С. Викерс

Программирование на языке Бейсик

С. Викерс

ZX SPECTRUM ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ БЕЙСИК

Под ред. Р.Бредбера

Печатается по изданию:

THE LEAGRAWE PRESS LTD., LUTTON AND LONDON, GREATE BRITAIN, THIRD EDITION. 1983

Подготовлено к печати:

МИГ ЛТД., РОСТОВ-НА-ДОНУ, 1991

ВНИМАНИЮ ЧИТАТЕЛЕЙ!

МИГ Лтд. ПРЕДЛАГАЕТ:

- Наборы микросхем для ZX SPECTRUM-совместимых компьютеров (ленинградского и краснодарского вариантов, вариантов "Спарк", "Пентагон" и других)
- Смонтированные и настроенные платы компьютера ZX SPECTRUM ленинградского варианта
- Корпуса из дюралюминия и клавиатуры на основе кнопок ВМ-16 (40 кнопок)
- Импульсный блок питания на плате 70х140мм со следующими параметрами: $U_{вх}$ -220В; $U_{вых}$ -5В (при $I_{нагр}$ -1А); регулировка $U_{вых}$ - 0.2В; пульсации $U_{вых}$ не более 50мВ; предусмотрена защита от КЗ и от перенапряжения по выходу
- Настроенные компьютеры ZX SPECTRUM ленинградского варианта в дюралюминиевом корпусе со встроенным блоком питания
- Программатор к компьютеру ZX SPECTRUM, работающий в нескольких режимах (перенос информации с эталонной микросхемы, запись информации от компьютера, ввод информации с магнитной ленты в компьютер) и характеризующийся следующими параметрами: программируемые микросхемы - 573РФ2, РФ4, РФ5, РФ6, РФ7, РФ8 и их аналоги (в т.ч. зарубежные); предусмотрена возможность регулировки $U_{прог}$; питание - от автономного блока; подключение к компьютеру - кабелем
- Разнообразные системные и игровые программы (возможна запись на кассеты заказчика)
- Декодеры ПАЛ-СЕКАМ на основе микросхемы TDA-4510 фирмы PHILIPS для телевизоров 2-го, 3-го и 4-го поколений
- Декодеры ПАЛ-СЕКАМ на основе отечественных микросхем ХА28 (КХА039) для телевизоров 2-го, 3-го и 4-го поколений
- Специально разработанный декодер ПАЛ-СЕКАМ для ламповых моделей телевизоров (марок 714-736)
- "НЧ вход-выход", позволяющий при подключении к телевизору тюнера или видеомагнитофона значительно улучшить качество изображения

МИГ Лтд. обеспечит высокое качество и надежность поставляемой продукции при выгодных для вас ценах

Наш адрес: 344022, Ростов-на-Дону, а/я 3678

Телефон: (863-2) 24-80-70

Глава 1

ВВЕДЕНИЕ

Если вы читаете эту книгу впервые или открыли ее на этом листе, то вы должны иметь представление о том, что команды Бейсика выполняются непосредственно, операторы начинаются с номера строки и сохраняются в памяти компьютера. Вы должны также представлять себе, что такие команды как PRINT, LET и INPUT используются во всех компьютерах, имеющих Бейсик, а такие команды как BORDER, PAPER и BEEP используются только в ZX SPECTRUM.

Изучение Бейсика начнем с повторения некоторых моментов, изложенных во вводной части, причем рассмотрим их достаточно подробно, уяснив, что можно делать, а чего нельзя.

Что бы вы не читали, старайтесь при этом использовать компьютер. Если у вас возник вопрос:"Что будет, если я сделаю так и так?", то ответ для вас очень прост: введите эти фразы в компьютер и вы увидите сами!

Всякий раз,когда в этой книге вы встретите предложение что-нибудь ввести в компьютер и выполнить на нем, спрашивайте сами себя:"Что я могу сделать вместо этого?" и попробуйте это проделать. Чем больше собственных программ вы напишите, тем лучше вы будете понимать как работает компьютер.

В конце этой книги имеется несколько приложений. Они содержат сведения по организации памяти, по операциям с числами, а также несколько примеров программ, иллюстрирующих возможности ZX SPECTRUM.

КЛАВИАТУРА

В ZX SPECTRUM клавиши содержат не только одиночные символы (буквы,цифры и т.п.), но также составные символы (ключевые слова, названия функций и т.п.) и все то, что вводится с клавиатуры не посимвольно. Для того, чтобы реализовать все эти функции и команды, некоторые клавиши клавиатуры имеют пять и более значений, получаемых либо путем выбора соответствующего регистра (т.е. путем нажатия клавиш CAPS SHIFT или SYMBOL SHIFT одновременно с необходимой клавишей), либо путем перевода компьютера в один из возможных режимов работы.

Состояние индицируется курсором - мерцающей буквой, которая показывает на экране место, где будет появляться следующий набираемый символ.

Режим (K) автоматически заменяет режим (L), когда компьютер ожидает команду или программную строку (отличающуюся от вводимых данных), и с этой позиции в строке курсором указывается, что ожидается ввод ключевого слова или строки. Это относится к началу строки или знакоместу сразу же после оператора THEN, или же к знакоместу сразу же после ":" (за исключением двоеточия в строке). Если режим не изменен, то нажатие следующей клавиши будет интерпретироваться как ключевое слово, написанное на клавише, либо как цифра.

Режим курсора (L) (для букв) появляется обычно во всех других случаях. Если он не меняется, то нажатие последующей клавиши будет интерпретироваться как основной символ на клавише. В большинстве случаев - это буквы.

В режимах (K) и (L) одновременное нажатие клавиши SYMBOL SHIFT и какой-либо клавиши воспринимается как вспомогательный символ, изображенный на этой клавише, а нажатие CAPS SHIFT с цифровой клавишей - как управляющая функция, написанная на цифровой клавише.

Нажатие клавиши CAPS SHIFT с другими клавишами в режиме курсора (K) не влияет на ключевые слова, а в режиме курсора (L) вызывает появление заглавных букв.

Режим курсора (C) (для заглавных букв) - это вариант режима (L), в котором все буквы на экране появляются как заглавные.

Нажатие клавиши CAPS LOCK приводит к смене режима курсора (L) на режим (C) или наоборот.

Режим курсора (E) (расширение) используется для получения дополнительных символов (обычно знаков). Курсор (E) появляется после одновременного нажатия обеих клавиш смены режима и сохраняется до нажатия какой-либо одной из них. В этом режиме нажатие клавиши приводит к появлению на экране графических символов и знаков, если режим сохраняется, и других, если одновременно нажата одна из клавиш смены режима.

Одновременное нажатие цифровых клавиш с клавишей смены режима SYMBOL SHIFT вызывает появление знака, в противном случае они дают появление символов, управляющих цветом.

Режим курсора (G) возникает после нажатия клавиши GRAPHICS (CAPS SHIFT и 9) и сохраняется до тех пор, пока не будет нажата клавиша CAPS SHIFT - одна или вместе с клавишей 9.

Цифровые клавиши дают также мозаичные символы, за исключением GRAPHICS или DELETE. Каждая из буквенных клавиш, кроме V, W, X, Y и Z, могут вызывать появление определенных пользователем графических символов.

Если некоторая клавиша удерживается в нажатом состоянии более, чем 2 или 3 секунды, это вызывает повторение ее действия. Ввод с клавиатуры осуществляется в нижнюю половину экрана. Каждый символ (или группа символов для ключевых слов) появляется перед курсором. Сам курсор может перемещаться по экрану клавишами:

влево - CAPS SHIFT и 5;

вправо - CAPS SHIFT и 8 и т.д.

Символ перед курсором может быть удален командой DELETE (CAPS SHIFT и 0).

Примечание. Целая строка может быть удалена вводом EDIT (CAPS SHIFT и 1) и последующим нажатием клавиши ENTER.

При нажатии ENTER строка, набранная в нижней части экрана либо выполняется как команда, либо вводится как очередная строка в программу, либо используется как список данных для INPUT-ввода. Если она содержит синтаксические ошибки, то ошибочное место указывается мерцающим знаком вопроса (?).

Когда вводятся строки программы, то листинг отображается в верхней половине экрана. Последняя введенная строка называется текущей и указывается символом (>). Его можно перемещать вниз и вверх, используя клавиши CAPS SHIFT и 6 или CAPS SHIFT и 7 соответственно. Если нажата клавиша EDIT (CAPS SHIFT и 1), то текущая строка переносится в нижнюю часть экрана, где она может редактироваться.

При выполнении команды или программы ввод осуществляется в верхнюю часть экрана и сохраняется до ввода строки программы либо нажатия клавиши ENTER при наличии пустой строки, либо до нажатия

клавиши перемещения вверх, вниз.

В нижнюю часть экрана выводятся также сообщения об ошибках, которые сохраняются там до нажатия любой клавиши (это идицируется переходом в режим (К)).

В определенных ситуациях клавиши CAPS SHIFT и SPACE действуют как BREAK, останавливая компьютер с выдачей сообщения "D" или "L". Это распознается:

а) в конце исполняемого оператора программы;

б) после завершения операции на принтере или магнитофоне.

ЭКРАН ТЕЛЕВИЗОРА

Экран содержит 24 строки по 32 символа в каждой и делится на две части. Верхняя часть экрана (22 строки) служит для отображения листинга программы. Когда верхняя часть экрана заполняется полностью, экран сворачивается на одну строку. Компьютер останавливается с выдачей сообщения: "SCROLL?". Ответ N, SPACE или STOP вызывает остановку с выдачей сообщения "D BREAK-CONT REPEATS". Нажатие любой другой клавиши разрешает свертку.

Нижняя часть экрана используется для ввода команд, строк программы и данных, а также вывода сообщений системы.

Глава 2

ОСНОВЫ ПРОГРАММИРОВАНИЯ НА ЯЗЫКЕ БЕЙСИК

Краткое содержание: программы, номера строк, редактирование программ с использованием клавиш <вверх>, <вниз>, EDIT;
командыRUN,LIST, GO TO; CONTINUE, INPUT, NEW, REM, PRINT, STOP в INPUT-данных, BREAK.

Наберите эти две строки программы вычисления суммы двух чисел:

```
10 LET A=10
20 PRINT A
```

так чтобы на экране появилось

```
10  >  LET A=10
20     PRINT A




K
```

Строка программы должна начинаться с номера, который не записывается в память, а служит лишь для указания порядка следования строк в программе, что важно при ее выполнении.

Теперь наберите:

```
15 LET B=15
```

и введите. Сделать подобное было бы невозможно, если бы нумерация начиналась с 1 и 2, а не с 10 и 20, как в нашем случае. (Номер может лежать в интервале от 1 до 9999).

Допустим, теперь вам понадобилось изменить строку 20 на следующую:

```
20 PRINT A+B.
```

Это можно сделать, используя команду EDIT.

Символ (>) в строке 15 называется программным курсором, а строка на которую он указывает, называется текущей. Это обычно последняя введенная строка, но вы имеете возможность переместить программный курсор выше или ниже, используя соответствующие клавиши управления курсором. Установите его в строку 20. Когда вы нажмете клавишу EDIT, в нижней части экрана появится копия текущей строки; в нашем случае - копия строки 20. Нажмите и держите клавишу перемещения курсора вправо до тех пор, пока курсор (L) не переместится на конец оператора , и затем введите "+B" (без нажатия ENTER). Строка в нижней части экрана примет вид:

```
20 PRINT A+B
```

Теперь нажмите ENTER. Это вызовет замену старой строки 20 на новую и на экране будет выглядеть так:

```
10      LET A=10
15      LET B=15
20  >   PRINT A+B



K
```

Запустив программу нажатием RUN и ENTER,получаем на экране сумму.

Выполните теперь команду PRINT A,B. Переменные сохраняются даже после завершения программы.

Есть еще одно применение команды EDIT. Допустим вам надо удалить всю строку, набранную в нижней части экрана. Для этого вы можете нажать и удерживать до конца строки клавишу DELETE, но можно сделать тоже самое быстрее: нажать EDIT, что вызовет копирование текущей строки в нижнюю часть экрана, затем нажать ENTER. Строка заменит такую же в программе, и нижняя часть экрана очистится.

Введите строку:

```
12 LET B=B
```

Теперь для удаления этой строки наберите:

```
12    (и затем ENTER)
```

Программный курсор станет между строками 10 и 15, но клавишами управления курсором вы можете установить его в любую строку. Еще раз введите 12 и ENTER, курсор снова установится между строками 10 и 15. Теперь нажмите EDIT и строка 15 будет копироваться в нижнюю часть экрана. Оператор EDIT копирует вниз строку, следующую за строкой с новым номером. Нажмите ENTER для очистки нижней части экрана.

Теперь введите:

```
30    ( и затем ENTER)
```

Программный курсор установится после конца программы. Если вы теперь нажмете EDIT, то вниз будет переслана строка 20.

Наконец, выполните команду:

```
LIST 15
```

Теперь вы увидите на экране

```
15 LET B=15
20 PRINT A+B
```

Строка 10 не отображается на экране, но она сохраняется в вашей программе. Вы можете убедиться в этом, нажав ENTER.

Команда LIST 15 указывает, что надо отобразить листинг со строки с номером 15, и устанавливает в эту строку программный курсор. Это бывает очень удобно при просмотре длинных программ.

Другое назначение номеров строк - служить именем оператора при ссылке на него из другого места программы (GO TO N).

Команда LIST без номера строки выдает листинг, начиная от первой строки.

Команда NEW очищает память компьютера от старых программ и переменных.

Теперь выполним программу, переводящую значения температуры в градусах по Фаренгейту в температуру по Цельсию:

```
10 REM TEMPERATURE CONVERSION
20 PRINT "DEG F","DEG C"
30 PRINT
40 INPUT "ENTER DEG F",F
50 PRINT F,(F-32)*5/9
60 GO TO 40
```

Вы увидите, что заголовок выводится в строке 20, и у вас возникает вопрос: "Что же делает строка 10?" Компьютер игнорирует эту строку. Это комментарий (REMARC или REMINDER). Все, что следует после REM, игнорируется компьютером до конца строки.

Вычисления доходят до строки 40 и компьютер переходит в ожидание ввода вами значения переменной F. Вы можете ввести это значение в режиме (L). Наберите число и нажмите ENTER. Компьютер выведет результат и снова перейдет в ожидание ввода следующего числа. Этот переход обеспечивается в строке 60 оператором GO TO 40. Если на запрос очередного числа ответить STOP, то компьютер остановится с выдачей сообщения: "H STOP IN INPUT IN LINE 40:1", которое поясняет причину и место останова (первый оператор в строке 40).

Если теперь вы желаете вновь продолжить выполнение программы, то введите CONTINUE, и компьютер запросит очередное число. При использовании CONTINUE компьютер запоминает (до выдачи им 'O ok') номер последней выполнявшейся строки и продолжает выполнение программы, начиная именно с этой строки.

Посмотрите внимательно на оператор PRINT в строке 50. Запятая в нем очень важна. Запятые в операторе PRINT используются для указания того, что вывод, следующий после запятой, должен продолжаться либо с левого края экрана, либо с его середины в зависимости от того, какая по порядку запятая в данном операторе.

Так, в строке 50 запятая предписывает выводить значения температуры в градусах Цельсия с середины экрана.

Если использовать в операторе PRINT вместо запятой точку с запятой (;), то очередные данные будут выводиться непосредственно после предыдущих.

Оператор в строке 30 выводит чистую строку.

Оператор PRINT всегда начинает вывод с начала следующей строки, но это можно изменить, поставив в конце предыдущего оператора PRINT запятую или точку с запятой:

```
50 PRINT F,
60 PRINT F;
```

Не путайте эти знаки с двоеточием (:), которое используется только для разделения разных операторов в строке.

Теперь наберите несколько программных строк:

```
100 REM THIS POLITE PROGRAM REMEMBER YOU NAME
110 IMPUT N$
120 PRINT "HELLO";N$;"!"
130 GO TO 110
```

Эта программа никак не связана с ранее набранной нами программой, но обе программы можно держать в памяти одновременно.

Для того, чтобы выполнить отдельно только последнюю программу, надо ввести команду:

```
RUN 100
```

Эта программа вводит строку символов, что должно указываться строковыми кавычками. Если их опустить, то компьютер попытается найти переменную с таким же именем и использовать ее значение в качестве IMPUT-данных. Например, ответьте программе

```
N$    (Удалив кавычки)
```

Это сделает оператор INPUT в строке 110 подобным оператору

```
LET N$=N$
```

Если вы решили ввести STOP при строковом вводе, то необходимо установить курсор в начало строки, используя клавишу управления курсором <влево>.

Действие команды RUN 100 подобно действию оператора GO TO, но имеются и различия. RUN 100 очищает все переменные и экран и после этого выполняет GO TO 100. Другое отличие состоит в том, что вы можете указать RUN без номера строки, и тогда выполнение начнется с первой строки, а оператор GO TO всегда должен содержать номер строки.

Обе приведенные программы останавливались нами вводом команды STOP. Но могут быть программы, которые невозможно остановить подобным образом, например:

```
200 GO TO 200
RUN 200
```

Остановить эту программу можно, если нажать клавиши CAPS SHIFT и SPACE. Это вызовет ввод команды BREAK, которая остановит работу с выдачей сообщения "L BREAK INTO PROGRAM". Команда BREAK может быть использована и во время выполнения операции на магнитофоне или принтере. В этом случае выдается сообщение "D BREAK-

COMT REPEATS". Команда CONTINUE в этом случае (как и в большинстве других) вызовет выполнение оператора, на котором произошла остановка. Ввод команды CONTINUE после сообщения "L BREAK INTO PROGRAM" продолжит выполнение программы со следующего оператора.

Запустите вторую программу снова и, когда она запросит ввод, введите:

```
N$   (Удалив кавычки)
```

Поскольку значение "N$" неопределено, то будет выдано сообщение "2 VARIABLE NOT FOUND". Если теперь вы введете:

```
LET N$="SOMETHING DEFINITE"
```

то компьютер ответит "0 OK, 0:1". Затем введите CONTINUE. Вы увидите, что программа завершится нормально.

Как уже отмечалось, сообщение "L BREAK INTO PROGRAM" имеет особенность, так как ввод после него CONTINUE не вызывает повтора команды, на которой произошел останов.

Все приведенные в этой главе утверждения PRINT, LET, INPUT, RUN, LIST, GO, CONTINUE, NEW, REM могут использоваться либо как операторы в программе, либо как команды. Хотя RUN, LIST, CONTINUE и NEW чаще используются как команды, они также могут быть использованы и в программах.

Глава 3

УСЛОВИЯ

Краткое содержание: IF, STOP, =, >, <, <=, >=, <>.

Последовательность выполнения операторов программы не всегда предсказуема. В определенных местах программы компьютер может принимать решение о дальнейшем ходе вычислений. Оператор, реализующий это, имеет форму:

```
IF - некоторое истинное или ложное условие
THEN - некоторое действие.
```

Например, введите команду NEW, а затем наберите и запустите на выполнение следующую программу (это игра для двух человек):

```
10 REM GUESS THE NUMBER    (Угадайте числа)
20 INPUT A: CLS
30 INPUT "GUESS THE NUMBER", B
40 IF A=B THEN PRINT "THIS IS CORRECT": STOP
50 IF B<A THEN PRINT "THIS IS TOO SMALL, TRY AGAIN"
60 IF B>A THEN PRINT "THIS IS TOO BIG, TRY AGAIN"
70 GO TO 30
```

Здесь IF-оператор имеет форму:

```
IF условие THEN ...,
```

где "..." - последовательность операторов, разделенных двоеточием обычным образом. Если "условие" истинно, то выполняются операторы, следующие после THEN. В противном случае они пропускаются, и выполнение программы продолжается со следующего оператора.

Простейшим условием может быть сравнение двух чисел или двух строк. Числа могут быть равны, либо одно больше другого, а строки либо равны, либо одна следует после другой в алфавитном порядке.

Для задания условий используются отношения: =, <, >, <=, >=, <>. Например, выражения 1<2, -2<1, -3<1 истинны, а выражения 1<0, 0>2 ложны.

Строка программы 40 сравнивает числа "A" и "B", и если они равны, программа завершает работу, выполнив команду STOP. При этом будет выдано сообщение "9 STOP STATEMENT, 30:3", показывающее, что команда STOP была выдана в 3-ем операторе в 30-й строке.

Знаки условий набирают на клавиатуре следующим образом:

> - SYMBOL SHIFT вместе с T - больше;
< - SYMBOL SHIFT вместе с R - меньше;
<= - SYMBOL SHIFT вместе с Q - меньше или равно (нельзя набрать < и = по отдельности);
>= - SYMBOL SHIFT вместе с E - больше или равно;
<> - SYMBOL SHIFT вместе с W - не равно.

Г л а в а 4

Ц и к л ы

Краткое содержание: FOR, NEXT, TO, STEP.

Допустим нам необходимо составить программу, подсчитывающую сумму пяти вводимых чисел. Это можно было бы сделать так:

```
10 LET TOTAL=0
20 INPUT A
30 LET TOTAL=TOTAL+A
40 INPUT A
50 LET TOTAL=TOTAL+A
60 INPUT A
70 LET TOTAL=TOTAL+A
80 INPUT A
90 LET TOTAL=TOTAL+A
100 INPUT A
110 LET TOTAL=TOTAL+A
120 PRINT TOTAL
```

Получилась большая и не очень оптимальная программа. Можно решить эту задачу более рационально, если ввести счетчик и оператор GOTO

```
10 LET TOTAL=0
20 LET COUNT=1
30 INPUT A
40 REM COUNT=NUMBER OF TIME THAT A HAS BEEN INPUT SO FAR
50 LET TOTAL=TOTAL+A
60 LET COUNT=COUNT+1
70 IF COUNT<=5 THEN GO TO 30
80 PRINT TOTAL
```

Теперь, изменив условие в строке 70, можно ввести не только 5, но и любое количество чисел. Для организации в программе таких счетчиков существуют специальные операторы FOR и NEXT, которые всегда используются вместе.

Наша программа при использовании этих операторов будет выглядеть так:

```
10 LET TOTAL=0
20 FOR C=1 TO 5
```

```
30 INPUT A
40 REM C=NUMBER OF TIME THAT A HAS BEEN INPUT SO FAR
50 LET TOTAL=TOTAL+A
60 NEXT C
70 PRINT TOTAL
```

Здесь "с" – управляющая переменная цикла – должна иметь имя в одну букву. "с" последовательно принимает значения 1, 2, 3, 4 и 5 (предел – конечное значение управляющего цикла) и при каждом переходе выполняются строки 30, 40, 50. После того, как "с" примет пятое значение, выполняется 70-я строка. Значение приращения управляющей переменной составляет 1. Но это значение можно изменить, используя указание STEP как часть оператора FOR. Таким образом, общая форма оператора FOR выглядит так:

```
FOR "управл.перем" = "нач.знач" TO "предел" STEP "шаг прираш"
```

Здесь "начальное значение", "предел", "шаг приращения" – есть выражения, принимающие числовые значения. Итак, если вы замените строку 20 программы на

```
20 FOR C=1 TO 5 STEP 3/2
```

то "с" последовательно примет значения 1, 2.5, 4.

Выполните программу, выводящую числа от 1 до 10 в убывающей последовательности:

```
10 FOR N=10 TO 1 STEP -1
20 PRINT N
30 NEXT N
```

Следующая программа выводит числа домино:

```
10 FOR M=0 TO 6
20 FOR N=0 TO M
30 PRINT M; ":"; N;" " ;
40 NEXT N
50 PRINT
60 NEXT M
```

Значение STEP, равное нулю, вызовет бесконечное повторение цикла. Этого не рекомендуется делать.

Глава 5

ПОДПРОГРАММЫ

Краткое содержание: GO SUB, RETURN.

Иногда бывает удобно некоторые фрагменты программы представить в виде отдельных частей, по нескольку раз используемых в различных местах программы. Такие части оформляются как подпрограммы, которые могут вызываться в любом месте программы.

Для этого используются операторы GO SUB (GO TO SUBROUTINE) и RETURN в форме:

```
GO SUB N
```

где N – номер строки в программе. Этот оператор подобен GO TO N с той разницей, что при использовании GO SUB компьютер запоминает следующий после GO SUB оператор, которому и передается управление

после выполнения подпрограммы. Делается это посредством помещения номера оператора (адреса возврата) в специальную область памяти, называемую GO SUB-стек.

RETURN выбирает верхний адрес возврата из GO SUB-стека и продолжает выполнение программы, начиная с оператора, следующего после оператора с этим номером.

Приведем пример использования подпрограммы:

```
100 LET X=10
110 GO SUB 500
120 PRINT S
130 LET X=X+4
140 GO SUB 500
150 PRINT S
160 LET X=X+2
170 GO SUB 500
180 PRINT S
190 STOP
500 LET S=0
510 FOR Y=1 TO X
520 LET S=S+Y
530 NEXT Y
540 RETURN
```

В общем случае подпрограмма может вызывать другие подпрограммы и даже саму себя (такая подпрограмма называется рекурсивной).

Глава 6

ОПЕРАТОРЫ READ, DATA И RESTORE

Краткое содержание: READ, DATA, RESTORE.

В некоторых предыдущих программах мы видели, что информация или данные могут быть введены в компьютер при помощи оператора INPUT. Иногда это может быть очень утомительно, особенно если многие данные повторяются каждый раз при выполнении программы. Вы можете сэкономить много времени, используя команды READ и DATA.

```
30 DATA "JUNE 1ST, 1982"
40 STOP
```

Это простой способ получения выражения из DATA списка: старт и выполнение от начала до тех пор, пока не будет достигнут конец. Однако вы можете использовать и программный переход для DATA списков. В этом случае используется оператор RESTORE с указанием после него номера строки с оператором DATA, и все последовательно встречающиеся в программе операторы READ вводят данные подряд, начиная от первого оператора DATA. Вы можете не указывать номер строки в операторе RESTORE, в этом случае - указатель данных становится на первый оператор в программе.

Попробуйте выполнить такую программу:

```
10 READ A,B
20 PRINT A,B
30 RESTORE 10
40 READ X,Y,Z
50 PRINT X,Y,Z
```

```
60 DATA 1,2,3
70 STOP
```

В этой программе переменным, вводимым в строке 10, будут присвоены значения A=1 и B=2. Оператор RESTORE 10 сбрасывает указатель данных в начальное положение, и строка 40 присвоит переменным X,Y,Z значения, начиная от первого значения в DATA.

Выполните программу без строки 30, и вы увидите что из этого получится.

Глава 7

АРИФМЕТИЧЕСКИЕ ОПЕРАЦИИ

Краткое содержание: операции +, -, *, /; выражения, условные обозначения, имена переменных.

Вы уже видели несколько примеров, в которых ZX SPECTRUM может оперировать числами. Можно выполнять четыре арифметические операции: +, -, * и / (помните, что * используется для умножения, а / используется для деления), и при этом определяется значение переменной, задаваемой именем.

Пример:

```
LET TAX=SUM*15/100
```

Из примера видно, что вычисления могут быть комбинированными. Комбинации такого типа, как SUM*15/100, называются выражениями. Выражение - самый короткий путь указания компьютеру на то, что вычисления надо делать одно за другим. В нашем примере выражение указывает: возьми значение переменной с именем "SUM", умножь его на 15, а затем раздели на 100. Если вы не можете еще этого сделать, мы рекомендуем вам посмотреть вводную часть этой книги, чтобы ознакомится с тем, как ZX SPECTRUM работает с числами и каков порядок, в котором выполняются математические выражения.

Краткое повторение: умножение и деление выполняются первыми, они имеют более высокий приоритет, чем сложение и вычитание; относительно друг друга умножение и деление имеют равные приоритеты; согласно существующему правилу умножение и деление выполняются последовательно слева направо, а после них также по порядку слева направо выполняются сложение и вычитание.

Для задания приоритета в компьютере ZX SPECTRUM используются числа от 1 до 16. Например, операции "*" и "/" имеют приоритет 8, а "+" и "-" - 6. Этот порядок вычислений является жестким, но его можно изменить при помощи скобок. Выражение в скобках вычисляется первым, а затем подставляется в общее выражение как одно число.

Вы можете использовать операцию сложения ("+") для сцепления строк (конкатенации) в выражениях.

Имя строковой переменной состоит из буквы с последующим знаком "$", имя управляющей переменной в FOR-NEXT-цикле должно состоять из одной буквы, а имена обычных числовых переменных могут выбираться произвольно. Они могут содержать несколько букв и цифр, но первой всегда должна быть буква.

Вы можете вставлять в имена пробелы для удобства чтения, поскольку компьютер считает их частью имени. Запись имени прописными или заглавными буквами не делает их различными.

