graphics`Animation`**
2) Формирование кадров для анимации двумерного графика с изменением параметра анимации от 1 до 6 с шагом 1	**Animate[**Plot[Sin[n*x], {x, 0, 2*Pi}, Axes → False], **{**n, 1, 6, 1**}]** **Animate[**график, {параметр анимации, нач._значение, кон._значение, шаг}**]**
3) Формирование кадров для анимации трехмерного графика с изменением параметра анимации от 0 до π с шагом π/10	**Animate[**Plot3D[Sin[t - (x + y)], {x, 0, 2*Pi}, {y, 0, 2*Pi}], **{**t, 0, 2*Pi, Pi/10**}]**
4) Просмотр анимационного графика	Двойной щелчок по любому из кадров или выделить группу группу ячеек с анимацией, **Ctri+Y**
Численное и аналитическое решение уравнений	
1) Аналитическое решение уравнения	**Solve[**x^3 + q/3*x + r/27 == 0, x**]** — логическое равенство, в отличие от знака присваивания, задается двойным знаком равенства "=="
2) Получение численного значения корней уравнения (в форме с плавающей запятой)	1. Переменной s присвоить значение "решение уравнения": s = **Solve[**x^5 - 1331*x + 11 == 0, x**]** 2. Получить численное значение s: **N[**s**]**
3) Численное решение уравнения	**NSolve[**3 + 3*x - 7*x^2 - x^3 + 2*x^4 + 3*x^7 - 3*x^8 - x^9 + x^10 == 0, x**]**
4) Найти приближенное численное решение уравнения	**FindRoot[**x*Sin[x] - 1/2 == 0, {x, 1}**]** — решается относительно x с начальным приближением x_0=1
Аналитическое решение систем уравнений	
1) Аналитическое решение системы уравнений	**Solve[{**x^2 + y^3 == x*y, x + y + x*y == **1}, {**x, y**}]**
2) Исключение переменных	**Eliminate[{**a*x^2 + b*x*y + c*y^2 == 0, x + y = = **1}**, x**]** исключение переменной **x** из системы; **Solve[%**, y**]** — решение полученного уравнения
Численное решение систем уравнений	
1) Численное решение системы	**NSolve[{**x^2 + y^2 + z^2 == 1, x + y + z == 0, x-y+z==1**}, {**x, y, z**}]** или **FindRoot[{**x^2 + y^2 == 1, y == x*Exp[x]**}, {**x, 0.1**}, {**y, 0.5**}]** — с заданными начальными приближениями

А.БЕЛОУСОВ,
E-mail: ark@mos.ru
WWW: http://hi-tech.nsys.by

Последние годы характерны невероятным ростом объемов и скорости жестких дисков. Из-за постоянно снижающихся цен на диски и растущих при этом потребностей программного обеспечения, целый класс машин от 486-х до Pentium может приобрести "вторую молодость" при замене жестких дисков на более современные. Однако замене дисков могут помешать ограничения на объемы HDD, присущие программному и аппаратному обеспечению компьютеров, не рассчитанному на такие скорости роста. Данная статья раскрывает проблемы, возникающие при замене IDE- и Enhanced IDE-дисков на компьютерах с BIOS, датированных 1992...1998 г.г.

Интерфейс Int 13h

При проектировании PC, специалисты IBM разместили интерфейс к дискам в системном BIOS. Когда приложение желает прочитать или записать данные на диск, оно вызывает DOS. DOS, зная структуру диска и место, где расположен требуемый файл, вычисляет CHS-адрес (Cylinder/Head/Sector — трек/поверхность/сектор) и вызывает BIOS через Int 13h. BIOS исполняет запросы на чтение и запись по заданному CHS-адресу, обращаясь к контроллеру диска через порты ввода/вывода (I/O — input/output). Затем результат передается DOS, который "отдает" его обратно приложению. Эта схема делает диски в DOS аппаратно независимыми, оставляя аппаратную специфику на откуп BIOS.

Традиционный интерфейс BIOS Int 13h имеет следующие ограничения: 1024 трека, 256 поверхностей и 63 сектора на треке (традиционно счет секторов начинается с единицы, а не с нуля, поэтому секторов 63, а не 64). При 512 байтах в секторе это в сумме дает до 8064 Мб (почти 8 Гб)!

Обратите внимание, мегабайт (Мб) — это 1048576 (т.е. 1024x1024), а не 1000000 байтов. Значение 1000000 для обозначения мегабайта обычно любят использовать производители дисков, поскольку это дает несколько большие значения объема. Но это только "замутняет" картину, поэтому будьте аккуратны с их заявлениями.

Бег с препятствиями — барьеры на пути увеличения объема дисков

Граница в 504 Мб

Первая известная граница объема жесткого диска — 504 Мб (528482304 байта) — обусловлена тем, что спецификации ATA (IDE) имеют ограничения, отличные от ограничений BIOS. Когда традиционный интерфейс BIOS Int 13h используется для управления IDE-дисками, ограничения комбинируются следующим образом (табл.1).

Табл. 1

Максимум	BIOS	IDE	Общее ограничение
Секторов/трек	63	255	63
Поверхностей	256	16	16
Треков	1024	65536	1024
Объем	8064 Мб	127, Гб	504 Мб

Для преодоления барьера в 504 Мб требуется реализовать одно из следующих решений:

- использовать транслирующий (Enhanced) BIOS;
- применить карту с BIOS, который перенаправляет на себя интерфейс Int 13h;
- использовать программное обеспечение типа Disk Manager от Ontrack или EZ-Drive от StorageSoft (ранее Micro House).

Начнем с последнего. Программное обеспечение требует для себя драгоценной оперативной памяти, а также поддержки со стороны операционной системы для размещения в памяти. Решение с дополнительной картой требует только выделения "окна" в области адресов от A000h до EFFFh, во всем остальном оставаясь идентичным решению с транслирующим BIOS. Наконец, решение с транслирующим BIOS является идеальным решением, не требующим более никаких усилий по преодолению барьера в 504 Мб или изменений в аппаратуре или программном обеспечении, использующем BIOS.

Транслирующий BIOS

Традиционный интерфейс BIOS Int 13h передает CHS-адрес непосредственно в контроллер диска, создавая таким образом барьер в 504 Мб в случае IDE-дисков. Транслирующий же BIOS может переводить CHS-адрес либо в другую геометрию, либо в LBA (Logical Block Address — логический адрес блока). Соответственно, используемый при вызове BIOS CHS-адрес теперь называется L-CHS (логический CHS), а используемый BIOS для управления устройством CHS-адрес называется P-CHS (физический CHS).

Первый метод прямо переводит L-CHS в P-CHS. Согласно спецификациям ATA-2, IDE-устройства объемом до 8 Гб подчиняются ограничению в 63 сектора на трек. На устройствах объемом больше 504 Мб с максимум 16 поверхностями (ограничение IDE) только количество треков будет больше 1024 (ограничение BIOS). Это дает простую схему трансляции CHS, в которой количество треков делится на 2, 4, 8 или 16, а количество поверхностей умножается на тот же коэффициент. При этом количество секторов на трек остается неизменным. Максимальный объем будет зависеть от количества секторов на трек и равен ограничению BIOS в 8064 Мб, если устройство имеет 63 сектора на трек (табл.2).

Табл. 2

Реальные		Измененные		Максимальный объем (Мб)
Cylinders	Heads	Cylinders	Heads	
1<C<1024	1<H<16	C=C	H=H	504
1024<C<2048	1<H<16	C=C/2	H=H*2	1008
2048<C<4096	1<H<16	C=C/4	H=H*4	2016
4096<C<8192	1<H<16	C=C/8	H=H*8	4032
8192<C<16384	1<H<16	C=C/16	H=H*16	8064

Пример. Устройство объемом 2014 Мб имеет геометрию CHS 4092x16x63, которая будет транслирована в геометрию 1023x64x63, приемлемую для интерфейса Int 13h. Другими словами, DOS (или любая другая ОС, использующая интерфейс BIOS Int 13h для тех же целей) "думает", что имеет дело с устройством, имеющим 1023 трека, 64 поверхности и 63 сектора на трек, в то время как BIOS управляет устройством с первоначальной геометрией.

Устройство, которое имеет 16 поверхностей и более 8192 треков (т.е. объемом больше 4 Гб) будет транслировано в геометрию с 256 поверхностями. Это является проблемой, поскольку DOS не может работать с 256 поверхностями, но она имеет простое решение (см. раздел "Граница в 4 Гб").

Поскольку коэффициент трансляции равен степени двойки, BIOS может выполнять трансляцию простым битовым сдвигом адресов треков и поверхностей на одну и более позиций вправо и влево соответственно. Поэтому эта трансляция также известна как bit-shifting translation.

Второй метод трансляции используется при работе с устройствами, допускающими обращение с LBA. При LBA все сектора последовательно пронумерованы, и P-CHS более не используется — BIOS посылает в устройство вместо P-CHS номер сектора (читай — LBA). Однако DOS продолжает работать с CHS-адресами, поэтому BIOS вычисляет искусственный L-CHS и предъявляет эту геометрию DOS. L-CHS вычисляется из общего объема (табл.3).

Из табл.3 видно, что количество секторов на трек всегда равно 63, а количество поверхностей равно 16 или кратно этому значению. Соответственно, количество треков вычисляется делением общего объема в байтах на количество байтов в треке. Этот метод трансляции называется LBA-assisted.

Табл. 3

Объем	Sectors	Heads	Cylinders
1 байт<X< 504 Мб	63	16	X/(512*63*16)
504 Мб<X<1008 Мб	63	32	X/(512*63*32)
1008 Мб<X<2016 Мб	63	64	X/(512*63*64)
2016 Мб<X<4032 Мб	63	128	X/(512*63*128)
4032 Мб<X<8032,5 Мб	63	255*	X/(512*63*255)

Последняя строка табл.3 показывает, что устройство объемом более 4 Гб

должно быть транслировано в геометрию с 255 поверхностями вместо 256. Это позволяет избежать проблем, когда DOS не может работать с 256 поверхностями (см. раздел "Граница в 4 Гб").

Описанные методы трансляции дают одинаковую L-CHS-геометрию, за исключением тех случаев, когда устройство имеет меньше 63 секторов на трек. Метод LBA всегда назначает 63 сектора на трек и, в отличие от метода bit-shifting, не ставит ограничений на сообщаемую геометрию устройства (P-CHS). Однако bit-shifting остается единственным методом, если устройство не может работать с LBA, а большинство транслирующих BIOS требуют указания значений P-CHS в настройках BIOS.

При любом методе трансляции BIOS обеспечивает доступ к секторам в том же порядке, как и при "нетранслированном" доступе. Однако так может быть не всегда, поэтому ОПАСНО менять метод трансляции для уже форматированного устройства!

Является ли BIOS транслирующим?

Теперь, когда теоретические основы рассмотрены, самое время ответить на вопрос из заголовка. Вначале следует проверить дату создания BIOS — если она от 7/94 или более поздняя, то шансы, что BIOS транслирующий, довольно высоки. Во-вторых, можно проверить BIOS SETUP на наличие типа устройства User defined и выбора режима трансляции. Режимы LARGE или ECHS обычно указывают на метод bit-shifting, режим LBA указывает, что BIOS поддерживает метод LBA-assisted.

Но чтобы выяснить точно, необходимо исследовать HDPT (Hard Disk Pararneter Table — таблица параметров жесткого диска). BIOS содержит эту 16-байтовую таблицу для каждого физического диска. Если BIOS поддерживает трансляцию, то используется разновидность этой таблицы EDPT (Enhanced Disk Parameter Table). Данные из HDPT/EDPT доступны через Int 13h/8h и Int 13h/48h, а для совместимости со старым программным обеспечением указатели на HDPT для двух первых устройств сохраняются в векторах прерывания 41h и 46h. В табл.4 показано, что хранится в HDPT и EDPT.

Если для некоторого устройства BIOS построил EDPT, то это транслирующий BIOS! Просмотреть EDPT можно с помощью утилиты WDTBLCHK от Western Digital, доступной в виде самораспаковывающегося архива CHKBIOS.EXE по адресам ftp://ftp.wdc.com/drivers/hdutil/chkbios.exe (56 Кб) или http://web.inter.nl.net/hcc/J.Steunebrink/chkbios.exe

Однако здесь есть проблема — некоторые транслирующие BIOS строят EDPT вместо HDPT только если устройство доступно в транслирующем режиме. В этом случае можно использовать "трюк" с указанием фиктивного устройства в настройках BIOS и игнорированием сообщения об отсутствии соответствующего устройства. После этого можно утилитой WDTBLCHK проверять корректность трансляции. Если это работает, то данную процедуру можно повторить с различными значениями CHS, чтобы проверить граничные значения транслирующих алгоритмов. Можно указать 2047x16x63/2048x16x63 для 1 Гб, 4095x16x63/4096x16x63 для 2 Гб, 8191x16x63/8192x16x63 для 4 Гб и 16320x16x63 для 8 Гб (последнее значение должно быть транслировано в 1024x255x63 в режиме LBA).

Чтобы обойти и это ограничение, многие производители дисков поставляют со своими дисками программное обеспечение типа Disk Manager или EZ-Drive. Однако лучше произвести апгрейд BIOS.

Для рассматриваемых BIOS "сценарий" проблемы может быть следующим:

- BIOS может видеть не более 4095 треков, усекая из-за этого доступный объем до 2015 Мб;
- BIOS использует только младшие 12 бит из 16-битного значения треков, из-за чего теряются 2015 Гб и больше от объема диска, если диск больше 2 Гб!
- BIOS вызывает зависание системы при загрузке, если в настройках BIOS указан счетчик треков более 4095.

Для первых двух случаев мне неизвестен надежный метод выявления ограничения в 2 Гб помимо подключения устройства с более чем 4095 треками. Но проблему также может выявить "трюк" с определением транслирующего BIOS, описанный в предыдущем разделе. BIOS, датированные маем 1996 г. или более поздние, должны быть свободны от этой проблемы и должны поддерживать трансляцию до 8 Гб.

Табл. 4

Offset	Type	HDPT	EDPT
0	Word	Physical Cylinders	Logical Cylinders, limit 1024
2	Byte	Physical Heads	Logical Heads, limit 256
3	Byte	Reserved	A0h Signature, indicating EDPT
4	Byte	Reserved	Physical Sectors/Track
5	Word	Precompensation (Obsolete)	Precompensation (Obsolete)
7	Byte	Reserved	Reserved
8	Byte	Drive Control Byte	Drive Control Byte
9	Word	Reserved	Physical Cylinders, limit 65536
11	Byte	Reserved	Physical Heads, limit 16
12	Word	Landing Zone (Obsolete)	Landing Zone (Obsolete)
14	Byte	Physical Sectors/Track	Logical Sectors/Track, limit 63
15	Byte	Reserved	Checksurn

Наконец, если BIOS поддерживает расширение Int 13h от IBM/Microsoft, то этот BIOS автоматически поддерживает и трансляцию. Такие BIOS появились в 1995 г. (см. раздел "Расширение Int 13h").

Граница в 2 Гб

Все основные производители жестких дисков сообщили об ограничении BIOS в 2 Гб. Оказалось, что некоторые выпущенные до мая 1996 г. транслирующие BIOS имеют проблемы с трансляцией устройств, имеющих более 4095 треков, что ограничивает допустимый объем величиной 2015 Мб (около 2 Гб).

К ограничению в 2 Гб относится и ограничение показываемого размера диска в Award BIOS. Все Award BIOS, датированные июлем 1994 г. и позднее, корректно поддерживают LBA-трансляцию до 8 Гб. Однако в BIOS, датированных до января 1996 г., имеется ошибка — в настройках BIOS и при загрузке, для дисков размером 2016, 4032 и 6048 Мб, показываемый объем стартует с нуля. Это похоже на описанное ограничение в 2 Гб, но в действительности —

это ошибка в процедуре вычисления объема диска, не влияющая на поддержку BIOS больших дисков. Для таких Award v4.50(P)(G) BIOS просто следует использовать HDD AUTO DETECTION, выбирать пункт с LBA и пренебрегать неправильным показом объема диска.

Граница в 4 Гб

Да, имеется и еще одно ограничение! На самом деле это проблема операционных систем, но подходящий путь обхода этой проблемы — учесть ее в системном BIOS. Дело в том, что в DOS и в Windows 95/98 есть ограничение в 255 поверхностей на устройство, и транслированная геометрия в 256 поверхностей будет соответственно создавать проблемы. Это происходит, когда устройство имеет 16 поверхностей и более чем 8192 треков (объем больше 4032 Мб).

Как было показано в разделе "Транслирующий BIOS", для устройств объемом больше 4032 Мб BIOS должен при методе LBA транслировать геометрию в 255 поверхностей вместо 256. Если же BIOS этого не делает, можно использовать метод bit-shifting (ECHS или LARGE в настройках BIOS) со следующим "трюком":

- указать значения CHS-устройства в настройках CMOS или выполнить HDD AUTO DETECTION и пока не выбирать режим трансляции;
- уменьшить число поверхностей с 16 до 15;
- умножить количество треков на 16, разделить на 15 и отбросить дробную часть;
- заменить количество треков на вычисленное большее значение;
- выбрать режим bit-shifting (ECHS, LARGE);
- сохранить настройки CMOS, разбить диск на разделы и отформатировать его.

С этим "трюком" будет использоваться транслированная геометрия с 15*16=240 поверхностями. Это ограничивает максимальный L-CHS форматом 1024x240x63, что эквивалентно объему 7560 Мб.

Расширение Int 13h — преодолевая барьер в 8 Гб

При той скорости, с которой развиваются технологии жестких дисков, встала проблема с IDE-устройствами, выходящими за ограничение интерфейса Int 13h в 8 Гб. Единственным способом преодолеть это ограничение является отход от CHS-адресации в пользу прямого LBA-интерфейса (SCSI-устройства используют разновидность LBA-интерфейса уже много лет).

LBA работает с 64-битным адресом сектора, так что (теоретически) доступен потрясающий объем в 8796093022208 Гб! Однако из-за свойств интерфейса ATA только младшие 28 битов адреса могут быть использованы в IDE-устройствах, что дает ограничение в 127 Гб. Это ограничение пока не преодолено для IDE-устройств, хотя уже доступны одиночные устройства со 180 Гб "на борту".

Windows 95/98 готовы к преодолению барьера в 8 Гб, поскольку уже используют LBA внутри. Исключая "Безопасный режим" или "Режим совместимости", дисковый драйвер защищенного режима обходит интерфейс BIOS Int 13h и может обращаться к диску непосредственно с использованием LBA.

Чтобы сохранить совместимость с операционными системами, которые все еще используют CHS-адресацию при вызове BIOS (включая Windows 95/98 во время загрузки), требуется расширение интерфейса BIOS Int 13h (как и доработка под это расширение собственно программного обеспечения). И здесь на сцену выходит расширение Int 13h от IBM/Microsoft. Эта спецификация придает интерфейсу Int 13h новую функциональность, принципиально отличающуюся от традиционного интерфейса, и позволяет вызывать BIOS непосредственно с указанием LBA.

BIOS с расширением Int 13h от IBM/Microsoft поддерживают устройства как без LBA, так и с LBA, предлагая разные методы трансляции: прямую CHS-адресацию (нормальный режим); трансляцию L-CHS в P-CHS; трансляцию L-CHS в LBA; прямую LBA-адресацию; трансляцию LBA в P-CHS. BIOS с расширением также поддерживают до 4 или 8 устройств, включая устройства со сменными носителями, и их можно найти в большинстве современных компьютеров. BIOS, датированные январем 1998 г. или позднее, обычно уже поддерживают расширение Int 13h. Чтобы определить наличие расширения BIOS, можно использовать утилиту EXTBIOS:

http://web.inter.nl.net/hcc/J.Steunebrink/extbio13.zip (6 Кб).

Как и в случае с барьером в 504 Мб, требуется реализовать одно из следующих решений для преодоления барьера в 8 Гб при отсутствии поддержки расширения Int 13h:

- провести апгрейд BIOS;
- применить карту с BIOS, поддерживающим расширение Int 13h и перенаправляющим на себя интерфейс Int 13h;
- использовать программное обеспечение типа Disk Manager или EZ-Drive.

Теперь, если имеется расширение Int 13h, что еще требуется для преодоления барьера в 8 Гб? Для этого необходима ОС, умеющая адресовать устройство с LBA — либо непосредственно в защищенном режиме, либо через расширенный интерфейс BIOS Int 13h. Windows 95 (все версии), Windows 98, Windows NT 4.0 (с SP4), Linux, OS/2 Warp 3 и 4 (на HPFS) готовы к этому. Однако MS-DOS 5.0 и 6.x, Windows 3.x и Windows NT 3.x по-прежнему ограничены 8 Гб.

Последний, но не менее важный момент — требуется утилита разбиения диска на разделы, умеющая создавать разделы выше границы 8 Гб и с новыми типами 0Eh и 0Fh (FAT16) или 0Bh и 0Ch (FAT32) и автоматически выбирающая новый тип раздела вместо старого при наличии расширения Int 13h.

Стоит ли говорить, что новые типы разделов невидимы для старых версий DOS и что совместимость была принесена в жертву во имя прогресса.

MS-DOS, Windows и диски

Помимо ограничений аппаратуры и системного программного обеспечения (BIOS), операционные системы (ОС) также устанавливают свои ограничения на объем дисков. Об ограничении на 255 поверхностей было сказано выше. Посмотрим, какие же еще ограничения в работе с дисками идут от ОС фирмы Microsoft — ведь именно они во многом определили нынешнее положение дел.

Поддержка жестких дисков в MS-DOS началась с версии 2.0 с введением понятия раздела. Единственный раздел на диске с файловой системой FAT12 имел тип 01. FAT12 использу-

ется до сих пор (например, для дискет). В MS-DOS 3.0 ввели новый тип раздела — 04 — с файловой системой FAT16, позволяющей иметь больше файлов (65525 вместо 4085) с кластерами меньшего размера и потому с более эффективным использованием дискового пространства. Однако размер этого раздела остался ограничен 32 Мб (из-за того что Int 25/26 в MS-DOS могли адресовать только 65535 секторов по 512 байтов).

В MS-DOS 3.3 появилась возможность иметь на диске более одного раздела за счет введения нового типа незагружаемого раздела 05 (Extended), который содержит в себе другие разделы. Таким образом, на диске могут находиться один первичный (primary) раздел и один дополнительный (extended), включающий до 23 логических (logical) разделов, доступных через буквы от C до Z. Однако стратегия назначения букв получилась "корявой" — сначала буква назначается первичному разделу на первом диске, затем первичному разделу на втором, и только потом буквы назначаются логическим разделам. Неудобство здесь в том, что при временной установке второго диска с первичным разделом все логические разделы с первого диска сдвигаются на одну букву.

В MS-DOS 4.0 ввели тип раздела 06. Это тоже раздел с FAT16, но ограничения Int 25/26 сняты, и он может иметь размер до 2047 Мб (ограничение FAT16, исходящее из того, что в разделе может быть только 65525 кластеров по 32 Кб). Windows NT поддерживает разделы FAT16 с кластерами по 64 Кб, но DOS и Windows не работают с таким размером кластера.

В MS-DOS 5.0 была введена поддержка до 8 устройств (вместо 2) и более одного первичного раздела на диске. Но стратегия назначения букв разделам осталась "корявой" — теперь буквы сначала назначаются первым первичным разделам на всех дисках, затем логическим разделам на всех дисках, и наконец — прочим первичным разделам на всех дисках.

В Windows 95 для поддержки дисков объемом больше 8 Гб были введены новые типы разделов — 0Eh и 0Fh. Эти типы ничем не отличаются от 06 и 05 (так же как и 06 не отличается от 04), но предполагают для своего использования обращение к расширению интерфейса BIOS Int 13h, как об этом говорилось выше.

Наконец, Windows 95 OSR2 и Windows 98 поддерживают еще два типа разделов — (0Bh и 0Ch) с файловой системой FAT32 и размером до 2048 Гб (ограничение, вызванное 32-битными счетчиками секторов в таблице разделов). Достоинства FAT32 в сравнении с FAT16 — те же, что и FAT16 в сравнении с FAT12, т.е. больший размер раздела и большее количество файлов при меньшем размере кластера.

Итак, следующий барьер, который потребует переработки программного обеспечения — это диски объемом больше 2 Тб (2048 Гб). Для этих дисков необходимо будет заново спроектировать схему разбиения на разделы и перейти к 64-битным счетчикам секторов (либо считать объекты крупнее секторов).

Терминология

IDE (Integrated Drive Electronics) — это реализация дисковых устройств с интегрированным контроллером, что позволяет уменьшить стоимость дисков и облегчить разработку их микропрограмм. В 1994 г. был принят стандарт ATA (AT Attachment) для IDE-устройств. В 1996 г. стандарт был пересмотрен, и появился стандарт ATA-2, затем ATA-3 и Ultra-ATA. В ATA-2 ввели поддержку более быстрых режимов PIO3/PIO4 и режимов прямого доступа к памяти — multiword DMA1/DMA2. Также в ATA-2 появилась поддержка LBA для линейной адресации и второго канала — для подключения до 4 устройств.

Недостаток стандарта ATA в том, что он разработан для дисков, поэтому в 1994 г. был разработан стандарт ATAPI (ATA Packet Interface) на подключение к IDE-портам устройств типа CD-ROM и ленточных накопителей.

EIDE (Enhanced IDE) — это предложенная Western Digital марка устройств, разработанных на базе ATA-2 и Ultra-ATA. В пику этому Seagate с Quantum предложили марку Fast-ATA, которая, по сути, предлагает то же самое.

В последнее время активно внедряются стандарты Ultra-ATA/33, Ultra-ATA/66 и Ultra-ATA/100, реализующие DMA-режимы со скоростями 33, 66 и 100 Мб/с соответственно. Более того, разработан стандарт SerialATA, первые версии которого предлагают пропускную способность до 1,5 Гб/с, сохраняя при этом совместимость со стандартами ATA на уровне портов ввода/вывода!

Дополнительная информация в Интернет

Вариант этой статьи на английском можно найти по адресу: http://web.inter.nl.net/hcc/J.Steunebrink/bioslim.htm

Обширным источником информации является "The Enhanced IDE/Fast-ATA/ATA-2 FAQ" от John Wehman и Peter den Haan. FAQ можно найти на странице Peter den Haan об IDE-устройствах (http://thef-nym.sci.kun.nl/~pieterh/storage.html). Здесь также можно найти ссылки на другие сайты.

На странице http://www.phoenix.com/PlatSS/products/specs.html можно найти ссылки на спецификации Enhanced Disk Drive (устройств с более чем 1024 треками и поддержкой расширения интерфейса BIOS Int 13h от IBM/Microsoft).

Дополнительную информацию от одного из производителей BIOS об ограничениях в 2 и 4 Гб можно найти в статьях Micro Firmwares "Notes on Installing Hard Drives Larger Than 2 GB" (http://www.firmware.com/pb4ts/over2gb.htm) и "Issues with Hard Drives Over 4 GB" (http://www.firmware.com/pb4ts/over4gb.htm).

Описание различных типов разделов дано в статье Q69912 в Microsoft Knowledge Base (http://support.microsoft.com/support/kb/articles/Q69/9/12.asp).

Статья о различных версиях FDISK и их ограничениях — "Notes on DOS FDISK Command" (http://www.firmware.com/support/bios/fdisk.htm).

Утилиты управления разделами типа Partition Magic от PowerQuest (http://www.powerquest.com/) или Partition Commander от V Communications (http://www.v-com.com/) могут также выходить за границу 8 Гб и облегчают создание, изменение размера, перемещение и выбор типов разделов, и все это без потери данных!

С.ШИРКО,
E-mail: S_Shirko@tut.by

Ностальгия по DOS

Для старой доброй MS DOS (Дисковой Операционной Системы от Microsoft) тяжелые времена наступили уже с появлением Windows 95, а затем и Windows 98. Эти графические операционные системы (ОС) использовали DOS практически только для своей загрузки, а все общение с пользователем и "железом" взяли на себя. Однако настоящая беда пришла с появлением Windows 2000/XP — там DOS вообще "изгнали", сделав "окна" полноценной ОС. Многие скажут: "Какая ж это беда? Наоборот, здорово!". Так-то оно, конечно, так. Но вот нейметься США (Сергею Ширко Александровичу), и пишет он эту статью в надежде, что те, кто связался с компьютерами в эпоху Windows, попробуют поработать и в ДОС'е, как их отцы. Некрасиво так вот просто забывать операционную систему, сыгравшую в свое время столь значительную роль в развитии ПК.

Что есть MS DOS

DOS присутствует на каждом компьютере, работающем под Windows 95/98, хотя бы потому, что "окна" грузятся поверх MS DOS и, соответственно, работать без нее не могут. Для того чтобы выйти в "чистую" DOS, необходимо в процессе загрузки компьютера удерживать клавишу F8 и в появившемся меню выбрать пункт "Command prompt only". После появления приглашения командного интерпретатора, можно начинать работу в MS DOS. Но об этом несколько позже, а пока выясним, что собственно собой представляет эта дисковая операционная система.

Самыми главными программами MS DOS являются Boot Record (загрузчик ОС), Io.sys и Msdos.sys. Загрузчик операционной системы — это короткая программа, записанная строго в определенный сектор системного диска. Компьютер (а точнее, BIOS) после включения и самотестирования, находит программу-загрузчик на диске и передает ей управление. Boot Record, в свою очередь, проверяет наличие в корневом каталоге системных файлов Io.sys и Msdos.sys. Если они найдены, то загружает их в оперативную память компьютера. В противном случае выдается сообщение о том, что диск не системный. Файлы Io.sys и Msdos.sys составляют ядро MS DOS, и именно благодаря им программы получают доступ к ресурсам компьютера.

Кроме описанных выше, MS DOS обычно содержит файлы конфигурации (Config.sys и Autoexec.bat), командный интерпретатор (Command.com), позволяющий пользователю "общаться" с дисковой ОС посредством ввода команд, а также набор дополнительных утилит и драйверов (программы для работы с дисками, текстовый редактор и т.п.).

Файлы конфигурации

Config.sys и Autoexec.bat находятся в корневом каталоге системного диска и играют важную роль в настройке компьютера, т.к. определяют режимы работы ОС и загружают в оперативную память драйверы устройств и резидентные программы. Рассмотрим их подробнее.

