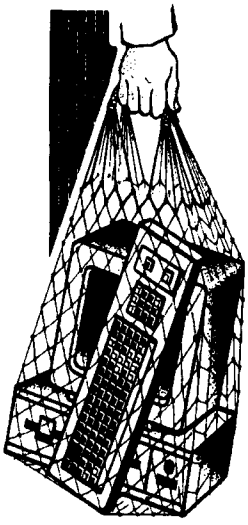
УВАЖАЕМЫЕ ЧИТАТЕЛИ!

У нас в редакции вы можете приобрести отдельные выпуски журналов «Персональный компьютер БК-0010 — БК-0011М», «Персональный компьютер УКНЦ» и «Информатика и образование». Здесь же можно оформить полугодовую подписку на все перечисленные издания с получением экземпляров лично в редакции или по почте. У нас вы также можете приобрести различное программное и аппаратное обеспечение для IBM, БК и УКНЦ.



ПРОГРАММНОЕ И АППАРАТНОЕ
ИНФО ИЕ ДЛЯ БК-0010(.01)
БК-0011(М)

матор ППЗУ.

роллер НГМД фирмы АЛЬТЕК для БК-0010(.01).

Усовершенствованный монитор среды БК-0010 для БК-0011(М) с возможностью реализации графики четкостью не хуже SVGA на IBM-совместимых.

Дисковая операционная система ANDOS v3.1

База данных, работающая DOS - BASIS 2.10

Мощный графический ред (М) Techno Art

Дисковая операционная с К-0010(.01) и БК-0011(М).

Музыкальный редактор а, ударные инструменты).

Профессиональный про копий экрана БК-0010 для переноса графики на IBM.

Конвертор листингов на ИСИКе в текстовые файлы формата EDASP и обратно.

Конвертор программ из внутреннего формата ФОКАЛа в листинги стандарта EDASP.

Резидентный драйвер ANIRAM AMADA, дополняющий ОС ANDOS следующими возможностями:

— копирование экрана в файлы;

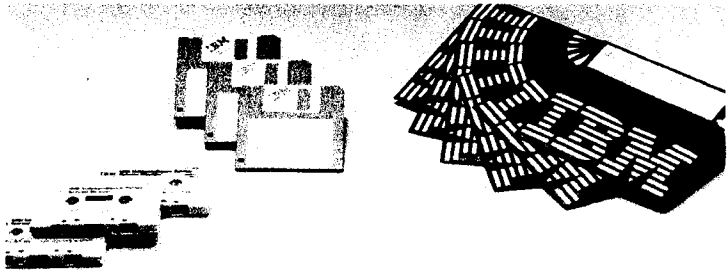
— сопровождение диалога с БК принтере;

— поддержка дополнительного ройства «P:» — «принтер».

Новые игры для БК-0011М: MAGICIAN, TOWER, DIZZY, JOE BLADE.

Широкий набор драйверов и утилит (в том числе для принтера D100) и многое другое.

Заключаем с авторами договоры на рекламу и коммерческое распространение программных и аппаратных разработок. Приглашаем к сотрудничеству книготорговые организации, фирмы и заинтересованных лиц для реализации нашей печатной продукции.



Телефон: 151-19-40
Факс: 208-67-37
E-Mail: mail@infoobr.msk.su

ПЕРСОНАЛЬНЫЙ КОМПЬЮТЕР БК-0010, БК-0011м

Приложение
к журналу
«ИНФОРМАТИКА
И ОБРАЗОВАНИЕ»

1'95 (7)

Издается с 1993 г.

В НОМЕРЕ

Москва «Информатика и образование» 1995

Авторы выпуска

Зальцман Ю. А.
Ивашинников С.
Ланеев А.
Леликов А. В.
Неробеев С. М.
Новиков Ф.
Румянцев С. К.
Рыбальченко Н. Д.
Самойлов В.
Семенов О. Ю.
Сорокин А. В.
Усенков Д. Ю.
Чабан С. П.
Юров В. П.

РЕДАКТОРЫ: *ВАСИЛЬЕВ Б. М.*
УСЕНКОВ Д. Ю.

«Библиотека журнала «Информатика и образование»
Свидетельство о регистрации средства массовой информации № 0110336
от 26 февраля 1993 г.

Телефон: (095) 151-19-40, 208-30-78
E-Mail: mail@infoobr.msk.su
Факс: (095) 208-67-37

Продолжаем публикацию неформального учебника по программированию на ассемблере для начинающих пользователей БК-0010(.01). Первые уроки этого компьютерного языка см. в № 1— 5 за 1994 г.

Ю. А. Зальцман,
г. Алма-Ата

МикроЭВМ БК-0010. Архитектура и программирование на языке ассемблера

Как известно, основное преимущество ассемблера перед прочими языками — скорость исполнения программ. Согласно технической документации, для БК-0010 она равна 300 тыс. операций в секунду. Но это не более чем абстрактная цифра, имеющая очень мало общего с фактическим быстродействием ЭВМ. Скорость исполнения программы, состоящей из реальных команд, зависит от многих факторов, из которых главными являются ТИП КОМАНД и СПОСОБЫ АДРЕСАЦИИ. С помощью специального устройства (электронного измерителя длительностей) в сочетании со специально написанными для этой цели программами скорость исполнения отдельных команд может быть непосредственно измерена, а затем путем логических рассуждений и расчетов можно найти правила для вычисления времени исполнения любых команд БК-0010 с любыми методами адресации. Автор проделал это исследование, результаты которого изложены ниже. Следует заметить, что правила вычисления времени исполнения команд для БК-0010 оказались заметно отличающимися от аналогичных для «прародителя» БК — компьютера PDP-11. Это естественно, так как эти ЭВМ, несмотря на одинаковую систему машинных команд, заметно различаются по внутренней структуре. Приступим к рассмотрению результатов, полученных на БК-0010.

Заметим сразу, что способы измерения времени исполнения команд могут быть различными и в зависимости от этого получаются заметно отличающиеся результаты. В частности, Б. Ф. Фролкин совершенно правильно обратил внимание автора на тот факт, что на время исполнения команд сильно влияет время отклика ОЗУ БК-0010 из-за особенностей работы контроллера ОЗУ К1801ВП1-037. Поэтому в конкретной программе время исполнения может несколько отличаться от вычисленного (в том числе это зависит и от типа предшествующей команды). Программы в ПЗУ БК-0010 исполняются несколько быстрее, чем в ОЗУ (по данным автора, увеличение скорости в ПЗУ может составлять от 10 до 30%, что в основном совпадает с цифрами, предоставленными Б. Ф. Фролкиным). При использовании приведенных ниже таблиц необходимо также учитывать следующее:

- полученные автором данные эмпирические и приближенные;
- данные относятся к программам, работающим в ОЗУ пользователя БК-0010.

Временные характеристики команд БК-0010

Прежде всего необходимо отметить, что тактовая частота процессора, установленного в БК-0010, в совокупности с особенностями этой ЭВМ обеспечивает реальное быстродействие примерно 250 тыс. операций в секунду. Но что это за операции? Речь идет об операциях типа **«регистр—регистр»**, т. е. об адресации **RN** (с кодом **0**). При всех прочих способах адресации скорость исполнения команд уменьшается. Время исполнения команд с кодом адресации 0 или безоперандных носит название ОСНОВНОГО ВРЕМЕНИ команды, для БК-0010 оно равно 4.0 мкс (1 мкс = 0.000001 с). Таким образом, за 4.0 мкс исполняются операторы NOP, CLC, CLV, CLZ, CLN, CCC, SLC, SLV, SLZ, SLN, SCC, а также все одно- и двухоперандные команды, т. е. CLR, INC, ASL, ASR, MTPS, SWAB, MOV, CMP, XOR, и прочие, если способ адресации операндов в них — RN (код 0).

Сложнее определить время исполнения команд при иных способах адресации. Вспомним, как мы обозначали операнды при описании операторов. Буквой **A** обозначался операнд-источник, а **B** — операнд-приемник в двухоперандных командах, например MOV A,B. Пара букв **AB** означала источник (он же и приемник) в одноперандных командах, например COM AB. Как **A1, A2** обозначались источник и приемник, не изменяющиеся в процессе исполнения команды, например CMP A1,A2. И наконец, операнд — адрес перехода обозначался как **N.** Используя те же обозначения, приводим таблицу ВРЕМЕНИ АДРЕСАЦИИ ОПЕРАНДОВ для БК-0010.

Таблица 1

Адресация		Тип операнда и добавочное время, мкс				
способ	код	A,A1	AB	B	A2, Nj	Ns
RN	0	0	0	0	0, —	—
@RN, (RN)+, -(RN), #X, #MET	1, 2, 4	4.0	5.4	6.8	6.8	10.8
@(RN)+, @-(RN), X(RN), MET(RN), @#X, MET	3, 5, 6	6.8	8.0	10.8	9.4	13.4
@X(RN), @MET(RN), @MET	7	9.4	10.8	13.4	12.0	16.0

Как пользоваться таблицей? Мы уже говорили, что ОСНОВНОЕ ВРЕМЯ для всех команд равно 4.0 мкс. Для вычисления времени исполнения одноперандной команды к основному времени необходимо прибавить время адресации операнда из таблицы. Для примера вычислим время исполнения ряда команд:

Таблица 2

Команда	Тип операнда, код адресации		Время исполнения осн. + добавочное, мкс	Сумма, мкс
TST R2	A1	0	4.0 + 0.0	4.0
TST @R4	A1	1	4.0 + 4.0	8.0
COM @R2	AB	1	4.0 + 5.4	9.4
MTPS #340	AB	2	4.0 + 5.4	9.4
CLR @#256	AB	3	4.0 + 8.0	12.0
INC @-(R4)	AB	5	4.0 + 8.0	12.0
ROR MET	AB	6	4.0 + 8.0	12.0
NEG @10(R5)	AB	7	4.0 + 10.8	14.8

С двухоперандными командами дело обстоит значительно сложнее, и для приближенного вычисления их времени исполнения можно предложить следующие правила. Для операнда-источника время берется из таблицы без всяких оговорок, тип операнда — A или A1 (в зависимости от конкретного оператора). А вот для операнда-приемника принимается тип B только тогда, когда источник — RN, в противном же случае тип приемника берется AB. Для операторов CMP и BIT, которые не меняют операнды, естественно, время адресации второго операнда короче: если источником является RN, то для приемника берется тип A2, при ином же способе адресации источника тип приемника принимается A1. При работе с оператором XOR для второго операнда принимается тип A2. Все эти сложности объясняются порядком, в котором процессор обрабатывает операнды на микропрограммном уровне, и уже отмеченными особенностями самой ЭВМ. Примеры:

Таблица 3

Команда	Тип операнда, коды адресации		Время исполнения, осн. + добавочное, мкс	Сумма, мкс
MOV R2,R4	A, B	0, 0	4.0 + 0.0 + 0.0	4.0
MOV R2,@R4	A, B	0, 1	4.0 + 0.0 + 6.8	10.8
MOV @R2,R4	A, AB	1, 0	4.0 + 4.0 + 0.0	8.0
MOV @R2,@R4	A, AB	1, 1	4.0 + 4.0 + 5.4	13.4
SUB @MET,R5	A, AB	7, 0	4.0 + 9.4 + 0.0	13.4
ADD MET,MET	A, AB	6, 6	4.0 + 6.8 + 8.0	18.8
CMP #100,R5	A, A1	2, 0	4.0 + 4.0 + 0.0	8.0
CMP R5,#100	A, A2	0, 2	4.0 + 0.0 + 6.8	10.8
BIT 'A',@#256	A, A1	2, 3	4.0 + 4.0 + 6.8	14.8
XOR R3,MET	A, A2	0, 6	4.0 + 0.0 + 9.4	13.4

Отметим, что время исполнения команд на БК-0010 не зависит от того, работает ли команда с байтом или со словом.

Еще ряд операторов имеет время исполнения, которое должно быть оговорено особо.

Таблица 4

Оператор или группа операторов	Общее время исполнения, мкс
JMP N	см. табл. 1, столбец Nj
JSR RN,N (или CALL N)	см. табл. 1, столбец Ns
RTS RN (или RET)	10.8
BR MET, BEQ MET и все прочие операторы ветвления независимо от того, выполняется ли переход по условию	5.4
SOB RN,MET независимо от того, выполняется ли ветвление	6.8
EMT, TRAP, BPT, IOT	22.8
HALT	48.0
Прерывания от внешних устройств	≈ 20 мкс + время завершения исполнения текущей команды
RTI, RTT	13.4
RESET	≈ 380 мкс

Приведем также время отработки некоторых часто используемых прерываний EMT БК-0010, что представляет интерес с практической точки зрения. Время приводится в миллисекундах (1 мс = 1000 мкс).

Таблица 5

Прерывание EMT	Общее время исполнения (приближенно)
EMT 4	0.2 мс
EMT 14	240 мс
EMT 16	от 0.4 (R0 = 0) до 1.7 мс (в R0 код символа); с погашенным курсором вывод символа — 1.1 мс; в режиме «ИНВ.С» время вывода больше на 100 мкс, в режиме «ПОДЧ.» — на 20 мкс; в режиме «32 символа в строке» все значения возрастают примерно вдвое; сброс экрана — 200 мс; перевод строки — 1.8 мс (с рулонным сдвигом — 18 мс)
EMT 20	(1.6 × N) + 0.5 мс, где N — число выводимых символов; в спецрежимах (инверсия, подчеркивание, 32 символа) — поправка на каждый символ как для EMT 16
EMT 22	от 1.1 (вывод символа) до 34 мс (сброс строки); в спецрежимах — поправка аналогично EMT 16
EMT 24	1.8 мс (в символьном режиме); 2.4 мс (в графическом режиме)
EMT 26	0.25 мс (в символьном режиме); 0.27 мс (в графическом режиме)
EMT 30	0.5 мс независимо от кода операции
EMT 32	(0.5 × N) + 0.5 мс, где N — число точек вектора (независимо от кода операции); время несколько варьирует в зависимости от направления черчения (наименьшее время — для горизонтальных и вертикальных векторов)

Контрольные вопросы и задания

1. Что выгоднее с точки зрения скорости работы ЭВМ ставить на первое место в двухоперандных командах с разными способами адресации: «быстрый» или «медленный» операнд? Подсчитайте время исполнения команд CMP @MET,RN и CMP RN,@MET.

— Медленный, при этом общая скорость исполнения команд заметно возрастает. Время — 13.4 и 16 мкс соответственно.

2. Что исполняется быстрее: обращение к подпрограмме или переход к обработке командного прерывания?

— Обращение к подпрограмме намного быстрее: JSR RN,N + RTS RN даже в самом худшем случае (код адресации 7) требует 26.8 мкс, а, например, EMT + RTI — 36.2 мкс. Это не говоря уже о том, что для обращения по EMT или TRAP необходимо потратить значительное время еще и на извлечение и обработку команды программой-диспетчером.

3. Подсчитайте, сколько времени займет исполнение команд:

MOV #12,R0
EMT 16

— Первая команда — 8 мкс, вторая, если не происходит рулонный сдвиг экрана, — 1.8 мс (при сдвиге — 18 мс). Очевидно, что по сравнению со второй командой время исполнения первой практически равно нулю.

4. Как лучше (с точки зрения быстродействия ЭВМ) организовать ввод чисел-констант в программе:

... каждый раз вписывая в нужное место программы операнд #X (например, MOV #1000,R0),

... составив таблицу констант и обращаясь к ней косвенно через регистр (например, MOV (R1)+,R0),

... составив таблицу констант, обозначив их метками и обращаясь к ним по меткам (например, MOV MET,R0)?

— Первым или вторым способом: они равнозначны по времени адресации, тогда как третий способ медленнее примерно в 1.5 раза. Но нужно учесть, что для обращения к таблице по второму способу необходимо еще, как правило, предварительно занести в RN адрес константы в таблице. Поэтому явно предпочтительнее, если констант не слишком много, просто напрямую указывать их в качестве операндов, например MOV #1250,R0 , а не составлять из них таблицы.

5. Пусть нужно вывести текст из N символов. Известно, что текст при убранном курсоре выводится быстрее. Начиная с какого значения N становится выгодно по времени убрать курсор, а потом его восстановить? Считайте, что команда гашения или восстановления курсора исполняется за 1.1 мс.

— Составим уравнение **1.7 × N = 2 ×1.1+ + 1.1 × N** и решим его относительно N. Получается, что N примерно равно 3.67. Ответ: выгодно убирать и затем восстанавливать курсор, если текст состоит из 4 и более символов, т. е. практически всегда.

★ ★ ★

Итак можно считать, что мы изучили в общих чертах архитектуру и ассемблер БК-0010. Что же дальше? Можно ли считать художником человека, который изучил химический состав красок, правила нанесения грунта на холст, способы изготовления рам для картин? Ясно, что нет. Пора, наверное, перейти к самим картинам, то бишь, к программам... И начнем мы с вопроса, которого уже касались, — с векторов прерывания, но теперь посмотрим на них не как теоретики, а с практической точки зрения: что можно сделать с их помощью (что делает ЭВМ — мы уже знаем).

Векторы прерывания и программирование

Большинство случаев практического использования сведений о векторах прерывания сводится к изменению этих векторов и созданию новых программ обработки прерываний. Например, таким путем можно запретить останов программы клавишей «СТОП», ввести дополнительные команды процессора, дополнительные символы клавиатуры, да и мало ли что еще.

Перейдем сразу к конкретным примерам. Предположим, что нам необходимо запретить остановку программы клавишей «СТОП», — это бывает необходимо во многих случаях, когда останов может привести к нежелательным последствиям. Поскольку клавиша «СТОП» внеприоритетная и просто запретить прерывание по вектору @#4 мы не можем, нам остается только изменить его содержимое и написать новую программу обслуживания прерывания. Включим в основную программу (где-нибудь в ее начале) такую команду: **MOV #V4,@#4** . С ее помощью сразу же после запуска мы перепишем вектор @#4. Теперь при нажатии на клавишу «СТОП» (и, равным образом, при отработке команды HALT) управление будет передаваться не по прежнему адресу, а на метку V4. Какую же программу там разместить? Это зависит от наших целей.

Если нужно только, чтобы клавиша «СТОП» вообще «не работала», достаточно, как может показаться, единственной команды возврата из прерывания RTI, ее выполнением обработка прерывания по вектору @#4 и закончится:

```
V4: RTI
```

Но не все так просто. Необходимо учитывать, что клавиша «СТОП» может быть нажата в совершенно произвольный момент и прерывает она программу не совсем так, как, скажем, команда EMT или нажатие клавиши на клавиатуре. При прерывании по клавише «СТОП» процессор может не закончить выполнение очередной команды, но при этом в стеке будет сохранено значение PC, соответствующее адресу следующей, с нее-то и продолжится выполнение программы после отработки нашего RTI. В зависимости от конкретных команд такой «финт» может привести к самым неприятным последствиям, вплоть до зависания системы. Как же тогда быть? Надо поступить хитрее. Необходимо предусмотреть в программе специальный, так называемый «холодный» вход (т. е. адрес, при передаче управления на который не произойдет нарушения работы программы). Впрочем, правильнее всего было бы называть его ПОВТОРНЫМ входом в программу. На этот вход и надлежит передать управление по клавише «СТОП», для чего его адрес указывается в качестве вектора прерывания. Пусть повторный вход в нашей программе (например, игре) — по метке RBG. Напишем:

```
V4: MOV #1000,SP
    JMP RBG
```

Вот и все. Вторая команда ясна — это просто передача управления на повторный вход. А зачем первая, что мы записываем в регистр SP и для чего? Вспомним, как выполняется прерывание: в стек заносятся для сохранения ССП и PC. При этом, конечно, изменяется и указатель стека SP — его значение уменьшается на 4. Команда RTI в конце программы обработки прерывания не только восстанавливает ССП и PC, но и «очищает» стек, приводя его указатель в исходное состояние. Но в нашей программе ее нет! Это значит, что при каждом нажатии «СТОП» стек будет «засоряться» все новыми значениями ССП и PC, а он не безграничен. Рано или поздно стек будет исчерпан, и тогда... Вот для предотвращения этого и служит первая команда — она восстанавливает вершину стека, «чистит» его. Число же 1000 — это начальное содержимое SP в мониторе, но ничего страшного обычно не происходит, если такая вершина стека будет установлена в любом режиме.

А теперь предположим, что нам нужно кроме перезапуска программы по «СТОП» сохранить и возможность выхода из нее. Как быть? Напишем такую программу обработки прерывания:

```
V4: MOV #1000,SP    Восстановить SP
    EMT 6           Ожидание нажатия клавиши
    CMPB #40,R0     Код  ПРОБЕЛ ?
    BEQ 0           Да - к метке 0
    JMP RBG         Иначе - повторный вход
0:  EMT 14          Восстановить векторы
    HALT            Останов и выход в ПМ
```

Первым делом мы «чистим» стек. Затем следует команда EMT 6 — ожидание ввода

символа с клавиатуры, его код будет записан в регистр R0. Если это код 40 — «ПРОБЕЛ», то должен последовать останов, для чего выполняется переход к метке 0. А что делает команда EMT 14? Она устанавливает состояние ЭВМ в «исходное положение», в том числе переписывает все векторы прерывания на стандартные адреса монитора. После этого команда HALT приводит к обычному результату — останову программы. Если же нажат не «ПРОБЕЛ», а иная клавиша, тогда перехода к метке 0 не будет и обработка прерывания закончится как предыдущая — перезапуском программы по метке RBG.

★ ★ ★

Довольно интересное применение может найти вектор @#10 — прерывание по резервному коду. Да что там может — уже нашел! В ФОКАЛе он использован для обработки псевдокоманд и функций. Эти псевдокоманды имеют коды в диапазоне 7000...7377. Такие коды отсутствуют в наборе команд ЦП, и при их обнаружении происходит прерывание по вектору @#10. А этот вектор в ФОКАЛе «направлен» на специальную программу интерпретатора этого языка, которая выделяет код, вызвавший прерывание, и в зависимости от него передает управление подпрограммам реализации команд ФОКАЛа. Совершенно аналогично можно организовать и выполнение специальных дополнительных команд в программе пользователя, лишь бы выбранные коды отсутствовали в наборе команд ЦП БК-0010. Так, в литературе (см. Микропроцессорные средства и системы. 1988. № 4. С. 47) описан ЭМУЛЯТОР (эмуляция — имитация функционирования одной системы средствами другой) команд расширенной арифметики для ЭВМ ДВК-2, основанный как раз на этом принципе. Он позволяет исполнять на ДВК-2 программы, написанные для СМ-4 и «Электроники-60», не прибегая к их АДАПТАЦИИ (переделке под другую систему команд)*.

★ ★ ★

А теперь займемся довольно сложным, но весьма интересным вектором @#14. Вам, наверное, приходилось отлаживать программы, и при этом, конечно, были случаи, когда вы никак не могли найти ошибку, а программа так и не желала работать как надо. При этом у вас возникало острое желание «заглянуть внутрь» работающей программы, посмотреть, в каком порядке выполняются команды и каковы результаты их действий. В большинстве языков высокого уровня такой ТРАССИРОВОЧНЫЙ РЕЖИМ предусмотрен. А для программ в кодах существуют специальные программы — ОТЛАДЧИКИ, позволяющие выполнять команды по одной. Как же они работают?

Если вы помните, при рассмотрении ССП (или PS) мы упоминали так называемый Т-разряд (бит 04 ССП). Если записать туда «1», то работа программы будет прерываться после выполнения каждой команды, по вектору @#14. Но ведь это как раз то, что нужно для отладки, тот самый трассировочный режим! Перепишем вектор @#14 и разместим по указанному в нем адресу программу, которая, например, будет выдавать на экран адреса и коды выполняемых команд, а также содержимое регистров, значение ССП и пр. Прекрасно, давайте попробуем:

MTPS #20

Что получается? Ничего — этой командой Т-разряд не изменить! А если бы она записала в Т-разряд «1», что бы произошло? Работа программы сейчас же прервалась бы по вектору @#14, а из его второго слова было бы переписано ССП с нулевым Т-разрядом. Но даже если занести и туда

* Следует отметить, что в ФОКАЛе вектор @#10 также использован прежде всего для эмуляции команд плавающей арифметики. Аналогичная система реализации расширенной арифметики имеется и в ПЗУ КНГМД, о чем можно прочитать в № 5 за 1994 г. — *Прим. ред.*

код 20, все равно ничего хорошего не получится — нам нужно прерывание не после записи Т-разряда, а после выполнения очередной команды ОТЛАЖИВАЕМОЙ программы. Нужно найти другой путь, чтобы, как только будет установлен Т-разряд, сразу начала выполняться очередная команда. А как только она будет выполнена — произойдет прерывание по вектору @#14, и мы сможем узнать о только что выполненной команде то, что хотим. Как это сделать? Как записать Т-разряд в нужный момент, «на самом краю» запуска очередной команды?