Примеры допустимых имен переменных:

```
X
T42
THIS NAME IS SO LONG THAT SMALL NEVER BE ABLE TO
     TYPE IT OUT AGAIN WITHOUT
MAKING A MISTAKE
NOW WE ARE SIX      }    эти два имени указывают на
NOWWEARESIX         }    одну и ту же переменную
```

Примеры недопустимых имен переменных:

```
20011                    (начинается с цифры)
3 BEARS                  (начинается с цифры)
M*A*S*H                  (знак "*" - не буква и не цифра)
FOTHERINGTON-THOMAS      (содержит знак "-")
```

Числа в выражениях могут задаваться в экспоненциальной форме. Попробуйте выполнить:

```
PRINT 2.34E0
PRINT 2.34E1
PRINT 2.34E2              И Т.Д. ДО
PRINT 2.34E15
```

Помните, что оператор PRINT дает лишь 8 значащих цифр числа. Попробуйте выполнить

```
PRINT 4294967295, 4294967295-429E7
```

и вы увидите, что компьютер может воспринять только цифры 4294967295.

Компьютер ZX SPECTRUM использует арифметику с плавающей точкой (запятой). При этом различные части числа (мантисса и порядок) хранятся в отдельных байтах, что приводит к не всегда точным результатам даже для целых чисел. Выполните:

```
PRINT 1E10+1-1E10, 1E10-1E10+1
```

1E10 и 1E10+1 не различаются компьютером как разные числа (1E10 усекается справа).

Еще один более наглядный пример:

```
PRINT 5E9+1-5E9
```

Погрешность в представлении числа 5E9 составляет около 1, а с прибавлением 1 фактически округлится до 2.

Числа 5E9+1 и 5E9+2 для компьютера равны. Наибольшее целое, которое может воспринять компьютер, равно $2^{32}-1$ или 4 294 967 295.

Строка "" без единого символа называется пустой или нулевой строкой (не путайте ее с пробелом). Наберите:

```
PRINT "HAVE YOU FINISHED "FINNEGANS WAKE" YET?"
```

Когда вы нажмете клавишу ENTER, вы получите мерцающий знак вопроса, указывающий ошибочное место в строке. Когда при интерпритации этой строки компьютер найдет двойную кавычку, открывающую "FINNEGANS WAKE", то сочтет ее закрывающей кавычкой для выражения "HAVE YOU FINISHED" и затем не сможет вывести "FINNEGANS WAKE". Здесь нужно помнить специальное правило: если вы хотите вывести кавычки внутри строки, то они должны удваиваться,

например:

```
PRINT "HAVE YOU FINISHED ""FINNEGANS WAKE"" YET?"
```

В данном случае будет выведена фраза, обрамленная двойными кавычками - "FINNEGANS WAKE".

Глава 8

СТРОКИ СИМВОЛОВ

Краткое содержание: сечения, использование TO.

Примечание: Эти операции отсутствуют в стандартном Бейсике.

Пусть имеется строка символов, тогда ее подстрокой будет некоторая последовательность символов из этой строки. Так, "STRING" является подстрокой от "BIGGER STRING", а "B STRING" и "BIG STRING" не являются.

Существует действие, называемое сечением, для определения подстрок, которое может применяться к строковым выражениям. Общая его форма:

"строковое выражение"("начало" TO "конец")

Следующее выражение истинно:

```
"ABCDE" (2 TO 5) = "BCDE"
```

Если опущено "начало", то по умолчанию подразумевается 1, если "конец", то подразумевается длина всей строки. Так:

```
"ABCDEF" (TO 5) = "ABCDEF" (1 TO 5) = "ABCDE"
"ABCDEF" (2 TO) = "ABCDEF" (2 TO 6) = "BCDEF"
"ABCDEF"  (TO)  = "ABCDE"  (1 TO 6) = "ABCDEF"
```

Последнее выражение можно записать и так: "ABCDEF"(), что тоже верно. Можно опускать и слово TO:

```
"ABCDEF" (3) = "ABCDEF" (3 TO 3) = "C"
```

"Начало" и "конец" должны находиться в пределах строки, иначе будет выдано сообщение об ошибке. Так, выражение "ABCDEF"(5 TO 7) вызывает сообщение "3 SUBSCRIPT WRONG", так как "конец" превышает длину строки (6).

Если "начало" больше, чем "конец", либо обе границы лежат за пределами строки, то результатом будет пустая строка:

```
"ABCDEF" (8 TO 7) = ""
"ABCDEF" (1 TO 0) = ""
```

"Начало" и "конец" не могут быть отрицательными, иначе выдается сообщение "B INTEGER OUT OF RANGE".

Следующая программа иллюстрирует эти правила:

```
10 LET A$="ABCDEF"
20 FOR N=1 TO 6
30 PRINT A$(N TO 6)
40 NEXT N
50 STOP
```

Можно также присваивать значения подстроке. Попробуйте:

```
10 LET A$="I AM ZX SPECTRUM"
```

```
20 PRINT A$
30 LET A$(5 TO 8)="******"
40 PRINT A$
```

Подстрока A$(5 TO 8) имеет длину только в 4 символа, поэтому будут использованы только первые четыре звездочки. Это - особенность присвоения значения подстроке: длинные данные усекаются справа, а короткие добавляются пробелами до длины подстроки. Это действие называют прокрустианом в честь мифического разбойника Прокруста, который своим гостям либо обрубал ноги, либо вытягивал их, если они не подходили по длине к его кровати.

Выполните теперь:

```
LET A$()="HELLO THERE"
PRINT A$;"."
```

Вы увидите, что будут выведены дополнительные пробелы, так как "A$()" считается подстрокой. Для правильного выполнения следует писать:

```
LET A$="HELLO THERE"
```

Можно использовать скобки, что позволяет вычислять значение строкового выражения перед тем, как брать сечение. Например:

```
"ABC" + "DEF"(1 TO 2) = "ABCDE"
("ABC" + "DEF")(1 TO 2) = "AB"
```

Глава 9

ФУНКЦИИ

Краткое содержание: DEF, LEN, STR$, VAL, SGN, ABS, INT, SQR, FN.

Функции - это защитые в Бейсик-систему программы, которые, получая на входе одни значения, называемые аргементами, возвращают другие значения - результат.

Функции используются в выражении простым включением в него имени функции с последующими аргументами.

При вычислении выражения вычисляется и значение функции. Например, функция LEN возвращает длину заданного в ней строкового аргумента. Вы можете записать:

```
PRINT LEN "SINCLAIR"
```

Компьютер выведет ответ 8, т.е. количество букв в слове "SINCLAIR" (для ввода с клавиатуры имени функции LEN, вы должны войти в необходимый режим, нажав клавиши CAPS SHIFT и SYMBOL SHIFT - курсор изменится с (L) на (K) - и нажать клавишу K.

Если в одном выражении используется и функция и оператор, то функции будут вычислены перед выполнением любых операций. Однако вы можете изменить этот порядок, применяя скобки.

Функция STR$ преобразует число в символьный вид, подобно формату вывода оператором PRINT:

```
LET A$=STR$ 1E2
```

Приведенная команда аналогична по действию

```
LET A$="100"
```

Выполните

```
PRINT LEN STR$ 100.000
```

Вы получите ответ – 3, так как STR$ 100.000="100".

Функция VAL – обратная по отношению к STR$ и преобразует строку в число. Так,

```
VAL "3.5" = 3.5
VAL "2*3" = 6
VAL ("2"+"*3") = 6
```

В последнем случае происходит вычисление двух выражений – сначала строкового с получением строки "2*3", а затем числового с получением "6".

Можно попасть в затруднительное положение, например:

```
PRINT VAL " VAL" "VAL" " " "2" " " " " " "
```

Помня, что внутри строки кавычки удваиваются, мы видим, что в нашем случае может понадобиться учетверение или даже увосьмирение.

Имеется еще одна функция, подобная VAL – VAL$. И аргументом, и результатом этой функции является строка символов. Она работает как VAL, примененная дважды, раскрывая кавычки в строках:

```
VAL$ """FRUIT PUNCH""" = "FRUIT PUNCH"
```

Введите LET A$="99" и выведите все следующие значения:

```
VAL A$
VAL "A$"
VAL """A$"""
VAL$ A$
VAL$ "A$"
VAL$ """A$"""
```

Некоторые из них сработают, а некоторые нет. Проанализируйте все ответы.

Функция SGN – это математическая функция, называемая сигн (знак). И аргумент, и результат ее – числовые величины. Результат равен 1,если аргумент положителен, равен 0, если аргумент равен нуль, и равен –1, если аргумент отрицателен.

Функция ABS преобразует аргумент в положительное число:

```
ABS -3.2 = ABS 3.2 = 3.2
```

Функция INT (от "integer part" – целая часть) преобразует дробное число к целому отбрасыванием дробной части:

```
INT -3.9 = -4
```

Функция SQR вычисляет квадратный корень от числа, например:

```
SQR 4 = 2
SQR 0.25 = 0.5
SQR 2 = 1.4142136    (приближенно)
```

Если аргумент отрицательный, то выводится сообщение: "A INVALID ARGUMENT".

Вы также можете сами определить для себя какую-нибудь функцию, указав FN и имя этой функции (букву, если аргумент числовой, или букву и $, если аргумент строковый). Аргументы обязательно должны быть заключены в скобки.

Вы можете определить функцию вводом оператора DEF в некотором месте программы. Например, зададим функцию, вычисляющую квадрат числа:

```
10 DEF FN S(X) = X*X: REM THE SQUARE OF X
```

DEF вводится в соответствующем режиме (SYMBOL SHIFT и 1).

После определения функция может использоваться в программе:

```
PRINT FN S(2)
PRINT FN S(3+4)
PRINT 1+INT FN S(LEN "CHICKEN"/2+3)
```

Функция INT всегда округляет до целого; для округления до 0.5 надо добавить к результату ".5". Вы можете задать для себя такую функцию:

```
20 DEF FN R(X)=INT(X+.5): REM GIVES X ROUNDED TO
                                  NEAREST INTEGER
```

и можете затем попробовать ввести:

```
FN R(2.9)=3      FN R(2.4)=2
FN R(-2.9)=-3    FN R(-2.4)=-2
```

Введите и выполните следующее:

```
10 LET X=0: LET Y=0: LET A=10
20 DEF FN P(X,Y)=A+X*Y
30 DEF FN Q()=A+X*Y
40 PRINT FN P(2,3), FN Q()
```

В этой программе есть одна тонкость: во-первых, функция FN Q не использует аргументов, но скобки при этом должны обязательно быть; во-вторых, операторы DEF - невыполняемые, компьютер после выполнения строки 10 переходит к выполнению строки 40 (помните, что DEF может быть только оператором, но не командой); в-третьих, "X" и "Y" - имена целых переменных в программе и в то же время - имена аргументов функции FN P.

Функция FN P использует при вычислении результата значения аргументов "X" и "Y" и переменной "A", не являющейся аргументом. Так, когда вычисляется FN P(2,3), значение "A" равно 10, как это и определено в программе, а значения "X" и "Y" - соответственно 2 и 3, так как они аргументы, и результат будет равен 10+2*3=16.

При вычислении FN Q() используются только переменные программы, так как аргументов нет, и результат в этом случае будет равен 10+0*0=10.

Теперь изменим строку 20 на

```
20 DEF FN P(X,Y)=FN Q()
```

В этом случае FN P(X,Y) будет возвращать значение 10.

Некоторые версии Бэйсика имеют функции LEFT$, RIGHT$ и TL$:

LEFT$ (A$,N) - возвращает подстроку, содержащую "N" первых символов строки "A$";

RIGHT$(A$,N) - возвращает подстроку, содержащую "N" последних символов строки "A$";

TL$(A$) - возвращает подстроку, содержащую все символы строки "A$", кроме последнего.

Вы можете определить следующие функции на своем компьютере:

```
10 DEF FN T$(A$)=A$(2 TO): REM TL$
```

```
20 DEF FN L$(A$,N)=A$(TO N): REM LEFT$
```

Проверьте работу функции со строками 1 и 0.

Примечание: Функция может иметь до 26 числовых аргументов и в то же время до 26 строковых.

Глава 10

МАТЕМАТИЧЕСКИЕ ФУНКЦИИ

Краткое содержание: **, PI, EXP, LN, SIN, COS, TAN, ASN, ACS, ATN.

В этой главе описываются математические функции, которые могут быть выполнены на ZX SPECTRUM. Вполне возможно, что вам никогда не придется воспользоваться ими, и если вы сочтете их слишком сложными, можете пропустить эту главу. Все сказанное относится к функциям: ** (возведение в степень), EXP, LN, тригонометрическим функциям SIN, COS, TAN и обратным к ним ASN, ACS, ATN.

Вы можете возвести число в некоторую степень путем многократного умножения его на себя необходимое число раз. Это обычно изображается записью числа, обозначающего степень, справа вверху от числа, обозначающего основание. Но такую запись трудно реализовать в компьютере, поэтому используется специальный символ (направленная вверх стрелка, в данном описании заменена двумя звездочками: "**"). Например, степени двойки можно представить так:

```
2**1=2
2**2=2*2=4      (два в квадрате)
2**3=2*2*2=8    (два в кубе)
```

Таким образом, запись A**B означает: умножить "A" само на себя "B" раз. Это предполагает, что "B" - положительное число.

Для нахождения определения для этого действия при других значениях A и B запишем выражение:

```
A**(B+C)=A**B+A**C
```

Здесь надо помнить, что операция "**" имеет более высокий приоритет, чем "*" и "/". Вы можете быть уверены в правильности этого выражения, если "B" и "C" целые положительные числа. Но если это не так, а вы все-таки решили выполнить возведение в степень, то вы должны знать,что

```
A**0=1
A**(-B)=1/(A**B)
A**(1/B)= корню B-ой степени из A
A**(B*C)=(A**B)**C
```

Полезно помнить, что

```
A**(-1)=1/A
A**(1/2)=SQR A
```

Поэкспериментируйте с этим, попробовав выполнить такую программу:

```
10 INPUT A,B,C
20 PRINT A**(B+C), A**B*A**C
```

```
30 GO TO 10
```

Компьютер будет выводить два числа, если вы конечно правильно набрали программу. Число A, кстати, не должно быть отрицательным.

Другой типичный пример использования этой операции - это вычисление дохода. Предположим, что вы вложили часть своих денег в общественное строительство, которое приносит вам 15% годовых. После года вы будете иметь уже не точно 100% от того, что имели вначале, а плюс 15% дохода, что составит 115%. При вычислении другим способом вы умножаете вашу сумму денег на 1.15 и получаете тот же результат. В конце следующего года вы снова будете иметь прибыль, что в сумме составит 1.15*1.15=1.15**2=1.3225 от вашей первоначальной суммы. В итоге после Y лет вы будете иметь в 1.15**Y раз больше денег.

Выполнив операторы:

```
FOR Y=0 TO 100: PRINT Y,10*1.15**Y: NEXT Y
```

вы увидите, что начиная с 10 фунтов можно получать все больший и больший доход.

Такой тип поведения функции, когда после фиксированного числа интервалов времени, значения функции пропорциональны количеству умножений этого числа самого на себя, называется экспоненциальным законом.

Предположим вы записали:

```
10 DEF FN A(X)=A**X
```

Здесь A определено в операторе LET и ее значение передается для вычисления степени.

Имеется определенное значение A, которое делает функцию FN A иллюстрирующей специальную функцию. Это значение называется "E". ZX SPECTRUM имеет специальную функцию, называемую EXP и определяемую как:

```
EXP X=E**X
```

К сожалению "E" не может быть представлено точным числом. Вы можете увидеть его пять первых десятичных знаков, выполнив

```
PRINT EXP 1
```

так как EXP 1=E**1=E. Конечно это лишь первое приближение, вы никогда не сможете записать "E" абсолютно точно.

Обратной к экспоненциальной функции является логарифмическая функция. Логарифм (по основанию A) числа X есть степень, в которую надо возвести A, чтобы получить X. Это записывается как LOGA X. (Выражение A**LOGA X=X так же верно, как и LOGA(A**X)=X).

Вам должно быть уже известно как используется логарифм по основанию 10 для умножения. Такой логарифм называется общим. ZX SPECTRUM имеет функцию LN, которая вычисляет логарифм по основанию "E", называемый натуральным. Для вычисления логарифма с другим основанием надо разделить натуральный логарифм искомого числа на натуральный логарифм основания:

```
LOGA X=LN X/LN A
```

Допустим имеется некоторый круг. Вы можете найти его периметр (длину окружности), умножив диаметр круга на число, называемое PI. Подобно "E" PI представляется бесконечной десятичной дробью, начало которой 3.141592653589...

Слово PI в ZX SPECTRUM обозначает это число. Выполните:

```
PRINT PI
```

Тригонометрические функции отражают случаи, когда точка перемещается по окружности единичного радиуса. Точка стартует с позиции 3-х часов и перемещается против часовой стрелки. Начало координат находится в центре этой окружности. Тогда SIN угла между радиусом, соединяющим движущуюся по окружности точку с началом координат, будет ордината этой точки, а COS - абцисса. Необходимо помнить, что если точка находится под осью X, то синус отрицательный. Необходимо также помнить, что

```
SIN (A+2*PI) = SIN A
COS (A+2*PI) = COS A
```

Имеются и другие тригонометрические функции:

```
TAN - тангенс
ASN - арксинус
ACS - арккосинус
ATN - арктангенс
```

Помните, что в ZX SPECTRUM тригонометрические функции вычисляются в радианах. Для перевода числа из градусов в радианы необходимо разделить его на 180 и умножить на PI, а для обратного преобразования - разделить на PI и умножить на 180.

Глава 11

СЛУЧАЙНЫЕ ЧИСЛА

Краткое содержание: RANDOMIZE, RND

В этой главе описывается функция RND и ключевое слово RANDOMIZE. Их не надо путать, хотя они оба расположены на клавише "T". Для RANDOMIZE допустимо сокращение RAND.

При обращении к функции RND возвращается случайное число в интервале от 0 до 1 (может быть равно 0, но всегда меньше 1). Попробуйте выполнить:

```
10 PRINT RND
20 GO TO 10
```

Вы увидите, как меняется результат.

Фактически RND не абсолютно случайное число, а выбирается из определенной последовательности длиной в 65536 чисел. Поэтому говорят, что RND - псевдослучайное число.

Для получения случайного числа в интервале, отличном от [0,1], можно использовать выражения. Например:

```
1.3+0.7*RND
```

даст интервал от 1.3 до 2.

Для получения целых случайных чисел используйте функцию INT (округляет число с отбрасыванием дробной части). Например:

```
1+INT(RND*6)
```

будет давать числа 1, 2, 3, 4, 5, 6.

Пусть имеется программа:

```
10 REM DICE THROWING PROGRAM   (выбрасывание кости)
20 CLS
30 FOR N=1 TO 2
40 PRINT 1+INT(RND*6);" ";)
50 NEXT N
60 INPUT A$: GOTO 20
```

Нажимая ENTER, вы каждый раз будете получать номер, выпавший на кости.

Утверждение RANDOMIZE используется для установления начала последовательности случайных чисел для функции RND. Как можно увидеть из программы

```
10 RANDOMIZE 1
20 FOR N=1 TO 5: PRINT RND: NEXT N
30 PRINT: GO TO 10
```

после каждого выполнения RANDOMIZE 1 случайная последовательность будет начинаться с числа 0.0022735596. В утверждении RANDOMIZE вы можете использовать любые числа в интервале от 1 до 65535. Нельзя использовать RANDOMIZE без числа, а также RANDOMIZE 0. Например:

```
10 RANDOMIZE
20 PRINT RND: GO TO 10
```

В каждой итерации будет печататься не случайное число. Для улучшения случайности распределения можно заменить GO TO 10 на GO TO 20.

В дополнение, большинство версий Бейсика используют RND и RANDOMIZE для генерации случайных чисел, но это не единственное их применение.

Ниже приводится текст программы, моделирующей выбрасывание монеты и подсчет числа выпадений "орла" и "решки". (Перевод имен программы HEADS - орлы, TAILS - решки, COIN - монета).

```
10 LET HEADS=0: LET TAILS=0
20 LET COIN=INT(RND*2)
30 IF COIN=0 THEN LET HEADS=HEADS+1
40 IF COIN=1 THEN LET TAILS=TAILS+1
50 PRINT HEADS;",";TAILS
60 IF TAILS <> 0 THEN PRINT HEADS/TAILS
70 PRINT: GOTO 20
```

Если программа работает достаточно долго, то отношение "орлов" к "решкам" будет приблизительно равно 1.

Глава 12

МАССИВЫ

Краткое содержание: DIM.

Допустим у вас имеется последовательность из чисел, каким-то образом описывающая 10 человек. Для записи в память компьютера вы должны будете завести переменную на каждого человека. Это неудобно, так как приходится обращаться к данным, вызывая каждый раз новую переменную, например, BLOGGS1, BLOGGS2 и т.д. до BLOGGS10. Как это неудобно вы можете убедиться из программы:

```
 5 REM THIS PROGRAM WILL NOT WORK
10 FOR N=1 TO 10
20 READ BLOGGSN
30 NEXT N
40 DATA 10, 2, 5, 19, 16, 3, 11, 1, 0, 6
```

Имеется специальный аппарат для подобного случая - применение массивов. Переменные в массиве являются его элементами, обладают общим именем и отличаются только номером, записываемым после имени (индексом).

В нашем примере именем будет B (подобно управляющим переменным в FOR-NEXT-утверждениях имя массива должно быть уникальным в данной программе), а десятью переменными будут B(1), B(2) и т.д. до B(10).

Элементы массива называют индексируемыми переменными. Перед использованием массива необходимо зарезервировать под него память. Это делается оператором DIM (от английского "dimension"). В нашем случае это будет DIM B(10), который определяет массив с именем B и размерностью 10 (т.е. 10 индексируемых переменных B(1), B(2), ..., B(10)) и присваивает всем элементам массива значение 0.

Итак теперь мы можем записать:

```
10 FOR N=1 TO 10
20 READ B(N)
30 NEXT N
40 DATA 10, 2, 5, 19, 16, 3, 11, 1, 0, 6
```

Можно также объявлять массивы с более, чем одной размерностью. Например, в двумерном массиве первый индекс можно сравнивать с номером строки, а второй - с позицией в строке. Такой массив как бы описывает страницу. Если ввести третье измерение для номера страницы, то массив будет описывать книгу в виде: номер страницы, номер строки, номер столбца.

Объявим массив C с размерностью 3 и 6:

```
DIM C(3,6)
```

Это даст 3*6=18 индексируемых переменных:

```
C(1,1), C(1,2), ..., C(1,6)
C(2,1), C(2,2), ..., C(2,6)
C(3,1), C(3,2), ..., C(3,6)
```

Могут использоваться строковые массивы. Строки в таких массивах отличаются от скалярных тем, что имеют фиксированную длину, а присваивание им значения осуществляется с усечением справа или добавлением до полной длины пробелами. Имя строкового массива образуется добавлением справа к имени специального символа - $.

Допустим вам необходимо объявить массив A$ на 5 строк по 10 символов в каждой строке. Вы должны записать:

```
DIM A$(1,10)
```

Теперь вы можете обращаться как целиком к отдельной строке, так и к каждому символу в строке:

```
A$(1)=A$(1,1)A$(1,2) ... A$(1,10)
A$(2)=A$(2,1)A$(2,2) ... A$(2,10)
.....................................
A$(5)=A$(5,1)A$(5,2) ... A$(5,10)
```

Можно также рассматривать элемент строкового массива как массив символов. Пусть объявлен массив A$(2,7). Можно записать и так A$(2)(7). Следующая программа:

```
10 LET A$(2)="1234567890"
20 PRINT A$(2), A$(2,7)
```

даст "123456789 7".

Можно использовать также сечение массивов:

```
A$(2,4 TO 8)= A$(2)(4 TO 8)="45678"
```

Помните, что в строковых массивах все строки имеют фиксированную длину. Эту длину определяет последнее число размерности массива в операторе DIM. Если объявлен одномерный массив, то он определяет массив символов: DIM A$(10).

Глава 13

ЛОГИЧЕСКИЕ ОПЕРАЦИИ

Краткое содержание: AND, OR, NOT.

Если мы взглянем на описанную в третьей главе форму оператора IF:

```
IF условие THEN
```

то увидим, что "условие" описывается отношениями (=, <, >, <=, >=, <>), связывающими два числа или строки. Здесь можно также использовать логические операции AND(и), OR(или), NOT(не).

Некоторое выражение "И" истинно, если истинны все выражения, связанные отношением "И". Например:

```
IF A$="YES" AND X>0 THEN PRINT X
```

"X" будет напечатано только тогда, когда A="YES" и X>0.

Некоторое выражение "ИЛИ" истинно, если истинно хотя бы одно из выражений, связанных отношением "ИЛИ".

"НЕ" выражение истинно, если ложно само выражение и наоборот.

OR имеет низший приоритет, затем идет AND, затем NOT.

Условие "<>" обратно в логическом отношении условию "=", т.е.

```
A<>B то же, что и NOT A=B
NOT A<>B то же, что и A=B
```

Тем, кто боится сложностей, можно опустить следующие разделы.

1) Условия =, <, >, <=, >=, <> дают числовой результат 1 для истины и 0, если выражение ложно. Например, оператор PRINT 1=2, 1<>2 выведет 0 для "1=2" и 1 для "1<>2".
2) В операторе "IF условие THEN ..." само условие может быть числовым выражением. Если его значение после вычисления равно 0, то считается, что это ложь, если другому числу (включая и 1), то считается, что это истина.

Таким образом IF-оператор можно представить:

```
IF условие <>0 THEN ...
```

Операции AND, OR, NOT могут также использоваться и в числовых выражениях:

X AND Y имеет значение X, если Y<>0, и 0, если Y=0;
X OR Y имеет значение 1, если Y<>0, и X, если Y=0;
NOT Y имеет значение 0, если Y<>0, и 1, если Y=0.

Например:

```
10 INPUT A
20 INPUT B
30 PRINT (A AND A>B)+(B AND A<B)
40 GO TO 10
```

В каждой итерации будет выводиться одно из двух чисел - A или B.
Пример использования OR:

```
LET TOTAL PRICE=PRICE LESS TAX*(1.15 OR V$="ZERO RATEC")
```

В условном выражении можно также использовать символьные строки, но только с операцией AND:

X$ AND Y имеет значение X$, если Y<>0, и "", если Y=0,
где "" - пустая строка.

Выполните следующую программу, которая вводит две строки, а затем выводит их в алфавитном порядке:

```
10 INPUT "TYPE IN TWO STRINGS";A$,B$
20 IF A$>B$ THEN LET C$=A$: LET A$=B$: LET B$=C$
30 PRINT A$, ;("<" AND A$<B$)+("=" AND A$=B$)
40 PRINT " ";B$
50 GOTO 10
```

Глава 14

НАБОР СИМВОЛОВ

Краткое содержание: CODE, CHR$, POKE, PEEK, USR, BIN.

Буквы, цифры, знаки пунктуации, обозначаемые символами, образуют алфавит или набор символов, используемый компьютером. Отдельные символы, называемые знаками, образуют целые слова, например, PRINT, STOP и т.д.

Компьютер ZX SPECTRUM использует 256 символов с кодами от 0 до 255. Все они приведены в Приложении A. Для преобразования из символьной формы во внутреннюю и наоборот служат две функции CODE и CHR$.

CODE применяется к строке символов и возвращает код внутреннего представления первого символа в строке или 0, если строка пустая.

CHR$ применяется к числу и возвращает один символ, код которого представлен этим числом.

Следующая программа выводит весь отображаемый символьный набор:

```
10 FOR A=32 TO 255: PRINT CHR$ A: NEXT A
```

Все эти символы (кроме знаков фунта и "c" в кружочке) образуют код ASCII (American Standart Codes for Information Interchange).

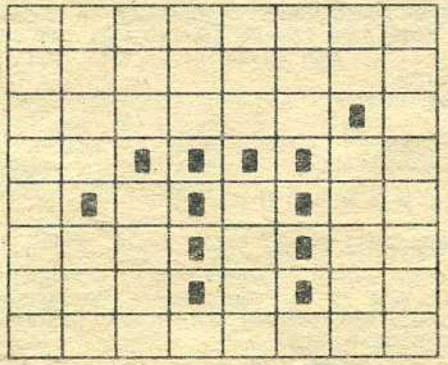
Следующие символы не входят в ASCII, но используются в ZX SPECTRUM. Первые из них - это 15 черно-белых значков, называемых графическими символами и используемых для изображения рисунков.

Их можно ввести с клавиатуры, используя так называемый графический режим. Если вы нажмете GRAPICS (CAPS SHIFT и 9), то курсор изменится на <G>. Теперь цифровые клавиши с 1 по 8 выдают графические символы, обозначенные на клавишах, а если при этом удерживать SHIFT - они будут выдавать инверсные символы, т.е. черное становится белым, а белое - черным.

Независимо от SHIFT клавиша с цифрой "9" обеспечивает вам возврат к обычному режиму (<L>-курсору), а клавиша "0" - функцию DELETE.

После графических символов на клавиатуре располагаются символы алфавита от A до U. Графические значения этих клавиш могут определяться пользователем, а затем использоваться в графическом режиме. Определение графики этих клавиш проиллюстрируем на примере определения символа греческого алфавита "пи".

