СЕРИЯ
"БИБЛИОТЕЧКА СИНКЛЕРИСТА"

ZX SPECTRUM

ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ BASIC

Стивен Викерс под редакцией Робина Бредбера

1992

ГЛАВА 1
Введение

Если Вы читаете эту книгу впервые или открыли её 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 в ZX SPECTRUM.

Что бы Вы не делали, старайтесь в своей деятельности использовать компьютер. Если у Вас возник вопрос 'что будет, если я скажу так и так?', тогда ответ для Вас очень прост: введите эти фразы в компьютер и Вы увидите!

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

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

Клавиатура

В ZX SRECTRUM клавиши содержат не только одиночные символы (буквы, цифры и т.д.), но также составные символы (ключевые слова, названия функций и т.п.) и всё то, что вводится с клавиатуры не посимвольно. Для того, чтобы реализовать все эти функции и команды, некоторые клавиши клавиатуры имеют 5 и более значений, получаемых либо путем выбора соответствующего регистра (т.е путем нажатия клавиш CAPS SHIFT или SYMBOL SHIFT одновременно с какой-либо необходимой клавишей), либо путем перевода компьютера в один из возможных режимов работы.

Состояние индицируется курсором - мерцающей буквой, которая показывает, где будет появляться на экране 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 не изменён режим, 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 следующей клавиши будет интерпретировано как основной символ на клавише. В большинстве случаев это буквы.

И в [K] и в [L] режимах одновременное нажатие клавиши SYMBOL SHIFT и какой-либо клавиши воспринимается как вспомогательный символ, изображённый на клавише, а в случае 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 появляются на экране как заглавные.

Нажатие клавиши CAPS LOCK приводит к смене курсора [L] на [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 клавиши. В этом режиме нажатие даёт один символ или знак, если режим сохраняется, и другой, 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 совместно с 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 определённых пользователем графических символов.

Если некоторая клавиша удерживается в нажатом состоянии более, чем 2 или 3 секунды, это вызовет повторение её 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 удалён 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 же она содержит синтаксические ошибки, то ошибочное место указывается мерцающим знаком вопроса (?).

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

При выполнении команды и программы, вывод осуществляется в верхнюю часть экрана и сохраняется до ввода строки программы либо нажатия клавиши ENTER при наличии пустой строки, либо нажатия клавиш перемещения вверх, вниз.

В нижнюю часть экрана выводятся также сообщения об ошибках, которые сохраняются там до нажатия любой из клавиш (это индицируется переходом в режим [K]).

В определённых состояниях клавиши CAPS SHIFT и SPACE действуют как BREAK, останавливая компьютер с выдачей сообщений "D" или "L", это распознается:

а) в конце выполняющего 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 и вывода результатов работы программы, когда верхняя часть экрана заполнится полностью, он сворачивается на одну строку, компьютер останавливается с выдачей сообщения "SCROLL?". Ответ N, SPACE или STOP вызовет остановку программы.

С выдачей сообщения "D BREAK-CONT REPEATS", нажатие любой другой из клавиш разрешает свёртку.

Нижняя часть экрана используется для ввода команд, строк программы и вводимых 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, команды RUN, LIST, GO TO, CONTINUE, INPUT NEW, REM, PRINT, STOP в INPUT-данных, BREAK

Наберите эти две строки программы вычисления суммы двух чисел

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

Так чтобы на экране появилось:

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

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

Теперь наберите:

```
15 LET B=15
```

и введите. Строка вставится между строками 10 и 20. Подобного невозможно было бы сделать, если бы нумерация начиналась с 1 и 2, а не с 10 и 20, как в нашем случае (номер может быть 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, то в нижней части экрана появится копия текущей строки, в нашем случае копия строки 20. Нажмите и удержите 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 и это вызовет замену старой строки 20 на новую, записанную в нижней части экрана, на экране это будет выглядеть так:

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

Запустив программу , нажав RUN и ENTER , получите на экране сумму.

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

Есть ещё 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 быстрее: нажать EDIT, что вызовет копирование текущей строки в нижнюю часть экрана, затем нажать ENTER, строка заменит такую же в программе, а 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. Ещё раз выполните 12 и ENTER. Курсор снова установится между строками 10 и 15. Теперь нажмите EDIT, и строка 15 будет скопирована 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 нажмёте EDIT, вниз будет переслана строка 20. И, 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 удобно при просмотре очень больших программ. Другое назначение номеров строк - это служить именем оператора при ссылке к нему из другого места программы (GO TO N).Команда LIST без операндов выдаёт листинг с 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 возникнет вопрос : что же делает строка 10? Компьютер игнорирует эту строку,это комментарий(REMARK или REMINDER). Всё, что следует после REM, компьютером игнорируется 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 перейдёт в ожидание следующего числа. Этот переход обеспечивается в строке 60 в операторе 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 останова 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 '0 OK') номер последней выполнявшейся строки и продолжает выполнение именно с этой строки.