Вспомним, как выполняется возврат из прерывания. Из стека восстанавливается сначала PC, а затем — ССП: то значение PS, которое было занесено в стек при переходе к обработке прерывания, пишется обратно в регистр PS. А потом? Потом начинается выполнение следующей команды программы... Ага, кажется, именно это нам и нужно! Что, если перед возвратом из прерывания «подменить» сохранение в стеке значение ССП на такое, в котором установлен Т-разряд? Процессор его «скушает», не подозревая «подвоха», и выполнит очередную команду, после чего проверит Т-разряд и обнаружит, что там «1». Ему ничего не останется, кроме как выполнить прерывание по вектору @#14. А тут-то мы его и будем ждать со своей программой, чтобы «подсмотреть» то, что нам нужно... Кажется, путь найден! Но сначала нужно организовать прерывание, чтобы при выходе из него «обмануть» ЦП. По какой команде это сделать? В принципе, можно по любой, лишь бы это было именно командное прерывание. Но в наборе команд есть специальная операция BPT, которая вызывает прерывание по тому же вектору @#14 (но не отладочное по Т-разряду, а просто командное, на манер EMT). Воспользуемся ею и не будем трогать другие векторы. А что мы будем делать во время «отладки»? Что хотим «подсмотреть»? Прежде всего, нам нужно определить, какая команда выполняется, а для этого желательно узнать ее адрес и код (пока мы не будем замахиваться на то, чтобы дизассемблировать команды — переводить их на язык ассемблера, — это чересчур сложно). И пусть нас еще интересует, к примеру, содержимое одного из регистров — R1. А затем напишем какую-нибудь программку, которая будет работать именно с R1, и посмотрим, что получится. Желательно также иметь хотя бы минимальный «сервис» — хоть как-то маркировать выводимые данные, чтобы можно было понять, где адрес, где код команды, а где — содержимое регистра. А как переходнть к следующей команде? Пусть для простоты «отлаживаемая» программа выполняется тогда, когда нажата любая клавиша, а если клавишу отпустить, можно просмотреть данные, выдаваемые «отладчиком». Итак, «техническое задание» на программный продукт готово. Осталось написать программу, а «что нам стоит дом построить», да еще из такого податливого «материала» как операторы ассемблера? Вот наш простейший «отладчик» и готов:

```
; Запуск отладчика
    MOV #BPT,@#14       Записать пусковой вектор @#14
    BPT                 Прерывание по вектору @#14

; Демонстрационная программа
    MOV #1,R1           Записать в R1 единицу
0:  ASL R1              Сдвиг влево R1
    ADC R1              Прибавить перенос
    BR 0                Программа зациклена

; Трассировка демонстрационной программы
```

```
TRS: MOV R0,-(SP)          Сохранить в стеке
     MOV R3,-(SP)          регистры,
     MOV R4,-(SP)          используемые
     MOV R5,-(SP)          в "отладчике"
     MOV 'A',R0            Сообщение "АДРЕС"
     EMT 16                выдать
     MOV '=', R0           на
     EMT 16                экран
     MOV 10(SP),R3         Извлечь из стека PC в R3
     MOV R3,R4             Печать адреса
     CALL @#163220         команды
     MOV 'K',R0            Сообщение "КОД"
     EMT 16                выдать
     MOV '=', R0           на
     EMT 16                экран
     MOV @R3,R4            Печать кода
     CALL @#163220         команды
     MOV 'R',R0            Сообщение "РЕГИСТР"
     EMT 16                выдать
     MOV '=', R0           на
     EMT 16                экран
     MOV R1,R4             Печать
     CALL @#163220         содержимого R1
     MOV #12,R0            Перевод
     EMT 16                строки
0:   BIT #100,@#177716     Клавиша нажата?
     BNE 0                 Нет - ждать, иначе продолжать
     MOV (SP)+,R5          Восстановить
     MOV (SP)+,R4          регистры
     MOV (SP)+,R3          из
     MOV (SP)+,R0          стека

; Запись T-разряда, выход из прерывания
BPT1:BIS #20,2(SP)         Занести T-разряд в ССП в стеке
     MOV #TRS,@#14         Записать рабочий вектор @#14
     RTT                   Выход из прерывания

END
```

Разберемся с некоторыми деталями. Сначала мы пишем в качестве вектора @#14 адрес метки BPT1 и даем прерывание BPT. Происходит переход к метке BPT1, а там мы заносим в ССП, хранящееся в стеке (оно на 2 выше вершины, ведь мы в прерывании!), единицу в Т-разряд. Затем записываем новый вектор @#14 («рабочий», т. е. тот, по которому будет передаваться управление при прерывании

по Т-разряду), после чего следует команда возврата из прерывания, но не RTI, а RTT. Она незначительно отличается от RTI (мы уже упоминали, чем именно: после ее выполнения не происходит прерывания по Т-разряду) и применяется для выхода из отладочного прерывания.

Теперь запускается наша демонстрационная программа. Она зациклена и делает следующее: сначала в R1 пишется 1, затем R1 сдвигается влево. Когда «1» достигнет старшего разряда R1, возникает перенос, но бит переноса каждый раз прибавляется к R1, поэтому единица снова попадет в младший разряд, опять начнет сдвигаться, и так до бесконечности.

После выполнения каждой команды происходит прерывание по Т-разряду (ведь наш «отладчик» включен!) и управление передается на метку TRS — «Трассировка».

Вот теперь в работе отладчик. Прежде всего, трассировки в нем самом уже нет — ведь после прерывания в ССП переписалось новое значение из адреса @#16 — второго слова вектора. Мы не знаем, что там было, но в данном случае это неважно: Т-разряд там равен нулю, будьте уверены! И отлично, иначе отладчик не смог бы работать, а прерывал бы сам себя после каждой команды. В отладчике мы будем использовать часть регистров, а чтобы не возникло потом неприятностей, сохраним их в стеке (в данном случае это не важно, но для других отлаживаемых программ пригодится). Сначала будем выдавать адрес команды. Даем сообщение «A=» — это наш «сервис», а потом... Вспомним, что у нас сейчас хранится в стеке. По порядку туда было записано: ССП, PC, R0, R3, R4, R5. Ага, PC нам и надо, ведь там адрес. Но какой? Адрес следующей команды. Что же, для простоты давайте индицировать как раз следующую команду, это даже удобнее — мы сначала посмотрим, что «там, за поворотом», а потом уже разрешим ей работать (кстати, именно так обычно и делается в настоящих отладчиках). Итак, PC в стеке на 8 байт от вершины (после него в стек мы «загнали» еще 4 регистра). Применяем косвенную индексацию: 10(SP). Мы «достали» значение PC, теперь надо его выдать на экран. Но нам он еще пригодится, поэтому не пожалеем для него отдельного регистра R3. А как выдать его на экран? В МСД есть подпрограмма, которая делает то, что нам нужно, — выдает на экран содержимое регистра R4. Чтобы не писать ее заново, просто обратимся к ней по адресу @#163220. Она не сохраняет регистры R0, R4 и R5, вот почему мы их оставили в стеке вместе с R3. Итак, число (значение PC) сначала копируется в R4, потом выводится на экран. Готово, поехали дальше! Сервис: «K=», это будет код команды. Вот теперь нам пригодится R3, ведь там адрес, а по этому адресу — сама команда. Словом, косвенное обращение через регистр. В R4 его и на экран! Ну, а теперь совсем просто: содержимое R1, которое мы хотим «подсмотреть», гоним туда же, но сначала опять сервис: «R=». Выведена строка параметров: адрес, код команды, содержимое R1. Чтобы не запутаться, переведем строку. Что мы еще хотели? Ах, да! Если клавиша нажата — программа работает, если нет — ждет. Регистр @#177716 позволяет решить и эту задачу. Вот мы нажали любую клавишу, и программа пошла дальше: восстановлены регистры, и... а дальше мы уже были — снова заносим вектор @#14 и Т-разряд (можно бы этого и не делать, но лучше перестраховаться) и возвращаемся из прерывания. Вперед, нас ждет следующая команда!

Выглядит не слишком сложно, не правда ли? А ведь наша программа имеет практически все основные черты настоящего отладчика. Конечно, есть масса деталей, которых мы еще не учли. Например, если команда отлаживаемой программы состоит из двух или трех слов, то будет выдано только ее первое слово. А если в отлаживаемой программе встретится EMT 14, то перепишутся все векторы (и 14-й в том числе), и прощай отладочный режим! И не все, что надо, мы видим на экране, и нет точки останова, и нет дизассемблера, и наш отладчик неперемещаемый, и... Ну ясно, а что же вы хотели? Чтобы, едва освоив ассемблер, сделать настоящий отладчик? Но главное — понять, как он работает, и эта цель достигнута. А дальше дело за вами.

Но прежде чем окончательно проститься с вектором @#14, рассмотрим еще один вопрос. При пользовании отладчиком (простейшим или самым сложным) любое командное прерывание, будь то EMT, TRAP, IOT или другое, отрабатывается в пошаговом режиме как одна команда. Это естественно — ведь при обращении к прерыванию переписывается ССП и Т-разряд сбрасывается, а с ним на время обработки прерывания отключается и трассировочный режим. Поэтому мы видим на экране просто «EMT 32» или «TRAP 114», не более. Но мы-то теперь знаем, что за такой скромной «вывеской» могут скрываться сотни команд... Как бы их «подсмотреть», увидеть, как работает EMT-диспетчер, где находится и что делает программа обработки команды IOT? Неужели ничего нельзя сделать? Можно. Достаточно переписать второе слово интересующего нас вектора прерывания, занеся туда Т-разряд, и прерывание это станет «видимым» для отладчика, т. е. и внутри него будет обеспечиваться пошаговый режим и индикация всего, что нас интересует.

★ ★ ★

Следующий по порядку вектор прерывания — @#20. Прерывание по нему возникает по команде IOT и аналогично командным прерываниям BPT, EMT и TRAP, только, в отличие от двух последних, не имеет аргументов. Оно может быть использовано для выполнения каких-нибудь действий. Допустим, нам нужно подать в определенный момент звуковой сигнал. В начале программы запишем: **MOV #IOT,@#20** — это мы переписали вектор @#20. Напишем теперь программу обработки прерывания:

```
IOT: MOV #7,R0     Код звукового сигнала
     MOV #100,R1   100 повторов (64д)
0:   EMT 16        Выдать звук (щелчок)
     SOB R1,0      Цикл по R1
     RTI           Выход из прерывания
```

Теперь в любой момент можно вызвать звуковой сигнал одной командой IOT. С равным успехом она же может применяться и для любых других целей. Вообще же команда IOT предназначена для вызова процедур и обработки ошибок в операционных системах, и по этой причине ее лучше не трогать «по мелочам», иначе при работе ваших программ с дисководом (если он у вас имеется) могут возникнуть проблемы. Впрочем, то же может быть отнесено к командам TRAP, EMT, BPT, к использованию свободных ячеек в системной области... Вообще, лучше ничего не трогать! И не писать программы — тогда и проблем не будет... А если серьезно, то в каждом конкретном случае нужно точно знать, что «трогать», а что — нет. (Обычно такие сведения — об используемых ячейках и ограничениях на изменение их содержимого — приводятся в документации к операционным системам, отладчикам, мониторам и т. д. — *Прим.ред.*)

★ ★ ★

Вектор @#24 для нас практически бесполезен, так как на БК-0010 он не имеет так называемой АППАРАТНОЙ ПОДДЕРЖКИ. Но можно убедиться, что программа его обработки в ПЗУ имеется. Дайте в МСД команду 160714G — и вы получите ответ «СБОЙ ПИТАНИЯ». Ну а если все-таки вспомнить, что прерывание по вектору @#24 возникает при подаче сигнала на вход ACLO ЦП? Если не побояться залезть в БК с паяльником? Да, кое-чего мы можем достигнуть. Если подать на вывод 13 микросхемы D 6.4 (К561ЛЕ5) положительный импульс, то произойдет прерывание по вектору @#24! Подать такой импульс можно, если сначала зарядить конденсатор небольшой емкости (0.03...0.05 мкФ) до напряжения +5 В, а затем разрядить его на вывод 13 D 6.4 (есть, впрочем, и другие способы). Чего мы достигнем? Прерывание ACLO — внеприоритетное, аналогично клавише «СТОП», и, переписав вектор @#24, мы можем нажатием дополнительной кнопки обратиться по этому вектору. А там, например, может быть программа графического копирования экрана. Или программа инициализации принтера. Или что-нибудь еще... Но тут пора сказать, что автор категорически не рекомендует лезть в БК с паяльником тем читателям, которые берут

его в руки не чаще, чем раз в месяц, да и то, чтобы напаять «жучок» на предохранитель. И никакой ответственности за их действия автор (как и редакция) на себя брать не собирается.

★ ★ ★

Теперь рассмотрим последний из оставшихся векторов прерывания — @#100. Это, как уже было сказано, прерывание по таймеру. Таймер имеется в виду ВНЕШНИЙ — это подключаемое к БК устройство, выдающее импульсы строго определенной частоты, например 10 импульсов в секунду. Импульсы, подаваемые на вход таймера (контакт В1 порта ввода-вывода), должны иметь уровень ТТЛ-логики («0, или НИЗКИЙ УРОВЕНЬ — от 0 до 0.4 В; «1», или ВЫСОКИЙ УРОВЕНЬ — от 2.4 до 5 В). При прохождении на вход таймера ЗАДНЕГО ФРОНТА импульса (т. е. когда «единица» сменяется «нулем») возникает прерывание по вектору @#100 (если, конечно, его разрешает приоритет ЦП). Как можно использовать этот вектор? Можно по прямому назначению, для отсчета времени. Но, кроме того, это дополнительный канал связи БК с внешним миром, и его применения могут быть различными. Если, допустим, у вас есть магнитофон, в котором имеется счетчик, выдающий импульсы при вращении кассеты, можно подать эти импульсы (конечно, преобразовав их к нужному уровню!) на вход таймера и, подсчитывая их, получить программный счетчик расхода ленты. Правда, тут есть одна неприятность — при записи на магнитофон или чтении прерывания должны быть запрещены, иначе информация будет искажена. Но можно сделать счетчик, работающий только при перемотке, и таким образом автоматизировать хотя бы поиск файлов (разумеется, для этого магнитофон должен иметь полное дистанционное управление, и вообще это не так просто). А можно подключить ко входу таймера, например, АНАЛОГО-ЦИФРОВОЙ ПРЕОБРАЗОВАТЕЛЬ (АЦП) типа ПНЧ — преобразователь напряжения в частоту и, пользуясь системным (встроенным) таймером, измерять число поступивших импульсов в единицу времени, а затем вычислять измеренное напряжение (или иной параметр). Или можно сделать так, чтобы ЭВМ при поступлении на вход В1 сигнала считывала информацию с внешних устройств (например, с микрокалькулятора или с цифрового измерительного прибора) или, наоборот, выдавала информацию на порт ввода-вывода либо на ТЛГ-канал. Приведем элементарный пример обработки прерывания по вектору @#100.*

```
; Инициализация программы
    CLR R1              Очистить счетчик
    MOV #TIM,@#100      Записать вектор @#100
    HALT                Конец

; Счет импульсов с индикацией
TIM: INC R1             Счетчик увеличить на 1
    MOV R1,R4           Выдать текущие
    CALL @#163220       показания на экран
    RTI                 Возврат из прерывания

END
```

* Прежде чем запускать эту программу, нужно подключить внешний таймер к контакту В1 порта ввода-вывода, иначе на БК ничего выполняться не будет. — *Прим. ред.*

Разумеется, в зависимости от наших целей программа обработки прерывания может быть и другой. Максимальная частота подаваемых на вход таймера импульсов ограничена временем, затрачиваемым на обработку каждого прерывания, и может при простейших подпрограммах обработки достигать 20—30 кГц. Только надо учитывать, что при этом скорость обработки компьютером текущей задачи резко снижается, ибо почти все время уходит на обработку прерываний.

Контрольные вопросы и задания

1. Придумайте способ, который бы позволил обращаться с помощью команды IOT не к одной программе обработки, а к нескольким, как EMT.

— Нужно после IOT записать в программу аргумент, например, так:

```
IOT
.#122
```

Программа же обработки прерывания должна содержать своеобразный диспетчер, который прочитает аргумент из памяти, выполнит по нему нужную программу, а перед возвратом из прерывания увеличит в стеке содержимое PC на 2, чтобы «обойти» аргумент и не передать на него управление. Про то, что команду IOT лучше вообще не трогать, уже говорилось, но это не причина чтобы лишить читателя удовольствия поломать голову.

2. Каково «нормальное» содержимое вектора прерывания @#4 в ПМ, МСД, ФОКАЛе и БЕЙСИКе? Предложите способы, которые бы позволили это узнать.

— Содержимое вектора @#4: 100442, 160722, 121110, 120234. Узнать его можно, если прочитать ячейку @#4. В МСД это делается непосредственно, в БЕЙСИКе и ФОКАЛе — с помощью средств этих языков (функции PEEK и FX соответственно), а в ПМ придется написать и запустить программу:

```
MOV @#4,R4
CALL @#163220
HALT
END
```

А что же векторы прерываний, «жизненно необходимые» самой ЭВМ, — прерывания от клавиатуры @#60 и @#274, а также уже не раз рассмотренные векторы командных прерываний @#30 и @#34? Не забыл ли о них автор? Нет, просто эти векторы недостаточно рассмотреть на отдельных примерах, как другие. Работа с ними может привести к таким последствиям, к таким ошеломляющим на первый взгляд возможностям и результатам, что их лучше оставить для отдельной темы, к рассмотрению которой мы теперь и переходим.

Концепция системных драйверов

Жил-был... нет, не серенький козлик. И не у бабушки. Был у пользователя редактор текста. Неплохой, в общем, редактор EDASP. И был к нему принтер, для которого в редакторе был драйвер. И все было хорошо, пока пользователю не встретился новый редактор, скажем МИКРО.10К. Он ему понравился, но вот беда — в этом редакторе нет драйвера принтера! Что делать — лезть в МИКРО.10К и пытаться «вставить» в него драйвер? На этот «подвиг», прямо скажем, способны немногие. Перекодировать тексты, написанные в МИКРО (благо программы «конверторов» текста есть), и печатать через EDASP? Ужасно неудобно. А если завтра понравится еще один редактор? Все проблемы решать заново?

Или вот на БК-0010 есть язык ФОКАЛ. Многие его любят, несмотря на низкую скорость и точность, — он очень гибок. (Попробуйте-ка в БЕЙСИКе обратиться к строке, задав ее номер в виде переменной или выражения, — а в ФОКАЛе это самая заурядная процедура. Или попытайтесь записать на МЛ саму программу программным путем, а не в диалоговом режиме. А ФОКАЛ это может!) К тому же этот язык очень экономно расходует память. Например, на ФОКАЛе есть программа, которая... сочиняет рассказы о Штирлице! (На полном серьезе, автор нисколько не шутит.) И причем довольно смешные рассказы! О возможности таких программ (но

только о возможности!) пишет Д. Кнут в своей знаменитой трехтомной книге «Искусство программирования на ЭВМ», а на БК — пожалуйста, вот она, эта программа. Вот бы распечатать «сочиненный» рассказ на бумаге! Но увы, в ФОКАЛе не предусмотрено выхода на принтер...

Или у вас есть принтер и есть БЕЙСИК. В БЕЙСИКе, конечно, есть специальные операторы для вывода на печать. Но, к сожалению, ваш принтер и БЕЙСИК взаимно несовместимы — формат выдачи БЕЙСИКа «не нравится» принтеру или наоборот. Причем принтер этот, как ни странно, работает с некоторыми редакторами текста, но, как назло, с теми, с которыми не любите работать вы... Что делать — покупать за большие деньги электронную «приставку» к принтеру для перевода его протокола обмена в стандартный формат? А для редактора текста еще одну приставку? А для того, чтобы распечатать красивую заставку к игре, что купить? Фотоаппарат?!

Или вот вы живете в «некотором царстве» (попросту говоря, в республике). Сейчас все вдруг вспомнили, что раз республика суверенная, то надо пользоваться национальным языком. Но вот беда — в школах, институтах и прочих заведениях «компьютерное население» сплошь «русскоязычное» (ну, конечно, еще и англоязычное, но ведь мы не в Лондоне или Брайтоне!). Как научить компьютер работать с национальным алфавитом, буквы которого зачастую отличаются от русского и английского по начертанию? Вы скажете, есть специальные редакторы текста, в которых предусмотрен спецалфавит? Но как быть с БЕЙСИКом и ФОКАЛом? Писать текстовые сообщения в программах по-русски? Или не использовать компьютер для этих целей?

Вот какие могут быть разные задачи. А решаются они, как ни странно, в принципе, все одинаково — с помощью СИСТЕМНЫХ ДРАЙВЕРОВ. Что же это такое — драйвер и чем отличается системный драйвер от обычного?

ДРАЙВЕР (от английского «driver» — «водитель») — это программа, управляющая другими программами или взаимодействующая с внешними устройствами ЭВМ. А что мы понимаем под системным драйвером?

Пусть нам нужно ввести в редактор текста возможность печати на принтере. Обычный путь таков: мы включаем в редактор дополнительную директиву, например «печать от метки до курсора», и по ней редактор, вместо того чтобы выдать, как обычно, текст на экран, выдает заданную часть текста в виде символьного кода на принтер (соблюдая формат обмена с принтером, разумеется). Но возможен и другой путь. Как выводится информация в редакторе текста на экран? Очевидно, через EMT 16 (может быть, и через EMT 20, но это менее вероятно). Заменим вектор @#30 и напишем свою программу его обработки («супервизор»):

```
; Инициализация драйвера
        MOV PC,R0           Вычислить адрес супервизора
        ADD (PC)+,R0        в перемещаемом
        .@EMT+2             формате
        MOV R0,@#30         Записать новый вектор @#30
        HALT                Стоп

; Супервизорная программа
EMT:    MOV R5,-(SP)        Сохранить R5 в стеке
        MOV 2(SP),R5        Адрес следующей команды
        CMP #104016,-(R5)   Текущая команда - EMT 16?
        BNE 3               Нет - выход
```

```
        MOV R0,-(SP)          Сохранить R0
1:      BIT #400,@#177714     Принтер готов?
        BEQ 1                 Нет - ждать
        BIS #400,R0           Строб-бит
        MOV R0,@#177714       Код символа и строб - в порт
2:      BIT #400,@#177714     Код принят?
        BNE 2                 Нет - ждать
        BIC #400,R0           Очистить строб-бит
        MOV R0,@#177714       Код в порт
        MOV (SP)+,R0          Восстановить R0
3:      MOV (SP)+,R5          Восстановить R5
        JMP @#100112          Выход в EMT-диспетчер
```

Что делает эта программа? Сначала она проверяет, какая команда вызвала прерывание по вектору @#30. Если это EMT 16, то надо с ней работать. А если другая — EMT 20, EMT 22, EMT 36? Тогда просто восстановим регистр R5, который использовался нами для служебных целей, и скатертью дорога в стандартный EMT-диспетчер БК-0010, пусть он сам с ней разбирается. А как опознать EMT 16? Конечно, по ее коду — 104016. А как найти код? Очень просто — мы в прерывании, адрес следующей команды — в стеке (за 2 байта от вершины), а код предыдущей — в памяти, за 2 от следующей...

Когда же команда EMT 16 опознана, мы вспоминаем, что у нас сейчас в младшем байте R0 записан код символа, выводимого на экран. А мы вместо экрана выдадим его на принтер! Понятно, что надо соблюдать протокол обмена — сначала дождаться сигнала готовности принтера (стандартно он подключен к разряду 08 порта ввода), затем записать строб-бит (сигнал принтеру: «Печатай!», он подается на разряд 08 порта вывода) и сразу же выдать на восемь младших разрядов порта вывода код символа. Теперь — внимание! Принтер должен принять код и начать его печатать. При этом он снимает сигнал готовности, сигнализируя, что занят другим делом и не может принять следующий код. Дождемся этого момента и снимем бит стробирования, теперь уж ясно, что принтер код принял и нам можно уходить восвояси за следующим кодом. А «чтобы чего не вышло», мы при входе в программу сохраняем регистр R0, а потом его восстанавливаем — ведь в его старшем байте, в принципе, тоже могла быть информация, которую мы испортили строб-битом. Теперь все в порядке, «словно ничего и не было», и можно передать управление в EMT-диспетчер, пусть он выведет символ на экран как обычно. Вот и все.

Конечно, программа печати и протокол обмена с принтером заведомо выбраны самые примитивные. Реальный принтер может принимать данные в прямом коде (а порт у нас — инвертирующий), тогда придется ввести команду COM R0. Или может потребоваться не одновременная подача строба и данных, а последовательная — сначала одно, потом другое. Вполне возможно также, что кодовая таблица принтера не совпадает с принятой на БК-0010 частично или полностью, тогда до подачи кода символа на принтер его надо перекодировать — привести в соответствие с кодом, требуемым принтеру. Но все это зависит от конкретного принтера и суть дела не меняет. А суть эта в том, что мы, не меняя программу пользователя, «сели» на нее своим драйвером и выводим информацию совсем не туда (или не только туда), куда она должна была выводиться. Это и есть (в нашем определении) системный драйвер, иначе называемый СУПЕРВИЗОРНЫМ драйвером или просто супервизором. (Для справки: супервизорная программа — машинная программа, являющаяся частью операционной системы, которая управляет выполнением других машин-

ных программ и регулирует поток работ в системе управления данными.)

А что если в программе пользователя имеется, например, команда EMT 14? Она ведь перепишет все векторы прерывания и тем самым «сбросит» супервизорный драйвер с системы. Нам нетрудно такую команду «поймать» до исполнения, ведь код ее известен — 104014 — и мы можем проверить его, как и 104016. Но как быть дальше? Если мы ее просто проигнорируем, то тем самым можем нарушить работу программы пользователя — ведь EMT 14 там появилась, наверное, не зря! Надо ее выполнить, но так, чтобы «не выпустить вожжи из рук». Сделаем это, переписав нашу программу начиная с метки «3»:

```
3:  CMP #104014,@R5       Текущая команда - EMT 14?
    BNE 4                 Нет - выход в EMT-диспетчер
    MOV @#30,-(SP)        Сохранить вектор @#30 в стеке
    MOV #100112,@#30      Записать "нормальный" вектор @#30
    EMT 14                Исполнить EMT 14
    MOV (SP)+,@#30        Восстановить вектор @#30
    MOV (SP)+,R5          Восстановить R5
    RTI                   Выход из прерывания

4:  MOV (SP)+,R5          Восстановить R5
    JMP @#100112          Выход в EMT-диспетчер
```

Вот так! Сначала восстановили обычный вектор @#30 (иначе по EMT 14 система снова выйдет на супервизор), а «супервизорный» вектор сохранили в стеке, ведь EMT 14 стек «не трогает». Потом выполнили EMT 14, и теперь можно восстановить «наш» вектор @#30 из стека. Атака отбита, «сбросить» драйвер программе пользователя не удалось! Но изменилась наша программа и в другом — мы уже не можем передать управление в EMT-диспетчер, ведь тогда он еще раз выполнит EMT 14, которое мы только что «обезвредили», так что закончим прерывание сами командой RTI*.

Теперь уже легче сообразить, как можно печатать на принтере не только по EMT 16, но и по EMT 20, — вспомним, какая должна быть исходная информация для этого прерывания.

В R1 — адрес начала строки текста, в R2 — признаки конца: длина строки или код последнего ее символа (а может быть, и то и другое). Конечно, это задача посложнее — надо не только вывести строку на принтер, но и выдать ее посимвольно на экран, а потом привести регистры R1 и R2 в такое состояние, как будто команда EMT 20 сработала без всяких фокусов. Выдавать символы придется через EMT 16, а весь этот процесс будет называться эмуляцией команды EMT 20. Мы не приводим соответствующую программу, а предоставляем сделать это читателю — ведь наша книга не справочник программ, ее задача — научить читателя самостоятельно эти программы писать. Привести готовые программы драйверов, эмуляторов и т. п. было бы куда как просто!

А как быть, если наш драйвер должен работать с языком высокого уровня, например с ФОКАЛом? Придется в него вве-

* Вместо восстановления стандартного значения @#30 и вызова EMT 14 можно просто обратиться к реализующей эту функцию подпрограмме в ПЗУ: JSR PC,@#100742. После возврата из нее нужно, как и в приведенном выше примере, вернуть в вектор @#30 «супервизорное» значение и выйти из прерывания по RTI.