Config.sys является обычным текстовым файлом, который обрабатывается модулем Io.sys в процессе загрузки ОС. Файл содержит набор команд, каждая из которых записывается с новой строки. Вот наиболее распространенные:

- **BREAK=ON|OFF** — устанавливает режим контроля за нажатием клавиш CTRL+C. Сочетание клавиш CTRL+C позволяет остановить выполнение программы или текущей операции (например, сортировки файлов). Как правило, MS-DOS отслеживает нажатие клавиш CTRL+C только в ходе считывания данных с клавиатуры и вывода на экран и принтер. Если задать для команды BREAK аргумент ON, действие сочетания CTRL+C распространится и на такие действия как чтение и запись на диск;
- **BUFFERS(BUFFERSHIGH)=n** — задает количество зон по 512 байтов каждая в памяти (верхней памяти), эти зоны винчестер использует в качестве буфера для накопления данных, что несколько повышает производительность системы (n — число от 1 до 99);
- **DEVICE(DEVICEHIGH)=путь/имя_драйвера** — загружает в память (в верхнюю память) необходимый драйвер;
- **FILES=n** — устанавливает предельное число файлов, которые MS-DOS позволяет открыть одновременно (n — число от 8 до 255);
- **INSTALL(INSTALLHIGH)=путь/имя_файла** — загружает в память (в верхнюю память) компьютера резидентные программы, которые остаются там вплоть до выключения или перезагрузки системы;
- **NUMLOCK=ON|OFF** — определяет состояние режима NUM LOCK после загрузки компьютера;
- **REM** — вставляет комментарии в файл Config.sys и командные файлы. Команда REM полезна также для быстрого отключения команд в указанных файлах. Записывается перед комментарием либо перед отключаемой командой.

Autoexec.bat — это пакетный (командный) файл, представляющий собой совокупность команд DOS, которые автоматически должны быть выполнены в процессе загрузки компьютера. Он чаще всего содержит команды для настройки режима работы клавиатуры и монитора, а также маршруты поиска наиболее часто используемых программ, что позволяет в командной строке для выполнения задавать только их имена (синтаксис: **PATH** путь1; путь2; путь3 и т.д.). Кроме того, в Autoexec.bat можно указать файлы, которые бы по вашему желанию автоматически загружались после окончания загрузки MS DOS или Windows. Например, строка ***D:\Audio\Winamp\winamp.exe*** запускала бы на моем компьютере проигрыватель Winamp при каждом старте Windows.

Работа в MS DOS

Общение пользователя с DOS осуществляется посредством ввода в командной строке определенных команд. Команда вводится после системного запроса или приглашения DOS, представляющего собой имя текущего диска или каталога. Далее записываются ее аргументы — объекты, с которыми команда работает. Кроме того, любая команда может иметь несколько ключей (параметров), отделенных от ее имени наклонной чертой. Например, если ввести в командную строку запись **C:\>format a:/s** и нажать клавишу Enter, то будет отформатирован диск A: с переносом на него системных файлов MS DOS (будет создана загрузочная дискета). В данном случае **C:\>** — приглашение; **format** — команда; ***a:*** — аргумент, указывающий, какой диск будет отформатирован; **/s** — ключ. Если ключ **/s** не вводить, то дискета будет отформатирована без переноса системных файлов. Аргументов и параметров для команды может быть несколько, либо вообще ни одного (например, команда **help**). Если каких-либо параметров в команде не хватает, то DOS расставляет их по умолчанию. Информацию о назначении конкретной команды и ее параметрах можно получить, если ввести имя команды с ключом **/?** (например, **format/?**).

В MS DOS полное количество команд составляет несколько десятков. Все они делятся на внутренние и внешние. Первые являются наиболее часто используемыми и размещаются непосредственно в командном интерпретаторе. Для выполнения внешних команд

необходимо наличие внешних утилит MS DOS (чаще всего они хранятся в папке Dos системного диска). Если command.com после ввода внешней команды не может найти на диске необходимую для ее выполнения утилиту, то выводится сообщение об ошибке. Отмечу, что каталог, содержащий пакет внешних утилит MS DOS, должен быть прописан в autoexec.bat в разделе **path**. Привожу список наиболее часто употребляемых команд MS DOS.

Внутренние:

- **cd** — смена каталога;
- **cls** — очистка экрана;
- **copy** — копирование файлов (позволяет также объединить несколько файлов в один);
- **del** — удаление файлов;
- **dir** — вывод оглавления диска или каталога;
- **exit** — выход из командного интерпретатора;
- **md** — создание каталога;
- **ren** — переименование файлов;
- **rd** — удаление каталога;
- **sys** — копирование системных файлов;
- **type** — просмотр содержимого файла (например, текстового).

Внешние:

- **chkdsk** — проверка диска на наличие ошибок;
- **diskcopy** — копирование с дискеты на дискету;
- **fc** — сравнение файлов;
- **fdisk** — разбиение диска на разделы;
- **find** — поиск заданного текста в файле;
- **format** — форматирование диска;
- **help** — вывод справки по MS DOS;
- **mem** — вывод информации о распределении оперативной памяти компьютера;
- **mirror** — создание образа диска;
- **sort** — сортировка файлов;
- **undelete** — восстановление удаленных файлов;
- **unformat** — восстановление отформатированного диска.

При работе в MS DOS не забывайте, что максимально возможная длина имен файлов составляет восемь символов. То, что идет после восьмого символа, ОС не воспринимается. Стоит отметить, что некоторые специальные зарезервированные имена позволяют обращаться к устройствам компьютера как к файлам: **prn** используется для обращения к принтеру, **con** — к стандартным устройствам ввода/вывода (клавиатуре и монитору соответственно), **nul** — обозначает пустое устройство. Это, вкупе со специальными символами переадресации (< и >), позволяет осуществлять перенаправление информации между файлами и некоторыми устройствами компьютера. Вот несколько примеров:

- **copy con text1** — данная команда позволяет набрать текст с клавиатуры и сохранить его в файле text1 (для окончания ввода текста нужно нажать одновременно клавиши Ctrl и Z);
- **mem>memory** — записывает информацию о распределении оперативной памяти компьютера в текстовый файл memory;
- **dir c:>con** — выводит оглавление системного диска на экран монитора (эквивалентно команде **dir c:**, т.к. монитор является стандартным устройством вывода информации);
- **dir c:>prn** — выводит оглавление диска на принтер. К слову, в MS DOS матричный принтер печатает значительно быстрее, чем в Windows, что позволяет в некоторых случаях сэкономить массу времени. Единственное, что требуется — это поддержка русских шрифтов со стороны принтера.

Командные файлы

Командными называют файлы с расширением **.bat**, являющиеся по своей сути обычными текстовыми файлами, в которых записан набор команд DOS. Эти команды последовательно выполняются процессором при запуске файла. Создать командный файл можно в любом текстовом редакторе. При этом каждая команда должна записываться с новой строки. Помимо перечисленных ранее, для создания bat-файла часто применяются следующие команды: **echo** — позволяет вывести сообщение на экран (текст сообщения вводится через пробел после имени команды); **@echo off** — отключает вывод на экран всех команд, которые прописывает MS DOS в процессе выполнения файла; **pause** — приостанавливает выполнение командного файла до нажатия любой клавиши; **call** — вызывает на выполнение внешний командный файл (его имя записывается через пробел). Ниже приведен пример простого командного файла, позволяющего проверить дискету на наличие ошибок:

```
cls
@echo off
echo Вставьте дискету в дисковод A:
pause
chkdsk a:
```

MS DOS и Windows

Вообще-то, все сказанное выше о дисковой операционной системе, в первую очередь относилось к работе в реальном режиме MS DOS. На самом же деле практически все приведенные команды можно опробовать и после загрузки Windows 95/98. Для этих целей в папке WINDOWS находится командный файл Command.com, который можно вызвать при помощи ярлыка "Сеанс MS-DOS" в меню Пуск. А в папке C:\WINDOWS\COMMAND — соответствующие драйверы и утилиты. Однако не все программы могут функционировать в таком режиме — некоторым из них (в основном это программы-тесты и старые игры) для корректной работы требуется "чистая" DOS.

Когда-то DOS была полноценной и, можно сказать, единственной операционной системой. Однако с момента появления "окон" ее статус значительно изменился. Первые версии Windows являлись лишь графической надстройкой над MS DOS и не могли перешагнуть возможности, предоставляемые дисковой операционной системой. Позднее появились более самостоятельные Windows 95/98, в которых внутренний мир DOS претерпел существенные изменения. И если мы, имея Windows 95/98 в качестве ОС, откроем при помощи блокнота файл Msdos.sys, то мы увидим, что этот фундаментальный для DOS системный файл превратился в обычный текстовый, содержащий сведения о Windows и ее настройках. В файл Io.sys Microsoft записала начальную заставку Windows 95/98, как бы подтвердив неотвратимость прогресса. Ну а Config.sys, который ранее отвечал за загрузку всех драйверов, теперь содержит от силы две-три строки — его обязанности возложили на себя такие файлы как Win.ini и System.ini.

Но, как бы там ни было, тот же Config.sys позволяет делать в Windows 95/98 некоторые интересные вещи. Например, если в нем прописать две строки: **device=C:\Windows\Himem.Sys** и **device=C:\Windows\Ramdrive.sys 5120 /e**, то у вас в оперативной памяти создастся электронный диск объемом 5 Мб, который можно будет использовать как обычный, только очень быстрый, диск.

Относительно недавно появилась еще одна версия "окон" — Windows Me. В ней поддержка DOS в реальном режиме вообще отсутствует. Поэтому если вам потребуется запустить программу, работающую в "чистой" DOS, придется воспользоваться загрузочной дискетой. Ну а такие файлы как Autoexec.bat и Config.sys вообще потеряли свою былую значимость. Все ясно — на наших глазах DOS доживает свои последние дни. С одной стороны, это вроде как хорошо — двигаемся, так сказать, вперед. А с другой — чего-то мне все-таки грустно. Ностальгия.

Г.ТРОЯН,
г.Минск, РИВШ БГУ,
E-mail: G-Trojan@yandex.ru

Эффективный поиск информации в Internet

(Окончание. Начало в №№9-11/2001)

Параметры эффективности поиска информации

После подробного изучения основных возможностей инструментов обратимся к проблеме эффективности поиска. Основными параметрами эффективности поиска являются:

- **полнота поиска** — отношение числа найденных документов к общему числу релевантных документов;
- **точность поиска** — отношение числа релевантных документов к общему числу полученных документов;
- **актуальность ссылок на документы** — существование найденных документов в сети в настоящий момент;
- **скорость поиска**.

Факторы, влияющие на эффективность поиска

Итак, мы выяснили, что в Internet существуют различные инструменты поиска, обладающие разными функциональными возможностями. Качество поиска, таким образом, зависит в первую очередь от параметров конкретной поисковой системы, например, от размеров индекса, от способа поиска (уточнение тем или поиск по запросу) и т.д. Далее, работая с конкретной поисковой системой, нужно иметь представление о методах составления запросов, знать необходимые операторы.

Таким образом, можно выделить следующие факторы, влияющие на эффективность поиска [7-9]:

- свойства и возможности поисковой системы;
- качество формулировки запроса пользователем.

Сравнение возможностей поисковых систем

Каким образом можно оценить качество поискового инструмента? Параметры, по которым обычно сравнивают поисковые системы [4, 7-9]:

- **количество проиндексированных страниц** (объем индекса);
- **период обновления индекса**. Этот показатель влияет на такой параметр как актуальность найденных ссылок. Чем чаще обновляется индекс, тем реже в результатах поиска будут встречаться устаревшие ссылки;
- **задержка перед прописки**. Данный параметр указывает на временной интервал перед занесением описания Web-страницы в индекс после просьбы ее автора;
- **количество поддерживаемых операторов**;
- **сортировка по категориям**;
- **стандартный оператор, объединяющий по умолчанию несколько ключевых слов**. Если стандартным оператором является оператор И, поисковая машина автоматически будет искать документы, на которых обязательно будут присутствовать все введенные ключевые слова. В противном случае (оператор ИЛИ) будут найдены документы со всеми ключевыми словами и с каждым по отдельности;
- **поиск точной фразы**;
- **поиск по шаблону** (поиск слов с различными окончаниями);
- **учет словоформ**. В случае автоматического режима учета словоформ система будет искать в документах слово со всеми его изменениями;
- **чувствительность к заглавной букве**. Если система не различает заглавные и строчные буквы, результаты поиска будут менее качественными;
- **форма представления результатов**;
- **дополнительные возможности** — поиск статей в группах новостей, людей, организаций, мультимедийных файлов, и т.д.

Планирование поисковой процедуры

Теперь рассмотрим, как пользователю лучше подготовиться к составлению запроса. Прежде всего, необходимо провести всесторонний лексический анализ информации, которую вы собираетесь искать. Затем желательно составить набор ключевых слов (при необходимости, на нескольких языках) в виде отдельных терминов и словосочетаний, специфичных для вашей предметной области [7].

Далее необходимо исследовать различные поисковые инструменты. Последовательность ваших действий:

- выбор поискового инструмента;
- точная формулировка запросов с использованием операторов, поддерживаемых данным поисковым инструментом;
- отправка тестовых запросов;
- анализ результатов поиска (по количеству и релевантности ссылок);
- при необходимости, корректировка запроса;
- повторный поиск;
- ...

Приемы эффективного поиска

Исходя из вышеизложенного, можно выделить следующие приемы эффективного поиска:

- *поиск информации общего характера в поисковых системах-каталогах*. В каталогах вы, как правило, найдете специализированные серверы в искомой области;
- *поиск узкоспециальной информации в поисковых машинах*. Для проведения более обширного поиска явно недостаточно использовать только системы-каталоги с ограниченным числом описанных ресурсов. Кроме того, узкоспециальная информация в каталогах может просто отсутствовать. Поэтому необходимо проводить поиск подобной информации в поисковых машинах, обладающих индексами большого объема;
- *использование операторов или бланка расширенного запроса для сужения области поиска*. Для проведения качественного поиска не-

обходимо ознакомиться с языком запросов конкретной поисковой машины. Эффективным и простым способом решения проблемы составления качественного запроса является использование режима расширенного поиска;

- *использование функции поиска в найденном*. Большинство поисковых систем поддерживают возможность поиска внутри полученных результатов. Как правило, для этого нужно включить специальный флажок **Искать в найденном** и ввести дополнительные слова для повторного поиска среди найденных по запросу страниц;

- *использование функции поиска похожих документов* для нахождения релевантных страниц по выбранному вами образцу;

- *использование метапоисковых систем и программ ускоренного поиска информации*. Для получения общего обзора документов целесообразно использовать возможности метапоисковых систем или программ ускоренного поиска. Напоминаем, что данные инструменты поиска отправляют ваш запрос сразу нескольким поисковым системам, и от каждой системы получают несколько самых релевантных ссылок;

- *просмотр раздела* ***Ссылки*** *на специализированных сайтах*. Авторы многих специализированных Web-узлов накапливают свои коллекции ссылок по тематике сайта. Зачастую вы найдете в этих коллекциях много полезных источников, сэкономив время, затрачиваемое на самостоятельный поиск с использованием рассмотренных выше инструментов;

- *поиск ответов на вопросы в группах новостей*. При желании можно обратиться с конкретным вопросом о помощи в специализированную группу новостей. Найти нужную группу можно, используя специальные инструменты поиска, которые мы рассмотрим далее;

- *подписка на специализированные списки рассылки*. После оформления подписки на специализированный список рассылки, вы сможете получать по электронной почте свежую информацию по выбранной тематике, а также задавать вопросы вашим коллегам по подписке.

Поиск статей в группах новостей

Обсудим проблему поиска статей в группах новостей. Инструментами поиска в данном случае могут являться некоторые поисковые машины WWW, которые индексируют не только пространство WWW, но и статьи в телеконференциях, и имеют специальный режим поиска именно в этом ресурсе. Поиск среди сообщений групп новостей, опубликованных за последние полгода, поддерживает, например, поисковый сервер Google (http://groups.google.com). Поисковые системы WWW весьма оперативно индексируют группы новостей и содержат информацию о статьях, реально существующих в сети. Для поиска в архивах новостей существуют специализированные системы, самой известной из которых до недавнего времени являлась система Deja (http://www.deja.com). В феврале 2001 г. компания Google Inc. объявила о приобретении системы Deja.com's Usenet Discussion Service. Так что теперь пользователи поисковой системы Google по адресу http://groups.google.com могут проводить поиск также и в подключенном архиве системы Deja, который содержит свыше 500 миллионов сообщений, индексируемых с 1995 г.

Поиск файлов

Теперь рассмотрим инструменты, позволяющие проводить поиск файлов. Многие поисковые системы WWW оказывают услугу поиска мультимедийных файлов (Altavista, Aport и др.). Для этого нет необходимости знать специальные операторы, а достаточно перейти с домашней страницы по ссылкам Картинки (Images), MP3/Audio или Video к специальному режиму поиска. Поиск проводится по возможному имени файла или по тексту в комментарии к ссылке на мультимедийный файл. Вы можете спрогнозировать имя файла, например, файл с изображением орла может называться eagle.gif. Или догадаться, что фото Билла Гейтса будет иметь соответствующую подпись.

Что касается поиска программного обеспечения, то во Всемирной Паутине существуют поисковые Web-серверы с коллекциями условно-бесплатного ПО; некоторые из них специализируются на поиске программного обеспечения для Internet, другие предлагают найти приложения для конкретной операционной системы. Эти системы в конечном итоге приведут вас к конкретному FTP-серверу, с которого и можно скачать искомый программный продукт. Следует упомянуть серверы Archie, также оказывающие услугу поиска файлов на FTP-серверах, однако пользоваться Web-серверами гораздо удобнее.

Адреса популярных серверов для поиска программного обеспечения:

- коллекция **TuCows (http://www.tucows.com)**;

- коллекция условно-бесплатного ПО — **CNET Shareware.com (http://www.shareware.com)**;

- система поиска ПО — **CNET Download.com (http://download.cnet.com)**;

- поиск файлов и ПО на сервере Lycos — **Lycos Computer Free Downloads (http://computers.lycos.com/downloads)**.

Поиск адресной информации организаций и людей

Рассмотрим поисковые инструменты для поиска адресной информации. Различают два способа поиска: белый (white) и желтый (yellow) поиск.

White-поиск — поиск адресной информации по заранее известному имени адресата (имя человека или название организации).

Yellow-поиск — поиск имени или названия и адресной информации по дополнительным признакам (по роду деятельности, по географическому признаку и т.д.).

Обычно системы Yellow Pages фактически сразу включают в себя и White Pages — у найденного адресата сразу видны его телефон и почтовый адрес. Кроме того, некоторые Yellow Pages позволяют искать просто в алфавитном списке своих абонентов (white-поиск). С другой стороны, White pages также содержат элементы yellow-поиска — кроме задания собственного имени, они обычно позволяют указать название города, штата и другие признаки, суживающие поиск данных (что необходимо в случае многих однофамильцев). Возможно, именно поэтому многие on-line-телефонные справочники, выполняю-

щие фактически white-поиск, называют себя Yellow pages.

Ниже приведены адреса некоторых Web-систем для поиска адресной информации для людей и организаций.

Поиск людей:

- поиск людей на Yahoo (http://people.yahoo.com);

- система WhoWhere (www.whowhere.com);

- система Bigfoot (www.bigfoot.com).

Поиск организаций:

- раздел **Желтые страницы (Yellow pages)** на поисковых системах;

- http://www.yellowpages.com — специализированный сервер для поиска в США и других странах.

Литература

1.Информационные системы и поисковые технологии. — РГГУ, 1999, http://s2.vntic.org.ru/iis_intro.htm.

2. Информационно-поисковые системы — http://www.comptek.ru/yandex/yand_about.html.

3. А.Аликберов. Поисковые машины. — http://citforum.ru/win/internet/search/index.shtml.

4. Danny Silivan. Search Engine Sizes. — http://www.searchenginewatch.com/reports/sizes.html.

5. Г.М. Троян. Поиск в русскоязычной части Интернет: поисковая система Rambler. — Радиолюбитель. Ваш компьютер, 1999, № 8-10.

6. Современный самоучитель работы в сети Интернет. Самые популярные программы. Под ред. Комягина В.Б. — М.: "Издательство ТРИУМФ", 1999, 368 с.: ил.

7. М.Талантов. Профессиональный поиск в Интернете: планирование поисковой процедуры. — КомпьютерПресс, 1999, № 7.

8. М.Талантов. Поиск информации в Интернете: подводные камни. — КомпьютерПресс, 1999, №9.

9. А.Коваль. Эффективный поиск в Интернет. — http://info.mplik.ru:8081/news/computerman/3_99/find.htm

10. Г.М. Троян. Поиск в русскоязычной части Интернет: поисковая система Yandex. — Радиолюбитель. Ваш компьютер, 2000, №1-3.

11. Г.М. Троян. Поиск в русскоязычной части Интернет: поисковая система Апорт. — Радиолюбитель. Ваш компьютер, 2000, № 4-6.

С.РЮМИК,
г.Чернигов,
E-mail: 1972r@mail.ru

Зачем модему "ручка громкости"?

Современные модемы рассчитаны на то, чтобы ими можно было пользоваться быстро, просто и удобно. На передней панели модема нет ни одного функционального переключателя или ручки управления (выключатель питания — не в счет). Настройка режимов работы модема производится в автоматическом режиме по командам от компьютера.

Громкость звука встроенного в модем динамика также регулируется программно по четырем градациям — тихо, средне, громко, очень громко. В наборе AT-команд [1] каждому из этих уровней соответствует своя команда — ATL0, ATL1, ATL2, ATL3. Кроме того, можно задавать четыре режима работы динамика: ATM0 — всегда выключен (OFF); ATM1 — включен (ON) до тех пор, пока не произошло соединение (CONNECT); ATM2 — всегда включен (ON); ATM3 — включен (ON) после набора номера до тех пор, пока не произошло соединение (CONNECT).

Как правило, по умолчанию устанавливаются режимы L1 и M1. На первых порах этого вполне достаточно, но по мере приобретения опыта работы в Интернете вкусы меняются, планка запросов поднимается все выше и выше.

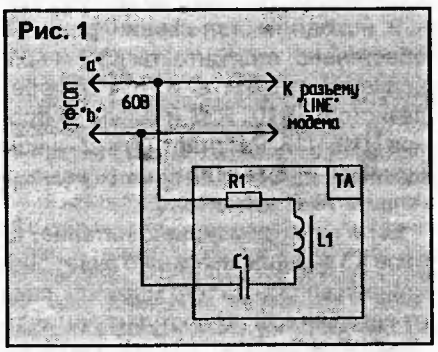
Представим себе ситуацию. Глубокая ночь, в одном из окошек дома виден приглушенный свет. Можно предположить, что в ночной "серфинг" по Интернету отправился очередной пользователь, благо, ночью тарифы ниже, и прохождение сигнала получше.

Однако полностью избежать конфликтов не удается — звуки, доносящиеся из динамика модема, и щелчки телефонного аппарата при автоматическом наборе номера отнюдь не укрепляют нервную систему окружающих. Помочь в данном случае могут не только организационные мероприятия (например, ночевка у друга), но и технические. На последних остановимся подробнее.

Устранение щелчков телефонного аппарата

Первым делом необходимо проверить схему подключения модема и телефонного аппарата (ТА) к телефонной сети общего пользования (ТфСОП). В простейшем случае их соединяют параллельно, как показано на рис.1. Названия цепей — "a" и "b" — являются у связистов стандартными для "двухпроводного окончания".

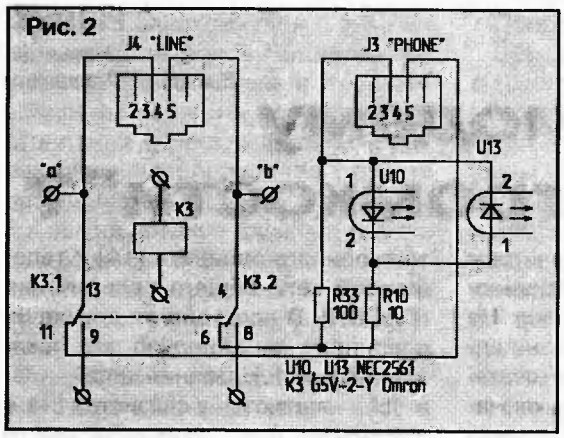
Рис. 1



Внутреннее устройство электромеханических ТА при положенной на рычаг телефонной трубке можно представить в виде цепочки элементов R1, L1, C1, где R1 и L1 — омическое сопротивление и индуктивность катушки звонка. В электронных ТА механический звонок заменяется электронной схемой, которая все равно будет подключена параллельно проводам "a" и "b".

При работе модема в режиме импульсного набора номера происходит поочередное замыкание/размыкание его внутренних контактов и соответствующее закорачивание/размыкание цепей "a" и "b". При этом напряжение 60 В периодически "подсаживается", а получающиеся импульсы напряжения прикладываются непосредственно к входу ТА. Кратковременное прохождение тока через катушку L1 приводит к срабатыванию молоточка звонкового механизма и появлению довольно громких щелчков и перезвонов.

Как избежать этого? Самый простой вариант — подключить ТА к специальному разъему "PHONE", который обычно располагается на корпусе модема в непосредственной близости от гнезда "LINE". На рис.2 для примера приведена схема входной части модема ACCURA-33600. Провода "a" и "b" ТфСОП подводятся к контактам 3 и 4



разъема J4 "LINE", а провода от ТА — к контактам 3 и 4 разъема J3 "PHONE".

В исходном положении реле К3 обесточено, его контакты К3.1 и К3.2 подключают цепи "a" и "b" к ТА через резисторы R10, R33 и диоды оптронов U10, U13. Работает ТА в обычном режиме — можно позвонить самому и принять звонок извне.

Если на модем подано питание, то реле К3 вначале все равно будет обесточено вплоть до того момента, пока от компьютера не поступит команда "НАБОР НОМЕРА". При этом реле К3 срабатывает, ТА полностью отключается от ТфСОП, и щелчков при наборе номера слышно не будет. После окончания сеанса связи в Интернете реле К3 переходит в исходный режим (обесточенное состояние), и ТА вновь подключается к линии.

Если в модеме отсутствует гнездо "PHONE" (что характерно для старых моделей), то отключать ТА от сети 60 В можно вручную при помощи обычного двухпозиционного переключателя. Кстати, такой вариант более предпочтителен, чем параллельная работа ТА и модема, поскольку контур L1, C1 (рис.1) может ухудшить амплитудные и фазовые характеристики канала связи.

Одна незадача — придется постоянно напрягать память, чтобы не забывать вовремя подключать ТА обратно к ТфСОП...

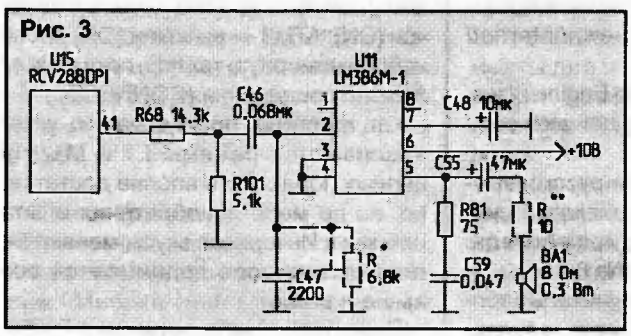
Уменьшение громкости звука динамика

Решение "в лоб" — установить минимальную громкость звука командой ATL0 и максимально ограничить время включения динамика командой ATM3. В некоторых случаях этого оказывается достаточно, однако минимальная громкость не всегда вписывается в понятие "тихо". Можно, конечно, полностью отключить динамик командой ATM0, но тогда появляется вероятность пропуска сбойного сеанса связи с модемом провайдера и, как следствие, лишняя трата финансов при поминутной тарификации переговоров.

Небольшое пояснение. Если бы звук был слышен, то опытный пользователь в первые же секунды после дозвона на слух смог бы определить неверную тональность сигнала ответа и быстро разорвать связь. Для сведения, телефонные тарификаторы городских переговоров обычно не засчитывают связь, если она длится менее 5...8 секунд с момента дозвона (после поднятия трубки на противоположной стороне).

Как было бы хорошо, если бы в модеме имелась "ручка громкости" или хотя бы подстроечный резистор для ее регулировки! Тем не менее, уменьшить громкость можно и самостоятельно, сделав небольшие доработки в электрической схеме модема.

На рис.3 приведена выходная часть схемы канала звука модема



ACCURA-33600. Звуковой сигнал генерируется микросхемой U15 RCV288DPI (Rockwell). После резистивно-емкостного делителя R68, R101, C46, C47 сигнал поступает на усилитель звука, выполненный на микросхеме U11 LM386M-1 (National Semiconductor). Ее параметры:

- напряжение питания — 4...12 В;
- ток покоя — 4...8 мА;
- выходная мощность — до 325 мВт;
- коэффициент нелинейных искажений — не более 0,2% на частоте 1 кГц;
- полоса пропускания — 0...300 кГц;
- корпус —SOP-8.

При "висящих в воздухе" выводах 1 и 8 коэффициент усиления LM386M-1 составляет 26 дБ (20 раз по напряжению) [2].

Уменьшить звуковой сигнал можно двумя способами. Во-первых, шунтирующим резистором R* по входу, во-вторых — последовательным резистором R** по выходу. Чтобы снизить звуковое напряжение частотой 1 кГц на динамике BA1 от 3 до 12 дБ (от 1,41 до 4 раз), сопротивления резисторов R* и R** должны изменяться в пределах 5...0,5 кОм и 3,3...24 Ом соответственно.

Разумеется, нет необходимости ставить в схему сразу два резистора, достаточно одного из них. Резистор R** обладает определенной универсальностью — его можно поставить в любой модем, даже не зная его электрической схемы. Резисторы R* и R** могут быть как постоянными, так и подстроечными, мощностью 0,125...0,25 Вт. Правда, найти подходящий низкоомный подстроечный резистор R** гораздо сложнее, чем высокоомный R*. Допускается также коммутация цепей регулировки громкости при помощи переключателей.

Подбор номиналов резисторов — дело сугубо индивидуальное. Разные люди обладают разной слуховой чувствительностью и требованиями к комфортному прослушиванию. Кроме того, исходные уровни громкости, устанавливаемые командами ATL0...ATL3, зависят от типа модема и могут иметь субъективно разную величину.

Схемные решения, приведенные в статье, были опробованы на практике. В частности, себе в модем я поставил резистор R* номиналом 1,5 кОм.

Литература

1. Рюмик С. Интернет на "хороших" и "плохих" линиях. — Радиомир. Ваш компьютер, 2001, №11, С.15.

2. LM386 Low Voltage Audio Power Amplifier. — National Semiconductor, http://www.national.com/ds/LM/LM386.pdf

Рисунок О.Попова

ЛИСТАЯ СТРАНИЦЫ

О том, как сделать "мобильным" винчестер вашего компьютера, рассказывает *А.Кузнецов* в статье "Mobile rack SNT — дом для вашего винчестера" **(Подводная лодка, №8/2001, С.25).**

Объем информации, которую во многих случаях приходится переносить с компьютера на компьютер, постоянно растет. Это программы, игрушки, тексты, рисунки, музыка, фильмы и многое другое. "Гибкие" носители — не лучший выбор из-за относительно малой емкости, быстродействия, высокой удельной цены хранения информации. Кроме того, в некоторых случаях приходится устанавливать новое ПО. Многие пытаются использовать винчестер для решения данной задачи. Однако это достаточно трудоемко, т.к. приходится снимать крышку с корпуса (часто крышка вообще не ставится), отвинчивать шурупы, крепящие винчестер, отсоединять кабели. Особо аккуратно нужно обращаться с винчестером при перевозке.