Просмотрите внимательно 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 это 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 наберем несколько программных строк:

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

Эта программа никак не связана с набранной нами ранее программой, но их обе можно держать в памяти компьютера 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 её значение в качестве INPUT-данных. Например, ответьте программе

```
N$ (удалив кавычки).
```

Это сделает оператор INPUT в строке 110 подобным оператору

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

Если Вы решили ввести STOP под строковый ввод, то должны 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 в том, что Вы можете указать RUN без номера строки, и тогда выполнение начнётся 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
```

С выдачей сообщения "L BREAK INTO PROGRAM". Команда BREAK может быть использована и во время выполнения операции на магнитофоне или принтере, в этом случае выдаётся сообщение "D BREAK-CONT REPEATS". Команда CONTINUE в этом случае (как и в большинстве других), вызовет повторение оператора, в котором произошла остановка. Ввод команды CONTINUE после сообщения "L BREAK INTO PROGRAM" продолжит выполнение со следующего оператора.

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

```
N$ (удалив кавычки).
```

Поскольку значение N$ не определено, то будет выдано сообщение "2 VARIABLE NOT FOUND". Если теперь Вы выполните:

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

на что компьютер ответит "0 OK, 0:1", а затем введёте CONTINUE, увидите, что программа завершится нормально.

Как уже отмечалось, сообщение "L BREAK INTO PROGRAM" особое, так как выдача после него CONTINUE не вызывает повторение команды, вызвавшей останов.

Все приведённые в этой главе утверждения PRINT, LET, INPUT, RUN, LIST, GO, CONTINUE, NEW, REM могут быть использованы либо как операторы в программе, либо как команды, хотя

ГЛАВА 3

Условия

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

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

IF - некоторое истинное или ложное выражение, THEN - некоторое действие.

Например, выполните команду NEW, затем наберите и выполните программу (это игра для двух человек):

```
10 REM GUESS THE NUMBER (угадайте числа)
20 INPUT A:CLS
30 INPUT "GUESS THE NUMBER", B (угадайте число)
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 GOTO 30
```

Здесь IF оператор имеет форму: IF условие THEN ... где '...' - последовательность операторов, разделённых двоеточием обычным образом. Если 'условие' истинно, то выполняется 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 либо равны, либо одно больше другого, а строки либо равны, либо одна следует после другой в алфавитном порядке. Для задания условия используются отношения:

=, < , > , <= , >= , <>

Например, выражение 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, завершает работу, выполняя 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 условия 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 набирать < и =)
>= - SYMBOL SHIFT вместе с E - больше или равно

ГЛАВА 4

Циклы

Краткое содержание: FOR, NEXT, TO, STEP

Допустим нам необходимо составить программу, подсчитывающую сумму вводимых пяти 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=Q
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 TIMES THAT A HAS BEEN INPUT SO FAR
50 LET TOTAL=TOTAL+A
60 NEXT C
70 PRINT TOTAL
```

Здесь 'C' управляющая переменная цикла должна иметь имя в одну букву. 'C' последовательно принимает значения 1, 2, 3, 4 и 5 (предел - конечное значение управляющей переменной цикла) и при каждом проходе выполняются строки 30, 40, 50. Затем после того, как 'C' примет пятое значение, выполнится 70-я строка. Приращение значения управляющей переменной составляет 1, но это значение можно изменить, используя указание STEP как часть оператора FOR.

Таким образом общая форма оператора имеет вид :

```
FOR 'упр.перем.'='нач.знач.' TO 'предел' STEP 'шаг приращ.'
```

Здесь 'начальное значение', 'предел', 'шаг приращения' есть выражения, принимающие числовое значение. Итак, если Вы замените строку 20 программы на

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

то 'C' последовательно примет значения 1, 2.5 и 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 0, 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 подпрограмме.

Этот оператор подобен GO TO N с той разницей, что при использовании GO SUB компьютер запоминает следующий после GO SUB оператор, которому и передаётся управление после выполнения программы. 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 с оператора, следующего после оператора с этим номером.

Приведём 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, DATA и RESTORE.

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

Это простой способ получения выражений 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, используя оператор RESTORE. 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 с первого оператора DATA. Вообще-то, Вы можете не указывать номер строки в операторе RESTORE и, 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 значения переменным X, Y, Z, начиная с 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 арифметических 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
```

Отсюда видно, что вычисления могут быть комбинированными. Комбинации такого типа, как **SUM*15/100** называются выражениями. Выражение - это самый короткий путь для указания компьютеру на то, что вычисления надо делать одно за другим.

В нашем примере выражение **SUM*15/100** указывает: возьми значение переменной с именем "SUM", умножь его на 15 и затем раздели на 100. Если Вы не можете ещё этого сделать, мы рекомендуем просмотреть вводную часть этой книги, чтобы ознакомиться 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. Существует правило, по которому умножение и деление выполняются последовательно слева направо. Когда все они выполнятся, то затем будут выполняться сложение и вычитание по порядку также слева направо.

Для задания приоритета в компьютере ZX SPECTRUM используются числа в интервале 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 сложение ('+') для сцепления строк (конкатеации) 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 не 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 I SHALL NEVER BE ABLE TO TYPE IT
OUT AGAIN WITHOUT MAKING A MISTAKE
NOW WE ARE SIX
NOWWEARESIX
```