сти «повторный выход» в ФОКАЛ. Вернитесь в раздел, посвященный МСД.

Там сказано, как выйти в ФОКАЛ без стирания ОЗУ. Если вы запомнили, что делают директивы МСД, вам не составит труда написать такую же программу на ассемблере. А чтобы в ФОКАЛе можно было нормально работать в присутствии драйвера, нужно драйвер защитить — сделать недоступной для ФОКАЛа зону памяти, где он будет размещен. Для этого достаточно перед выходом в ФОКАЛ записать адрес начала драйвера в ячейку @#1744 — эта ячейка в ФОКАЛе «отвечает» за зону ОЗУ, доступную интерпретатору. Естественно, найти адрес начала нужно с помощью псевдооператора «.@», чтобы не нарушить условия перемещаемости нашего драйвера. Вот теперь все. Разве что надо будет ввести еще запрос выхода из инсталлятора драйвера. Если в ПМ или МСД, то в системную область не нужно ничего писать, как это делается для выхода в ФОКАЛ, чтобы ничего не напортить. Тогда одна и та же программа-драйвер будет одинаково успешно работать в ФОКАЛе и в кодах, например в МСД.

А БЕЙСИК? С ним все так же, только надо знать, что он выдает свои тексты (все, кроме сообщения «Ok.») не по EMT 16, а по EMT 20, так что без эмуляции этой команды не обойтись. А специальный выход в БЕЙСИК не обязателен — можно ведь загрузить и запустить драйвер прямо в БЕЙСИКе, определив его как пользовательскую функцию в кодах, а затем к ней обратиться, например, так (драйвер размещен по адресу 30000):

```
DEF USR=&O30000
? USR(A)
```

После этого достаточно дать команду на выход в коды — БЕЙСИК выдаст, конечно, сообщение «СТОП» и будет работать как ни в чем не бывало дальше, но теперь все, что выдается на экране, будет печататься и на принтере.

Но постойте! Ведь это совсем не то, что надо! Нам нужно печатать не все, а только часть, и не всегда, а только тогда, когда это нужно! Как же быть? Мы этого, кажется, не учли? Ну что же, учтем. Ведь у нас много «свободных» кодов, а код символа в нашей программе всегда можно проверить. Введем какую-нибудь ячейку-«переключатель» (например, «на самом дне» стека по адресу @#374) и будем делать следующее:

- инициализируя драйвер, всегда сбрасываем ячейку,
- символ на печать выдаем только тогда, когда в ячейке не нуль,
- независимо от того, печатается ли символ, код его всегда проверяем, и если он равен «ключу», инвертируем нашу ячейку-«переключатель».

Понятно? Пусть наш «ключевой» код — 1, что соответствует клавише «СУ/А» или команде БЕЙСИКа **?CHR$(1)** (а также команде ФОКАЛа **X FCHR(1)**). Мы всегда можем выдать этот код как в диалоговом режиме, так и в программном. Содержимое ячейки @#374 проинвертируется, станет не нулевым, и символы будут выводиться не только на экран, но и на принтер. А если нам это не нужно — снова дадим тот же ключ, содержимое ячейки станет равным нулю и печать прекратится. (Каждый может придумать такой порядок управления печатью, какой ему больше нравится, и написать программу тоже по своему вкусу. А автор приводить полную программу, как уже было сказано, не собирается, так как считает, что в этом случае незачем было писать книгу — различных драйверов сегодня тиражируется достаточно.)

Вот теперь наш драйвер как будто полностью готов и можно с ним работать. Каковы его слабые места? Первое — это то, что супервизорная программа может быть «затерта» данными или текстом программы пользователя (ЭВМ тогда «зависнет»), поэтому о ее присутствии в ОЗУ программист всегда должен помнить. А второе — программа пользователя все-таки может «сбросить» супервизорный драйвер, отключить его, если она перепишет вектор @#30 напрямую, без всякого EMT 14 — тут уж ничего не поделаешь. К счастью, программ, изменяющих вектор @#30, не так уж много.

А как быть, если нам нужна графическая копия экрана? Нужно написать специальную подпрограмму графического копиро-

вания. Мы могли бы ее привести, но ведь почти для каждого принтера она будет своя, отличная от других, да это и не входит в нашу задачу. В общих чертах подпрограмма эта будет выглядеть так. Сначала нужно перевести принтер в графический режим (в соответствии с его системой команд), а затем выдавать на него и печатать по нескольку (в соответствии с его же графическим форматом) точек экранного ОЗУ. Достигнув конца графической строки экрана, надо перейти к следующей и т. д., а распечатав весь экран — выйти из подпрограммы. Можно построить графическую печать и таким образом, чтобы копия экрана выводилась как в «нормальном» положении, так и с поворотом на 90° — каждый способ имеет свои преимущества. А обращаться к данной подпрограмме можно из того же драйвера, только по иной команде-ключу, например по коду 2.

Здесь есть еще одна существенная деталь. Если вы хотите графически копировать экран «из любого исходного положения», то следует помнить, что он в БК-0010 организован по типу рулона, и его начало в каждом конкретном случае может быть с любого адреса. Определить адрес начала экрана нетрудно — в этом может помочь, например, регистр @#177664 или, еще лучше, ячейка @#204. Но если пытаться распечатывать фрагмент ОЗУ с начала экрана и длиной 40000, то мы рано или поздно «выскочим» в область ПЗУ (если только адрес начала рулона также не равен 40000). Нужно своевременно зафиксировать этот момент и скорректировать адрес, перейдя от конца ОЗУ экрана к его началу. Похоже, это довольно сложная задача и придется-таки привести пример листинга. Программа будет рассчитана на довольно распространенный принтер типа МС-6312, на его систему команд и на его формат графического вывода. Копироваться экран будет в «нормальном» положении, т. е. горизонтально на листе бумаги, а с целью увеличения размеров изображения и компенсации искажений, возникающих при переходе от формата экрана к формату принтера, высота по вертикали «удваивается» — на данном типе принтера это дает наиболее приемлемые результаты. Кроме того, такой вариант программы можно считать самым сложным из всех возможных, и если вы разберетесь, как работает эта подпрограмма, то сможете написать аналогичную и для другого типа принтера. Понять же, как она работает, вам помогут предшествующие разъяснения и построчные комментарии. Формат подпрограммы — перемещаемый, она сопровождается двумя вспомогательными подпрограммами и двумя строками команд принтера. Система команд МС-6312 близка к EPSON-стандарту. Для тех, кто не знаком с описанием этого принтера, поясним коды команд:

- строка K1 — 30,33,100,7, что означает:

30 — отмена строки, т. е. сброс «мусора» (CAN);

33 — регистровый код (ESCAPE, ESC);

100 — «@», ESC-код инициализации принтера;

7 — признак конца командной последовательности;

- строка K2 — 33, 63, 30, 12, 33, 113, 0, 2, 7:

33 — ESC;

63 — символ «3», ESC-код выбора N/216д-дюймового интервала между строками;

30 — число N в обозначении «N/216д-дюймовый интервал», т. е. устанавливается интервал 24д/216д;

12 — перевод строки (LF);

33 — ESC;

113 — символ «K» (лат), выбор графического режима одинарной плотности;

0, 2 — количество точек в графической строке режима «K», что значит (0 + 256д × 2) = 512д точек, это формат экрана БК-0010;

7 — признак конца командной последовательности.

А теперь сама подпрограмма:*

```
; Подпрограмма графического копирования экрана
GRK: JSR R4,@#110346      Сохранить
     MOV R5,-(SP)         регистры
     CALL KM1             Инициализация принтера
     MOV @#204,R1         Адрес
     ADD #36000,R1        начала экрана
     MOV #100,R5          Цикл из 64д строк по 4 точки
0:   CALL KM2             Перевод строки на 24д/216д дюйма
     MOV #100,R4          Цикл из 64д знакомест
1:   MOV #10,R3           Цикл из 8 столбцов точек
2:   MOV #4,R2            Цикл из 4 вертикальных точек
     CLR R0               Очистить буфер граф.элемента
3:   ASLB R0              Сдвиг буфера
     ASLB R0              на 2 точки
     RORB @R1             Очередная точка = 0 ?
     BCC 4                Да - следующая точка
     BISB #3,R0           Иначе - 2 точки в буфер
     BISB #200,@R1        Восстановить точку на экране
4:   ADD #100,R1          Следующая точка по вертикали
     BPL 5                в пределах экрана ?
     SUB #40000,R1        Нет - коррекция "на рулон"
5:   SOB R2,3             Цикл из 4 вертикальных точек
     CALL TYP             Выдать графический элемент на печать
     SUB #400,R1          Адрес прежний - начало столбца
     CMP #40000,R1        В пределах экрана ?
     BLOS 6               Да, иначе
     ADD #40000,R1        коррекция
6:   SOB R3,2             Цикл из 8 столбцов точек
     INC R1               Следующий байт
     SOB R4,1             Цикл из 64д знакомест (байт)
     ADD #300,R1          Следующая строка печати
     BPL 7                в пределах экрана ?
     SUB #40000,R1        Нет - коррекция
7:   SOB R5,0             Цикл строк
     MOV #12,R0           Перевод
     CALL TYP             строки
     CALL KM1             Инициализация принтера
```

* Ее работоспособность редакцией не проверялась. — *Прим. ред.*

```
        MOV (SP)+,R5          Восстановить
        JSR R4,@#110362       регистры
        RET                   Выход

; Подпрограмма выдачи кодов на принтер
TYP:  JSR R4,@#110346         Сохранить регистры
        MOV #177714,R1        Адрес порта
        BIC #177400,R0        Выделить младший байт
1:      BIT #400,@R1          Принтер готов?
        BEQ 1                 Нет - ждать
        CLR @R1               Очистка порта
        BIS #400,R0           Строб - бит
        MOV R0,@R1            Код и строб в порт
2:      BIT #400,@R1          Код принят?
        BNE 2                 Нет - ждать
        JSR R4,@#110362       Восстановить регистры
        RET                   Выход

; Подпрограмма выдачи команд на принтер
KM1:  MOV PC,R3               Команды: "сброс",
        ADD (PC)+,R3          "инициализация"
        .@K1+2
        BR KOM                Исполнение
KM2:  MOV PC,R3               Команды: "графика", "перевод
        ADD (PC)+,R3          строки на 24д/216д дюйма"
        .@K2+2
KOM:  MOVB (R3)+,R0           Очередной код команды
        CMPB #7,R0            Если 7 -
        BEQ 1                 конец
        CALL TYP              Передать код на принтер
        BR KOM                Следующий код
1:      RET                   Выход

; Командные коды принтера:
; сброс строки, инициализация принтера
K1:   .B:30,33,100,7
; перевод строки на 24д/216д дюйма, задание граф.режима
K2:   .B:33,63,30,12,33,113,0,2,7.E
```

Вот теперь, чтобы получить копию экрана, вам уже не понадобится фотоаппарат...

★ ★ ★

А что еще можно сделать с помощью вектора @#30? Допустим, вы изобрели свой формат записи на МЛ, который дает, например, в 4 раза более высокую плотность записи и в 3 раза более высокую надежность чтения (заметим, что такой

формат на БК-0010 уже есть*). Этого еще мало для успеха! Что можно сделать с помощью спецкопировщика? Только организовать надежный и компактный архив. Вы не сможете загружать в таком формате ни тексты для редакторов, ни программы на БЕЙСИКе и ФОКАЛе — придется каждый раз вначале «распаковывать» файлы, т. е. перезаписывать их в стандартном формате БК-0010. Есть ли выход? Да, это опять супервизорный драйвер. Пусть он перехватывает прерывание по вектору @#30, но на этот раз выделяет EMT 36. Затем надо обратиться к блоку параметров EMT 36 и проанализировать его. В зависимости от результатов анализа выполняется чтение или запись, но только с помощью не стандартного драйвера магнитофона БК, а своего спецдрайвера. Ведь в блоке параметров для EMT 36 мы найдем все: и команду (запись или чтение), и адрес, и длину файла... А после выполнения заданной операции с требуемым файлом нужно установить все системные ячейки ЭВМ, стек и регистры в такое состояние, как будто «сработала» стандартная команда EMT 36. Кажется, эта фраза нам уже встречалась? Ну да, это же эмуляция, но на этот раз не EMT 20, а EMT 36. Конечно, эта задача гораздо сложнее, но и цель ведь куда серьезнее. Если удастся реализовать такой системный драйвер, то он будет работать со всеми, без исключения, программами пользователя на любых языках и в любых режимах, осуществляя обмен с магнитофоном в спецформате, если только программа пользователя выходит на обмен по стандартному прерыванию EMT 36. Автору это удалось, и теперь он работает с магнитофоном на скорости 4000 бод вместо стандартных 1200 (поясним, что бод — это единица скорости обмена информацией, соответствующая одному бит/с).

Точно так же можно организовать обмен с дисководом (без всякой операционной системы), с квазидиском на базе дополнительного ОЗУ (которое на БК может в принципе иметь объем порядка 2-4 Мб) и даже с принтером — при этом на принтер будут выдаваться для печати те самые блоки информации, которые мы хотели бы записать на МЛ. Правда, для принтера мы уже знаем более удобный способ. Важно то, что для такого обмена не потребуется опять-таки никакого вмешательства в программу пользователя, лишь бы она была рассчитана на работу с магнитофоном**.

А другие EMT? К примеру, вам не нравится алгоритм векторной графики БК-0010 (он, и правда, не самый лучший, прежде всего слишком медленный). Отлично, напишем свою программу, но не станем просить завод, чтобы он выпустил новое ПЗУ, а просто перехватим EMT 32 и заменим векторную графику БК на свою.

Хотите заменить знакогенератор? Нет проблем. Все то же EMT 16 (и эмулированное EMT 20) позволят решить задачу. Надо только выделять часть символов, которые мы хотим заменять (это могут быть, например, латинские символы или псевдографика), перехватывать их коды и, вместо того чтобы вывод этих символов «отдать на откуп» EMT 16, рисовать их на экране самостоятельно, пользуясь своими масками начертания. Конечно, для этого надо хорошо знать, как это делает драйвер дисплея БК-0010, но это уже дело вашей настойчивости и упорства. Подскажем лишь, что тут «замешаны» две подпрограммы по адресам @#102764 и @#103222. С их помощью, указав в регистре R1 адрес начала строки

* О подобном драйвере можно прочитать в журнале «Информатика и образование», № 3 за 1991 г., с. 52. — *Прим. ред.*

** Именно так, путем перехвата EMT 36, организован обмен данными с диском практически во всех современных дисковых ОС для БК (NORD, ANDOS и др.). А перехват EMT 36 для вывода текста на принтер использован в универсальном драйвере SCREW ANIRAM MIRIADA (см. №1 за 1994 г., с. 33, а также вторую страницу обложки). — *Прим. ред.*

маски, можно вывести на экран символ независимо от того, находится его маска в ОЗУ или в ПЗУ, стандартная она или «самодельная».

А прерывание TRAP? Например, в языках высокого уровня по «трапам» выполняются многие процедуры. Почему бы их не «поймать», как мы это делали с EMT, и не извлечь из этого пользу? Например, ввести свои дополнительные процедуры или изменить алгоритмы имеющихся. Конечно, для этого надо знать структуру интерпретатора языка (например, ФОКАЛа) так, как будто это вы его придумали...

А прерывания от клавиатуры? Векторы @#60 и @#274 не изменяются почти ни в каких программах пользователя. Этим можно воспользоваться и ввести по ним дополнительные директивы почти в любую программу. Удобнее всего использовать коды в регистре «АР2» для клавиш «РУС», «ЛАТ» и «ВВОД» — эти коды не обрабатываются практически никакими программами и их можно применять для своих целей без помех. Например, нажатием клавиши «АР2»/«ЛАТ» можно вызвать ту же подпрограмму графического копирования экрана. Покажем, как это можно сделать, на примере:

```
; Инициализация драйвера
        MOV PC,R0           Вычислить адрес
        ADD (PC)+,R0        супервизора
        .@SQN+2             в перемещаемом формате
        MOV R0,@#274        Записать новый вектор @#274
        HALT                Стоп

; Супервизорная программа - обработка прерывания по "АР2"
SQN:    CMPB #17,@#177662   Клавиша "ЛАТ" по регистру "АР2"?
        BNE 0               Нет - выход
        CALL GRK            Графическая копия экрана
0:      JMP @#101362        Передача управления в драйвер
                            ; клавиатуры БК-0010
```

После всех предшествующих примеров можно было бы лишний раз и не повторять, что эту программу можно также использовать в составе общего универсального драйвера принтера, что ее тоже надо защитить от EMT 14, и т.д.

Ясно, что тем же путем можно ввести дополнительные директивы в любую программу пользователя, например, не выходя из редактора МИКРО.10К, вызвать программу печати «от метки до курсора», надо лишь знать, что в МИКРО адрес метки хранится в ячейке @#426, а адрес курсора — в ячейке @#432. Удобно, не правда ли? И программа получится очень простой, всего в несколько десятков команд.

Итак, теперь вы знаете, что такое системные драйверы. А почему автор назвал это «концепцией»? Потому что, как было сказано в начале раздела, есть два пути — либо приспосабливать КАЖДУЮ программу пользователя для данной цели, например для печати на конкретном принтере, либо сделать это сразу ДЛЯ ВСЕХ программ, написав резидентный драйвер. Это именно КОНЦЕПЦИЯ, или, как сказано в словаре иностранных слов, «система взглядов на те или иные явления». До сего дня мало кто из программирующих на БК решает указанные проблемы с системных позиций, вследствие чего и множится необоснованно число версий редакторов и языков, например варианты EDASP для печати на «Роботроне», на МС-6312, на МС-6313, на телетайпе... И все это — самостоятельные программы, а что в них «самостоятельного»? Только драйвер принтера. И ради его изменения нужно «перепахать» такую сложную программу, как EDASP. Стоит ли?

Контрольные вопросы и задания

1. Предложите способ, который позволит всю информацию, выводимую любой программой на экран, записать в виде файла на МЛ. Какое значение может иметь такой способ, например, для работы с ФОКАЛом? С дизассемблером?

— Нужно написать системный драйвер, аналогичный драйверу принтера для печати текста. Но вместо вывода символов на принтер надо выделить свободную область памяти, куда записывать последовательно коды появляющихся на экране символов. Количество байтов получившегося массива надо подсчитывать. Окончание формирования массива может быть по достижении им заданной длины или по специальной директиве, после чего с помощью EMT 36, зная адрес и длину массива, надо записать его на МЛ в виде файла. При работе с ФОКАЛом такой прием дает возможность, например, записать листинг ФОКАЛ-программы на МЛ, а затем загрузить его в редактор текста. Это пригодится, если нужно обработать описание или учебную программу, написанные на ФОКАЛе, включить ФОКАЛ-программу в текст, написанный с учебными целями, и т.п. При работе с дизассемблером это позволит дизассемблированный текст программы в кодах прокомментировать в редакторе, а отредактировав его и приведя к формату, например, МИКРО.10К, даже оттранслировать в ассемблер-системе или включить в качестве блока в свою программу*.

2. Напишите блок для «повторного», без разрушения памяти, выхода из МСД в ФОКАЛ, пользуясь указаниями о порядке такого выхода, приведенными в конце раздела о директивах МСД. Предусмотрите защиту блока выхода в ФОКАЛе.

— Для выхода из МСД в ФОКАЛ надо проделать следующее:

Щ 120020А26Д0П 262А177777И «СТОП»

Реализуем это в виде программного блока:

```
BEG: CLR R0             Адрес пересылки информации
     MOV #120020,R1     Адрес информации в ПЗУ-ФОКАЛ
1:   MOV (R1)+,(R0)+    Пересылка блока
     CMP #120046,R1     информации длиной 26
     BGE 1              в системную область
     MOV #-1,@#262      Запись признака кода "ВВОД"
     MOV PC,R0          Вычисление адреса
     ADD (PC)+,R0       начала блока в
     .@BEG+2            перемещаемом формате
     MOV R0,@#1744      Ограничить доступную ФОКАЛу
                      ; область памяти началом
                      ; блока выхода в ФОКАЛ
     HALT               "СТОП" - выход в ФОКАЛ
```

Конечно, реализовать эту программу можно и по-другому, это только один из возможных вариантов.

(Продолжение следует)

---

* В редакции можно приобрести программы, дающие возможность преобразовывать в текстовый формат листинги программ на ФОКАЛе и БЕЙСИКе, а также перегонять в текстовый файл дизассемблированные кодовые фрагменты не очень большой длины. — *Прим. ред.*

В.П. Юров,
Москва

Музыка, речь и цветомузыка на БК

Синтезаторы музыки

Впервые музыка на БК прозвучала в 1985 г. в программе **«Screen Editor»** фирмы DEC (многим пользователям она известна под именем **«МУЗРЕД»**). Годом позже в программе **«БЛОКНОТ музыкальный»** на Т-языке было создано несколько файлов (**«КОНЦЕРТ»**, **«КОНЦЕРТ1»**, **«КОНЦЕРТ2»** и т. д.), в которых исполнялось по нескольку одноголосных мелодий с вполне приличным звучанием. Изменить эти мелодии можно было довольно просто, умея программировать на Т-языке, что позволило наиболее любознательным пользователям создать на их основе целую серию музыкальных файлов.

(Для справки: Т-язык — это довольно простой язык программирования, созданный в те времена, когда БК была оснащена только штатным ФОКАЛом, и предназначенный для разработки прежде всего обучающих программ. После оснащения БК «встроенным» БЕЙСИКом, а в особенности после появления микроассемблеров этот язык несколько устарел, но он тем не менее все еще может быть полезен тем, у кого возникли проблемы при освоении программирования.*)

Первыми очень несложными программами, которые позволяли использовать БК как примитивный клавишный музыкальный инструмент — синтезатор, были разработки неизвестных авторов **«Пианино»** и **«Орган»**. Эти две практически одинаковые программы, датируемые примерно 1987 г., позволяют при нажатии на клавиши БК извлекать ноты (в пределах 1.5 октавы) и исполнять одноголосные мелодии. При этом БК через магнитофонный вход подсоединяется к усилителю, в качестве которого часто используется магнитофон в режиме записи с включенной паузой**.

Указанные программы обеспечивают на БК качество звучания примерно такое же, как у продающихся сегодня зарубежных детских электронных игрушек электронного пианино или органа. Несмотря на свою примитивность, они до сих пор могут быть полезны детям от трех лет и старше.

* Более подробные сведения о Т-языке можно прочитать в журнале «Информатика и образование» №2 за 1987 г., стр. 64. — *Прим. ред.*

** Кодовый листинг подобной программы был опубликован в журнале «Наука и жизнь», №4 за 1987 г. — *Прим. ред.*

Дальнейшим развитием идеи явилась рижская программа **«Sound»**. Она выполнена в цвете, на экране изображена клавиатура пианино и указано ее соответствие клавиатуре БК (рис. 1). При нажатии на клавишу БК на экране индицируется соответствующее нажатие клавиши пианино. Можно также выбирать длительность звучания, тембр и высоту звука (октаву). К сожалению, эта программа разрабатывалась в расчете на БК-0010, а на БК-0010.01 она иногда работает недостаточно корректно: при малой длительности вместо звучания нот издается лишь хрипение.

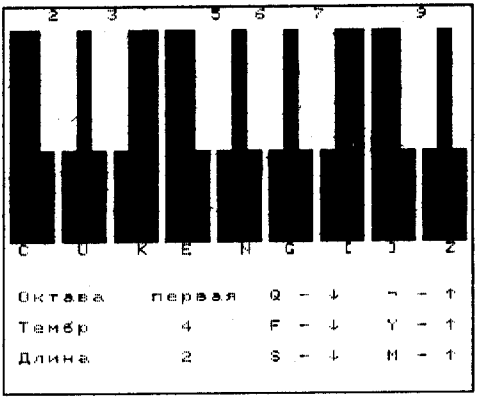


Рис. 1. Вид экрана при работе с программой «Sound»

Музыкальные редакторы

Первым музыкальным редактором на БК, который позволяет простым способом писать музыкальные тексты и воспроизводить одноголосные мелодии или записывать их в виде исполняемого файла, является программа **«Meloman»**, существующая в различных версиях. На рис. 2 показан общий вид экрана для версии 0786 с набранным музыкальным текстом. Ноты записываются при нажатии на соответствующие клавиши, которые указаны на экране рядом с изображением клавиатуры пианино. Запись нот с указанием их длительности, а также пауз осуществляется с помощью цифр и букв. Темп игры устанавливается отдельно. Диапазон исполнения охватывает четыре октавы. В нем также предусмотрена возможность подстройки частоты ноты ля (в соответствии с которой автоматически настраивается весь звукоряд). В других музыкальных редакторах эта очень полезная функция, особенно для пользователей с развитым музыкальным слухом, к сожалению, отсутствует. Однако в настоящее время этот редактор устарел, и, кроме того, на БК-0010.01 он работает неустойчиво.

В 1987 г. этот редактор был усовершенствован и получил название **«Meloman 2»**. В новом варианте с экрана исчезло изображение клавиатуры пианино, но зато запись нот приобрела привычный вид (рис. 3). Нотный диапазон в нем, так же как и в его предшественнике, охватывает четыре октавы с длительностями нот от 1/32 до 1/2, имеет семь ступеней темпов и три приема исполнения: «staccato» (отрывисто), «non legato» (отрывисто, но с меньшими промежутками между нотами) и «legato» (плавно). Ввод нот может осуществляться в режиме «записи с листа», когда их длительность записывается в нотный стан равной предварительно выбранному значению, или в режиме «композиции», когда длительность каждой ноты устанавливается в зависимости от времени удержания нажатой соответствующей ей клавиши. Есть также режим «инструмент», в котором можно исполнять мелодии без записи в ОЗУ. Воспроизведение мелодий, а также звучание нот во время их набора осуществляется через

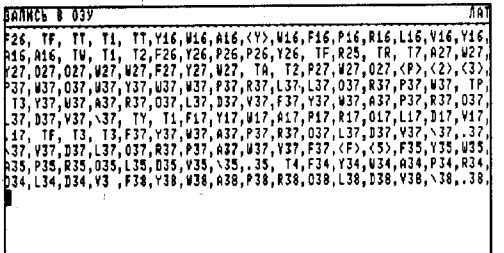


Рис. 2. Вид экрана при работе с программой «Meloman» в режиме редактирования и записи мелодии в ОЗУ

встроенный в компьютер пьезодинамик или подключенный к магнитофонному выходу внешний усилитель.