Практически все эти проблемы решаются при использовании mobile rack (съемного модуля для транспортировки жесткого диска). Устройство предназначено для установки в свободный 5,25"-отсек корпуса. Оно состоит из двух частей — рамки с направляющими, которая крепится в отсеке, и коробки для размещения винчестера. В рамке имеются внешние разъемы для подключения стандартного шлейфа IDE и силового кабеля. С внутренней стороны рамки размещен единый разъем, к которому через ответную часть подключается коробка с винчестером. В ней жесткий диск при установке подключается стандартным образом.

Для тестирования использован mobile rack марки SNT-129A66. Кратко рассмотрены его достоинства и недостатки. Следует отметить, что проблема разогрева скоростного винчестера решается за счет вентилятора, установленного у решетчатой передней стенки коробки. Возможна установка еще двух дополнительных миниатюрных кулеров.

Технические характеристики одной из первых серийных видеокарт Leadtek Winfast GeForce3 на базе видеочипа GeForce3 от nVIDIA обсуждает *А. Кузнецов* **(Подводная лодка, № 8/2001, С.26).**

Прежде всего отмечены основные особенности графического процессора GeForce3. Так же как и его предшественник GeForce2, он имеет четыре конвейера рендеринга с двумя текстурными блоками на каждом. Правда, теперь он может накладывать по четыре текстуры за один проход в два такта. Используется та же 128-разрядная шина памяти с поддержкой DDR SDRAM.

Главные новшества скрываются в новой архитектуре nfiniteFX Engine. В ней реализованы на аппаратном уровне все возможности Microsoft DirectX 8.0. Главные элементы архитектуры — программируемые вершинные и пиксельные шейдеры. С помощью первых осуществляются операции над вершинами полигонов. Конечным результатом является достижение ряда специальных эффектов, позволяющих добиться большей реалистичности графических объектов. Пиксельные шейдеры служат для расширения возможностей по обработке пикселов при рендеринге. В итоге получается более правдоподобное изображение поверхностей объектов

Еще одним нововведением в GeForce3 является новая архитектура подсистемы видеопамяти — Lightspeed Memory. В ней используются четыре 64-разрядных контроллера вместо одного 256-разрядного. В результате этого уменьшаются временные задержки и сводится к минимуму передача ненужной информации при работе с данными небольшого объема. Кроме того, реализованы специальные технические решения, уменьшающие обмен информацией между памятью и Z-буфером.

Эволюционный характер носит совершенствование технологии получения полноэкранного сглаживания. Специальный метод High-resolution Antialiasing (HRAA) позволяет повысить разрешение финального изображения при увеличенной скорости вывода кадров.

Технические характеристики видеокарты Leadtek Winfast GeForce3:

- графический процессор — nVIDIA GeForce3;
- объем DDR SDRAM — 64 Мбайт;
- RAMDAC — 350МГц;
- интерфейс — AGP 2X/4X с поддержкой режима Fast Writes;
- скорость текстурирования — 3,2 млрд. пикселов;
- пропускная способность памяти — 7,36 Гбайт/с;
- оптимизация и поддержка — Microsoft DirectX 8 и OpenGL 1.2;
- поддержка драйверов — Win 9x, Win ME, Win NT4, Win2k;
- ТВ выход — S-video/RCA ТВ-выход с разрешением до 800x600 пикселов;
- разрешение — до 1280x1024 пикселов;
- имеется DVI-разъем.

Первое, что отмечает автор — внушительную систему охлаждения. Вентилятор на чипе утоплен в широкий серебристый радиатор с ребрами, нависающими над модулями памяти. Радиаторы на последних, хотя и выполнены по безреберной схеме, но имеют впечатляющую толщину — около 5 мм. Они полностью закрывают модули памяти EliteMT со временем доступа 4 нс. Кроме стандартного VGA-выхода, на карте находится ТВ-выход (S-Video) и разъем DVI для подключения ЖК-панелей. Таким образом, видеокарта вполне универсальна и может выводить изображение на различные внешние устройства.

Программное обеспечение написано достаточно грамотно. Программисты Leadtek пишут собственные драйверы на основе референсных от nVIDIA. Вместе с утилитой WinFox, они позволяют оперативно получать информацию о параметрах работы и легко осуществлять управление тонкими настройками. Опция Speed Runner утилиты дает возможность владельцу данной модели изменять частоту ядра процессора и чипов памяти, тем самым осуществляя ее разгон. Необходимость в программах сторонних разработчиков для проведения этой операции отпадает.

Рассматриваемое изделие сравнивалось с видеокартой на наиболее мощном чипсете предыдущего поколения GeForce2 Ultra. Ею стала модель компании Creative. В качестве тестовых программ были выбраны 3Dmark2001 компании MadOnion и Vulpine GL Mark. Измерения проводились в экранных разрешениях 800x600 и 1600x1200, как наиболее показательных с точки зрения скорости и качества соответственно. Глубина цвета экрана и текстур была 32-битной.

Все результаты были получены на следующей платформе:

- системная плата — Chaintech 6ATA4;
- процессор — Intel Pentium III 667 МГц;
- ОЗУ — 256 Мбайт;
- жесткий диск — Quantum AS 40 Гбайт UDMA66;
- ОС — Windows 98 SE с установленным DirectX 8;
- в качестве драйвера при тестировании использовался референсный Detonator версии 12.41.

Результаты сравнения приведены в таблице

	Creative GeForse2 Ultra	Leadtek Winfast GeForce3
3Dmark2001 800x600 (3DMarks)	2698	3639
3Dmark2001 1600x1200 (3DMarks)	2075	2796
Gmark 800x600 (fps)	46,6	49,8
Gmark 1600x1200 (fps)	20,5	27,5

Кроме того, автор отмечает большее сходство с реальностью при просмотре специализированных тестовых сцен, поддерживающих новые аппаратные возможности GeForce3.

Основные параметры, достоинства и недостатки сверхминиатюрного компьютера рассмотрены в статье *Н. Прокофьева* "Скутер по-российски" **(Компьютер-Пресс, №7/2001, С.112...113).**

Главной особенность этого компьютера является компактный и легкий системный блок. Комплектация его следующая:

- процессор — Intel Celeron 766 МГц или Intel Pentium III 866 1000 МГц;
- память: кэш-память — 128/256 Кб встроенная, оперативная память — 128 Мб SDRAM (расширяемая до 256 Мб заменой модуля);
- видеоадаптер — Intel 82810E;
- видеопамять — 4 Мб SGRAM (SMA);
- дисковые накопители — жесткий диск E-IDE 6...30 Гб 2,5" UDMA/33;
- флоппи-дисковод — 3,5" 1,44 Мб (внешний, через LPT);
- CD-ROM/DVD-ROM — 24x CD-ROM (встроенный)/8x DVD-ROM E-IDE/ATAPI (встроенный);
- встроенная полудуплексная 16-битная 3D-звуковая система, встроенный динамик;

- габариты — 157x146x45 мм;
- вес — 950 г;
- разъемы внешних устройств: IrDA, линейный аудиовыход, микрофонный вход, монитор, разъем для подключения монитора, композитный видеовыход, компонентный видеовыход, два разъема USB, разъем для подключения блока питания, гнезда для подключения к телефонной линии и сети Ethernet 10/100, два разъема PS/2 для подключения мыши и клавиатуры;
- встроенный факс-модем MDC 56 Кбит/с (K56Flex & V.90);
- встроенный сетевой адаптер Ethernet 10Base-T/100Base-TX;
- источник питания — блок питания 110...230 В;
- программное обеспечение — Microsoft Windows Millennium Edition (рус).

Внутреннее устройство компьютера напоминает структуру notebook'a — ATA-винчестер в портативном исполнении, максимальная интеграция периферийного оборудования в материнскую плату и множество разъемов подключения на задней и боковой стенках. Значительную часть внутреннего пространства занимает CD-ROM (возможна установка DVD-ROM). Таким образом, по техническим характеристикам его можно отнести к офисным компьютерам.

Главным недостатком является отсутствие возможностей модернизации и расширения (за исключением наращивания оперативной памяти и установки другого, более мощного процессора). Интерфейс USB для этого мало пригоден, т.к. он достаточно медленный, и далеко не все устройства существуют во внешнем исполнении. Скоростные характеристики компьютера ограничивает достаточно медленный винчестер ATA-33. Неудачным является размещение CD-ROM в нижней части корпуса.

Главные его достоинства — миниатюрность и мобильность. Такой системный блок займет очень мало места на рабочем столе. Учитывая наличие в компьютере сетевой карты (10/100 Мбит) и модема, можно сделать вывод, что это хороший офисный компьютер.

Возможности программ, определяющих конфигурацию компьютера, рассматривает *С.Андрианов* в статье "А что же там внутри?" (**Мир ПК, №8/2001, С.44...50**).

Покупка компьютера, особенно если это ваш первый ПК — очень ответственное дело. Необходимо хорошо знать, что покупаешь. В прайс-листах сообщаются только основные технические характеристики, ну а при покупке с рук есть риск приобрести "кота в мешке". Некоторую информацию можно почерпнуть при изучении сообщений BIOS, кое-что знает Windows. Однако последняя может и не определить тип устройства или вообще его не обнаружить. В этом случае должны помочь специальные утилиты, предназначенные для определения конфигурации компьютера. Многие из них работают под DOS. Это объясняется тем, что в DOS программа имеет полный доступ к "железу", тогда как в Windows он ограничен. Да и загрузить DOS вместе с утилитой можно с одной дискеты.

Тестировались программы, доступные через Internet. При этом использовался компьютер следующей конфигурации:

- процессор Celeron 566 (разогнан до 664 МГц);
- материнская плата Micro Star MS-6361 Pro;
- оперативная память 128 Мб;
- видеокарта на чипе Riva TNT2 с 32 Мб видеопамяти;
- жесткий диск Fujitsu MPE3170AT, 17 Гб;
- звуковая карта SB Live! с цифровым сигнальным процессором EMU10K;
- TV-тюнер на микросхеме Bt878;
- сетевая плата на шине ISA;
- внешний модем, подключенный к порту COM2;
- накопитель Iomega ZIP на 100 Мб.

Таким образом, компьютер имеет ряд нестандартных устройств. Кроме того, при включении компьютера под DOS не загружались драйверы CD-ROM и мыши.

Рассмотрены как бесплатные:

- AIDA (http://hardware.jatekok.hu/aida.shtml);
- Informer (http://www.informer.newmail.ru);
- NSSI (http://www.navsoft.cz/);
- PC Analyzer (http://ic.doma.kiev.ua);
- System Speed Test (http://www.tcms12.rt.ru/dxover/indexr.htm);

так и условно-бесплатные программы:

- PC Config (http://www.holin.com/);
- Diag (http://www.diagnoseprogramm.de/indexe.htm);
- Dr.Hardware (http://www.drhardware.de/);
- HWiNFO (http://www.hwinfo.com/);
- System Analyzer (http://ourworld.compuserve.com/homepages/hniekus);
- WT Professional (http://titanic.nyme.hu/-wyx/wtpro).

Все они имеют свои достоинства и недостатки. Так, AIDA дала наиболее подробную информацию о компьютере, но не опознала цифровой сигнальный процессор EMU10K. Informer только определяет конфигурацию компьютера, хотя такие устройства как сетевая карта и модем не опознает, измерений быстродействия не проводит. NSSI показала частоту системной шины и коэффициент умножения, но неправильно. Кроме того, она не нашла мыши. HWiNFO определила не только частоту процессора, но и коэффициент умножения и частоту шины, его температуру, величины питающих напряжений и многое другое, но "не увидела" цифровой сигнальный процессор EMU10K, сетевую плату и TV-тюнер. Аналогичные замечания есть и по другим программам.

В целом, по результатам тестирования автор делает вывод, что программы, пытающиеся определить конфигурацию компьютера, нередко ошибаются, и для контроля лучше пользоваться не одной, а двумятремя, сравнивая между собой полученные данные. Поэтому предпочтение следует отдать бесплатным программам, в первую очередь — AIDA и Informer. Из условно-бесплатных можно рекомендовать только HWiNFO, да и то при обязательной ее регистрации. Остальные утилиты стоит использовать только как дополнение к перечисленным для уточнения деталей.

Долгожданному появлению компьютеров на базе 64-разрядного процессора от Intel посвящена статья *Д.Суворова* "Intel Itanium. Входит дракон" (**КомпьютерПресс, №7/2001, С.101**).

Ожидается, что в 2001 г. ряд компаний (среди них IBM, Dell, HP, Bull и др.) выпустят более 35 моделей серверов и рабочих станций на основе Itanium, а сотни поставщиков оборудования и программного обеспечения предложат различные продукты для систем, работающих на 64-разрядном процессоре. Для нового поколения процессоров уже написаны оптимизированные операционные системы: MS Windows, HP-UX 11i, AIX-5L и Linux.

В России система на базе процессора Itanium была опробована в лаборатории химической кибернетики химического факультета МГУ. На примере задачи квантово-химических расчетов сравнивалась производительность различных кластеров и систем на базе Pentium III, Pentium III Xeon и Itanium. Использование программы, оптимизированной для архитектуры Itanium, показало, что мощность двухпроцессорной рабочей станции на базе процессора Itanium с частотой 667 МГц сопоставима с быстродействием 9-10 однопроцессорных рабочих станций с процессором Intel Pentium III 500 МГц. При максимальной производительности компьютера примерно 5,3 Гфлоп достигнута скорость 3,3...3,5 Гфлоп.

Intel собирается выпускать процессоры Itanium с тактовой частотой 733 и 800 МГц, кэш-памятью третьего уровня 2 или 4 Мб, с возможностью установки от 2 до 32 процессоров в систему. Сейчас в стадии разработки находятся шесть процессоров семейства Itanium, а в ближайшее время планируются разработка и выпуск процессоров и систем второго поколения McKinley.

Itanium, по мнению Intel, является ее самой значительной разработкой в области микропроцессорной архитектуры с момента выпуска процессора 80386 в 1985 г. Лицо Itanium — это не только 64-разрядность, но и явный параллелизм EPIC (Explicity Parallel Instruction Computing), позволяющий разбивать выполняемую программу на несколько фрагментов, работающих параллельно. Архитектура IA-64 использует самые современные принципы: VLIW (Very Long Instruction Word — инструкции со сверхдлинными словами), улучшенный механизм предварительной подачи данных, предсказание ветвлений и др. Среди особенностей первого IA-64-процессора можно отметить увеличенное адресное пространство, средства обнаружения и исправления ошибок, а также пропускную способность до 2,1 Гб/с. Картридж Itanium предназначен для установки в комбинированный процессорный разъем Slot M. Статическая кэш-память новой конструкции размещена на одной плате с ядром процессора и работает с ним на одинаковой частоте.

Процессор Itanium лишь условно пригоден для работы с существующими 32-битными приложениями (падение производительности слишком заметно). Он плохо адаптирован для смешанных 32/64-разрядных сред, так как рассчитан на полный переход к 64-разрядным вычислениям.

Журналы листал И.Шевкун.

УСЛОВИЯ ЗАДАЧ IOI'2001

(Окончание. Начало №11/2001)

День 2

Задача 1. Игра "Score"

"Score" — это настольная игра для двух игроков, которые двигают одну и ту же фишку с позиции на позицию на доске. Доска имеет N позиций, пронумерованных от 1 до N, и множество дуг. Каждая дуга ведет от одной позиции к другой. Каждая позиция принадлежит первому или второму игроку, который называется владельцем этой позиции. Кроме того, каждая позиция имеет положительную цену. Все цены различны. Позиция 1 — это начальная позиция. Исходно оба игрока имеют счет 0.

Игра проводится по следующим правилам. Обозначим C — позицию, в которой находится фишка в начале хода. В начале игры позиция C — это 1. Шаг игры состоит из следующих операций:

1. Если величина C больше чем текущий счет владельца C, то величина C становится новым счетом владельца C. Иначе, счет владвльца C остается тем же. Счет другого игрока не изменяется в любом случае.

2. Владелец C выбирает одну из дуг от текущей позиции фишки, окончание дуги указывает новую текущую позицию фишки. Заметим, что игрок может сделать несколько последовательных ходов.

Игра завершается, после того как фишка возвращается в исходную позицию. Побеждает игрок с большим счетом в конце игры.

Дуги всегда организованы так, что выполняются следующие условия:

- из любой текущей позиции имеется хотя бы одна дуга;
- каждая позиция P достижима из стартовой позиции, то есть существует последовательность дуг от стартовой позиции к позиции P;
- гарантировано завершение игры через конечное число ходов.

Напишите программу, которая играет в эту игру и побеждает.

Все позиции, которые предъявляются вашей программе, таковы, что победа возможна вне зависимости от того, делаете ли вы первый ход. Программа противника играет оптимально, и это дает ей шанс выиграть у вашей программы.

Ввод и вывод

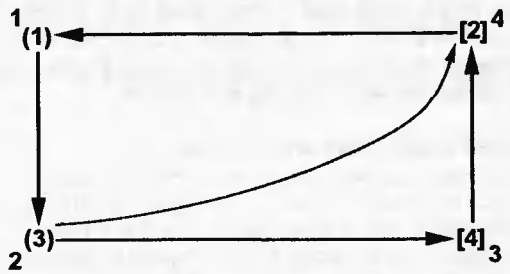
Ваша программа читает входные данные из стандартного ввода и пишет их в стандартный вывод. Ваша программа — Игрок 1, а программа жюри — Игрок 2. Когда ваша программа начинает работать, она должна считать следующую информацию из стандартного ввода:

- первая строка содержит одно целое число — количество позиций N, $1 \le N \le 1000$. Каждая из следующих N строк содержит N целых чисел с информацией о дугах. Если есть дуга из позиции i в позицию j, то j-е число в i-й строке из этих N строк есть 1, иначе — 0;
- следующая строка содержит N целых чисел — владельцев позиций. Если позиция i принадлежит игроку 1 (то есть вам), то i-е число есть 1, иначе — 2;
- следующая строкв содержит N целых чисел — цены позиций. Если i-е целое число есть j, то цена позиции i есть j. Все эти величины j удовлетворяют свойствам — $1 \le j \le N$, и все цены j различны.

Игра начинается с положения фишки в позиции 1. Ваша программа должна играть как описано ниже и закончить игру, когда фишка вернется в позицию 1:

- если сейчас очередь хода вашей программы, то она должна написать номер следующей позиции P, $1 \le P \le N$, в стандартный вывод;
- если сейчас очередь хода вашего противника, то ваша программа должна прочитать номер следующей позиции P, $1 \le P \le N$, из стандартного ввода.

Рассмотрим следующий пример. Ход игры представлен на рисунке



Позиции, помеченные (), принадлежат игроку 1, а позиции, помеченные [], принадлежат игроку 2. Каждая позиция имеет собственную цену, записанную внутри () или [], и номер позиции рядом. Запись игры имеет вид:

stdin	stdout	Пояснения
4		N
0 1 0 0		Информация о дугах из позиции 1
0 0 1 1		Информация о дугах из позиции 2
0 0 0 1		Информация о дугах из позиции 3
1 0 0 0		Информация о дугах из позиции 4
1 1 2 2		Владельцы позиций
1 3 4 2		Цены позиций
	2	Игрок 1 делает ход
	4	Игрок 1 делает ход
1		Игрок 2 делает ход в начальную позицию – игра окончена

По завершении игры игрок 1 имеет счет 3, игрок 2 имеет счет 2. Игрок 1 побеждает.

Инструкции по программированию

В приведенном ниже примере target — это целая переменная для позиции.

Если вы программируете на C++ и используете потоки ввода/вывода, вы должны организовать ввод и вывод следующим образом:

```
cin>>target;
cout<<target<<endl<<flush;
```

Если вы программируете на C или C++ и используете scanf и printf, вы должны так организовать ввод и вывод:

```
scanf ('%d', &target);
printf('%d\n',target); fflush (stdout);
```

Если вы программируете на Паскале, вы должны так организовать ввод и вывод:

```
Readln(target);
Writeln(target);
```

Инструмвнтальные средства

Вам дается программа (score2 для Linux, score2.exe для Windows). Эта программа читает описание игры из файла score.in в вышеописанном формате. Программа пишет эту информацию в стандартный вывод в том же формате. Этот вывод может быть использован для ввода в вашу программу в тестовых целях. После этого программа играет со случайной стратегией, читая ходы вашей программы из стандартного ввода и записывая собственные ходы в стандартный вывод.

Оценивание

За каждый тест, если вы выиграли, вы получаете все баллы, иначе — 0 баллов. Вначале ваша программа играет против соперника с предвлом времени, на 1 с превышающим предел времени на тест для задачи (в 2 раза — стандартное время на тест равно 1 с). Ввод и вывод вашей программы сохраняются. Затем ваша программа исполняется второй раз с вводом, перенаправленным из файла, и с учетом официального времени на тест. Ваша программа должна давать те же самые результаты, что и при первом запуске.

Задача 2. Двойное шифрованив

Известный стандарт шифрования AES (Advanced Encryption Standard) использует новый сильный алгоритм шифрования. Он работает с тремя блоками по 128 бит каждый. По блоку сообщения p и ключевому блоку k функция шифрования возвращает зашифрованный блок c, c=*E(p, k)*.

Обратная к функции AES-шифрования E есть функция дешифрования, такая, что *D(E(p, k), k)=p*, *E(D(c, k), k)=c*.

При двойном шифровании (Double AES) два независимых блока (*k1* и *k2*) используются последовательно — сначала *k1*, потом *k2*, *c2=E(E(p, k1), k2)*.

В данной задаче задано также целое число s. Только самые левые *4*s* битов всех ключей — релевантные, а остальные биты (самые правые *128-4*s* битов) все равны 0.

Вы должны восстановить пары шифровальных ключей для нескольких сообщений, зашифрованных методом Double AES. Вам даются исходный текст p, соответствующий результат его двойного шифрования c2 и структура шифровальных ключей, выраженная целым числом *s*.

Алгоритмы шифрования-дешифрования доступны как библиотечные функции. Вы должны прислать расшифрованные ключи, а не расшифровывающую программу.

Ввод

Вам дано 10 образцов проблемы в текстовых файлах, названных от double1.in до double10.in соответственно. Каждый из этих входных файлов состоит из трех строк. Первая строка содержит целое число *s*, вторая строка — исходный блок *p*, а третья строка — зашифрованный блок *c2*, полученный из *p* Double AES-шифрованием. Оба блока записаны как строки 32 шестнадцатеричных цифр (0...9, A...F). Библиотека содержит подпрограммы для преобразования строк в блоки. Все примеры имеют решение.

Вывод

Вы должны послать 10 выходных файлов, соответствующих заданным входным файлам. Каждый выходной файл содержит 3 строки.

Первая строка содержит текст

#FILE double I,

где I — это номер соответствующего входного файла.

Вторая строка содержит блок ключа 1 (*k1*), а третья строка содержит блок ключа 2 (*k2*), такие, что *c2=E(E(p, k1), k2)*.

Оба блока должны быть записаны как строки из 32 шестнадцатеричных цифр (0...9, A...F). Библиотека содержит процедуры преобразования блоков в строки. Если имеется несколько решений, нужно послать только одно из них.

Пример

Для примера мы используем файл с номером 0.

double0.in	A possible output file
1	#FILE double 0
00112233445566778899AABBCCDDEEFF	A0000000000000000000000000000000
6323B4A5BC16C479ED6D94F5B58FF0C2	70000000000000000000000000000000

Библиотека

Программа aestoolp.pas иллюстрирует, как использовать эту библиотеку из FreePascal:

```
FreePascal library
Linux:   aeslibp.p, aeslibp.ppu, aeslibp.o;
Windows: aeslibp.p, aeslibp.ppw, aeslibp.ow*

type
  HexStr = String [ 32 ]; { only '0'..'9', 'A'..'F' }
  Block  = array [ 0..15 ] of Byte; { 128 bits }

procedure HexStrToBlock ( const hs: HexStr; var b: Block );
procedure BlockToHexStr ( const b: Block; var hs: HexStr );
procedure Encrypt ( const p, k: Block; var c: Block );
  { c = E(p,k) }
procedure Decrypt ( const c, k: Block; var p: Block );
  { p = D(c,k) }
```

Программа aestoolc.c иллюстрирует, как использовать GNU C/C++ library:

```
GNU C/C++ library
(Linux и Windows: aeslibc.h, aeslibc.o*):

typedef char HexStr[33]; /* '0'..'9', 'A'..'F', '\0'-terminated
*/
typedef unsigned char Block[16];  /* 128 bits */

void hexstr2block ( const HexStr hs, /* out-param */ Block b );
void block2hexstr ( const Block b, /* out-param */ HexStr hs );
void encrypt ( const Block p, const Block k, /* out-param */
Block c );
  /* c = E(p,k) */
void decrypt ( const Block c, const Block k, /* out-param */
Block p );
  /* p = D(c,k) */
```

Ограничения

$1 \le s \le 5$ — количество дешифрируемых шестнадцатеричных цифр.

Совет

Хорошая программа сможет восстановить ключи менее чем за 10 с.

Задача 3. Склад

Финская компания имеет большой прямоугольный склад. На складе работают рабочий и менеджер. Стороны склада в порядке обхода называются — левая, верхняя, правая и нижняя. Площадь склада разбита на квадраты с равной площадью, составляющие строки и столбцы. Строки пронумерованы сверху целыми числами 1,2, ..., а столбцы пронумерованы слева целыми числами 1,2, ...

На складе есть контейнеры, которые используются для хранения отдельных устройств. Контейнеры имеют различные идентификационные номера. Каждый контейнер занимает ровно один квадрат. Склад такой большой, что количество контейнеров, которые прибывают, меньше чем количество строк и количество столбцов. Контейнеры не удаляются со склада, но иногда прибывают новые контейнеры. Вход в склад — в левом верхнем углу склада.

Рабочий разместил контейнеры возле левого верхнего угла таким образом, чтобы он мог найти их по их идентификационным номерам. Он использует следующий метод.

Предположим, что идентификационный номер следующего контейнера, который нужно вставить, есть *k* (для краткости — контейнер *K*). Рабочий идет вдоль первой строки и ищет первый контейнер с идентификационным номером больше чем *k*. Если он не находит такого контейнера, то он ставит контейнер *K* непосредственно за самым правым контейнером в строке. Если такой контейнер *L* найден, то контейнер *L* заменяется контейнером *K*, а контейнер *L* вставляется в следующую строку, используя тот же самый метод. Если

рабочий достигает строки, в которой нет ни одного контейнера, он ставит контейнер *K* в позицию самого левого квадрата этой строки.

Предположим, что контейнеры 3, 4, 9, 2, 5, 1 прибыли на склад в этом порядке. Тогда размещение контейнеров на складе будет таким:

```
1 4 5
2 9
3
```

К рабочему подходит менеджер, и между ними происходит следующий диалог:

Менеджер: Контейнер 5 прибыл перед контейнером 4?

Рабочий: Нет, это невозможно

Менеджер: Так вы можете рассказать мне порядок прибытия контейнеров по их размещению?

Рабочий: Вообще говоря, нет. Например, контейнеры, находящиеся на складе, могли прибыть в порядке 3, 2, 1, 4, 9, 5 или в порядке 3, 2, 1, 9, 4, 5, или в одном из 14 других порядков.

Поскольку менеджер не хочет выглядеть намного глупее рабочего, он уходит. Вы должны помочь менеджеру и написать программу, которая по данному размещению контейнеров вычисляет все возможные порядки, в которых они могли прибывать.

Ввод

Имя входного файла — depot.in. Первая строка содержит одно целое число — количество строк, в которых содержатся контейнеры. Следующие R строк содержат информацию о строках —1, ..., R, начиная с верхней в следующем порядке. В начале каждой из этих строк находится число M — количество контейнеров в этой строке. За ним находятся M целых чисел — идентификационные номера контейнеров в строке, начиная слева. Все идентификационные номера контейнеров I удовлетворяют условию $1 \le I \le 50$. Число N контейнеров в складе удовлетворяет условию $1 \le N \le 13$.

Вывод

Имя выходного файла — depot.out. Выходной файл состоит из строк, количество которых равно числу различных вариантов прибытия контейнеров. Каждая из этих строк содержит N целых чисел идентификационных номеров контейнеров в соответствующем порядке прибывания. Все строки должны быть уникальны.

Примеры ввода и вывода

Пример 1:

depot.in	depot.out
3	3 2 1 4 9 5
3 1 4 5	3 2 1 9 4 5
2 2 9	3 4 2 1 9 5
1 3	3 9 2 1 4 5
	3 4 2 9 1 5
	3 4 9 2 1 5
	3 2 4 9 1 5
	3 2 9 4 1 5
	3 9 2 4 1 5
	3 4 2 9 5 1
	3 4 9 2 5 1
	3 2 9 4 5 1
	3 9 2 4 5 1

Пример 2:

depot.in	depot.out
2	3 1 2
2 1 2	1 3 2
1 3	

Оценивание

Если выходной файл содержит невозможный порядок или не содержит порядков вообще, вы получаете 0 баллов за соответствующий тест. В противном случае ваш счет вычисляется следующим образом. Если выходной файл содержит все возможные варианты, вы получаете 4 балла. Если выходной файл содержит половину или более возможных вариантов, и все они различны, вы получаете 2 балла. Если выходной файл содержит менее половины различных возможных вариантов, или некоторые из них встречаются более чем 1 раз, вы получаете 1 балл.

HI-TECH,
http://hi-tech.nsys.by/
E-mail: serrgio@gorki.unibel.by

НИЗКОУРОВНЕВОЕ ПРОГРАММИРОВАНИЕ

(Продолжение. Начало в №№9-11/2001)

1.10. Разборка с процедурами

#1. В разделе 1.7. #4 мы сделали глупую линейную программульку, выводящую окошки. Обещали, что в следующей главе сделаем ее менее "тупой", да отвлеклись почему-то на циклы и стек. То есть мы-то знаем, ПОЧЕМУ, но вот вам об этом — не скажем! Догадайтесь сами. Итак, поехали...

Шаг первый. Внимательно посмотрев на "линейную" прогу из 1.7. #4 и прочитав "условие задачи" из главы 1.7. #1, вы обязаны возмутиться — зачем мы использовали команду MOV, если и ежу понятно, что отличия следующего окошка от предыдущего можно выразить более лаконично: BH=BH+10, CH=CH+1, CL=CL+1, DH=DH-1, DL=DL-1? И не нужно напрягать мозги, подсчитывая новое значение регистра вручную.