1) Каждый символ представляется точками в матрице 8x8, поэтому мы вначале начертим диаграмму, приведенную на рисунке. Мы оставим по одной клетке по периметру символа для отделения его от соседних знаков.
2) Закрепим данный символ за клавишей "P", так чтобы при нажатии клавиши в графическом режиме выдавался символ "пи"



3) Запрограммируем это изображение. Каждый определяемый пользователем символ запоминается в памяти восемью знаками по одному на каждый ряд. Вы можете записать их, используя функцию BIN с обозначением цифрой 0 чистой точки и цифрой 1 - закрашенной точки.

```
BIN 00000000
BIN 00000000
BIN 00000010
BIN 00111100
BIN 01010100
BIN 00010100
BIN 00010100
BIN 00000000
```

Эти восемь десятичных чисел запоминаются в памяти в восьми ячейках, каждая из которых имеет свой адрес. Для нашего символа адрес первого из восьми байтов в группе будет USR "P". Второй байт

имеет адрес USR "P"+1 и т.д. до USR "P"+7.

USR - это функция преобразования строки символов в адрес первого байта в строке. Строковый аргумент может содержать единственный символ, который будет обозначать символ, определяемый пользователем. Имеются и другие применения функции USR с числовым аргументом, но об этом позже.

Поясним все сказанное программой:

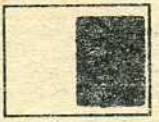
```
10 FOR N=0 TO 7
20 INPUT ROW: POKE USR "P"+N,ROW
30 NEXT N
```

Данная программа вводит 8 двоичных чисел, определяющих графику символа, закрепляемого за клавишей "P".

Оператор "POKE" записывает данные непосредственно в память, минуя обычный аппарат Бейсика. Обратным оператору "POKE" является оператор "PEEK", который служит для отображения содержимого области памяти. Об этом подробнее будет сказано в главе 24.

Графические символы

Символ	Код	Набор	Символ	Код	Набор
	128	<G> 8		143	<G> SHB
	129	<G> 1		142	<G> SH1
	130	<G> 2		141	<G> SH2
	131	<G> 3		140	<G> SH3
	132	<G> 4		139	<G> SH4

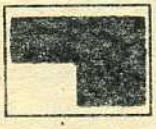


133 <G> 5

138 <G> SH5

134 <G> 6

137 <G> SH6



135 <G> 7

136 <G> SH7

Вернемся к знакам. Мы еще не поговорили о первых 33 знаках с кодами от 0 до 32. Это управляющие символы. Они не отображаются, вместо них на телевизоре отображается "?". Значение этих символов описано в приложении А.

Три символа с кодами 6,8 и 13 имеют специальное назначение при работе с телевизором.

CHR$ 6 печатает пробел, используемый как запятая в операторе PRINT.

```
PRINT 1;CHR$ 6;2
```

даст тот же результат, что и оператор

```
PRINT 1,2
```

Но это не совсем корректное использование. Вернее будет:

```
10 LET A$="1"+CHR$ 6+"2"
20 PRINT A$
```

CHR$ 8 – это символ забоя. Обеспечивает возврат на одну позицию назад. Оператор

```
PRINT "1234";CHR 8;"5"
```

даст строку: "1235".

CHR$ 13 – перевод строки. Продолжает вывод с новой строки. Также используются символы с кодами 16 и 23, но об этом поговорим в главах 15, 16.

Все символы расположенны в кодовой таблице в алфавитном порядке по возрастанию кодов, причем все прописные буквы расположены после заглавных, так что "a" следует после "Z".

```
CHR$ 34" ZOOLOGICAL GARDENS"
CHR$ 84"AARDVARK HUNTING"
"(PARENTHETICAL REMARK)"
"100"
"129,95 INC.VAT"
"AARVOGEL"
```

```
"AARDVARK"
"PRINT"
"ZOO"
"[INTERPOLATION]"
"AARDVARK"
"AASVOGEL"
"ZOO"
"ZOOLOGY"
```

Существует правило, по которому сортируются две строки. Сначала сравниваются первые символы. Если они различаются, то строка, содержащая символ с меньшим кодом, является "меньшей", а если равны, то выбираются для сравнения следующая пара символов, и т.д. до тех пор, пока не встретятся несовпадающие символы, либо пока одна из строк не закончится. Окончившаяся строка и будет "меньшей". В противном случае строки считаются равными.

Отношения =,<,>,<=,>=,<> применяются к строкам символов также как и к числам: знак "<" означает - "находится впереди в кодовой таблице", а знак ">" - "находится позади". Так что выражения

```
"AAMAN" < "AARDVARK"
"AARDVARK" > "AAMAN"
```

оба истинны.

Для иллюстрации всего сказанного приведем программу, которая вводит две строки, а затем выводит их в упорядоченном виде:

```
10 INPUT "TYPE IN TWO STRING", A$,B$
20 IF A$>B$ THEN LET C$=A$:LET A$=B$:LET B$=C$
30 PRINT A$;" ";
40 IF A$<B$ THEN PRINT "<";: GOTO 60
50 PRINT "="
60 PRINT " ";B$
70 GOTO 10
```

Следующая программа закрепляет определенные пользователем символы для игры в шахматы за клавишами:

```
P - за пешкой;   (POWN)
R - за ладьей;   (ROCK)
N - за конем;    (KNIGHT)
B - за слоном;   (BISHOP)
K - за королем;  (KING)
Q - за ферзем;   (QUEEN).
```

```
  5 LET B=BIN 01111100:LET S= BIN 00111000:LET D=BIN 00010000
 10 FOR N=1 TO 6 : READ P$: REM 6 PIECES
 20 FOR N=1 TO 7 :REM READ PIECE INTO 8 BUTES
 30 READ A: POKE USR$ P$+1,A
 40 NEXT F
 50 NEXT N
100 REM BISHOP
110 DATA "B", 0, D, BIN 00101000, BIN 01000100
120 DATA BIN 01101100,C,B,0
130 REM KING
140 DATA "K",0,D,C,D
150 DATA C, BIN 01000100,C,0
160 REM ROCK
170 DATA  "R",0, BIN 01010100
```

```
180 DATA C,B,B,C
190 REM QUEEN
200 DATA "Q" ,0, BIN 01010100, BIN 00101000,D
210 DATA BIN 01101100,B,B,C
220 REM POWN
230 DATA "P",C,C,D,C
240 DATA C,D,C,C
250 REM KNIGHT
260 DATA "N",C,D,C, BIN 01111000
270 DATA BIN 00011000,C,B,C
```

Глава 15

ДОПОЛНИТЕЛЬНЫЕ СВЕДЕНИЯ ОБ ОПЕРАТОРАХ "PRINT" И "INPUT"

Краткое содержание: CLS, PRINT - параметры: их отсутствие вообще, выражение (числовое или строковое); TAB - числовое выражение, AT - числовое выражение; PRINT - разделители:",", ";", "'"; INPUT - параметры;LINE - строчная переменная; SCREEN$ - свертка.

Выражение, значение которого используется в операторе PRINT, называется PRINT-выражением. Оно разделяется запятыми или точкой с запятой, называемыми PRINT-разделителями.

В операторе PRINT возможно отсутствие каких-либо параметров. В этом случае ставятся две запятые подряд.

```
PRINT AT 11,16;"*"
```

печатает "*" в середине экрана.

AT 'строка', 'столбец'

этот параметр перемещает позицию вывода в место, определяемое номером строки и столбца. Номер строки имеет значения от 0 до 21, а номер столбца - от 0 до 31.

Действие оператора SCREEN$ обратно действию оператора PRINT AT. Он использует те же параметры, но их значения заключаются в скобки.

```
PRINT SCREEN$ (11,16)
```

Мы возвратимся на "*", выведенную предыдущим оператором.

Как символы вывода могут использоваться алфавитно-цифровые символы, специальные символы, пробелы.

Линии, нарисованные с помощью операторов PLOT, DRAW, CIRCLE и определяемые пользователем с помощью графических символов, возвращаются как пустая строка. Однако имеется и другое применение, когда функция OVER используется для построения комбинированных знаков.

TAB столбец

Этот параметр перемещает позицию вывода в заданный столбец в той же строке или переходит на новую строку, если столбец был последним. Заметим, что "TAB 33" равнозначно "TAB 1". К примеру:

```
PRINT TAB 30,1; TAB12; "CONTENTS"; AT 3,1; "CHAPTER";
```

```
TAB 24; "PAGE"
```

выведет на экране две первые страницы книги.

Рассмотрим пример:

```
10 FOR N=0 TO 20
20 PRINT TAB 8*N;N
30 NEXT N
```

Мы увидим, что значения параметра TAB получаются в результате вычитания по модулю из 32. Более наглядный пример получается при замене в 20-й строке "8" на "6". Несколько замечаний:

1) В рассмотренных примерах в качестве ограничителя использовалась ";". Можно использовать "," (или пробел, как конец оператора). При этом нужно следить за тем, чтобы позиция вывода не переместилась в часть экрана, предназначенную для вывода сообщений.
2) Мы не можем использовать две нижние строки экрана (22 и 23), так как они используются оператором INPUT при получении данных, т.е. последняя используемая строка - 21.
3) Мы можем использовать параметр AT для установки позиции вывода в то место, где уже имеется выведенная информация. При этом новый символ уничтожает старый.

Еще одним оператором, используемым с оператором PRINT, является CLS. Он производит очистку экрана подобно операторам CLEAR и RUN. При заполнении экрана происходит его свертка. В этом можно убедиться, проделав:

```
CLS: FOR N=1 TO 22: PRINT N: NEXT N
```

и далее "PRINT 99" некоторое количество раз.

При выводе текста на экран происходит останов вывода для просмотра. Чтобы убедиться в этом, выполним:

```
CLS: FOR N=1 TO 100: PRINT N: NEXT N
```

Когда экран заполнится, вывод остановится и в нижней части экрана появится запрос "SCROLL ?". После просмотра нажмите "Y" (да) - и вывод продолжится. Возможен отрицательный ответ "N" (нет), STOP (SYMBOL SHIFT и "A") или SPACE (BREAK). Компьютер остановит программу и выдаст сообщение "D BREAK-CONT REPEATS?".

INPUT-операторы используются для ввода различных значений. Например:

```
INPUT "HOW OLD ARE YOU?".AGE
```

Компьютер выведет на экран (в нижней его части) вопрос, в ответ на который вы должны ввести свой возраст. Фактически INPUT содержит тоже параметры, что и PRINT. Так "HOW OLD ARE YOU?" и AGE являются параметрами INPUT. Однако существуют и значительные различия.

Первое: параметры INPUT - переменные, которые вы вводите сами. Ввод может начинаться с буквы, которая является вводимой переменной.

Второе: Вы можете вводить значение переменной как часть титра, заключив ее для этого в скобки.Пример:

```
LET MY AGE=INT(RND*100):INPUT("I AM", MY AGE;".");
"HOW OLD ARE YOU?", YOUR AGE
```

Значение "MY AGE" выдает компьютер, значение "YOUR AGE" вы вводи-

те сами. По мере выдачи операторов ввода происходит свертка экрана. Рассмотрим пример использования AT в INPUT-операторе:

```
10 INPUT "THIS IS LINE 1";A$:AT 0,0;"THIS IS LINE 0";A$; AT
   2,0;"THIS IS LINE 2";A$;AT 1,0;"THIS IS STILL LINE 1";A$
```

Когда "THIS IS LINE 2" будет выведено, нижние строки будут сдвигаться вверх. Выполним:

```
10 FOR N=0 TO 19: PRINT AT N,0;N;NEXT N
20 INPUT AT 0,0;A$;AT 1,0;A$;AT 2,0;A$;AT 3,0;A$;AT 4,0;A$;
   AT 5,0;A$;
```

Когда информация начинает смещаться в область действия операторов PRINT, произойдет свертка экрана. Еще одним параметром оператора INPUT является LINE. Он предназначен для ввода строчных переменных. Рассмотрим пример:

```
INPUT LINE A$
```

Если ввести какую-либо строчную переменную, то ее значение будет присвоено A$. Заметим, что мы не можем использовать параметр LINE для числовых переменных.

Управляющие символы CHR$22 и CHR$23 выполняют функции, подобные TAB и AT. Их преимущество состоит в том, что можно задавать более, чем два значения, а для TAB и AT это невозможно. Эти управляющие символы обрабатываются как числа.

Аналогом AT является управляющий символ CHR$22: первое значение определяет строку, второе - столбец.

```
PRINT CHR$22+CHR$1+CHR$C -
```

то же, что и PRINT AT 1,C. Как значения параметров рассматриваются CHR$1 и CHR$C (CHR$22 не учитывается).

Аналогом TAB является управляющий символ CHR$23. Значения задаваемых им параметров лежат в пределах от 0 до 65535.

```
PRINT CHR$23+CHR$A+CHR$ -
```

то же, что и PRINT TAB A+256*B.

Вы можете использовать POKE для остановки компьютера, запрашивающего свертку. Выполните:

```
POKE 23692,255.
```

Компьютер выдаст 255 значений, прежде чем запросит свертку. Так в примере:

```
10 FOR N=0 TO 10000
20 PRINT N: POKE 23692,255
30 NEXT N
```

на экран будут выведены часы.

Запустим следующую программу, проверяющую таблицу умножения:

```
 10 LET A$= " "
 20 LET A=INT(RND*12)+1:LET B=INT(RND*12)+1
 30 INPUT(A$);"WHAT IS";(A);"*";(B);"?";C
100 IF C=A*B THEN LET A$="RIGHT": GOTO 20
110 LET A$="WRONG. TRY AGAIN.": GOTO 30
```

Можно несколько изменить программу так, чтобы не зная правильного ответа можно было узнать его. К примеру, компьютер спрашивает сколько будет 2*3. Не зная ответа, вы вводите 2*3 и получаете

его. Для этого измените "c" в строке 30 на "c$", в строке 100 на "VAL c$" и дополнительно введите строку

```
40 IF c$ <> STR$ VAL c$ THEN LET n$= "TYPE
   IT PROPERLY AS NUMBER.": GOTO
```

Для исключения подсказки поменяйте "c$" в строке 30 на "LINE c$".

Глава 16

ЦВЕТА

Краткое содержание: INK, PAPER, FLASH, BRIGHT, INVERSE, OVER, BORDER.

Выполним следующую программу:

```
10 FOR m=0 TO 1: BRIGHT m
20 FOR n=1 TO 10
30 FOR c=0 TO 7
40 PAPER c: PRINT" ": REM 4 COLOURED SPACES
50 NEXT c: NEXT n: NEXT m
60 FOR m=0 TO 1: BRIGHT m: PAPER 7
70 FOR c=0 TO 3
80 INK c: PRINT c;" ";
90 NEXT c: PAPER 0
100 FOR c=4 TO 7
110 INK c: PRINT c;" ";
120 NEXT c: NEXT m
130 PAPER 7: INK 0: BRIGHT 0
```

Программа демонстрирует возможности вывода компьютера ZX-SPECTRUM на цветной телевизор восьми цветов (включая черный и белый) и двух уровней яркости. Если телевизор черно-белый, вы увидите различные градации серого цвета.

Ниже дана кодировка цветов:

0 - черный;
1 - синий;
2 - красный;
3 - фиолетовый;
4 - зеленый;
5 - голубой;
6 - желтый;
7 - белый.

Для черно-белого телевизора этот ряд представляет собой последовательность перехода серых полутонов от черного до белого.

Для использования цвета участки строения графического экрана. Он состоит из 768 позиций (24 строки на 32 знакоместа), каждая из которых представляет собой матрицу 8 на 8 пикселей. Вспомним из главы 14:

0 - белая точка;
1 - черная точка.

Позиция символа (знакоместо) также рассматривается с этой точки зрения: INK - цвет тона, PAPER - цвет фона, т.е. знакоместо состоит из INK и PAPER (рисунок на бумаге). Можно говорить о INK и PAPER обычной и повышенной яркости, а также о мерцающих символах и мерцающих знакоместах. Все это имеет кодировку:

1) Для знакоместа (8 на 8 пикселей) форму символа определяют чистые и закрашенные точки (0 и 1), цвета тона и фона определяются PAPER и INK.
2) Цвета тона и фона кодируются от 0 до 7 каждый.
3) Яркость: 0 - обычная, 1 - повышенная.
4) Мерцание: 0 - постоянно, 1 - отсутствует.

Заметим, что для одного знакоместа в 64 пикселя, мы не можем установить более одного цвета для фона и одного цвета для тона. Это же относится и к яркости, и к мерцанию. Цвет, яркость и мерцание задаются для знакоместа (а не для отдельного пикселя) и являются его атрибутами. Для изменения этих атрибутов предназначены операторы: INK, PAPER, BRIGHT, FLASH.

Выполним

```
PAPER 5
```

теперь вывод будет осуществляться на голубой фон (т.к. 5 - код голубого цвета).

Форматы операторов:

```
PAPER - число от 0 до 7;
INK - число от 0 до 7;
BRIGHT - число 0 или 1 (0-выкл.,1-вкл.);
FLASH - число 0 или 1 (0-выкл.,1-вкл.).
```

Отметим, что использование чисел, больших чем указывалось выше, допустимо, но дает другой эффект. К примеру, "8" может использоваться во всех четырех операторах как средство, позволяющее определить значения ранее установленных атрибутов. Так,

```
PAPER 8
```

не установит цвет фона (так как такого цвета нет), а поможет выяснить значение предыдущего PAPER. Операторы INK 8, BRIGHT 8, FLASH 8 выдадут значения соответствующих атрибутов.

"9" может использоваться только для INK и PAPER как средство "контраст". Цвета INK и PAPER, которые вы используете, должны быть контрастны друг другу. Так, к белому цвету подходят темные тона: черный, синий, красный, фиолетовый; к черному цвету подходят светлые тона: зеленый, голубой, желтый, белый.

Выполним:

```
INK 9: FOR C=0 TO 7: PAPER C: PRINT 8: NEXT C
```

Можно запустить программу, выдающую на экран дисплея цветные полосы

```
INK 9:PAPER 5:PRINT AT 0,0;:FOR N=1 TO 1000:PRINT N;:NEXT N
```

Цвет фона будет контрастен цвету тона в любой выводимой позиции. Цветной телевизор построен на следующем принципе: человеческий глаз способен воспринимать только три первичных цвета - синий, красный и зеленый. Другие цвета образуются из их сочетаний. К примеру, фиолетовый цвет образуется как комбинация синего и красного (код фиолетового цвета - "3" является суммой кодов синего - "1" и красного - "2").

Видеть все восемь цветов на одном участке экрана невозможно, так как это будет темное пятно. Но там, где цвета частично накладываются друг на друга, мы увидим гамму. В качестве примера выполним программу (отметим, что INK получена с использованием

SHIFT M в G-режиме):

```
 10 BORDER 0:PAPER 0: INK 7: CLS
 20 FOR A=1 TO 6
 30 PRINT TAB 6; INK 1;"()()()()()":REM 18 INK SQUARES
 40 NEXT A
 50 LET DATA LINE=200
 60 GOSUB 1000
 70 LET DATA LINE=210
 80 GOSUB 1000
 90 STOP
200 DATA 2,3,7,5,4
210 DATA 2,2,6,4,4
1000 FOR A=1 TO 6
1010 RESTORE DATA LINE
1020 FOR B=1 TO 5
1030 READ C: PRINT INK C; "()()()";: REM 6 INK SQUARES
1040 NEXT B: PRINT : NEXT A
1050 RETURN
```

Существует функция ATTR, позволяющая определить какие атрибуты были заданы для позиции экрана. Это сложная функция будет рассмотрена в конце главы.

Операторы INVERSE и OVER не управляют атрибутами, но тем не менее определяют способ вывода на экран. В этих операторах используются значения параметров "0" и "1". Если вы дадите INVERSE 1, то выводимый символ изменит свою обычную форму (вывод будет осуществляться в негативном изображении). В обычном виде мы пишем черным по белому, в инверсном – белым по черному.

Оператор OVER 1 устанавливает режим расширенного вывода. В обычном режиме при выводе символа на знакоместо там стирается все выведенное ранее. При расширенном выводе – можно накладывать изображения друг на друга. Это позволяет выводить выводить составные символы, например, стилизованные шрифты. Программа для вывода готического шрифта:

```
10 OVER 1
20 FOR N=1 TO 32
30 PRINT "O"; CHR$8;""""";
40 NEXT N
```

Отметим, что управляющий символ CHR$8 определяет одно место.

Возможен еще один способ использования INK и PAPER. Их можно вводить как параметры PRINT. Точно также можно использовать и другие операторы, рассмотренные в этой главе, учитывая при этом, что их действие распространяется только до конца PRINT. В примере

```
PRINT PAPER 6; "X":: PRINT"Y"
```

только "X" будет выведен на желтый фон.

INK и другие операторы не действуют в нижней части экрана, предназначенной для ввода команд и INPUT-данных. Для изменения цветов в этой части экрана служит оператор

```
BORDER цвет
```

Кодировка цветов прежняя. Возможны мерцание и повышенная яркость. Для этого используйте соответствующие параметры в INPUT

(наподобие как в PRINT). Эти параметры действуют до конца оператора или до тех пор, пока запрашиваемые данные не будут введены. Выполним:

```
INPUT FLASH 1; INK 1; "WHAT IS YOUR NUMBER ?";N
```

Возможно изменение цветов с помощью управляющих символов, подобных управляющим символам для AT и TAB, описанным в главе 15.

```
CHR$16 - INK
CHR$17 - PAPER
CHR$18 - FLASH
CHR$19 - BRIGHT
CHR$20 - INVERSE
CHR$21 - OVER
```

Так, PRINT CHR$16+CHR$9 - то же, что и PRINT INK 9.

Можно пользоваться либо управляющими символами, либо операторами. Их можно ставить как после номера строки, так и в конце строки. Для удобства можно пользоваться цифрами в расширенном режиме (E). Цифры от 0 до 7 устанавливают цвет INK, если CAPS SHIFT нажата , и цвет PAPER, если не нажата. Если нажать цифру в E-режиме, то будут выведены: CHR$17 и CHR$код цвета. Если в это время была нажата CAPS SHIFT, то будут выведены CHR$16 и CHR$код цвета.

Если вы хотите уничтожить вводимое, то нажмите DELETE два раза. После первого раза появится знак вопроса или что-нибудь еще. Не пугайтесь и нажмите DELETE еще раз. Управление курсором также не работает обычным образом до тех пор, пока курсор не выйдет к предыдущему управляющему символу.

Действие в расширенном режиме (E):

8 дает CHR$19 и CHR$0 - нормальная яркость;
9 дает CHR$19 и CHR$1 - повышенная яркость;
CAPS SHIFT с 8 дает CHR$18 и CHR$0 - не мерцающее;
CAPS SHIFT с 9 дает CHR$18 и CHR$1 - мерцающее.

В (L)-режиме:

CAPS SHIFT с 3 дает CHR$20 и CHR$0 - обычный вывод;
CAPS SHIFT с 4 дает CHR$20 и CHR$1 - инверсный (негативный) вывод.

Функция ATTR имеет следующий формат:

```
ATTR (строка, столбец)
```

Значения двух параметров функции подобны значениям параметров в AT. В результате выполнения будут выведены значения атрибутов для соответствующей позиции экрана. Выводимый результат - это число, представляющее собой сумму четырех чисел:

1) 128 - если знакоместо мерцающее, 0 - если обычное;
2) 64 - если повышенная яркость, 0 - если обычная;
3) 8* код цвета фона;
4) код цвета тона.

Пример: знакоместо мерцающее, обычной яркости, желтый фон, синий тон

```
128+0+(8*6)+1=177
```

Проверим это, выполнив:

```
PRINT AT 0,0;FLASH 1;PAPER 6; INK 1; " "; ATTR(0,0)
```

Упражнения:

1) PRINT "B"; CHR$8;OVER 1;"/";

Здесь "/" перечеркнет "B". Этим способом можно выводить слова из комбинированных знаков ZX-SPECTRUM. Два фона или два тона дают фон. Один из них дает тон. Это интересное свойство. Если вы повторите вывод одного символа дважды, то он не будет отображен. Так, если ввести:

PRINT CHR$8;OVER 1;"/",

дополнительно к введенной ранее команде, то в результате мы увидим "B", не перечеркнутое "/". Так ли это?

2) Выполним:

PAPER 0; INK 0.

Действуют ли эти операторы в нижней части экрана? Что мы увидим, если добавить BORDER 0?

3) Выполним программу:

```
10 POKE 22527+RND*704,RND*127
20 GOTO 10
```

Результатом действия программы явится смена цветов знакомест, распределенных по экрану случайным образом. Возможно вы увидите смену цветов на диагональных ступенях. Это является следствием того, что мы пользуемся квазислучайным распределением, которое лишь приближенно воспроизводит случайное распределение.

4) Введем вручную или с помощью оператора LOAD программу для вывода шахматных фигур из главы 14 и введем программу разбивки экрана под шахматную доску:

```
  5 REM DRAW BLANKBOARD
 10 LET BB=1: LET BW=2: REM RED AND BLUE FOR BOARD
 15 PAPER BW:INK BB: CLS
 20 PLOT 79,128: REM BORDER
 30 DRAW 65,0: DRAW 0,-65
 40 DRAW -65,0: DRAW 0,65
 50 PAPER BB
 60 REM BOARD
 70 FOR N=0 TO 3: FOR M=0 TO 3
 80 PRINT AT 6+2*N,11+2*M; " "
 90 PRINT AT 7+2*N,10+2*M; " "
100 NEXT M: NEXT N
110 PAPER 8
120 LET PW=6: LET PB=5: REM COLOURS OF WHITE AND BLACK
    PIECES
200 DIM B$(8,8): REM POSITIONS OF PIECES
205 REM SET AT INITIAL POSITION
210 LET B$(1)="RNBQKBNR": REM LITTLE LINE 240
220 LET B$(2)="PPPPPPPP": REM LITTLE LINE 230
230 LET B$(7)="PPPPPPPP": REM BIG LINE 220
240 LET B$(8)="RNBQKBNR": REM BIG LINE 210
300 REM DISPLAY BOARD
310 FOR N=1 TO 8: FOR M=1 TO 8
320 LET BC=CODE B$(N,M): INK PW
325 LET BC=CODE " " THEN GOTO 350: REM SPACE
330 IF BC>CODE "Z" THEN INK PB:LET BC=BC-32:REM LOWER
```

```
    CASE FOR BLACK
340 LET BC=BC+79: REM CONVERT TO GRAPHICS
350 PRINT AT 5+N,9+M;CHR$BC
360 NEXT M: NEXT N
400 PAPER 7: INK 0
```

Глава 17

ГРАФИКА

Краткое содержание: PLOT, DRAW, CIRCLE, POINT.

В этой главе описываются возможности компьютера ZX SPECTRUM по отображению графической информации. Экран компьютера содержит 22 строки по 32 символа в каждой, что составляет 22x32=704 символьные позиции. Как вы уже поняли из главы 16, каждая символьная позиция представляется квадратом 8x8 точек, называемых пикселями.

Пиксель задается двумя числами – его координатами. Первое число задает координату X, т.е. удаление (в пикселях) от нижней границы экрана, второе – координату Y – удаление от нижней границы экрана. Координаты записываются в скобках. Так, (0,0), (255,0), (255,175) и (0,175) задают соответственно нижний левый, нижний правый, верхний правый и верхний левый углы экрана.

Оператор PLOT X,Y вызывает зачерчивание закрашивающим цветом (INK) пикселя с указанными координатами. Например, программа:

```
10 PLOT INT(RND*256),INT(RND*176)
20 INPUT A$
30 GOTO 10
```

будет высвечивать некоторый случайный пиксель при каждом нажатии ENTER.

Есть и более интересные программы. Например, следующая программа вычерчивает график функции SIN X для X в интервале от 0 до 2*PI:

```
10 FOR N=0 TO 255
20 PLOT N,86+80*SIN(N/128*PI)
30 NEXT N
```

или программа:

```
10 FOR N=0 TO 255
20 PLOT N,80*SQR(N/64)
30 NEXT N
```

которая чертит график SQR X (часть параболы) в интервале от 0 до 4.

Помните, что координаты пикселей отличаются от адресации строк и позиций в подкоманде AT.

Пользуйтесь диаграммой из главы 15.

При построении изображений вам будут полезны операторы DRAW и CIRCLE.

Оператор DRAW чертит линию, заданную в форме:

```
DRAW X,Y.
```

Началом линии является пиксель, на котором завершилось выполнение одного из операторов PLOT, DRAW или CIRCLE. (Этот пиксель называется текущей PLOT-позицией. Операторы RUN, CLEAR, CLS и NEW

устанавливают ее в левый нижний угол экрана). Таким образом, оператор DRAW задает длину и направление вычерчивания линии, но не задает ее начальную точку.