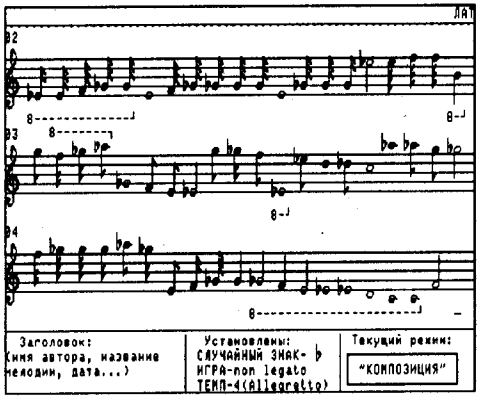


Рис. 3. Вид экрана при работе с программой «Meloman 2» в режиме редактирования и записи мелодии в ОЗУ

«Meloman 2» обладает приятным звучанием и полезен при освоении музыкальной грамоты, а также для написания мелодий с целью их дальнейшего использования в программах или для отдельного исполнения. Во многих программах для БК звучат мелодии, созданные в этом редакторе. Однако работа с ним на БК-0010.01 несколько усложняется, поскольку он рассчитан на клавиатуру БК-0010.

В дополнение к «Melomany» была создана программа под названием **«Drummachine v3.1»**. С ее помощью можно имитировать ударную установку, однако в ней не предусмотрена возможность записи в файл, а только прослушивание до выхода из программы. В настоящее время эта разработка устарела.

В начале 1990 г. был написан трехголосный музыкальный редактор **«KLAWESIN»** (рис. 4). Созданная в нем музыка также использована во многих программах, она имеет характерное звучание, напоминающее клавесин, но с заметной примесью шумов*. Редактор рассчитан на работу в цветном режиме. Запись нот осуществляется в привычном виде путем перемещения курсора по нотному стану.

В начале 1992 г. для этого редактора была разработана программа **«Laser»**, дополняющая его ударниками, а в начале 1994 г. — программа **«Memory 2»**, сжимающая его музыкальные файлы более чем в три раза и улучшающая их звучание за счет снижения шумов и расширения диапазона. («KLAWESIN» в основном охватывает ноты в диапазоне скрипичного ключа, а «Memory 2» расширяет его до басового диапазона.)

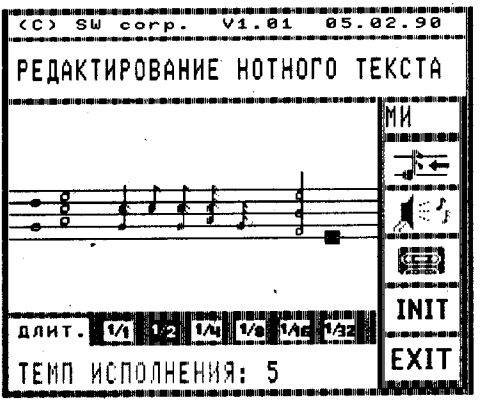


Рис. 4. Вид экрана при работе с программой «KLAWESIN» в режиме редактирования и записи мелодии в ОЗУ

Менее известен трехголосный музыкальный редактор **«ThreeVoiceMelom 3»**, или **«TVM3»** (рис. 5), поскольку он рассчитан в основном для БК-0011М, а на БК-0010(.01) его звучание неудовлетворительное. Кроме того, некоторые версии этого редактора работают только со специальным блоком, подключаемым к порту. «TVM3» позволяет создавать трехголосные мелодии и сопровождать их ударными инструментами. Он также имеет характерное звучание, немного напоминающее клаве-

* Этот шум особенно заметен при использовании магнитофона в качестве усилителя, в этом случае он практически полностью подавляет саму мелодию. — *Прим. ред.*

син, с примесью шумов, особенно значительных на БК-0010(.01).

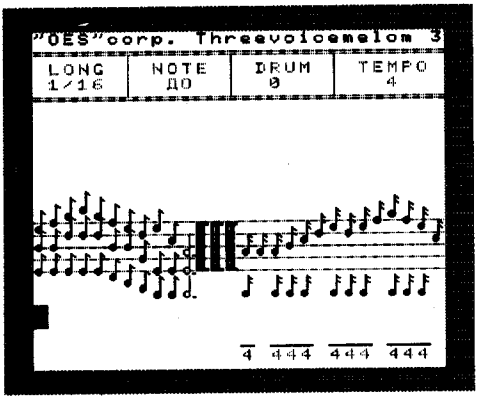


Рис. 5. Вид экрана при работе с программой «TVM3» в режиме редактирования и записи мелодии в ОЗУ

Последней (на начало 1994 г.) разработкой в области музыкальных редакторов является пакет **«Maestro»***. Он состоит из собственно редактора и двух подгружаемых драйверов (двух- и четырехголосного), имеется также целый ряд демонстрационных программ. В редакторе «Maestro» могут быть написаны двух- или четырехголосные мелодии в сопровождении ударных инструментов. Запись нот осуществляется путем перемещения курсора по нотному стану. Кроме того, на клавиатуре БК имитируется клавиатура одной октавы пианино, на которой можно проиграть (но не записать в нотный стан!) мелодию.

Музыка, созданная для воспроизведения с помощью двухголосного драйвера, имеет достаточно хорошее звучание, хотя ее написание не столь наглядно и удобно, как в «Meloman 2», и требует от пользователя большей квалификации. Мелодии же, исполняемые с помощью четырехголосного драйвера, по звучанию напоминают клавесин и сопровождаются шумами.

Специально для «Maestro» разработана программа **«Memory 3»**, которая позволяет примерно в три раза сжать файлы, созданные с помощью четырехголосного драйвера.

Говоря о качестве воспроизведения мелодий, следует заметить, что пока звучание всех мелодий, содержащих три и более голоса и созданных тем или иным способом на БК, сопровождается шумами, уровень которых зависит не только от типа компьютера (БК-0010(.01) или БК-0011М), но и от конкретного его экземпляра. Даже мелодии, для улучшения звучания которых предусмотрено использование **«Covox»** (специальная схема, подключаемая к порту, — см. рис. 6), звучат на разных компьютерах по-разному и часто также сопровождаются шумами, которых не было при отладке на другом компьютере. Несколько лучшие результаты получаются при использовании программно-музыкального комплекса **«Менестрель»**, который выпускался в Павлово-Посаде. С помощью этого комплекса могут быть созданы и проиграны мелодии со стереофоническим звучанием (три голоса по каждому каналу) и специальными эффектами. Но способ написания му-

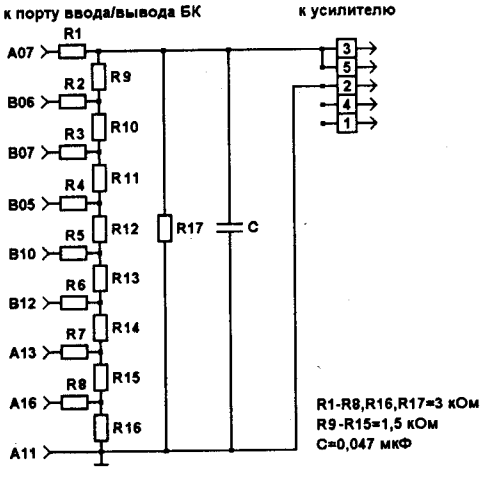


Рис. 6. Принципиальная схема блока «Covox», подключаемого к порту ввода/вывода

* Более подробное описание этой разработки приведено в статье ее автора, А. В. Леликова, в этом выпуске журнала. — *Прим. ред.*

зыки в нем больше подходит программисту, чем музыканту или простому пользователю. Этот комплекс (как и многое другие специальные разработки в области синтеза музыки на БК, не описанные в этой обзорной статье) не получил широкого распространения, однако для некоторых пользователей может представлять определенный интерес.

В другой программе — **«Rock Monitor»** (Самара-1993—94) предусмотрено уже использование специального музыкального процессора YM—2149F (AY—3—8910), применяемого, например, в компьютере ZX—Sinclair Spectrum. Эта разработка является аналогом одноименной программы на ZX—Sinclair Spectrum. (Такие программы широко используются на различных компьютерах и позволяют создавать по-настоящему синтезированную компьютерную музыку. Способ написания мелодий в таких редакторах значительно отличается от принятого у музыкантов: ноты записываются в виде специальных кодов.)

Всего же в настоящее время на компьютерах можно выделить три вида редакторов: программы наподобие «Meloman 2», «KLAWESIN» и пр. для неспециалистов; музыкальные редакторы, призванные облегчить труд профессиональных музыкантов (на БК таких редакторов пока нет, примером же могут служить разработки «CUBASE» и «PRO 24» на компьютере Atari ST—1040) и программируемые редакторы типа «Rock Monitor» для создания компьютерной музыки. При использовании последних требуется не только знание музыкальной грамоты, но и умение программировать в редакторе (не путайте это с программированием на ассемблере или БЕЙСИКе).

Конечно, подключая к БК различные процессоры, можно значительно улучшить его музыкальные (да и не только музыкальные) возможности. Но это немного напоминает ситуацию из сказки, когда солдат варил щи из топора. Можно, конечно, подключить к клавиатуре БК процессор «Motorola» и восхищаться новыми возможностями, но автору статьи наиболее интересными представляются те решения, которые программным путем позволяют на БК получить то, что раньше трудно было предположить. Аппаратные же доработки при этом должны сводиться к минимуму (примером может служить показанная выше схема «Covox»).

Безусловно, каждый вправе поступать со своим БК так, как ему заблагорассудится, и если Денис Сотченко (автор схемного и программного решения использования штатного БЕЙСИКа с дисководом) расскажет о своих экспериментах с БК (а идей и разработок у него много, в том числе и связанных с музицированием), то ему будут благодарны многие читатели. В то же время возможности БК (без его каких-либо переделок) еще не до конца использованы, что подтверждается вновь появляющимися программами (музыкальный редактор «Maestro», операционная система «ANDOS v3.1», игры «Armageddon3.SK», «Savage», «Goblins», «WarLords» и пр.).

Синтезаторы речи

Впервые синтезированная речь на БК прозвучала в 1986 г. в программах **«Рисуем домик»** и **«Цифры»**, написанных на Т-языке. В программе «Рисуем домик» одновременно с рисованием фрагментов домика танцующий робот произносит соответствующие фразы. Эти фразы звучат довольно разборчиво, и для большей убедительности они повторяются дважды. Умея программировать на Т-языке, эти фразы можно не только несколько скорректировать, но и получить другие, заметно отличающиеся от исходных. В программе «Цифры», предназначенной для обучения детей, синтезированный голос предлагает набрать на клавиатуре то или иное число и оценивает правильность ввода. Еще в одной программе — **«Слепой»**, появившейся примерно в то же время, синтезированный голос произносит довольно внятно значение нажатой клавиши.

Первые синтезаторы речи (1986-1987 гг.), позволяющие самому пользователю записать в память БК небольшой по длительности (до нескольких секунд) фрагмент речи или музыки (фонограмму) и затем воспроизвести его, были примитивно просты:

без меню и каких-либо пояснений и часто без документации. Таких программ довольно много («**Voice**», «**Голос**», «**Sintez**» и т. д.), в настоящее время они считаются устаревшими.*

Позже (начиная с 1988 г.) был разработан целый ряд программ под теми же или другими названиями, которые содержали меню, пояснения и снабжались документацией. Эти программы позволяют не только ввести фонограмму в компьютер и прослушать ее, но и записать в виде файла с блоком или без блока воспроизведения, а также вручную (с помощью встроенного редактора) внести в полученную фонограмму требуемые изменения. Записанные в виде файла фонограммы в дальнейшем можно использовать самостоятельно или для озвучивания других программ.

Ниже названы лучшие из подобных разработок, которые могут представлять интерес для многих пользователей БК, увлекающихся синтезом речи или музыки путем непосредственной записи фонограммы в память компьютера. Запись в них осуществляется через усилитель, подключенный к магнитофонному входу БК. В качестве такого усилителя обычно используют магнитофон, на котором проигрывается фонограмма или подключается микрофон (через него наговаривается текст). Воспроизведение осуществляется на внутренний динамик компьютера или на внешний усилитель, подключенный к его магнитофонному выходу. (В качестве внешнего усилителя опять же обычно используют магнитофон.) Конечно, тех, кто хочет получить наиболее качественное звучание фонограммы, не устроит использование магнитофона, и им придется использовать высококлассные усилители с полосовыми фильтрами и хорошими микрофонами.

Программы **«VOICE.S»** (Красногорск—88) и **«Синтезатор речи.KVIKOS»** (Москва—88) позволяют записать введенную в компьютер фонограмму в виде файла с блоком воспроизведения. В обеих этих программах имеется возможность установки скорости ввода фонограммы, а во второй из них — также и скорости ее вывода (воспроизведения).

Программы **«VOICE.ADW»** (Москва—92) и **«Speak Machine Editor»** (Москва—89) позволяют перед записью фонограммы в виде файла дополнительно вручную откорректировать ее. Полученные файлы записываются без блока воспроизведения (листинг блока воспроизведения на ассемблере приведен в описании к «Speak Machine Editor»).

Качество звучания фонограмм, полученных с помощью четырех вышеназванных программ, зависит не столько от них, сколько от возможностей усилителя и микрофона, и от трудолюбия пользователя.

* * *

Следующая группа разработок позволяет синтезировать речь по тексту, введенному в компьютер (озвучивать текст). В их основе лежит фонемный цифровой способ синтеза речи: слова складываются из отдельных фонем (звуков), заложенных в программу.

Впервые такие программы на бытовых 8-разрядных компьютерах (естественно, зарубежных и служащих для синтеза речи на английском языке) появились примерно лет десять назад. В них текст, подлежащий синтезу (озвучиванию), не только записывался в виде транскрипции (т. е. как он

* Одна из подобных простейших разработок была опубликована в журнале «Вычислительная техника и ее применение», №2 за 1990 г., с. 34 (см. также исправление опечаток в №9 за 1991 г., с. 36). Алгоритм работы этого «синтезатора речи» был максимально простым: при записи в цикле тестировалось состояние пятого разряда регистра управления внешними системными устройствами (@#177716) и в ОЗУ потоком бит за битом заносились нули и единицы. При воспроизведении же содержимое ОЗУ побитно переписывалось в шестой разряд регистра @#177716, таким образом записанная речь выводилась на динамик БК и на подключенный в качестве усилителя магнитофон. В данном случае БК как бы превращался в некий «цифровой магнитофон», а ввод-вывод речи осуществлялся по стандартному аппаратному механизму чтения-записи файлов. — *Прим. ред.*

должен произноситься: на русском языке вместо «что» нужно было бы писать «што»), но также следовало указывать в тексте длительность каждого гласного звука путем его повторения или введения специального символа длительности. Кроме того, перед согласным звуком и после него указывалась длительность паузы. (Пауза перед согласным звуком обеспечивала получение «взрывного» характера звучания, как, например, в звуке «б» русского языка, а пауза в конце использовалась для смягчения звука или, наоборот, для его более твердого произношения.) Два-три часа экспериментирования с такой программой позволяли получать и вполне приличное звучание, почти без английского акцента, русских фраз.

Такой принцип синтеза речи лежит в основе программ **«Ворчун.KVIKOS»** (Москва—89) и **«SPEECH.NPC»** (Новосибирск—90). Для первой из них текст (с учетом транскрипции) пишется в текстовом редакторе формата EDASP (максимальный объем файла около 10 кб), а для второй, также с учетом транскрипции, — во встроенном редакторе (максимальный объем около 700 символов). В обеих программах текст при его озвучивании одновременно выводится на экран. Несмотря на ограниченный объем текста в программе «SPEECH.NPC», пользоваться ею удобнее, так как в ней проще откорректировать и записать текст (не надо загружать внешний редактор), да и звучание разборчивее.

В более современных программах для озвучивания текста не требуется предварительно записывать его в виде транскрипции со специальными длительностями и паузами. Более того, компьютер при необходимости сам может показать транскрипцию, по которой он произносит текст, и дать возможность скорректировать ее (например, «SMOOTHTALKER» для компьютера Atari ST, где текст может произноситься мужским или женским голосом с различной скоростью и тембром). Такие программы широко используются для озвучивания текста в текстовых редакторах (например, «ProWrite» для Amiga 500/1200), а также в самообучающихся диалоговых программах (например, «Racter», написанная в 1984 г. также для компьютера Amiga 500/1200).

Конечно, БК трудно соревноваться с компьютерами, имеющими не менее 0,5 Мб памяти и встроенный дисковод, а также 32-разрядный процессор (пусть даже с 16-разрядной шиной данных), но и здесь он не слишком отстает. В 1989 г. для БК разработана программа **«Говорун»** (о творческих муках авторов при ее создании рассказано в журнале «Информатика и образование», №2 за 1991 г., с.93), которая позволяет довольно внятно произносить обычный русский текст, выводимый на экран, и одноразрядные цифры. (Латинские символы произносятся как названия букв английского алфавита.) Эта программа создавалась по заказу Общества слепых для облегчения ввода текста с клавиатуры, однако представляет значительный интерес для всех пользователей. Из-за того, что она не содержит в себе каких-либо пояснений, для многих пользователей возможности ее применения неясны. Эта небольшая программа (8 кб) довольно просто объединяется с разработками на ФОКАЛе, БЕЙСИКе или в машинных кодах. Можно просто загрузить ее в монитор или в БЕЙСИК, и она будет проговаривать любой текст, появляющийся на экране, в том числе вводимый с клавиатуры.* Примером такого совмещения может служить

* При этом следует учесть, что программа «Говорун» неперемещаема, а для вывода текста на экран следует использовать только стандартную функцию EMT 16 или EMT 20 (при вводе текста с клавиатуры для его «проговаривания» нужно производить эхо-печать символов с помощью все той же EMT 16). Основная часть «Говоруна» представляет собой специализированный EMT-драйвер с отловом EMT 14, EMT 16 и EMT 20, так что в работающей с ним программе нельзя изменять содержимое вектора прерывания EMT @#30 (требуемое значение начального адреса EMT-драйвера «Говоруна» заносится туда вручную или «стартовой оболочкой» самой этой программы). — *Прим. ред.*

игра **«TANKER»**. Однако наибольшего эффекта от программы «Говорун» можно достичь при ее совмещении с самообучающимися диалоговыми программами.

Поскольку далеко не всем известно, что такое «самообучающаяся диалоговая программа», то вначале скажем о ней несколько слов. Упомянутая выше программа «Racter» позволяет вести «разумную» беседу с компьютером в диалоговом режиме на различные темы. Все начинается с взаимных приветствий и вопросов, а затем выбирается и уточняется тема беседы (о перспективах коммунизма, размышления о жизни, спорте, музыке и пр.). Ответы компьютера комбинируются из ранее заложенной в него информации и из ответов пользователя. Пользователь вводит их с клавиатуры, а компьютер свои вопросы и ответы выводит на экран (можно также и на принтер) и проговаривает их синтезированным голосом. По мере необходимости накопленная информация записывается на дискету или с нее считывается ранее введенная. Эту программу можно также использовать для изучения английского языка и, по мнению автора, она для этих целей более эффективна, чем специальные разработки. (Подобная система с использованием дисковода и программы «Говорун» может быть создана и на БК.)

На БК существует несколько диалоговых игровых программ на английском и русском языках. Но мало кому известно, что есть и самообучающаяся диалоговая программа **«Кот»**, написанная на ФОКАЛе. Это очень простая и в то же время очень занимательная игра. Цель ее — отгадать животное, задуманное пользователем*. Поначалу для программы все животные — это коты. Но по мере диалога и наводящих вопросов программа самообучается. Полученная от пользователя информация о животных записывается в файлы данных. С этими данными обыграть программу уже невозможно, если только не задумать животное, которое ранее не называлось. Но в конечном итоге, как только у пользователя иссякнет фантазия (и знания), БК одержит победу. Такая программа даже начинающим программистом легко может быть переведена на БЕЙСИК (или ассемблер) и дополнена программой «Говорун» (объединить с «Говоруном» можно и существующую программу «Кот»).

Цветомузыка

Идея цветомузыки выдвигалась различными учеными (в том числе и нашими соотечественниками) начиная с XVIII в., но реальное воплощение она получила лишь в начале XX в. (отчасти благодаря работам композитора А. Н. Скрябина). Позже был осуществлен переход от цветомузыки к светомузыке, которая включает не только смену цветов, но и сложные пространственно-графические проекции.

Примерно в конце 1986 или в начале 1987 г. почти одновременно появились две небольшие (объемом около 1 кб) программы неизвестных авторов под названиями **«CMP»** и **«CMU»**. Вскоре, опять же неизвестными авторами, программа «CMU» была модифицирована (файлы **«CMU1»** и **«CMU4»**), ее объем сократился примерно до 0.3 кб и были изменены шкалы соответствий (алгоритмы вывода рисунка на экран).

Эти программы, как и большинство разработок того времени на БК (да и не только на БК), не имели описаний и каких-либо «helpов», поэтому для большинства пользователей, к которым они попали различными окольными путями, остались загадками и за ненадобностью (точнее, из-за их непонятности: программа запускается, включается черный экран — и что дальше?) были стерты. В то же время работа этих программ довольно проста: достаточно запустить ее и вставить в магнитофон кассету с музыкой, как на экране вспыхнет фантастическая череда цветовых рисунков, зависящих от громкости, ритма и высоты тона музыкального произведения. В «CMP» этот рисунок образуется чередованием трехцветных прямоугольников, а в «CMU» — многоцветных (рис. 7). Этот рисунок, как и последующие, фиксирующие мгновенную картину

* Аналогичная программа на БЕЙСИКе (правда, не для версии, имеющейся на БК) была опубликована под названием «Животные» в журнале «Наука и жизнь», №12 за 1986 г., с.93. — *Прим. ред.*

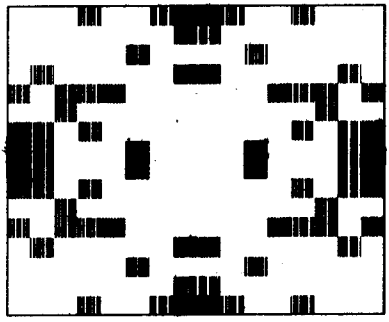


Рис. 7. «Застывшая» копия экрана при работе с программой «CMU»

на экране, возникающую при проигрывании музыкального произведения, конечно, не может дать полного представления о фантастической игре цветов на экране цветного монитора и приведен здесь только для примера.

Названные программы работают в автоматическом режиме, т.е. когда чередование цветов и их яркость зависят лишь от заложенной в алгоритме шкалы соответствий. При этом лишь отдельные музыкальные произведения получают наибольшую выразительность за счет их сопровождения цветомузыкой.

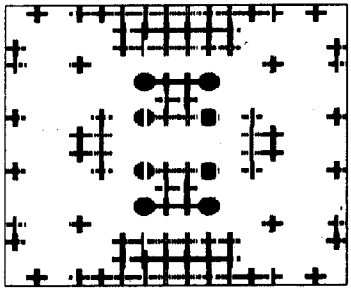


Рис. 8. «Застывшая» копия экрана при работе с программой «ЦМУ.3»

В 1988 г. по заказу одной из музыкальных групп была разработана программа, позволяющая БК управлять мощной светомузыкальной установкой, которая через специальное устройство подсоединяется к порту ввода-вывода БК. Программа позволяет писать специальную световую партитуру для каждого музыкального произведения, как это в свое время предусматривал А. Н. Скрябин.

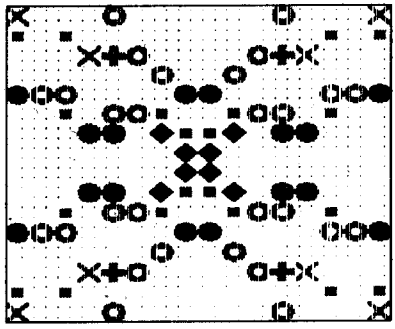


Рис. 9. «Застывшая» копия экрана при работе с программой «MORTLEY.DDV»

Идеи «CMU» получили дальнейшее развитие в программах **«ЦМУ.2»** и **«ЦМУ.3»** С. А. Кумандина (рис. 8) и в **«MORTLEY.DDV»** (рис. 9). Они позволяют изменять форму фигур, выводимых на экран (прямоугольники, круги, кольца и прочее), их динамичность и шкалу соответствий. В «ЦМУ.3», кроме того, можно также изменять цвет фона экрана. Эти программы позволяют получить максимальную выразительность для большей части музыкальных произведений. Однако в некоторых случаях программы серии «CMU» остаются более эффективными.

А. В. Леликов,
Москва

Многодрайверный музыкальный редактор MAESTRO

Работа по созданию музыкального редактора MAESTRO была начата еще в 1992 г. Тогда на БК уже имелось немало музыкальных редакторов, но, к сожалению, не существовало никакого стандарта кодирования нотного текста. Приходилось выбирать какой-то конкретный редактор, и если этот выбор был неудачен, то в другом редакторе музыку нужно было переписывать заново. Кроме того, все существовавшие в то время музыкальные редакторы были написаны довольно неоптимально с точки зрения программного кода. И наконец, действительно многоголосного редактора, по сути, не было вообще (достаточно сказать, что TVM позволяет записывать ноты только одной длительности, а KLAWESIN некорректно воспроизводит фрагменты с нотами разной длительности на одной позиции). Поэтому родилась идея написать некий универсальный редактор, имеющий единный формат записи файлов и позволяющий работать с многоголосными музыкальными произведениями. Он должен был удовлетворять следующим условиям: допустимость дальнейшего расширения возможностей и поддержки различных музыкальных приставок, минимальный объем занимаемой памяти, использование всех предоставляемых аппаратурой ресурсов и совместимость со всеми ранее созданными редакторами. И конечно же, большое внимание уделялось удобству работы с программой. Все эти требования были реализованы с помощью описываемых ниже принципов.

Независимые драйверы

При написании MAESTRO прежде всего был реализован принцип независимых драйверов, который заключается в следующем. Редактор делится на две части: основную (назовем ее ядром), которая осуществляет собственно редактирование нотного текста, и подгружаемый драйвер, обеспечивающий проигрывание мелодии и ряд дополнительных функций. Это означает, что такие характеристики, как количество голосов, качество звучания, наличие ударников и даже дополнительных команд, зависит только от драйвера. Принцип независимых драйверов хорош и тем, что он значительно сокращает затраты сил и времени при разработке программного обеспечения. Допустим, вы разработали новую музыкальную приставку, которая поддерживает восемь голосов. Ни одна из существующих программ, конечно же, не рассчитана на восьмиголосное исполнение, и тем не менее вам не придется создавать специально для нее новый музыкальный редактор, достаточно написать небольшой драйвер, работающий с MAESTRO и обеспечивающий воспроизведение музыки на вашей приставке. И тогда вы сможете слушать по-новому всю музыку, ранее созданную вами на БК. Или вот другой пример. Вы установили в свой компьютер музыкальный процессор. После подгрузки драйвера вся музыка, созданная до этого момента, будет исполняться на нем без каких-либо сложностей с ее переделкой.