Если вы так подумали, то оказались совершенно правы! Программу из #4 запросто можно было представить в таком вот виде:

```
:0100 XOR AL,AL ;окошко первое
:0102 MOV BH,10
:0104 MOV CH,05
:0106 MOV CL,10
:0108 MOV DH,10
:010A MOV DL,3E
:010C MOV AH,06
:010E INT 10
:0110 ADD BH,10 ;окошко второе
:0113 ADD CH,01
:0116 ADD CL,01
:0119 SUB DH,01
:011C SUB DL,01
:011F INT 10
:0121 ADD BH,10 ;окошко третье
:0124 ADD CH,01
:0127 ADD CL,01
:012A SUB DH,01
:012D SUB DL,01
:0130 INT 10
:0132 ADD BH,10 ;окошко четвертое
:0135 ADD CH,01
:0138 ADD CL,01
:013B SUB DH,01
:013E SUB DL,01
:0141 INT 10
:0143 ADD BH,10 ;окошко пятое
:0146 ADD CH,01
:0149 ADD CL,01
:014C SUB DH,01
:014F SUB DL,01
:0152 INT 10
:0154 INT 20    ;конец программы
```

И несмотря на то что размер ее оказался несколько большим, она тоже будет работать правильно :).

Но тут любой более или менее наблюдательный программер возмутится повторно — да что это за программа такая? В ней целых четыре раза повторяется один и тот же кусок:

```
:0143 ADD BH,10
:0146 ADD CH,01
:0149 ADD CL,01
:014C SUB DH,01
:014F SUB DL,01
:0152 INT 10
```

И знаете что? Этот наблюдательный программер будет прав! А если он еще и нехорошо выразится по поводу такого "неправильного" стиля программирования — будет прав еще больше :)

Есть такой процесс — оптимизация, одной из особенностей которой является уменьшение параметризации (параметризация — вставка процедур в место вызова или фиксация значения отдельных параметров) и развертка циклов. То, что обычно такими вещами должен заниматься компилятор, суть меняет не сильно — тем более, что ассемблер оптимизацией сам не занимается :). Но до ассемблера мы с вами еще не добрались, поэтому говорить об этом пока еще рано...

Внимательно всмотритесь в полный текст программы и в этот выделенный кусок. И помедитируйте над ним до полного просветления текущей "обстановки"...

#2. Итак, у нас есть ПОВТОРЯЮЩАЯСЯ ЧАСТЬ программы. А еще у нас есть пальцы, которым, как правило, лень набивать длинные "простыни" программного кода. Это одна из многочисленных причин, по которым и придумали такого "зверя" как ПОДПРОГРАММУ (она же — ПРОЦЕДУРА, она же — ФУНКЦИЯ). Остальные причины мы рассмотрим попозже, а вот насчет "лени" поговорим прямо сейчас.

Если мы возьмем наш "часто повторяющийся" кусок программы и допишем в конец команду RET, то получится у нас именно ПРОЦЕДУРА — во всей своей красе:

```
:011E ADD BH,10 ;"точка входа"; она же - начало "тела".
:0121 ADD CH,01
:0124 ADD CL,01
:0127 SUB DH,01
:012A SUB DL,01
:012D INT 10    ;конец "тела"
:012F RET
```

Красота ее вот в чем заключается — процедуру можно "вызвать" командой CALL :)))

Все более чем просто. Когда в программе встречается CALL с указанием АДРЕСА-НАЧАЛА-ПРОЦЕДУРЫ (в нашем случае это 011E), то компьютер "идет" по этому адресу и выполняет все команды, расположенные между "точкой входа" (включительно) и командой RET, то есть так называемое "тело" процедуры.

RET — это тоже команда, но к "телу" (адреса 11E... 12D) она не относится. Она является "ОРГАНИЗАТОРОМ" этого "тела". Процессор, встретив команду RET, возвращает управление обратно после последнего CALL (т.е. "перепрыгивает" на строчку ниже "вызвавшего" данную процедуру CALL'а)...

Короче, CALL XXXX означает — "выполнить процедуру, начинающуюся по адресу XXXX". А RET означает — "конец процедуры" и, соответственно, переход на строчку ниже вызвавшего его CALL'а.

Если же говорить более формально и строго, то процедура — это средство обобщения, когда некоторая общая последовательность действий получает "имя", и потом при необходимости ПОВТОРНОГО ИСПОЛЬЗОВАНИЯ данного кода к нему просто идет обращение по "имени" (напомним, что в отличие от языков высокого уровня и даже ассемблера, в машинном коде от имен остаются одни только адреса, а язык, принимаемый DEBUG, является промежуточным между машинным кодом и ассемблером). Более того, следующим логическим шагом после обобщения является параметризация, когда некоторые части общего кода зависят от передаваемой извне информации (параметров), с чем очень хорошо знакомы программисты на языках высокого уровня. Но о параметризации и ее применении в ассемблере мы поговорим в другой раз:-).

Не ругайтесь. Мы знаем, что вы ни черта не поняли. А посему набьем в debug'е эту прогу и посмотрим, что она делает.

Те, кто читал внимательно, могут отметить, что инструкция CALL по своему действию очень похожа на инструкцию генерации прерывания INT, с той лишь разницей, что аргументом CALL является адрес процедуры, а не индекс в таблице "векторов прерываний", где и хранится адрес обработчика прерывания (той же процедуры). А для особо продвинутых отметим, что в ранних моделях процессоров от Intel при подаче запроса на обработку прерывания от внешнего устройства контроллер прерываний, помимо собственно сигнала прерывания, посылал в процессор инструкцию CALL.

#3. Кстати, вы уже поняли, почему мы называем debug "до боли любимой программой"? Нет? Неужели вы еще не полюбили это произведение программерского гения всеми фибрами своей души? Еще нет? М-да... мы в вас разочаровались.

А посему: "НАБИВАЕМ!" — злобно кричим, брызгая слюной на эргономичный коврик:

```
:0100 XOR AL,AL  ;первое окошко рисуем, как и раньше...
:0102 MOV BH,10
:0104 MOV CH,05
:0106 MOV CL,10
:0108 MOV DH,10
:010A MOV DL,3E
:010C MOV AH,06
:010E INT 10
:0110 CALL 011E  ;четыре раза вызываем подпрограмму,
:0113 CALL 011E  ;начинающуюся по адресу 011E
:0116 CALL 011E
:0119 CALL 011E
:011C INT 20     ;выход из программы...
:011E ADD BH,10  ;начало процедуры
:0121 ADD CH,01
:0124 ADD CL,01
:0127 SUB DH,01
:012A SUB DL,01
:012D INT 10
:012F RET        ;конец процедуры
```

Не правда ли, красиво получилось? А то!

Первое, что вас может смутить — это то, что команда выхода (INT 20) расположена не там, где вы привыкли, то есть не в конце программы.

Ну что я вам могу на это ответить? Концы — они-то разные бывают! Последняя строчка в листинге вовсе не означает, что последней будет выполняться именно она. И это не должно вас смущать! А если все же смущает — смотрим, как работает эта прога из-под отладчика.

Итак, до адреса 0110 вам все должно быть понятно, мы это рассматривали. Трассируем дальше...

- Что значит "трассируем"? — попросите вы напомнить.

Мысленно мы ругаем вас нехорошими словами (ну сколько раз повторять-то можно!), а вслух скажем: "Команда "T" и Enter. Команда "T" и Enter. Команда "T" и Enter..."

Команда CALL 011E по адресу 0110 говорит процессору: "Дальше мы не пойдем, пока не выполним простыню, начинающуюся по адресу 011E". И далее, естественно, следует переход на этот адрес.

Входим в тело процедуры, начиная с 011E, и выполняем команды до 012D включительно...

А теперь внимательно смотрим, на какой адрес нас "перекинет" команда RET.

На 113-й? И это правильно! По 113-му адресу у нас ка-

кая команда? Да вот опять CALL 011E!

Опять процедура с адреса 011E, опять RET[URN] на строку ниже, то есть на 116...

Так и далее, до того момента, пока следующей строчкой не окажется INT 20 — собственно, на этом и программе конец.

Ну оно и ежу понятно, что, несмотря на то что INT 20 — не в конце программы, последним выполнится именно он.

Короче, куда бы вас не посылали всяческие "столбы с указателями", конец вашего пути только один... А плутать вокруг да около этого конца вы можете сколько вам заблагорассудится...

Кстати, именно это и является одной из многочисленных тайн программинга.

#4. Тот, кто внимательно ознакомился с циклами, и на этом не остановится! Посмотрев на адреса 110...119, они вообще возьмут и возомнят себя воистину крутыми парнями! Знаете, что они напишут? А вот что (предвидим!):

```
:0100 XOR AL,AL
:0102 MOV BH,10
:0104 MOV CH,05
:0106 MOV CL,10
:0108 MOV DH,10
:010A MOV DL,3E
:010C MOV AH,06
:010E INT 10
:0110 MOV CX,0004
:0113 CALL 011A
:0116 LOOP 0113
:0118 INT 20
:011A ADD BH,10
:011D ADD CH,01
:0120 ADD CL,01
:0123 SUB DH,01
:0126 SUB DL,01
:0129 INT 10
:012B RET
```

То бишь еще и CALL в цикл при помощи MOV CX,4 и LOOP'а "закрутят". И что? А попробуйте!

Что, не "пашет"? А что надо делать, если "не пашет, а должно бы"? Правильно! Смотреть из-под отладчика!

Смотрим? Если посмотрите, то сразу же и "загвоздку" увидите — CX, использованный в качестве "счетчика" циклов, "перебивает" тот же CX, но используемый как "координаты верхнего левого угла окна". И что с этим делать прикажете?

Вот, вы и столкнулись с одной из самых больших проблем. В процессорах фирмы Intel есть только 4 регистра общего назначения (и то в большинстве случаев — специализированных). Помните, мы вам говорили об этом?

А теперь попробуйте выкрутиться из этой нехорошей ситуации с использованием стека :). Кстати, весьма "мозгопрочищающая" задачка :).

1.11. Разборки с переходами

#1. "Переходы" бывают разные. Если вы пришли в гости, а вас просто послали к черту — такой переход называется "безусловный". А нежели вам сказали: "Если без пива — то иди к черту, а если с пивом — тогда проходи", — то это уже "условный" переход.

Соответственно, для успешного перехода необходимо указать: ПРИ КАКОМ УСЛОВИИ выполнить переход, КУДА ПЕРЕЙТИ, ну и, наконец, сам пинок под зад нужно СДЕЛАТЬ, чтобы переход "гостя" в заданном направлении все-таки "состоялся".

Безусловный переход у нас "делает" мнемоническая команда JMP, после которой следует указать адрес, на который "компьютер" должен пойти "на" ;). В данном случае УСЛОВИЕМ у нас будет "при любых обстоятельствах": хоть пустой, хоть с пивом, хоть с ... — все равно. Когда рисовали окошки, вы уже использовали эту команду для создания "спецдефекта". Если кто еще не понял, что делает эта команда — к нему (см. 1.7 #4) и отсылаю. Сделайте "спецдефект" и посмотрите на него под отладчиком. Когда до вас дойдет, почему мы там не предусмотрели выхода (INT 20h) — можете переходить к п.2 текущей главы.

#2. Условный переход у нас организуется в два шага. На первом шаге мы вычисляем условие ("принес ли пиво?"), на втором шаге "посылаем" или не "посылаем" — в зависимости от результатов вычислений. Можно привести такую аналогию — на первом шаге два груза кладутся на весы, сравнивающие их массу. Соответственно, возможны только три их положения: наклон влево (груз в левой чашке тяжелее), наклон вправо (груз в правой чашке тяжелее) и равновесие. На втором шаге мы предпринимаем действия в зависимости от положения весов.

Например, на первом шаге можно использовать как "аптекарские весы" инструкцию CMP, которой обязательно нужно указать, ЧТО и С ЧЕМ она будет сравнивать.

Пишем, например,

```
CMP AX,BX
```

В зависимости от значений регистров у нас возможны следующие состояния: "наклон влево" (AX > BX), "наклон вправо" (AX < BX) и "равновесие" (AX = BX). Таким образом ВЫЧИСЛЕНИЕ УСЛОВИЯ у нас уже организовано :). Только условие не бинарное, а есть еще и "серединный вариант" (и даже несколько других!). Это нормально. Это для того сделано, чтобы мы могли выражения типа "больше-или-равно", "меньше-или-равно" да и просто "равно" в своих программах использовать...

Итак, УСЛОВИЕ есть. Теперь решаем, что нам делать при том или ином условии. Вот далеко не полный список возможных "прыг-скоков":

```
JE  - переход если равно;
JNE - переход если не равно;
JA  - переход, если больше;
JAE - переход, если больше или равно;
JB  - переход, если меньше;
JBE - переход, если меньше или равно...
и т.д.
```

Естественно, что после мнемоники ("прыгнуть, если") должен стоять АДРЕС, куда нужно "прыгнуть", если условие соблюдено. Если же условие не соблюдено, то прыжок не происходит, и выполняется нижеследующая строка программы.

Задание на медитирование — зрительно представьте себе "весы правосудия". И побросайте на их чашки разную шестнадцатеричную дрянь в различных "пропорциях" и "комбинациях". Просветлиться вы должны следующим образом — в какую бы сторону эти ваши "весы" ни склонялись, вы все равно заставите систему работать так, как ЭТО вам угодно! "Весы" — они только констатируют факт. А вот "приговор" выносят судьи. Гы... и пусть после этого только кто-нибудь скажет, что программерам чужда политика — дело, как известно, весьма грязное...

А о чем это мы? Ах да, переходы...

#3. Продолжим программировать, что ли? Напишем что-нибудь красивое и неизменно тупое? С использованием условных и безусловных переходов?

Поехали! Слабаем мы сейчас что-то наподобие графического редактора :)). Не верите?

У-у-у... Сложная задачка! Если въедете, что да как — значит, молодцы! Значит, разобрались-таки с dzebagom!

Значит, подключились-таки к программерскому эгрегору и более или менее привели в порядок свои мозги :)... А это сложная штука, мы вам скажем — мозги в порядок приводить! Особенно когда есть куча инструментов, которые "порядок в коде" сами как бы наводят :).

Думаете, вы по нашим текстам программировать учитесь? По-настоящему программировать мы еще не начали! Все, чем мы пока занимаемся — это приводим в порядок свои мозги и тренируемся на кнопки клавиатуры нажимать :)). А вот ско-о-оро НАЧНЕМ... тогда "прощай, здоровье" будет настоящее!

Итак, сначала рассмотрим прерывания, которые в нашем "графическом редакторе" будут использоваться. Их три штуки, и все — BIOS'вские:

```
mov AH,00 ;функция 0 прерывания 10h устанавливает "режим видео"
mov AL,04 ;"на входе" (в регистре AL) - номер режима .
int 10h   ;если AL=4, то устанавливается цветной 320x200
          ;графический режим
```

Подробности об этом прерывании ищите в "справочных системах". У нас на сайте их аж четыре штуки... ;). О "номерах" других режимов — см. там же.

Бумажной литературы, где имеются более или менее нормальные описания прерываний, мы нашли немного, целых две:

- П.Нортон. Персональный компьютер фирмы IBM и операционная систма MS-DOS. — Радио и связь, 1992. ББК 32.973. (в русском переводе с десятичной нотацией функций — просто мурашки бегут по коже!);

- Ж.К.Голенкова, А.В.Заболоцкий, М.Л.Мархасин. Руководство по архитектуре IBM PC AT. — Консул, 1992. УДК 681. (там только BIOS'овские прерывания описаны).

Сорри, но больше книжек никаких не знаем, а вот с Инета настоятельно рекомендуем скачать RBIL, то бишь Ralf Brown Interrupt List, который находится по http://pobox.com/~ralf/. Это наиболее полное из известных нам описание прерываний.

Мы ж и говорим — без справочников в низкоуровневом программировании ну ни шагу ступить нельзя!

```
mov CX,64h  ; CX и DX - координаты точки
mov DX,64h
mov AH,0Ch  ; функция 0Ch (в регистре AH!)
            ; прерывания 10h рисует точку.
mov AL,1Bh  ;в AL - "код цвета" этой точки.
int 10h
```

Опять-таки — подробности о координатах и "кодах" цвета ищите сами!! Благо, знаете, где искать.

```
mov AH,00   ;функция 0 прерывания 16h
int 16h     ;"читать код нажатой клавиши"
```

Эта функция считывает код сканирования и код символа (клавиши на клавиатуре и соответствующий ей ASCII-код) из буфера клавиатуры (есть такой). Если в буфере ничего нет — она ждет, пока там что-нибуть появится. То есть ЖДЕТ, чтобы вы нажали на какую-нибудь клавишу, код которой будет занесен в регистр AX. Причем в AL — "символ", а вот в AH — так называемый "код сканирования"...

Кодами вы пока голову не забивайте. Достаточно знать, что после нажатия клавиши Up в AH "попадет" значение 48h, Down — 50h, Left — 4Bh, Right — 4Dh.

Как работает последний кусок кода, обязательно проверьте под отладчиком, это полезно :).

#4. И лезем, лезем в наш горячо любимый DZEBUG, дабы набить там драгоценные строчки машинного мнемонического никому-кроме-вас-непонятного кода!

```
-a
:0100 MOV AH,00     ;устанавливаем графический режим
:0102 MOV AL,04
:0104 INT 10
:0106 MOV CX,0064   ;координаты Первой Точки
:0109 MOV DX,0064
:010C MOV AH,0C     ;рисуем точку!
:010E MOV AL,1B
:0110 INT 10
:0112 MOV AH,00     ;ждем нажатия на клавишу
:0114 INT 16
:0116 CMP AH,4B     ;а не нажат ли у нас Left?
:0119 JE 012A       ;если да - то "прыг"!
                    ;если нет - то следующая строчка
:011B CMP AH,4D     ;а не нажат ли у нас Right?
:011E JE 012D
:0120 CMP AH,48     ;а не нажат ли у нас Up?
:0123 JE 0130
:0125 CMP AH,50     ;а не нажат ли у нас Down?
:0128 JE 0133
:012A DEC CX        ;задаем новые координаты, в зависимости
:012B JMP 010C      ;от нажатой клавиши - и скок в начало!
:012D INC CX
:012E JMP 010C
:0130 DEC DX
:0131 JMP 010C
:0133 INC DX
:0134 JMP 010C
```

Тут один из авторов (не будем показывать пальцем) такую шпильку вставил:

"Не хватает проверки и выхода (со сбросом видеорежима!) по Esc — такие действия должны быть обязательным атрибутом, а не домашним заданием."

Совершенно верная шпилька, товарищи! Но все равно — пусть это будет домашним заданием.

Если вы все ввели правильно — должно заработать! Полюбуйтесь плодами своей медитации... Красиво?

#5. А сейчас мы это все дело прокомментируем:

- адреса 100...200 — устанавливаем графический режим, указываем функцию (100), указываем номер режима (102) и вызываем прерывание (104);

- адреса 106...109 — инициализируем координаты ***первой*** точки. Координаты последующих точек будут определяться "динамически" — в зависимости от нажатой клавиши;

- адреса 10C...110 — рисуем точку. Первый раз — в координатах, инициализированных командами по адресам 106 и 109. Все последующие разы — по координатам, "инкрементированным" или "декрементированным" (во словеса!) по адресам: 12A, 12D, 130 и 133;

- адреса 112...114 — ждем нажатия на клавишу;

- адреса 116...128 — "щемим" нужные нам клавиши. "Взвешиваем". На каждую из курсорных клавиш по адресам 12A...134 приготовлены "обработчики". Если найдена "нужная клавиша", то делаем прыг на "обработчик" этой клавиши;

-адреса 12A...134 — в этом блоке определяется, что делать с координатами следующей точки. После чего — прыжок на "рисуем точку" :).

Правда, здорово получилось?

#6. Как говорил Петя Нортон (настоящая фамилия Норкин — см. http://www.assembler.ru/13_arts/13000500.htm). Совершенствовать программу можно бесконечно. Вопрос здравого смысла — "КОГДА целесообразно остановиться".

А вы думайте: как сократить размер этой программы (может, что-то там лишнее?), как сделать нормальный выход..? И вообще... побольше думайте. Над "рисовалкой" еще работать и работать! Заготовку получили? Дополняйте и оптимизируйте!

И да пребудет с вами сила, братья по безумию!

(Продолжение следует)

А.ШАКИРИН,
г.Минск, БАТУ, каф.ВТ.

Программирование алгоритмов с использованием массивов

Цель этой статьи — освоить применение компонента StringGrid и создать приложение, в котором используются массивы.

1. Пример создания приложения

Необходимо создать Windows-приложение для вычисления вектора $x=\{x_1,x_2,\ldots,x_m\}$, равного p-й строке матрицы $A=\{a_{ij}\}(x_j=a_{pj}, j=1,2,\ldots,m)$; и вектора $y=\{y_1,y_2,\ldots,y_n\}$, равного q-му столбцу матрицы $A=\{a_{ij}\}(y_i=a_{iq}, i=1,2,\ldots,n)$ $(n\le 6, m\le 8)$. В панели интерфейса следует предусмотреть возможность управления размерностью массивов.

Один из возможных вариантов панели интерфейса создаваемого приложения показан на рисунке.

1.1. Размещение компонентов на Форме

При работе с массивами ввод и вывод информации на экран удобно организовывать с помощью компонента StringGrid.

Компонент **StringGrid** используется для отображения информации в виде таблицы. Таблица содержит две зоны — фиксированную и рабочую. Фиксированная зона служит для вывода наименований строк и столбцов рабочей зоны и управления их размерами с помощью мыши. Она выделена другим цветом, и в нее запрещен ввод информации с клавиатуры. Количество строк и столбцов фиксированной зоны устанавливается в свойствах FixedRows и FixedCols соответственно.

Рабочая зона содержит RowCount строк и ColCount столбцов информации, которую можно изменять как программно, так и с помощью мыши или клавиатуры.

Доступ к информации в программе осуществляется с помощью свойства Cells[ACol, ARow: integer]: string, где Acol — номер столбца, а ARow — номер строки таблицы, причем нумерация начинается с нуля.

Пиктограмма компонента StringGrid находится на странице Additional Палитры Компонентов. Так как в нашем случае для всех компонентов StringGrid фиксированная зона не используется, в Инспекторе Объектов значения свойств FixedCols и FixedRows следует установить равными 0. Установим предельные значения количества строк n и столбцов m для компонента StringGrid1 — ColCount=8, а RowCount=6 (восемь столбцов и шесть строк). Для компонента StringGrid2 — ColCount=1, RowCount=8, а для компонента StringGrid3 — ColCount=1, RowCount=6.

По умолчанию в компонент StringGrid запрещен ввод информации с клавиатуры, поэтому для компонента StringGrid1 необходимо в Инспекторе Объектов дважды щелкнуть мышью на символе "+" свойства +Options, и в открывшемся списке опций установить значение goEditing в True.

Для удобства работы с компонентами SpinEdit установим для компонента SpinEdit1 следующие значения свойств — MinValue=1, MaxValue=6, а для компонента SpinEdit2 — MinValue=1, MaxValue=8.

1.2. Создание процедур обработки событий SpinEdit1Change и SpinEdit2Change

События SpinEdit1Change и SpinEdit2Change возникают при любом изменении значения в поле редактора SpinEdit1 и SpinEdit2 соответственно. Создадим процедуры обработки этих событий, в которых присвоим значения n и m, полученные из полей редакторов SpinEdit, свойствам ColCount и RowCount компонентов StringGrid. Это позволит управлять размерами таблиц StringGrid с помощью компонентов SpinEdit без дополнительных кнопок, так как изменение значений в поле редактора SpinEdit сразу приведет к изменению размера таблиц StringGrid.

Дважды щелкните мышью на компоненте SpinEdit1 — курсор установится в тексте процедуры-обработчика события SpinEdit1Change: **procedure TForm1.SpinEdit1Change(Sender: TObject)**. Внимательно наберите операторы этой процедуры, используя текст модуля UnMas. Аналогичным образом создайте процедуру-обработчик события SpinEdit2Change: **procedure TForm1.SpinEdit2Change(Sender: TObject).**

1.3. Текст модуля UnMas

```
Unit UnMas;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls,
    Forms, Dialogs, StdCtrls, Spin, Grids;
type
  TForm1 = class(TForm)
    Label1: TLabel;
    SpinEdit1: TSpinEdit;
    SpinEdit2: TSpinEdit;
    Label8: TLabel;
    StringGrid1: TStringGrid;
    StringGrid2: TStringGrid;
    StringGrid3: TStringGrid;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    SpinEdit3: TSpinEdit;
    SpinEdit4: TSpinEdit;
    Label6: TLabel;
    Label7: TLabel;
    Button1: TButton;
   procedure FormCreate(Sender: TObject);
   procedure SpinEdit1Change(Sender: TObject);
```

```
    procedure SpinEdit2Change(Sender: TObject);
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form1: TForm1;
implementation
{$R *.DFM}
var
 A:array[1..6,1..8] of extended; // объявление двумерного массива A
 X:array[1..8] of extended;      // объявление одномерного массива X
 Y:array[1..6] of extended;      // объявление одномерного массива Y
 n,m,p,q:integer;                // объявление глобальных переменных
procedure TForm1.FormCreate(Sender: TObject);
begin
 SpinEdit1.Text:='4';              // начальное значение n
 SpinEdit2.Text:='6';              // начальное значение m
 SpinEdit3.Text:='2';              // начальное значение p
 SpinEdit4.Text:='3';              // начальное значение q
 StringGrid1.RowCount:=4;     // количество строк массива A
 StringGrid1.ColCount:=6;     // количество столбцов массива A
 StringGrid2.RowCount:=6;     // количество строк массива X
 StringGrid3.RowCount:=4;     // количество строк массива Y
end;
procedure TForm1.SpinEdit1Change(Sender: TObject);
begin
 n:=StrToInt(SpinEdit1.Text); // n присваивается содержимое
                              // поля редактора
 StringGrid1.RowCount:=n;     // устанавливается количество
                              // строк массива A
 StringGrid3.RowCount:=n;     // устанавливается количество
                              // строк массива Y
end;
procedure TForm1.SpinEdit2Change(Sender: TObject);
begin
 m:=StrToInt(SpinEdit2.Text); // m присваивается содержимое
                              // поля редактора
 StringGrid1.ColCount:=m;     // устанавливается количество
                              // столбцов массива A
 StringGrid2.RowCount:=m;     // устанавливается количество
                              // строк массива X
end;
procedure TForm1.Button1Click(Sender: TObject);
var
 i,j:integer;                 // объявление локальных переменных
begin
 n:=StrToInt(SpinEdit1.Text);
 StringGrid1.RowCount:=n;
 StringGrid3.RowCount:=n;
 m:=StrToInt(SpinEdit2.Text);
 StringGrid1.ColCount:=m;
 StringGrid2.RowCount:=m;
 p:=StrToInt(SpinEdit3.Text);
 q:=StrToInt(SpinEdit4.Text);
// ввод значений из таблицы в массив A
 for i:=1 to n do
  for j:=1 to m do
   A[i,j]:=StrToFloat(StringGrid1.Cells[j-1,i-1]);
 for j:=1 to m do     // формирование массива X и вывод его
                      // значений в таблицу
  begin
    X[j]:=A[p,j];
    StringGrid2.Cells[0,j-1]:=FloatToStrF(X[j],ffFixed,3,1);

    end;
 for i:=1 to n do     // формирование массива Y
                      // и вывод его значений в таблицу
  begin
    Y[i]:=A[i,q];
    StringGrid3.Cells[0,i-1]:=FloatToStrF(Y[i],ffFixed,3,1);
  end;
 end;
end.
```

1.4. Работа с приложением

Теперь можно запустить созданное приложение, занести числовые значения в элементы матрицы A и убедиться в том, что приложение функционирует.

2. Индивидуальные задания

Выберите себе индивидуальное задание из приведенных ниже. В соответствии с выбранным заданием создайте приложение и протестируйте его работу.

1. Задана целочисленная матрица *A* размером *NxM*. Получить массив *B*, присвоив его *k*-му элементу значение *0*, если все элементы *k*-го столбца матрицы нулевые, и значение *1* в противном случае *(k=1,2,...,M)*.

2. Задана целочисленная матрица *A* размером *NxM*. Получить массив *B*, присвоив его *k*-му элементу значение *1*, если элементы *k*-й строки матрицы упорядочены по убыванию, и значение *0* в противном случае *(k=1,2,...,N)*.

3. Задана целочисленная матрица *A* размером *NxM*. Получить массив *B*, присвоив его *k*-му элементу значение *1*, если *k*-я строка матрицы симметрична, и значение *0* в противном случае *(k=1,2,...,N)*..

4. Задана целочисленная матрица размером *NxM*. Определить *k* — количество "особых" элементов матрицы, считая элемент "особым", если он больше суммы остальных элементов своего столбца.

5. Задана целочисленная матрица размером *NxM*. Определить *k* — количество "особых" элементов матрицы, считая элемент "особым", если в его строке слева от него находятся элементы, меньшие его, а справа — большие.

6. Задана символьная матрица размером *NxM*. Определить *k* — количество различных элементов матрицы (т.е. повторяющиеся элементы считать один раз).

7. Дана вещественная матрица размером *NxM*. Упорядочить ее строки по неубыванию их первых элементов.

8. Дана вещественная матрица размером *NxM*. Упорядочить ее строки по неубыванию суммы их элементов.

9. Дана вещественная матрица размером *NxM*. Упорядочить ее строки по неубыванию их наибольших элементов.

10. Определить, является ли заданная квадратная матрица n-го порядка симметричной относительно побочной диагонали.

11. Для заданной целой матрицы размером *NxM* вывести на экран все ее седловые точки. Элемент матрицы называется седловой точкой, если он является наименьшим в своей строке и одновременно наибольшим в своем столбце или, наоборот, является наибольшим в своей строке и наименьшим в своем столбце.

12. В матрице *n*-го порядка переставить строки так, чтобы на главной диагонали матрицы были расположены элементы, наибольшие по абсолютной величине.

Литература

1. В.В.Феофанов. Delphi 3. Учебный курс. — М.: Нолидж, 1981.

2. Э.Возневич. Delphi. Освой самостоятельно. — М.: Восточная книжная компания, 1996.

3. Дж.Матчо, Д.Р.Фолкнер. Delphi. — М.: БИНОМ, 1995.

4. М.Канту. Delphi 2 для Windows 95/NT. — М.: ООО "Малип", 1997.

ОБ ОДНОМ МЕТОДЕ КРАСИВОГО ВЫВОДА ТЕКСТА

Часть 2

О.ЧЕРЕДНИЧЕНКО,
Днепропетровская обл.
E-mail: olegka.cher@newmail.ru
FidoNet: 2:464/86.62

В первой части статьи [1] было рассмотрено построение статических текстов различного размера и начертания с примерами на ассемблере Z80, ориентированными на компьютер "ZX-Spectrum". Здесь же я покажу вам малую толику из всего многообразия возможностей рисования текстов указанным методом на языке Turbo Pascal для PC, а также изложу некоторые соображения по применению и развитию кода.