Поэкспериментируйте с такими командами:

```
PLOT 0,100: DRAW 80,-35
PLOT 90,150: DRAW 80,-35
```

Чертить можно в цвете, но при этом нужно иметь ввиду, что цвет устанавливается для целой символьной позиции и не может быть задан для отдельного пикселя. Следующая программа демонстрирует это:

```
10 BORDER 0: PAPER 0: INK 7: CLS: REM BLACK OUT SCREEN
20 LET X1=0: LET Y1=0: REM START OF LINE
30 LET C=1: REM FOR INK COLOUR, STARTING BLUE
40 LET X2=INT(RND*256): LET Y2=INT(RND*176): REM RANDOM
   FINISH OF LINE
50 DRAW INK C; X2-X1,Y2-Y1
60 LET X1=X2: LET Y1=Y2: REM NEXT LINE STARTS WHERE LINE
   ONE FINISHED
70 LET C=C+1: IF C=8 THEN LET C=1: REM NEW COLOUR
80 GOTO 40
```

Вы можете использовать в операторах PLOT и DRAW управляющие символы PAPER, INK, FLASH, BRIGHT, INVERSE и OVER также, как и в операторах PRINT и INPUT. Управляющие символы записываются между ключевым словом и координатами и оканчиваются запятой или точкой с запятой (смотри строку 50).

При помощи DRAW можно также вычертить отрезок дуги, используя для этого дополнительное число, задающее угол (в радианах) этой дуги:

```
DRAW X,Y,A
```

Если "A" положительное число, то дуга вычерчивается влево, а если отрицательное – то вправо. При "A", равном 2*PI, вычерчивается полная окружность. Например:

```
10 PLOT 100,100: DRAW 50,50,PI
```

вычертит полуокружность с начальной точкой (100,100) и с конечной точкой (150,150). Вычерчивание начнется в направлении на юго-восток и закончится в направлении на северо-запад.

Оператор CIRCLE вычерчивает полный круг, задаваемый координатами его центра и радиусом:

```
CIRCLE X,Y,радиус
```

Как и в операторах PLOT и DRAW, вы можете указать в этом операторе различные цвета.

Функция POINT возвращает характеристики цвета заданного пикселя. Например, строка программы:

```
CLS:PRINT POINT(0,0):PLOT (0,0): PRINT POINT(0,0)
```

выведет: PAPER 7:INK 0.

Допускается также задавать управляющие символы INVERSE и OVER в операторе PLOT. По умолчанию они предполагаются равными 0 (отключено), но вы можете задать их и равными 1, при этом:

PLOT INVERSE 1 – устанавливает для заданного пикселя цвет

фона;

PLOT OVER 1 - изменяет цвет пикселя на противоположный. Закрашивающий цвет становится цветом фона и наоборот;

PLOT INVERSE 1; OVER 1; - сохраняет цвет пикселя без изменения, но меняет текущую PLOT-позицию

Другой пример использования OVER с записью черным по белому:

```
PLOT 0,0: DRAW OVER 1;255,175
```

вычерчивает линию по диагонали.

Теперь попробуйте:

```
PLOT 0,0;DRAW INVERSE 1;256,175
```

и перечертите ее командой

```
DRAW OVER 1;-250,-175
```

Это не изменит картины, так как при черчении как вперед, так и назад используются одни и те же пиксели.

Имеется способ получения обычных цветов в одном квадрате с использованием определяемых пользователем символов. Выполните эту программу:

```
1000 FOR N=0 TO 6 STEP 2
1010 POKE USR "A"+N, BIN 01010101: POKE USR "A"+N+1,
     BIN 10101010
1020 NEXT N
```

Она задает определяемый пользователем символ для шахматной доски, который закрепляется за клавишей "A". Для символа используется красный закрашивающий цвет и желтый цвет фона, но на экране этот символ будет казаться оранжевым.

Еще один пример - программа, которая строит график некоторой функции. На первый ее вопрос вы отвечаете числом "N", задающим область значений аргумента (т.е. график будет строиться для значений аргумента в диапазоне от -N до +N). Второй ответ - это выражение в виде символьной строки, задающей функцию, использующую "X" в качестве аргумента:

```
10 PLOT 0,87: DRAW 255,0
20 PLOT 127,0: DRAW 0,175
30 INPUT S,E$
40 FOR F=2 TO 255
50 LET X=(F-128)*S/128: LET Y=VAL E$
60 IF ABS Y>87 THEN LET T=0: GOTO 100
70 IF NOT T THEN PLOT F,Y+88: LET T=1: GOTO 100
80 DRAW 1,Y-OLDY
100 LET OLDY=INT(Y+5)
110 NEXT F
```

Выполните ее, введя 10 для числа "N" и "10*TAN X" для функции. Будет вычерчен график tg X при X, изменяющемся от -10 до +10.

Глава 18

УКАЗАНИЯ

Краткое содержание: PAUSE, INKEY$, PEEK.

Если вы решили задержать выполнение программы на некоторое время, то вам следует использовать оператор PAUSE N, который останавливает выполнение программы и отображает картину в течение "N" телевизионных кадров (50 кадров в секунду в Европе или 60 кадров в секунду в Америке). "N" может принимать значения вплоть до 65535, что составляет 22 минуты. Если N=0, то это означает, что оператор PAUSE не имеет ограничений по времени.

Выполнение программы всегда может быть возобновлено до окончания времени, определенного в операторе PAUSE, нажатием любой клавиши (надо помнить, что CAPS SHIFT будет вызывать прерывание).

Пример программы моделирования секундной стрелки часов:

```
 10 REM FIRST WE DRAW THE CLOCK FASE
 20 FOR N=1 TO 12
 30 PRINT AT 10-10*COS(N/6*PI),16+12*SIN(N/6*PI);N
 40 NEXT N
 50 REM NOW WE START THE CLOCK
 60 FOR T=0 TO 200000: REM THIS IS THE TIME IN SECONDS
 70 LET A=T/30*PI: REM A IS THE ANGLE OF THE SECOND
    HAND IN RADIANS
 80 LET SX=80*SIN A: LET SY=80*COS A
200 PLOT 128,88: DRAW OVER 1;SX,SY: REM DRAW SECOND HAND
210 PAUSE 42
220 PLOT 128,88: DRAW OVER 1;SX,SY: REM ERASE SECOND HAND
400 NEXT T
```

Эти часы остановятся, проработав приблизительно 55,5 часов, что задается в операторе с номером 60. Оператор 210 производит отсчет времени. Казалось бы здесь должен быть оператор PAUSE 50 (Европа) для точного отсчета одной секунды, но тогда бы мы не учли время, затрачиваемое на выполнение других операторов программы. Рассматриваемый вариант часов обеспечивает 2%-точность или иными словами уход на полчаса в день.

Возможны и более точные способы измерения времени. Для этого нужно использовать содержимое специальных областей памяти. В этом случае данные из памяти могут быть вызваны с помощью функции PEEK, подробно рассмотренной в главе 25. Здесь же в качестве примера рассмотрим выражение:

```
(65536*PEEK 23674+256*PEEK 23673 +PEEK 23672)/50
```

Оно дает количество секунд, прошедших с тех пор, как компьютер был включен (вплоть до 3-х суток и 21-го часа). Ниже приводится модифицированная программа моделирования часов:

```
 10 REM FIRST WE DRAW THE CLOCK FASE
 20 FOR X=1 TO 12
 30 PRINT AT 10-10*COS(N/6*PI),16+12*SIN(N/6*PI);N
 40 NEXT X
 50 DEF FN T()=INT((65536*PEEK 23674+256*PEEK 23673+PEEK
    23672)/50): REM NUMBER OF SECOND SINC START
100 REM NOW WE START THE CLOCK
110 LET T1=FN T()
120 LET A=T1/30*PI: REM A IS THE ANGLE OF THE SECOND HAND
    IN RADIANS
130 LET SX=72*SIN A: LET SY=72*COS A
```

```
140 PLOT 131,91: DRAW OVER 1; SX,SY: REM DRAW HAND
200 LET T=FN T()
210 IF T<=T1 THEN GOTO 200: REM WAIT UNTIL TIME FOR NEXT HAND
220 PLOT 131,91: DRAW OVER 1; SX,SY: REM RUB OUT OLD HAND
230 LET T1=T: GOTO 120
```

Эти часы обеспечивают точность 0.01% или уход на 10 секунд в день. Однако это возможно при условии, что вы не использовали оператор NEW, ввод/вывод на магнитофон и принтер. Все эти операции увеличивают погрешность.

Числа PEEK 23674, PEEK 23673 и PEEK 23672 выделяют адреса ячеек памяти компьютера, используемых для подсчета 1/50 долей секунды. В каждой из ячеек подсчитывается сумма от 0 до 255. После достижения величины 255 в любой из ячеек она сбрасывается в 0. Первой начинает отсчитывать ячейка PEEK 23672. Каждые 1/50 сек ее содержимое увеличивается на 1. Когда в ячейке накопится величина, равная 255, она сбрасывается в 0, а содержимое ячейки PEEK 23763 увеличивается на 1. Через каждые 256/50 сек содержимое этой ячейки переходит через 255 в 0, а содержимое ячейки PEEK 23674 увеличивается на 1. При значениях 0 для ячейки PEEK 23674 и 255 для ячеек PEEK 23673 и PEEK 23672 (этот момент наступит через 21 минуту) наше выражение примет значение:

```
(655536*0+256*255+255)/50=1310.7
```

Здесь имеется скрытая опасность. Через следующие 1/50 сек ячейки будут содержать соответственно значения: 1, 0, 0. Пока производится вычисление выражения, компьютер может оценить значение ячейки PEEK 23674 как 0 до завершения циклического переноса. В результате получим:

```
(65536*0+256*0+0)/50=0,
```

что безнадежно неверно.

Простое правило позволяет решить эту проблему: "Следует вычислять выражение дважды в некоторой последовательности и использовать сохраненный ответ".

Пример:

```
10 DEF FN M(X,Y)=(X+Y+ABS(X-Y))/2: REM THE LARGER OF X AND Y
20 DEF FN U()=(65536*PEEK 23674+256*PEEK 23673+PEEK
   23672)/50: REM TIME, MAY BE WRONG
30 DEF FN T()=FN M(FN U(),FN U()): REM TIME RIGHT
```

Вы можете изменять значения числовых счетчиков так, чтобы получать реальное время того момента, когда был включен компьютер. Например, надо установить 10 часов вечера. Вы посчитали, что 12*60*50=180000 50-х долей сек, и значит:

```
180000=65536*27+256*119+64.
```

Для присвоения трем ячейкам значений 27, 119 и 64 необходимо выполнить

```
POKE 23674,27:POKE 23673,119:POKE 23672,64
```

Функция INKEY$ (без аргументов) считывает с клавиатуры, если вы нажали некоторую клавишу (или SHIFT и какую-нибудь клавишу). Результатом будет символ, который имеет эта клавиша в режиме курсора <L>, или пустая строка.

Выполните программу, которая использует эту функцию:

```
10 IF INKEY$ <> " " THEN GOTO 10
20 IF INKEY$ = " " THEN GOTO 20
30 PRINT INKEY$;
40 GOTO 10
```

Помните, что функция INKEY$ не будет подобно INPUT ждать вашего ввода. Если вы не выполните ввод, то считайте, что вы упустили свой шанс.

Глава 19

ПРОГРАММИРОВАНИЕ ЗВУКОВ

Краткое содержание: BEEP.

ZX-SPECTRUM может воспроизводить звуки при помощи оператора BEEP:

BEEP продолжительность, высота звука,

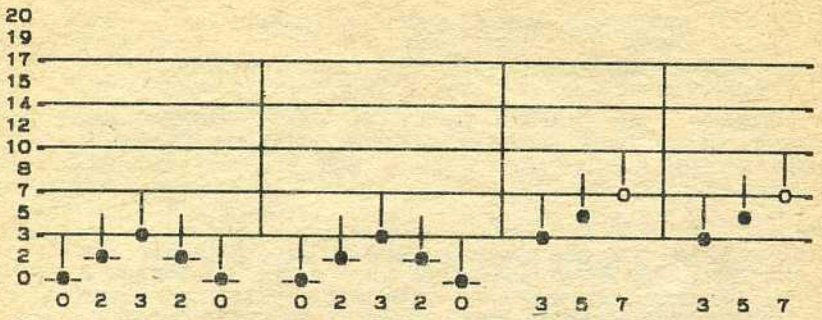
где "продолжительность" и "высота" звука – некоторые выражения. Продолжительность задается в секундах, а высота в полутонах от основного тона До: при положительных числах – выше ноты До, а при отрицательных – ниже ноты До. На диаграмме приведены все значения нот одной октавы:

	A# BB -2		D# DB 1	D# EB 3		F# GB 5	G# AB 6	A# BB 10	C# DB 13	D# EB 15	
Ля -3	Си -1	До 0	Ре 2	Ми 4	Фа 5	Соль 7	Ля 9	Си 11	До 12	Ре 14	Ми 16
		C	D	E	F	G	A	B	C		

Для получения более высоких или более низких нот, вы должны прибавить или отнять 12 для каждой октавы вверх или вниз. Например:

```
10 PRINT "FRERE GUSTAV"
20 BEEP 1,0: BEEP 1,2: BEEP .5,3:BEEP .5,2: BEEP 1,0
30 BEEP 1,0: BEEP 1,2: BEEP . ,3: BEEP .5,2: BEEP 1,0
40 BEEP 1,3: BEEP 1,5: BEEP 2,7
50 BEEP 1,3: BEEP 1,5: BEEP 2,7
60 BEEP .75,7: BEEP .25,8: BEEP .5,7: BEEP .5,5: BEEP .5,3:
   BEEP .5,2: BEEP 1,0
70 BEEP .75,7: BEEP .25,8: BEEP .5,7: BEEP .5,5: BEEP .5,3:
   BEEP .5,2: BEEP 1,0
80 BEEP 1,0: BEEP 1,-5: BEEP 2,0
90 BEEP 1,0: BEEP 1,-5: BEEP 2,0
```

Когда вы запустите эту программу, вы услышите похоронный марш из первой симфонии Мольера – ту часть, когда Гобблины хоронят рыцаря. Запись начала этой мелодии в ключе До-минор с указанием значений нот приведена на рисунке:



Если вы желаете исполнить мелодию в другом ключе, вы должны вставить в выражение некоторую переменную - "KEY". Например, для второй строки программы:

```
20 BEEP 1,KEY+0: BEEP 1,KEY+2: BEEP .5,KEY+3: BEEP .5, KEY+2:
   BEEP 1,KEY+0
```

Теперь при выполнении программы вы можете присвоить переменной "KEY" значения: 0 - для До-минор, 2 - для Ре-минор, 12 - для До-минор верхней октавы и т.д. Переменная "KEY" может принимать значения, кратные 1/2, 1/4, и т.д.

Таким же образом можно изменять и длительность звучания нот, но помните, что компьютер может в один момент времени исполнять только одну ноту. Это не позволяет воспроизводить сложные мелодии.

Попробуйте запрограммировать собственную мелодию. Начните с самой простой. Если вы не знаете нотной грамоты, то можете изучить ее прямо на компьютере. Например, фрагмент программы:

```
FOR N=0 TO 1000: BEEP .5,N: NEXT N
```

будет последовательно исполнять ноты до предельно высокой и завершится сообщением об ошибке "B". Вы можете также параллельно выводить значения "N", чтобы знать прямое значение ноты.

Фрагмент программы:

```
10 BEEP .5,0: BEEP .5,2: BEEP .5,4: BEEP .5,5: BEEP .5,7:
   BEEP .5,11: BEEP .5,12: STOP
```

исполняет гамму До-минор, в которой используются чистые ноты от среднего до верхнего До. Однако в этой гамме - неестественные интервалы. Скрипач бы исполнил ее так:

```
10 BEEP .5,0: BEEP .5,2.039: BEEP .5,3.86: BEEP .5,4.98:
   BEEP .5,7.02: BEEP .5,8.84: BEEP .5,10.88:BEEP .5,11:
   BEEP .5,12: STOP
```

Эти же интервалы будут естественными для гаммы, исполняемой в любом ключе, отличном от До.

Некоторая музыка, например, индийская, использует интервалы, меньшие чем полутон. Вы можете без особого труда запрограммировать это, используя оператор BEEP. Например, для звука на четверть тона выше среднего До надо указать значение высоты, равное .25.

Вы можете сделать клавиатуру компьютера клавишами

музыкального инструмента, выполнив переключение:

```
POKE 23609,255
```

Второе число здесь определяет продолжительность нахождения в этом состоянии (попробуйте изменить его от 0 до 255).

Можно также вывести музыку на внешние устройства, подключаемые к разъемам "MIC" и "EAR".

Глава 20

ВНЕШНЯЯ ПАМЯТЬ НА МАГНИТНОЙ ЛЕНТЕ

Краткое содержание: LOAD, SAVE, VERIFY, MERGE.

Основные команды работы с магнитофоном SAVE, LOAD и VERIFY уже рассматривались во введении. Вы могли видеть, что LOAD затирает старую программу в памяти компьютера при загрузке новой программы с ленты. Есть другая команда – MERGE, не делающая этого. Эта команда стирает лишь те строки старой программы или переменные, которые совпадают с номерами строк новой программы или именами новых переменных. Программу "DICE" ("игральная кость") из главы 11 запишем на ленту под именем "DICE". А теперь введем и выполним следующую программу:

```
 1 PRINT 1
 2 PRINT 2
10 PRINT 10
20 LET X=20
```

Осуществим ее проверку, заменив команду VERIFY "DICE" на команду MERGE "DICE". Вы увидите, что строки 1 и 2 сохраняются, а строки с номерами 10 и 20 заменяются на строки с такими же номерами из программы "DICE", переменная X тоже сохраняется (повторите PRINT X).

Теперь вы знаете четыре оператора для работы с магнитофоном:

SAVE – записывает программу и переменные на магнитофон;

VERIFY – проверяет программу и переменные в памяти компьютера по их копии на ленте;

LOAD – очищает память компьютера ото всех программ и загружает в нее новые, считанные с магнитофона;

MERGE – подобна LOAD, только не очищает всю память, а лишь заменяет те строки программы или переменные, у которых совпадают номера или имена с такими же на магнитной ленте.

За каждой из этих команд следует ключевое слово – имя программы, определенное первоначально командой SAVE. Пока компьютер ищет указанную программу, он выводит имена всех программ, уже прочитанных с ленты. Имеются две возможности для снятия справки с ленты.

Вариант 1. В операторах VERIFY, LOAD и MERGE вместо имени можно указать пустую строку. Тогда будет взят первый встретившийся файл.

Вариант 2. С использованием оператора SAVE:

```
SAVE STRING LINE NUMBER
```

Программа запишется на ленту так, что когда она будет вновь считана по команде LOAD (но не MERGE), она автоматически установится на строку с указанным номером и сама инициирует свое выполнение.

Кроме текстов программ, на ленту можно записать также массивы и данные.

Записать на ленту массив вы можете, используя команду SAVE с DATA таким образом:

```
SAVE STRING DATA ARRAY NAME()
```

Здесь "STRING" - имя, присваиваемое файлу данных, которое может состоять из букв или букв и символа "$" (перечеркнутая буква "Ѕ"). Для строковых данных это требование здесь не важно. Загружаются такие данные по команде:

```
LOAD STRING DATA ARRAY NAME()
```

Нельзя использовать оператор MERGE!

Если загружается строковый массив, то после обнаружения его на ленте, компьютер выдаст: "CHARACTER ARRAY:" и далее имя этого массива.

Существует возможность записи на магнитную ленту и отдельных байтов информации. Так, например, это могут быть телевизионная картинка, определяемые пользователем графические символы и т.д. Для этого используется ключевое слово CODE. Например:

```
SAVE "PICTURE" CODE 16384,6912
```

Здесь первое число - адрес первого байта в области памяти, где расположены данные, а второе число - количество байтов, которое нужно записать на ленту (6912 - объем в байтах одного экрана, 16384 - адрес экрана в памяти). Загружаются эти данные по команде:

```
LOAD "PICTURE" CODE
```

После CODE можно указать числа:

```
LOAD "PICTURE" CODE START, LENGTH
```

LENGTH (длина) - определяет сколько данных (в байтах) надо загрузить с ленты. Если длина больше, чем записано на ленту, выдается сообщение "R TAPE LOADING ERROR" (ошибка загрузки с ленты). Этот параметр можно опустить, тогда компьютер считает все данные, которые записаны на ленте.

START (начало) - указывает адрес, с которого должны загружаться данные, и может быть отличным от адреса, указанного в SAVE. Вы можете опускать этот параметр в команде LOAD.

Выражение CODE 16364, 6912 можно заменить на SCREEN$:

```
SAVE "PICTURE" SCREEN$ и затем
LOAD "PICTURE" SCREEN$
```

Это тот случай, когда VERIFY не работает. В остальных случаях VERIFY можно использовать везде, где используется SAVE.

В заключение, везде, где указывается имя файла на ленте, используются только первые 10 символов. Существует четыре типа

информации, которые могут быть записаны на ленту:
программы и переменные (совместно);
числовые массивы;
строковые массивы;
непосредственно байты.

Когда команды VERIFY, LOAD и MERGE осуществляют поиск данных на ленте, они выводят на экран все считанные ими с ленты имена с указанием типа данных в виде:

"PROGRAM:", "NUMBER ARRAY:", "CHARACTER ARRAY:" или "BYTES:"

Если имя – пустая строка, эти команды берут первый встретившийся файл с указанным типом.

Команда SAVE служит для записи информации на ленту под заданным именем. Сообщение об ошибке F выдается, если вместо имени указана пустая строка или число символов в имени 11 и более. SAVE всегда выдает сообщение "START TAPE, THEN PRESS ANY KEY" ("Запусти магнитофон и нажми любую клавишу") и ждет нажатия, после чего записывает данные на ленту.

1) Программа и переменные.

```
SAVE NAME LINE NUMBER
```

записывает программу на ленту таким образом, что последующая команда LOAD автоматически вставляет в программу "GOTO LINE NUMBER" и начинает ее выполнять.

2) Байты.

```
SAVE NAME CODE START, LENGHT
```

записывает на ленту "LENGHT" байт, начиная с адреса "START".

```
SAVE NAME SCREEN$
```

эквивалентно

```
SAVE NAME CODE 16384, 6912
```

и записывает один телевизионный экран.

3) Массивы.

```
SAVE NAME DATA LETTER() или
SAVE NAME DATA LETTER$()
```

записывают числовой или строковый массив (требование $ не относится к "NAME").

Команда VERIFY проверяет (сравнивает) информацию в памяти и на ленте и может выдавать сообщение: "R TAPE LOADING ERROR".

1) Программы и переменные.

```
VERIFY NAME
```

2) Байты.

```
VERIFY NAME CODE START, LENGTH
```

Если данных в файле "NAME" больше, чем указано в "LENGHT", то выдается сообщение об ошибке "R".

```
VERIFY NAME CODE START
```

Здесь осуществляется сравнение байтов в файле "NAME" с данными в памяти, начиная с адреса "START".

```
VERIFY NAME CODE
```

Этот оператор осуществляет сравнение данных на ленте с данными в памяти, начиная с адреса, с которого записывался на ленту первый байт данных.

```
VERIFY NAME SCREEN$ ИЛИ ЭКВИВАЛЕНТНО
VERIFY NAME CODE 16384, 6912
```

Однако это будет проверка уже проверенного файла.

3) Массивы.

```
VERRIFY NAME DATA LETTER()
VERIFY NAME DATA LETTER$()
```

Команда LOAD загружает новые данные с ленты, стирая старые данные в памяти.

1) Программы и переменные.

```
LOAD NAME
```

Может быть выдано сообщение "4 OUT OF MEMORY", если нет места для новой программы. В этом случае старая программа не уничтожается.

2) Байты.

```
LOAD NAME CODE START, LENGTH
```

Если данных в файле "NAME" больше, чем указано в "LENGTH", то выдается сообщение "R".

```
LOAD NAME CODE START
```

производит загрузку данных из "NAME" в память, начиная с адреса "START".

```
LOAD NAME CODE
```

загружает данные по адресу, начиная с которого записывались данные на ленту, в файл "NAME".

3) Массивы.

```
LOAD NAME DATA LETTER() ИЛИ
LOAD NAME DATA LETTER$()
```

Уничтожает в памяти массив с именем "LETTER" или "LETTER$", формирует новый массив и переписывает туда данные из файла "NAME". Может выдавать сообщение "4 OUT OF MEMORY" при нехватке памяти под массив. В этом случае старый массив не уничтожается.

Команда MERGE загружает новые данные с ленты, не уничтожая старых.

1) Программы и переменные.

```
MERGE NAME
```

Дописывает программу "NAME" к некоторой программе, находящейся в памяти. Может выдавать сообщение "4 OUT OF MEMORY".

2) Байты.

Не поддерживается.

3) Массивы.

Не поддерживается.

Пример. Записать на ленту информацию о 21-м определенном пользователем символе.

```
SAVE "CHESS" CODE USR "A", 21*8
```

Обратная загрузка

```
LOAD "CHESS" CODE
LOAD "CHESS" CODE USR "A"
```

Глава 21

УСТРОЙСТВО ПЕЧАТИ

Краткое содержание: LPRINT, LLIST, COPY.

Эта глава описывает операторы, необходимые для работы с принтером ZX SPECTRUM.

Два оператора LPRINT и LLIST подобны операторам PRINT и LIST, но с той разницей, что они работают не с телевизором, а с принтером. Попробуйте для примера выполнить следующую программу:

```
10 LPRINT ' "THIS PROGRAM".'
20 LLIST
30 LPRINT ' "PRINTS OUT THE CHARACTER SET".'
40 FOR N=32 TO 255
50 LPRINT CHR$ N
60 NEXT N
```

Оператор COPY позволяет распечатать экран телевизора. Например, по LIST текст программы будет выведен на экран, а затем по COPY его можно распечатать на принтере.

Вы всегда можете прекратить вывод на печать, введением BREAK (CAPS SHIFT и SPACE).

Если вы задали операторы управления принтером без подключения реального устройства, то вывода просто не будет, и выполнение программы продолжится со следующего оператора.

Теперь попробуйте выполнить следующую программу:

```
10 FOR N=31 TO 0 STEP -1
20 PRINT AT 31-N,N; CHR$(CODE "O"+N);
30 NEXT N
```

Вы получите последовательность символов, расположенных по диагонали экрана, начиная с правого верхнего угла. Теперь заменим в строке 20 "AT 31-N,N" на "TAB N" – и программа будет работать также, как и прежде. Теперь заменим в строке 20 "PRINT" на "LPRINT" и заметим, что развертки по диагонали не получится. Заменив теперь "TAB N" на "AT 31-N,N" и сохранив "LPRINT" получим по одному символу на строку, что и требовалось получить.

Вообще при печати перевод строки осуществляется в следующих случаях:

а) при заполнении буфера строки;

б) после LPRINT, если это не конец оператора и в нем встретилась запятая или точка с запятой;

в) если запятая, апостроф или TAB требуют новой строки;

г) при окончании программы, если остались невыведенные данные.

Глава 22

ДРУГОЕ ПЕРИФЕРИЙНОЕ ОБОРУДОВАНИЕ

Имеется ряд других устройств, которые могут быть подключены к компьютеру ZX SPECTRUM.

ZX MICRODRIVE - высокоскоростное устройство памяти - может быть использовано вместо кассетного магнитофона, однако оно не может управляться командами SAVE, VERIFY, LOAD и MERGE, а лишь командами PRINT, LIST, INPUT и INKEY$.

При помощи этого устройства можно организовать сеть из нескольких компьютеров ZX SPECTRUM.

Стандартным интерфейсом для ZX SPECTRUM является RS-232, посредством которого подключаются клавиатура, принтер и любые другие устройства, отвечающие стандартам этого интерфейса. При работе с такими устройствами могут использоваться на основной клавиатуре дополнительные ключевые слова OPEN#, CLOSE#, MOVE, ERASE, CAT и FORMAT.

Глава 23

ВВОД И ВЫВОД

Краткое содержание: OUT, IN.

Компьютер может считывать некоторую информацию и записывать ее в свою оперативную память по командам PEEK и POKE. Вся память компьютера - и ПЗУ, и ОЗУ - представляется совокупностью адресов от 0 до 65536, каждый из которых адресует один байт.

Таким же образом можно адресовать и еще 65536 адресов, называемых портами ввода-вывода. Они используются процессором для связи с клавиатурой и принтером и могут управляться операторами IN и OUT.

IN аналогичен оператору PEEK:

```
IN ADDRESS
```

Он использует один аргумент - адрес порта и позволяет считать один байт из указанного порта.

UT подобен оператору POKE:

```
OUT ADDRESS, VALUE
```

и записывает указанные данные в заданный порт вывода.

ZX SPECTRUM оперирует шестнадцатиразрядными адресами, которые мы будем обозначать буквой A:

```
A15,A14,A13,...,A1,A0.
```

Биты адреса A0,A1,A2,A3 и A4 очень важны. Как правило, они равны 1, если хотя бы один из них - 0. Это предписывает компьютеру некоторые действия. Не более, чем один из этих пяти битов может быть - 0.