Распределение памяти

Владельцы БК-0010 уже наверняка догадались, о чем дальше пойдет речь, — конечно же, об объеме памяти их компьютера. Всем понятно, что 16 кб — это мало. Поэтому при написании редактора MAESTRO особое внимание обращалось на решение проблемы

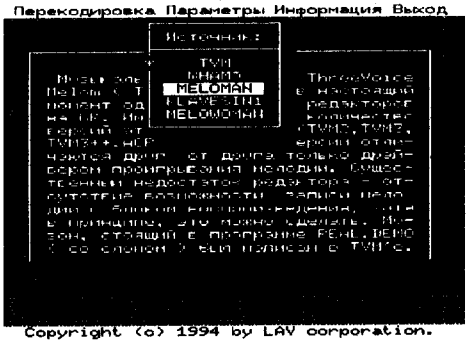
нехватки памяти. Во первых, после загрузки и автоматической настройки редактора его длина уменьшается до 6 кб. Во-вторых, в отличие от всех остальных программ редактор размещается в памяти начиная с адреса 0, а не 1000, что позволяет сэкономить еще полкилобайта. Далее, в MAESTRO реализован достаточно компактный формат хранения нотного текста, что позволяет работать с файлами мелодий рекордной длины, но и это еще не все. Редактор поддерживает работу с дополнительным ОЗУ 8 кб или с редко пока кем используемыми в полной мере богатыми ресурсами памяти БК-0011(М). А для тех, у кого имеется лишь БК-0010(.01) без дополнительного ОЗУ, реализована работа с расширенной памятью, когда верхняя часть картинки с экрана убирается, а освободившееся место используется для хранения нотного текста. Включается эта функция командой «?», поданной в основном режиме. (Кстати, по причине забывчивости автора эта функция не описана в руководстве пользователя.)

Возможности редактора MAESTRO

При написании редактора автор строго следил за тем, чтобы нотный текст записывался и воспроизводился по общепринятым в музыке правилам. Важное его отличие от остальных имеющихся на БК программ аналогичного назначения заключается в корректном воспроизведении нот с разными длительностями (в одной «многоголосной позиции»). Не менее удобно и наличие отдельных звуковых каналов, что дает возможность редактировать как все голоса одновременно, так и каждый из них в отдельности. (Можно писать басы на одном канале, а основную мелодию — на другом.) В числе прочих преимуществ назовем также возможность набора нотного текста с помощью не только курсора (как это сделано, например, в KLAWESINe), но и в режиме «эмуляции» клавиатуры фортепиано; реализацию буферных операций (наложение музыкальных фрагментов, воспроизведение помещенной в буфер части мелодии, «перемещение» из одной октавы в другую с понижением или повышением тона); возможность установки ключевых знаков альтерации и аккомпанемента ударников, а также корректную работу со звуком на всех моделях, включая БК-0011.

Работа с дисководом

Каждый пользователь БК, имеющий дисковод, сталкивался с непростой проблемой: в какой операционной системе работать. В настоящее время на БК сложилась уникальная ситуация — одновременно существует десяток операционных систем, никак не совместимых между собой. Кроме того, не стоит забывать и о пользователях с «пустым» контроллером. Поэтому в пакет MAESTRO входнт программа MDISK, автоматически определяющая, из какой ОС она была загружена, и настраивающая на нее редактор. Например, при работе с «пустым» контроллером обращения к магнитофону заменяются на аналогичные операции с диском, а в MKDOS и ANDOS v3.1 можно воспользоваться файлером.



Программа CONVERTER

К моменту MAESTRO в других музыкальных редакторах на БК уже было создано огромное количество мелодий. Поэтому в комплект включена программа CONVERTER, позволяющая переводить в формат MAESTRO музыку, написанную в одном из следующих редакторов: TVM, MELOMAN, KLAVESIN, MELOWOMAN и WHAM (Sinclair). Запуск CONVERTERa производится также через программу MDISK.

По вопросам приобретения пакета MAESTRO можно обращаться в редакцию журнала (телефон: (095) 151-19-40). Если же у вас возникли проблемы при использовании редактора или нужна техническая документация для написания собственного драйвера, звоните по телефону (095) 194-38-87 с 10 до 22 часов.

С. М. Неробеев, А. В. Сорокин,
Москва

ПОДКЛЮЧЕНИЕ МУЗЫКАЛЬНОГО ПРОЦЕССОРА К БК-0011М

Музыкальный процессор (часто неправильно именуемый сопроцессором) служит для воспроизведения музыки, специальным образом закодированной в виде текста. В отличие от оцифрованных мелодий, для проигрывания которых используется простой цифро-аналоговый преобразователь (ЦАП) типа *Covox*, кодированные процессорные мелодии, во-первых, занимают в десятки раз меньше места (так как представляют собой не переведенный в цифровой код звуковой сигнал, а список команд музыкального процессора) и, во-вторых, обеспечивают лучшее качество звучания. Кроме того, музыкальный процессор во время работы не «отвлекает» центральный процессор (ЦП) и может воспроизводить мелодии параллельно с выполнением любой программы, практически не замедляя скорость работы компьютера. Подключение музыкального процессора AY-3-8912 (28 выводов) или его более полного аналога AY-3-8910 (40 выводов) открывает совершенно иной музыкальный мир с точки зрения как слушателя (полифоническая музыка со стереоэффектом), так и компьютерного композитора (возможность программирования игры под музыкальный синтезатор).

К сожалению, музыкальные процессоры не имеют аналогов среди микросхем отечественного производства. Варианты заменяющих их схем хотя и предлагаются, но содержат большое количество корпусов — до 42 штук.

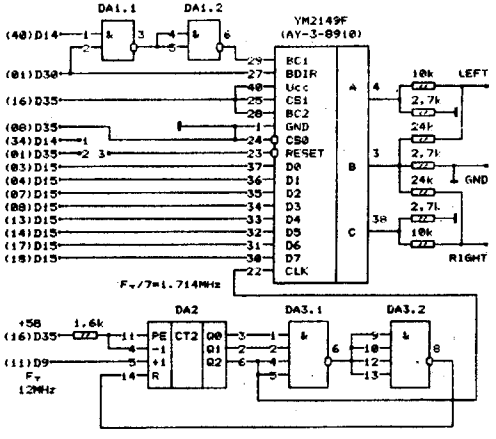
В последиее время появляется все больше игровых программ, работающих с музыкальным процессором (например, *CYBERNOID* by Caroline software Inc., новый *JOE BLADE 1* by Award&Maker). Существует также большое количество музыкальных демонстрационных программ (*LYRA II*, *SHOCK*, *MUSIC BOX*, *TOP DEMO* и др.) и просто мелодий со специальными программами для их воспроизведения (всевозможных *PLAYER*'ов). Кроме того, самаровские программисты из Caroline software Inc. написали «по мотивам» спектрумовского редактора *SOUND TRACKER* собственный музыкальный редактор *ROCK MONITOR*, превосходящий по возможностям свой аналог с ZX Spectrum и использующий все возможности музыкального процессора.

Тем не менее музыкальный процессор до сих пор остается экзотикой для большинства пользователей БК. Мало того, даже среди тех немногих, кто решился установить его на своем компьютере, получила распространение не самая лучшая схема его включения, разработанная Д. Сотченко (RDC), основной недостаток которой — подаваемая на музыкальный процессор слишком низкая тактовая частота 1.5 МГц. Стандартное же ее значение должно быть в пределах 1.71—1.78 МГц. Именно на такой частоте музыкальный процессор работает на ZX Spectrum и «Ямахе». Под эту частоту написаны все мелодии, которые в большинстве своем переписывают с ZX Spectrum, на эту же частоту настроены и все задержки в музыкальном редакторе *ROCK MONITOR*. Чтобы в этом убедиться, достаточно взять камертон. Разница между одной и той же нотой, играемой на музыкальном процессоре с тактовой частотой 1.5 МГц и звучащей на камертоне, составляет более полутора октав! Естественно, это не может удовлетворить людей с музыкальным слухом.

В отличие от предложенной ранее, главная особенность нашей схемы — делитель тактовой частоты генератора БК (12 МГц) на 7, в результате чего и получаются искомые 1.71 МГц. Конечно, собрать делитель 6 МГц на 4, который используется в схеме Д. Сотченко, проще, однако у музыкального процессора изменение тактовой частоты даже в 0,01..0,02 МГц ведет к довольно

значительному (до полутона) изменению частоты звука. Поэтому мы рекомендуем всем, у кого музыкальный процессор подключен по схеме Д. Сотченко, вместо делителя 6 МГц на 4 установить предлагаемый нами 12 МГц на 7. Заодно имеет смысл подключить сигнал RESET по самарскому варианту, что позволит избежать неприятностей при работе с некоторыми программами. Так, в варианте RDC (что на схеме соответствует замкнутым контактам 2 и 3) выключение музыкального процессора возможно только при перезапуске БК, поэтому, например, после выхода из программы *ROCK MONITOR* компьютер продолжает издавать последний звук игравшейся мелодии, пока не будет отключено питание. Самарский же вариант (при замкнутых контактах схемы 1 и 3), обеспечивающий выключение музыкального процессора по сигналу RESET, таких «фокусов» себе не позволяет.

Отметим, что на данной схеме показано подключение музыкального процессора AY-3-8910 как более распространенного (и более дешевого). Если же у вас AY-3-8912, нужно учесть соответствие выводов (см. приложение).



Слева, в обозначениях типа «(xx)Dyy» пара цифр xx показывает номер вывода, а Dyy — позиционное обозначение для микросхем, установленных на плате БК-0011М.

Микросхема	+5 В	GND
DA1 — К555ЛА3	014	7
DA2 — К555ИЕ7	016	8
DA3 — К555ЛА6	014	7

Выходные сигналы микросхемы AY-3-8910 (8912) образуют три отдельных музыкальных канала, два из которых (A и C) подаются на левый и правый каналы стереоусилителя, а третий (B) подмешивается в пропорции приблизительно 1:2 к каждому из каналов A и C. Смесители (как и в данной схеме) обычно выполняются на резисторах, хотя нет никаких ограничений и для применения операционных усилителей.

Диапазон выходных сигналов AY-3-8910 при синхронизирующей частоте 1.71 МГц достаточно широк: от 27 Гц до 115 кГц. Форма выходных сигналов — прямоугольная, т. е. их спектр содержит большое число гармоник. Следовательно, для получения высококачественного звучания необходим широкополосный стереоусилитель (20 — 20000 Гц) с выносными (лучше всего многополосными) колонками, хорошо воспроизводящими низкие и высокие частоты.

Программирование музыкального процессора

1. Работа с регистрами

Выбор номера регистра производится путем занесения его кода по адресу 177714. Пример: **MOV #370,@#177714**.

Установка регистра производится занесением байта значения по адресу 177714. Пример: **MOVB #7,@#177714**.

Таким образом, для включения голосов каналов A и B нужно выполнить следующие команды:

MOV #370,@#177714 — выбрали регистр,

MOVB #3,@#177714 — занесли число

Внимание! Следует помнить о том, что данные, заносимые в музыкальный процессор, — инверсные.

2. Кодирование регистров

Таблица 1

Регистр	Код
0	377
1	376
2	375
.....	...
11	364
12	363
13	362

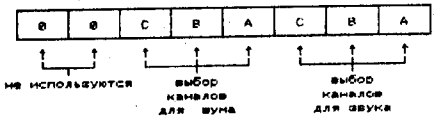
3. Назначение регистров

Таблица 2

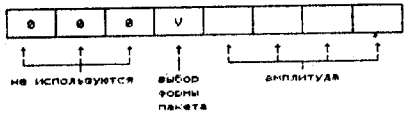
Номер регистра	Назначение	Записываемые значения
0, 2, 4	Нижние 8 бит частоты голосов A, B, C	0 — 377
1, 3, 5	Верхние 4 бита частоты голосов A, B, C	0 — 17
6	Регистр генератора шума	37 — 0
7	Регистр управления смесителем (см. п. 4)	0 — 77
8, 9, 10	Амплитуда каналов A, B, C (см. п. 5)	17 — 0
11	Нижние 8 бит периода пакета	377 — 0
12	Верхние 8 бит периода пакета	377 — 0
13	Форма волнового пакета (см. п. 6)	

Примечание. В третьей графе через тире указаны минимальные и максимальные значения.

4. Регистр управления смесителем

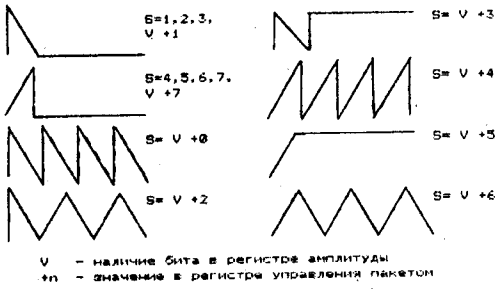


5. Регистр управления амплитудой



6. Формы волновых пакетов

В регистр, управляющий формой волнового пакета, заносятся значения 0—7, в регистре амплитуды необходимо сформировать бит V.



Отдельно следует поговорить о подключении музыкального процессора к БК-0010(.01). Здесь имеется своя специфика, в частности отсутствие таймера (имеется в виду не тот встроенный таймер, ктоторый управляется через регистры 177706—177712) и 14-го разряда регистра 177662 по записи, который управляет включением таймера, а также некоторые другие препятствия. Их ликвидацию мы оставляем владельцам БК-0010(.01) для самостоятельного решения.

В заключение сообщим, что для поддержки работы с сопроцессором нами написан драйвер *AY.DRV*, который воспроизводит мелодии, написанные на ZX Spectrum в музыкальном редакторе *SOUND TRACKER*. Драйвер абсолютно перемещаем и работает по прерываниям от таймера, благодаря чему его можно использовать для исполнения мелодии параллельно работе любой другой программы (чтобы «озвучить» не

блещущую музыкальным оформлением игру или просто не скучать во время работы). При этом наличие драйвера внешне никак не отражается на скорости ее исполнения. Пример использования нашего драйвера вы можете увидеть в программе *JOE BLADE 1*. Другой пример — плейер для воспроизведения процессорных мелодий — программа PL, работающая, правда, исключительно в дисковой ОС МКДОС. Его запуск предельно прост: нужно лишь навести курсор в файловой панели МКДОС на требуемый музыкальный файл (с расширением .SND), набрать в командной строке пару символов PL и нажать «ВВОД».

Приложение. Соответствие выводов микросхем AY-3-8912 и AY-3-8910

	RES	CLK	D0–D7	BDIR	BC1	BC2	CS1	CS0	A	B	C	+E
AY-3-8912	016	015	28–21	18	020	019	017	—	5	4	1	3
AY-3-8910	023	022	37–30	27	029	028	025	024	4	3	038	040

Примечания. Вывод 24 микросхемы AY-3-8910 (CS0) нужно соединить с общим проводом (GND). Вывод 6 микросхемы AY-3-8912 — общий.

По вопросам приобретения игры для БК-0011М JOE BLADE 1, комплекта воспроизведения мелодий (плейер PL, драйвер AY.DRV и пример музыкального файла), а также доработанной версии графического редактора для БК-0011М TECNO ART I, работающего в любой дисковой ОС, с принтером и с мышью без блока КМ, можно обращаться в редакцию по телефону: (095) 151-19-40. E-Mail: mail@infoobr.msk.su

Внимание! Опечатка

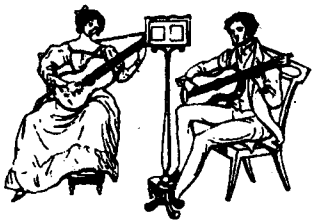
В статье Д. Ю. Усенкова «Мышка для БКшки» (№3 за 1994 г.) в листинге MOUSE.SYS были допущены две ошибки:

— начало листинга (стр. 39) продублировано дважды;

— пропущена часть команд ассемблера (стр. 40, вверху):

```
MOUSYM: JSR R7,INMOUS
  ; эмулирует клавиатуру, возвращая
  ; коды клавиш, соответствующих
  ; производимым мышью манипуляциям.
  ; R0 - код перемещения курсора (с
  ; диагональными стрелками !) или 0,
  ; если мышь неподвижна;
  ; R1 - код, соответствующий нажатию
  ; кнопок, или 0, если кнопки не
  ; нажимались.
  ; ВНИМАНИЕ! Возможно нажатие кнопки
  ; одновременно с движением, т.е.
  ; и R0 не равно 0, и R1 не равно 0.
      MOV R2,-(R6)
      JSR R7,%A61
       @%TABL3
      MOV R0,-(R6)
      MOV #10,R2
      BIC #177760,R0
1:    CMPB R0,(R1)+
```

Приносим читателям свои извинения.



С многоголосными программами большинство пользователей БК уже знакомы — тот же KLAWESIN широко разошелся по рукам практически сразу же после его появления «в свете». Но вот сама методика получения эффекта многоголосности на одном-единственном пьезодинамике для большинства БКманов оставалась недоступна — авторы многоголосных музыкальных редакторов относят это к коммерческой тайне.
Впрочем, не всех программистов влекут лавры всем известного любителя тайн — Мальчиша Кибальчиша — автор данной статьи делится с читателями журнала секретами исполнения на БК двухголосных мелодий.

Н. Д. Рыбальченко,
г. Майский

Двухголосные мелодии на БК-0010(.01)

Работа над реализацией двухголосного драйвера на БК-0010(.01) велась во многих направлениях, с использованием разных методов, есть даже вариант с применением встроенного таймера (регистр 177710). Но таймер имеется не во всех экземплярах БК, так что удачным можно назвать лишь вариант, в котором объединены два способа: отдельное формирование частот в звуковых каналах путем вычитания содержимого двух регистров, соответствующих каждому из каналов, и последующее сложение каналов (суперпозиция) на одном вращающемся регистре.

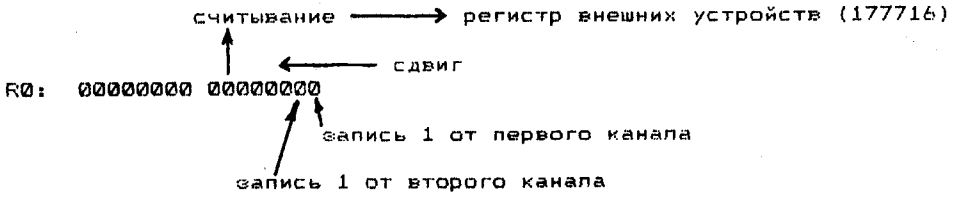
Константы, определяющие частоту звука, загружаются в регистры R1 и R2. В цикле из R1 и R2 вычитается по единице, и, как только в каком-либо из них оказывается нуль, в него вновь загружается та же исходная константа, а на пьезодинамик через «суммирующий» регистр R0 выдается импульс. Для этого в младшие разряды R0 (нулевой байт для первого, первый — для второго канала) записывается «1», после чего регистр R0 арифметически сдвигается влево и переписывается по адресу 177716.

Наглядно этот процесс можно представить так:

Если в младшие разряды R0 каждый раз записывать не одну единицу, а несколько (до шести), то громкость звука возрастает. А если удвоить значения констант, частота понижается на октаву.

(В БК-0010(.01) использованная в данном алгоритме прямая запись содержимого R0 в регистр 177716 с помощью оператора MOV не приводит к неприятным последствиям, так как по записи в этом регистре задействованы, кроме «звукового» бита, лишь разряды вывода на радиальную линию и управления двигателем магнитофона. В отличие от этого на БК-0011(М) старший байт регистра 177716 используется для переключения страниц памяти и, следовательно, использовать MOV R0,@#177716, скорее всего, нельзя. Во всяком случае, получаемый при этом на БК-0011(М) результат, редакцией не проверялся. — *Прим. ред.*)

Иллюстрацией описанного метода является приведенный ниже листинг на ассемблере (стандарт МИКРО-11), а также блок-схема и ассемблерный листинг программы, генерирующей аккорд (высота нот зависит от значений констант CON1 и CON2).



Листинг 1

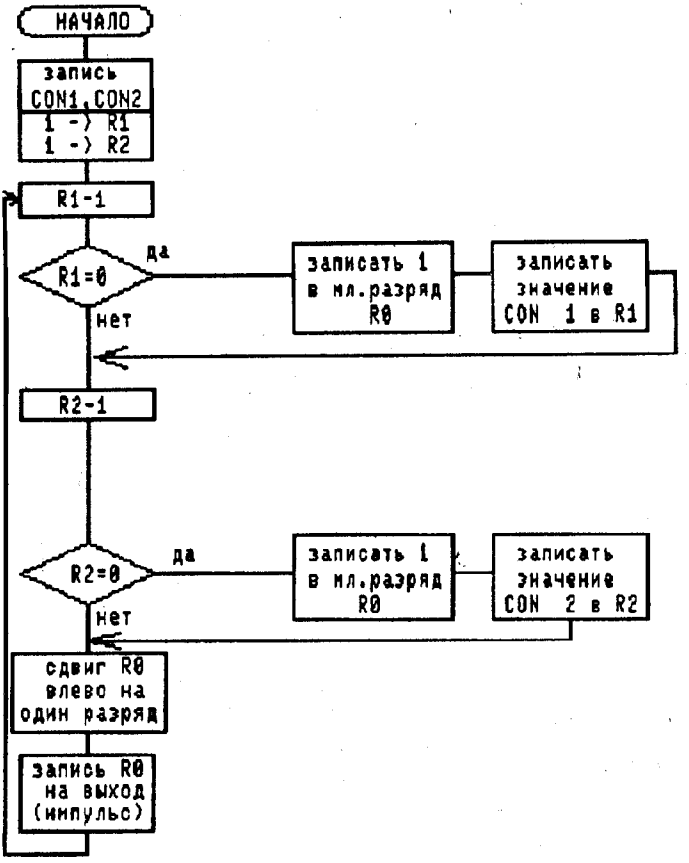
```
            ; ****************************************
            ; *       ДВУХГОЛОСИЕ НА БК-0010         *
            ; *      ЭТЮД ТАЛАКИНА (ФРАГМЕНТ)        *
            ; *   РЫБАЛЬЧЕНКО  Г.МАЙСКИЙ  КБАССР     *
            ; ****************************************
START:  MOV #MEL1,R4
      ; НАЧАЛЬНАЯ ЗАГРУЗКА
      ; АДРЕСОВ МЕЛОДИЙ
        MOV #MEL2,R5
        MOV #1,DL1
        MOV #1,DL2
ZAGR:   DEC DL1
      ; РАСХОД ДЛИТ. ЗВУКА
        BEQ WYB1
      ; ВЫБОРКА ЗВУКА, 1 КАНАЛ
A2:     DEC DL2
      ; РАСХОД ДЛИТ. ЗВУКА
        BEQ WYB2
      ; ВЫБОРКА ЗВУКА, 2 КАНАЛ
        BR ST
WYB1:   MOVB (R4)+,DL1
      ; БЛОК ВЫБОРКИ ДЛИТЕЛЬНОСТИ
      ; И ЧАСТОТЫ ЗВУКА 1 КАНАЛА
        SUB #60,DL1
        BEQ END
        MOVB (R4)+,R0
        JSR PC,NOTY
        MOV R0,CON1
        BR A2
WYB2:   MOVB (R5)+,DL2
      ; БЛОК ВЫБОРКИ ДЛИТЕЛЬНОСТИ
      ; И ЧАСТОТЫ ЗВУКА 2 КАНАЛА
        SUB #60,DL2
        BEQ END
        MOVB (R5)+,R0
        JSR PC,NOTY
        ASL R0
      ; НА ОКТАВУ ВНИЗ
        MOV R0,CON2
ST:     MOV #6000,R3
      ; ДЛИТЕЛЬНОСТЬ=1
        MOV #1,R1
        MOV #1,R2
        MOV #0,R0
CI:     DEC R1
      ; БЛОК ГЕНЕРАЦИИ КАНАЛОВ: 1 КАНАЛ
        BNE 1
        MOV CON1,R1
        BMI 1
        BIS #1,R0
1:      DEC R2
      ;                              2 КАНАЛ
        BNE 2
        MOV CON2,R2
        BMI 2
        BIS #3,R0
2:      DEC R3
      ; R3 -> ДЛИТЕЛЬНОСТЬ
        BEQ ZAGR
3:      ASL R0
        MOV R0,@#177716
      ; ЗВУЧАНИЕ
        BR CI
NOTY:   MOV #BUF1,R1
      ; ПОДПРОГРАММА ПЕРЕКОДИРОВКИ
      ; СИМВОЛОВ НОТ В ЧАСТОТЫ
        MOV #BUF2,R3
CIKL:   TSTB (R3)+
        CMPB (R1)+,R0
        BNE CIKL
        MOVB -(R3),R0
END:    RTS PC
        HALT
CON1:   .#0
CON2:   .#0
DL1:    .#0
DL2:    .#0
BUF1:   .A:F CYUWAEPNRGOL]DZV\:.
        .E ; ЗВУКОРЯД
BUF2:   .B:77.B:377.B:73.B:70.B:65
        .B:62.B:57.B:55.B:52.B:50
        .B:45.B:43.B:41.B:40.B:36
        .B:34.B:32.B:31.B:30.B:26.B:25
```

```
        .E ; ЧАСТОТНЫЕ КОНСТАНТЫ
MEL1:   .A:1F1Y1W1A4P1A1P1A1W3Y1 1Y1W1A1P2R2O2R2P2A2W1F1Y1W1A4P8 0
        .E ; МЕЛОДИЯ ПЕРВОГО ГОЛОСА
MEL2:   .A:3F1 3P1 3Y1 3P1 3P1 3O1 3F1 3P1 3R1 4L8 0
        .E ; МЕЛОДИЯ ВТОРОГО ГОЛОСА
```

Листинг 2

```
        ; *****************
        ; ГЕНЕРАЦИЯ АККОРДА
        ; *****************
        MOV #150,CON1
        MOV #70,CON2
        MOV #1,R1
        MOV #1,R2
ST:     DEC R1
        BNE 1
        BIS #1,R0
        MOV CON1,R0
1:      DEC R2
        BNE 2
        BIS #1,R0
        MOV CON2,R2
2:      ASL R0
        MOV R0,@#177716
        RTS PC
CON1:   .#0
CON2:   .#0
```

Блок-схема программы 2



Предлагаемая в данной статье программа обеспечивает автоматическое перекодирование автономных подпрограмм исполнения мелодий, полученных с помощью музыкального редактора KLAWESIN, обратно во внутренний формат нотного текста. Так что теперь интересные мелодии можно будет не только услышать, но и увидеть.