Для использования в собственных программах всей красоты и шарма графических текстов вам необходим модуль NiceType.

```
Unit NiceType; {
+-------------------------------------------------------------+
¦ Модуль предназначен для графического вывода шрифтами BIOS   ¦
¦       текстов различного размера, стиля и начертания        ¦
¦   Code (c) 2001 by Oleg N. Cher [aka ExCite o'Myth Corp.]   ¦
¦                  e-mail:  olegka.cher@newmail.ru            ¦
¦                  FidoNet: 2:464/86.62  ZXNet: 500:562/1.16  ¦
+-------------------------------------------------------------+
)   (* ---------------------------------------- *)     Interface
uses Graph;                   { Прототипы BIOS-шрифтов: }
type FontAspect = (int1Fh,    { текущий для INT 1Fh     }
                   int43h,    { текущий для INT 43h     }
                   _8x14,     { ROM 8x14                }
                   _8x8d,     { ROM 8x8 двойной         }
                   _8x8h,     { -'- (старшие 128 симв.) }
                   _9x14,     { ROM alt. 9x14 (EGA, VGA)}
                   _8x16);    { ROM 8x16 (MCGA, VGA)    }
const nTypColor: word = White;
procedure nTypFont  (Ft:FontAspect);
procedure mSprite (sX,sY:integer; sColor:word; sImage:string);
procedure nType (x,y,Dx,Dy,lp,hp,lcr,hcr:real;img,txt:string);
procedure nTypeC (x,y,Dx,Dy,lp,hp,lcr,hcr:real;img,txt:string;
                  C:word);
{       x, y - координаты левого угла первого символа текста;
        Dx   - расстояние между буквами по горизонтали;
        Dy   - расстояние между буквами по вертикали;
        lp   - наклон литер по горизонтали:
                    0 - прямо, +N - влево, -N - вправо;
        hp   - расстояние между спрайтами букв вертикальное;
        lcr  - расстояние между спрайтами букв горизонтальное;
        hcr  - наклон литер по вертикали:
                    0 - прямо, +N - вниз, -N - вверх;
        img  - образ спрайта
        text - непосредственно текстовая строка
        C    - глобальный цвет текста  }
    (* ---------------------------------- *)     Implementation
var nTypFt, FtHgt : byte;
procedure nTypFont (Ft : FontAspect);
    begin
        nTypFt := Ord (Ft); case Ft of
            int1Fh: begin FtHgt :=  8 end;
            int43h: begin FtHgt := 16 end;
            _8x14 : begin FtHgt := 14 end;
            _8x8d : begin FtHgt :=  8 end;
            _8x8h : begin FtHgt :=  8 end;
            _9x14 : begin FtHgt := 16 end; {?}
            _8x16 : begin FtHgt := 16 end end
    end;
type     byteptr = ^byte;
function GetSymbPtr (Symbol : char) : byteptr; assembler;
    asm  push bp
         mov  ax,$1130
         mov  bh,nTypFt
         int  10h
         mov  dx,bp
         pop  bp
         mov  al,Symbol
         xor  ah,ah
         mov  cl,FtHgt
         mul  cl
         add  ax,dx
         mov  dx,es
    end;
procedure mSprite (sX,sY:integer; sColor:word; sImage:string);
var i, j , bitptr, scntr, sprlen : byte; tX : integer;
  begin
    tX := sX; sprlen := (length (sImage) - 1) div 16 + 1;
    scntr := sprlen;
    for i := 1 to length (sImage) do
      begin
        bitptr := byte (sImage [i]);
        for j:= 0 to 7 do
          begin
            if (bitptr and $80) = $80
                then PutPixel (tX,sY,sColor);
            bitptr := bitptr+bitptr; inc (tX)
          end;
        dec (scntr);
        if scntr = 0
            then begin scntr := sprlen; tX := sX; inc (sY) end
      end
  end;
procedure nType (x,y,Dx,Dy,lp,hp,lcr,hcr:real;img,txt:string);
var i, j, n, str8pix : byte; fontptr : byteptr;
    arjx, arjy, xt, yt: real;
  begin
    for n := 1 to length (txt) do
      begin
        fontptr := GetSymbPtr (txt [n]); arjx := x; arjy := y;
        for i := 1 to FtHgt do
          begin
            str8pix := fontptr^; inc (fontptr);
            xt := x; yt := y;
            for j := 0 to 7 do
              begin
                if (str8pix and $80) = $80
                    then mSprite (round(xt),
                                  round(yt), nTypColor, img);
                str8pix := str8pix+str8pix;
                xt := xt+lcr; yt := yt+hcr;
              end;
            x := x + lp; y := y + hp
          end;
        x := arjx + Dx; y := arjy + Dy
      end
  end;
procedure nTypeC (x,y,Dx,Dy,lp,hp,lcr,hcr:real;img,txt:string;
                  C:word);
  begin nTypColor := C;
        nType (x,y,Dx,Dy,lp,hp,lcr,hcr,img,txt) end;
    begin nTypeFont (int43h) end.
```

Ох, эти строчки, строченьки, что набраны петитом...

Для воплощения своих идей в жизнь модуль предоставляет вам на выбор фонты BIOS следующих размеров: CGA 8x8, EGA 8x14 и VGA 8x16. Кроме того, можно применить с NiceType пользовательские таблицы образов шрифтов указанных выше стандартных размеров.

Впрочем, давайте по порядку.

Первый в прототипе фонт, на который указывет вектор прерывания 1Fh, и является такой таблицей размером 8x8. Перед использованием таблицы (в ней можно хранить национальные символы, псевдографику, текстуры) необходимо загрузить шрифт в память и передать указатель на него вектору. Таблица требует для своего размещения 128 * 8 байтов памяти — 128 символов * 8 байтов на каждую литеру.

Следующий фонт — 8x16 (на него ссылается вектор 43h) — имеет те же свойства. Шрифты _8x14 и _8x16 — системные BIOS-шрифты со стандартным расширением второй половины таблицы ("кракозябры" и псевдографика). Фонты _8x8d и _8x8h — соответственно, основная и расширенная таблицы ASCII.

Можно предположить, что при наличии "правильного" русификатора, не меняя ни единой строчки кода, с помощью процедуры **nType** удастся вывести и русский текст (для экономии памяти скорее всего русифицирован только шрифт int43h). Но согласитесь, что рассчитывать на это, как минимум, глупо, да и не является в программировании хорошим тоном. Советую приспособить модуль для работы с некоторыми подгружаемыми фонтами, что сулит неплохие возможности в области текстостроения. Это могут быть фонты, подготовленные самостоятельно в каком-нибудь редакторе шрифтов или же готовые растровые фонты, например, .chr от Windows, только сначала как следует разберитесь в их формате. Попутно придется также несколько изменить текст программы — добавить в тип **FontAspect** не-BIOS-прототипы для различных пользовательских шрифтов. Изменения также коснутся процедуры **GetSymbPtr**, вычисляющей сегмент и смещение необходимого символа в памяти, и процедуры **nType**, которая почему-то считает, что все растровые шрифты в мире имеют ширину 8 пикселов.

Пару слов о формате монохромного спрайта для **mSprite**. Обратите внимание, мы нигде явно не задаем высоту и ширину спрайта, а вычисляется она по количеству байтов. Этот формат таков, что спрайты всегда квадратные, независимо от того, сколько байтов отпущено под их представление. Если байтов восемь или меньше, то спрайт — это квадратик 8x8; если больше восьми — 16x16, если больше 32 — 24x24 и т.д.

Образ спрайта задается в процедуре символьной строкой, каждый бит значений которой управляет светимостью пиксела. При размере спрайта до 8x8, первый байт определяет 8 пикселов первой строки на экране, второй байт — 8 пикселов второй строки и т. д. В случае размера спрайта от 8x8 до 16x16, первые два байта определяют 16 пикселов первой строки, следующие — 16 пикселов второй строки и т. д.

"Я хочу, чтобы картинка...". Текст в динамике

Итак, со статикой более или менее понятно. А в качестве базиса для собственных экспериментов воспользуйтесь примерами из NiceDEMO.

Эффекты, вошедшие в минидемо: гигантская надпись, выложенная из спрайтов-снежинок, "раскладывающаяся" сверху и оставляющая "снежный" след; разноцветная "тень" у символов; конструирование литер большим "рубленым" шрифтом с помощью цикла; вращение надписи подобно часовой стрелке и вывод ее по линеечке под разными углами; "декоративные" тексты различных размеров, оттененные, штриховые, текстурные; многоцветный вращающийся знак вопроса; скроллинг текста бегущей строкой и рисование текста чередующимися спрайтами. Также заложена возможность проецирования на экран плавающих в "трехмерном" пространстве плоскостей с лежащими на них буковками, размер которых, расстояние между ними и поверхность со стороны наблюдателя меняются динамически, впрочем, тут все очень упрощено; и, наконец, слетающиеся отовсюду точки, которые образуют требуемую надпись.

```
program NiceTypeDemo;                          { This easy DEMO coded }
uses CRT, Graph, NiceType;                       { by Oleg N. Cher }
const
    GD:Integer = VGA;
    GM:Integer = VGAhi;
    pi = 3.1415926535898;
    z1 = 0; z2 = 340; z3 = 160; z4 = 479; {область скроллинга}
var r : real; i, c, txp, mov : byte; p : pointer;
    size : word; text, spr : string; j : integer;
begin
    InitGraph (GD, GM, ''); nTypFont (_8x14);
for c := 1 to 32 do for i := 0 to 1 do begin
  nTypColor := i*Cyan; if c = 32 then nTypColor := LightGreen;
  (* Гигантская надпись, выложенная из спрайтиков-снежинок *)
  nType (110,-54,155,32,-10,c,18,4,
      #0#128#4#144#32#130#18#164#72#136#4#145#18#164#1#192 +
      #34#162#4#144#72#168#16#133#34#144#8#162#2#160, 'Myth')
  end; spr := #0#0#0#1#0#2#1#0#2#0#1#1#1#2#2#2;
  (* Многоцветное оттенение буковок *)   nTypFont (int43h);
  for i := 1 to 13 do nTypeC (119+i,99+i,40,16, -2,6,4,2,
      #90#165#90#165#90#165#90#165, 'Corporation', i);
  (* С помощью цикла конструируем "жирный" рубленый шрифт *)
  for i := 0 to 28 do nTypeC (230+i,10,00,57, 0,4,16,0,
      #255#255#255#255#255#255#255#255#0#0#0#0#0#0#0#0,
      'presents', White); ClearDevice;
size := ImageSize (z1,z2,z3,z4); GetMem (p, size);
  text := 'Hi folx! =)  This easy DEMO was written 4 da ' +
        'magazine ~Radioamateur~ especially..!     ';
  (* Примеры текстурных, штрихованных, оттененных текстов *)
nType (160,445,11,0,0,2,1,0,#16#17, '(C) 2001  Oleg N. Cher');
nTypeC (456,125,33,-2,2,2,4,-1,#16#17,    'DOS',        Cyan);
nTypeC (441,1,18,6,0,2,2,1,#112#112,      'WINDOWS',    Green);
nTypeC (431,19,18,6,-1,2,2,0,#10#10,      'OS/2',   LightGray);
nTypeC (463,48,20,6,1,4,2,0,#16#16#16#16,'LinuX',       Red);
nTypeC (23,34,16,3,-1,2,2,0,#128,  'Assembler...', LightCyan);
nTypeC (62,91,8,0,0,2,1,0,#16,            'BASIC',  LightRed);
nTypeC (29,125,13,0,0,4,1,0,#128#3#3#3#3,'C++',   LightBlue);
nTypeC (100,178,-13,25,-1,2,2,0,#80,      'PASCAL',   Yellow);
  for c := 0 to 1 do begin nTypColor := 4+c*10;
      nType (419+4*c,377+3*c,16,-0.9, 0,4,2,0, #128#64#128#64,
                                          'What to do?');
      nType (419+4*c,420+3*c,18,0.9,  0,4,2,0, #128#64#128#64,
                                          'How to be?')  end;
c := 1; txp := 1; mov := 0; repeat r := 0; repeat
  inc (c,2); if c = 15 then c := 1; nTypFont (int43h);
  (* Сменяющими друг друга спрайтами имитируем горение букв *)
  nTypeC (301,233,8,0,0,1,1,0,spr[c]+spr[c+1], 'O.K.',Yellow);
  nTypFont (_8x8d); nTypColor := c;
  (* Вращающийся по часовой стрелке знак вопроса *)
nType (559-45*cos(r+pi/4),279-45*sin(r+pi/4),0,0, -10*sin(r),
       10*cos(r),10*cos(r),10*sin(r), #1#2#4#8#16#32#64, '?');
nTypColor := LightGreen; for i := 0 to 1 do begin
  (* Вывод надписей по линеечке под различными углами *)
  nType (cos(r-0.1)*28+313-i,sin(r-0.1)*28+240-i,
        8*cos(r),8*sin(r), -sin(r),cos(r),cos(r),sin(r),
        #1, 'Myth Corp. is the Best!'); nTypColor := Green end;
nTypeC (cos(r+pi/16-0.06)*80+313,sin(r+pi/16-0.06)*80+240,
        20*cos(r+pi/16),20*sin(r+pi/16),-2*sin(r+pi/16),
        2*cos(r+pi/16),2*cos(r+pi/16),2*sin(r+pi/16),
        #112#112, 'Cool!', Magenta);
  (* Скроллинг текста диагональной бегущей строкой *)
inc (mov); if mov = 9 then begin mov := 0; nTypFont (_8x14);
  nTypeC (121,406,0,0, -2,4,4,2, #90#165#90#165#90#165#90#165,
        text[txp],Cyan); if txp = length (text) then txp := 0;
  inc (txp)  end;
GetImage (z1+3,z2+2,z3,z4,p^); PutImage (z1,z2,p^,0);
  { nTypFont (_8x8d); )  (* "Тасим" знак вопроса *)
  nTypeC (559-45*cos(r+pi/4),279-45*sin(r+pi/4),10*cos(r),
  10*sin(r),-10*sin(r),10*cos(r),10*cos(r),10*sin(r),
  #1#2#4#8#16#32#64, '?',0); r := r + pi/8;
  nTypeC (301,233,8,0,0,1,1,0,spr[c]+spr[c+1], 'O.K.',Red)
  until r >= 2*pi  until KeyPressed; FreeMem (p, size);
ClearDevice; SetFillStyle (8, 2); Bar (0,0,GetMaxX,GetMaxY);
  (* Плавающие в "трехмерном" пространстве литеры текста *)
  nTypFont (_8x8d); r := 60; for j := -200 to 200 do begin
nTypeC (200+j,97-j,51,16, 0,r,r,-1, #28#30#119#99#119#30#28,
        'Myth', LightGreen); Delay(100);
nTypeC (200+j,97-j,51,16, 0,r,r,-1, #28#30#119#99#119#30#28,
        'Myth', Green); r := r - 0.5  end; nTypFont (int43h);
  (* Точки слетаются из пространства и образуют надпись *)
ClearDevice; for c := 30 downto 2 do begin
  nTypeC (170,190,18,1,0, c,c,0,#1#2, 'Satisfaction!',
        LightGreen); delay (400);
  nTypeC (170,190,18,1,0, c,c,0,#1#2, 'Satisfaction!', Green)
end;
  nTypeC (170,190,18,1,0, c,c,0,#1#2, 'Satisfaction!', White);
  Delay (999); ReadKey; CloseGraph
end.
```

Несколько идей по построению динамических текстовых эффектов. Можно:

- менять размер символов в динамике. Увеличивающиеся и уменьшающиеся буковки — здесь нам очень помогут вещественные числа. Например, такой эффект— из-за края экрана появляется махонькая надпись, вылетает в центр и начинает "распухать". Тут лучше представить символ сначала спрайтом-точкой, а потом, по мере выращивания литер, придавать ему форму, все более соответствующую задуманной;

- "крутить" надпись вокруг ее центра или вокруг начального (конечного) символа. Попутно можно и размер менять;

"крутить" не саму надпись, а только ее буквы, по часовой стрелке или против. Также можно крутить литеры относительно их горизонтальной, вертикальной или средней диагональной оси;

- сначала пишется надпись, а потом "разлетается" по точкам во все стороны. Можно и наоборот — из-за экрана слетаются точки, образуя текст. Точки могут оставлять за собой "снежный" след;

- NiceType позволяет сравнительно легко описать в трехмерном пространстве плоскость с лежащими на ней литерами текста (или несколько плоскостей, на каждой из которых — литера). Несмотря на несовершенство модели, можно вращать в пространстве буковки, с тем ограничением, что "глубина" их (т.е. размерность по координате Z) будет неуклонно стремиться к нулю. Реализация эффекта приближения/удаления от наблюдателя несколько проблематична, но тоже реализуема за счет изменения спрайта символов (от мелкого к более крупному или наоборот) и всяческих расстояний — между литерами, между спрайтами. Понятно, что это десятки строк кода, но почему бы и нет?

- динамическое изменение спрайта в ходе процесса вывода. Представляются надписи из вращающихся палочек, мерцающих звезд, летящих бабочек.

И, наконец, возможны различные комбинации этих и других эффектов.

Если рисование литер в динамике производить только на одной странице (вывели текст, подождали чуть-чуть, стерли, опять вывели, но немножко другого размера, или в другом месте экрана), большинство эффектов, даже несмотря на оптимизацию **mSprite**, будут нещадно мерцать. Самый очевидный путь преодоления этого — использование страниц видеопамяти (см. стандартные процедуры библиотеки графики **SetActivePage** и **SetVisualPage** языка Pascal) или виртуального экрана.

Оптимизировать "под самую крышу"

Изначально демо ориентирована на режим VGA 640x480x16. Поскольку весь вывод спрайтов для упрощения реализован через стандартную процедуру **PutPixel** модуля Graph (заведомо неэффективное решение), путем изменения процедуры рисования точки (или графического драйвера) изменяется и видеорежим — он может быть абсолютно любым.

Можно несколько повысить эффективность работы модуля, и весь вывод осуществлять вместо **mSprite** стандартной процедурой **PutImage**, которая, разумеется, быстрее, а также написать код, который будет заниматься перекодированием аргумента-спрайта **nType** в формат BGI. Но тогда придется ограничить универсальность NiceType каким-то одним режимом (или несколькими, но только 16-цветными, или только 256-цветными). Ибо, насколько мне известно, внутренний формат спрайта BGI различен для разных графических драйверов.

Более правильным решением здесь будет применение процедуры вывода прозрачного спрайта (с возможностью наложения на имеющееся изображение), специально оптимизированной под используемый режим дисплея. Специально писать такую процедуру, очевидно, не стоит, гораздо проще ее найти готовой в библиотеках процедур.

Можно пойти еще дальше. Мне лично очень по вкусу bitblt-компиляция — идея, которую использует для значительного программного повышения графической производительности подавляющее большинство кодеров и демомейкеров на платформе "ZX-Spectrum". Я перерыл множество графических библиотек в поисках необходимой процедуры для собственных нужд с приемлемой скоростью (в 16-цветных режимах). Но среди множества процедур вывода спрайтов мне ни разу не попалась процедура, реализующая этот подход. Это позволяет предположить, что bitblt-компиляция остается почти неизвестной для большого количества программистов видеоигр на PC. А зря, ведь, по самым скромным прикидкам, она позволяет ускорить вывод спрайтов как минимум в три раза. Способ особенно хорош, когда приходится выводить несколько одинаковых спрайтов, как в нашем случае. И здесь ему нет равных.

Bitblt *(bit-Block Transfer)* — это операции объединения участков одной или нескольких битовых карт-источников и битовой карты-приемника с помощью логических операций. Bitblt-компилирование — метод, при котором процедура компилирования преобразует спрайт в машинный код, а процедура вывода спрайта просто выполняет этот код с максимальным быстродействием.

В [2] вы можете найти очень быструю процедуру для стандартного одностраничного режима 320x200x256, основанную на bitblt-компиляции, которую с минимальными изменениями можно использовать вместо **mSprite**. А как работать со страницами на низком уровне — можно посмотреть в [3]. Очень информативная статья, всячески рекомендую.

А что же дальше?

Некоторые мысли по поводу усовершенствования. Можно попытаться сделать спрайтики многоцветными, это еще более оживит картину. Можно выводить их не просто как normal put, а с помощью логических операций NOT, AND, OR, XOR, что несомненно позволит добиться интересных и красивых эффектов, если спрайты налагаются друг на друга или на что-либо уже находящееся на экране.

К сожалению, объем журнальной статьи не позволяет привести код как я себе его представляю в идеале — быстрый, с одновременно подгружаемыми несколькими пользовательскими шрифтами любого размера, процедурами для динамических манипуляций с текстами... Идей-то еще много! А процесс совершенствования подобен бесконечности, и тут для программиста самое главное — вовремя остановиться. ;)

Лучший домашний компьютер — ZX-Profi 512.

Наикомфортнейшая из оболочек — DOS Navigator.

Самая подходящая ОС для работы — OS/2.

Литература

1. О.Чередниченко. Об одном методе красивого вывода текста. Часть 1. — Радиомир. Ваш компьютер, 2001, №10, С.31.

2. М.Эйбраш. Быстродействующие адаптеры VGA и трехмерная анимация. — Журнал д-ра Добба, 1993, № 1.

3. Б.Телеснин. Адаптер VGA. Режим 256 цветов. — Монитор, 1993, №1, 2.

4. А.Тихонов. И надпись написал... — Монитор, 1993, № 3.

А.УВАРОВ,
г.Белгород.

Установи Windows сам

В данной статье речь пойдет о правильной установке Windows. Казалось бы, написано по этому поводу немало, но, как показывает практика, пользователи с завидным упорством продолжают повторять старые ошибки. Я постарался обобщить и проанализировать свой, не всегда положительный, опыт в надежде на то, что это поможет кому-нибудь сберечь деньги, время и нервные клетки, что тоже немаловажно.

Останавливаться на том, для чего требуется установка Windows, большого смысла не имеет, об этом и так много написано, да и известно, пожалуй, каждому, поэтому уделим внимание другому вопросу — как это сделать.

Прежде всего надо предостеречь пользователя от достаточно распространенной ошибки — установки "окон" поверх себя, или, что еще хуже, поверх предыдущей версии — более надежный способ нажить себе проблемы придумать трудно. Старые "глюки", как правило, сохраняются, а системные папки разрастаются раза в полтора (как минимум), кроме того, велика вероятность, что система начнет тормозить. Поэтому не пожалейте времени на тщательное удаление старой ОС и переустановку остального софта, зато потом можно будет смело забыть о проблемах с "окнами" на полгода-год (тут уж кто как пользуется).

Итак, приступим. Прежде чем бежать в магазин или к товарищу за компакт-диском с дистрибутивом, следует, трезво оценив возможности своего компьютера, а также исходя из собственных ощущений быстродействия, решить, какая версия Windows наиболее вам подходит. На старые и медленные машины (486, Pentium 100) лучше установить Windows 95 OSR2. Хотя автор как-то устанавливал на P100 16 Мб RAM Windows 98SE — не летало, конечно, но работало вполне шустро. Если компьютер более совершенный, то конечно — Windows 98SE. Владельцы новеньких "целеронов" и PIII могут попробовать Windows ME (миллениум) — сама система очень неплоха, однако могут возникнуть проблемы с драйверами к оборудованию. Короче говоря, окончательный выбор за вами.

Следующим шагом является выбор файловой системы. При использовании дисков емкостью более 1...2 Гб есть смысл перейти на файловую систему FAT32. Владельцы винчестеров малой емкости, если они не собираются использовать сжатые диски, также могут использовать FAT32, однако Win 95 на FAT32 работает медленнее и иногда сбоит. Начиная с последних версий Windows 95 OSR2, преобразовать диск FAT16 в FAT32 можно без потери данных, обратное же преобразование невозможно без форматирования диска. Так что есть над чем подумать. Владельцам 486 и P100 лучше использовать FAT16, и уже после установки, в среде Windows уплотнить диск. Чтобы там ни говорили о сжатых дисках, я некоторое время был вынужден использовать винчестер 600 Мб, сжатый до 1,27 Гб, и могу заверить, что выигрыш за счет высвобождаемого пространства с лихвой окупит мелкие неудобства, связанные с уплотнением дисков.

После того как пользователь определился с типом ОС и файловой системы, а также прикупил очередной пиратский диск (взял у товарища и т.п.), следует еще один тонкий момент. Проверьте, сможете ли вы загрузиться с этого диска. Дело в том, что очень часто такой фокус не проходит. И независимо от того, грузится компьютер с диска или нет, создайте "спасательную" дискету, а еще лучше — две. Очень рекомендую спасательную дискету EMRD [1] — на ней, кроме всех необходимых утилит, находится "Волков коммандер" (файловый менеджер типа нортоновского), что несомненно очень удобно. Очень часто бывает так, что пользователь, загрузившись с диска или дискеты, не может запустить из командной строки программу установки. Если на вашей дискете нет файлового менеджера, то потренируйтесь, перед тем как срубать "окна" и начинать установку. После того как вы почувствуете в себе силы смело вступить в единоборство с командной строкой, можно приступать к удалению ОС. Еще раз предупреждаю, не очень надейтесь на CD-диск — очень часто после удаления Windows он перестает читаться (обычно DOS его просто не видит). Вот здесь и пригодится спасательная дискета, на которой имеется драйвер CD-ROM. Лучше всего, если позволяет дисковое пространство, записать дистрибутив Windows на винчестер. Это позволит избежать ряда потенциальных проблем. Сразу приготовьте диски с драйверами оборудования.

Следующим шагом является тотальная чистка винчестера. Прежде всего на "Панели управления" выберите пункт "Установка и удаление программы" и смело удаляйте все зарегистрированные программы (работать после переустановки "окон" они вряд ли будут). После чего "пройдитесь" по папкам на диске и смело удаляйте все лишнее — в это время можно смело срубать файлы, не очень-то задумываясь, для чего они нужны. Не забудьте заглянуть в папки со шрифтами и курсорами, чтобы потом "под горячую руку" не удалить что-либо особо милое сердцу. В идеале на диске должна остаться одна папка с сохраняемыми файлами (обычно это содержимое папки "Мои документы", записи к любым игрушкам, наработки из других программ (AutoCad, например) и музыка с картинками), а также системные папки Windows, Program Files и Recycled (Корзина). Файлы из корневого каталога также следует удалить. После этого перезагрузитесь со спасательной дискеты и удаляйте оставшиеся системные папки (из Windows это делать проблематично, да и нежелательно). Также можно быстро избавиться от "лишнего", или отформатировать диск C, предварительно убедившись, что все необходимое надежно сохранено.

Теперь все готово к установке Windows. При перегрузке следует зайти в CMOS SETUP, нажав DEL или F1 во время перегрузки, и отключить защиту от вирусов. Затем, загрузив компьютер, следует запустить программу установки. После этого автоматически запускается утилита ScanDisk, которая проверяет и, в случае необходимости, устраняет ошибки на диске. Иногда после этого появляется сообщение о том, что продолжение установки невозможно — недостаточно памяти, не подходит конфигурация и т.п. Большинство этих проверок можно отключить, запустив программу установки из командной строки с соответствующим ключом. Если все прошло удачно, на экране появится окно установки. Здесь следует ввести сведения о пользователе, регистрационный код диска и т.п. Потом, при выборе типа установки, нажать "Выборочная", несмотря на то что там написано — "для опытных пользо-

вателей". Опытным пользователем может считать себя каждый, кто не просто бездумно нажимает кнопки. В появившемся окне выбора устанавливаемых компонентов выбирайте необходимые вам. Сетевые компоненты при отсутствии сети лучше не выбирать — можно нажить проблемы, мелкие, но неприятные. Если нет полного понимания, для чего нужен тот или иной компонент, оставьте его отключенным, добавить всегда можно позже.

Закончив с выбором, жмите "Далее", задав парочку вопросов, программа установки начнет копирование файлов. Теперь, по совету Билла Гейтса, можно откинуться в кресле, расслабиться и молиться, чтобы фортуна не отвернулась от вас. На мониторе появляется индикатор копирования, который показывает ход установки. Иногда индикатор копирования замирает, винчестер не "шуршит", а палец тянется к кнопке "Reset", но делать этого не стоит — минут через 10...20 копирование продолжится как ни в чем не бывало. Особенно грешит этим Windows 95. Если компьютер все-таки завис, перегрузитесь, удалите все, что уже успело установиться, и начните сначала. В режиме продолжения лучше программу установки не запускать, так как проблема обычно никуда не исчезнет, и компьютер опять зависнет.

Иногда программа установки может зависать при определении конфигурации системы в конце установки. Тогда следует вытащить из материнской платы все устройства, кроме видеоадаптера, и продолжить установку в режиме продолжения. Не стоит также надеяться на то, что программа установки правильно определит тип всех установленных устройств. Например, у меня звуковая карта C-Media стабильно определяется как "другое устройство". В этом случае необходимо зайти в "Панель управления"-"Система"-"Устройства" и удалить неверно установленное устройство из конфигурации. После этого следует зайти в "Установку оборудования" и, отказавшись от настоятельно рекомендуемого Windows автоматического поиска новых устройств, установить драйвера вручную — для более или менее сообразительного пользователя это труда не составит. Для особо капризных устройств бывает полезно удалять устройство из конфигурации в защищенном режиме (Safe mode), после чего, загрузившись в нормальном режиме, Windows начнет определять неизвестное устройство и попросит подтверждения на поиск обновленного драйвера. Здесь следует ответить "Да", и после того как "окна" завершат автоматический поиск, следует вручную указать путь к драйверам устройства. Бывает, что тип устройства определен правильно, но драйвера не установлены (например, видеокарта может быть определена как стандартный видеоадаптер). В этом случае надо выбрать "Панель управления"-"Система"-"Устройства", два раза щелкнуть мышкой на необходимом устройстве и на вкладке "Драйвер" выбрать "Обновить". Далее следует действовать так же, как было описано выше.

Наконец Windows и драйвера оборудования успешно установлены, но это еще не все. Для долгой и правильной работы Windows (безглючных "окон", к сожалению, еще не придумали) следует иметь на своем компьютере набор утилит для поддержания ОС в "спортивной" форме — стандартные утилиты с этим справляются слабо. Неплохим вариантом является установка Norton Utilities — лучше всего, если это будет русская версия. Данная программа представляет собой набор утилит, позволяющих успешно решать проблемы с дисками, системным реестром, восстанавливать файлы после удаления, оптимизировать быстродействие и т.п. Для периодической очистки диска от временных файлов следует установить какой-нибудь деинсталлятор. Неплохо справляется с этим и утилита из Norton Utilities. Однако она работает довольно медленно, а в английской версии очень трудно понять, чего же она там "просит" (пользователю задается огромное количество вопросов).