Биты A6 и A7 игнорируются, так что, если вы знакомы с электроникой, то можете использовать их по своему усмотрению.

Биты A8 и A9 используются иногда для получения дополнительной информации.

Информационный байт мы будем обозначать буквой D:

D7,D6,D5,...,D1,D0.

Теперь представим список адресов портов. Имеется целый ряд входных адресов для чтения с клавиатуры, а также входного разъема "EAR". Сама клавиатура делится на 8 полурядов по 5 клавиш в ряду.

IN 65278 считывает ряд от CAPS SHIFT до V,
IN 65022 считывает ряд от A до G,
IN 64510 считывает ряд от Q до T,
IN 63486 считывает ряд от 1 до 5,
IN 61438 считывает ряд от 0 до 6,
IN 57342 считывает ряд от P до Y,
IN 49150 считывает ряд от ENTER до H,
IN 32766 считывает ряд от SPACE до B.

Эти адреса могут быть вычислены из выражения:

254+256*(255-2**N) при N, пробегающем от 0 до 7.

В байте, считанном с клавиатуры, биты от D0 до D4 служат для обозначения пяти клавиш в данном полуряду: D0 - для крайней клавиши, а D4 - для той, что ближе к центру. Состояние одного из этих битов - 0 указывает, что соответствующая ему клавиша нажата. D6 принимает свое значение при чтении с разъема "EAR".

Выходной порт 254 обеспечивает громкоговоритель (D4) и разъем "MIC" (D3), а также установку цвета (D2, D1, D0).

Порт 251 обеспечивает связь с принтером: как чтение, так и запись. Чтение - для проверки готовности принтера к работе.

Порты 254, 247 и 239 используются для связи с дополнительными устройствами, описанными в главе 22.

Запустите следующую программу:

```
10 FOR N=0 TO 7: REM HELP=ROW NUMBER (номер полуряда)
20 LET A=254+256*(255-2**N)
30 PRINT AT=0,0;IN A: GOTO 30
```

и понажимайте по одной клавише в каждом полуряду. После нажатия очередной клавиши введите BREAK, а затем NEXT N.

Ниже на рисунке показано распределение контактов разъема:

```
A14 ----+---- A15
A12 ----+---- A13
+5V ----+---- D7
+9V ----+
 0V ----+---- D0
 0V ----+---- D1
 CK ----+---- D2
 A0 ----+---- D6
 A1 ----+---- D5
 A2 ----+---- D3
 A3 ----+---- D4
-IORQGE ----+---- -INT
 0V ----+---- -NMI
VIDEO ----+---- -HALT
  Y ----+---- -MREQ
```

```
      V ----+---- ^IOREQ
      U ----+---- ^RD
 ^BUSRQ ----+---- ^WR
 ^RESET ----+---- +5V
     A7 ----+---- ^WAIT
     A6 ----+---- +12V
     A5 ----+---- -12V
     A4 ----+---- ^MI
 ^ROMCS ----+---- ^HFSH
^BUSACK ----+---- A8
     A9 ----+---- A10
    A11 ----+----
```

Глава 24

ПАМЯТЬ

Краткое содержание: CLEAR.

Вся память компьютера разбита на байты, каждый из которых представим числом от 0 до 255. Каждый байт может быть записан в памяти по определенному адресу от 0 до FFFFH (H - здесь и далее означает шестнадцатиричное представление числа). Сам адрес может быть записан в памяти как два байта.

На диаграмме показано распределение памяти компьютера ZX-SPECTRUM:

0 3FFFH ROM (ПЗУ)	4000H 7FFFH RAM (ОЗУ) (Системная область)	8000H FFFFH Допустимо для программ
1 16383	16384 32767	32768 65536

Для получения любой области из памяти используется функция PEEK с адресом в качестве аргумента. Функция возвращает значение байта по этому адресу.

Рассматриваемая ниже программа выводит содержимое первых 21 байтов из ROM с их адресами:

```
10 PRINT "ADDRESS"; TAB 8; "BYTE"
20 FOR A=0 TO 20
30 PRINT A ;TAB 8; PEEK A
40 NEXT A
```

Для изменения содержимого памяти (только для RAM (ОЗУ)) используется оператор POKE в форме:

```
POKE ADDRESS, NEW CONTENTS
```

где "ADDRESS" и "NEW CONTENTS" - числовые выражения. Например:

```
POKE 31000,57
```

"NEW CONTENTS" может принимать значения от -255 до +255.

Вся память подразделяется на области, предназначенные для хранения информации различного назначения, что показано на диаграмме:

область экрана TV	атрибуты	буфер принтера	системные переменные	план дополнительной памяти
16384	22528	23296	23562	23734

канальная информация	80H	программа на Бейсике	переменные программы	80H	редактир. строки программы	NI
CHANS		PROG	VARS		E LINE	

считанные данные	NL	рабочая область	стек калькулятора	резерв	аппаратный стек
WORKSP			STKBOT	STKEND	SP

стек переходов к подпрог.	7	ЗЕН	определяемые пользователем символы
		RAMTOP	UDG R RAMT

Область телевизионного экрана содержит образ текущего кадра. Она доступна для операторов PEEK и POKE. Каждая позиция экрана представима матрицей 8x8 точек (один байт на каждый ряд из 8 точек). Однако эти восемь байт хранятся в памяти не вместе.

Полный экран представляет собой 24 строки по 32 символа. Каждая строка экрана прописывается 8-мью строками развертки телевизионного экрана. Итого, для записи одного экрана выполняется 191 сканирование, и в памяти хранятся байты одноименных рядов матриц соседних позиций экрана.

Область атрибутов содержит данные о цвете и других параметрах каждой позиции экрана. Используется в формате ATTR.

Буфер принтера содержит символы, передаваемые на печать.

Область системных переменных содержит данные управления вычислениями. Они полностью описаны в следующей главе. Некоторые из них (CHANS, PROG, VARS, E LINE и т.д.) содержат адреса границ между системными областями памяти, но это не переменные Бейсика, и их имена не распознаются компьютером.

Область планов дополнительной памяти используется только с MICRODRIVE.

В области канальной информации содержатся данные об устройствах ввода-вывода, а именно: о клавиатуре (с нижней половиной экрана), о верхней половине экране и о принтере.

Каждая строка области Бейсик-программ имеет формат:

Старший байт Младший байт

2 байта	2 байта	т е к с т	0DH

номер строки длина текста+ENTER END

Числовые константы в программе представлены в двоичной форме при использовании CHR$14 и следующих за ним 5 байт самого числа.

Переменные имеют различные форматы представления в памяти. Ниже представлен формат записи числа, имя которого состоит из одной буквы.

бон	1 байт порядка	бит знака	4 байта мантиссы

буква значение числа

Формат размещения числа, если его имя имеет более, чем одну букву:

бон	ннн	...	ннн	5 байт значения числа

1 буква 2 буква последняя буква

Формат размещения числового массива:

ннн	2 байта	1 байт	2 байта	...	2 байта	по 5 байтов на каждый элемент

буква общая длина элементов +1 на каждое измерение | номера размерностей | первая размерность | последняя размерность

Порядок следования элементов следующий:

1. Элементы, для которых первая размерность равна 1;
2. Элементы, для которых первая размерность равна 2;
3. Элементы, для которых первая размерность равна 3 и т.д.

Затем в том же порядке, но для следующей размерности, и т.д. Например, элементы массива с размерностью (3x6) расположатся в памяти в следующем порядке:

```
B(1,1), B(1,2), B(1,3), B(1,4), B(1,5), B(1,6), B(2,1),
B(2,2), ... , B(3,4)
```

Формат размещения управляющих переменных для FOR-NEXT операторов:

	5 байт	5 байт	5 байт	2 байта	1 байт
буква	значение	ограни-чение	прираще-ние	строка цикла	номер оператора

Формат размещения строки символов:

	2 байта	текст (может быть пустая строка)
буква	количество символов	

Формат размещения строкового массива:

	2 байта	1 байт	2 байта
буква	общее число элементов+1 на каждую размерность	номера раз-мерностей	1-я размерность

...	2 байта	по одному байту на каждый элемент
	последняя размерность	элементы

Программируемый стек есть часть интерпретатора Бейсика.

Аппаратный стек используется микропроцессором z80 для запоминания адресов возврата.

Резерв в данной версии не используется.

Назначение стека переходов к подпрограммам описано в главе 5.

Байт, адресуемый по RAMTOP, содержит верхний адрес, используемый Бейсиком.

Даже оператор NEW, который очищает ОЗУ, не изменяет содержимого области определяемых пользователем символов.

Вы можете изменить адрес RAMTOP в операторе CLEAR:

```
CLEAR NEW RAMTOP
```

при этом:

а) очищаются все области переменных;

б) очищается область экрана (подобно CLS);

в) переустанавливается позиция PLOT в левый нижний угол экрана;

г) выполняется функция RESTORE;

д) очищается стек переходов и устанавливается новое значение RAMTOP.

Команда RUN также выполняет действия команды CLEAR, хотя и не изменяет RAMTOP.

Используя функцию CLEAR, вы можете смещать RAMTOP, увеличивая тем самым область для Бейсика и уменьшая область определяемых пользователем символов. Можно несколько увеличить доступную часть RAM (ОЗУ), используя команду NEW. Например, выполнение NEW, а затем CLEAR 23800 помогает компьютеру при переполнении ОЗУ.

Все указанные действия могут приводить к двум сообщениям об ошибке и выдаче звукового сигнала:

"4 MEMORY FULL" (переполнение памяти);
"G NO ROOM FOR LINE" (нет места для строки программы).

Можно изменить длительность подачи звукового сигнала, изменяя число по адресу 23608. По умолчанию оно предполагается равным 64.

Числа (за исключением 0) могут записываться в показательной форме как

```
+M *2 **E,
```

где M - мантисса в интервале 0.5...1 (меньше 1);
E - экспонента, положительное или отрицательное число.

Допустим вы записали "м" в двоичной системе счисления. "м" - дробное и имеет двоичную точку (подобно десятичной точке). Тогда будет:

```
1/2  ⟶  .1
1/4  ⟶  .01
3/4  ⟶  .11 и т.д.
```

Наше число "м" меньше, чем 1. Значит у него нет битов перед двоичной точкой, а поскольку оно больше 0.5, то левый бит, следующий за точкой - это 1.

Для записи чисел в память мы используем 5 байтов в следующем порядке:

а) записываем первые 8 битов мантиссы во второй байт (мы помним, что первый бит - это 1), вторые 8 битов - в третий байт и так до пятого байта;

б) заменяем первый бит второго байта, в котором записана 1, на знаковый бит (0 для +, и 1 для -);

в) записываем в первый байт экспоненту +128.

Например, мы хотим записать число 1/10:

```
1/10=4/5*2** -3
```

Мантисса будет равна .1001100110011001100110011001100в (поскольку 33-й бит равен 1, мы должны округлить 32-й бит, записав 1 вместо 0). Экспонента равна -3.

Теперь, применив наши три правила, запишем 5 байт:

↓— знак числа

111 1101	0	100 1100	1100 1100	1100 1100	1100 1100

-3 +128 мантиса 4/5, исключая левый знаковый бит

Имеется альтернативный способ записи целого числа в интервале -65535...+65535:

а) первый байт равен 0;
б) второй байт равен 0 для положительного числа и FFH - для отрицательного;
в) третий и четвертый байты содержат младшие и старшие значащие биты числа (или число +131072, если оно отрицательное);
г) пятый байт равен 0.

Глава 25

СИСТЕМНЫЕ ПЕРЕМЕННЫЕ

Байты памяти с 23532 до 23733 предназначены для специального использования. В них размещаются так называемые системные переменные. Не надо путать их имена с именами переменных в программе. Компьютер не распознает ссылки на эти переменные из программы по их именам. Имена используются только для мнемонического обозначения этих переменных в этом описании.

Информация, записанная в первом столбце таблицы, имеет следующее значение:

X - переменная не должна меняться, так как это может нарушить работу системы;
N - изменение переменной не приводит к длительному эффекту;
число - число байтов в переменной (для двухбайтовых переменных младший байт - первый).

Например, необходимо изменить значение на v в двухбайтовой переменной по адресу N:

```
10 POKE N,V-256*INT(V/256)
20 POKE N+1,INT(V/256)
```

Для просмотра нового значения можно использовать оператор:

```
PEEK N+256*PEEK(N+1)
```

Зн.	Адрес	Имя	Содержание
N8	23552	KSTATE	Используется при чтении с клавиатуры
N1	23560	LAST K	Запоминается вновь нажатая клавиша
1	23561	REPDEL	Время в 50-ых долях секунды, в течение которого клавиша должна быть зафиксирована в нажатом состоянии. Начальное значение 35, но может быть изменено
1	23562	REPPER	Задержка в 50-х долях секунды между последовательными опросами клавиш. Начальное значение - 5
N2	23563	DEFADD	Адрес аргументов функций пользователя, если они используются, иначе 0
N1	23565	K DATA	Второй байт управления цветом с клавиатуры

N2	23566	TVDATA	Байты цвета, AT,TAB управления TV
X38	23568	STRMS	Адреса подключенных каналов
2	23606	CHARS	Адрес символьного набора -255. Обычно этот набор находится в ПЗУ, но может быть размещен и в ОЗУ с указанием в CHARS адреса размещения
1	23608	RASP	Продолжительность звукового сигнала
1	23609	PIP	Длительность задержки, устраняющей дребезг контактов
1	23610	ERR NR	Код сообщения -1, начальное значение 255 (для "-1"), т.е. РЯМК 23610 - 255
X1	23611	FLAGS	Управляющие флажки Бейсика
X1	23612	TV FLAG	Флажок телевизора
X2	23613	ERR SP	Адрес в аппаратном стеке, используемый как адрес возврата при ошибке
N2	23615	LIST SP	Адрес возврата из автоматического листинга
N1	23617	MODE	Режим, спецификация [K], [L], [C], [E] или [G] курсора
2	23618	NEW PPC	Номер строки, на которую должен быть сделан переход
2	23621	PPC	Номер строки, оператор в которой выполняется
1	23623	SUB PPC	Порядковый номер выполняющегося оператора в строке
1	23624	BORDCR	Цвет рамки экрана, содержит атрибуты
2	23625	E PPC	Количество текущих строк (с курсором)
X2	23627	VARS	Адреса переменных
N2	23629	DEST	Адрес переменной в задании
X2	23631	CHANS	Адрес канала данных
X2	23633	CURCHL	Адрес данных для ввода-вывода
X2	23636	PROG	Адрес Бейсик-программы
X2	23637	NXTLIN	Адрес следующей строки программы

X2	23639	DATADD	Адрес терминатора последнего символа в DATA
X2	23641	E LINE	Адрес выведенной команды
2	23643	K CUR	Адрес курсора
X2	23645	CH ADD	Адрес следующего интерпретируемого символа: символ аргумента в PEEK, NEWLINE или POKE операторах
2	23647	X PRT	Адрес символа, следующего за маркером [?]
X2	23649	WORK SP	Адрес временной рабочей области
X2	23651	STK BOT	Адрес ДНА программируемого стека
X2	23653	STK END	Адрес начала резервной области памяти
N1	23655	BREG	B-регистр калькулятора
N2	23656	MEM	Адрес области, используемой как память калькулятора (обычно MEMBOT, но не всегда)
1	23658	FLAG 52	Старшие флажки
X1	23659	DF SZ	Число строк (включая и одну чистую) в нижней части экрана
2	23660	S TOP	Количество верхних строк программы в автоматическом листинге
2	23662	OLDPPC	Номер строки, на которую указывает CONTINUE
1	23664	OSPPC	Номер оператора в строке, на который указывает CONTINUE
N1	23665	FLAGX	Переменные флажки
N2	23666	STR LEN	Размер расстояний между строками
N2	23668	T ADDR	Адрес следующего символа в синтаксической таблице
2	23670	SEED	Начальное значение для RND. Изменяется функцией RANDOMIZE
3	23672	FRAMES	Счетчик кадров - приращение через каждые 20 ms (см. главу 18)
2	23675	UDG	Адрес первого определяемого пользователем символа

1	23677 23678	COORDS	X-координата точки графопостроителя, Y-координата точки графопостроителя
1	23679	P POSN	33-позиционное число для позиционирования принтера
1	23680	PR CC	Младший байт адреса позиции для LPRINT для печати
1	23681		Не используется
2	23682	ECHO E	33-позиционное и 24-строковое числа (в нижней половине конца входного буфера)
2	23684	DF CC	Адрес PRINT-позиции в области экрана
2	23686	DF CCL	Подобно DF CC для нижней части экрана
X1 X1	23688 23689	S POSN	33-позиционное число для PRINT-позиции, 23-строковое число для PRINT-позиции
X2	23690	S POSNL	Подобно S POSN для нижней части
1	23692	SCR CT	Счетчик сверток: всегда на 1 больше числа сверток, которые должны быть проведены перед остановом со сверткой. Если вы установите это число, больше чем на 1 (скажем 255), то экран будет сворачиваться без запроса к вам
1	23693	ATTR P	Сплошные цвета
1	23694	MASK P	Используется для высвечивания цветов. Бит, установленный в 1, показывает. Биты атрибутов берутся не из ATTR P, а из того, что указано на экране
N1	23695	ATTR T	Временный указатель цветов
N1	23696	MASK T	Временный MASK P
1	23697	P FLAG	Старшие флажки
N30	23698	MEMBOT	Область памяти для калькулятора. Используется для записи чисел, которые не могут быть размещены в программируемом стеке калькулятора
2	23728		Не используется
2	23730	RAMTOP	Адрес последнего байта области программы
2	23732	P-RAMT	Адрес последнего байта физического ОЗУ

Следующая программа выдает вам первые 22 байта области системных переменных:

```
10 FOR N=0 TO 21
20 PRINT PEEK (PEEK 23627+256*PEEK 23628+N)
30 NEXT N
```

Теперь замените строку 20 на следующую

```
20 PRINT PEEK(23765+N)
```

и вы дополнительно получите ДАМП самой программы.

Глава 26

ИСПОЛЬЗОВАНИЕ МАШИННЫХ КОДОВ

Краткое содержание: USR с числовым аргументом.

Эта глава описывает применение машинных команд микропроцессора Z80 (U880 (ГДР), 1810ВМ80 (СССР)).

Программы в машинных кодах пишут обычно на ассемблере с последующей трансляцией. (Перечень мнемокодов команд микропроцессора Z80 приведен в Приложении А). Транслятор с ассемблера встроен в компьютер ZX SPECTRUM.

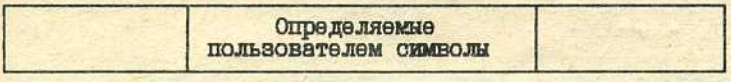
Приведем пример программы:

```
LD BC,99
RET
```

которая загружает в "BC"-регистр число 99. Эта программа будет транслироваться в 4-х байтный машинный код:

```
байты 1,99,0 для LD BC,99 и
      201 для RET
```

Следующим шагом является загрузка программы в компьютер. Для этого используется дополнительная память, получаемая между Бейсик-областью и областью определяемых пользователем символов. Допустим вы имели следующее распределение последней части ОЗУ:

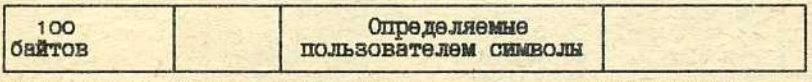


RAMTOP=32599 UDG=32600 PRAMT=32767

Если вы теперь выполните

```
CLEAR 32499,
```

то получите 100 байтов памяти, начиная с адреса 32500



RAMTOP=32499 32500 UDG=32600 PRAMT=32767

Для загрузки программы в машинных кодах вы можете выполнить следующую Бейсик-программу:

```
10 LET A=32500
```

```
20 READ N :POKE A,N
30 LET A=A+1: GOTO 20
40 DATA 1,99,0,201
```

(Программа может завершиться сообщением "E OUT OF DATA", если переполняются отведенные вами 4 байта).

Для выполнения загруженных машинных кодов используется функция USR, но с числовым аргументом, определяющим начальный адрес.

Если вы выполните:

```
PRINT USR 32500
```

то получите ответ: 99.

Возврат в Бейсик-программу осуществляется обычным образом по команде микропроцессора RET. В машинной программе вы не должны использовать регистры IY и IX.

Вы можете записать вашу программу на ленту:

```
SAVE "NAME" CODE 32500,4
```

Можно записать эту программу и так, что она будет автоматически выполняться после загрузки:

```
10 LOAD " " CODE 32500,4
20 PRINT USR 32500
```

Для этого надо сделать:

```
SAVE NAME LINE
```

а затем:

```
SAVE "XXXX" CODE 32500,4
LOAD "NAME"
```

Это приведет к тому, что вначале будет загружена и автоматически выполнена Бейсик-программа, которая, в свою очередь, загрузит и выполнит программу в машинных кодах.

Приложение А

ПОЛНЫЙ НАБОР СИМВОЛОВ

Дес. код	Символ	Шест код	Ассемблер. мнемоника	DEH...	HDH...
0	Не использ.	00	NOP	RLC B	
1	Не использ.	01	LD BC,NN	RLC C	
2	Не использ.	02	LD (BC),A	RLC D	
3	Не использ.	03	INC BC	RLC E	
4	Не использ.	04	INC B	RLC H	
5	Не использ.	05	DEC B	RLC L	
6	PRINT упр.	06	LD B,N	RLC (HL)	
7	EDIT	07	RLCA	RLC A	
8	Курс.влево	08	EX AF,AF'	RRC B	
9	Курс.вправо	09	ADD HL,BC	RRC E	
10	Курс.вниз	0A	LD A,(BC)	RRC D	
11	Курс.вверх	0B	DEC BC	RRC E	

12	DELETE	0C	INC C	RRC H	
13	ENTER	0D	DEC C	RRC L	
14	Число	0E	LD C,N	RRC (HL)	
15	Не использ.	0F	RRCA	RRC A	
16	INC упр.	10	DUNZ DIS	RL B	
17	PAPER упр.	11	LD DE,NN	RL C	
18	FLASH упр.	12	LD (DE),A	RL D	
19	BRIGHT упр.	13	INC DE	RL E	
20	INVERSE упр.	14	INC D	RL H	
21	OVER упр.	15	DEC D	RL L	
22	AT упр.	16	LD D,N	RL (HL)	
23	TAB упр.	17	RLA	RL A	
24	Не использ.	18	JR DIS	RR B	
25	Не использ.	19	ADD HL,DE	RR C	
26	Не использ.	1A	LD A,(DE)	RR D	
27	Не использ.	1B	DEC DE	RR E	
28	Не использ.	1C	INC E	RR H	
29	Не использ.	1D	DEC E	RR L	
30	Не использ.	1E	LD E,N	RR (HL)	
31	Не использ.	1F	RRA	RR A	
32	Пробел	20	JR NZ,DIS	SLA B	
33	!	21	LD HL,NN	SLA C	
34	"	22	DL (NN),HL	SLA C	
35	#	23	INC HL	SLA E	
36	$	24	INC H	SLA H	
37	%	25	DEC H	SLA L	
38	&	26	DL H,N	SLA (HL)	
39	'	27	DAA	SLA A	
40	(	28	JR Z,DIS	SRA B	
41	)	29	ADD HL,HL	SRA C	
42	*	2A	LD HL,NN	SRA D	
43	+	2B	DEC HL	SRA E	
44	,	2C	INC L	SRA H	
45	-	2D	DEC L	SRA E	
46	.	2E	LD L,N	SRA (HL)	
47	/	2F	CPL	SRA A	
48	0	30	JR NC,DIS		
49	1	31	LD SP,NN		
50	2	32	LD (NN),A		
51	3	33	INC SP		
52	4	34	INC (HL)		
53	5	35	DEC (HL)		
54	6	36	LD (HL),N		
55	7	37	SCF		
56	8	38	JB C,DIS	SRL B	
57	9	39	ADD HL,SP	SRL C	
58	:	3A	LD A,(NN)	SRL D	
59	;	3B	DEC SP	SRL E	
60	<	3C	INC A	SZL H	
61	=	3D	DEC A	SRL L	
62	>	3E	LD A,N	SRL (HL)	
63	?	3F	CCF	SRL A	
64	@	40	LD B,B	BIT 0,B	IN B,(C)
65	A	41	LD B,C	BIT 0,C	OUT (C),B

66	B	42	LD B,D	BIT 0,D	SBC HL,BC
67	C	43	LD B,E	BIT 0,E	LD (NN),BC
68	D	44	LD B,H	BIT 0,H	NEG
69	E	45	LD B,L	BIT 0,L	RETN
70	F	46	LD B,(HL)	BIT 0,(HL)	IM 0
71	G	47	LD B,A	BIT 0,A	LD I,A
72	H	48	LD C,B	BIT 1,B	IN C,(C)
73	I	49	LD C,C	BIT 1,C	OUT C,(C)
74	J	4A	LD C,D	BIT 1,D	ADD HL,BC
75	K	4B	LD C,E	BIT 1,E	LD BC,(NN)
76	L	4C	LD C,H	BIT 1,H	
77	M	4D	LD C,L	BIT 1,L	RET I
78	N	4E	LD C,(HL)	BIT 1,(HL)	
79	O	4F	LD C,A	BIT 1,A	LD R,A
80	P	50	LD D,B	BIT 2,B	IN D,(C)
81	Q	51	LD D,C	BIT 2,C	OUT (C)
82	R	52	LD D,D	BIT 2,D	SBC HL,DE
83	S	53	LD D,E	BIT 2,E	LD (NN),DE
84	T	54	LD D,H	BIT 2,H	
85	U	55	LD D,L	BIT 2,L	
86	V	56	LD D,(HL)	BIT 2,(HL)	IM 1
87	W	57	LD D,A	BIT 2,A	LD A,I
88	X	58	LD E,B	BIT 3,B	IN E,(C)
89	Y	59	LD E,C	BIT 3,C	OUT (C)
90	Z	5A	LD E,D	BIT 3,D	ADC RL,DE
91	[	5B	LD E,E	BIT 3,E	LD DE,(NN)
92	\	5C	LD E,H	BIT 3,H	
93	]	5D	LD E,L	BIT 3,L	
94	^	5E	LD E,(HL)	BIT 3,(HL)	IM 2
95	_	5F	LD E,A	BIT 3,A	LD A,R
96	`	60	LD H,B	BIT 4,B	IN H,(C)
97	a	61	LD H,C	BIT 4,C	OUT (C)
98	b	62	LD H,D	BIT 4,D	SBC HL,HL
99	c	63	LD H,E	BIT 4,E	LD (NN),HL
100	d	64	LD H,H	BIT 4,H	
101	e	65	LD H,L	BIT 4,L	
102	f	66	LD H,(HL)	BIT 4,(HL)	
103	g	67	LD H,A	BIT 4,A	RRD
104	h	68	LD L,B	BIT 5,B	IN L,(C)
105	i	69	LD L,C	BIT 5,C	OUT (C),L
106	j	6A	LD L,D	BIT 5,D	ADC HL,HL
107	k	6B	LD L,E	BIT 5,E	LD HL,(NN)
108	l	6C	LD L,H	BIT 5,H	
109	m	6D	LD L,L	BIT 5,L	
110	n	6E	LD L,(HL)	BIT 5,(HL)	
111	o	6F	LD L,A	BIT 5,A	RID
112	p	70	LD (HL),B	BIT 6,B	IN F,(C)
113	q	71	LD (HL),C	BIT 6,C	
114	r	72	LD (HL),D	BIT 6,D	SBC HL,SP
115	s	73	LD (HL),E	BIT 6,E	LD (NN),SP
116	t	74	LD (HL),H	BIT 6,H	
117	u	75	LD (HL),L	BIT 6,L	
118	v	76	HALT	BIT 6,(HL)	
119	w	77	LD (HL),A	BIT 6,A	

120	x	78	LD A,B	BIT 7,B	IN A,(C)
121	y	79	LD A,C	BIT 7,C	OUT (C)
122	z	7A	LD A,D	BIT 7,D	ADC HL,SP
123	{	7B	LD A,E	BIT 7,E	LD SP,(NN)
124	|	7C	LD A,H	BIT 7,H	
125	}	7D	LD A,L	BIT 7,L	
126	~	7E	LD A,(HL)	BIT 7,H	
127	⌂	7F	LD A,A	BIT 7,A	
128	ÇÇ	80	ADD A,B	RES 0,B	
129	[illegible]	81	ADD A,C	RES 0,C	
130	[illegible]	82	ADD A,D	RES 0,D	
131	[illegible]	83	ADD A,E	RES 0,E	
132	[illegible]	84	ADD A,H	RES 0,H	
133	[illegible]	85	ADD A,L	RES 0,L	
134	[illegible]	86	ADD A,HL	RES 0,(HL)	
135	[illegible]	87	ADD A,A	RES 0,A	
136	[illegible]	88	ADC A,B	RES 1,B	
137	[illegible]	89	ADC A,C	RES 1,C	
138	[illegible]	8A	ADC A,D	RES 1,D	
139	[illegible]	8B	ADC A,E	RES 1,E	
140	[illegible]	8C	ADC A,H	RES 1,H	
141	[illegible]	8D	ADC A,L	RES 1,L	
142	[illegible]	8E	ADC A,(HL)	RES 1,(HL)	
143	[illegible]	8F	ADC A,A	RES 1,A	
144	(A)	90	SUB B	RES 2,B	
145	(B)	91	SUB C	RES 2,C	
146	(C)	92	SUB D	RES 2,D	
147	(D)	93	SUB E	RES 2,E	
148	(E)	94	SUB H	RES 2,H	
149	(F)	95	SUB L	RES 2,L	
150	(G)	96	SUB (HL)	RES 2,(HL)	