Д. Ю. Усенков,
Москва

Декомпилятор мелодий для KLAWESINa

Любому пользователю БК-0010(.01) или БК-0011(М) наверняка известен музыкальный редактор KLAWESIN (SW corp.). Помимо возможности исполнения трехголосных мелодий он обладает еще и тем преимуществом, что набранные в нем музыкальные фрагменты можно сохранять на магнитной ленте или диске не только в файлах нотного текста (чтобы вновь загружать их в KLAWESIN для дополнения или редактирования), но и в виде автономных подпрограмм воспроизведения музыки. Последние можно вставлять в свои программы в качестве музыкального оформления, как это и делается во многих играх, формировать из них «музыкальные шкатулки» и т. д.

Но есть у KLAWESINa и кое-какие недостатки, один из которых — невозможность обратной загрузки в музыкальный редактор сгенерированной им автономной исполняющей подпрограммы. Восстановить же исходный нотный текст играемой мелодии вручную можно, лишь обладая отменным музыкальным слухом и располагая изрядным количеством времени.

Предлагаемая вниманию читателей несложная программа KLDECOMP позволяет почти полностью разрешить указанные проблемы. Она представляет собой автоматический декомпилятор исполняемых модулей KLAWESINa во внутренний формат нотных текстов этого музыкального редактора.

```
;
;       Декомпилятор исполняемых мелодий для KLAWESINa
;
;              SCREW (c)    MOCKBA    1994
;-------------------------------------------------------------
START:  MOV @#4,-(R6)       ; сохранить вектор выхода в ДОС
        EMT 14
        MOV (R6)+,@#4       ; восстановить вектор выхода
        MOV #232,R0         ; выключить курсор
        EMT 16
        INC R0              ; R0=233 - режим 32 символа в строке
        EMT 16
        MOV #52525,@#212    ; синий фон
        MOV #125252,@#214   ; зеленый цвет
        MOV #14,R0          ; закрасить экран синим
        EMT 16
        MOV #40,R5          ; рисование муара
        MOV PC,R0           ; R0 не равно 0
        CLR R4              ; X средней точки
```

```
1:      CLR R1              ; — проход влево —
        CLR R2
        EMT 30
        MOV R4,R1
        MOV #170,R2
        EMT 32
        CLR R1
        MOV #357,R2
        EMT 32
        ADD #10,R4
        SOB R5,1
        MOV #40,R5
        MOV #377,R4
2:      MOV #377,R1         ; — проход вправо —
        CLR R2
        EMT 30
        MOV R4,R1
        MOV #170,R2
        EMT 32
        MOV #377,R1
        MOV #357,R2
        EMT 32
        SUB #10,R4
        SOB R5,2            ; === муар выведен ===
        MOV #1,R1
        MOV #2,R2
        EMT 24
        MOV #177777,@#214   ; красный цвет
        MOV #ZAST,R1        ; строка - заставка
        CLR R2
        EMT 20
        MOV #1,R1
        MOV #24,R2
        EMT 24
        MOV #ZAIM1,R1       ; запрос имени для чтения
        CLR R2
        EMT 20
        MOV #BUFNM1,R1      ; буфер для первого имени
        JSR PC,INPNAM       ; ввод имени с клавиатуры
        MOV #1,R1
        MOV #25,R2
        EMT 24
        MOV #ZAIM2,R1       ; запрос имени для записи
        CLR R2
        EMT 20
        MOV #BUFNM2,R1      ; буфер для второго имени
        JSR PC,INPNAM       ; ввод имени с клавиатуры
        CLR @#212
```

```
        MOV #214,R0         ; включить режим РП
        EMT 16
        MOV #233,R0         ; режим 64 символа в строке
        EMT 16
        MOV #TXT1,R1        ; — Чтение исходного файла —
        CLR R2
        EMT 20
        MOV #320,R1         ; заполнение буфера EMT 36
        MOV #3,(R1)+
        MOV #10000,(R1)+    ; адрес загрузки исходного файла
        CLR (R1)+
        MOV #20,R2
        MOV #BUFNM1,R0
3:      MOVB (R0)+,(R1)+
        SOB R2,3
4:      MOV #320,R1
        EMT 36
        TSTB @#321
        BEQ 5
        MOV #ERR,R1         ; Ошибка !!!
        CLR R2
        EMT 20
        EMT 6
        BR 4
5:      MOV #10000,R5     ; адрес исполняемой нотной последовательности
        CMP 6(R5),#12705  ; имеется ли блок воспроизведения?
        BNE 6             ; нет
        MOV #TXT2,R1      ; — Блок воспроизведения найден —
        CLR R2
        EMT 20
        MOV 164(R5),R0    ; темп
        ADD #170,R5       ; коррекция адреса начала нот
        CLR R4
7:      CMP TABTMP(R4),R0 ; TABTMP - таблица темпов
        BEQ 8             ; темп найден
        ADD #2,R4
        CMP R4,#20        ; вся ли таблица просмотрена?
        BLT 7             ; еще нет
        MOV #TXT3,R1      ; ... Темп не распознается ...
        CLR R2
        EMT 20
        EMT 6
        BR 9              ; на обработку нот
8:      ASR R4            ; R4=R4/2
        ADD #61,R4        ; номер темпа в символьном виде
        MOV #TXT4,R1      ; "... Темп исполнения:"
        CLR R2
        EMT 20
```

```
        MOV #233,R0       ; режим 32 символа в строке
        EMT 16
        MOV R4,R0         ; вывести значение темпа
        EMT 16
        MOV #233,R0       ; режим 64 символа в строке
        EMT 16
        MOV #12,R0        ; перевод строки
        EMT 16
        EMT 6
        BR 9
6:      TSTB 7(R5)        ; это длительность первой ноты?
        BNE 10            ; нет - блок воспроизведения нестандартный
        MOV #TXT5,R1      ; Блок воспроизведения отсутствует
        CLR R2
        EMT 20
        EMT 6
        BR 9
10:     MOV #TXT6,R1      ; Нестандартный блок воспроизведения
        CLR R2
        EMT 20
        MOV #TXT7,R1      ; "Нажмите любую клавишу"
        CLR R2
        EMT 20
        EMT 6
        MOV #214,R0
        EMT 16
        HALT              ; аварийный выход в ДОС
9:      MOV #TXT8,R1      ; "Ждите. Идет декомпиляция..."
        CLR R2
        EMT 20
        MOV #40000,R4     ; буфер формирования нотного текста
100:    MOV (R5)+,NOTA1   ; буфер первой ноты
        MOV (R5)+,NOTA2   ; буфер второй ноты
        MOV (R5)+,NOTA3   ; буфер третьей ноты
        MOV (R5)+,DLIT    ; буфер длительности
        MOV @#202,R1      ; адрес начала поля служебной строки
        ADD #10,R1        ; и
        MOV #10,R0        ;  н
17:     MOV R1,-(R6)      ;   д
        MOV NOTA1,(R1)+   ;    и
        MOV NOTA2,(R1)+   ;     к
        MOV NOTA3,(R1)+   ;    а
        MOV DLIT,(R1)+    ;   ц
        MOV (R6)+,R1      ;  и
        ADD #100,R1       ; я
        MOV #2000,R2      ; задержка, чтобы индикация
30:     SOB R2,30         ; была заметна
        SOB R0,17         ; .........................
```

```
        TST NOTA1          ; нуль - конец исходного файла
        BEQ 111
        CMP NOTA1,#77777   ; проверка на паузу
        BNE 11             ; нет
        CMP NOTA2,#77777
        BNE 11
        CMP NOTA3,#77777
        BNE 11
        MOVB #40,(R4)+     ; пауза: #40 в мл. байт
        JSR PC,FNDDLT      ; длительность по DLIT -> R0
        MOVB R0,(R4)+      ; запись длительности паузы
        CLR (R4)+          ; остальные ноты обнулить
        CLR (R4)+
        BR 100             ; следующая нота
11:     CMP NOTA1,#77777   ; первый голос выключен?
        BNE 12             ; нет
        CLR (R4)+          ; а если да, то обнулить ноту
        BR 13              ;  и перейти ко второй
12:     MOV NOTA1,R3
        JSR PC,FNDNOT      ; нота по R3 -> R0
        MOVB R0,(R4)+
        JSR PC,FNDDLT      ; определить длительность
        MOVB R0,(R4)+
13:     CMP NOTA2,#77777   ; второй голос выключен?
        BNE 14             ; нет
        CLR (R4)+          ; а если да, то обнулить ноту
        BR 15              ;  и перейти к третьей
14:     MOV NOTA2,R3
        JSR PC,FNDNOT      ; определить ноту
        MOVB R0,(R4)+
        JSR PC,FNDDLT      ; определить длительность
        MOVB R0,(R4)+
15:     CMP NOTA3,#77777   ; третий голос выключен?
        BNE 151            ; нет
        CLR (R4)+          ; а если да, то обнулить ноту
        BR 100             ;  и перейти к следующей триаде
151:    MOV NOTA3,R3
        JSR PC,FNDNOT      ; определить ноту
        MOVB R0,(R4)+
        JSR PC,FNDDLT      ; определить длительность
        MOVB R0,(R4)+
        BR 100             ; следующая триада
111:    MOV #14,R0         ; в конец нотного файла добавить 12 нулей
16:     CLR (R4)+
        SOB R0,16
        MOV #TXT9,R1       ; — Запись созданного файла —
        CLR R2
        EMT 20
```

```
        MOV #320,R1
        MOV #2,(R1)+
        MOV #40000,(R1)+
        MOV R4,@R1
        SUB #40000,(R1)+
        MOV #20,R0
        MOV #BUFNM2,R2
20:     MOVB (R2)+,(R1)+
        SOB R0,20
22:     MOV #320,R1
        EMT 36
        TSTB @#321
        BEQ 21
        MOV #ERR,R1
        CLR R2
        EMT 20
        EMT 6
        BR 22
21:     MOV #TXT10,R1  ; — Обработка завершена —
        CLR R2
        EMT 20
        MOV #TXT7,R1
        CLR R2
        EMT 20
        EMT 6
        MOV #214,R0
        EMT 16
        HALT           ; выход в ДОС
;
ZAST:   .A: SCREW (c) MUSIC DECOMP  v1.0
        .B:40.E
ZAIM1:  .A:Читаем :
        .B:40.E
ZAIM2:  .A:Пишем  :
        .B:40.E
TXT1:   .A: — Чтение исходного файла —
        .B:12.E
ERR:    .B:234.B:7.B:7.B:7.B:7.B:7
        .A:  Ошибка чтения/записи
        .B:40.B:40.B:234.B:12.E
TXT2:   .A: — Блок воспроизведения найден —
        .B:12.E
TXT3:   .B:234.B:7.B:7.B:7.B:7.B:7
        .A:  Нераспознаваемый темп
        .B:40.B:40.B:234.B:12.E
TXT4:   .A:     ... Темп исполнения:
        .B:40.E
TXT5:   .A: — Блок воспроизведения отсутствует —
```

```
        .B:12.E
TXT6:   .B:234.B:7.B:7.B:7.B:7.B:7
        .A:  Нестандартный блок воспроизведения
        .B:40.B:40.B:234.B:12.E
TXT7:   .A:          Нажмите любую клавишу
        .B:12.E
TXT8:   .A: — Ждите. Идет декомпиляция... —
        .B:12.E
TXT9:   .A: — Запись созданного файла —
        .B:12.E
TXT10:  .A: — Обработка завершена —
        .B:12.E
TXT11:  .B:234.B:7.B:7.B:7.B:7.B:7
        .A:  Нераспознаваемая длительность
        .B:40.B:40.B:234.B:12.E
TXT12:  .B:234.B:7.B:7.B:7.B:7.B:7
        .A:  Нераспознаваемый тип ноты
        .B:40.B:40.B:234.B:12.E
BUFNM1: .+30
BUFNM2: .+30
NOTA1:  .#0
NOTA2:  .#0
NOTA3:  .#0
DLIT:   .#0
;
; ввод имени в буфер (начало - @R1)
INPNAM: MOV R1,-(R6)
        MOV #20,R0
1:      MOVB #40,(R1)+
        SOB R0,1
        MOV @R6,R1
        MOV #5020,R2
        EMT 10
        MOV (R6)+,R1
        MOV #20,R0
2:      CMPB (R1)+,#12
        BEQ 3
        SOB R0,2
        BR 4
3:      MOVB #40,-(R1)
4:      RTS PC
;
; перекодировка длительности и запись результата в R0
FNDDLT: CLR R1
1:      CMP TABDL1(R1),DLIT  ; TABDL1 - табл.длит.исходн.
        BEQ 2
        ADD #2,R1
        CMP R1,#30           ; всего &014 значений длительности
```

```
        BLT 1
        MOV #TXT11,R1        ; нераспознаваемая длительность
        CLR R2
        EMT 20
        MOV #TXT7,R1
        CLR R2
        EMT 20
        EMT 6
        MOV #214,R0
        EMT 16
        HALT                 ; аварийный выход в ДОС
2:      MOV TABDL2(R1),R0    ; TABDL2 - табл.длит.результ.
        RTS PC
;
; перекодировка нот (из R3) и запись результата в R0
FNDNOT: CLR R1
1:      CMPB TABNT1(R1),R3   ; TABNT1 - ноты исходного файла
        BEQ 2
        ADD #2,R1
        CMP R1,#140          ; всего &060 значений нот
        BLT 1
        MOV #TXT12,R1        ; нераспознаваемый тип ноты
        CLR R2
        EMT 20
        MOV #TXT7,R1
        CLR R2
        EMT 20
        EMT 6
        MOV #214,R0
        EMT 16
        HALT                 ; аварийный выход в ДОС
2:      MOVB TABNT2(R1),R0   ; TABNT2 - ноты результата
        RTS PC
;
; таблица значений темпа исполнения
TABTMP: .#500.#700.#1100.#1300
        .#1500.#1700.#2100.#2300.E
;
; таблица длительностей исходного файла
TABDL1: .#100.#40.#20.#10.#4.#2
        .#140.#60.#30.#14.#6.#3.E
;
; таблица длительностей результирующего файла
TABDL2: .#1.#2.#3.#4.#5.#6
        .#201.#202.#203.#204.#205.#206.E

; таблица частот нот исходного файла
TABNT1: .#334.#321.#305.#272.#257.#246
```

```
            .#235.#224.#213.#203.#174.#166
            .#156.#150.#143.#134.#127.#122
            .#116.#111.#105.#101.#76.#72
            .#66.#63.#61.#56.#53.#51.#46
            .#44.#42.#40.#36.#34.#33.#31
            .#27.#26.#25.#24.#22.#21.#20
            .#17.#16.#15.E
;
; таблица частот нот результирующего файла
TABNT2:     .#1.#201.#2.#202.#3.#4.#204.#5.#205
            .#6.#206.#7.#10.#210.#11.#211.#12.#13
            .#213.#14.#214.#15.#215.#16.#17.#217
            .#20.#220.#21.#22.#222.#23.#223.#24
            .#224.#25.#26.#226.#27.#227.#30.#31
            .#231.#32.#232.#33.#233.#34.E
;
; ---------------------------------------------------------

            END
```

Работать с программой несложно. После запуска и вывода на экран «муаровой» заставки декомпилятор запрашивает имена читаемого (содержащего автономную подпрограмму воспроизведения мелодии) и записываемого файла (содержащего сгенерированный нотный текст для загрузки в KLAWESIN). Ввод имен производится по общепринятым правилам (исправление ошибок с помощью клавиши «ЗАБОЙ», окончание ввода — клавиша «ВВОД», можно указывать имя устройства A:, B: или M:, и прочее, если это позволяет данная дисковая ОС, и т. д.).

После этого программа переходит в автоматический режим, и пользователю остается только ждать и принимать к сведению выводимые на экран сообщения (каждое действие, выполняемое программой, обязательно сопровождается хотя бы одной комментирующей строкой).

Вначале читается файл с заданным именем, при ошибке программа останавливается с выводом соответствующего сообщения и после нажатия любой клавиши повторяет попытку чтения. Затем считанный модуль анализируется на наличие в его начале блока воспроизведения (которого может и не быть — KLAWESIN позволяет записывать и просто исполняемую последовательность нот в расчете на использование другого воспроизводящего блока). Если этот блок найден, из него извлекается и выводится на экран константа темпа исполнения (ее нужно записать для последующей установки в KLAWESINe), иначе выдается сообщение об отсутствии блока (тогда темп придется подбирать на слух). Если же блок воспроизведения нестандартный (например, в играх он часто дорабатывается для обеспечения возможности прерывания исполнения нажатием на любую клавишу), программа выдает об этом сообщение и прекращает работу. Нестандартные блоки не обрабатываются, так как неизвестно, где заканчивается сам блок и начинается последовательность нот в каждом конкретном варианте. В таком случае можно попытаться загрузить исходную подпрограмму в отладчик (например, DEBU10) и «отсечь» от исполняемой нотной последовательности «попорченный» блок воспроизведения. То же самое можно попробовать сделать, если программа вроде бы опознает блок как стандартный, но при начале декомпиляции сразу же сообщает о неопознанной ноте или длительности — алгоритм анализа блока воспроизведения упрощен и может оказаться, что данный блок похож на стандартный, но его длина другая,

и вместо нот программа пытается декомпилировать коды команд. В том же отладчике при наличии известного опыта можно найти и константу темпа.

Далее производится собственно декомпиляция нотной последовательности, что индицируется слева в служебной строке, и если все прошло успешно — запись созданного файла также с контролем на ошибку (при необходимости с повторной попыткой). Во всех случаях приостанова работы программы для продолжения нужно нажимать пробел.

Сгенерированный декомпилятором файл — это и есть исходный нотный текст музыкального фрагмента, который затем можно загрузить в KLAWESIN, просматривать в нем, редактировать, менять темп, а при желании — записать новую, измененную автономную подпрограмму воспроизведения.

Следует обратить внимание на две особенности, связанные с работой самого музыкального редактора KLAWESIN. Во-первых, в нем (в частности, в общераспространенной версии 1.01, для которой и был написан декомпилятор) неправильно реализована обработка мелодии, когда в одной и той же триаде стоят ноты разной длительности. При декомпиляции исполняемой последовательности для подобного музыкального фрагмента возможно нераспознавание нот или длительностей в такой триаде либо их неправильная расшифровка. Во-вторых, так как частоты «полутоновых» нот (которые на фортепиано играются на черных клавишах) одни и те же, скажем, для ре-диез и ми-бемоль или для ля-диез и си-бемоль, соответствующие значения высоты тона в исполняемой последовательности для таких пар нот одинаковы. Определить точно, имелся ли в виду диез или бемоль, в данном случае невозможно, и декомпилятор распознает «полутона» только как диезы (хотя звучать сгенерированная нотная запись будет абсолютно так же, как исходная). В остальном же программа KLDECOMP работает достаточно надежно (автору удалось без каких-либо проблем декомпилировать все исполняемые модули ранее набранных мелодий, а также модули, «вырезанные» с помощью отладчика из KLAWESINa и ряда игровых программ.

В заключение (для желающих доработать декомпилятор по своему вкусу, модернизировать его и т. п.) приведем некоторые справочные сведения о формате записи нотных текстов и автономных исполияемых модулей музыкального редактора KLAWESIN.

Формат записи нотного текста

Каждая триада нот (соответствующая трем голосам в одной нотной позиции) занимает три двухбайтных машинных слова, в которых в старшем байте кодируется длительность, а в младшем — тип ноты (а также признаки паузы и репризы). Если в данной нотной позиции задействованы не все три голоса, неиспользуемые машинные слова содержат нули. Завершается кодированная запись двенадцатью (число десятичное) нулевыми двухбайтными словами.

Длительность нот и пауз (для реприз — всегда ноль) в старшем байте кодируется в соответствии с табл. 1 (все числа здесь и в других таблицах — восьмеричные), из которой видно, что признаком наличия точки, означающей увеличение длительности в 1.5 раза, является «1» в старшем бите (седьмом — для байта, пятнадцатом — для всего соответствующего данной ноте машинного слова в целом).

Таблица 1. Коды длительностей нотной записи

Длительность	Целая	1/2	1/4	1/8	1/16	1/32
Код	1	2	3	4	5	6
Код при наличии точки	201	202	203	204	205	206

Тип ноты, записанный в младшем байте, можно расшифровать с помощью табл. 2.

Таблица 2. Коды типов нот в нотной записи

Нота/Октава	Малая	Первая	Вторая	Третья
до	1	10	17	26
до #	201	210	217	226
ре b	102	111	120	127
ре	2	11	20	27
ре #	202	211	220	227
ми b	103	112	121	130
ми	3	12	21	30
фа	4	13	22	31
фа #	204	213	222	231
соль b	105	114	123	132
соль	5	14	23	32
соль #	205	214	223	232
ля b	106	115	124	133
ля	6	15	24	33
ля #	206	215	224	233
си b	107	116	125	134
си	7	16	25	34

Знаки «#» и «b» здесь означают «диез» и «бемоль» соответственно. Из табл. 2 видно, что признаком диеза является «1» в старшем бите (седьмом как для байта, так и для машинного слова в целом), а признаком бемоля — «1» в шестом бите. Значения остальных битов определяются названием ноты, например, ре-диез первой октавы — это код 211 (ре — 11, диез — «1» в седьмом бите), а ми-бемоль той же октавы — это уже 112 (ми — 12, бемоль — «1» в шестом бите).

В том же младшем байте записываются коды паузы (восьмеричное 40) и реприз (левая, открывающая — 41, а правая, закрывающая — 42).

Формат записи автономной подпрограммы и исполняемой последовательности

KLAWESIN позволяет записывать исполняемые последовательности нот как отдельно, так и вместе с блоком воспроизведения. Признаком наличия стандартного блока воспроизведения в наиболее упрощенном алгоритме анализа можно считать восьмеричное число 12705 в ячейке с адресом <a>+6, где <a> — адрес начала загруженного в память файла (признаком наличия любого блока воспроизведения можно считать и просто ненулевой старший байт в данной ячейке: как мы убедимся чуть позже, при отсутствии блока на этом месте должна была бы оказаться длительность первой триады нот, а длительности всегда имеют нулевой старший байт). Именно так организован анализ блока воспроизведения в декомпиляторе, если же число в этой ячейке не равно 12705, но ее старший байт ненулевой, считается, что блок нестандартный (при желании можно реализовать более «обстоятельный» алгоритм анализа).

Если имеется стандартный блок воспроизведения, сама исполняемая нотная последовательность начинается с адреса <a>+170, а в ячейке <a>+164 хранится код темпа исполнения (табл. 3).

Таблица 3. Коды темпа исполнения

Темп	1	2	3	4	5	6	7	8
Код	500	700	1100	1300	1500	1700	2100	2300

Каждая триада (трехголосная нотная позиция) соответствует в исполняемой нотной последовательности четырем двухбайтным машинным словам, из которых первые три кодируют высоты тонов, а четвертое — длительность. Если в данной триаде задействованы не все три голоса, неиспользуемые машинные слова содержат восьмеричный код 77777, пауза же означает, по сути, временное выключение всех трех голосов и 77777 во всех трех машинных словах (а в четвертом записана, как и для нот, длительность паузы). Каких-либо отдельных кодов для реприз в исполняемой последовательности не предусмотрено: отмеченные ими музыкальные фрагменты просто повторены дважды. Признаком окончания исполняемой нотной последовательности являются четыре нулевых двухбайтных машинных слова.

Длительности нот и пауз (в четвертом слове) кодируются так, как показано в табл. 4. Как видим, при наличии точки длительность действительно увеличивается в 1.5 раза.

Таблица 4. Коды длительностей исполняемых нот

Длительность	Целая	1/2	1/4	1/8	1/16	1/32
Код	100	40	20	10	4	2
Код при наличии точки	140	60	30	14	6	3

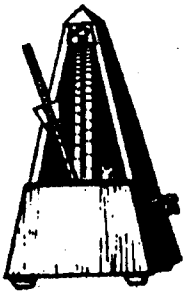
Высоту тона нот (первые три слова) можно определить, воспользовавшись табл. 5.