Два последних имени указывают на одну и ту же переменную

Примеры недопустимых имен переменных:

```
2001 (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 еще:

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

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 чисел. Выполните:

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

1E10 и 1E10 +1 не различаются компьютером как разные числа (1e10 усекается справа). Ещё один, более наглядный пример:

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

Погрешность в 5E9 составляет около 1, а с прибавлением единицы 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 её 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 интерпретации этой строки компьютер найдет двойную кавычку, открывающую "FINNEGANS WAKE", то сочтёт её закрывающей кавычкой для строки "HAVE YOU FINISHED " и затем сможет вывести "FINNEGANS WAKE". Здесь надо помнить специальное правило: если Вы хотите вывести кавычки внутри строки, они должны удваиваться. Например:

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

В данном случае будет выведена только 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 её 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 и которое может применяться к строковым выражениям. Общая его форма:

'строковое выражение '('начало' TO 'конец')

Следущее выражение истинно:

"ABCDEF" (2 TO 5) = "BCDE"

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

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

Последнее выражение можно было бы 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 THE 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 дополняются 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.

Если Вы теперь выполните

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

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 запитые в бейсик-систему подпрограммы, которые получая на входе одни значения, называемые аргументами, возвращают другие значения - результаты.

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"

а 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 (E), и нажать клавишу K.

Если в одном выражении используются и функции, и операции, то функции будут вычислены перед выполнением любых операций. Однако, Вы можете изменить этот порядок, применяя скобки.

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

LET A$ = STR$ 1E2

аналогично по действию команде

LET A$ = "100"

Или выполните

PRINT LEN STR$ 100.0000

и получите ответ 3, так как STR$ 100.0000 = "100".

Функция VAL обратится к функции STR$ и преобразует строку в число. Так,

VAL "3.5" = 3.5

или

VAL "2*3" = 6

или даже так

VAL ("2" + "*3") =6.

В последнем случае происходит вычисление двух выражений, сначала строкового с получением строки "2*3", затем числового с получением строки "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 - это VAL$. И аргументом и результом этой функции является строка символов. Она работает как VAL, примененная дважды, раскрывая все кавычки в строках:

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

Сделайте

LET A$ = "99"

и затем 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 так называемая математическая функция сигнум (знак). И аргумент и результат её числовые. Результат равен: **1, если аргумент положителен; 0, если аргумент равен 0; -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 =3

Сложности возникают при отрицательном аргументе, так как округление происходит к ближайшему целому не большему, чем аргумент:

INT -3.9 = -4

Функция SQR вычисляет корень квадратный от числа, например:

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

Если аргумент отрицательный, то выдается 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 строковый). Аргументы должны быть обязательно 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), теперь 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 с точностью 0.5 надо добавить к результату '.5'. Вы можете задать для себя такую функцию:

```
20 DEF FN R(X) = INT(X + 0.5): REM GIVES X ROUNDED TO THE
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()
```

Есть одна тонкость в этой программе. Во-первых, функция FN Q не использует аргументов, но скобки при этом должны обязательно использоваться. Во-вторых, операторы DEF не выполняемые, компьютер после выполнения строки 10 просто 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 в функции FN P.

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

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

Теперь изменим строку 20 на:

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

В этом случае FN P(2, 3) будет возвращать значение 10.

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

LEFT$ (A$, N) - возвращает подстроку, содержащую 'N' первых символов строки 'A$';
RIGHT$(A$, N) - возвращает подстроку, содержащую 'N' последних символов в строке 'A$';
TL$(A$) - возвращает подстроку, содержащую все символы строки 'A$', кроме первого.

Вы можете определить такие 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 их работу со строками длиной 0 и 1.

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 сочтёте их слишком сложными, можете пропустить эту главу. Всё сказанное относится к функциям: ** (возведение в степень), exp, LN, тригонометрическим функциям: SIN, COS, TAN, и обратных к ним: ASN, ACS, ATN.

**** и EXP**

Вы можете возвести число в некоторую степень путем многократного умножения его самого на себя необходимое число раз. Это обычно изображается записью числа обозначающего степень справа вверху от числа обозначающего основание, но такую форму записи трудно реализовать в компьютере, поэтому там используются специальный символ (направленная вверх стрелка), в данном описании заменённый двумя звездочками: '**'. Например, степень 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:'умножь 'A' само на себя 'B' раз'. Но это предполагает, что 'B' положительное целое 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 всё-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 станет выводить два числа, если Вы правильно, конечно, 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 в начале, а плюс 15% дохода, что составит 115 %. Для вычисления другим способом, Вы умножаете Вашу сумму денег на 1.15 и получаете тот же результат. В конце следующего года Вы снова получите 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 с капитала.

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, его значение передается для вычисления степени.

Имеется определённое значение A, которое делает функцию FN A иллюстрирующей специальную математическую 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 пять его 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 является логарифмическая функция. Логарифм (по основанию a) числа X есть степень, в которую надо возвести a, чтобы получить X. Это записывается так: LOGa X. (выражение a**LOGaX=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 его диаметр на число, называемое PI. Подобно 'e', PI представляется бесконечной десятичной дробью, его начало:

```
3.141592653589...
```

Слово 'PI' в ZX SPECTRUM обозначает это число. Выполните, например:

```
PRINT PI
```

SIN, COS, TAN и ASN, ACS, ATN

Тригонометрические функции описывают те случаи, когда точка перемещается вокруг 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 - абсцисса. Необходимо помнить, что если точка находится слева от оси Y, то косинус отрицательный, а если точка находится под осью X, то отрицательный синус. Необходимо так же 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, в ZX SPECTRUM тригонометрические функции вычисляются в радианах. Для перевода из градусов в радианы необходимо число разделить на 180 и умножить его на PI, а для обратного преобразования необходимо 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, она возвращает случайное число в интервале от 0 до 1 (может принимать значение 0, но никогда 1). Попробуйте выполнить:

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

Вы увидите, как меняется результат.

Фактически RND не абсолютно случайное число, а выбирается из определённой последовательности длиной в 65536 чисел, поэтому обычно 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 случайных целых чисел используйте функцию INT (округляет с отбрасыванием дробной части). Например:

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

будет давать числа 1, 2, 3, 4, 5, 6.

Пусть имеется программа:

```
10 REM DICE TURNING PROGRAM (выбрасывание кости)
20 CLS
30 FOR N=1 TO 2
40 PRINT 1+INT (RND*6);' ';
50 NEXT N
60 INPUT A$:GO то 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 последовательсть будет начинаться с числа 0, 0022735596. В утверждении RANDOMIZE Вы можете использовать любые числа в интервале от 1 до 65535. RANDOMIZE без числа, а также RANDOMIZE 0 задают случайный параметр команды.