151	(H)	97	SUB A	RES 2,A	
152	(I)	98	SBC A,B	RES 3,B	
153	(J)	99	SBC A,C	RES 3,C	
154	(K)	9A	SBC A,D	RES 3,D	
155	(L)	9B	SBC A,E	RES 3,E	
156	(M)	9C	SBC A,H	RES 3,H	
157	(N)	9D	SBC A,L	RES 3,C	
158	(O)	9E	SBC A,(HL)	RES 3,D	
159	(P)	9F	SBC A,A	RES 3,E	
160	(Q)	A0	AND B	RES 4,B	LDI
161	(R)	A1	AND C	RES 4,C	CPI
162	(S)	A2	AND D	RES 4,D	INI
163	(T)	A3	AND E	RES 4,E	OUT I
164	(U)	A4	AND H	RES 4,H	
165	RND	A5	AND L	RES 4,L	
166	INKEY$	A6	AND (HL)	RES 4,(HL)	
167	PI	A7	AND A	RES 4,A	
168	FN	A8	XOR B	RES 5,B	LDD
169	POINT	A9	XOR C	RES 5,C	CPD
170	SCREEN$	AA	XOR D	RES 5,D	IND
171	ATTR	AB	XOR E	RES 5,E	OUT O
172	AT	AC	XOR H	RES 5,H	
173	TAB	AD	XOR L	RES 5,L	
174	VAL$	AE	XOR (HL)	RES 5,(HL)	
175	CODE	AF	XOR A	RES 5,A	
176	VAL	B0	OR B	RES 6,B	LDIR
177	LEN	B1	OR C	RES 6,C	CPIR
178	SIN	B2	OR D	RES 6,D	INIR
179	COS	B3	OR E	RES 6,E	OTIR
180	TAN	B4	OR H	RES 6,H	
181	ASN	B5	OR L	RES 6,L	
182	ACS	B6	OR (HL)	RES 6,(HL)	
183	ATN	B7	OR A	RES 6,A	
184	LN	B8	CP B	RES 7,B	LDDR
185	EXP	B9	CP C	RES 7,C	CPDR
186	INT	B9	CP D	RES 7,D	INDR
187	SQR	BA	CP E	RES 7,E	OTDR
188	SGN	BB	CP H	RES 7,H	
189	ABS	BD	CP L	RES 7,L	
190	PEEK	BE	CP (HL)	RES 7,(HL)	
191	IN	BF	CP A	RES 7,A	
192	USR	C0	RET NZ	SET 0,B	
193	STR$	C1	POP BC	SET 0,C	
194	CHR$	C2	JP NZ,NN	SET 0,D	
195	NOT	C3	JP NN	SET 0,E	
196	BIN	C4	CALL NZ,NN	SET 0,H	
197	OR	C5	PUSH BC	SET 0,L	
198	AND	C6	ADD A,N	SET 0,(HL)	
199	<=	C7	RST 0	SET 0,A	
200	>=	C8	RET Z	SET 1,B	
201	<>	C9	RET	SET 1,C	
202	LINE	CA	JP Z,NN	SET 1,D	
203	THEN	CB		SET 1,E	
204	TO	CC	CALL Z,NN	SET 1,H	

206	STEP	CD	CALL NN	SET 1,L	
206	DEF FN	CE	ADC A,N	SET 1,(HL)	
207	CAT	CF	RST 8	SET 1,A	
208	FORMAT	D0	RET NC	SET 2,B	
209	MOVE	D1	POP DE	SET 2,C	
210	ERASE	D2	JP NC,NN	SET 2,D	
211	OPEN#	D3	OUT (N),A	SET 2,E	
212	CLOSE#	D4	CALL NC,NN	SET 2,H	
213	MERGE	D5	PUSH DE	SET 2,L	
214	VERIFY	D6	SUB N	SET 2,(HL)	
215	BEEP	D7	RST 16	SET 2,A	
216	CIRCLE	D8	RET C	SET 3,B	
217	INK	D9	EXX	SET 3,C	
218	PAPER	DA	JP C,NN	SET 3,D	
219	FLASH	DB	IN A,(N)	SET 3,E	
220	BRIGHT	DC	CALL C,NN	SET 3,H	
221	INVERSE	DD	PREFIXES IN-STRUCTIONS USING IX	SET 3,L	
222	OVER	DE	SBC A,N	SET 3,(HL)	
223	OUT	DF	RST 24	SET 3,A	
224	LPRINT	E0	RET PO	SET 4,B	
225	LLIST	E1	POP HL	SET 4,C	
226	STOP	E2	JP PO,NN	SET 4,D	
227	READ	E3	EX (SP),HL	SET 4,E	
228	DATA	E4	CALL PO,NN	SET 4,H	
229	RESTORE	E5	PUSH HL	SET 4,L	
230	NEW	E6	AND N	SET 4,(HL)	
231	BORDER	E7	RST 32	SET 4,A	
232	CONTINUE	E8	RET PE	SET 5,B	
233	DIM	E9	JP (HL)	SET 5,C	
234	REM	EA	JP PE,NN	SET 5,D	
235	FOR	EB	EX DE,HL	SET 5,E	
236	GOTO	EC	CALL PE,NN	SET 5,H	
237	GOSUB	ED		SET 5,L	
238	INPUT	EE	XOR N	SET 5,(HL)	
239	LOAD	EF	RST 40	SET 5,A	
240	LIST	F0	RET P	SET 6,B	
241	LET	F1	POP AF	SET 6,C	
242	PAUSE	F2	JP P,NN	SET 6,D	
243	NEXT	F3	DI	SET 6,E	
244	POKE	F4	CALL P,NN	SET 6,H	
245	PRINT	F5	PUSH AF	SET 6,L	
246	PLOT	F6	OR N	SET 6,(HL)	
247	RUN	F7	RST 48	SET 6,A	
248	SAVE	F8	RET M	SET 7,B	
249	RANDOMIZE	F9	LD SP,HL	SET 7,C	
250	IF	FA	JP M,NN	SET 7,D	
251	CLS	FB	EI	SET 7,E	
252	DRAW	FC	CALL M,NN	SET 7,H	
252	CLEAR	FD	PREFIXES IN-STR.USING IY	SET 7,L	
254	RETURN	FE	CP N	SET 7,(HL)	
255	COPY	FF	RST 56	SET 7,A	

Приложение В

СООБЩЕНИЯ

Они появляются в нижней части экрана, если компьютер остановился при выполнении некоторого оператора Бейсика, и указывают причину, вызвавшую останов.

Сообщение содержит кодовый номер или букву. Краткое содержание помогает найти ошибочную строку и ошибочный оператор в этой строке. (Команда указывается как строка 0, оператор 1 в этой строке располагается первым, оператор 2 после первого или тем и т.д.).

От состояния CONTINUE зависит очень многое в сообщениях. Обычно продолжение начинается с оператора, специфицированного в предыдущем сообщении, но имеются и исключения - сообщения 0, 9 и D. (См. также Приложение С).

Код	Значение	Ситуация
0	OK (О'кей! Порядок!) Успешное завершение или переход на строку с номером, большим чем всего имеется. Это сообщение не меняет строки или оператора, определенного для CONTINUE	Разное
1	NEXT WITHOUT FOR (NEXT без FOR) Управляющей переменной нет (не была определена в операторе FOR), но есть обычная переменная с тем же именем	NEXT
2	VARIABLE NOT FOUND (Переменная не найдена) Для простой переменной выдается, если она используется без предварительного определения в операторах LET, READ или INPUT, или загружается с ленты, или устанавливается в операторе FOR. Для индексируемых переменных сообщение выдается, если она не была предварительно определена в операторе DIM перед использованием или загрузкой с ленты	Разное
3	SUBSCRIPT WRONG (Ошибочный индекс) Индекс превышает размерность массива, либо ошибочное число задает индекс. Если индекс отрицательный либо больше 65535, то выдается сообщение 8	В индексной переменной или подстроке
4	OUT OF MEMORY (Вне памяти) В памяти недостаточно места для ваших действий. Вы можете освободить себе память, удалив командные строки, используя DELETE, затем удалить одну или две строки программы (с целью возврата их в последствии), получить дополнительную память, маневрируя оператором CLEAR	LET INPUT FOR DIM GO SUB LOAD MERGE

5	OUT OF SCREEN (Вне экрана) Если INPUT оператор генерирует больше, чем 23 строки в нижней половине экрана. Также встречается с PRINT AT 22, ...	INPUT PRINT
6	NUMBER TOO BIG (Число больше максимально допустимого) В результате вычислений получилось число, большее чем 10**38	Арифметические операции
7	RETURN WITHOUT GO SUB (RETURN без GO SUB) Встретилось больше операторов RETURN, чем было операторов GO SUB	RETURN
8	END OF FILE (Конец файла)	Операции с внешней памятью
9	STOP STATEMENT (Оператор STOP) После этого сообщения CONTINUE не может повторить STOP, но может передать управление на следующий оператор	STOP
A	INVALID ARGUMENT (Ошибочный аргумент) Аргумент функции не допустим в данной версии Бейсика	SQR LN EN CS USR (со строковым аргументом)
B	INTEGER OUT OF RANGE (Переполнение целого) Выдается, когда аргумент с плавающей точкой округляется к целому (для случая массивов см. также сообщение 3)	RUN RANDOM POKE GOTO GOSUB LIST LLIST PAUSE PLOT CHR$ PEEK USR (с числовым аргументом)
C	NONSENSE IN BASIC (Выражение не из Бейсика) Текст (строка) не распознается Бейсиком как допустимое выражение	VAL VAL$
D	BREAK-CONT REPEATS Клавиша BREAK нажата во время действия периферийной операции. Действия CONTINUE после этого оператора - обычные, т.е. те, что указаны в операторе. Сравните с сообщением L	LOAD, SAVE, VERIFY, MERGE, LPRINT, LLIST, COPY (при свертке)

M	OUT OF DATA (Вне данных) Попытка выдать READ, когда список данных в DATA кончился	READ
F	INVALID FILE NAME (Неверное имя файла) Оператор SAVE с пустой строкой вместо имени или с именем, которое длиннее 10 символов	SAVE
G	NO ROOM FOR LINE (Нет места для строки) Недостаточно места в памяти для записи очередной строки программы	Ввод строки в программу
H	STOP IN INPUT Некоторые введенные данные начинаются с оператора STOP или была нажата INPUT LINE. Действие CONTINUE – обычное	INPUT
I	FOR WITHOUT NEXT (FOR без NEXT) Цикл FOR ни разу не выполнялся, не найден NEXT оператор	FOR
J	INVALID I/O DEVICE (Неверное устройство ввода-вывода)	В операциях с внешними устройствами
K	INVALID COLOUR (Неверный цвет) Специфицированное число имеет неверное значение	INK PAPER BORDER FLASH BRIGHT INVERSE OVER также после одной из передач упр. символов
L	BREAK INTO PROGRAM (BREAK во время выполнения программы) Нажата клавиша BREAK: это обнаруживается между двумя операторами. Строка и номер оператора в строке указывают на оператор, выполняемый перед нажатием BREAK, но CONTINUE переходит к следующему оператору	Разное
M	RAMTOP NO GOOD (Адрес RAMTOP не годен) Число, указанное для RAMTOP, слишком велико или слишком мало	CLEAR возможно RUN
N	STATEMENT LOST (Оператор отсутствует) Переход к оператору, которого уже нет	RETURN NEXT RUN
O	INVALID STREAM (Ошибочный поток данных)	В операциях ввода-вывода

P	FN WITHOUT DEF (FN без DEF) Определяемая пользователем функция не определена в операторе DEF FN	FN
Q	PARAMETER ERROR (Ошибка в параметре) Ошибочное число аргументов или один из них не того типа, какой был указан при описании	FN
R	TAPE LOADING ERROR (Ошибка загрузки с ленты) Файл на ленте найден, но не может быть считан	VERIFY LOAD MERGE

Приложение С (часть 1)

ОПИСАНИЕ МИКРОКОМПЬЮТЕРА ZX SPECTRUM

КЛАВИАТУРА

Каждая клавиша клавиатуры компьютера ZX SPECTRUM имеет многофункциональное назначение и позволяет вводить как отдельные символы, так и целые слова. Действие, производимое клавишей, определяется частично переключающими клавишами (CAPS SHIFT и SIMBOL SHIFT), а частично - режимом, в котором находится компьютер.

Режим отображается курсором - мерцающей буквой, указывающей позицию, в которую будет вводиться очередной символ с клавиатуры.

Возможны следующие режимы:

K - (для ключевых слов) KEYWORDS.

Этот режим автоматически сменяет режим L, если компьютер переходит в ожидание ввода команды или строки программы. Это может быть либо в начале строки, либо после THEN, либо после ":", и если не было нажатия переключающих клавиш, то нажатие любой клавиши будет интерпретироваться как ключевое слово (написанное на клавише) или цифра.

L - (для букв) LETTER.

Основной режим для компьютера. Если не было переключения регистров, то клавиша интерпретируется как основной символ, нанесенный на эту клавишу.

Для обоих режимов L и K при нажатии с клавишей одновременно и клавиши SYMBOL SHIFT, клавиша будет интерпретироваться как вспомогательный символ, а при нажатии CAPS SHIFT с цифрой, клавиша будет интерпретироваться как управляющая функция, написанная на белом поле клавиши. Нажатие CAPS SHIFT с любой из клавиш не вызывает ключевого слова в режимах K и L.

C - (для заглавных букв) CAPITAL.

Режим представляет собой вариант режима L, в котором используются заглавные буквы.

CAPS LOCK используется для перехода из режима L в C и обратно.

E - (для расширения) EXTEND.

Используется для получения дальнейших символов, главным образом - знаков. Этот режим вводится одновременным нажатием двух переключающих клавиш с удержанием затем только одной клавиши. В

этом режиме клавиша дает один символ или знак (изображенный на зеленом поле клавиши), если не нажата переключающая клавиша, или знак, изображенный на красном поле, если удерживается управляющая клавиша. Цифровые клавиши выдают знак, если нажимаются вместе с SYMBOL SHIFT, в противном случае они выдают последовательность управления цветом.

G - (для графики) GRAPHICS.

Режим вводится после нажатия GRAPHICS (CAPS SHIFT и 9) и сохраняется до следующего нажатия этой клавиши. Цифровые клавиши будут выдавать мозаичные графические символы, сохраняя GRAPHICS и DELETE, а каждая алфавитная клавиша, кроме V,W,X,Y и Z, будет выдавать определенный пользователем графический символ.

Если некоторая клавиша будет удерживаться более 2-3-х секунд, это вызовет повторение производимого ею действия.

Ввод с клавиатуры производится в нижнюю часть экрана. Каждый символ (или составной знак) вставляется перед курсором. Курсор может быть переслан влево (действием CAPS SHIFT и 5). Целая строка может быть удалена вводом EDIT (CAPS SHIFT и 1) и последующим нажатием ENTER. Когда нажимается ENTER, выполняется набранная строка, либо она вводится в программу, либо она используется как входные данные для INPUT-операторов. Если в строке имеются синтаксические ошибки, в этом случае нажатием [?] происходит переход на ошибку.

Когда строки программы введены, листинг отображается в верхней части экрана. Более подробно этот процесс описан в главе 2.

Последняя введенная строка называется текущей и отмечается символом [>]. Ее можно изменить, используя клавиши перемещения курсора вверх и вниз (CAPS SHIFT и 6; CAPS SHIFT и 7). Если введен EDIT (CAPS SHIFT и 1), то текущая строка переносится в нижнюю часть экрана и становится доступной для редактирования.

Если выполняется команда или целая программа, то результат отображается в верхней части экрана и сохраняется до ввода строки программы, ввода пустой строки или нажатия клавиш управления курсором вверх, вниз. В нижней части экрана выдаются сообщения, приведенные в Приложении в. В сообщении указывается номер ошибочной строки (для команды) и позиция оператора в этой строке. Сообщение сохраняется на экране до нажатия любой клавиши (отображается переходом в режим K).

В определенных обстоятельствах CAPS SHIFT и SPACE действуют как BREAK, останавливая компьютер с сообщениями D или L. При этом до останова:

а) завершается выполнявшийся оператор или команда;

б) завершаются действия, выполняемые компьютером с магнитофоном или принтером.

Пример использования клавиатуры:

```
   !
 ┌─────┐
 │  LN │
 │Z  : │
 │COPY │
 └─────┘
  BEEP
```

При курсоре [K]:
1. Простое нажатие – на экране ⟶ COPY
2. SYMBOL SHIFT и клавиша – на экране ⟶ Z

При курсоре [L]:
3. CAPS SHIFT и клавиша – на экране ⟶ :
4. BREAK/SPACE и клавиша – на экране ⟶ Z

SYMBOL SHIFT и CAPS SHIFT – курсор ⟶ [E]

При курсоре [E]:
5. Простое нажатие клавиши – на экране ⟶ LN
6. CAPS SHIFT и клавиша – на экране ⟶ Z

ЭКРАН ТЕЛЕВИЗОРА

Экран телевизора содержит 24 строки по 32 позиции в каждой и делится на две части. Верхняя часть в 22 строки отображает листинг или вывод из программы. Когда вывод в верхней части достигает низа, необходима свертка на одну строку, при этом может захватываться строка, которую вам хочется сохранить. Компьютер в этом случае останавливается с запросом "SCROLL?". Если теперь нажать клавиши N, SPACE или STOP, то программа остановится с выдачей сообщения "D BREAK-CONT REPEATS". Нажатие других клавиш разрешает свертку и продолжение выполнения.

Нижняя часть используется для ввода команд, строк программы и входных INPUT-данных, а также для отображения сообщений. Нижняя часть экрана состоит из двух строк (верхняя из них чистая - для расширения). При переполнении верхней строки осуществляется свертка.

Каждая позиция имеет атрибуты, определяющие ее как чистую (цвет фона), либо как закрашенную (основной цвет), с повышенной или с пониженной яркостью, мерцающую или нет.

Доступны цвета: черный, голубой, красный, пурпурный (фиолетовый), зеленый, желтый и белый.

Края экрана могут быть установлены в определенный цвет использованием оператора BORDER.

Каждая позиция подразделяется на 8x8 точек, а графика символов обеспечивается индивидуальным определением каждой точки. Атрибуты каждой позиции настраиваются при записи символа или при установке точки (PIXEL). Способ настройки определяется параметрами вывода, имеющими две установки (постоянную и временную) в шести операторах:

PAPER, INK, FLASH, BRIGHT, INVERSE и OVER.

Постоянные параметры для верхней части экрана устанавливаются в операторах PAPER, INK и т.д. Обычно они имеют черный цвет для закрашенной точки (INC) и белый - для фоновой (PAPER), нормальную яркость, не мерцающие, не инверсные. Постоянные параметры для нижней части экрана используют цвет рамки (BORDER COLOUR) как цвет фона (незакрашенный), с черным или белым цветом, нормальную яркость, не мерцающие.

Временные параметры устанавливаются командами

PAPER, INK и т.д.,

вставляемыми в операторы

PRINT, LPRINT, INPUT, PLOT, DRAW, CIRCLE, а также
PAPER, INK и тому подобными управляющими символами,

когда они выводятся на телевизор.

Переменные параметры сохраняются до конца действия оператора PRINT (или других).

Параметры PAPER и INK могут принимать значения от 0 до 9. Значения от 0 до 7 определяют цвет выводимого символа:

0 - черный	(BLACK)
1 - голубой	(BLUE)
2 - красный	(RED)
3 - пурпурный	(MAGENTA)
4 - зеленый	(GREEN)

5 - синий (CYAN)
6 - желтый (YELLOW)
7 - белый (WHITE)

Значение 8 определяет, что цвет должен остаться при выводе без изменения.

Значение 9 (контрастность) определяет, что цвет должен стать либо белым, либо черным для выделения его от других цветов.

Параметры FLASH и BRIGHT могут принимать значения 0,1 или 8. Значение 1 указывает, что включается повышенная яркость и мерцание. Значение 0 указывает, что повышенная яркость и мерцание отключаются. Значение 8 указывает, что все остается без изменений.

Параметры OVER и INVERSE могут принимать значения 0 или 1:

OVER 0 - новый символ затирает старый;
OVER 1 - код старого символа и нового символа соединяются операцией "Исключающего ИЛИ", образуя новый символ (OVER PRINTING);
INVERSE 0 - новый символ печатается в неинверсном (позитивном) виде;
INVERSE 1 - новый символ печатается в инверсном (негативном) виде.

Когда на телевизор передается управляющий символ TAB, то два старших байта используются для спецификации TAB STOP N (первый байт является старшим). Это обеспечивается прогоном от 32 до "N" (указанном в TAB) и затем выводом нужного количества пробелов для смещения текущей позиции вывода в колонку "N".

Если на вывод передается запятая как управляющий символ, то выводится нужное количество пробелов для перевода текущей позиции вывода в позицию 0 или 16.

Если передается управляющий символ ENTER, то позиция вывода переводится на следующую строку.

П Р И Н Т Е Р

Вывод на принтер осуществляется через буфер длиной в 32 символа. Очередная строка выдается из буфера на принтер в следующих случаях:

а) когда окончен вывод одной строки и вывод переходит к другой строке;
б) при передаче в буфер символа ENTER;
в) при завершении программы, если еще остались другие невыведенные данные;
г) если встретились управляющие символы TAB или запятая, требующие перевода строки.

Управляющие символы TAB и запятая производят вывод пробелов при работе с телевизором.

Управляющий символ AT изменяет позицию вывода, используя число, задающее позицию.

Принтер также правильно реагирует на управляющие символы INVERSE, OVER (и операторы с тем же именем), но не воспринимает PAPER, INK, FLASH и BRIGHT.

При вводе BREAK принтер останавливается с выдачей сообщения "B". При отсутствии принтера вывод просто не осуществляется.

Приложение С(часть 2)

ЯЗЫК ПРОГРАММИРОВАНИЯ БЕЙСИК (СПРАВОЧНОЕ ПОСОБИЕ)

Все числа в системе могут иметь точность 9 или 10 знаков. Наибольшее число 10**38, наименьшее положительное число 4*10**-39. Числа имеют внутреннее представление как числа с плавающей (двоичной) точкой с выделением одного бита на показатель степени "в" (экспоненты) в интервале от 1 до 255 и четырех байтов на мантиссу "м" в интервале от 0.5 до 1 (м<1), т.е. представляются числом м*2**в(-128).

Поскольку 1/2<м<1, старший значащий бит мантиссы всегда 1. Следовательно, мы можем заменить его на бит, равный 0 для положительного числа и 1 - для отрицательного.

Наименьшее целое имеет специальное представление, в котором первый байт - 0, второй - байт знака (0 и FFH), а третий и четвертый - само число в дополнительном коде (младшие значащие цифры в первом байте).

Числовые переменные имеют имя произвольной длины, начинающееся с буквы и продолжающееся буквами или цифрами. Пробелы и символы управления цветом игнорируются, и все буквы преобразуются к минимально упакованному виду.

Управляющие переменные для FOR-NEXT циклов имеют имена длиной в одну букву.

Числовые массивы имеют имена длиной в одну букву, которая может быть такой же, как имя скалярной переменной. Эти массивы могут иметь произвольное количество измерений и произвольный размер. Начальный индекс всегда равен 1.

Строки символов - более гибкие в своей длине. Имя строковой переменной в отличие от простой переменной заканчивается знаком доллара ($).

Строковые массивы также могут иметь произвольное количество измерений и размер. Их имена представляют собой одну букву и следующий за ней символ $, но не могут совпадать с именем простой строки символов.

Все строки в массивах имеют фиксированную длину, которая определяется числом, заданным последней размерность в операторе DIM. Начальный индекс равен 1.

Подстрока от строки может быть получена как сечение. Сечение может быть:

а) пустым;

б) числовым выражением;

в) некоторым "числовым выражением" "то" другим "числовым выражением" и использоваться в

*) строковых выражениях (сечениях);

**) строковых массивах переменных (индекс 1, индекс 2, ..., индекс м, сечение) или, что тоже самое, (индекс 1, индекс 2,..., индекс м) (сечение).

В случае *) строка выражения имеет значение а$. Если сечение массива пусто, то а$ считается подстрокой от самой себя.

Если сечение представляется числовым выражением со значением "м", то результатом будет м-й символ от а$ (подстрока длиной 1(а, если сечение представлено в форме в)) и первое числовое выражение

имеет значение "M" (умалчиваемое значение 1), а второе "N" (умалчиваемое значение S$), и если 1≤M≤N≤ чем длина S$, то результатом будет подстрока от S$ с M-ым начальным и с N-ым конечным символами.

Если 0≤N<M, то результатом будет пустая строка. В любом другом случае выдается сообщение об ошибке "3".

Сечение выполняется перед функцией или операцией, которая осуществляется, если скобками не предписано сделать иначе.

Подстрока может назначаться (смотри оператор LET). Если часть строки записывается в строковый литерал, она должна удваиваться.

Ф У Н К Ц И И

Имя функции	Тип аргумента	Действие (возвращаемое значение)
ABS	Число	Абсолютное значение
ACS	Число	Арккосинус в радианах. Выдает сообщение об ошибке A, если X не лежит в пределах от -1 до 1
AND	Логическая операция. Правый операнд всегда число. Слева может быть: - число - строка	A AND B= { A, если B≠0 ; B, если B=0 } A$ AND B= { A$, если B≠0 ; " ", если B=0 }
ASN	Число	Арксинус в радианах. Выдает сообщение, если X не лежит в интервале от -1 до 1
ATN	Число	Арктангенс в радианах
ATTR	Два числовых аргумента X и Y, заключенные в скобки	Число, двоичный код которого представляет собой атрибуты Y-ой позиции X-ой строки экрана. Бит 7 (старший) равен 1 для мерцающего поля и 0 для немерцающего. Биты с 5 по 3 - цвет фона. Биты 1 и 2 - цвет закрашивания. Выдает сообщение B, если 0≤X≤23 и 0≤Y≤31
BIN		Это не обычная функция. За BIN записывается последовательность нулей и единиц, представляющая собой двоичное представление числа, которое записывается в память

CHR$	Число	Символ, код которого представим числом X, округленным до ближайшего целого
CODE	Строка символов	Код первого символа в строке X (или 0, если X - пустая строка)
COS	Число в радианах	Косинус X
EXP	Число	E в степени X
FN		FN с последующим именем определенной пользователем функции (см. DEF). Аргументы заключаются в скобки. Даже если нет аргументов, скобки все равно должны записываться
IN	Число	Осуществляется ввод на уровне микропроцессора из порта X (0≤X≤FFFFH). Загружается пара регистров BC и выполняется команда ассемблера IN A(C)
INKEY$	Нет	Чтение с клавиатуры. Возвращает символ, введенный с клавиатуры (в режиме [L] или [C]), если было действительное нажатие клавиши, или пустую строку - в противном случае
INT	Число	Округленное к ближайшему меньшему целому
LEN	Строка символов	Длина строки
LN	Число	Натуральный логарифм. Выдает сообщение A, если X≤0
NOT	Число	0, если X≠0; 1, если X=0. Операция имеет четвертый приоритет
OR	Логическая операция. Оба операнда - числа	A OR B = { 1, если B≠0; A, если B=0 } Операция имеет второй приоритет
PEEK	Число	Значение байта в памяти по адресу X, округленному к ближайшему целому
PI	Нет	Число Пи (3,14159265...)
POINT	Два числ. аргумента X и Y, заключенные в скобки	1, если точка экрана с координатами (X,Y) закрашена. 0, если эта точка имеет цвет фона. Выдает сообщение B, если не выполняются условия 0≤X≤255 и 0≤Y≤175

RND	Нет	Очередное псевдослучайное число из псевдослучайной последовательности, получаемой возведением в 75-ю степень модуля числа 65537, вычитанием 1 и делением на 65536. Число лежит в интервале 0≤X≤1
SCRN$	Два числ. аргумента X и Y, заключенные в скобки	Символ (обычный или инверсный), который появляется на экране в строке X, позиции Y. Дает пустую строку, если символ не опознан
SGN	Число	-1, если X<0 0, если X=0 1, если X>0
SIN	Число в радианах	Синус x
SQR	Число	Корень квадратный. Выдает сообщение A, если X<0
STR$	Число	Строка символов, которая должна быть отображена, если выводится x
USR	Число	Вызывает подпрограмму в машинных кодах, начальный адрес которой x. При возврате результатом будет содержимое регистровой пары BC
USR$	Строка символов	Адрес группы байтов, задающих определенный пользователем символ для закрепления его за x
VAL	Строка символов	Вычисление x как числового выражения. Выдает сообщение C, если x содержит синтаксические ошибки или дает строковое (не числовое) значение. Возможны и другие ошибки
VAL$	Строка символов	Вычисляет x как строковое выражение. Выдает сообщение C, если x содержит синтаксическую ошибку или дает не строковое (числовое) значение

О П Е Р А Ц И И

Префиксные:

\- число отрицательное значение

Инфиксные (двухоперандовые):

\+ сложение для чисел, конкатенция для строк
\- вычитание

*	умножение	
/	деление	
**	возведение в степень (стрелка вверх). Сообщение в, если левый операнд отрицательный	
=	равенство	оба операнда должны быть одного типа. Результат равен 1, если сравнение истинно, и равен 0, если нет
>	больше	
<	меньше	
>=	больше или равно	
<=	меньше или равно	
<>	не равно	

Функции и операции имеют следующие приоритеты:

индексация и сечения	- 12
все функции за исключением NOT и префиксного минуса	- 11
возведение в степень	- 10
префиксный минус	- 9
*,/	- 8
+,-(вычитание)	- 6
=, >, <, <=, >=, <>	- 5
NOT	- 4
AND	- 3
OR	- 2

О П Е Р А Т О Р Ы

Принятые обозначения:

A - одна буква;
V - переменная;
X,Y,Z - числовые выражения;
M,N - числовые выражения, которые округляются к ближайшему целому;
E - некоторое выражение;
F - выражение, имеющее строковое значение;
S - последовательность операторов, разделенных двоеточием ":";
C - последовательность символов управления цветом. Каждый заканчивается запятой или точкой с запятой. Цветовой символ имеет форму операндов PAPER, INK, FLASH, BRIGHT, INVERSE или OVER.