Таблица 5. Коды высоты тона исполняемых нот

Нота/Октава	Малая	Первая	Вторая	Третья
до	334	156	66	33
до #, ре b	321	150	63	31
ре	305	143	61	27
ре #, ми b	272	134	56	26
ми	257	127	53	25
фа	246	122	51	24
фа #, соль b	235	116	46	22
соль	224	111	44	21
соль #, ля b	213	105	42	20
ля	203	101	40	17
ля #, си b	174	76	36	16
си	166	72	34	15

Ассемблерный листинг стандартного блока воспроизведения

```
; Адреса     Коды      Мнемоники          Комментарии
  001000:    010700    MOV  R7,R0         ; R0 <- адрес начала последовательности
  001002:    062700    ADD  #164,R0       ;  нот (перемещаемый формат)
  001004:    000164
  001006:    012705    MOV  #100,R5       ; маска звукового бита в рег. @#177716
  001010:    000100
  001012:    016001    MOV  2(R0),R1      ; чтение высот тонов
  001014:    000002                       ;
  001016:    016002    MOV  4(R0),R2      ;  трех голосов
  001020:    000004                       ;
  001022:    016003    MOV  6(R0),R3      ;  текущей нотной позиции
  001024:    000006
  001026:    016067    MOV  10(R0),DL     ; чтение длительности в буфер
  001030:    000010
  001032:    000126
  001034:    016704    MOV  TMP,R4        ; загрузка темпа из буфера
  001036:    000124
  001040:    005301    DEC  R1            ; отсчет первого голоса
  001042:    003010    BGT  001064        ; еще не исчерпано?
  001044:    050537    BIS  R5,@#177716   ; если уже нуль, то
  001046:    177716                       ;  выдать "1" на динамик,
  001050:    016001    MOV  2(R0),R1      ;  вновь загрузить высоту тона
  001052:    000002                       ;
  001054:    005303    DEC  R3            ;  и вести отсчет для
  001056:    005302    DEC  R2            ;  двух других голосов,
  001060:    040537    BIC  R5,@#177716   ;  а потом "0" на динамик
  001062:    177716
  001064:    005302    DEC  R2            ; отсчет второго голоса
  001066:    003010    BGT  001110        ; еще не исчерпано?
  001070:    050537    BIS  R5,@#177716   ; если уже нуль, то
  001072:    177716                       ;  выдать "1" на динамик,
  001074:    016002    MOV  4(R0),R2      ;  вновь загрузить высоту тона
  001076:    000004                       ;
  001100:    005301    DEC  R1            ;  и вести отсчет для первого
  001102:    005303    DEC  R3            ;  и третьего голосов,
  001104:    040537    BIC  R5,@#177716   ;  а потом "0" на динамик
  001106:    177716
  001110:    005303    DEC  R3            ; отсчет третьего голоса
  001112:    003010    BGT  001134        ; еще не исчерпано?
  001114:    050537    BIS  R5,@#177716   ; если уже нуль, то
  001116:    177716                       ;  выдать "1" на динамик,
  001120:    016003    MOV  6(R0),R3      ;  вновь загрузить высоту тона
  001122:    000006                       ;
  001124:    005301    DEC  R1            ;  и вести отсчет для первого
  001126:    005302    DEC  R2            ;  и второго голосов,
  001130:    040537    BIC  R5,@#177716   ;  а потом "0" на динамик
  001132:    177716
  001134:    077437    SOB  R4,001040     ; число повторов - код темпа
  001136:    005367    DEC  DL            ; отсчет длительности
  001140:    000020
  001142:    001334    BNE  001034        ; если еще не все, повтор заново
  001144:    062700    ADD  #10,R0        ; иначе - следующая нотная позиция
  001146:    000010
  001150:    005760    TST  10(R0)        ; длительность равна 0
  001152:    000010                       ;  (т.е. конец мелодии)?
  001154:    001316    BNE  001012        ; нет - играть следующую ноту
  001156:    000207    RTS  PC            ; да - выход из подпрограммы
  001160:    000000    .#0                ;
  001162:    000000    .#0                ; DL: - буфер длительности
  001164:    001300    .#1300             ; TMP: - значение темпа
  001166:    000000    .#0                ;
  001170:    ; отсюда начинается исполняемая последовательность
```



Жалобы многострадальных пользователей БК-0011, который по странной прихоти его создателей оказался не совместим ни с БК-0010(.01), ни с новой моделью БК-0011М, давно уже не новость на страницах многих изданий, публикующих (или публиковавших ранее) материалы об этом семействе компьютеров. Программ, написанных специально для БК-0011, практически нет (или очень мало), а ПО от «десятки» можно использовать лишь при наличии монитора среды БК-0010(.01) и только после некоторой доработки. В частности, приходится вносить изменения в команды генерации звука: на одиннадцатой модели с пьезодинамиком связан совсем другой бит регистра 177716, нежели в БК-0010(.01).
Для облегчения этой долгой и нудной работы автор данной статьи предлагает автоматический модификатор программ с звуковыми эффектами, делающий их универсальными — работающими на БК любой модели.

С. П. Чабан,
г. Томск-51

Чтобы зазвучал БК-0011...

Известно, что в БК-0011 аппаратные средства подачи звуковых сигналов отличаются от принятых в БК-0010(.01) и БК-0011М; поэтому в некоторых программах, написанных для последних, «молчит» клавиатура, не работают директивы вывода из программ, звуковые эффекты, речь. Эти недостатки можно устранить с помощью отладчика, найдя и исправив в программе команды «генерации звука»: на БК-0011 обращаться к регистру управления системными внешними устройствами (@#177716) по записи следует только как к целому слову, устанавливая «1» в его 11-й разряд.

Работа эта кропотливая и рутинная, и требует определенного навыка пользователя, что для начинающего не всегда легко, а для профессионала неинтересно и отнимает время, необходнмое для творческой работы.

Большинство звуковых эффектов на БК-0010(.01) и БК-0011М достигается непосредственной записью в регистр @#177716 констант 220 и 320 (MOV #220,@#177716 и MOV #320,@#177716) или других, устанавливающих/снимающих «1» в шестом разряде системного регистра.

Способ, предлагаемый А. М. Надежиным в журнале «Вычислительная техника и ее применение», №1 за 1992 г., — замена констант 220 и 320 на 16220 и 16320 соответственно — озвучивает программы для БК-0011 лишь при условии, что эмулятор БК-0010 находится в четвертой странице ОЗУ, а использование переделанных программ на БК-0011М или при другой конфигурации страниц ОЗУ опять требует их корректировки. Есть ли выход из положения? Да, и очень простой.

Некоторыми программистами давно применяется следующий принцип: использование в качестве команд обращения к регистру 177716 (для установки или сброса 6-го разряда) MOV @#102064,@#177716 и MOV @#102076,@#177716. Дело в том, что в системных ячейках монитора 102064 и 102076 содержатся константы подзвучки нажатия клавиш и подключения к адресному пространству страниц ОЗУ/ПЗУ (на БК-0011) для различных операционных систем. Программа, используя эти ячейки, как бы распознает тип БК и засылает в регистр 177716 нужные константы. Поэтому она будет «звучать» на всех типах БК.

Призываю всех программирующих на БК именно к такому способу обращения к регистру 177716 для работы со звуком (конечно, если это позволяет объем памяти).

Предлагаемый «звукоконвертор» SOUND.CON использует вышеуказанный принцип и работает под ANDOS (на БК-0011 в среде БК-0010). Звукоконвертор разработан в системе «Автор 3.1» (НПО «Альтернатива», г. Барнаул) на базе класса УКНЦ.01 (БК-0011) с использованием кодовых подпрограмм, тексты которых приводятся ниже. После успешной отладки любая программа со звуком работает на любой модели БК в различных ОС (ANDOS, NORD, NORTON и др.), а также в классе УКНЦ.01 с локальной сетью под управлением NETALT, при условии, что в эмуляторе БК-0010 работает «щелчок» клавиатуры.

Хочу сразу предупредить читателя, что «звукоконвертор» не всесилен и успешно обрабатывает лишь программы с непосредственным обращением к регистру 177716 трехадресными командами. Во всех других случаях озвучивание требует специальных подходов и в данной статье не рассматривается.

Описание работы с программой

«Звукоконвертор» работает под ANDOS V2.23 на БК-0011(М). После запуска на экране появляется меню: «LOAD», «A:», «B:», «SAVE» и «ввод имени файла». Выбор осуществляется перемещением стрелки-указателя и нажатием клавиши «ВВОД».

Перемещать указатель можно от мыши или удержанием в нажатом состоянии клавиш «←», «→», «↑», «↓». При начале работы указатель автоматически устанавливается на «LOAD». После нажатия клавиши «ВВОД» (для чтения файла) предлагается выбрать дисковод A: или B: (по умолчанию A:), затем остается набрать имя файла и нажать клавишу «ВВОД». После загрузки возможны сообщения: «NO-FILE» — файл не найден, «NO-COD» — перекодировка данного файла невозможна. В случае успешной обработки указатель автоматически устанавливается на «SAVE» (запись на диск). При записи файла с тем же именем достаточно в ответ на запрос имени нажать клавишу «ВВОД».

Автоматическое перемещение стрелки-указателя осуществляется по принципу «не прочитаешь — не запишешь, не запишешь — не прочитаешь», большинство перемещений делается по умолчанию, что существенно повышает надежность и удобство работы с программой.

Особенности работы программы

«Звукоконвертор» SOUND.CON блокирует сообщения ANDOS, поэтому при ошибках чтения/записи (компьютер «ничего не делает», а диск «крутится») не забывайте нажимать соответствующие клавиши.

Записываемый файл с тем же именем, что и существующий на диске, пишется поверх старого без запросов ANDOS!

Программа использует третью страницу ОЗУ, поэтому часть данных на электронном диске (VDISK) при ее работе может быть испорчена.

SOUND.CON может обрабатывать файлы длиной до &O37776 байт.

Листинг кодовых подпрограмм (ассемблер TURBO4)

```
        ;подпрограммы указания устройства ANDOS (A:,B:)
DRIVEA: MOVB #101,@#326
        MOVB #72,@#327
        RETURN
DRIVEB: MOV #102,@#326
        MOV #72,@#327
        RETURN
        ;подпрограмма набора имени файла с клавиатуры
        ;для компоновки системой "АВТОР 3.1"
```

```
NAME:   MOV @#274,-(SP)       ;сохранение вектора прерывания
                              ;от клавиатуры в стеке
        MOV @#60,@#274        ;новый адрес вектора прерывания
        MOV #31,R1            ;установка
        MOV #5,R2             ;курсора
        EMT 24                ;
        MOV #232,R0           ;включение
        EMT 16                ;курсора
        CLR @#177660          ;прерывание от клавиатуры
                              ;разрешено
        EMT 6                 ;чтение кода с клавиатуры
        CMPB #12,R0           ;нажат "ВВОД"
        BEQ N3                ;переход к N3
        MOV #7,R4             ;очистка
        MOV #330,R5           ;буфера
 N1:    CLR (R5)+             ;имени
        SOB R4,N1             ;файла
        MOV #CLS,R1           ;очистка
        CLR R2                ;имени файла
        EMT 20                ;на экране
        BR N2
CLS:    .WORD 20040,20040,20040,20040,20040,20040,0
N2:     MOV #31,R1            ;установка
        MOV #5,R2             ;курсора
        EMT 24                ;на экране
        MOVB R0,@#330         ;передача кода в блок параметров
        EMT 16                ;на экран
        INC R1                ;перемещение
        EMT 24                ;курсора
        MOV #331,R1           ;
        MOV #5013,R2          ;
        EMT 10                ;чтение строки с клавиатуры
        MOV #330,R3
N4:     CMPB #12,(R3)+        ;поиск кода "12"
        BNE N4                ;и замена
        DEC R3                ;его кодом
        MOV #0,(R3)           ; "0"
N3:     MOV #232,R0           ;выключение
        EMT 16                ;курсора
        BIS #100,@#177660     ;запрет прерывания от клавиатуры
        MOV (SP)+,@#274       ;восстановление вектора 274
        RETURN

        ;подпрограмма чтения файла с диска и перекодировки
LOAD:   MOV #3,@#320          ;формирование блока на чтение
        MOV #40000,@#322      ;начальный адрес загрузки
        MOV #34000,@#177716   ;3 стр. ОЗУ к &040000-&0100000
        MOV #100256,@#120150  ;блокировка сообщений ANDOS
```

```
        MOV #320,R1             ;блок параметров по адресу &0320
        EMT 36                  ;чтение файла
        TSTB @#321              ;успешное чтение - переход к
        BEQ L3                  ;метке L3
        MOV #14000,@#177716     ;5 стр. ОЗУ к &040000-&0100000
        CLR @#120150            ;сообщения ANDOS разрешены
        MOV #5,R3
1:      CALL UST                ;начало блока сообщения
        MOV #TEX1,R1
        CLR R2
        EMT 20                  ;сообщение "NO FILE"
        CALL PAUSE
        CALL UST
        MOV #TEX3,R1
        CLR R2
        EMT 20                  ;очистка сообщения
        CALL SOUND              ;звуковой сигнал сообщения
        SOB R3,1                ;повтор блока сообщения
        JMP @#1000              ;возврат на начало "звукоконвертора"
L3:     MOV @#346,ADDN          ;начальный адрес в ячейку ADDN
        MOV @#350,DLIN          ;длина файла в ячейку DLIN
        CLR TEST                ;очистка флага операции
                                ;преобразования файла
        MOV #40000,R1           ;адрес начала обрабатываемого файла
        MOV DLIN,R5             ;длина файла в R5
        ADD R1,R5               ;физический адрес конца файла
C1:     MOV (R1)+,R2            ;три первых слова
        MOV (R1)+,R3            ;файла
        MOV (R1)+,R4            ;в регистры R2, R3, R4
        CMP #177716,R4          ;проверка на @#177716
        BNE C100                ;если нет - переход к C100
        CMP #100,R3             ;проверка на
        BNE C2                  ;
        CMP #12737,R2           ;MOV #100,@#177716
        BEQ YES                 ;
        CMP #52737,R2           ;BIS #100,@#177716
        BEQ YES                 ;
        CMP #42737,R2           ;BIC #100,@#177716
        BEQ NO                  ;
        JMP C100                ;
C2:     CMP #12737,R2           ;проверка на MOV #XXXXXX,@#177716
        BNE C100                ;
        BIT #100,R3             ;в XXXXXX  шестой разряд = 1 ?
        BEQ NO                  ;
YES:    MOV #177716,-(R1)       ;замена на
        MOV #102064,-(R1)       ;команду
        MOV #13737,-(R1)        ;MOV #102064,@#177716
        ADD #2,R1               ;
```

```
        MOV #1,TEST          ;установить 1 в TEST
        JMP C101             ;
NO:     MOV #177716,-(R1)    ;замена на
        MOV #102076,-(R1)    ;команду
        MOV #13737,-(R1)     ;MOV #102076,@#177716
        ADD #2,R1            ;
        JMP C101             ;
C100:   SUB #4,R1            ;переход к следующей тройке.слов
C101:   CMP R1,R5            ;проверка на конец файла
        BMI C1               ;переход к обработке очередной
                             ;тройки слов
        TST TEST             ;проверка содержимого TEST
        BNE EXIT             ;переход к EXIT
        MOV #14000,@#177716  ;5 стр. ОЗУ к &040000-&0100000
        MOV #5,R3            ;
1:      CALL UST             ;начало блока сообщения
        MOV #TEX2,R1         ;
        CLR R2               ;
        EMT 20               ;сообщение "NO COD"
        CALL UST             ;
        CALL PAUSE           ;
        MOV #TEX3,R1         ;
        CLR R2               ;
        EMT 20               ;очистка сообщения
        CALL SOUND           ;звуковой сигнал сообщения
        SOB R3,1             ;повтор блока сообщения
        JMP @#1000           ;возврат на начало "звукоконвертора"
EXIT:   MOV #14000,@#177716  ;5 стр. ОЗУ к &040000-&0100000
        RETURN


        ; подпрограмма записи на диск
SAVE:   MOV ADDN,@#120152    ;адрес загрузки в ячейку 120152
        MOV #202,@#320       ;код операции в блок параметров
        MOV #40000,@#322     ;фактический адрес начала
        MOV DLIN,@#324       ;длина в блок параметров
        MOV #34000,@#177716  ;
        MOV #100256,@#120150 ;блокировка сообщений ANDOS
        MOV #12705,@#121322  ;отключение подтверждения
        MOV #120,@#121324    ;перезаписи файла
        MOV #320,R1          ;
        EMT 36               ;запись файла на диск
        MOV #14000,@#177716  ;5 стр. ОЗУ к &040000-&0100000
        MOV #4767,@#121322   ;включение подтверждения
        MOV #3414,@#121324   ;перезаписи файла
        CLR @#120150         ;разрешение сообщений ANDOS
        RETURN
UST:    MOV #33,R1           ;подпрограмма установки
```

```
        MOV #7,R2               ;курсора
        EMT 24                  ;
        RETURN

SOUND:  MOV DLI,R2              ;подпрограмма звукового
1:      MOV @#102064,@#177716   ;сигнала
        MOV TON,R1              ;
2:      SOB R1,2                ;
        MOV @#102076,@#177716   ;
        MOV TON,R1              ;
3:      SOB R1,3                ;
        SOB R2,1                ;
        RETURN

PAUSE:  MOV #177777,R4          ;подпрограмма паузы
1:      SOB R4,1                ;
        RETURN

ADDN:   .BLKW 1                 ;рабочие ячейки "звукоконвертора"
ADDK:   .BLKW 1                 ;
DLIN:   .BLKW 1                 ;
TEST:   .BLKW 1                 ;
DLI:    .WORD 300               ;
TON:    .WORD 200               ;
TEX1:   .ASCIZ /NO FILE !!!/
TEX2:   .ASCIZ /NO COD  !!!/
TEX3:   .ASCII <40><40><40><40><40><40><40><40><40><40><40><0>
        .EVEN
        HALT
        .END
```

Тексты этих подпрограмм не являются оптимальными и при желании могут быть доработаны пользователем.

От редакции

К сожалению, небольшой объем журнала не позволяет опубликовать кодовый листинг готовой к работе программы SOUND.CON, длина которой превышает 10000_8 байт. Для получения копии программы SOUND.CON и ее модифицированной версии на дискете вы можете обращаться в редакцию по телефону: (095) 151-19-40 и E-MAIL: mail@infoobr.msk.su

TUTTI FRUTTI

Под этой рубрикой, открытой по просьбе наших читателей, будут помещаться небольшие программы, советы, рекомендации. Среди них вы увидите наиболее удачные разработки из числа опубликованных ранее в различной литературе по БК. Ведь не секрет, что даже в столичных библиотеках не всегда удается найти интересующий журнал или книгу.

В. Самойлов,
С.-Петербург

Программирование мелодий на БК-0011М

Среди EMT-прерываний БК-0011М имеется не описанная в руководствах функция EMT 101, которая выдает звуковой сигнал в соответствии с содержимым регистров R0 (длительность) и R1 (высота тона). Значения последних используются те же, что и при работе с оператором БЕЙСИКа BEEP.

Ниже приводятся коды высоты тона для нот первой октавы:

Нота	Код	
	восьм.	дес.
до	644	420
до-диез (ре-бемоль)	614	396
ре	566	374
ре-диез (ми-бемоль)	540	352
ми	515	333
фа	472	314
фа-диез (соль-бемоль)	450	296
соль	430	280
соль-диез (ля-бемоль)	410	264
ля	372	250
ля-диез (си-бемоль)	353	235
си	336	222

Для получения кодов нот из других октав необходимо умножать или делить указанный в таблице код для первой октавы на 2, 4 и т.д.

Пример:

ля первой октавы — 250;

ля второй октавы — 250×2=500;

ля малой октавы — 250/2=125.

Код длительности звучания определяется исходя из того, что его произведение на код высоты тона для нот одной и той же длительности должно быть одинаковым, в зависимости от заданного темпа исполнения.

Пример: если для 1/8 ля первой октавы (высота 250) выбрана длительность 160, то 250×160=40000, тогда:

1/8 си первой октавы (высота 222) — длительность 40000/222=180;

1/8 до первой октавы (высота 420) — длительность 40000/420=95 и т. д.

Кроме рассмотренной функции, интерес представляет и другая «звуковая» EMT-функция — EMT 31. Эта команда позволяет по желанию включать и выключать «щелчок» при нажатии на клавишу: если перед ее вызовом в регистре R0 записано число 100 (восьм.), то звук выключается, другое значение включает его снова.

(Публикуется по материалам журнала «Информатика и образование», №2 за 1993 г. — *Прим. ред.*)

Звук на ассемблере (простейшая подпрограмма)

```
START:    MOV #<высота>,R1
          MOV #<длительность>,R2
1:        MOV R1,R0
          MOV @#102064,@#177716
2:        SOB R0,2
          MOV R1,R0
          MOV @#102076,@#177716
3:        SOB R0,3
          SOB R2,1
          RTS PC
```

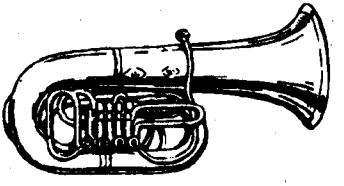
Примечание. Ячейки ПЗУ монитора с адресами 102064 и 102076 содержат значения, которые необходимо записать в регистр 177716, чтобы установить его «звуковой» бит в «0» или «1» (в мониторе эти ячейки входят в состав подпрограммы генерации «щелчков» при нажатии на клавиши). Благодаря их использованию данная подпрограмма универсальна и может использоваться на всех моделях БК-0010(.01) и БК-0011(М).

Ф. Новиков,
г. Ярославль

«Звуковая» подпрограмма в ПЗУ

В ПЗУ монитора БК-0010(.01) содержится подпрограмма, обеспечивающая «щелчок» при нажатии на клавиши. Ее же можно использовать в своих программах на ассемблере для генерации звука возрастающей частоты. Исходные значения высоты тона и длительности задаются перед вызовом в регистрах R2 (длительность) и R3 (высота тона), например:

```
MOV #1000,R2
MOV #1500,R3
JSR PC,@#102062
```



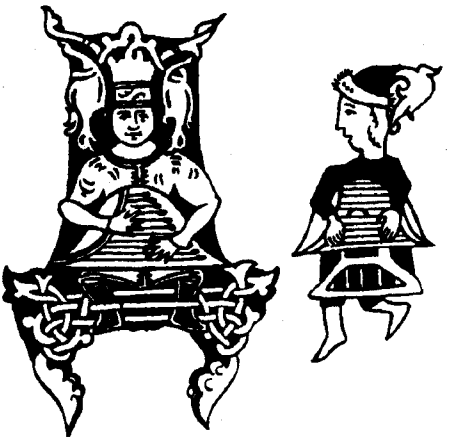
В любом случае значение в R2 должно быть меньше, чем в R3.

(Публикуется по материалам журнала «Информатика и образование», №2 за 1993 г. — *Прим. ред.*)

«Журчание» на БК-0010(.01)

Следующая небольшая подпрограмма позволяет генерировать характерный «журчащий» звук, пока не будет нажата какая-нибудь клавиша, после чего производится выход в вызвавшую программу.

```
   CLR R0
1: MOV #100,R2
2: MOV #3000,R1
3: BIC R0,@#177716
   ADD R2,R0
   CLR @#177716
   BIT #100,@#177716
   BEQ 4
   SOB R1,3
   SOB R2,2
   BR 1
4: RTS PC
```



Изменяя константы в строках с метками 1 и 2, можно получать различный характер «журчания».

С. Ивашинников,
Приморский край

Звуковые эффекты с помощью встроенного таймера

Предлагаемая несложная программа обеспечивает получение разнообразных звуковых эффектов с помощью встроенного таймера БК-0010(.01). Использование таймера (регистры: @#177706 — диапазон счета, @#177710 — сам счетчик, @#177712 — управление скоростью счета; подробности см. в журнале «Информатика и образование», №1 за 1990 г., с. 65, а также №2 за 1990 г., с. 53 и №6 за 1991 г., с. 81) позволяет реализовать несколько звуковых каналов, их количество указывается в ячейке @#1016. Изменяя это значение, а также содержимое ячеек @#1002 и @#1010, можно подобрать различные эффекты. Например, при @#1002 = 2000, @#1010 = 160, @#1016 = 5 (все значения восьмеричные) получается звук скачущего металлического шарика, а при @#1002 = 600, @#1010 = 60, @#1016 = 4 — эффект «заточки ножа». Вообще же ячейка @#1010 может содержать числа 20, 40, 60, 100, 140, 160, 200, 300, 340, 360, а @#1016 (количество псевдоканалов) — от 1 до 340.

```
;  адрес    код      мнемоника
   001000   012737   MOV  #140, @#177706
   001002   000140
   001004   177706
   001006   012737   MOV  #40, @#177712
   001010   000040
```

```
001012   177712
001014   012700   MOV  #4, R0
001016   000004
001020   012701   MOV  #2000, R1
001022   002000
001024   005341   DEC  -(R1)
001026   003402   BLE  1034
001030   077003   SOB  R0, 1024
001032   000770   BR   1014
001034   013711   MOV  @#177710, (R1)
001036   177710
001040   052737   BIS  #100, @#177716
001042   000100
001044   177716
001046   042737   BIC  #100, @#177716
001050   000100
001052   177716
001054   000757   BR   1014
```

(Публикуется по материалам журнала «Информатика и образование». — *Прим. ред.*)

А. Ланеев

Цветомузыка в БЕЙСИКе

Небольшая программа на БЕЙСИКе БК-0010.01 превращает компьютер в цветомузыкальную установку. После запуска командой RUN нужно поставить в магнитофон кассету с хорошей музыкой и включить воспроизведение. (Конечно, картинка на экране простовата по сравнению, например, с получаемой в программе ЦМУ2, но, исследовав алгоритм работы, читатели смогут написать свои, лучшие программы.)

```
10 CLS
20 T=3
30 A$=MID$(BIN$(PEEK(&O177716)),11,1)
40 IF A$="1" THEN 70 ELSE IF T>-60 THEN T=T-1
50 CIRCLE (122,122),T,4
60 GOTO 30
70 CIRCLE (122,122),T,INT(RND(1)*3)+1
80 IF T<10 THEN T=T+2 ELSE T=T-2
90 GOTO 30
```

(Публикуется по материалам журнала «Информатика и образование». — *Прим. ред.*)

Простейший звуковой сигнал в БЕЙСИКе

При ошибке или по окончании работы программы, чтобы обратить на это внимание пользователя, кроме BEEP часто используется еще один простейший звуковой сигнал — серия щелчков, обычно выдаваемых при нажатии на клавиши. Это делается с помощью выполняемого в цикле оператора **PRINT CHR$(7),** что требует как минимум три программных строки. То же самое можно сделать и одним оператором **PRINT STRING$(N,7)** , где N — количество «щелчков».

(Публикуется по материалам журнала «Вычислительная техника и ее применение». — *Прим. ред.*)

Звук в БЕЙСИКе? Проще простого!

Эта программа демонстрирует наиболее простой способ реализации оператора SOUND (звуки различной частоты и длительности) средствами БЕЙСИКа без использования машинных кодов. Переменная **S** — условное значение частоты (рекомендуется от 1 до 10). Хотя эта программа и не позволяет обеспечить всю полноту звуковой гаммы, доступную USR-подпрограммам, для получения нескольких различных «писков» ее возможностей вполне достаточно.

```
10 CLS
20 T=40   ' длительность "писка"
30 INPUT "Частота:";S
40 FOR I=1 TO T
50 OUT &O177716,64%,1%
60 S0=S
70 S0=S0-1
80 IF S0>0 GOTO 70
90 OUT &O177716,64%,0%
100 S0=S
110 S0=S0-1
120 IF S0>0 GOTO 110
130 NEXT
140 GOTO 30
```

О. Ю. Семенов

Драйвер бесшумной клавиатуры

При работе в среде вильнюсского БЕЙСИКа БК-0010.01 иногда бывает необходимо временно отключить подзвучку (щелчок) при нажатии клавиш. В таком случае окажется полезной небольшая кодовая программа — драйвер.

НАЧАЛЬНЫЙ АДРЕС: 000400 ДЛИНА: 000140

```
000437 012737 124300 002100 000207 012767 101150 000024
000403 012767 101374 000014 004437 110340 010546 004767
000004 000137 000000 005737 000224 001402 004737 110536
012737 177777 000222 012700 005670 077001 000137 102112
012767 104004 177672 012737 000412 000060 012737 000422
000274 012737 000400 002100 000137 120200 000000 000000
```

КОНТРОЛЬНАЯ СУММА: 015176

Коды набираются в режиме ТС, в отладчике или непосредственно в БЕЙСИКе с помощью операторов POKE. Можно также записать их в составе операторов DATA и загрузить в ОЗУ в цикле:

```
10 DATA . . .         ' список кодов
20 FOR I%=256 TO 346 ST 2%
30 READ A%
40 POKE I%,A%
50 NEXT I%
60 END
```

После этого сформированная в стековой области кодовая программа записывается на магнитную ленту командой BSAVE "SNDOFF",256,348.