Например, имеется программа:

```
10 RANDOMIZE 1
20 PRINT RND:GO TO 10
```

В каждой операции будет печататься неслучайное число. Для улучшения случайности распределения можно заменять 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 подсчета числа выпадений 'орла' и 'решки' (перевод имен программы: HEADS - орлы, TAIL - 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:GO TO 20
```

Если программа выполняется достаточно долго, то отношение 'орлов' к 'решкам' приблизительно равно 1.

ГЛАВА 12

Массивы

Краткое содержание: DIM.

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

```
5 REM THIS PROGRAMM 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, это применение массивов. Переменные в массиве являются его элементами, обладают общим именем и различаются только номером записываемым после имени (индексом).

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

Элементы массивов называют индексируемыми переменными. Перед использованием массива необходимо зарезервировать под него память. Это делается в операторе DIM (от английского DIMENSION). В нашем случае это будет оператор 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, 3, 11, 1, 0, 6
```

Можно также объявлять массивы с более чем одной размерностью. Например, в двухмерном массиве первый индекс можно сравнить 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 двухмерный массив 'C' с размерностью 3 и 6:

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

что 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 быть также 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, перечеркнутой буквы S (знак денежной единицы - $).

Допустим, Вам необходимо объявить массив A$ на 5 строк по 10 символов в каждой, Вы должны записать:

```
DIM A$(5,10)
```

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(НЕТ).

Некоторое выражение 'И' выражение истины - истина, если истинны оба этих выражения, например:

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

'X' будет напечатано только тогда, когда

```
A$="YES" и X>0
```

Некоторое выражение 'ИЛИ' некоторое другое выражение - истина, если истинно хотя бы одно из этих выражений. 'НЕ' выражение истинно, если ложно само выражение и наоборот.

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

Условие '<>' обратно в логическом смысле условию '=', то есть:

```
A <> B тоже, что и NOT A=B
NOT A <> B тоже, что и A=B
```

Тем, кто боится сложностей, следующие разделы можно опустить.

1. Условия =, < , >, <=, >= , <> дают числовой результат 1 для истины и 0, если ложь. Например, оператор PRINT 1=2, 1<>2 выведет 0 для '1=2', которое ложно и 1 для '12', которое истинно.

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 другое значение (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 большее из двух чисел A или B.

Пример использования OR:

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

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 GO TO 10
```

ГЛАВА 14

Набор символов

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

Буквы, цифры, знаки пунктуации обозначаются символами и 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 кодовую 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
```

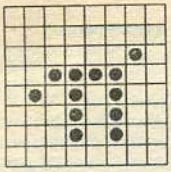
Все эти символы (кроме знака фунта и "c" в кружочке) образуют код ASCII (AMERICAN STANDART CODES FOR INFORMATION INTERCHANGE).

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 GRAPHICS(CAPS SHIFT 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, то они будут выдавать инверсные символы, т.е. чёрное становится белым, а белое чёрным.

Независимо от SHIFT, клавиша с цифрой "9" обеспечивает Вам возврат к обычному ([L] - курсор) режиму, а клавиша "0" функцию DELETE.

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

1.Каждый символ представляется точками в матрице 8 x 8, 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 1 закрашенной точки.

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

Эти восемь двоичных 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 -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 всё 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, но об этом подробнее будем говорить в главе 24.

Графические символы

Символ	Код	Набор	Символ	Код	Набор
	128	<C> 8		143	<C> SH8
	129	<C> 1		142	<C> SH1
	130	<C> 2		141	<C> SH2
	131	<C> 3		140	<C> SH3
	132	<C> 4		139	<C> SH4
	133	<C> 5		138	<C> SH5
	134	<C> 6		137	<C> SH6
	135	<C> 7		136	<C> SH7

Вернёмся к знакам. Мы ещё не поговорили о не напечатанных первых 32 знаках с кодами от 0 до 32. Это управляющие символы. они не отображаются, вместо них на телевизоре отображается знак '?'. Назначение 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 сделать

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

CHR$ 8 -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. Телевизором также используются символы с кодами 16 и 23, но об этом поговорим в главах 15, 16.

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

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 выбирается для сравнения следующая пара символов. Так до тех пор, пока встретятся несовпадающие символы, либо пока одна из строк не кончится, она 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 " > " - "находится позади". Так, что выражения

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

оба истинны.

Для иллюстрации всего сказанного приведём 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 " < ";:GO TO 60
50 PRINT "=";
60 PRINT " ";B$
70 GO TO 10
```

Следующая программа закрепляет определённые 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 королевой; QUEEN