И, наконец, теперь можно заняться установкой остального софта, а также подстройкой окошек Windows под свои вкусы и потребности.

Литература

1. http://www.emergency.f2c.com

А.СЕДУНОВ,
г.Улан-Удэ, ВСГТУ.

Программирование параллельного порта ЭВМ для управления внешними процессами

Каждому инженеру-программисту известно, что параллельный порт компьютера можно использовать не только по его стандартному назначению — для связи с печатающим устройством. Параллельный порт имеет ряд необходимых сигналов, при помощи которых можно организовать управление внешними устройствами, а также передавать данные. Следует заметить, что параллельный порт намного проще программировать, нежели устройства, работающие через ISA-шину компьютера. Управлять внешними устройствами через параллельный порт намного легче и удобнее.

Рассмотрим архитектуру параллельного порта LPT1. На рис.1 приведена функциональная схема, а в таблице — разводка его сигналов. В адресном пространстве компьютера LPT1 обычно имеет стандартные адреса, которые установлены в конфигурациях SETUP:

- #378 — адрес для записи или считывания 8 битов данных;
- #37A — адрес для посылки управляющих сигналов из ПК;
- #379 — адрес для приема сигналов от управляемого устройства.

Используя порт #378, можно выставить 8-битовое значение на шину данных и тем самым подавать управляющие сигналы на внешнее устройство. Также порт #378 может использоваться для проверки только что записанного в него байта информации. Это можно пояснить на небольшом примере с использованием языка программирования Assembler для процессора Intel 8086:

```
MOV   A,#FF        ; Заносим в аккумулятор #FF,
                   ; это соответствует битовой
                   ; комбинации из единичек - 11111111.
OUT   (#378),A     ; Посылаем в порт наше значение #FF
IN    (#378),A     ; Считываем из порта значение в аккумулятор.
```

Рис. 1

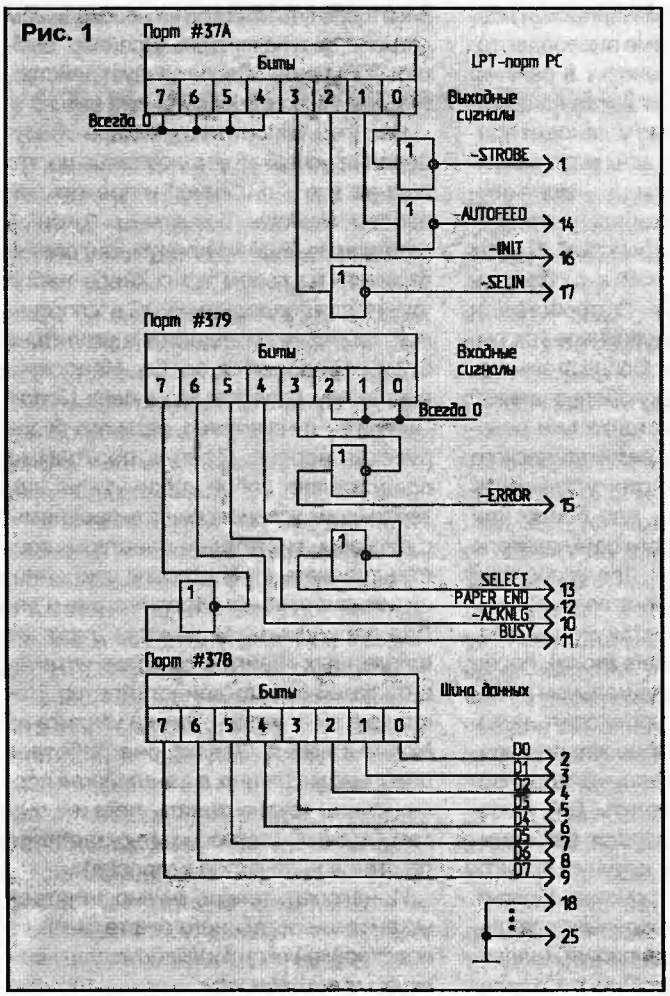


Табл. 1

Наименование сигнала	Источник сигнала	Номер контакта соединителя		Назначение сигнала
		PC	Принтер	
-AUTOFEED	PC	14	14	Низкий уровень сигнала обеспечивает перемещение бумаги на одну строку
INITIALIZE PRINTER	PC	16	31	Низкий уровень сигнала обеспечивает установку принтера в исходное состояние и очистку его буферной памяти
BUSY	Принтер	11	11	Высокий уровень сигнала указывает PC, что принтер не может принимать данные
-ERROR	Принтер	15	32	Низкий уровень сигнала указывает PC, что принтер установлен в режим "OFF LINE", нарушена его работоспособность, или отсутствует бумага
SELECT INPUT	PC	17	36	Низкий уровень сигнала разрешает работу принтера
GND	-	18...25	16, 19...30, 33	Схемная "земля"
-STROBE	PC	1	1	Принтер должен принимать данные из PC во время действия сигнала -STROBE
DATA0... DATA7	PC	2...9	2...9	Информационные разряды байта данных
-ASKNLG	Принтер	10	10	Низкий уровень сигнала указывает PC, что принтер готов к приему очередного байта
PAPER END	Принтер	12	12	Высокий уровень сигнала указывает PC на отсутствие бумаги в принтере
SELECT	Принтер	13	13	Высокий уровень сигнала указывает PC на готовность принтера к работе

Далее можно сравнить значение аккумулятора с числом 255 (#FF) и тем самым убедиться, действительно ли мы занесли в порт #378 число 255.

Биты порта #37A можно использовать в качестве управляющих сигналов для подключенного к параллельному порту устройства, а биты порта #379 — в качестве приемных сигналов, то есть чтобы анализировать отклик подключенного устройства. Поясним это на небольшом примере.

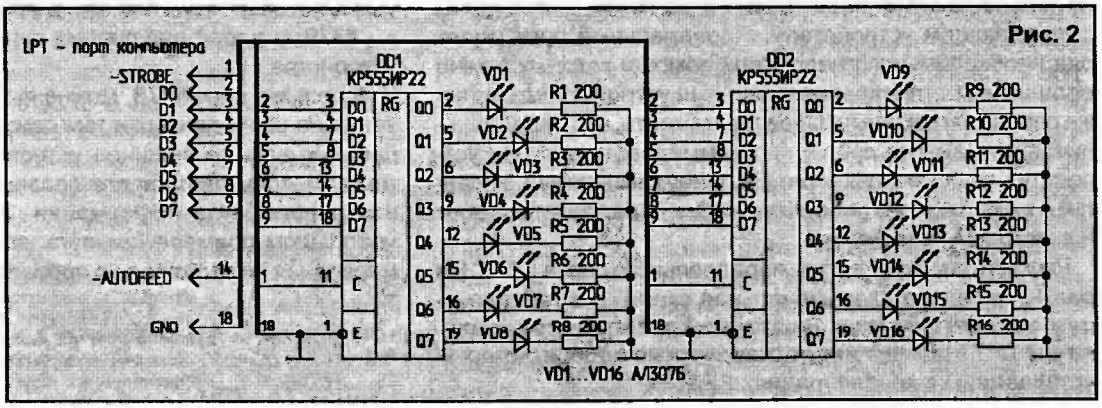
Предположим, что нам необходимо "включать" шестнадцать объектов через параллельный порт. Но как это сделать, ведь шина данных у нас 8-разрядная. Чтобы решить эту проблему, можно использовать не один порт #378, а комбинацию портов #378 и #37A. Возьмем для пояснения шестнадцать светодиодов и два регистра на ИМС КР555ИР22 (рис.2).

Микросхема DD1 выбирается инверсным сигналом -STROBE, микросхема DD2 выбирается инверсным сигналом -AUTOFEED, шина данных у них общая. Теперь можно управлять шестнадцатью светодиодами, самое главное — выполнять поочередное включение микросхем для занесения в них данных — сначала DD1, а затем DD2.

В ассемблерной мнемонике это выглядит примерно таким образом:

```
MOV     A,10          ; Заносим в аккумулятор 10.
OUT     (#37A),A      ; Открываем микросхему DD1.
MOV     A,#FF         ; Заносим в аккумулятор 255.
OUT     (#378),A      ; Зажигаем все светодиоды.
MOV     A,9           ; Заносим в аккумулятор 9.
OUT     (#37A),A      ; Открываем микросхему DD2.
MOV     A,#FF         ; Заносим в аккумулятор 255.
OUT     (#378),A      ; Зажигаем все светодиоды.
```

Рис. 2



В качестве объектов управления в данном примере используются светодиоды. Каждый бит порта #378 соответствует номеру светодиода, поэтому, если мы заносим в данный порт число 255 (#FF), то зажигаем восемь светодиодов VD1...VD8 (или VD9...VD16). Если же мы занесем число 254 (#FE), то загорятся только последние семь светодиодов, а первый будет погашен, так как число 254 в двоичном коде имеет значение 11111110.

Переключение микросхем происходит при подаче в порт #37A значения 10, в этот момент на выводе 1 LPT-порта появится логическая "1", которая разрешит работу микросхемы DD1. Если мы подадим в порт значение 9, то тем самым активизируем микросхему DD2, так как на выводе 14 LPT-порта появится логическая "1".

Надеюсь, что данный материал может быть полезен тем, кто разрабатывает различные автоматизированные устройства управления или оборудование, которым необходимо управлять в реальном масштабе времени. Если же вместо светодиодов в схеме на рис.2 подключить микросхему передатчика кода, а микросхема приемника будет находиться на противоположном конце канала передачи информации, то такое схемотехническое решение можно вполне использовать в системе передачи данных.

Литература

1. Джордейн Р. Справочник программиста персональных компьютеров типа IBM PC, XT и AT. — М.: Финансы и статистика, 1992.

2. Микропроцессорные системы контроля и управления: Сборник научных трудов. — Улан-Удэ, ВСТИ, 1989.

3. П.Гель. Как превратить персональный компьютер в измерительный комплекс / Пер. с фр.; Под ред. Г.В.Куликова. — М.: "ЛАЙТ Лтд.", 1999.

4. Измерительная техника и приборы — М.: Стандарты, 1997.

Ю.ТКАЧЕНКО,
г.Томск.
E-mail: studio-t@mail.ru

Два компьютера — общее управление

В наши дни все чаще встречаются ситуации, когда приходится одновременно работать на двух, а то и более компьютерах. У каждого из них — своя клавиатура, своя мышь, монитор и т.д. Эти устройства загромождают рабочий стол, создают путаницу, заставляют тратить время на перестановку предметов, на адаптацию к ним.

В компьютерных салонах уже давно появились в продаже красивые коробочки — переключатели с названиями: "Переключатель ручной на 2(4) положения (для мониторов и клавиатур, для серверов", "Переключатель для монитора/клавиатуры/мыши", "Data switch 4/1 ручной (Din5+ComPort9pin+VGA15pin", "Ручной переключатель (2 компьютера к 1 принт.)". Стоят они с десяток-другой и более долларов, да и не всегда и не везде есть в продаже. Однако именно эти нехитрые устройства могут быть изготовлены самостоятельно радиолюбителями любой квалификации без ущерба для семейного бюджета.

Для самостоятельной сборки подобного переключателя достаточно найти коробочку подходящего размера, желательно металлическую, и установить в ней галетный или кнопочный переключатель и разъемы. Поскольку кнопочные переключатели, хотя и более удобны в эксплуатации, но менее надежны, рекомендуется установить в корпусе переключатель галетного типа с посеребренными контактами.

Пример распайки проводов для коммутатора принтеров показан на рисунке. Проводники рекомендуется использовать минимального размера. В первую очередь это относится к коммутации мониторов, поскольку излишнее удлинение их входных кабелей может привести к появлению искажений на экране.

Особое внимание при монтаже и подключении переключателя следует обратить на общий провод. Ни при каких обстоятельствах он не должен обрываться или коммутироваться. Подключение разъемов допускается только при полностью выключенном компьютере. Последующие же переключения с пульта управления можно производить неограниченное количество раз по ходу работы на компьютере.

Подобный переключатель может быть изготовлен для одновременного или раздельного переключения нескольких мониторов, клавиатур, мышей, джойстиков, принтеров, сканеров и других периферийных устройств.

И.РОЩИН,
г.Москва,
Fido: 2:5020/689.53
ZXNet: 500:95/462.53
E-mail: asder_ffc@softhome.net
WWW: http://www.ivr.da.ru

Менеджер вызова подпрограмм из различных банков памяти

Немного теории

Адресное пространство процессора Z80 невелико — всего 64 Кб. Для доступа к большему количеству памяти в компьютере "ZX-Spectrum 128" используется страничная адресация. Оперативная память (а именно она будет нас интересовать) разбита на банки по 16 Кб (всего получается 8 банков с номерами от 0 до 7). На адреса #4000...#7FFF и #8000...#BFFF постоянно подключены банки 5 и 2 соответственно, а на адреса #C000...#FFFF может быть подключен любой из банков (рис.1).

Рис. 1

Банки 5 и 2 я буду в дальнейшем называть нижней памятью. Они всегда находятся в адресном пространстве процессора. Все остальные банки (их я буду называть верхней памятью) не обладают этим свойством. Из них может быть подключен только один.

Номер подключенного на адреса #C000...#FFFF банка памяти задается битами 0, 1 и 2 числа, выводимого в порт #7FFD. Чтение из этого порта невозможно. Поэтому, чтобы можно было узнать, какой банк подключен, надо при каждом выводе в порт запоминать выводимое значение в специальной переменной.

При выводе в порт мы не можем обратиться только к трем младшим его разрядам, не трогая остальные. Поэтому придется рассказать и о назначении других битов порта #7FFD. В третьем бите указывается номер видеостраницы: 0 — стандартная, 1 — расположенная в 7-м банке памяти. В четвертом бите — номер подключенного банка ПЗУ: 0 — BASIC 128, 1 — BASIC 48. И, наконец, вывод единицы в пятый бит приведет к отключению дополнительной памяти до аппаратного "сброса" компьютера. Остальные биты (6 и 7) в "ZX-Spectrum 128" не используются.

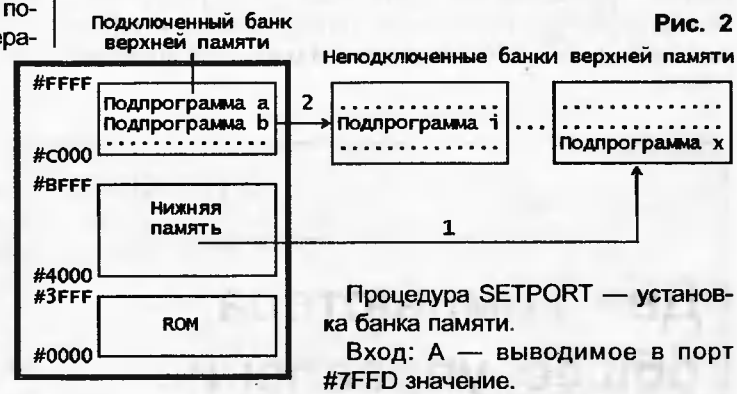
Подпрограммы в верхней памяти и особенности их вызова

При написании программы может возникнуть необходимость размещения отдельных ее подпрограмм в различных банках верхней памяти. Причины этого могут быть самыми разными. Может быть, программа получается настолько большой, что по-другому просто не умещается в памяти. Может быть, требуется выделить как можно больше нижней памяти под данные, которые должны быть всегда доступны из любого банка памяти, и из-за этого приходится уменьшать место, занимаемое в нижней памяти кодом программы, перенося большую его часть в верхнюю память. Может быть, программа пишется с учетом неодинаковой скорости доступа к различным банкам памяти (такой особенностью обладает как фирменный "ZX-Spectrum 128", так и некоторые совместимые модели), и исполняемый код требуется размещать только в "быстрых" банках. А может быть, для обработки данных нужен непрерывный участок памяти как можно большей длины.

Так вот, при обращении к размещенным в верхней памяти подпрограммам, если банк, в котором находится вызываемая подпрограмма, не подключен, возникают сложности. Например, как вызвать из нижней памяти подпрограмму, находящуюся в неподключенном на данный момент банке верхней памяти (рис.2, стрелка 1)? Или как вызвать из подключенного банка верхней памяти подпрограмму, находящуюся в другом банке верхней памяти (рис.2, стрелка 2)?

Очевидно, без переключения банков не обойтись. Поэтому начнем с рассмотрения используемой для этого процедуры.

Рис. 2



Процедура SETPORT — установка банка памяти.

Вход: A — выводимое в порт #7FFD значение.

Выход: значение выведено в порт и записано в переменную BANK.

Значения регистров: не изменены.

```
SETPORT   PUSH   BC
                 LD   BC,#7FFD
                 LD   (BANK),A
                 OUT  (C),A
                 POP  BC
                 RET

BANK             DS   1;Хранится текущее состояние порта #7FFD.
```

При вызове этой процедуры в аккумуляторе должно содержаться уже подготовленное для вывода в порт значение, т.е., кроме того что в битах 0...2 должен быть номер требуемого банка памяти, остальные биты также должны быть установлены соответствующим образом (см. предыдущий раздел). Если, например, при работе программы подключено ПЗУ BASIC-48, для вывода изображения используется стандартная видеостраница, и надо подключить 3-й банк ОЗУ, то для этого нужно вывести в порт #7FFD число #13.

В дальнейшем, когда я буду упоминать "номер банка", я буду иметь в виду значение, уже подготовленное для вывода в порт (если явно не подразумевается иное).

И еще, обратите внимание — в процедуре сначала производится запись выводимого значения в переменную BANK, а только потом вывод в порт #7FFD. Почему нужна именно такая последовательность действий? Предположим, что между командами записи и вывода в порт произошло прерывание, а процедура обработки прерывания устроена так: сначала она запоминает, какой банк памяти был подключен, по содержимому переменной BANK после этого устанавливает нужный ей банк, выполняет какие-то действия, а затем восстанавливает "старый" банк памяти. Так вот, если бы процедура SETPORT сначала выводила значение в порт, а потом записывала в переменную BANK, то в этом случае после окончания обработки

прерывания оказался бы установлен банк памяти, соответствующий старому значению переменной BANK. А это нам совершенно ни к чему.

С подключением банков, кажется, разобрались. Вернемся теперь к нашим подпрограммам. Как же быть с их вызовом? Рассмотрим сначала вызов из нижней памяти подпрограммы, находящейся в неподключенном на данный момент банке верхней памяти. Очень часто при этом еще требуется, чтобы после вызова был подключен тот же банк памяти, который был до вызова.

Последовательность необходимых для этого действий такова:

- запомнить номер текущего банка памяти;
- установить банк, в котором находится вызываемая подпрограмма;
- вызвать ее;
- установить тот банк памяти, который был до вызова (т.е. с ранее запомненным номером).

Тогда вызов подпрограммы можно оформить следующим образом:

```
    LD      A,(BANK)    ;Запоминаем номер
    PUSH    AF          ;текущего банка.
    LD      A,N         ;Устанавливаем банк, в котором
    CALL    SETPORT     ;находится вызываемая подпрограмма.
    CALL    subrout     ;Вызываем ее.
    POP     AF          ;Устанавливаем банк,
    CALL    SETPORT     ;который был до вызова.
```

Как видим, по сравнению с обычным CALL'ом тратятся лишних 13 байтов. И это еще не все. Если для передачи данных в вызываемую подпрограмму используется аккумулятор, то может потребоваться сохранение его значения (например, в каком-либо свободном регистре) перед установкой нужного банка памяти и восстановление перед вызовом подпрограммы, а на это также потратятся лишние байты. Точно так же, если аккумулятор (или регистр флагов!) используется для возвращения результата работы подпрограммы, возможно, потребуется где-то сохранять его значение (я говорю "возможно", потому что в некоторых случаях можно сразу же обработать возвращенный подпрограммой результат и уже потом, когда он больше не нужен, выполнить команды POP AF: CALL SETPORT). В итоге, как видим, дополнительные 13 байтов на вызов подпрограммы — это минимум, а в действительности может быть и больше.

Если одну и ту же подпрограмму требуется вызывать в нескольких местах программы, то можно, конечно, сам ее вызов оформить как подпрограмму.

Рассмотрим теперь другой случай — вызов из подключенного банка верхней памяти подпрограммы, находящейся в другом банке верхней памяти. Очевидно, что фрагмент программы, переключающий банки, придется вынести в нижнюю память. В результате вызов будет выглядеть примерно так:

```
;В банке памяти, откуда происходит вызов:

            CALL    l_subrout

;В нижней памяти:

l_subrout   LD      A,(BANK) ;Запоминаем номер банка верхней
            PUSH    AF       ;памяти, откуда был вызов.
            LD      A,N      ;Устанавливаем банк, в котором
            CALL    SETPORT  ;находится вызываемая подпрограмма.
            CALL    subrout  ;Вызываем ее.
            POP     AF       ;Устанавливаем банк, откуда был вызов,
            JP      SETPORT  ;и возвращаем туда управление (JP здесь
                             ;используется вместо CALL: RET, чтобы
                             ;сэкономить байт).
```

Как видите, на каждую вызываемую таким образом подпрограмму тратится, не считая CALL'а, еще не менее 16 байтов.

Итак, при вызове расположенных в верхней памяти подпрограмм приходится использовать громоздкие конструкции, причем находящиеся в нижней памяти, а ее и так не хватает. И если таких подпрограмм много (а если программа большая, то их много!), расход нижней памяти также составит значительную величину.

Менеджер вызова: что это такое и как он работает

Как уменьшить расход памяти на вызовы подпрограмм? При написании своей программы BestView я решил сделать так: пусть в нижней памяти будет расположена специальная процедура — так называемый менеджер вызова. При необходимости вызова какой-либо подпрограммы достаточно обратиться к менеджеру, передав ему номер этой подпрограммы, а он уже осуществит все необходимые действия по ее вызову — запомнит, какой банк памяти подключен; установит банк, в котором находится вызываемая подпрограмма; вызовет ее; после этого подключит банк памяти, бывший установленным перед вызовом, и вернет управление.

Адрес вызываемой подпрограммы и номер банка памяти, в котором она расположена, менеджер извлекает из соответствующих таблиц.

Естественно, значения всех регистров должны сохраняться: с какими значениями был вызван менеджер — с такими же он должен вызвать требуемую подпрограмму; какие значения оказались в регистрах после вызова подпрограммы — такие же должны остаться и после окончания работы менеджера.

И еще одно требование — реентерабельность. То есть нужно учитывать, что в любой момент работы менеджера, от обращения к нему и до выхода из него, может произойти прерывание, и в процедуре его обработки также может быть обращение к менеджеру (и, разумеется, когда менеджер запускает подпрограмму, она, в свою очередь, также может содержать обращения к менеджеру).

После того как требования к менеджеру были сформулированы, дело было за малым — написать его. И тут возникли вопросы.

Вопрос 1. Как передавать менеджеру номер вызываемой подпрограммы?

Передавать номер в каком-либо регистре? Но, во-первых, тогда потребуется дополнительная память на команду его загрузки в каждой точке вызова, а во-вторых, быть может, именно этот регистр используется для передачи параметров в вызываемую подпрограмму.

Указывать номер в байте, непосредственно следующем за командой вызова менеджера? Тогда в каждой точке вызова будет тратиться лишний байт (а если подпрограмм больше 256, то даже два байта). Казалось бы, один байт — пустяки, но если подсчитать общее количество вызовов... К тому же, затруднится отладка программы, из-за того что отладчик будет считать байт номера байтом начала следующей команды.

К счастью, есть оригинальный выход, лишенный перечисленных выше недостатков! Сделаем для вызова каждой подпрограммы свою точку входа, так чтобы в итоге управление передавалось на один и тот же адрес:

```
;Точки входа:

subr_1 NOP          ;Для вызова подпрограммы SUBR_1.
subr_2 NOP          ;Для вызова подпрограммы SUBR_2.
.....................................
subr_n NOP          ;Для вызова подпрограммы SUBR_N.

;Началась обработка...
```

Тогда можно установить, какая точка входа была использована. Пусть n — адрес команды обращения к менеджеру (рис.3). При ее выполнении в стек будет помещен адрес первого байта следующей команды — n+3. Взяв этот адрес и уменьшив его на два, мы получим адрес n+1, с которого размещается адрес точки входа. Взяв адрес точки входа и отняв от него адрес первой точки входа (subr_1), мы получим номер использованной точки входа, т.е. номер вызванной подпрограммы. Легко и просто, не правда ли? Номер не требуется явно указывать ни в регистрах, ни в памяти, и на это не тратятся лишние байты!

Рис. 3

n	n+1	n+2	n+3
Код команды CALL	Младший байт адреса точки входа	Старший байт адреса точки входа	Первый байт следующей команды

Тем не менее, у этого способа есть свои особенности, которые нужно учитывать.

Во-первых, при вызове подпрограммы какое-то время будет затрачено на выполнение цепочки NOP'ов (4 такта на каждый), и чем ближе точка входа к началу, тем больше будет задержка. Так что, если для каких-то подпрограмм задержка при вызове нежелательна, лучше размещать их точки входа ближе к концу.

Во-вторых, часто используют такой прием оптимизации — последовательность CALL subrout: RET заменяют просто на JP (или JR) subrout, тем самым экономя память и повышая быстродействие (выигрыш составляет 1 байт/17 тактов для JP и 2 байта/15 тактов для JR). Но обращение к менеджеру должно происходить ***только*** с помощью команды CALL! Иначе в стек не будет занесен адрес следующей после CALL команды, и, соответственно, нельзя будет определить номер вызванной подпрограммы. Так что данный способ оптимизации в случае, когда подпрограмма вызывается с помощью менеджера, использовать нельзя.

Впрочем, если имена самих подпрограмм записывать прописными буквами, а имена точек входа — строчными, то в тексте программы сразу будет видно, какая подпрограмма как вызывается: с использованием менеджера или нет. Например, видим в тексте CALL PRINT — это обычный вызов, видим CALL cls — это вызов с использованием менеджера. Так что сразу становится понятно, где можно выполнять оптимизацию, а где нельзя.

Вопрос 2. Как определять номер банка, в котором находится вызываемая подпрограмма?

Наиболее естественный ответ — по таблице. Можно в каждом байте таблицы хранить номер банка соответствующей подпрограммы, уже подготовленный для вывода в порт. Можно использовать тот факт, что на хранение номера банка требуется только три бита, и хранить данные более экономно: в одном байте — два номера или (что более сложно) в трех байтах — восемь номеров. Памяти будет расходоваться меньше, но придется затратить больше времени на извлечение номера банка из таблицы и подготовку к выводу в порт.

Но обратите внимание: с одной стороны, таблица неизбежно займет дополнительное место в памяти, а с другой — мы имеем столько же занятых командами NOP ячеек памяти, сколько имеется подпрограмм. Нельзя ли их совместить?

Оказывается, можно! Команды NOP в этих ячейках нужны лишь для того, чтобы, не выполняя каких-либо действий, перейти к началу обработки. А теперь вспомним, что в системе команд Z80 есть и другие команды, также не выполняющие каких-либо действий:

Команда	Код	Три младших бита кода
LD A,A	#7F=%01111111	7
LD B,B	#40=%01000000	0
LD C,C	#49=%01001001	1
LD D,D	#52=%01010010	2
LD E,E	#5B=%01011011	3
LD H,H	#64=%01100100	4
LD L,L	#6D=%01101101	5

Всего, вместе с NOP'ом, получается восемь команд, и банков памяти тоже восемь! Значит, между ними можно установить однозначное соответствие. И если для каждой подпрограммы мы поставим по адресу точки входа для ее вызова одну из восьми команд, соответствующую банку памяти, в котором находится эта подпрограмма, то лишнюю память на таблицу банков тратить не придется!

Вот она, красота кода!!! Команды, которые казались совершенно бесполезными, вдруг оказались единственно нужными! При этом еще смотрите, как удачно получается — у всех семи NOP-подобных команд разные три младших бита кода (см. таблицу). Так что удобно поставить в соответствие каждой из этих команд банк памяти, номер которого — три младших бита кода этой команды. Тогда для определения номера банка по коду команды достаточно будет обнулить пять старших битов командой AND %111.

Как можете видеть, в таблице нет команды, у которой три младших бита кода равны 6. Поэтому шестому банку мы поставим в соответствие команду NOP. Код этой команды — 0. Было бы, конечно, гораздо удобнее, если бы три младших бита кода NOP были бы равны 6 (или если бы среди семи NOP-подобных команд не оказалось команды с тремя младшими битами кода, равными 0). Тогда при определении номера банка по коду команды не потребовалось бы дополнительных проверок. Но чего нет, того нет...

Вопрос 3. Как уже упоминалось выше, для определения номера вызываемой подпрограммы нужно снять со стека адрес возврата и произвести определенные действия. При этом, очевидно, будут использованы некоторые регистры. Поэтому первоначальные значения этих регистров нужно сохранить, а перед вызовом подпрограммы восстановить.

Но если в реентерабельном участке кода требуется что-то сохранить, а затем восстановить, то использовать для этого можно только стек! В самом деле, смотрите, что будет, если сохранять значения в фиксированных ячейках памяти — если между сохранением и восстановлением произойдет прерывание, и процедура обработки прерывания обратится к этому же участку кода, то сохранение опять будет выполнено в те же ячейки памяти, и их первоначальное значение будет потеряно!

Но если мы сначала сохраним в стеке значения используемых регистров, то уже не сможем получить доступ к адресу возврата — ведь он теперь не будет на вершине стека! И это не единственная коллизия такого рода. Смотрите, перед запуском подпрограммы нам надо запомнить в стеке номер текущего банка памяти, чтобы потом, после запуска, восстановить его. Но если сначала мы запомним в стеке значения регистров, а потом — номер банка, то как будем восстанавливать значения регистров перед вызовом подпрограммы? Они ведь уже не будут на вершине стека! И еще, как будем извлекать из стека этот номер банка после окончания работы запущенной подпрограммы? Перед этим придется сохранить в стеке значения используемых регистров, а значит, номер банка уже не будет на вершине стека. И как же быть???

Вопрос 4. А как, собственно, запускать подпрограмму? Пусть нам известен ее адрес, ну и что? Записать этот адрес в поле операнда команды CALL и выполнить ее? Ага,

разбежались! Еще раз повторю — если в реентерабельном участке кода требуется что-то сохранить (в данном случае адрес запуска), чтобы потом использовать (в данном случае при выполнении команды CALL), то фиксированные ячейки памяти (в данном случае поле операнда команды CALL) использовать для этого нельзя!

Ответы на эти два вопроса заключаются в нетрадиционном использовании стека и команд работы с ним. Думаю, лучший способ объяснить это — подробно рассмотреть все манипуляции со стеком в менеджере.

На рисунках справа от каждого элемента стека указывается его длина в байтах. Вершина стека изображена сверху, но учитывайте, что в памяти стек хранится перевернутым — вершине соответствует самый низкий адрес, или, как говорят, стек растет вниз. Адрес вершины стека содержится в регистре SP.