Текст произвольного выражения может располагаться в любом месте строки (за исключением номера строки, который должен размещаться в начале строки).

Все операторы, кроме INPUT, DEF и DATA, могут использоваться и как команды, и в программах.

Команда или строка программы может содержать несколько операторов, разделенных двоеточием (":"). Нет ограничений на положение операторов в строке, хотя есть некоторые ограничения в IF и REM.

Все операторы языка сведены в следующую таблицу.

Оператор	Действие оператора
BEEP X,Y	Воспроизводит звук длительностью X сек и высотой Y полутонов вверх от основного тона До (или вниз, если Y отрицательное)

BORDER M	Устанавливает цвет рамки (бордюра) экрана. Выдает сообщение об ошибке K, если O>M>7
BRIGHT M	Устанавливает яркость выводимого символа: 0 - для обычной яркости; 1 - для повышенной яркости; 8 - сохраняет существующую яркость
CAT	Без MICRODRIVE не работает
CIRCLE X,Y,Z	Изображает дугу или окружность с центром в точке с координатами (X,Y) и радиусом Z
CLEAR	Уничтожает все переменные и очищает занимаемую ими память. Выполняет RESTORE и CLS, устанавливает PLOT позицию в нижнюю левую точку экрана и очищает GO SUB стек
CLEAR N	Подобно CLEAR, но дополнительно изменяет системную переменную RAMTOP на "N" и задает новый GO SUB стек
CLOSE	Без MICRODRIVE не работает
CLS	(CLEAR SCREEN) Очищает файл экрана
CONTINUE	Продолжает выполнение программы, начатой ранее и остановленной сообщением, отличным от 0. Если сообщение 9 или 0, то выполнение продолжается со следующего оператора. В других случаях с того оператора, где случилась ошибка. Если сообщение возникло в командной строке, то CONTINUE вызовет попытку повторить командную строку и перейти в цикл, если было сообщение 0:1, дает сообщение 0, если было 0:2, или дает сообщение N, если было 0:3 или более. В качестве CONTINUE используется ключевое слово CONT на клавиатуре
COPY	Пересылает копию 22 строк экрана на принтер, если он подключен. Помните, что по COPY нельзя распечатать находящийся на экране автоматический листинг. Выдает сообщение B, если нажать клавишу BREAK
DATA E1,E2,E3,...	Часть списка данных должна располагаться в программе

DEF FN A(A1,A2,..Ak)=E	Определяемая пользователем функция. Должна располагаться в программе. A,A1,A2 и т.д. - единственные буквы или буквы и $ для строковых аргументов. Используется форма DEF FN A(), если нет аргументов
DELETE F	Без MICRODRIVE не работает
DIM A(N1,N2,...,Nk)	Уничтожает массив с именем "A", устанавливает числовой массив "A" с "k" измерениями и присваивает всем его элементам значение 0
DIM A$(N1,N2,...,Nk)	Уничтожает массив или строку с именем "A$", устанавливает символьный массив с "k" измерениями и присваивает всем его элементам значение " ". Массив может быть представлен как массив строк фиксированной длины Nk с k-1 размерностью. Сообщение 4 выдается, если недостаточно места для размещения массива. Массив не определен до его описания в операторе DIM
DRAW X,Y	Как и DRAW X,Y,0 чертит прямую линию
DRAW X,Y,Z	Изображает линию от текущей графической позиции в точку с приращениями X,Y по дуге в Z радиан. Выдает сообщение 8 при выходе за пределы экрана
ERASE	Без MICRODRIVE не работает
FLASH N	Определяет будет ли символ мерцающим или с постоянным свечением. N=0 для постоянного свечения, N=1 - для мерцания, N=8 - для сохранения предыдущего состояния
FOR A=X TO Y	FOR A=X TO Y STEP 1
FOR A=X TO Y STEP Z	Уничтожает скалярную переменную "A" и устанавливает управляющую переменную "X", предел "Y", шаг приращения "Z", зацикливает адрес оператора, указанный в утверждении после FOR. Если начальное приращение больше (если STEP ≥0), чем предел, или меньше (если STEP<0), чем предел, то происходит переход к утверждению NEXT A или выдача сообщения 1, если нет (см.NEXT). Сообщение 4 выдается, если недостаточно места для размещения "X"

FORMAT F	Без MICRODRIVE не работает
GO SUB N	Проталкивает строку с оператором gosub в стек для использования затем как GOTO N. Выдает сообщение 4, если не все программы завершились RETURN
GO TO N	Продолжает выполнение программы со строки "N". Если "N" опущено, то с первой строки после этой команды
IF X THEN S	Если "x" истинно (не равно 0), то выполняется "s". "s" включает все операторы до конца строки. Форма IF x THEN "номер строки" недопустима
INK N	Устанавливает цвет закрашивания (т.е. цвет, которым будут изображаться символы на цвете фона). "N" в интервале от 0 до 7 указывает цвет. N=8 - оставить цвет без изменения. N=9 - увеличение контраста. Выдает сообщение K, если "N" не лежит в интервале 0 до 9
INPUT ...	Здесь "..." есть последовательность вводимых символов, разделяемых как в операторе PRINT запятыми, точками с запятыми или апострофами. Вводимыми символами могут быть: а) некоторый PRINT-символ, начинающийся не с буквы; б) имя переменной; в) строка имен переменных строкового типа. INPUT-символы в случае а) представляются также, как и в операторе PRINT, за исключением того, что они все выводятся в нижнюю часть экрана. В случае б) компьютер останавливается и ждет ввода некоторого выражения с клавиатуры, значение которого будет присвоено переменной. Ввод осуществляется обычным образом, а синтаксические ошибки выводятся мерцающим [?]. Для строкового выражения вводный буфер устанавливается для размещения двух таких строк (который при необходимости может быть увеличен). Если первый вводимый символ STOP, то програма останавливается с сообщением H. Случай в) подобен случаю б) с той лишь разницей, что вводимая информация представляет собой

	строковый литерал неограниченной длины, и STOP в этом случае не срабатывает. Для останова вы должны нажать клавишу "курсор вниз"
INVERSE N	Символ управления инверсией выводимого символа. Если N=0, символ выводится в обычном виде с прорисовкой цветом закрашивания (INK) на цвете фона (PAPER). Если N=1, то цветовое решение изображения символа меняется на обратное. Смотри Приложение в. Выдает сообщение K, если "N" не 0 и не 1
LET V=E	Присваивает значение "E" переменной "V". Ключевое слово LET не может быть опущено. Скалярная переменная не определена, пока не встретится в операторах LET, READ или INPUT. Если "V" - индексируемая строковая переменная или сечение строкового массива (подстрока), то присваивание происходит с усечением или дополнением пробелами до фиксированной длины
LIST	То же, что и LIST 0
LIST N	Записывает текст программы в верхнюю часть экрана, начиная с первой строки, меньшей чем "N", и делает "N" текущей строкой
LLIST	То же, что и LLIST 0
LLIST N	Подобо LIST, но вывод осуществляется на принтер
LOAD F	Загружает программу и переменные
LOAD F DATA ()	Загружает числовой массив
LOAD F DATA$ ()	Загружает строковый массив
LOAD F CODE M,N	Загружает старшие "M" байтов, начиная с адреса "N"
LOAD F CODE M	Загружает байты, начиная с адреса "M"
LOAD F CODE	Загружает байты по тому же адресу, с которого они были записаны
LOAD F SCREEN$	Аналогично LOAD F CODE 16384,6912

	очищает файл экрана и загружает его с кассетного магнитофона. См. главу 20
LPRINT	Подобно PRINT, но использует принтер
MERGE F	Подобно LOAD F, но не затирает всю программу в памяти, а заменяет только те строки и переменные, у которых совпадают номера или имена с такими же на ленте
NEW	Запускает по новой систему программирования Бейсик, уничтожая старую программу, переменные и используемую память, включая и байт адреса в системной переменной RAMTOP, но сохраняет системные переменные UDG, P RAMT, RASP И PIP
NEXT A	а) Находит управляющую переменную "A"; б) прибавляет к ней значение STEP; в) если STEP>0, а значение "A" стало больше значения "предел", или STEP<0, а значение "A" меньше, чем значение "предел", то происходит переход к оператору цикла. Сообщение 2 выдается, если не найдена переменная "A". Сообщение 1 выдается, если "A" не является управляющей переменной цикла
OPEN#	Без MICRODRIVE не работает
OUT M,N	Выводит байт "N" в порт "M". Операция выполняется на уровне микропроцессора (загружает в регистровую пару вс адрес "M" и выполняет команду ассемблера OUT (C),A). 0≤M≤65535. -255≤N≤255 - иначе выдается сообщение B
OVER N	Управляющий символ надпечатывания по выведенной строке. Если N=0, выводимый символ затирает находящийся в данной позиции. Если N=1, то новый символ соединяется со старым, образуя закрашивающий цвет при условии, что старший символ имел указание цвета, отличное от старого, или цвет фона, если оба указывают на один и тот же цвет (либо фона, либо закрашивания). См. Приложение B

PAPER N	Подобен INK, но управляет цветом фона
PAUSE N	Останавливает выполнение программы и задерживает изображение на экране на "N" кадров (50 кадров в сек – частота кадровой развертки) или до нажатия любой клавиши. 0≤N≤65535, иначе выдается сообщение B. При N=0 величина задержки не учитывается и продолжается до первого нажатия клавиши
PLOT C;M,N	Выводит точку закрашивающего цвета (обработанную OVER и INVERSE) с координатами (ABS(M),ABS(N)). Смещает графическую (PLOTPOSITION) позицию. Если цветовой символ "C" не специфицирован иначе, то цвет закрашивания в позиции, где расположена эта точка, изменяется на текущий сплошной цвет и другие указания (цвет фона, мерцание, яркость) остаются без изменения. 0≤M≤65535, -255≤N≤255, иначе выдается сообщение B
POKE M,N	Записывает значение "N" в байт памяти "M". 0≤M≤65535, -255≤N≤255, иначе выдается сообщение B
PRINT ...	Здесь "..." есть последовательность PRINT-символов, разделяемых запятыми, точками с запятыми или апострофами, которые выводятся в экранный файл для отображения на экране телевизора. Точка с запятой сама действия не вызывает, а используется для разделения символов. Запятая порождает управляющий символ ENTER. В конце оператора PRINT, если он не заканчивается точкой с запятой, запятой или апострофом, автоматически выводится символ ENTER. а) Пустая строка (т.е. ничего). б) Числовое выражение. Если значение выражения отрицательно, то выводится знак минус. Если X<10**(-5) или X≥10**13, вывод осуществляется в показательной форме. Мантисса представляется 8-мью цифрами (с нормализацией) и десятичной точкой (отсутствует только тогда, когда в мантиссе одна цифра) после первой цифры. Показатель степени – после буквы "E" с последующим знаком и двумя цифрами порядка. Иначе X вы-

	водится как обычное десятичное число с 8-мью значащими цифрами. в) Строковое выражение. В строке возможны пробелы до и после символов. Управляющие символы вызывают определяемое ими действие. Неотображаемые на экране символы выводятся как "?". г) AT M,N. Вывод в строку "M", позицию "N". д) TAB N. Вывод управляющего символа TAB с последующими двумя байтами "N" (первый байт старший). Вызывает TAB-останов. е) Цветовой символ в форме PAPER, INC, BRIGHT, INVERSE или OVER оператора
RANDOMIZE N	Устанавливает системную переменную SEED, используемую для вычисления очередного значения функции RND. Если N≠0, то SEED принимает значение "N". Если N=0, то SEED принимает значение другой системной переменной FRAMES, подсчитывающей кадры, отображаемые на экране, что обеспечивает вполне случайное число. Оператор запускает сокращение RAND (см. клавишу). Выдает сообщение B, если "N" не лежит в пределах от 0 до 65535
READ V1,V2,...,Vk	Присваивает переменным одной за другой значения, последовательно представленные в списке DATA
REM ...	Не выполняется. "..." может быть последовательностью символов (исключая ENTER). Может включать двоеточие (":") для указания отсутствия операторов в строке REM
RESTORE	То же самое, что и RESTORE 0
RESTORE N	Перезаписывает указатель данных в первый оператор DATA в строке, меньшей чем "N". Следующий оператор READ начнет считывание отсюда
RETURN	Ссылается на оператор GOSUB в стеке и передает управление строке, расположенной после него. Выдает сообщение 7, если нет указанного оператора в стеке. Характерная ошибка - операто-

	ры GOSUB не сбалансированы операторами RETURN
RUN	То же самое, что и RUN 0
RUN N	CLEAR и затем GO TO N
SAVE F	Записывает на ленту программу и переменные
SAVE F LINE M	Записывает на ленту программу и переменные таким образом, что при загрузке программа автоматически выполняется со строки "M"
SAVE F DATA ()	Запись на ленту числового массива
SAVE F DATA$ ()	Запись на ленту строкового массива
SAVE F CODE M,N	Записывает на ленту "M" байтов, начиная с адреса "N"
SAVE F SCREEN$	Аналогично SAVE F CODE 16384,6912. Выдает сообщение F, если "F" - пустая строка или имеет длину более 10. Смотри главу 20
STOP	Останавливает выполнение программы с выдачей сообщения 9. CONTINUE (продолжение) будет осуществляться со следующего оператора
VERIFY	То же, что и LOAD, за исключением того, что данные загружаются в ОЗУ, но сравниваются с находящимися там. Выдает сообщение B, если обнаружен хотя бы один несовпадающий байт

Приложение D

СИСТЕМА КОМАНД МИКРОПРОЦЕССОРА ZILOG Z-80

Обозначения:

[R], [R'] регистр, 8 бит
A=111 / B=000 / C=001 / D=010 / E=011 / H=100 / L=101

DD, QQ, FP, RR регистровая пара, 16 бит

DD: BC=00 / DE=01 / HL=10 / SP=11
QQ: BC=00 / DE=01 / HL=10 / AF=11
FP: BC=00 / DE=01 / IX=10 / SP=11
RR: BC=00 / DE=01 / IY=10 / SP=11

T [R], (HL), (IX+DIS) ИЛИ (IY+DIS)
N данные 8 бит / NN данные 16 бит
DIS адресное смещение 8 бит
(при -128+127 при базовой адресации, -126+129 при относительной адресации)
[B] позиция бита в байте D0=000...D7=111
ccc условие ветвления NZ=000 Z=001
NC=010 C=011
PE=100 PE=101
P=110 M=111
[P] адрес рестарта, [P]=00H,08H,10H,18H,20H,28H,30H,38H

Мнемоника	T	Код	Действие	C Z P S N H
Пересылки, 8 бит				
LD [R],[R']	4	01[R] [R']	[R]:=[R']	. . . . .
LD [R],(HL)	7	01[R] 110	[R]:=(HL)	. . . . .
LD (HL),[R]	7	01 110[R]	(HL):=[R]	. . . . .
LD [R],N	7	00 [R]110 - N -	[R]:=N	. . . . .
LD (HL),N	10	00110110 - N -	(HL):=N	. . . . .
LD [R],(IX+DIS)	19	11011101 01[R] 110 - DIS -	[R]:=(IX+DIS)	. . . . .
LD [R],(IY+DIS)	19	11111101 01[R] 110 - DIS -	[R]:=(IY+DIS)	. . . . .
LD (IX+DIS),[R]	19	11011101 01110 [R] - DIS -	(IX+DIS):=[R]	. . . . .
LD (IY+DIS),[R]	19	11111101 01110 [R] - DIS -	(IY+DIS):=[R]	. . . . .
LD (IX+DIS),N	19	11011101 00110110 - DIS - - N -	(IX+DIS):=N	. . . . .
LD (IY+DIS),N	19	11111101 00110110 - DIS - - N -	(IY+DIS):=N	. . . . .
LD A,(BC)	7	00010100	A:=(BC)	. . . . .
LD A,(DE)	7	00110100	A:=(DE)	. . . . .
LD A,(NN)	13	00111010 - N - - N -	A:=(NN)	. . . . .
LD (BC),A	7	00000010	(BC):=A	. . . . .
LD (DE),A	7	00010010	(DE):=A	. . . . .
LD (NN),A	13	00110010 - N - - N -	(NN):=A	. . . . .

LD A,I	9	11101101 01010111	A:=I	. ? IF? . .
LD A,R	9	11101101 01011111	A:=R	. ? IF? . .
LD I,A	9	11101101 01000111	I:=A	. . . . .
LD R,A	9	11101101 01001111	R:=A	. . . . .
Пересылки, 16 бит				
LD DD,NN	10	00 DD001 - N - - N -	DD:=NN	. . . . .
LD IX,NN	14	11011001 00100001 - N - - N -	IX:=NN	. . . . .
LD IY,NN	14	11011101 00100101 - N - - N -	IY:=NN	. . . . .
LD HL,(NN)	16	00101010 - N - - N -	H:=(NN+1) L:=(NN)	. . . . .
LD DD,(NN)	20	11101101 01 DD1011 - N - - N -	DD:=(NN)	. . . . .
LD IX,(NN)	20	11011101 00101010 - N - - N -	IX:=(NN)	. . . . .
LD IY,(NN)	20	11111101 00101010 - N - - N -	IY:=(NN)	. . . . .
LD (NN),HL	16	00100010 - N - - N -	(NN):=HL	. . . . .
LD (NN),DD	20	11101101 01DD0 011 - N - - N -	(NN):=DD	. . . . .
LD (NN),IX	20	11011101 00100010 - N - - N -	(NN):=IX	. . . . .
LD (NN),IY	20	11111101 00100010 - N - - N -	(NN):=IY	. . . . .

Команды работы со стеком				
LD SP,HL	6	11111001	SP:=HL	. . . . .
LD SP,IX	10	11011101 11111001	SP:=IX	. . . . .
LD SP,IY	10	11111101 11111001	SP:=IY	. . . . .
PUSH QQ	11	11 QQ0101	(SP)-:=QQ	. . . . .
PUSH IX	15	11011101 11100101	(SP)-:=IX	. . . . .
PUSH IY	15	11111101 11100101	(SP)-:=IY	. . . . .
POP QQ	10	11QQ0 001	QQ:=(SP)+	. . . . .
POP IX	14	11011101 11100001	IX:=(SP)+	. . . . .
POP IY	14	11111101 11100001	IY:=(SP)+	. . . . .
Команды обмена				
EX DE,HL	4	11101011	DE:==:HL	. . . . .
EX AF,AF'	4	00001000	AF:==:AF'	. . . . .
EXX	4	11011001	BC:==:BC' DE:==:DE' HL:==:HL'	. . . . .
EX (SP),HL	19	11100011	HL:==:(SP)	. . . . .
EX (SP),IX	23	11011101 11100011	IX:==:(SP)	. . . . .
EX (SP),IY	23	11111101 11100011	IY:==:(SP)	. . . . .
Команды обработки блоков данных				
LDI	16	11101101 10100000	(DE):=(HL) HL:=HL+1 DE:=DE+1 BC:=BC-1	. . ? . 0 0 BC=0 P=0 BC<>0 P=1
LDIR	21 /16	11101101 10110000	WHILE BC<>0 DO (DE):=(HL) DE:=DE+1 HL:=HL+1 BC:=BC-1 END	. . 0 . 0 0
LDD	16	11101101 10101000	(DE):=HL HL:=HL-1 DE:=DE-1 BC:=BC-1	. . ? . 0 0 BC=0 P=0 BC<>0 P=1
LDR	21 /16	11011011 10111000	WHILE BC<>0 DO (DE):=(HL) DE:=DE-1 HL:=HL-1 BC:=BC-1 END	. . 0 . 0 0

CPI	16	11101101 10100001	A-(HL)=? HL:=HL+1 BC:=BC-1	. ? ? ? 1 ? A-(HL)=>Z BC=>P
CPIR	21 /16	11101101 10110001	A-(HL)=? HL:=HL+1 BC:=BC-1 IF BC<>0 AND A-(HL)<>0 THEN BEGIN	. ? ? ? 1 ? A-(HL)=>Z. BC=>P
CPR	16	11011011 10101001	A-(HL)=? HL:=HL-1 BC:=BC-1	. ? ? ? 1 ? A-(HL)=>Z BC=>P
CPDR	21 /16	11101101 10111001	A-(HL)=? HL:=HL-1 BC:=BC-1 IF BC<>0 AND A-(HL)<>0 THEN BEGIN	. ? ? ? 1 ? A-(HL)=>Z BC=>P
Арифметические и логические команды 8 бит				
ADD A,[R]	4	10000 [R]	A:=A+[R]	? ? V ? 0 ?
ADD A,N	7	11000110 - N -	A:=A+N	? ? V ? 0 ?
ADD A,(HL)	7	10000110	A:=A+(HL)	? ? V ? 0 ?
ADD A,(IX+DIS)	19	11011101 10000110 - DIS -	A:=A+(IX+DIS)	? ? V ? 0 ?
ADD A,(IY+DIS)	19	11111101 10000110 - DIS -	A:=A+(IY+DIS)	? ? V ? 0 ?
ADC A,S		001	A:=A+S+CY	? ? V ? 0 ?
SUB A,S		010	A:=A-S	? ? V ? 1 ?
SBC A,S		011	A:=A-S-CY	? ? V ? 1 ?
AND A,S		100	A:=A AND S	0 ? P ? 0 1
OR A,S		110	A:=A OR S	0 ? P ? 0 0
XOR A,S		101	A:=A XOR S	0 ? P ? 0 0
CMP A,S		111	A-S=?	? ? P ? 1 ?
INC [R]	4	00 [R] 100	[R]:=[R]+1	. ? V ? 0 ?
INC (HL)	11	00110 100	(HL):=(HL)+1	. ? V ? 0 ?
INC (IX+DIS)	23	11011101 00110100	(IX+DIS):= =(IX+DIS)+1	. ? V ? 0 ?
INC (IY+DIS)	23	11111101 00110100	(IY+DIS):= =(IY+DIS)+1	. ? V ? 0 ?
DEC T		101	T:=T-1	. ? V ? 1 ?
DAA	4	00100111	Десят.коррекц.	? ? P ? . ?
CPL	4	00101111	A:=NOT A	. . . . 1 1
NEG	8	11101101 01000100	A:=-A	? ? V ? 1 ?
CCF	4	00011111	CY:=NOT CY	? . . . 0 X
SCF	4	00110111	CY:=1	1 . . . 0 0

Арифметические команды 16 бит				
ADD HL,DD	11	00DD1 001	HL:=HL+DD	? . . . 0 X
ADC HL,DD	15	11101101 01DD1 010	HL:=HL+DD+CY	? ? V ? 0 X
SBC HL,DD	15	11101101 01DD0 010	HL:=HL-DD-CY	? ? V ? 1 X
ADD IX,PP	15	11011101 00PP1 001	IX:=IX+PP	? . . . 0 X
ADD IY,RR	15	11111101 00RR1 001	IX:=IX+RR	? . . . 0 X
INC DD	6	00DD0 011	DD:=DD+1	. . . . .
INC IX	10	11011101 00100011	IX:=IX+1	. . . . .
INC IY	10	11111101 00100011	IY:=IY+1	. . . . .
DEC DD	6	00DD1 011	DD:=DD-1	. . . . .
DEC IX	10	11011101 00101011	IX:=IX-1	. . . . .
DEC IY	10	11111101 00101011	IY:=IY-1	. . . . .
Команды сдвигов				
RLCA	4	00000111	CY=A7,A7=A6... ..A0=A7	? . . . 0 0
RLA	4	00010111	CY=A7,A7=A6... ..A0=CY	? . . . 0 0
RRCA	4	00001111	A7=A0,A6=A7... ..CY=A0	? . . . 0 0
RRA	4	00011111	A7=CY,A6=A7... ..CY=A0	? . . . 0 0
RLC [R]	8	11001011 00 000[R]		? ? P ? 0 0
RLC (HL)	15	11001011 00 000110		? ? P ? 0 0
RLC (IX+DIS)	23	11011101 11001011 - DIS - 00 000110	См. RLCA	? ? P ? 0 0
RLC (IY+DIS)	23	11111101 11001011 - DIS - 00 000110		? ? P ? 0 0
RL T		010	См. RLA	? ? P ? 0 0
RRC T		001	См. RRCA	? ? P ? 0 0
RR T		011	См. RRA	? ? P ? 0 0
SLA T		100	CY=T7,T7=T6... ..T0=0	? ? P ? 0 0
SRA T		101	T7=T7,T6=T7... ..CY=T0	? ? P ? 0 0
SRL T		111	T7=0,T6=T7... ..CY=T0	? ? P ? 0 0

RLD	18	11101101 01101111	A:=H,H:=L,L:=A	. ? P ? 0 0
RRD	18	11101101 01100111	A:=L,L:=H,H:=A	. ? P ? 0 0
Битовые операции				
BIT [B],[R]	8	11001011 01 [B] [R]	Z:= значение бита B [R]	. ? X X 0 1
BIT [B],[HL]	12	11001011 01 [B] [R]	Z:= значение бита B (HL)	. ? X X 0 1
BIT [B],(IX+IY)	20	11011101 11001011 - DIS - 01 [B]110	Z:= значение бита B (IX+DIS)	. ? X X 0 1
BIT [B],(IY+DIS)	20	11111101 11001011 - DIS - 01 [B]110	Z:= значение бита B (IY+DIS)	. ? X X 0 1
SET [B],T		11 [B]...	T [B]:=1	. . . . .
RES [B],T		10 [B]...	T [B]:=0	. . . . .
Команды передачи управления				
JP NN	10	11000011 - N - - N -	PC:=NN	. . . . .
JP CCC,NN	11	11 CCC010 - N - - N -	PC:=NN если условие CCC истинно	. . . . .
JR DIS	12	00011000 - DIS-2 -	PC:=PC+DIS	. . . . .
JR C,DIS	12	00111000	PC:=PC+DIS C=1	. . . . .
	/7	- DIS-2 -	PC:=PC+2 C=0	
JR NC,DIS	12	00110000	PC:=PC+DIS C=0	. . . . .
	/7	- DIS-2 -	PC:=PC+2 C=1	
JR Z,DIS	12	00101000	PC:=PC+DIS Z=1	. . . . .
	/7	- DIS-2 -	PC:=PC+2 Z=0	
JR NZ,DIS	12	00100000	PC:=PC+DIS Z=0	. . . . .
	/7	- DIS-2 -	PC:=PC+2 Z=1	
JP (HL)	4	11101001	PC:=HL	. . . . .
JP (IX)	8	11011101 11101001	PC:=IX	. . . . .
JP (IY)	8	11111101 11101001	PC:=IY	. . . . .
DJNZ DIS	13 /8	00010000 - DIS-2 -	B:=B-1 B:=0:PC:=PC+2 B<>0:PC:=PC+DIS	. . . . .