5 LET B=BIN 01111100:LET S=BIN 00111000:
LET D=BIN 00010000
10 FOR N=1 TO 6:READ P$:REM 6 PIECES
20 FOR F=0 TO 7:REM READ PIECE INTO 8 BYTES
30 READ A:POKE USR P$+F, A
40 NEXT F
50 NEHT N
100 REM BISHOP
110 DATA "B", 0, D, BIN 00101999, 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 ROOK
170 DATA "R", 0, BIN 01010100, B, C
180 DATA C, B, B, 0
190 REM QUEEN
200 DATA "Q", 0, BIN 01010100, BIN 00101000, D
210 DATA BIN 01101100, B, B, 0
220 REM PAWN
230 DATA "P", 0, 0, D, C
240 DATA C, D, B, 0
250 REM KNIGHT
260 DATA "N", 0, D, C, BIN 01111000
270 DATA BIN 00011000, C, B, 0
```

ГЛАВА 15

Дополнительные сведения об операторах PRINT и INPUT

Краткое содержание: PRINT в формате: выражение (числового или строчного типа);TAB числовое выражение, AT числовое выражение PRINT разделители: " ," , " ; " , " " " ; INPUT формат: переменная числовая или строчная LINE строчная переменная, PRINT первый символ не буква.

Выражение, значение которого выводится 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;"*"
```

Печатается '*' в середине экрана.

```
AT 'строка','столбец'
```

Этот параметр перемещает позицию вывода в место, определяемое номером строки и столбца. Номер строки меняется от 0 до 21, а номер столбца от 0 до 31.

```
SCREEN$.
```

Его действие противоположно 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 указанный столбец на 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;TAB 12;"CONTENTS";AT 3, 1;"CHAPTER"; TAB 24;"PAGE"
```

выведет на экран две первые строки 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 получится при замене в 20 строке '8' на '6'.

Несколько замечаний:

1.В рассмотренных примерах в качестве ограничителя использовалась ';'. Можно использовать ',' (или пробел, как конец оператора). При этом необходимо следить, чтобы позиция вывода не переместилась в часть экрана, предназначенную для вывода сообщения.

2.Мы не можем использовать две нижние строки экрана (22 и 23), т.к. они используются для получения оператором INPUT данных. 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.

CSL

Ещё одним оператором, используемым совместно с PRINT является CLS, он производит очистку экрана подобно операторам CLEAR и RUN. При заполнении всего экрана происходит его свёртка. 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 текста. 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: 'SCROOL ?'. 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 части) вопрос, в ответ на который Вы должны ввести свой возраст.

Фактически INPUT содержит те же 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, INPUT может начинаться с буквы, которая является вводимой переменной.

Второе: Вы можете вводить значение переменной, как часть титра, заключив её 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' выдаёт компьютер, значение 'YOUR AGE' вводите Вы сами. По мере выдачи операторов INPUT происходит свёртка экрана. Рассмотрим пример использования AT в INPUT операторе:

10 INPUT"THIS IS LINE 1", A$;AT 0, 0;THIS LINE IS 0", A$; 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 через начерченную линию.

Выполним:

10 FOR N=0 TO 19:PRINT AT N, 0;N:NEXT N

Когда информация начнет смещаться в область действия операторов PRINT, произойдет свёртка экрана. Ещё одним параметром оператора INPUT является LINE, он предназначен для ввода строчных переменных. Рассмотрим пример:

INPUT LINE A$

Если ввести какую-либо строчную переменную, то её 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 параметрам 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 только CHR$1 и CHR$C (CHR$22 не учитывается).

Аналогом TAB является управляющий символ CHR$23, значения задаваемых им параметров находятся в пределах от 0 до 65535.

PRINT CHR$23+CHR$A+CHR$B

То же, что и

PRINT TAB A+256*B

Вы можете использовать POKE для остановки компьютера, запрашивающего свёртку, выполнив:

POKE 23692, 255

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

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

на экран будут выведены часы.

Стартуем следующую программу, проверяющую знание таблицы умножения:

```
10 LET M$=" "
20 LET A=INT(RND*12)+1:LET B=INT(RND*12)+1
30 INPUT(M$), "WHAT IS";(A);"*";(B);"?";C
100 IF C=A*B THEN LET M$="RIGHT":GOTO 20
111 LET M$="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 замените в 30 строке 'C' на 'C$', в 100 строке 'C' на 'VAL C$' и дополнительно введите строку:

```
40 IF C$<>STR$ VAL C$ THEN LET M$="TYPE IT PROPERLY AS NUMBER.":GOTO 30
```

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
```

Она продемострирует Вам возможности вывода компьютером ZX SPECTRUM на цветной телевизор восьми цветов (включая чёрный и белый) и двух уровней яркости. Если телевизор чёрно-белый, Вы увидите различные градации серого цвета.

Ниже дана кодировка цветов:

```
0-чёрный
1-синий
2-красный
3-фиолетовый
4-зеленый
5-голубой
6-жёлтый
7-белый
```

Для чёрно-белого телевизора этот ряд представляет собой последовательность перехода серых полутонов от чёрного до белого.

Для использования цветов уясним строение графического экрана. Он состоит из 768 позиций(24 на 32), каждая из которых представляет из себя матрицу 8 на 8 пикселей. Вспомним из 14 главы:

```
0S-белая точка
1S-чёрная точка
```

Позиция символа (знакоместо) также рассматривается с этих позиций: INK-цвет тона, PAPER-цвет фона, т.к. знакоместо состоит из INK и PAPER (рисунок на бумаге). Можно так же говорить о INK и PAPER. Всё это имеет следующую кодировку:

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

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 (так как 5 - код голубого цвета).

Формат операторов:

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

Отметим, что использование чисел больших, чем указывалось выше допустимо, но даёт другой эффект.

К примеру, '8' может использоваться во всех четырёх операторах как средство, позволяющее определить значение ранее установленных атрибутов. Так:

```
PAPER 8
```

не установит цвета фона (т.к. такого цвета нет), а поможет выяснить значение предыдущего PAPER. Операторы: INK 8, BRIGHT 8 FLASH 8 выдадут значения этих 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: чёрный, синий, красный, фиолетовый; к чёрному цвету подходят светлые тона: зелёный, голубой, жёлтый, белый.

Выполним:

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

Можно запустить программу, выдающую на экран дисплея цветные полосы:

```
INK 9:PAPER 8: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 с красным (код фиолетового цвета '3', он является суммой кодов синего'1' и красного '2').