Исходное состояние при вызове менеджера:

Адрес возврата	2

Резервируем в стеке 5 байтов. Для этого достаточно уменьшить SP на 5 (не забываем — стек растет вниз!). Уменьшение выполняется так — DEC SP: PUSH HL: PUSH HL. Что будет занесено в стек командами PUSH, в данном случае совершенно не важно — мы используем их лишь для уменьшения SP (потому что PUSH на байт короче, чем две команды DEC SP):

Резерв	5
Адрес возврата	2

Сохраняем в стеке HL, DE, AF:

AF	2	11
DE	2	
HL	2	
Резерв	5	
Адрес возврата	2	

Зная адрес вершины стека и смещение элемента в стеке, можно получить к нему доступ, даже если это не верхний элемент! А смещение адреса возврата мы знаем — как видно из рисунка, оно равно 11. По адресу возврата определяем номер вызываемой подпрограммы и номер банка памяти, в котором она находится. Запоминаем этот номер банка в стеке:

Номер банка п/п	2	12
AF	2	
DE	2	
HL	2	
Резерв	4	
Резерв	1	
Адрес возврата	2	

Определяем адрес вызываемой подпрограммы по ее номеру. Сохраняем номер текущего (т.е. установленного при вызове менеджера) банка памяти в одном из ранее зарезервированных байтов стека, по адресу SP+12:

Номер банка п/п	2
AF	2
DE	2
HL	2
Резерв	4
Номер банка, подключенного при вызове менеджера	1
Адрес возврата	2

Записываем в оставшиеся 4 байта резерва адрес, по которому будет передано управление при выполнении команды RET в конце вызываемой подпрограммы (это адрес SEL_EXIT, относящийся к менеджеру), и адрес вызываемой подпрограммы:

Номер банка п/п	2
AF	2
DE	2
HL	2
Адрес п/п	2
Адрес SEL_EXIT	2
Номер банка, подключенного при вызове менеджера	1
Адрес возврата	2

Снимаем со стека номер банка подпрограммы. Подключаем этот банк. Снимаем ранее запомненные в стеке значения HL, DE, AF. Теперь значения всех регистров — такие же, какими они были при вызове менеджера:

Адрес п/п	2
Адрес SEL_EXIT	2
Номер банка, подключенного при вызове менеджера	1
Адрес возврата	2

Выполняем команду RET, при этом процессор снимет со стека адрес подпрограммы и передаст по нему управление. Как видите, RET здесь используется для вызова подпрограммы, хотя обычное назначение этой команды противоположное — выход из подпрограммы. Вот такой нестандартный прием. :-)):

Адрес SEL_EXIT	2
Номер банка, подключенного при вызове менеджера	1
Адрес возврата	2

После того как подпрограмма выполнится, при выходе из нее по команде RET управление будет передано на следующую команду менеджера (с адресом SEL_EXIT):

Номер банка, подключенного при вызове менеджера	1
Адрес возврата	2

Запоминаем в стеке HL и AF:

AF	2	4
HL	2	
Номер банка, подключенного при вызове менеджера	1	
Адрес возврата	2	

По адресу SP+4 находится номер банка памяти, который был подключен при вызове менеджера. Устанавливаем этот банк. Снимаем со стека AF и HL. Теперь содержимое всех регистров такое же, какое было при выходе из подпрограммы:

Номер банка, подключенного при вызове менеджера	1
Адрес возврата	2

Командой INC SP убираем из стека ненужный теперь номер банка:

Адрес возврата	2

Работа менеджера завершена! По команде RET возвращаем управление.

И еще, обратите внимание — область применения менеджера оказалась шире, чем задумывалось при его создании, т.е. его можно использовать не только для вызова подпрограмм, расположенных в верхней памяти.

Пусть, например, нам надо вызвать подпрограмму, находящуюся в нижней памяти, но обрабатывающую данные, расположенные в определенном банке верхней памяти. Ничего нет проще! Помещаем сведения о ней (адрес и номер банка с данными) в менеджер, и можно будет ее вызывать, причем из любого банка памяти. А если подпрограмму надо вызвать сначала для обработки данных в одном банке памяти, потом — в другом банке, то можно просто несколько раз описать ее в менеджере, указывая каждый раз один и тот же адрес, но различные номера банков памяти.

Ну что же, осталось лишь привести листинг менеджера. После столь подробных объяснений, думаю, непонятных мест в нем не будет!

Листинг менеджера вызова

```
;Коды NOP-подобных команд, соответствующих каждому
;из 8 банков памяти:

bank_0    EQU    #40     ;LD    B,B
bank_1    EQU    #49     ;LD    C,C
bank_2    EQU    #52     ;LD    D,D
bank_3    EQU    #5B     ;LD    E,E
bank_4    EQU    #64     ;LD    H,H
bank_5    EQU    #6D     ;LD    L,L
bank_6    EQU    #00     ;NOP
bank_7    EQU    #7F     ;LD    A,A

;Таблица адресов подпрограмм:

SEL_TAB   DW     SUBR_1
          DW     SUBR_2
          ..........
          DW     SUBR_N

;В принципе, таблицу адресов можно разместить и в верхней
;памяти, тогда в нижней памяти затраты составят лишь 1 байт
;на каждую подпрограмму (без учета длины исполняемого кода
;менеджера).

;Точки входа:

SEL_BEG
subr_1    DB     bank_3
subr_2    DB     bank_6
.....................
subr_n    DB     bank_1

;Началась обработка:

          DEC    SP              ;Резервируем в стеке
          PUSH   HL              ;5 байтов.
          PUSH   HL

          PUSH   HL              ;Сохраняем
          PUSH   DE              ;используемые
          PUSH   AF              ;регистры.

          LD     HL,11
          ADD    HL,SP
          LD     E,(HL)
          INC    HL
          LD     D,(HL)

;DE - адрес возврата после вызова;
;перед этим адресом стоит команда
;CALL XXXX; вот этот адрес XXXX и берем:

          EX     DE,HL
          DEC    HL
          LD     D,(HL)
          DEC    HL
          LD     E,(HL)
          EX     DE,HL

;В HL адрес XXXX; смотрим, какая команда по нему расположена, и
;по ее коду определяем номер банка памяти, в котором расположена
;вызываемая подпрограмма:

          LD     A,(HL)
          AND    A
          JR     Z,BANK_M_1      ;Если NOP,

          AND    7               ;иначе - номер банка в младших трех
          OR     #10             ;битах кода команды; OR #10 - для
          JR     BANK_M_2        ;установки остальных разрядов
                                 ;порта.

BANK_M_1  LD     A,#16           ;NOP соответствует 6 банку.

;Внимание! Если в программе используется не основной экран и/или
;не ПЗУ BASIC-48, то значения в командах OR #10 и LD A,#16
;должны быть скорректированы!

;Теперь в аккумуляторе число, которое нужно вывести в порт #7FFD
;для подключения банка памяти, где расположена подпрограмма.

BANK_M_2  PUSH   AF

          LD     DE,-SEL_BEG
          ADD    HL,DE           ;Смещение
          ADD    HL,HL           ;*2.
          LD     DE,SEL_TAB
          ADD    HL,DE
          LD     E,(HL)
          INC    HL
          LD     D,(HL)          ;Куда.

;Заносим в стек номер банка:

          LD     HL,12
          ADD    HL,SP
          LD     A,(BANK)
          LD     (HL),A

;Адрес возврата:

          DEC    HL
          LD     (HL),SEL_EXIT/256  ;Старший байт.
          DEC    HL
          LD     (HL),SEL_EXIT\256  ;Младший байт.

;Адрес вызываемой подпрограммы:

          DEC    HL
          LD     (HL),D
          DEC    HL
          LD     (HL),E

;Устанавливаем банк вызываемой подпрограммы:

          POP    AF
          CALL   SETPORT             ;N банка

          POP    AF
          POP    DE
          POP    HL

;В стеке сейчас адрес вызываемой подпрограммы, а за ним - адрес
;команды SEL_EXIT.

          RET                        ;Запуск подпрограммы

;Восстанавливаем банк, номер которого был сохранен в стеке:

SEL_EXIT  PUSH   HL
          PUSH   AF
          LD     HL,4
          ADD    HL,SP
          LD     A,(HL)
          CALL   SETPORT
          POP    AF
          POP    HL
          INC    SP

          RET
```

3,5" FDD на Sinclair

Е.ИЛЯСОВ,
412302, г.Балашов, ул. Красина, 82.

Введение в проблему

Настоящие компьютеры класса Sinclair — это весьма оригинальные и своеобразные изделия фирмы Sinclair Research Ltd. Всевозможные фирменные периферийные устройства для таких компьютеров также проектировались довольно специфично и учитывали схемотехнические особенности этого класса машин (ZX-принтер, световое перо, мыши Kempston и AMX и многое другое). Это позволяло упрощать их конструкцию, добиваясь максимальной эффективности работы при минимальной себестоимости.

В то же время, развитие вычислительной техники последнего десятилетия подразумевает и другой подход к оснащению компьютеров периферийными устройствами. Это — стандартизация и унификация. Одной из первых такой принцип стала широко использовать в своих разработках фирма IBM. Результаты этого решения (плюс концепция открытой архитектуры) известны — IBM-совместимые машины сейчас составляют около 90% парка всех персональных компьютеров.

Sinclair-совместимые компьютеры, при всей своей оригинальности, также не чужды такого подхода. Пример — большинство дисковых интерфейсов для них, в том числе и ставший в СНГ стандартом де-факто интерфейс Beta-Disk. Он может работать практически с любыми накопителями на гибких магнитных дисках, известными на момент его разработки.

Безусловно, использование дисковой операционной системы коренным образом меняет стиль общения с компьютером и поднимает работу с ним на качественно новый уровень. Тем же, кто еще недавно терпеливо терзал магнитофон, порхание пластиковых конвертов в дисковод и обратно и вовсе представляется верхом совершенства. Но после того как немного стихает эйфория, приходит понимание того, что новое устройство ввода-вывода несет с собой и целый ряд новых проблем.

Дело в том, что большинство 5,25-дюймовых двусторонних дисководов двойной плотности (Double Side, Double Density — DS/DD), получивших широчайшее распространение в границах СНГ, уже морально устарели, а многие из них и физически изношены, и за пределами бывшего Союзв встречаются крайне редко. Кроме того, практически все отечественные модели таких приводов имеют значительный разброс регулировок после юстировки их на заводах-изготовителях. Это приводит к тому, что записи, сделанные на разных дисководах, могут быть плохо совместимыми между собой (особенно на последних дорожках). Доходит до того, что продукция некоторых производителей (например, бакинского завода) котируется, в основном, на уровне запчастей. А неприлично большой уровень шума при позиционировании головок у "родных" 5-дюймовиков даже считается их "визитной карточкой". К сказанному остается добавить ставший хроническим дефицит новых 5,25"-дискет и их недостаточную защищенность (вспомним целую галерею грозных пиктограмм и запретительных надписей на обратной стороне защитных конвертов).

В то же время, более совершенные 3,5-дюймовые дисководы и по сей день широко применяются во всех современных компьютерных системах. И нет никаких серьезных причин (кроме, разве что, более высокой стоимости таких дисководов и дискет для них), препятствующих их использованию и для компьютеров системы Sinclair. А вот преимущества этих периферийных устройств более чем очевидны: компактность как дисководов, так и дискет, скромные требования к источнику питания, высокая надежность хранения информации на 3,5"-дискетах, более точная сборка и юстировка и, как следствие — гораздо лучшая совместимость между разными дисководами... Список преимуществ этим не ограничивается. Так почему же в домашних и бытовых компьютерах вообще, и в "Синклерах" в частности, используются главным образом именно 5,25"-дисководы, несмотря на все их очевидные технические недостатки?

На сложившееся положение вещей влияют, в основном, два фактора.

Во-первых, очень важен фактор стоимости. В данном случае стоимость складывается из стоимости самого дисковода и стоимости дискет для него. Устаревшая техника никогда не стоила особенно дорого (разве только, раритетные экземпляры для музея). Это также относится и к дисководам. Стоимость же одной 5,25"-дискеты в 2...3 раза меньше стоимости обычной компакт-кассеты для бытового магнитофона.

Во-вторых, фактор традиции. Он даже еще более важен, чем фактор стоимости. Возможно, это произошло потому, что именно 5,25" DS/DD-дисководы, как значительно более скоростные устройства внешней памяти, пришли в нашей стране на смену бытовым и специальным цифровым магнитофонам как для домашних компьютеров, так и для многих профессиональных вычислительных систем. Естественно, такая революция в технологиях хранения информации способствовала самому широкому распространению дисководов этого формата. Соответственно, все коллекции "синклеровских" системных и прикладных программ, многочисленные игровые архивы также хранятся записанными на таких дискетах. Следует добавить, что эти дискеты активно используются для обмена информацией и программами по почте, так как благодаря своей небольшой толщине их очень удобно пересылать в конвертах формата 1/2 A4.

Но в подобной монополии заключается одна очень серьезная опасность. Выпуск 5,25"-дисководов практически прекращен. Через какое-то время, обусловленное выработкой ресурса службы дисководов, начинают возникать проблемы, связанные с заменой вышедших из строя драйвов. Многие пользователи уже с этим столкнулись и хранят где-нибудь в шкафу один-два запыленных неисправных экземпляра.

Таким образом, оптимальным решением, позволяющим сочетать в себе как перспективность и новые возможности, так и совместимость с накопленными архивами плюс уникальную возможность информационного обмена как между синклеристами, так и с пользователями других платформ, может стать использование 3,5"-дисководов не ВМЕСТО 5,25", а ВМЕСТЕ с 5,25"-приводами.

3,5" FDD — одно из немногих периферийных устройств, которые можно подключить к компьютеру "малой кровью" — без паяльника, дополнительного программного обеспечения и т.п. Попробуем рассказать о технологии такого подключения подробнее. Пользователям, ранее уже решившимся на такую модернизацию, приведенные сведения могут помочь обойти некоторые "подводные камни", не варясь "в собственном соку". Тем же, кто еще не сделал свой выбор, эти выкладки, надеемся, послужат информацией к размышлению, чтобы, взвесив все "за" и "против", принять разумное решение.

Все приведенные здесь данные являются компиляцией нашего собственного опыта в этой области и дополнены выборками на эту тему из доступных печатных и электронных изданий.

Этапы большого пути

Этап первый — установка в корпус компьютера. При этом, если корпус изначально рассчитан на установку двух дисководов (как, например, очень популярный среди синклеристов пластико-

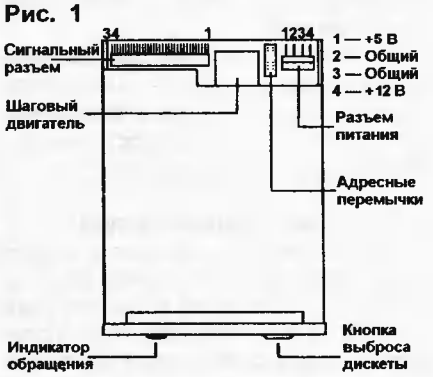
вый корпус от ПК "Корвет"), то эта задача упрощается и сводится к приобретению или изготовлению переходника для закрепления 3,5"-дисковода в габаритном отсеке для 5,25"-устройства. Для сохранения эстетичесности компьютера можно доработать ставшую лишней пластмассовую заглушку второго отсека, аккуратно сделав в ней вырез по размеру лицевой панели 3,5"-драйва.

Если конструкция корпуса не предусматривает двухдисководный вариант — не стоит переживать по этому поводу. Благодаря тому что габариты устанавливаемого дисковода невелики (примерно 25x100x150 мм, занимаемый объем — около 375 см^3), для него, как правило, всегда можно найти место в корпусе. Закрепить дисковод можно, используя как доступные элементы самого корпуса, так и дополнительные крепежные детали. В простейшем случае это могут быть даже металлические планки из детского конструктора, что предоставляет большую вариабельность в выборе места и способа крепления.

Второй этап — подведение питающего напряжения. Конструктивно разъем питания в 3-дюймовках выполнен иначе чем в 5-дюймовых устройствах, хотя назначение контактов и порядок их следования в разъеме не изменены (рис.1). Если нет возможности приобрести фирменную контактную фишку для подключения питания, можно временно использовать в качестве нее аккуратно обрезанную до 4 контактов гнездовую часть однорядного разъема. Такие разъемы широко применяются, например, для межмодульного соединения в отечественных цветных телевизорах третьего поколения. Следует знать, что напряжение +12 В служит только для сохранения совместимости, так как используется лишь в некоторых "древних" образцах 3-дюймовых дисководов. Все современные модели в нем не нуждаются, и их разъемы питания хотя и имеют соответствующий контактный штырек, но он обычно никуда не ведет — "висит в воздухе".

К источнику питания особых требований не предъявляется. Если 5,25"-дисководы, в зависимости от конкретной

Рис. 1



модели, потребляют токи около 0,55...0,7 А по +5 В и 0,45...0,7 А по +12 В (в пике — до 0,9/1,5 А соответственно), то 3,5"-приводы обычно укладываются в диапазон нагрузок по току 0,6...0,75 А по +5 В. Таким образом, если блок питания компьютера имеет некоторый запас по мощности, то подключение дополнительного трехдюймовика чаще всего происходит безболезненно.

Пункт третий — сигнальный разъем. Здесь дело облегчается тем, что цоколевки разъемов 3,5"- и 5,25"-приводов поконтактно совпадают между собой. Сложность же состоит в том, что конструкция фишки сигнального разъема здесь иная, чем в 5,25"-устройствах. Один из распространенных вариантов расположения сигнального разъема также показан на рис.1.

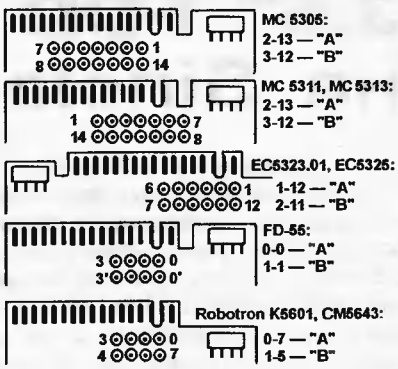
Поймать сразу нескольких зайцев можно, если использовать в качестве сигнального стандартный 34-жильный IBM-овский шлейф для флоппи-дисководов. Он, как правило, имеет длину около 50...60 см. Этого более чем достаточно для использования в любых корпусах, применяющихся для компьютерного конструирования. На него обычно "посажен" двойной комплект сигнальных разъемов для 5,25"- и 3,5"-драйвов, которые находятся: один — на конце шлейфа, а другой — ближе к его середине. Это позволяет подключить к контроллеру пару "разнокалиберных" дисководов в любом их сочетании.

Но здесь следует обратить внимание на одну тонкую особенность. Возможно, не будучи уверенными в достаточном коэффициенте интеллекта своих пользователей, производители IBM-совместимой периферии и соединителей для нее пошли по пути наименьшего сопротивления и стали выпускать 3,5"-дисководы с жестко заданным логическим адресом — чаще всего, "A". В тех же случаях, когда требуется установка двух дисководов — "A" и "B", изменение адресации второго дисковода достигается не перестановкой адресных перемычек (как в отечественных 5,25"-моделях), а "перепутыванием" части адресных сигналов, ведущих к крайнему (концевому) комплекту разъемов сигнального шлейфа.

Таким образом, если предполагается использовать такой шлейф для Sinclair, нужно будет аккуратно снять крайнюю пару разъемов и, развернув среднюю группу проводников (с 10-й по 16-ю жилы) на их законное место, вновь установить разъемы на шлейф. При этом 3,5"-флоп будет иметь логический адрес "A", а 5,25"-привод, как более покладистый, лучше установить с помощью соответствующих перемычек в "B" (рис.2).

На этом можно было бы и закрыть "железный" вопрос, если бы его не "подогревали" производители программного обеспечения. Здесь имеется в виду тот трудноообъяснимый нюанс, что авторы дисковых адаптаций Sinclair'овских программ (да и некоторых исконно дисковых программных продуктов) считали и считают второй дисковод излишней роскошью, и либо сами его не имеют, либо не предполагают его наличие у пользователей.

Рис. 2



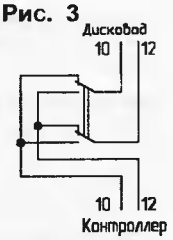
Оставим в стороне споры о правомерности такого подхода к программированию. Для нас важнее, что на практике это проявляется в том, что код таких программ содержит команды обмена данными с диском, жестко настроенные на дисковод "A". Нетрудно догадаться, какая масса проблем при этом гарантирована, если возникнет необходимость запустить такую программу с любого другого дисковода. Следуя известному правилу, что "нормальные герои всегда идут в обход", напомним способ обхода этого затруднения. Он позволяет без вмешательства в схему компьютера, с минимальными затратами сил и средств, в любой момент сменить текущий дисковод по умолчанию — 3,5" или 5,25". Идея эта не нова, и в разное время приходила в разные головы — что называется, "витала в воздухе". Вполне возможно, что она была навеяна уже упоминавшимся "айбиэмовским" подходом к адресации дисководов.

Напомним, что информацию о том, какой дисковод выбран системой в качестве текущего — "A" или "B" — несут сигналы контроллера BH0 и BH1 (выбор накопителя). В схемах разных компьютеров они могут называться по-разному — DSEL_0/DSEL_1 (ATM-turbo и Turbo-2+), DISKA/DISKB (Profi, Profi+), DS0/DS1 (Scorpion ZS-256, Scorpion ZS-256 Turbo+) или как-то по-другому. Общим же у них является то, что эти сигналы снимаются с контактов 10 и 12 сигнального разъема и передаются через одноименные жилы шлейфа. Весь фокус состоит в том, чтобы "по дороге" к дисководам "на ходу" менять эти сигналы, соответственно меняя и адресацию драйвов.

Это делается с помощью любого переключателя с фиксацией, желательно малогабаритного, например, П2К или подходящего тумблера — МТ-3, ПТ73-2-2.

Чтобы избежать чреватого неприятностями "копания" во внутренностях компьютера, безопаснее всего впаять переключатель в разрыв шлейфа от контроллера к дисководам (рис.3).

Рис. 3



Теперь появляется возможность давать любому из двух дисководов любое логическое имя ("A" или "B") простым переключением тумблера. Можно забыть про "упрямые" программы — достаточно "назначить" дискете с такой программой имя дисковода "A". По этому же принципу отпадают проблемы и с использованием защищенных от копирования авторских дискет — они распространяются преимущественно на дисках формата 5,25". Есть у такого способа "обмана" и другие, менее очевидные возможности, которые раскрываются по мере накопления опыта работы с использованием этого совета.

Новые диски к старому пулемету

Проблема выбора дополнительного 3,5"-дисковода и дискет для него часто выходит за рамки только технической задачи и лежит, скорее, в плоскости экономического планирования. Решение этой задачи сводится к правильной формулировке исходных условий. Это нужно, чтобы пользователь смог определиться прежде всего для себя — какая степень надежности хранения информации ему требуется и какие средства он готов для этого затратить.

Чтобы дать читателям возможность сделать оптимальный выбор, приведем минимально необходимые сведения, основанные на статистике после использования нескольких 3,5"-дисководов в течение последних лет.

Самыми недорогими считаются устаревшие модели этого семейства — DS/DD, или так называемые "720-килобайтные" (не будем забывать об их IBM-овской "родословной"). Их цена (без гарантии) на радиолюбительских рынках может быть эквивалентна 1...2 USD, со словесной гарантией (поскольку в магазинах их не встретишь) — на 30...50% дороже. У продавцов, заботящихся о своей репутации и выполняющих предпродажную подготовку (тестирование/регулировку) с гарантией, такие дисководы будут стоить существенно дороже — до 6...8 USD. Но в любом случае нормально работать подобные дисководы обязаны только с соответствующими дискетами — DD. Вот только встречаются в продаже эти диски (особенно новые) довольно редко.

Поэтому приходится использовать дискеты высокой плотности — HD (High Density) — "1,44-мегабайтные" по IBM-овской терминологии. Не все 3,5" DD-дисководы могут нормально работать на таком материале. Это объясняется неоптимальностью режимов работы узла записи-чтения и (или) конструкции магнитных головок, которая возникает при взаимодействии с высококоэрцитивным (трудно намагничиваемым) магнитным слоем HD-дискет.

Как правило, если удалось "с ходу" без проблем отформатировать системную дорожку дискеты, то и остальные треки, скорее всего, тоже будут работать без сбоев. Если этого не удается добиться, можно попробовать повторить форматирование, попытаться подобрать другой, более "надежный" тип дискет или же вовсе отказаться от таких планов (в общем — Retry, Ignore, Abort).

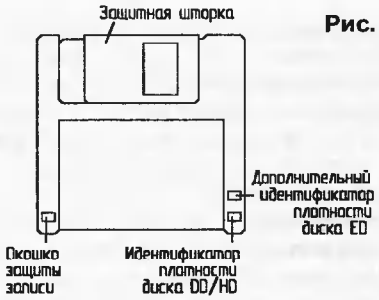
Таким образом, небольшие первоначальные расходы могут дополниться в будущем более значительными затратами средств на приемлемые по надежности работы носители информации. К слову сказать, автору в этом смысле повезло — 10-летней давности 3,5" DS/DD FDD "TEAC" без разбору "жует" дискеты любой разумной плотности, даже такие, о которые "ломали зубы" и более "крутые" машины.

Чаще всего все-таки для Sinclair-совместимых компьютеров используются 3,5"-дисководы высокой плотности (DS/HD). Новые приводы такого формата стоят около 10...15 USD. Разброс цен зависит в основном от фирмы-производителя и, соответственно, от качества исполнения устройства. Подержанные дисководы или изделия непрестижных производителей, естественно, могут стоить и меньше — порядка 7...9 USD. А продукция уважающих себя и держащих марку brand'ов — до 20 USD с гарантией.

Утешить в какой-то мере, как правило, небогатых пользователей Sinclair-совместимых компьютеров может то обстоятельство, что "разорившись" однажды на HD-дисковод, далее почти гарантированно можно свести к минимуму проблемы с подбором дискет для него. Необходимо только выполнить одно, хотя и очень важное, но довольно простое условие.

Дело в том, что БИС КР1818ВГ93, которая является "сердцем" контроллера дисководов для Sinclair-совместимых, не может работать в режиме высокой плотности (HD). Соответственно и HD-дисковод для нормальной работы на Sinclair должен быть переключен в режим двойной плотности (DD). Обычно дисковод выполняет такое переключение сам, ориентируясь на тип дискеты, которая в него вставлена. Если присмотреться к 3,5"-дискете со стороны этикетки, можно увидеть по ее краям два сквозных окошка (рис.4).

Рис. 4



Левое окошко служит для блокировки режима записи (форматирования) на дискете, аналогично вырезу на краю 5,25"-дискет. Но, в отличие от них, здесь открытое окошко означает запрет записи, а чтобы запись стала возможной, надо перекрыть окошко защелкой, находящейся с обратной стороны дискеты.

А вот правое окошко — это и есть определитель дисков высокой плотности и, как правило, защелки не имеет. На DD-дисках об этом окошке напоминает только неглубокий контур от пресс-формы на этом месте. Именно по этому окошку (или его отсутствию) 3,5" HD-дисковод определяет тип дискеты, а значит, и последующий режим своей работы — DD или HD. Добиться нужного режима работы драйва можно двумя способами:

1. "Хирургическое" вмешательство в сам дисковод — запайка выводов соответствующего микропереключателя (изредка это бывает оптоэлектронный датчик) таким образом, чтобы "микрик" постоянно выдавал на дисковод сигнал "DD". Достоинство этого способа в том, что нет необходимости в какой-либо доработке дискет — с этого момента все они будут считаться DD (конечно, с точки зрения именно этого дисковода). Недостатки — проблемы с чтением таких записей на других, не переделанных HD-дисководах (несоответствие формата самой дискеты и формата записи на ней). И самое главное — требуется высокая квалификация для такого рода вмешательства в схему, иначе очень велик риск испортить дисковод. Поэтому более безопасным выглядит второй способ.

2. Не трогая дисковод, доработать для него все HD-дискеты. Доработка заключается в превращении дисков из HD в DD. Для этого окошко определителя плотности нужно заклеить с нижней стороны — там, где на дискете нет этикетки, но есть металлический диск центратора. Материал для заклеивания может быть самый разный — вырезанные по размеру полоски самоклеящейся бумаги (стикерсов, ценников, этикеток от аудио- или видеокассет), изоляционной ленты или скотча. Недостатки такого способа — большой объем кропотливой работы, требующей аккуратности (если много дискет); лю-

бой липкий материал "собирает на себе" грязь и пыль; иногда такие "лэйблы" отклеиваются. Достоинства — не требуется ювелирная работа с паяльником, практически нет проблем с чтением таких дискет на других компьютерах.

Для полноты картины следует сказать, что существуют также 3,5"-дискеты с еще более высокой плотностью записи — так называемые ED (Extra Density), или, "по-IBM"овски" — 2,88-мегабайтные. Такие дискеты имеют еще одно окошко определения плотности, чуть выше основного (рис.4). Соответственно и реализуется этот режим записи/чтения только на специальных дисководах, имеющих такой определитель. Разумеется, для использования в Sinclair-совместимых компьютерах этот режим смысла не имеет. А поскольку встречаются такие дискеты и дисководы для них достаточно редко, то и в работе они не испытывались, и какие-либо рекомендации по их использованию отсутствуют.

Иногда встречаются также и комбинированные дисководы — устройства, содержащие в одном корпусе стандартной половинной высоты два привода формата 5,25" и 3,5". Работоспособность таких агрегатов в составе Sinclair-совместимой аппаратуры также не тестировалась.

Самое главное

На этом можно завершить "разбор полетов" и посоветовать пользователям не торопиться с принятием окончательного решения — быть или не быть трехдюймовому дисководу на своем компьютере:

1) постарайтесь тщательно взвесить свои потребности в пользовательском смысле, и свои возможности в финансовом плане;

2) попробуйте трезво оценить свою квалификацию, как монтажника, для выполнения необходимых операций;

3) просто подумайте — а надо ли?

По страницам электронных изданий

Подключение PC-мониторов к "ZX-SPECTRUM"

(c) MPR Hard.

Табл.1

N/N контакта	CGA monitor	EGA monitor
1	GND	GND
2	GND	RED INT
3	RED	RED
4	GREEN	GREEN
5	BLUE	BLUE
6	INTENSITY	INT / INT GREEN
7	–	INT BLUE / VIDEO
8	HORIZONTAL SYNC	HORIZONTAL SYNC
9	VERTICAL SYNC	VERTICAL SYNC

Меня очень часто спрашивают, как подключить различные мониторы к нашей любимой платформе. И вот, я не вытерпел и решил покончить с этим раз и навсегда. Итак, сначала распайка разъемов мониторов, а потом пояснения. В табл.1 приведена распайка для CGA- и EGA-мониторов, а в табл.2 — для VGA.