Команды организации подпрограмм				
CALL NN	17	11001101 - N - - N -	-(SP):=PC PC:=NC	. . . . .
CALL CCC,NN	17 /10	11 CCC100 - N - - N -	CALL NN ЕСЛИ CCC ИСТИННО	. . . . .
RET	10	110010011	PC:=(SP)+	. . . . .
RET CCC	11 /5	11 CCCOOO	RET ЕСЛИ CCC ИСТИННО	. . . . .
RETI	14	11101101 01001101	ВОЗВРАТ ИЗ INT	. . . . .
RETN	14	11101101 01000101	ВОЗВРАТ ИЗ NMI	. . . . .
RSTP	11	11 P 111	CALL P	. . . . .
Команды ввода/вывода				
IN A,(N)	11	11011011 - N -	A:=(N), ADR:=A/N	. . . . .
IN [R],(C)	12	11101101 01 [R]000	[R]:=(C), ADR:=B/C	. ? P ? O ?
INI	16	11101101 10100010	(HL):=(C), ADR:=B/C HL:=HL+1 B:=B-1	. ? X X 1 X B-1=0 Z=1 B-1=1 Z=0

INIR	21
	/16
		11101101
		10110010
			(HL):=(C),
			ADR:=B/C
			HL:=HL+1
			BC:=BC-1
			IF BC<>O THEN
			BEGIN
				. 1 X X 1 X

IND	16	11101101 10101010	(HL):=(C), ADR:=B/C HL:=HL-1 B:=B-1	. ? X X 1 X B-1=0 Z=1 B-1=1 Z=0

INDR	21
	/16
		11101101
		10111010
			(HL):=(C),
			ADR:=B/C
			HL:=HL-1
			BC:=BC-1
			IF BC<>O THEN
			BEGIN
				. 1 X X 1 X

OUT (N),A	11	11010011 - N -	(N):=A, ADR:=A/N	. . . . .
OUT (C),[R]	12	11101101 01 [R]001	(C):=[R], ADR:=B/C	. ? P ? O ?
OUTI	16	11101101 10100011	(C):=(HL), ADR:=B/C HL:=HL+1 B:=B-1	. ? X X 1 X B-1=0 Z=1 B-1=1 Z=0

OUTIR	21
	/16
		11101101
		10110011
			(C):=(HL),
			ADR:=B/C
			HL:=HL+1
			BC:=BC-1
			IF BC<>0 THEN
			BEGIN
				. 1 X X 1 X

OUTD	16	11101101 10101011	(C):=(HL), ADR:=B/C HL:=HL-1 B:=B-1	. ? X X 1 X B-1=0 Z=1 B-1=1 Z=0

OUTDR	21
	/16
		11101101
		10111011
			(HL):=(C),
			ADR:=B/C
			HL:=HL-1
			BC:=BC-1
			IF BC<>0 THEN
			BEGIN
				. 1 X X 1 X

Команды управления процессором				
NOP	4	00000000	Нет операции	. . . . .
HALT	4	01110110	Останов	. . . . .
DI	4	11110011	Запрет прерываний	. . . . .
EI	4	11111011	Разрешение прерываний	. . . . .
IM 0	8	11101101 01000110	Режим прерывания 0	. . . . .
IM 1	8	11101101 01010110	Режим прерывания 1	. . . . .
IM 2	8	11101001 01011110	Режим прерывания 2	. . . . .

Приложение Е

ПРОШИВКИ ПЛМ

Прошивка 556РТ1 (1) "Графика" (13 000 кГц)

	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00	7 L	6 L	5 L	4 L	3 L	2 L	1 H	0 L
00	-	-	-	-	-	-	-	-	-	-	L	L	L	L	L	L	.	.	.	.	.	.	▲	.
01	-	-	-	-	-	-	-	-	-	-	H	-	-	-	-	H	.	.	.	.	.	.	▲	.
02	-	-	-	-	-	-	-	-	-	-	H	-	-	-	H	-	.	.	.	.	.	.	▲	.
03	-	-	-	-	-	-	-	-	-	-	H	-	-	H	-	-	.	.	.	.	.	.	▲	.
04	-	-	-	-	-	-	-	-	-	-	H	-	H	-	-	-	.	.	.	.	.	.	▲	.
05	-	-	-	-	-	-	-	-	-	-	H	H	-	-	-	-	.	.	.	.	.	.	▲	.
06	-	-	-	-	-	-	-	-	-	-	H	L	H	-	-	-	.	.	.	.	▲	.	.	.
07	-	-	-	-	-	-	-	-	-	-	H	L	L	H	H	-	.	.	.	.	▲	.	.	.
08	-	-	-	-	-	-	-	-	-	-	H	H	L	H	L	L	.	.	.	▲	.	.	.	.
09	-	L	H	H	-	-	-	-	-	-	-	-	-	-	-	-	.	.	.	.	.	.	▲	.
10	-	H	L	L	-	-	-	-	-	-	-	-	-	-	-	-	.	.	.	.	.	.	▲	.
11	-	-	-	-	-	-	-	-	-	-	H	L	H	H	L	H	▲	.	.	.	.	.	.	.
12	-	-	-	-	-	-	-	-	-	-	H	L	H	H	H	-	▲	.	.	.	.	.	.	.
13	-	H	L	L	H	H	H	L	L	H	-	-	-	-	-	-	.	▲	▲	.	.	.	.	.
14	L	-	-	-	-	-	-	-	-	-	L	L	L	-	-	-	.	▲	.	.	.	.	.	.
15	-	H	-	-	-	-	-	-	-	-	H	L	L	H	H	H	.	.	.	.	.	.	.	▲
16	-	L	H	H	L	-	-	-	-	-	H	L	L	H	H	H	.	.	.	.	.	.	.	▲
17	-	L	H	H	H	L	L	L	L	-	H	L	L	H	H	H	.	.	.	.	.	.	.	▲
18	-	L	H	H	H	L	L	L	H	L	H	L	L	H	H	H	.	.	.	.	.	.	.	▲
19	-	L	H	H	H	-	H	-	-	-	H	L	L	H	H	H	.	.	.	.	.	.	.	▲
20	-	L	H	H	H	H	H	-	-	-	H	L	L	H	H	H	.	.	.	.	.	.	.	▲
21	-	L	H	H	H	L	H	H	H	-	H	L	L	H	H	H	.	.	.	.	.	.	.	▲
22	-	H	-	-	-	-	-	-	-	-	H	L	H	L	-	-	.	.	.	.	.	.	.	▲
23	-	L	H	H	H	-	-	-	-	-	H	L	H	L	-	-	.	.	.	.	.	.	.	▲
24	-	L	H	H	H	L	L	L	L	-	H	L	H	L	-	-	.	.	.	.	.	.	.	▲
25	-	L	H	H	H	L	L	L	H	L	H	L	H	L	-	-	.	.	.	.	.	.	.	▲
26	-	L	H	H	H	-	H	-	-	-	H	L	H	L	-	-	.	.	.	.	.	.	.	▲
27	-	L	H	H	H	H	L	-	-	-	H	L	H	L	-	-	.	.	.	.	.	.	.	▲
28	-	L	H	H	H	L	L	H	H	-	H	L	H	L	-	-	.	.	.	.	.	.	.	▲
29	-	L	H	H	H	L	L	L	H	H	L	-	-	-	-	-	.	.	.	.	.	.	.	▲
30	-	L	H	H	H	L	L	L	H	H	H	-	L	L	-	-	.	.	.	.	.	.	.	▲
31	-	L	H	H	H	L	L	L	H	H	H	L	L	H	L	-	.	.	.	.	.	.	.	▲
32	-	L	H	H	H	L	L	L	H	H	H	L	L	H	H	L	.	.	.	.	.	.	.	▲
33	-	L	H	H	H	L	L	L	H	H	H	L	H	H	-	-	.	.	.	.	.	.	.	▲
34	-	L	H	H	H	L	L	H	L	-	L	-	-	-	-	-	.	.	.	.	.	.	.	▲
35	-	L	H	H	H	L	L	H	L	-	H	-	L	L	-	-	.	.	.	.	.	.	.	▲
36	-	L	H	H	H	L	L	H	L	-	H	L	L	H	L	-	.	.	.	.	.	.	.	▲
37	-	L	H	H	H	L	L	H	L	-	H	L	L	H	H	L	.	.	.	.	.	.	.	▲
38	-	L	H	H	H	L	L	H	L	-	H	L	H	H	-	-	.	.	.	.	.	.	.	▲
39	-	L	L	-	-	-	-	-	-	-	H	L	L	H	H	H	.	.	.	.	.	.	.	▲
40	-	L	H	L	-	-	-	-	-	-	H	L	L	H	H	H	.	.	.	.	.	.	.	▲
41	-	L	L	-	-	-	-	-	-	-	H	L	H	L	-	-	.	.	.	.	.	.	.	▲
42	-	L	H	L	-	-	-	-	-	-	H	L	H	L	-	-	.	.	.	.	.	.	.	▲
43	-	L	H	H	L	H	H	H	-	-	-	-	-	-	-	-	.	.	.	.	.	▲	.	.
44	-	L	H	H	H	L	-	-	-	-	-	-	-	-	-	-	.	.	.	.	.	▲	.	.
45	-	L	H	H	H	H	L	L	-	-	-	-	-	-	-	-	.	.	.	.	.	▲	.	.
46	-	L	H	H	H	H	L	H	L	L	-	-	-	-	-	-	.	.	.	.	.	▲	.	.
47	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	▲	▲	▲	▲	▲	▲	▲	▲

Прошивка 556РТ1 (2) "Процессорная"

	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00	7 L	6 L	5 L	4 L	3 L	2 L	1 H	0 L
00	H	H	H	L	H	L	-	-	-	-	-	-	-	-	L	L	.	.	.	.	.	.	.	A
01	H	L	H	L	H	L	-	-	-	-	-	-	-	-	L	L	.	.	.	.	.	.	A	.
02	L	-	H	L	H	L	-	-	-	-	-	-	-	-	L	L	.	.	.	.	.	A	.	.
03	-	-	H	L	H	L	-	-	-	-	-	-	-	-	H	L	.	.	.	.	.	A	.	.
04	-	-	H	L	H	L	-	-	-	-	-	-	-	-	-	H	.	.	.	.	.	A	.	.
05	-	-	-	-	L	H	L	H	H	H	H	H	H	H	-	-	.	.	.	.	A	.	.	.
06	-	-	-	-	L	H	-	-	L	H	H	L	L	L	-	-	.	.	.	A	.	.	.	.
07	-	-	-	-	L	H	-	L	L	L	H	L	L	L	-	-	.	.	A	.	.	.	.	.
08	H	-	L	H	H	L	-	-	-	-	-	-	-	H	L		.	A	.	.	.	.	.	.
09	-	H	L	H	H	L	-	-	-	-	-	-	-	H	L		.	A	.	.	.	.	.	.
10	-	L	L	H	H	L	-	-	-	-	-	-	-	L	L		.	A	.	.	.	.	.	.
11	-	-	L	H	H	L	-	-	-	-	-	-	-	-	-		.	.	.	.	.	A	.	.
12	-	-	L	H	L	H	H	H	H	H	L	-	L	L	L	L	A	.	.	.	.	.	.	.
13	-	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	A	A	A	A	A	A	A	A
14	-	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	A	A	A	A	A	A	A	A
15	-	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	A	A	A	A	A	A	A	A
16	-	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	A	A	A	A	A	A	A	A
17	-	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	A	A	A	A	A	A	A	A
18	-	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	A	A	A	A	A	A	A	A
19	-	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	A	A	A	A	A	A	A	A
20	-	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	A	A	A	A	A	A	A	A
21	-	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	A	A	A	A	A	A	A	A
22	-	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	A	A	A	A	A	A	A	A
23	-	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	A	A	A	A	A	A	A	A
24	-	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	A	A	A	A	A	A	A	A
25	-	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	A	A	A	A	A	A	A	A
26	-	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	A	A	A	A	A	A	A	A
27	-	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	A	A	A	A	A	A	A	A
28	-	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	A	A	A	A	A	A	A	A
29	-	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	A	A	A	A	A	A	A	A
30	-	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	A	A	A	A	A	A	A	A
31	-	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	A	A	A	A	A	A	A	A
32	-	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	A	A	A	A	A	A	A	A
33	-	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	A	A	A	A	A	A	A	A
34	-	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	A	A	A	A	A	A	A	A
35	-	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	A	A	A	A	A	A	A	A
36	-	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	A	A	A	A	A	A	A	A
37	-	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	A	A	A	A	A	A	A	A
38	-	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	A	A	A	A	A	A	A	A
39	-	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	A	A	A	A	A	A	A	A
40	-	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	A	A	A	A	A	A	A	A
41	-	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	A	A	A	A	A	A	A	A
42	-	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	A	A	A	A	A	A	A	A
43	-	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	A	A	A	A	A	A	A	A
44	-	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	A	A	A	A	A	A	A	A
45	-	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	A	A	A	A	A	A	A	A
46	-	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	A	A	A	A	A	A	A	A
47	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	A	A	A	A	A	A	A	A

Таблица прошивки микросхемы ПЗУ DD31 (К155РЕ3)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	C7	C6	CE	CC	CC	DE	DC	9C	B5	F7	D3	D2	CA	C8	48	4F
0010	C7	C6	CE	CC	CC	DE	DC	9C	BD	FF	DF	DF	DF	DF	DF	DF

Таблица кодировки клавиш клавиатуры (разъем X4)

N		6	9	10	12	11	8	7	5
		АК0	АК1	АК2	АК3	АК4	АК5	АК6	АК7
13	DK0	CAPS SHIFT	A	Q	1	0	P	ENTER	SPACE BREAK
15	DK1	Z	S	W	2	9	O	L	SYMBOL SHIFT
4	DK2	X	D	E	3	8	I	K	M
16	DK3	C	F	R	4	7	U	J	N
3	DK4	V	G	T	5	6	Y	H	B

1	Инверсия цвета
17	Сброс

Приложение F

ПРИМЕРЫ ПРОГРАММ

Это приложение содержит некоторые примеры программ, демонстрирующие возможности ZX SPECTRUM.

Первая из этих программ требует ввести дату и дает день недели, который соответствует этой дате.

```
 10 REM CONVERT DATE TO DAY
 20 DIM D$(7,6): REM DAYS OF WEEK
 30 FOR N=1 TO 7: READ D$(N): NEXT N
 40 DIM M(12): REM LENGTHS OF MONTHS
 50 FOR N=1 TO 12: READ M(N): NEXT N
100 REM INPUT DATE
110 INPUT "DAYS?"; DAY
120 INPUT "MONTH?"; MONTH
130 INPUT "YEAR (20TH CENTURY ONLY)?"; YEAR
140 IF YEAR<1901 THEN PRINT "20TH CENTURY STARTS AT
    1901": GO TO 100
150 IF YEAR>2000 THEN PRINT "20TH CENTURY ENDS AT
    2000": GO TO 100
160 IF MONTH<1 THEN GO TO 210
170 IF MONTH>12 THEN GO TO 210
```

```
180 IF YEAR/4-INT(YEAR/4)=0 THEN LET M(2)=29: REM LEAP YEAR
190 IF DAY>M(MONTH) THEN PRINT "THIS MONTH HAS ONLY";
    M(MONTH); "DAYS": GO TO 500
200 IF DAY>0 THEN GO TO 300
210 PRINT "STOFF AND NONSENSE. GIVE ME A REAL DATE."
220 GO TO 500
300 REM CONVERT DATE TO NUMBER OF DAY SINCE START OF CENTURY
310 LET Y=YEAR-1901
320 LET B=365*Y+INT(Y/4):REM NUMBER OF DAYS TO START OF YEAR
330 FOR N=1 TO MONTH-1: REM ADD ON PREVIOUS MONTH
340 LET B=B+M(N): NEXT N
350 LET B=B+DAY
400 REM CONVERT TO DAY OF WEEK
410 LET B=B-7*INT(B/7)+1
420 PRINT DAY;"/";MONTH;"/";YEAR
430 FOR N=6 TO 3 STEP-1: REM REMOVE TRAILING SPACES
440 IF D$(B,N)<>" " THEN GO TO 460
450 NEXT N
460 LET E$=D$(B)TO N)
470 PRINT "IS A";E$;"DAY"
500 LET M(2)=28: REM RESTORE FEBRUARY
510 INPUT "AGAIN?",A$
520 IF A$="N" THEN GO TO 540
530 IF A$<>"N" THEN GO TO 100
1000 REM DAYS OF WEEK
1010 DATA "MON","TUES","WENDES"
1020 DATA "THURS","FRI","SATUR","SUN"
```

Эта программа устанавливает соответствие между ярдом, футом и дюймом:

```
 10 INPUT "YARDS?",YD,"FEET",FT,"INCHES",IN
 40 GO SUB 2000: REM PRINT THE VALUES
 50 PRINT "="
 70 GO SUB 1000: REM THE ADJUSTMENT
 80 GO SUB 2000: REM PRINT THE ADJUSTED VALUES
 90 PRINT
100 GO TO 10
1000 REM SUBROUTINE TO ADJUST YD,FT,IN TO THE NORMAL FORM FOR
     YARDS, FEET AND INCHES
1010 LET IN=36*YD+12*FT+IN: REM NOW EVERYTHING IS IN INCHES
1030 LET S=SGN IN: LET IN=ABS IN
1060 LET FT=INT(IN/12): LET IN=(IN-12*FT)*S: REM NOW IN IS OK
1080 LET YD=INT(FT/3)*S: LET FT=FT*S-3 YD: RETURN
2000 REM SUBROUTINE TO PRINT YD,FT, AND IN
2010 PRINT YD;"YD";FT;"FT";"IN";:RETURN
```

Эта программа моделирует выбрасывание монеты для игры в "китайку":

```
 5 RANDOMIZE
10 FOR M=1 TO 6: REM FOR 6 THROWS
20 LET C=0: REM INITIALIZE COIN TOTAL TO 0
30 FOR N=1 TO 3: REM FOR 3 COINS
40 LET C=C+2+INT(2*RND)
50 NEXT N
60 PRINT " "
```

```
 70 FOR N=1 TO 2: REM 1ST FOR THE THROWN HEXAGRAM, 2ND FOR
    CHANGES
 80 PRINT "---";
 90 IF C=7 THEN PRINT "-";
100 IF C=8 THEN PRINT " "
110 IF C=6 THEN PRINT "X": LET C=7
120 IF C=9 THEN PRINT "O";: LET C=8
130 PRINT "---"
140 NEXT N
150 PRINT
160 INPUT A$
170 NEXT M: NEW
```

Введите программу в компьютер и запустите ее на выполнение, затем нажмите клавишу ENTER 5 раз для получения двух гексаграмм. Просмотрите "Китайскую книгу изменений". Текст будет описывать ситуацию и последовательность соответствующих этому действий, а вы должны оценить глубину параллелей между ней и вашей собственной жизнью. Нажмите клавишу ENTER шестой раз и программа будет обнуляться - это избавит вас от легкомысленного использования результатов.

Многие пользователи найдут тексты всегда более вероятными, нежели они сами могли предполагать.

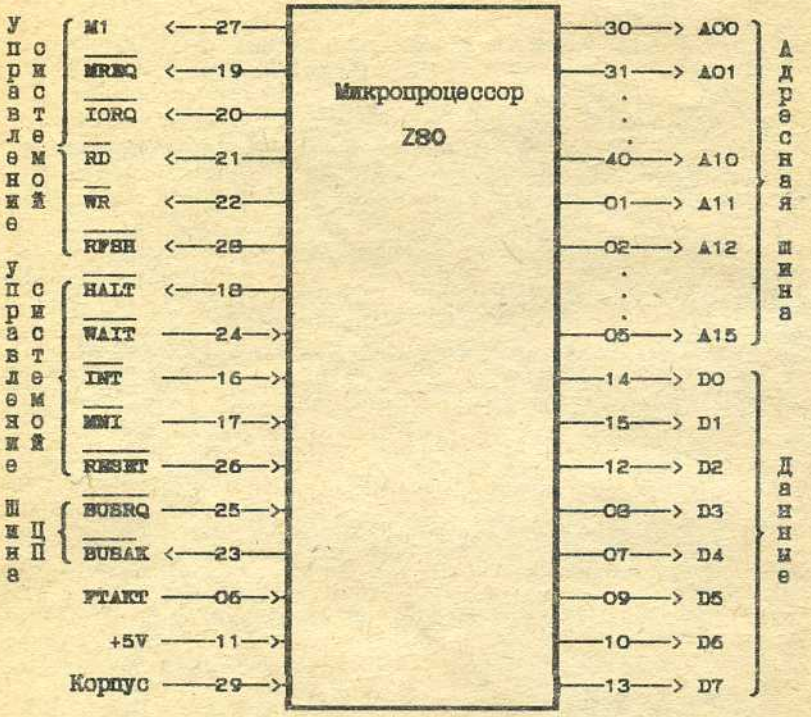
Следующая программа - игра "ящеры". Вы задумываете название некоторого животного, а компьютер пытается его отгадать, задавая вам вопросы, на которые вы должны отвечать "Да" или "Нет". Если компьютер не был ранее знаком с таким животным, то он попросит задать вам наводящие вопросы, которые помогут ему найти правильный ответ, или он попросит вас предложить ему задать название нового животного.

```
  5 REM PANGOLINS
 10 LET NQ=100: REM NUMBER OF QUESTIONS AND ANIMALS
 15 DIM Q$(NQ,50): DIM A(NQ,2): DIM R$(1)
 20 LET QF=8
 30 FOR N=1 TO QF/2-1
 40 READ Q(N): READ A(N,1): READ A(N,2)
 50 NEXT N
 60 FOR N=N TO QF-1
 70 READ Q$(N): READ A(N,1): READ A(N,2)
100 REM START PLAYING
110 PRINT "THINK OF ANIMAL.","PRESS ANY KEY TO CONTINUE."
120 PAUSE 0
130 LET C=1: REM START WITH 1ST QUESTION
140 IF A(C,1) THEN GO TO 300
150 P$=Q$(C): GO SUB 910
160 PRINT "?": GO SUB 1000
170 LET IN=1: IF R$="Y" THEN GO TO 210
180 IF R$="Y" THEN GO TO 210
190 LET IN=2: IF R$="N" THEN GO TO 210
200 IF R$<>"N" THEN GO TO 150
210 LET C=A(C,IN): GO TO 140
300 REM ANIMAL
310 PRINT "ARE YOU THINKING OF"
320 LET P$=Q$(C): GO SUB 900: PRINT "?"
330 GO SUB 1000
```

```
340 IF R$="Y" THEN GO TO 400
350 IF R$="Y" THEN GO TO 400
360 IF R$="N" THEN GO TO 500
370 IF R$="N" THEN GO TO 500
380 PRINT "ANSWER ME PROPERLY WHEN I'M","TALKING
    TO YOU":GO TO 300
400 REM GUESSED IT
410 PRINT "I THOGHT AS MUSH.": GO TO 800
500 REM NEW ANIMAL
510 IF QP>NQ-1 THEN PRINT "I'M SURE YOUR ANIMAL IS VERY",
    "INTERESTING, BUT I DON'T HAVE","ROOM FOR IT
    JUST NOW.": GO TO 800
520 LET Q$(QP)=Q$(C): REM MOVE OLD ANIMAL
530 PRINT "WHAT IS IT, THEN?": INPUT Q$(QP+1)
540 PRINT "TELL ME A QUESTION WHICH DIST","INQUISHES BETWEEN"
550 LET P$=Q$(QP): GO SUB 900: PRINT "AND"
560 LET P$=Q$(QP+1): GO SUB 900: PRINT " "
570 INPUT S$: LET B=LEN S$
580 IF S$(B)="?" THEN LET B=B-1
590 LET Q$(C)=S$(TO B): REM INSERT QUESTION
600 PRINT "WHAT IS THE ANSWER FOR"
610 LET P$=Q$(QP+1): GO SUB 900: PRINT "?"
620 GO SUB 1000
630 LET IN=1: LET IO=2: REM ANSWER FOR NEW AND OLD ANIMALS
640 IF R$="Y" THEN GO TO 700
650 IF R$="Y" THEN GO TO 700
660 LET IN=2: LET IO=1
670 IF R$="N" THEN GO TO 700
680 IF R$="N" THEN GO TO 700
690 PRINT "THAT'S NO GOOD.": GO TO 600
700 REM UPDATE ANSWERS
710 LET A(S,IN)=QP+1: LET A(S,IO)=QP
720 LET QP=QP+2: REM NEXT FREE ANIMAL SPACE
730 PRINT "THAT FOOLED ME."
800 REM AGAIN?
810 PRINT "DO YOU WANT ANOTHER GO?": GO SUB 1000
820 IF R$="Y" THEN GO TO 100
830 IF R$="Y" THEN GO TO 100
840 STOP
900 REM PRINT WITHOUT TRAILING SPACES
905 PRINT " "
910 FOR N=50 TO 1 STEP -1
920 IF P$(N)<>" " THEN GO TO 940
930 NEXT N
940 PRINT P$(TO N): RETURN
1000 REM GET REPLY
1010 INPUT R$: IF R$=" " THEN RETURN
1020 LETR$=R$(1): RETURN
2000 REM INITIAL ANIMALS
2010 DATA "DOES IT LIVE IN THE SEA",4,2
2020 DATA "IS IT SCALY",3,5
2030 DATA "DOES IT EAI ANTS",6,7
2040 DATA "A WHALE","A BLANCMANGE","A PANGOLIN","AN ANT"
```

Приложение 6

НАЗНАЧЕНИЕ ВЫВОДОВ МИКРОПРОЦЕССОРА Z80, Z80A



A0–A15 – адресная шина. Выходы с тремя состояниями. Активный уровень – высокий. Адресует ОЗУ или УВВ (до 64 к для ОЗУ).

D0–D7 – шина данных. Входы-выходы с тремя состояниями. Активный уровень – высокий.

$\overline{M1}$ – машинный цикл. Выход. Активный уровень – низкий. Указывает, что в текущем цикле осуществляется выборка кода операции.

$\overline{MREQ}$ – запрос памяти. Выход с тремя устойчивыми состояниями. Активный уровень – низкий. Сигнал указывает, что на адресной шине установлен адрес для операции чтения или записи в память.

$\overline{IORQ}$ – запрос ввода/вывода. Выход с тремя устойчивыми состояниями. Активный уровень сигнала – низкий. Сигнал указывает, что младший байт шины адреса содержит адрес УВВ. Кроме того, этот сигнал генерируется после выдачи подтверждения прерывания, тем

самым указывая, что вектор прерывания может быть помещен на шину данных.

$\overline{RD}$ – чтение из памяти. Выход с тремя устойчивыми состояниями. Активный уровень сигнала – низкий. Сигнал указывает, что ЦП готов к чтению данных из памяти или из устройства ВВ. Адресованное УВВ или память используют этот сигнал для стробирования при подаче данных на шины данных ЦП.

$\overline{WR}$ – запись в память. Выход с тремя устойчивыми состояниями. Активный уровень сигнала – низкий. Сигнал указывает, что на шине данных содержатся данные, предназначенные для записи в память или вывода на устройство вывода.

$\overline{RFSH}$ – восстановление. Выход. Активный уровень – низкий. Сигнал указывает, что младшие семь разрядов шины адреса содержат адрес восстановления для ОЗУ и текущий сигнал $\overline{MREQ}$, который должен использоваться для восстановления динамической памяти.

$\overline{HALT}$ – останов. Выход. Активный уровень – низкий. Сигнал указывает, что ЦП выполнил команду HALT и ожидает появления либо немаскируемого прерывания, либо маскируемого прерывания, после которого он продолжит работу. Перед выполнением HALT ЦП заносит в ОЗУ информацию, которая нужна для восстановления.

$\overline{WAIT}$ – ожидание. Вход. Активный уровень – низкий. Сигнал указывает микропроцессору, что адресуемая память или устройство ввода/вывода не готовы к передаче данных. ЦП ждет, пока активен этот сигнал.

$\overline{INT}$ – запрос на маскируемое прерывание. Вход. Активный уровень низкий. Запрос будет воспринят ЦП в конце выполнения текущей команды, если триггер разрешения прерывания, неуправляемый внутренними программными средствами, установлен в определенное состояние.

$\overline{NMI}$ – запрос на немаскируемое прерывание. Вход. Активный уровень – низкий. Это прерывание имеет более высокий приоритет, чем INT. Распознается в конце текущей команды. Сигнал автоматически переводит ЦП к выполнению программы с адреса 0066(HEX).

$\overline{RESET}$ – сброс. Вход. Активный уровень – низкий. При поступлении сигнала выполняются следующие действия:

а) сброс триггера разрешения прерывания IFF;
б) очистка счетчика команд и регистров I, R;
в) шины адреса и данных в Z-состоянии;
г) для всех управляющих выходных сигналов устанавливается неактивный уровень.

$\overline{BUSRQ}$ – запрос шин. Вход. Активный уровень – низкий. Сигнал имеет более высокий приоритет, чем NMI и всегда распознается в конце текущего машинного цикла. Он используется для организации прямого доступа к памяти (ПДП) и переводит в состояние высокого сопротивления все шины и тристабильные выходы сигналов управления, после чего этими шинами могут управлять другие внешние устройства.

$\overline{BUSAK}$ – подтверждение перевода шин в Z-состояние. Выход. Активный уровень – низкий. Сигнал подается на запрашивающее внешнее устройство.

ОГЛАВЛЕНИЕ

1

□□□МИГ□□□