Видеть все восемь цветов на одном участке экрана невозможно, т.к. это будет темное пятно. Но там, где цвета частично накладываются друг на друга мы увидим цветовую гамму. В качестве примера выполним программу (отметим, что INK получается с использованием SHIFT и 8 в G-режиме):

```
10 BORDER 0:PEPER 0:INK 7:CLS
20 FOR A= 1 TO 6
30 PRINT TAB 6 ; INK 1 ; "▯▯▯▯▯▯":REM: 18 INK SQUA
40 NEXT A
50 LET DATA LINE=200
60 GO SUB 1000
70 LET DATA LINE=200
80 GO SUB 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 и она 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 чёрным по белому, в инверсном белым по чёрному.

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

Возможен ещё один способ использования INK и PAPER, их можно вводить как параметры PRINT. Точно так же можно использовать и другие операторы, рассмотренные в этой главе, отметив при этом что их действие распространяется только до конца PRINT.

В примере:

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

только 'X' будет выведен на жёлтый 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 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 и с помощью управляющих символов, подобных управляющим символам для AT и TAB в главе 15.

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 расширением клавиатуры с цифрами 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 захотите уничтожить вводимое, нажмите DELETE два раза. После первого раза на экране высветится знак вопроса, или что-нибудь ещё. Не пугайтесь и нажмите DELETE ещё раз.

<= и >= тоже не работают обычным образом до тех пор, пока курсор не выйдет к предыдущему управляющему символу.

Режимы:

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

ATTR - это функция, имеет следующий формат:

ATTR (строка, столбец)

Значения двух параметров функции подобно значению параметров в AT. В результате выполнения будут выведены значения атрибутов для соответствующей позиции экрана. Выводимый результат - это число, представляющее сумму четырёх 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, жёлтый 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 '/' перечеркнёт 'B'. Этим способом можно выводить слова из комбинированных знаков на ZX SPECTRUM; два фона или два тона дают фон, один из них даёт тон. Это интересное свойство, если Вы повторите вывод одного символа дважды, то он не будет отображён. Так, если дать: `PRINT CHR$8;OVER 1;"/"` дополнительно к описанному выше утверждению, то мы в результате увидим 'B', не перечёркнутое '/'. 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 GO TO 10
```

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

4) Введём вручную, или с помощью оператора LOAD программу для ввода шахматных фигур из главы 14, и введём программу разбивки экрана под шахматную доску:

```
5 REM DRAW BANKBOARD
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 TO3
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$(8,8):REM POSITIONS OF PIECES
205 REM SET AP INITIAL POSITION
210 LET B$(1)="RNBQKBNR"
220 LET B$(2)="PPPPPPPP"
230 LET B$(7)="PPPPPPPP"
240 LET B$(8)="RNBQKBNR"
300 REM DISRLAY BOARD
310 FOR N=1 TO 8:FOR M=1 TO 8
320 LET BC=CODE B$(N, M): INK PW
325 IF BC=CODE " " THEN GO TO 350:REM SRACE
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

Эта глава описывает возможности компьютера ZX SPECTRUM по отображению графической информации. Экран компьютера содержит 22 строки 32 символа в каждой, что составляет 22x32=704 символьные позиции. Как Вы уже поняли из 16 главы, каждая символьная позиция

представляется квадратом 8x8 точек, называемых пикселями. Пиксель задается двумя числами, его координатами. Первое задаёт координату X, тоесть удаление (в пикселях) от левой границы экрана, второе задаёт координату Y, удаление от нижней границы экрана. Координаты записываются в скобках: (0, 0), (255, 0), правый, верхний правый и верхний левый углы экрана.

Оператор PLOT X, Y вызывает высвечивание 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, 88+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 (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.

Помощь при построении изображений Вам могут оказать операторы DRAW и CIRCLE.

Оператор DRAW чертит линию, заданную в форме

```
DRAW X, Y
```

Началом линии является пиксель, на котором завершился один из предыдущих операторов PLOT, DRAW или CIRCLE (этот пиксель называется текущей PLOT-позицией. Операторы RUN, CLEAR, CLS и NEW устанавливают её в левый нижний угол экрана). Таким образом, оператор DRAW задаёт длину и направление вычерчивания линии, но не начальную точку.

Поэкспериментируйте с такими командами:

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

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