Обозначения в таблице:

- RED, GREEN, BLUE — соответственно сигналы красного, зеленого и синего цветов;
- INTENSITY — то же, что INT — сигнал яркости (интенсивности);
- HORIZONTAL SYNC — строчные импульсы;
- VERTICAL SYNC — кадровые импульсы;
- GND — земля (общий).

С подключением сигналов GND советую поэкспериментировать. Подача сигналов на выводы 6 и 7 разъема монитора EGA зависит от того, имеет ли данный монитор возможность переключения в режим CGA.

Советы по подключению:

1. Все сигналы советую брать непосредственно с тех элементов, где они вырабатываются (у меня, например, R, G, B и I — с КП11, а кадровые строчные синхроимпульсы — с ТМ2).

Табл.2

N/N контакта	VGA monitor
1	RED
2	GREEN
3	BLUE
4	–
5	test
6	GND *
7	GND *
8	GND *
9	–
10	GND
11	GND
12	–
13	HORIZONTAL SYNC
14	VERTICAL SYNC
15	–

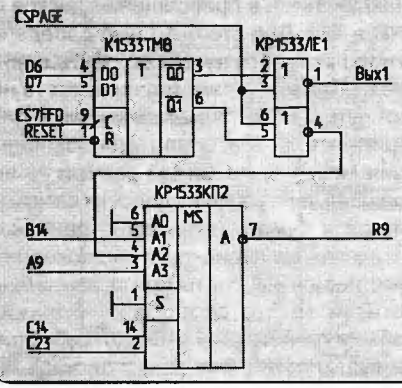
2. Если экран сдвинут влево (вправо), а подстройка монитора не помогает, проинвертируйте строчные импульсы.

3. Кадровые импульсы лучше пропустить через буфер, т.к. на некоторых моделях мониторов они сильно "подсаживаются", а на некоторых моделях компьютера они "завязаны" с прерываниями.

Доработка "ПЕНТАГОНА" до 512 К

(c) Roman Milukov

На рисунке приведена схема доработки.



Вместо м/с КР1533КП2 можно использовать КП12, а также применить микросхемы серии 555.

Предварительно необходимо сделать изменения в электрической схеме компьютера:

- заменить все микросхемы ОЗУ РУ5 на РУ7;
- сигнал Вых подать на мультиплексор КП11 вместо сигнала A9;
- вместо сигнала B14 (на КП11) подать GND;
- на КП11 поменять местами сигналы B11 и C35, A8 и C34.

В таблице приведены точки подключения схемы доработки к компьютеру "Пентагон-128":

Схема доработки	"Пентагон-128"
CS7FFD	Вывод 13 DD64
CSPAGE	Вывод 6 DD63
A9	Вывод 5 DD18
B14	Вывод 5 DD16
B11	Вывод 2 DD16
C35	Вывод 13 DD17
A8	Вывод 2 DD18
C34	Вывод 13 DD19
C14	Вывод 3 DD45
C23	Вывод 5 DD15

Ну, кажется, все... Да, адресная шина показана как A1...A16! Это, кажется, обозначения по схеме В.М.Г.

Подготовил Е.Зарецкий, (ZEG/Fenomen/Psyho), E-mail: zeg@tut.by

А.ВЕНДИЛОВСКИЙ,
г.Минск.

Max Payne

Глава первая

Темный переулок, в который едва пробивались огни с улиц. В самом дальнем углу четыре темные фигуры непринужденно вели разговор. Доминировал в этой беседе высокий тощий негр, которого все в округе звали Тертым, одетый по последней моде и с толстой золотой цепью на шее. Его вольная манера держаться и строить из себя крутого должна была убедить собеседников, двух мексиканцев, что с ним можно иметь дело. Его брат, нервно сжимавший сейчас "пушку" в одной руке и дипломат с товаром в другой, настороженно смотрел как на покупателей, так и на далеких прохожих.

— Послушай меня, брат! — Тертый улыбнулся, блеснув золотыми фиксами, — это настоящий и качественный товар. Прямо из Китая, только вчера расфасовали! Никаких тебе гонконгских подделок, последнее изобретение химиков! Кайф схватит сразу!

— Так уж Китай! — мексиканец, который назвался Мигелем, рассмеялся во весь голос — это дерьмо притащили с какой-нибудь рыболовецкой шхуны, идущей из Малайзии, и теперь ты хочешь его впарить мне. Мы торчим здесь уже полчаса, и это мне надоело. Я хочу видеть товар!

— А я деньги... — Тертый злобно ухмыльнулся, — и лучше скажи своему поделыцику, чтобы не тянулся за стволом, а то ко всем его шрамам добавится еще один, но уже фатальный!

Мигель снова рассмеялся, но быстро дал знак напарнику, и тот распахнул сумку, набитую долларами.

— Брат! Вот это уже дело! — Тертый открыл свой чемоданчик, с аккуратно уложенными пакетиками с белым порошком. Быстро достав один из пакетиков, он протянул его Мигелю. — Проба за счет фирмы, тогда ты поймешь, барахло наш Хозяин не держит!

Мигель передал пакет своему коренастому напарнику, который очень аккуратно вскрыл его ножом и, набрав этим же ножом небольшую порцию, поднес к ноздре. Через секунду его взгляд стал отрешенно-туповатым, на лице расплылась идиотская улыбка, и он со страшным акцентом произнес:

— Я вишшжу ангелов! — и опустился на колени.

— Не обращайте внимания, — Мигель был слегка удивлен столь сильным и быстрым эффектом — он раньше учился в церковной школе, вот и видит всякую чертовщину, когда нанюхается. — Он еще раз осмотрел пакет и порошок. — Мы будем вести бизнес, сейчас...

В этот момент, завывая сиреной, в переулок свернула полицейская машина. Продавцы и покупатели отступили в сумрак, куда не попадал свет автомобильных фар. Тертый что-то шепнул брату, и тот спрятал оружие, ему удалось удержать руку Мигеля с пистолетом. За рулем полицейского пикапа сидел грузный сержант Джонс, проработавший в полиции, ни много ни мало, а уже четверть века. Его напарник был с ним вместе уже десять лет во всех их делах, во всех...

— Билли, ты только посмотри, что это наши мальчики тут затеяли! — хохотнул Джонс, выбираясь из автомобиля. — Почему-то мне кажется, что это противозаконно! Может, отвезем их в участок?

Билли радостно осклабился, наблюдая, как Джонс, поигрывая дубинкой, направлялся к компании, не забыв направить револьвер в сторону всей честной компании. Так, на всякий случай.

— Спокойно, старина! — Тертый улыбнулся полицейскому, — Хозяин просил тебе кое-что передать.

В мгновение ока в его руке оказалось плотная стопка баксов, которая не менее быстро перекочевала в карман Джонса (он много раз говорил, что нужно "доить" этих черномазых уродов, пока они вообще на шею не сели). Заговорщицки подмигнув, Джонс возвращался к машине, мысленно радуясь, что покупка маленького домика на побережье уже не за горами, и можно будет послать эту работу к сеиньям, и жить на собственные сбережения до конца жизни, лежа на пляже и рассматривая красивых девушек в бикини. Хозяин платил щедро, и счет в мексиканском банке у Джонса был кругленький, с хорошим количеством нулей и единиц, когда...

Гулко прозвучавший выстрел оказался для всех полной неожиданностью. Тело Мигеля было просто подброшено в воздух, а его голова разлетелась на мелкие кусочки, расписав ближайшие стены красным орнаментом. Словно застыв на мгновение, оно рухнуло грузным мешком под ноги Тертого.

— А-а-а! Выходи сука! — Тертый и его брат стреляли в мельтешение теней. Джонс достал свой револьвер, когда ему показалось, что время стало замедляться, растягиваться и становиться тягучим. Из-за автомобиля, одним прыжком, не прекращая стрелять из обеих "Беретт", на освещенный участок вылетел незнакомец. Джонс отрешенно заметил, как медленно впиваются пули в тело брата Тертого, выбрасывая фонтанчики плоти и крови, заставляя его танцевать последнюю пляску смерти. Падая, тот не выпустил из рук автомат, и длинная очередь, выпущенная рукой умирающего, хлестко ударила по полицейской машине, разбивая стекла и пригвоздив Билли к сидению. Что-то тяжелое ударило в грудь полицейского, выбивая из его рук оружие и отбрасывая в сугроб. Темнота всей силой навалилась на него...

Он пришел в себя от дикого визга Тертого. Незнакомец не терял времени даром, и держал этого подонка за горло. Одно колено Тертого было прострелено, и судя по лицу неизвестного, скоро будет прострелено и второе.

— Да кто ты такой, мать твою! — Тертый еще пытался сохранить остатки достоинства. — Что тебе нужно, деньги, наркотики?! Хозяин найдет и душу из тебя вышибет! Ты труп, урод! Труп! — он пытался сказать еще что-нибудь, но тут раздался выстрел, и пуля разнесла второе колено. После этого крики перешли в невнятное и жалостливое бормотание.

— Где находится Хозяин? — голос незнакомца был холоднее январского воздуха. В этот момент Джонс заметил, что его револьвер лежит недалеко от него. Нужно только протянуть руку. Но руки были, словно чужие, и не хотели слушаться.

— Не знаю! — Тертый уже визжал от боли, — не знаю! Не знаю! — повторял он как заведенный, завидев, как поднимается пистолет неизвестного к его голове. — Но я знаю, кто должен знать! — его голос стал тихим, всхлипывающим. —...Он сейчас в отеле на Медисон-стрит.

— ...В борделе на Медисон-стрит — поправил его незнакомец. Легким движением руки он подхватил один из выпавших пакетиков с наркотиком, разорвал его, и, несмотря на отчаянные протесты торговца, стал всыпать содержимое ему в рот. Через несколько минут все было кончено.

Джонс практически дотянулся до своего оружия, когда незнакомец четким ударом отбросил ногой револьвер в сторону и склонился над ним.

— Чего ты хочешь? — Джонс скривился от боли — Я полицейский, вызови 911! — незнакомец никак не отреагировал на его просьбу, и продолжал рассматривать его. — Там, в сумке, наркотики и деньги, возьми их и уходи, только вызови врачей!

Услышав о деньгах, неизвестный словно очнулся, быстро залез в карман Джонса и вытащил мятую пачку купюр.

— Да, ты полицейский, продажный полицейский... — незнакомец разжал руку, и ветер подхватил помеченные кровью купюры, разбрасывая их по переулку. На секунду его лицо оказалось освещено чудом не разбитой фарой машины, и Джонс понял, кто стоит перед ним. Он не боялся смерти, но то, что он увидел в глазах незнакомца, напугало его больше, чем если бы перед ним появился сам дьявол. Он даже не понял что кричит, и кричал до того момента, пока тяжелый каблук не опустился ему на горло.

Незнакомец подошел к мексиканцу, который продолжал стоять на коленях и никак не реагировал на происходящие события. Казалось, что он абсолютно не заметил ни перестрелки, ни трупов.

— Я вижшшу Господа! — прокричал он в лицо неизвестного.

— Ты ошибаешься, — голос незнакомца был печальным. — Бог давным-давно мертв.

Грохот "Беретты", и вновь тишина. Снежный буран быстро заносил следы побоища. И только снежинки плавились в еще горячей крови.

В пелене снега исчез и незнакомец, его путь пролегал в обход районов, где он мог встретить полицейских, и вел его к дешевому мотелю на краю города. Это было одно из тех мест, где доживали свой век отбросы общества. Здание, с въевшимся от подвала до чердака запахом немытых человеческих тел, мочи и дешевого виски, где никто никогда не смотрит друг другу в глаза. Брошенная на стол хозяина мятая "двадцатка", и вот уже никто тебя не помнит и не знает. Он шел по коридору, под его каблуками ломались использованные шприцы, и крысы бросались в разные стороны. Временами ему приходилось переступать через лужи блевотины или через постояльцев, грезивших в своих нар-

котических снах. Маленькая комната с дырявым матрасом и засиженной мухами лампочкой под потолком — все, что у него сейчас было. Не раздеваясь, он лег на кровать и закрыл глаза. И то, что причиняло ему боль, с новой силой бросилось на его измученную душу, заставляя переживать этот ад снова и снова...

... Он, Макс Пейн — полицейский, работающий под прикрытием — возвращается к себе домой. Он улыбается — ведь столько времени не видел своего ребенка и жену. Утомительно работать внедренным агентом среди банд подонков, но нужно было узнать, кто производит новый наркотик, и отсечь его каналы поступления на улицу. А дома можно расслабиться и обрести покой. В тот момент он еще не знал, что мафия уже подозревала его, и кто-то из продажных коллег "сдал" и его, и семью. Тихо скрипнула входная дверь, и перед глазами открылась страшная картина — вся мебель перевернута, повсюду следы борьбы и сверху доносятся умоляющие крики жены. Не помня себя от ужаса, он побежал наверх, на ходу доставая пистолет. Бандиты не ожидали столь быстрой и яростной атаки. Пейн стрелял и стрелял, пока не закончились патроны, пока последний враг не был нафарширован свинцом. Но было поздно, жена умерла у него на глазах, а в детской, в луже крови, лежал его сын. Боль и ярость, крик отчаянья и ненависти, который испугал бы даже дьявола, вырвались из его горла.

Теперь он бежал по следам своих врагов, не зная, что скоро на его глазах будет убит его друг и начальник, а те же продажные личности в полицейской конторе запишут эти преступления на его счет. Он станет парией, его будут преследовать и враги, и бывшие коллеги. Но теперь для него это не имело значения. Он потерял в жизни все, что было ему дорого, и ради чего стоило жить. Он не искал справедливости — он больше не верил в справедливость. Он не просил прощения — он не верил, что ему оно будет даровано. Он больше ни во что не верил, и только месть поддерживала его и наполняла безумной и невиданной силой, которой враги его ничего не могли противопоставить.

... Короткий сон не дал покоя. Он поднялся и пошел вновь в снежную бурю. Боль и ненависть, разбавленные местью, навсегда оставили в его душе отпечаток, первые снежинки коснулись его волос. Ангел Смерти и Мести вернулся на улицы города.

Глава вторая

Честно признаюсь, я очень не люблю экшены от третьего лица, а "Томб Рейдер" является для меня ругательным словом. И когда поступил очередной результат игрового долгостроя, я весьма прохладно отнесся к этому. Восторженные вопли коллег меня мало волновали, но когда восторги даже самых ярых лароненавистников стали переходить все мыслимые границы, я взял эту игру на вечер, посмотреть, что же это такое.

Поспать в ту ночь не удалось, игра захватила полностью и безоговорочно. Днем, перебиваясь короткой дремой в общественном транспорте и даже на рабочем месте, я уже ждал, когда смогу опять сесть за "Макса". Следующую ночь опять не удалось поспать. И только когда вместо мыши я стал водить по коврику чашкой с кофе, удивляясь, что курсор остается неподвижным, а в зеркале мое лицо от недосыпания и сидения за компьютером стало чем-то похоже на некоторых особо симпатичных представителей Ноктюрна, я понял, нужно менять свою жизнь, и выделить время на сон, часика этак два в сутки — иначе на игру времени не остается...

И в первую очередь внимание приалекает не движок игры, AI противника или сюжетная линия, а геймплей. Его здесь много, хочешь — ложкой ешь, хочешь — на хлеб намазывай. Все настолько логично, понятно и добротно сделано, что очень скоро ты вживаешься в роль персонажа, и самое главное, от этого тебя в течение всей игры ничего не отвлекает! Все словно сдвлано из плотно подогнанных друг под друга кирпичиков, создавая твердый шедевр.

Первым таким кирпичиком является интерфейс. Уж сколько нареканий на это получали другие подобные игры — и движение персонажа, и камера то слишком тупа, то настолько "гениальна", что невозможно играть. Это был самый большой минус, который отбивал всякое желание играть в подобные игры, за что и получил прозвище "вид через задницу". И вот чудо! Впервые создан экшен от третьего лица, в котором управление полностью идентично шуттерам от первого лица, причем игра при этом не теряет всех преимуществ своего жанра. Где-то на третьей минуте для вас уже не будет иметь никакого значения, от первого лица или от третьего, а все "комбо" и прочие прыжки будут просто приятным дополнением, которого так не хватало в том же Анрыле.

Движок игры заслуживает всяческих похвал, системные требования к нему могут напугать кого угодно, но, как показывает практика, игра прекрасно "бегает" на конфигурации, намного меньше минимальной. И если вы владелец Целерона 300 и Карточки Вуду 2, то спокойно сможете поиграть в "Макса".

Теперь о качестве графики. Самое близкое слово — это фотореалистичность и кинематографичность. Создатели вполне оправданно решили, что если они делают экшен, то пусть он будет похож на голливудские блокбастеры. На персонажах лица настоящих актеров, а их движения очень точно отображают реальные. Изменяется даже мимика! Четко прослеживаются всевозможнейшие повреждения, которые вы наносите окружающей местности — ни кровь, ни дырки в стенах никуда не исчезают. А как выглядит сама стрельба! Мне пришлось заложить душу, чтобы посмотреть, как эта игра будет выглядеть на 1000-герцовом Атлоне с 3 Гефорсвм. Это было просто шикарно — герой в прыжке вышибает окно и в облаке падающего стекла влетает в комнату, включается режим замедления, видны всполохи ружейных выстрелов, медленный полет пули, оставляющий реверсивный след в воздухе, видно, как она впивается в жертву, разрывая плоть и выбрасывая фонтанчик крови, и тело отбрасывается от сильного удара. Одна из пуль попадает в ящик с бутылками (Коктейль Молотова), и всполох огня пожирает комнату, разбивая и разворачивая на своем пути все, что попадется. Люди, отброшенные ударной волной, пятна крови на стенах...Такое я видел только в кино (недаром пароль на одном из уровней будет "Джон Ву" :). А передать, что выделывает перспектива в кошмарных снах главного героя, просто невозможно, по-моему, такие спецэффекты реализованы в играх впервые. И немалую роль в создании атмосферы играет дизайн уровней. Все учтено до мелочей — если это дешевый бордель, то тут и ободранные стены, и капающая с труб вода, и работающие телевизоры с очередной серией "мыльной оперы" (которые разлетаются вдребезги после хорошего удара битой), и наркоманы в туалетах (не считая шприцов, крыс и валяющихся по углам журнальчиков пикантного содержания). Ваши противники тоже не стоят столбом в ожидании, когда вы придете и застрелите их. Они РАЗГОВАРИВАЮТ между собой, по большей части о вас, но также и о природе, деньгах и женщинах — могут даже поспорить и поссориться, и тогда часть работы за вас будет сдвлана другими.

Отсюда плавно переходим к вопросу об AI — очень неплохо, сделанное разработчиками весьма впечатляет. Враг атакует, делает кульбиты, приседает, в какой-то мере старается использовать естественные заграждения. Очень часто может швырнуть в вас гранату и быстренько смыться туда, где его ожидает помощь.

Здравый смысл не покидал разработчиков при конструировании уровней — заблудиться невозможно, но нет ощущения, что твоя свобода ограничена, как это было в других играх, все идет строго сюжетно и очень последовательно, что теперь большая редкость.

Сюжет — много ли вы можете вспомнить экшенов с сюжетом, и не просто с вводным текстом из серии " Вы упали (попали, влипли, нужное подчеркнуть) куда-то, и — бла-бла-бла — иди от начала к концу и, самое главное, убей всех во имя (нужное вставить), и пусть улыбнется тебе удача"? А такие, чтобы сюжет был закручен как в хорошем детективе, открывался постепенно по ходу игрового процесса, чтобы вы хотели узнать, что же произойдет дальше, даже больше, чем пристрелить того поганца с пулеметом, который спрятался за ящиками на втором уровне? Если смогли вспомнить хотя бы пяток, значит, у вас богатый игровой опыт. А значит, в полной мере оцените литературный вклад в это игровое произведение. Совсем не удивительно, что кинематографы уже стали уделять пристальное внимание этой игре, и пошли упорные слухи о возможности производства фильма по сюжету игры...

Вывод — хит, с самой большой буквы, определяющий высотные планки качества для будущих произведений компьютерного производства. Больше сказать нечего.

Minimum (minimum graphical detail): 450 MHz AMD / Intel Processor (or compatible) 16 MB Direct3D Compatible Graphics Card 96 MB RAM

Recommended (medium graphical detail): 700 MHz AMD / Intel Processor (or compatible) 32 MB Direct3D Compatible Graphics Card 128 MB RAM

Supported Operating Systems: Windows 95 (OSR2 or later), Windows 98, Windows ME, Windows 2000.

КУПЛЮ, ПРОДАМ, ОБМЕНЯЮ

*Для публикации бесплатных объявлений **некоммерческого характера** о покупке и продаже радиодеталей, бытовой и радиолюбительской аппаратуры, их текст можно присылать в письме по адресам:*

141406, г.Москва, Химки-6, ул.Библиотечная, 18-84 или

220095, г.Минск-95, а/я 199,

*передавать по тел. **+(37517) 221-01-10** либо через E-mail:*

rl@radiopage.by

rm@radio-mir.com

*URL: **http://radio-mir.com***

Разыскиваю схему струйного принтера "Электроника МС-6317" и головку к этому принтеру, либо ее возможную замену.
E-mail: saniok@ic.km.ua

Ищу пакеты программ для "ZX-Spectrum": iS-DOS Assembler, iS-DOS ZX-Modem.
E-mail: tzaplin@mail.ru

Куплю ксерокопию принципиальной схемы и описание источника бесперебойного питания BACK-UPS mod.250I ф.APC (American Power Conversion).
Сергей.
E-mail: sergeis@vmail.ru

Куплю недорого компьютер "Scorpion-ZS-256" не ниже 7 МГц, 256 Кб, AY8912, не "пиратский", с техдокументацией, литературой и ПО на дискетах. Желательно наличие подключенного принтера. Возможен обмен (с доплатой) на АОН-30В (на Z80) и частотомер 20 МГц — в рабочем состоянии, а также "Электронику ВМ-12", два "ZX-Spectrum 48K" и др. — на запчасти.
150533 Россия, Ярославская обл., Ярославский р-н, с.Курба, ул.Юбилейная, д.11, кв.15.
Тел. (0852) 66-31-58.
Бутов А.Л.

Инвалид 2 группы с детства **купит** по сходной цене или примет в дар компьютер IBM AMD 486DX4-100; комплектующие к 486-му; дискеты с различными программами 5,25" и 3,5" и CD-ROMы.
682905, Россия, Хабаровский край,
р-н им.Лазо, п.Дурмин,
ул.Комсомольская, 1,
Ковалевич Я.В.

Со времен конструирования "ZX-Spectrum" осталось много деталей: ПЗУ (2764, РФ4), Z80, кварцы и др. **Отдам** за символическую плату или **поменяю** на другие детали.
620039, Екатеринбург, а/я 172,
Сергей.
E-mail: banan@pisem.net.

Куплю недорого к IBM Электроника МС1502: контроллер винчестера; винчестер; руководство на дискете 5,25" 720К; дискеты.
682905, Россия, Хабаровский край,
р-н им.Лазо, п.Дурмин,
ул.Комсомольская, 1,
Ковалевич Я.В.

Уважаемые читатели!

Те, у кого возникли проблемы с подпиской на наши журналы, могут получить их из редакции. Там же можно заказать имеющиеся в наличии отдельные номера журналов за предыдущие годы.

Год	Имеющиеся в наличии номера			Расценки на 1 экз. (с учетом пересылки)		
	Радиолюбитель	Радиолюбитель. Ваш компьютер	Радиолюбитель. КВ и УКВ	По России и СНГ (рос. руб.)	По Беларуси (бел. руб.)	По Украине (гривны)
1998	4, 8	1 — 5, 7 — 12	1, 4, 5, 6, 7, 11, 12	25	800	4
1999	3, 9, 10, 11	1 — 6, 10, 11, 12	3, 5, 6	30	1000	5
2000	4	1 — 12	4, 7, 10, 11, 12	35	1200	6
2001	3 — 6	1 — 6	2 — 6	40	1400	6,5
	Радиомир	Радиомир. Ваш компьютер	Радиомир. КВ и УКВ	По России и СНГ (рос. руб)	По Беларуси (бел. руб.)	По Украине (гривны)
2001	7 — 12	7 — 12	7 — 12	40	1400	6,5

Наши платежные реквизиты:

- для жителей Беларуси —

получатель: УП "РЛД", УНН 190218688
р/с 3012000004882 в ф-ле №524 АСБ "Беларусбанк" в г.Минске код 121.
Адрес банка: 220028, г.Минск, ул.Физкультурная, 31;

- для жителей России, Украины и др. стран СНГ —

получатель : ООО "НТК ИНФОТЕХ", ИНН 7703155561,
р/с 40702810100022120172 в АКБ "Межтопэнергобанк"
корр. счет 30101810900000000237 БИК 044585237
Адрес банка: 107078, г.Москва, ул.Садовая-Черногрязская, 6.

На бланка почтового перевода необходимо очень четко написать свой почтовый индекс, полный адрес, фамилию, имя и отчество полностью. В графе "Для письма" необходимо точно перечислить, какие конкретно номера какого из журналов Вы заказываете.

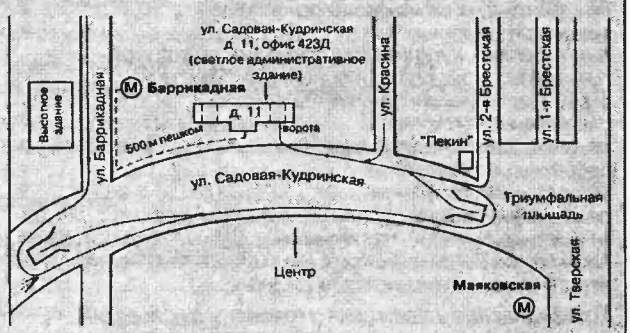
При оплате платежным поручением нужно предварительно выписать счет.

Вся информация по тел. в г.Москве (095) 139-75-77, в г.Минске (017) 221-01-10 или по E-mail: rm-sales@radio-mir.com

Приобретение отдельных номеров журналов

В магазинах радиодеталей **"ЧИП и ДИП"** по адресам:
- г.Москва, ул.Гиляровского, д.39, (ст. метро "Проспект Мира" — радиальная);
- г.Москва, ул.Ивана Франко, д.40, к.1, стр. 2,
(платф.Рабочий поселок, 15 мин. от Белорусского вокзала);
- г.Москва, ул.Беговая, д.2;
- г.Ярославль, ул.Нахимсона, 12, тел. (0852) 27-57-15.

В **"Бермосе"** по адресу: г.Москва, ул.Садовая-Спасская, 19/1
(ст. метро "Красные ворота").
На радиорынках в Москве: Митинском (места R4, S8, K52, E50, G56), Царицынском (место 121).
В интернет-магазине изд-ва **"DMK-Press"** по адресу www.dmk.ru с бесплатной доставкой по Москве.
На Украине: г.Киев, фирма **"Торм"**, тел. +(38044) 227-72-73.
В Беларуси: г.Минск, пр.Ф.Скорины, д.92, магазин **"Книга XXI век"**, тел. (017) 264-27-97 (ст.метро "Московская").
В издательстве **"Солон-Р"**, по адресу: г.Москва, ул.Садовая-Кудринская, д.11, офис 423Д (ст. метро Баррикадная, ст. метро Маяковская).
Время работы:
с 09.00 до 18.00
кроме субботы,
воскресенья.
Телефоны:
(095) 254-44-10,
252-36-96,
факс: (095) 252-72-03.
Радиорынки: Царицынский (место 13/А), Митинский (ряд 8, контейнер 12, место Z25).
E-mail: Sokolov-Solon@cobb.ru и Solon-R@coba.ru



ВАШ КОМПЬЮТЕР — 2001

УРОКИ ПРОГРАММИРОВАНИЯ

ДИАЛОГ ПРОГРАММИСТОВ

РЕЦЕПТЫ

МИР 8 БИТ

ИГРОТЕКА

РАДИОЛЮБИТЕЛЬСКАЯ ЯРМАРКА

ВАШ КОМПЬЮТЕР — 2001

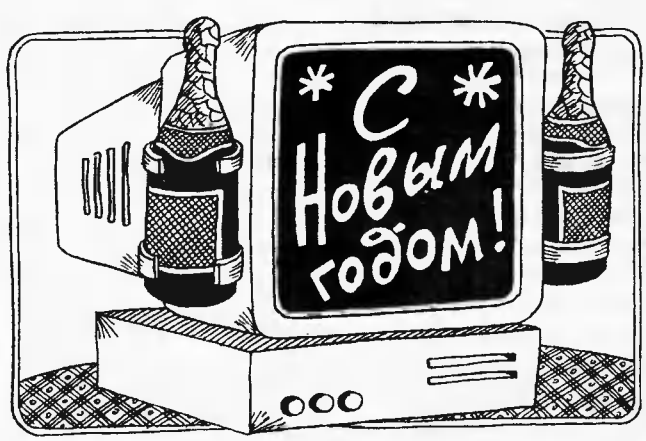


Рисунок О.Попова

Max Payne

Новый магазин "ЧИП и ДИП"

www.chip-dip.ru

ЧИП и ДИП

ЭЛЕКТРОННЫЕ КОМПОНЕНТЫ И ПРИБОРЫ

Чип и Дип поставляет электронные компоненты и приборы для инженеров-разработчиков, конструкторских бюро, опытного и мелкосерийного производства, служб эксплуатации и ремонта, сервисных центров и учебных заведений.

РОССИЯ 129110 г. Москва, а/я 996
e-mail: sales@chip-dip.ru

Оптовый офис
г. Москва,
ул. Гиляровского, 39
Тел/факс: (095)973-70-73
(многоканальный)
факс: (095)971-31-45

Магазины:
Центральный: г. Москва,
ул. Беговая, 2
с 09:00 до 19:30
суббота до 18:00
Тел. для справок (095)284-56-78,
284-36-69, 281-99-17, 971-18-27

г. Москва,
ул. Гиляровского, 39
с 09:00 до 19:30
суббота до 18:00
Тел. для справок (095)284-56-78,
284-36-69, 281-99-17, 971-18-27

г. Москва,
ул. Ивана Франко,
д.40, к.1, стр.2
Тел.: (095)417-33-55
e-mail: dipkorpus@platan.ru

г. С.-Петербург,
Кронверкский просп., 73
Тел.: (812)232-83-06, 232-59-87

г. Ярославль, пр. Ленина, 8а.
Тел.: (0852) 30-15-68
e-mail: chip-dip@yarteleport.ru

ПЛАТАН ЭЛЕКТРОННЫЕ КОМПОНЕНТЫ ДЛЯ ПРОМЫШЛЕННЫХ ПОТРЕБИТЕЛЕЙ

тел./факс: (095) 737-59-99
http://www.platan.ru