Для загрузки драйвера необходимо набрать BLOAD "SNDOFF",R (обязательно с R!), после успешного чтения звук при нажатии клавиш будет отключен. Включить его вновь можно командой A=USR(A) (USR-функция определяется автоматически, оператор DEF USR не требуется).

(Публикуется по материалам журнала «Вычислительная техника и ее применение». — *Прим. ред.*)

♦ ♦

Синтезатер речи для БК-0010(.01)

Предлагаемая ниже программа в машинных кодах позволяет загрузить в память компьютера небольшой речевой или музыкальный фрагмент (с кассеты или с микрофона, подключенного к магнитофону в режиме записи с нажатой кнопкой паузы) и воспроизводить его любое число раз. (После некоторой доработки можно получить и возможность сохранения автономной воспроизводящей речь подпрограммы на магнитной ленте или диске.) Из всех вариантов, опубликованных в различной литературе по БК, данная программа обеспечивает наилучшее качество воспроизведения речи и содержит удобный интерфейс пользователя.

Программа неперемещаема, вводится в БК и запускается с адреса 1000, а после окончания работы происходит выход в монитор.

НАЧАЛЬНЫЙ АДРЕС: 001000 ДЛИНА: 001320

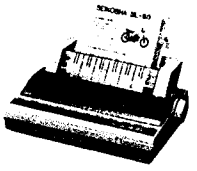
```
104014 104034 032700 000001 001003 012700 000233 104016
012700 000232 104016 012700 000001 012701 000005 010102
104030 012701 000372 104032 012702 000353 104032 012701
000005 104032 010102 104032 104024 012701 002044 012702
```

```
000021 104020 012701 000012 012702 000007 104024 012701
002066 012702 000014 104020 012701 000003 012702 000012
104024 012701 002102 012702 000020 104020 012701 000003
012702 000013 104024 012701 002122 012702 000021 104020
012701 000003 012702 000014 104024 012701 002144 012702
000023 104020 012701 000005 012702 000020 104024 012701
002170 012702 000020 104020 104004 104006 020027 000060
001532 020027 000061 001407 020027 000062 001526 012700
000030 104016 000762 104016 012701 000003 012702 000022
104024 012701 002210 012702 000025 104020 012701 000007
012702 000024 104024 012701 002236 012702 000014 104020
104006 012704 000003 012701 002306 012702 000001 062701
000002 005011 032737 000040 177716 001402 050211 000402
000240 000240 010403 077301 000241 006302 103363 020127
037776 001353 012701 000003 012702 000022 104024 012701
000025 012700 000040 001402 104016 077103 012701 000007
012702 000024 104024 012701 000014 012700 000040 001402
104016 077103 012701 000025 012702 000020 104024 012700
000030 104016 000401 000522 000640 104016 012701 000003
012702 000022 104024 012701 002210 012702 000025 104020
012701 000007 012702 000024 104024 012701 002252 012702
000014 104020 104006 012704 000003 012701 002306 012702
000001 062701 000002 000240 030211 001404 052737 000100
177716 000403 042737 000100 177716 010403 077301 000241
006302 103362 020127 037776 001352 012701 000003 012702
000022 104024 012701 000025 012700 000040 001402 104016
077103 012701 000007 012702 000024 104024 012701 000014
012700 000040 001402 104016 077103 012701 000025 012702
000020 104024 012700 000030 104016 000656 104016 012701
000003 012702 000022 104024 012701 002266 012702 000020
104020 000000 171452 167351 162764 160772 167764 020362
162762 164776 000052 116223 166440 167345 020340 020072
111234 026460 167753 162756 020343 160762 167742 174764
035440 026461 160772 164760 174363 172040 165745 172363
020341 000073 026462 167767 170363 167762 175351 162767
162744 164756 020345 000056 173621 162767 164744 162764
165440 160354 020376 020072 160756 166766 172351 020345
160354 172742 020340 166353 173741 175751 000365 020351
167747 167767 164762 162764 027040 020351 166363 175765
165341 162764 027040 167744 171440 164767 160744 164756
020361 115441 115040 000000 000000 000000 000000 000000

КОНТРОЛЬНАЯ СУММА: 034056
```

(Публикуется по материалам журнала «Вычислительная техника и ее применение», №2 за 1990 г. — *Прим. ред.*)

HARD & SOFT

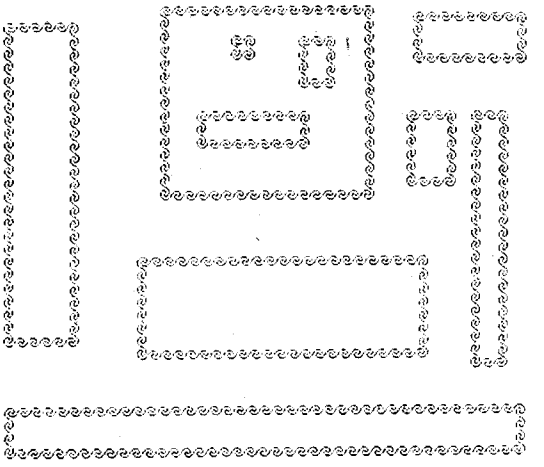


Принтер исправно служит пользователям компьютеров в повседневной работе для печати листингов программ, их описаний и т.д. При этом принтер выполняет обязанности автоматической пишущей машинки, от которой только и требуется, что быстро вывести текст на бумагу.
Но когда мы хотим напечатать поздравление к празднику или рекламный листок, естественно, возникает желание как-то украсить текст, придать ему привлекательный вид. Здесь нам на помощь приходят графические возможности печати. Если вы располагаете современным принтером с системой команд EPSON, а таковыми являются, например, весьма распространенные в последнее время модели отечественного производства МС-6312М и МС-6313М, вам, несомненно, будет полезна предлагаемая ниже программа.

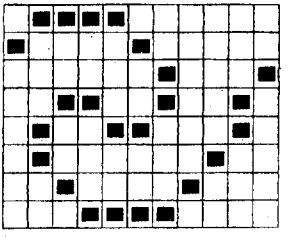
С. К. Румянцев,
пос. Локня, Псковская обл.

ПРИНТЕР – ХУДОЖНИК

С помощью описываемой программы можно дополнить текст красивой рамкой в «греческом» стиле. Можно печатать рамки любых размеров на произвольном расстоянии от края листа, в зависимости от расположения текста. Для этого надо лишь изменить некоторые входные параметры.



Узор рамок, печатаемых программой, довольно распространен в современном дизайне (например, он украшает станцию метро «Пушкинская» в Санкт-Петербурге). В цифровом виде этот орнамент представляет собой повторяющиеся блоки 8×11 пикселов. С одной стороны, это удобно, так как графическая печать производится столбцами высотой именно по восемь точек, так что горизонтальные фрагменты рамки можно напечатать, просто повторив исходный блок нужное число раз. При выводе же вертикальных сторон блок нужно повернуть на 90°, и на 8 игл печатающей головки будут приходиться уже 11 точек блока. В программе предусмотрены специальные средства, позволяющие обойти эту сложность и делать рамки любых размеров.



Читателям, программирующим на ассемблере, видимо, будет интересно узнать, как это делается. В этом им поможет листинг, снабженный подробными комментариями. В нем приводятся также машинные коды, которые в случае отсутствия ассемблер-транслятора, можно набрать в отладчике или режиме ТС. Программа перемещаема, так что загружать ее можно не только с адреса 1000, но и с любого другого.

Первые команды программы засылают в регистры исходные параметры: ширину и высоту рамки, а также отступ от края листа. Их числовые значения можно изменять по своему усмотрению, необходимо только учитывать, что значение ширины рамки должно не превышать #63 при нулевом отступе, иначе она не поместится на стандартном листе. Высота может быть любой при использовании рулонной бумаги, но на стандартном листе умещается не более #113 «волн» рамки. Кроме того, ее ширина и высота не должны быть равны нулю, иначе процесс печати может затянуться до бесконечности.

Некоторые модели принтеров имеют инверсный вход. На этот случай в программе предусмотрен резерв — две команды NOP, на месте одной из которых можно поместить команду инверсии данных COMB R0 (код 105100).

Напечатайте несколько рамок, и вы увидите, как расширятся рамки ваших возможностей в применении компьютера!

Листинг программы печати рамок

```
Программа печати на принтере декоративной рамки
Автор:  Румянцев С.К.        п.Локня        1992

001000   012701   MOV  #51, R1       ;Ширина рамки
001002   000051
001004   012702   MOV  #103, R2      ;Высота рамки
001006   000103
001010   012703   MOV  #0, R3        ;Отступ от края листа
001012   000000
001014   110367   MOVB R3, 1503      ;Отступ - в область данных
001016   000463
001020   010167   MOV  R1, 1334      ;Ширина - в область данных
001022   000310
001024   010100   MOV  R1, R0        ;Подготовка аргумента
001026   012701   MOV  #12, R1       ;графической команды
001030   000012                     ;принтера
001032   062701   ADD  #13, R1
001034   000013
001036   077003   SOB   R0, 1032
001040   010167   MOV  R1, 1514      ;Аргумент - в область
001042   000450                     ;данных
001044   162701   SUB  #20, R1
001046   000020
001050   010167   MOV R1,  1206      ;Количество нулей
001052   000132                     ;в середине рамки
001054   010200   MOV  R2, R0
```

```
001056  012702  MOV  #177772, R2   ;Вычисление количества
001060  177772                    ;точек по вертикали
001062  062702  ADD  #13, R2
001064  000013
001066  077003  SOB  R0, 1062
001070  010703  MOV  PC, R3        ;Адрес
001072  062703  ADD  #403, R3      ;начала данных
001074  000403
001076  004767  JSR  PC, 1406      ;Настройка принтера
001100  000304
001102  004767  JSR  PC, 1406      ;Графическая команда
001104  000300                    ;первой строки
001106  004767  JSR  PC, 1326      ;Печать данных
001110  000214                    ;первой строки
001112  012704  MOV  #10, R4       ;Данные
001114  000010                    ;средних строк -
001116  012363  MOV  (R3)+, 16(R3) ;в динамический
001120  000016                    ;буфер
001122  077403  SOB  R4, 1116
001124  010305  MOV  R3, R5        ;Сохранить текущий адрес
001126  010503  MOV  R5, R3        ;Вернуть в цикле
001130  162703  SUB  #61, R3       ;Вернуть адрес
001132  000061                    ;графической команды
001134  004767  JSR  PC, 1406      ;и подать ее на принтер
001136  000246
001140  005001  CLR  R1            ;Маска
001142  162702  SUB  #10, R2       ;нужна ?
001144  000010
001146  100006  BPL  1164          ;Нет
001150  010204  MOV  R2, R4        ;Формирование маски
001152  000261  SEC                ;остатка узора
001154  006101  ROL  R1            ;для предпоследней
001156  005202  INC  R2            ;строки
001160  100774  BMI  1152
001162  010402  MOV  R4, R2        ;Маска готова
001164  010503  MOV  R5, R3        ;Вернуть адрес буфера
001166  012704  MOV  #10, R4       ;8 столбцов
001170  000010                    ;из буфера
001172  012300  MOV  (R3)+, R0     ;средней строки
001174  040100  BIC  R1, R0        ;очистить по маске
001176  004767  JSR  PC,  1422     ;и подать на принтер,
001200  000220                    ;формируя начало
001202  077405  SOB  R4, 1172      ;средней строки
```

```
001204   012704   MOV #675, R4
001206   000675
001210   005000   CLR  R0          ;Подать на принтер
001212   004767   JSR  PC, 1422    ;нули в промежуточной
001214   000675                   ;части средней строки
001216   077404   SOB  R4, 1210
001220   004767   JSR  PC, 1356    ;Модифицировать буфер
001222   000132                   ;для печати окончания
001224   004767   JSR  PC, 1356    ;средней строки
001226   000126
001230   010503   MOV R5, R3       ;Вернуть адрес буфера
001232   012704   MOV  #10, R4     ;8 столбцов
001234   000010                   ;из буфера
001236   012300   MOV  (R3)+, R0   ;средней строки
001240   040100   BIC  R1, R0      ;очистить по маске
001242   004767   JSR  PC, 1422    ;и подать на принтер,
001244   000154                   ;формируя конец
001246   077405   SOB  R4, 1236    ;средней строки
001250   004767   JSR  PC, 1356    ;Модифицировать буфер
001252   000102                   ;для следующей строки
001254   005702   TST  R2          ;Вертикаль исчерпана ?
001256   003323   BGT  1126        ;Нет - на продолжение
001260   005402   NEG  R2          ;Формирование
001262   010205   MOV  R2, R5      ;величины отката
001264   006302   ASL  R2          ;для печати остатка
001266   060502   ADD  R5, R2      ;предпоследней строки
001270   110263   MOVB R2, 3(R3)   ;Откат - в данные
001272   000003
001274   004767   JSR  PC, 1406    ;Выполнить откат
001276   000106
001300   010305   MOV  R3, R5      ;Сохранить текущий адрес
001302   162703   SUB  #105, R3    ;Вернуться к адресу
001304   000105                   ;графической команды
001306   004767   JSR  PC, 1406    ;и подать ее на принтер
001310   000074
001312   010503   MOV  R5, R3      ;Вернуть текущий адрес
001314   004767   JSR  PC, 1326    ;Печать последней строки
001316   000006
001320   004767   JSR  PC, 1406    ;Нормализация принтера
001322   000062
001324   000000   HALT             ;Останов программы
; Подпрограмма печати первой и последней строк
001326   004767   JSR  PC, 1406    ;Печать начальных
```

```
001330   000054                        ;данных строки
001332   012704   MOV  #51, R4         ;Количество повторов
001334   000051                        ;среднего блока
001336   010305   MOV  R3, R5          ;Сохранить адрес
001340   010503   MOV  R5, R3          ;Вернуть в цикле
001342   004767   JSR  PC, 1406        ;Напечатать средний
001344   000040                        ;блок требуемое
001346   077404   SOB  R4, 1340        ;число раз
001350   004767   JSR  PC, 1406        ;Печать конечных
001352   000032                        ;данных строки
001354   000207   RTS PC

            ; Подпрограмма модификации буфера средних строк

001356   010503   MOV  R5, R3          ;Вернуть текущий адрес
001360   012704   MOV  #10, R4         ;И в количестве
001362   000010                        ;8 штук,
001364   032713   BIT  #40, (R3)       ;со смещением
001366   000040                        ;в 11-8=5 бит,
001370   001402   BEQ  1376            ;учитывая
001372   000261   SEC                  ;установленные
001374   000401   BR   1400            ;и
001376   000241   CLC                  ;сброшенные биты
001400   006023   ROR  (R3)+           ;циклически сдвинуть
001402   077410   SOB  R4, 1364        ;данные средних строк
001404   000207   RTS PC

; Подпрограмма подачи цепочки кодов на принтер

001406   112301   MOVB (R3)+, R1       ;Количество кодов
001410   112300   MOVB (R3)+, R0       ;Текущий код
001412   004767   JSR  PC, 1422        ;подать
001414   000004                        ;на принтер
001416   077104   SOB  R1, 1410        ;требуемое число раз
001420   000207   RTS PC

; Подпрограмма подачи кода из R0 на принтер

001422   010146   MOV R1, -(SP)        ;Сохранить регистр
001424   012701   MOV #177714, R1      ;Адрес порта
001426   177714
001430   000240   NOP                  ;Резерв
001432   000240   NOP
```

```
001434   042700   BIC #177400, R0    ;Выделить младший байт
001436   177400
001440   032711   BIT  #400, (R1)    ;Принтер готов ?
001442   000400
001444   001775   BEQ  1440          ;Нет - ожидание
001446   052700   BIS  #400, R0      ;Установить строб
001450   000400
001452   010011   MOV  R0, (R1)      ;Код - на принтер
001454   032711   BIT  #400, (R1)    ;Код принят ?
001456   000400
001460   001375   BNE  1454          ;Нет - ожидание
001462   042700   BIC  #400, R0      ;Сбросить строб
001464   000400
001466   010011   MOV  R0, (R1)      ;Код - на принтер
001470   012601   MOV  (SP)+, R1     ;Вернуть регистр
001472   000207   RTS PC

; Данные, используемые программой

001474   003400   007.000   ;Интервал между строками
001476   031433   063.033   ;24/216 дюйма
001500   015430   033.030   ;Установка позиции
001502   000104   000.104   ;горизонтальной табуляции
001504   004000   010.000
001506   006412   015.012   ;Перевод строки,
001510   015411   033.011   ;табуляция
001512   002452   005.052   ;и команда
001514   000715   001.315   ;графической печати
001516   017002   036.002   ;Начало первой строки
001520   005441   013.041   ;Середина первой строки
001522   106100   214.100
001524   110622   221.222
001526   044611   111.211
001530   001061   002.061
001532   014004   030.004
001534   004040   010.040   ;Окончание первой строки
001536   104107   210.107
001540   111620   223.220
001542   040214   100.214
001544   017041   036.041
001546   017003   036.003   ;Данные средних строк
001550   020404   041.004
001552   040210   100.210
```

```
001554  046111  114.111
001556  031106  062.106
001560  001040  002.040
001562  102020  204.020
001564  074017  170.017
001566  017003  036.003  ;Динамический буфер
001570  020404  041.004  ;средних строк
001572  040210  100.210
001574  046111  114.111
001576  031106  062.106
001600  001040  002.040
001602  102020  204.020
001604  074017  170.017
001606  015403  033.003  ;Откат бумаги назад
001610  007552  017.152
001612  074010  170.010  ;Начало последней строки
001614  001204  002.204
001616  144461  311.061
001620  010411  021.011
001622  005742  013.342  ;Середина последней строки
001624  014004  030.004
001626  040040  100.040
001630  111214  222.214
001632  104621  211.221
001634  030511  061.111
001636  001002  002.002  ;Окончание последней строки
001640  074204  170.204
001642  015404  033.004  ;Установка стандартного
001644  005100  012.100  ;режима принтера, перевод
001646  000015  000.015  ;строки и возврат каретки
```

Работа программы редакцией не проверялась.

Уважаемые читатели!

Мы обращаемся ко всем, кто так или иначе связан с БК: к изготовителям самого компьютера и периферийных устройств для него, к кооперативам и клубам пользователей, ко всем профессиональным разработчикам программного обеспечения или аппаратных устройств и к программистам-любителям, к нашим авторам, наконец, просто к читателям журнала. Мы с большим вниманием относимся к любому вашему письму, статье, телефонному звонку — даже к сердитым, критическим, пустым и невыполнимым по запросам.

Однако, приглашая вас к сотрудничеству, следует поговорить и о том, как осуществить это сотрудничество наиболее рациональным путем, чтобы снизить затраты сил и времени как на подготовку материалов, так и на их публикацию и одновременно уменьшить вероятность появления ошибок.

Тем, кто пишет нам впервые. Статью нужно начинать с объяснения, для кого и для чего она написана, на каком конкретно компьютере вы работаете. При описании программ нужно указать, чем они отличаются от ранее опубликованных (если таковые были), описать их особенности. Не бойтесь перегрузить материал подробностями и формулами. Важно, чтобы все было понятно даже новичку, впервые увидевшему БК. Лишнее редакция уберет.

Всех, обращающихся в редакцию, просим указывать в письме полностью и разборчиво: имя, фамилию и отчество, адрес (с индексом), паспортные данные, телефон (служебный и домашний), год рождения. Просим также предоставлять нам (письменно) право помещать в конце вашей статьи сведения об авторе (адрес, телефон и т. п.) или, наоборот, предупреждать, что этого делать не следует.

Что касается содержания присылаемых материалов, прежде всего для нас (и для читателей!) интересны статьи, содержащие работоспособные листинги программ и (или) принципиальные схемы аппаратных разработок. Не менее важны материалы по методике алгоритмизации и программирования и прочие «теоретические изыски, опирающиеся на практику»: справочные данные, исследования схемотехники БК и содержимого его ПЗУ, опыт работы с БК и периферией и др.

Лучше отпечатать текст статьи на принтере через два интервала. Иллюстрации и таблицы желательно выполнять каждую на отдельном листе. То же относится и к листингам программ. Отправляемые материалы (2 экз. — на бумаге) должны быть выверены (программы — отлажены). Неоконченные разработки (в том числе на уровне идей) редакцией рассматриваются, условия их предоставления необходимо заранее обсудить по телефону. Необходимо оставить у себя копию.

Из программ в первую очередь рассматриваются распечатанные на принтере. Все, что вручную переписано с экрана, как правило,

содержит ошибки, а следовательно, может потребовать серьезной доработки, что, в свою очередь, еще больше увеличивает время подготовки статьи к печати.

Известно, что при перенаборе ошибки неизбежны. Поэтому **лучше всего предоставлять редакции файлы на магнитных носителях** (носители возвращаются автору). Тексты статей могут быть записаны на дискетах IBM 3,5' и 5,25' (Лексикон, Word), на дискетах БК (операционные системы ANDOS, NORD или Нортон) или на кассетах БК (в двух последних случаях — текстовые редакторы TED, VorteX! или EDASP). Если вы не имеете возможности преобразовывать листинги на БЕЙСИКе и ФОКАЛе в текст обычного вида, а кодовые программы в текстовые таблицы машинных кодов, лучше всего передать на кассете «живые» программы (для БЕЙСИКа — предпочтительно в формате .ASC). При всем этом на дискете желательно иметь резервную копию (в специально созданном для этого подкаталоге или под другими именами), на кассете же наличие не менее двух копий обязательно.

Аналогичным способом могут быть подготовлены иллюстрации к статье. Формат представления — файл, содержащий копию экранного ОЗУ (начиная с адреса 40000 или 42000, т. е. с учетом служебной строки или без нее), причем рулонное смещение должно быть нормализовано (адрес 40000 должен соответствовать верхнему левому углу изображения). При выполнении рисунков в графических редакторах серии GRAF или БК-PAINT редакции могут быть переданы файлы, записанные в формате этих программ. И наконец, при подготовке иллюстраций на IBM-совместимых компьютерах нужно применять формат .PCX (черно-белый вариант). Для этой цели проще всего использовать резидентную программу типа PZP, позволяющую сохранять в файле (либо распечатывать на принтере) копию экрана IBM. (В меню PZP нужно выбрать следующие установки: режим качества «B&W Standart», без инверсии, окно выделения во весь экран, размеры для печати соответствуют листу A4, формат при записи в файл — «Ventura».) В любом случае, тем более если файлы иллюстраций подготовлены в ином формате (например, в векторном, принятом в графических системах AUTOCAD или PCAD), необходимо присылать в редакцию их распечатки на принтере или графопостроителе (изображение должно быть ярким и четким). Если возможностей для подготовки рисунков на компьютере нет, годятся рисунки и чертежи, аккуратно выполненные вручную тушью, тонким фломастером или шариковой ручкой (желательно черного цвета)

Адрес для переписки: 125315, Москва, а/я 17

Телефон: (095) 151-19-40

Факс: (095) 208-67-37

E-Mail: mail@infoobr.msk.su

ЗАЯВКА

на журнал «Персональный компьютер БК-0010 — БК-0011М»

_____

(адрес подписчика с указанием почтового индекса)

_____

(фамилия, имя, отчество полностью)

_____

(номер выпуска и год издания)

_____

(общее количество экземпляров)

Перечислено на расчетный счет_____

_____руб.

(общее количество экземпляров, стоимость одного экземпляра)

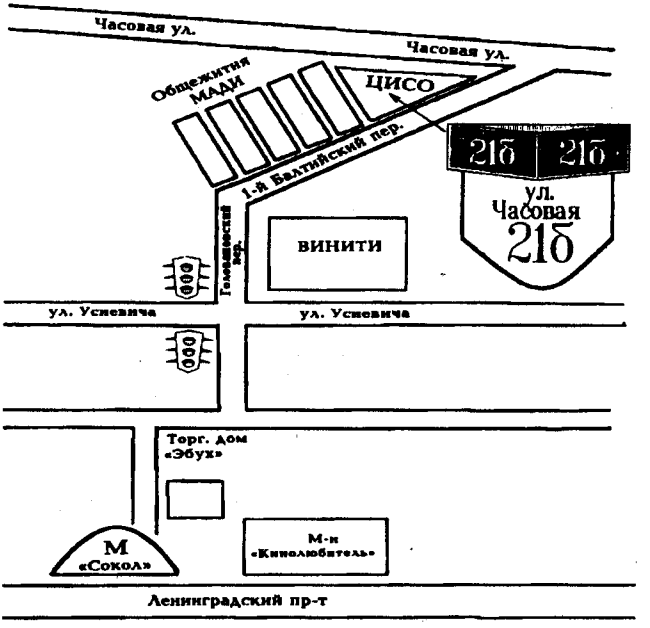
Платежное поручение № _____ от _____ 199 ____ г.

СОДЕРЖАНИЕ

ПЕРСОНАЛЬНЫЙ КОМПЬЮТЕР БК-0010 – БК-0011М

Наш адрес: Москва, ул. Часовая, 21Б, помещение Центра Интерактивных Средств Обучения (ЦИСО), комн.20
Телефон: (095) 151-19-40

Как к нам добраться:



Адрес для переписки: 125315, Москва, а/я 17.

ПЕРСОНАЛЬНЫЙ КОМПЬЮТЕР БК-0010 - БК-0011М

Главный редактор
Васильев Б. М.

Редактор
Усенков Д. Ю.

Корректор
Антонова В. С.

Компьютерная верстка
Макеев А. Б.

Подписано в печать с оригинал-макета издательства «Информатика и образование» 23.01.95. Формат 70×100 1/16. Бумага офсетная. Усл.печ.л. 6,5. Тираж 1500 экз. Заказ № 93. . Цена 4000 руб. (по подписке). В розничной продаже цена договорная.
Ордена Трудового Красного Знамени Чеховский полиграфический комбинат Комитета Российской Федерации по печати.
142300, Чехов Московской обл.

Планируемое содержание выпуска 2'95:

- **Архитектура и программирование на языке ассемблера** *(продолжение)*
- **Дисководы MC5305 и MC5311: техническое описание**
- **Принтер D100**
- **Многоликий БЕЙСИК**

"Шитые коды" и их использование

Добавление в БЕЙСИК новых операторов

Работа с программами увеличенного объема

Передача данных в USR-подпрограммы: новое решение

Многомерные массивы в вильнюсском БЕЙСИКе

READ-DATA: новые возможности

- **БК плюс магнитофон: продолжение вечной темы**
- **"Есть идея": буферизация ввода-вывода в ANDOS**
- **"Потехе час": экстрасенсы, проверьте свои возможности**