```
10 BORDER0: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 OY LINE
50 DRAW INK C;X2-X1, Y2-Y1
60 LET X1=X2:LET Y1=Y2:REM NEXT LINE STARTS WHERE LAST ONE FINISHED
70 LET C=C+1:IF C=8 THEN LET C=1:REM NEW COLOUR
80 GO TO 40
```

Вы можете использовать в операторах PLOT и DRAW управляющие символы PAPER, INK, FLASH, BRINHT, INVERSE и OVER так же, 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' положительно, то дуга вычерчивается влево, а если отрицательно, 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 конечной точкой (150, 150). Вычерчивание начнется в направлении юго-восток, а 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 и 1, при этом:

PLOT INVERSE 1 -устанавливает для заданного пикселя цвет фона,

PLOT OVER 1 -изменяет цвет пикселя на противоположный, если был закрашивающий цвет, то становится цвет фона и наоборот.

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

Другой пример использования OVER с записью чёрным по белому:

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

вычерчивает линию по диагонали.

Теперь попробуйте:

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

и перечертите её командой

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

это не изменит картинку, так как при черчении как вперед, так и назад используются одни и те же пиксели. Имеется способ получения необычных 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 задаёт определяемый пользователем символ для шахматной доски, который закрепляется за клавишей 'A'. Для символа используется красный закрашивающий цвет и жёлтый цвет фона, но на экране этот символ будет казаться оранжевым.

Ещё один пример. Программа, которая строит график некоторой функции. На первый её запрос 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 0,1 75
30 INPUT S, E$
40 FOR F=0 TO 255
50 LET X=(F-128)*S/128:LET Y=VAL E$
60 IF ABS Y > 87 THEN LET T=0:GO TO 100
70 IF NOT T THEN PLOT F, Y+88:LET T=1:GO TO 100
80 DRAW 1, Y-OLDY
100 LET OLDY=INT(Y+5)
110 NEXT F
```

Выполните её, введя 10 для числа 'N' и '10*TAN X' для функции. Будет вычерчен график функции TAN 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/сек в Европе и 60 в Америке). N может быть 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, определённого в операторе PAUSE, нажатием любой клавиши (надо помнить, что CAPS SHIFT будет вызывать прерывание).

Пример программы моделирования секундной стрелки часов:

```
10 REM FIST WE DRAW THE CLOCK FACE
20 FOR N=1 TO 12
30 PRINT AT 10-10*COS(N/6*PI), 16+10*SIN(N/6*PI);N
40 NEXT N
50 REM NOW WE START THE CLOCK
60 FOR T=0 TO 200000:REM T 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 задаётся в операторе с номером 60. Оператор 210 производит отсчёт времени. Казалось бы, здесь должен быть оператор PAUSE 50 (Европа) для точного отсчёта одной секунды, но тогда бы мы не учли время, затрачиваемое на выполнение остальных операторов программы. Рассматриваемый вариант часов обеспечивает 2%-ю точность или иными словами уход на полчаса в день.

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

```
(65536*PEEK 23674 + 256*PEEK 23673 + PEEK 26672)/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 моделирований часов:

```
10 REM FIRST WE DRAW THE CLOCK FACE
20 FOR N=1 TO 12
30 PRINT AT 10-10*COS(N/6*PI), 16+10*SIN(N/6*PI);N
40 NEXT N
50 DEF FN T()= INT((65536*PEEK 23674+256*PEEK 23673+PEEK 23672)
   /50):REM NUMBER OF SECOND SINCE 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 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 GO TO 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:GO TO 120
```

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

Числа PEEK 23674, PEEK 23673 и PEEK 23672 выделяют адреса ячеек памяти компьютера и 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, то она сбрасывается в 0, а значение ячейки PEEK 23673 увеличится на 1. Через каждые 256/50 сек содержимое этой ячейки переходит из состояния 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:

```
(65536*0 + 256*255 + 255)/50=1310, 7
```

но здесь имеется скрытая опасность. Через следующую 1/50 сек ячейки будут содержать соответственно следующие 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 сохранённый 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 BY 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 компьютер был включен. Например, надо установить 10 часов вечера. Вы подсчитали, что это 10*60*60*50 = 1800000 50-х долейсекунды и значит 1800000 = 65536*27 + 256*119 + 64 для присвоения трём 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 дает эта клавиша в режиме маркера , или пустая строка. Выполните программу, которая использует эту функцию:

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

Помните, что функция INEKY$ не будет подобно INPUT ждать Вас. Если Вы не выполните ввод, то считайте, что Ваш шанс упущен.

ГЛАВА 19

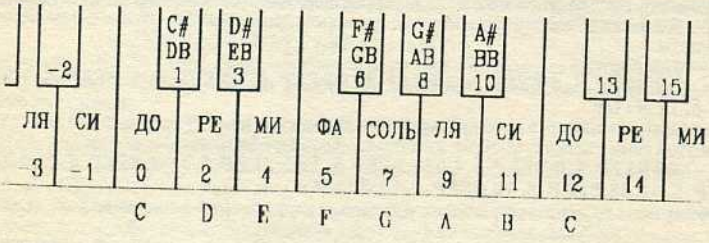
Программирование звуков

Краткое содержание : BEEP.

ZX SPECTRUM может воспроизводить звуки при помощи оператора BEEP:

BEEP продолжительность, высота звука,

где продолжительность и высота звука некоторые числовые выражения. Продолжительность задаётся в секундах, а высота в полутонах от основного тона 'ДО'; при положительных числах - выше ноты 'ДО', при отрицательных - ниже ноты 'ДО'. На диаграмме приведены все значения нот одной октавы:

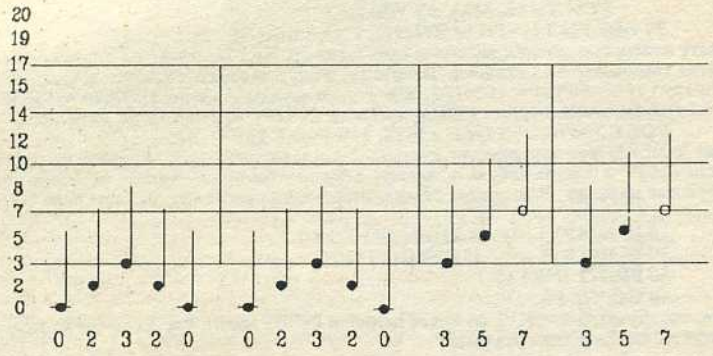


Для получения более высоких или более низких нот, Вы должны прибавить или отнять 12 для каждой октавы вверх или вниз.

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 .5, 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:BEEP1,0
70 BEEP.75, 7:BEEP.25, 8:BEEP.5, 7:BEEP.5, 5:BEEP.5, 3:BEEP.5, 2:BEEP1,0
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 также принимать значения кратные 1/2, 1/4 и т.д.

Таким же образом можно изменять и длительность звучания нот. Но помните, что компьютер может одновременно исполнять только одну ноту, что не позволяет воспроизводить сложные мелодии.

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

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

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

Фрагмент программы:

10 BEEP .5, 0:BEEP .5, 2:BEEP .5, 4:BEEP .5, 5:BEEP .5, 7: BEEP .5, 9:BEEP .5, 11:BEEP .5, 12:STOP

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

20 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, 10 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 в операторе BEEP. Например, для звука на четверть тона выше среднего ДО надо указать значение высоты звука равное .5,

Вы можете сделать клавиатуру компьютера клавишами музыкального инструмента, выполнив переключение:

POKE 23609, 255

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

Можно также вывести музыку на внешние устройства, подключаемые к выходным разъёмам '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 вводном описании. 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 введём и выполним следующую программу:

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

а затем осуществим её проверку, заменив команду VERIFY "DICE" на команду MERGE "DICE". Вы увидите, что строки 1 и 2 сохраняются а строки 10 и 20 заменяются на строки с этими номерами из программы "DICE", переменная X тоже сохраняется (проверьте PRINT X).

Теперь Вы знаете четыре оператора для работы с кассетным 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 от 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, определённое первоначально в команде 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 записывать также массивы или 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 '$'. 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 выдает: "CHARACTER ARRAY:" и далее имя этого массива.

Существует возможность записи на магнитную ленту и отдельных байтов информации. Так, например, это может быть телевизионная картинка или 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-объём в байтах одного экрана, а 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, то выдаётся сообщение "R TAPE LOADING ERROR" (ошибка загрузки с ленты). Этот параметр можно опустить и 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 16384, 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 заключении, везде, где указывается имя файла на ленте, используются только 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 выдаёт сообщение: "START TAPE, THEN PRESS ANY KEY" ("запусти магнитофон и нажми любую клавишу") и ждёт 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 **GO TO LINE NUMBER** и начинает выполнять её.

2. Байты.

SAVE NAME CODE START, LENGTH

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. Может выдавать сообщение 'R TAPE LOADING ERROR'.

1. Программа и переменные.

VERIFY NAME

2. Байты.

VERIFY NAME CODE START, LENGTH

Если данных в файле "NAME" более, чем указано в "LENGTH", то выдаётся 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$ эквивалентно VERIFY NAME CODE 16384, 6912

Однако, это будет проверка уже проверенного файла.

3. Массивы.

VERIFY NAME DATA LETTER()

VERIFY NAME DATA LETTER$()

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

1. Программа и переменные.

LOAD NAME

Может выдавать 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, с которого записывались данные на ленту в файл "NAME".
3. Массивы.
LOAD NAME DATE LETTER() или LOAD NAME DATE LETTER$()
уничтожает в памяти массив с именем "LETTER" или "LETTER$", формирует новый массив и переписывает туда данные из файла "NAME". Может выдать 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 старые.
1. Программа и переменные.
MERGE NAME
дописывает программу "NAME" к некоторой программе, находящейся в памяти. Может выдать сообщение
'4 OUT OF MEMORY'
2. Байты.
Не поддерживается.
3. Массивы.
Не поддерживается.
Пример: записать на ленту информацию о 21-м определённом пользовательском 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 Бейсика, необходимые для работы с принтером ZX.
Два оператора LPRINT и LLIST подобны операторам PRINT и LIST, но с той лишь 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, выдав BREAK (CAPS SHIFT и SPACE).
Если Вы задали операторы управления принтером без подключённого реального устройства, то вывода просто не будет и выполнение программы продолжится со следующего оператора.
Теперь попробуйте выполнить такую программу:

```
10 FOR N=31 TO 0 STEP -1
20 PRINT AT 31-N, N;  CHR$(CODE"0");
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', программа будет работать также, как и прежде. Теперь заменим в строке 20 PRINT на LPRINT и заметим, что развёртки по диагонали не получается. А 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 имеющиеся на 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 её в свою оперативную память по командам PEEK и POKE. Вся память компьютера, и ПЗУ и ОЗУ, представляется совокупностью адресов от 0 до 65535, 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 Бейсика IN и OUT.

N аналогичен оператору PEEK:

IN ADRESS

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

OUT подобен оператору POKE:

OUT ADRESS, VLUE

и записывает указанные данные в заданный порт вывода.

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

A15, A14, A13, A12. . . . . , A1, A0.

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

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

Биты A8, A9 используются иногда для получения дополнительной информации.

Информационный байт 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 разъёма 'EAR'. Сама клавиатура разбита 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 7,
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 разъёма 'EAR'.

Выходной порт 254 обеспечивает громкоговоритель (D4) и разъём '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 HELF=ROW NUMBER (номер полуряда)
20 LET A=254+256*(255-2**N)
30 PRINT AT 0, 0: IN A : GO TO 30
```

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

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 память по определённому адресу от 0 до FFFFH (H - здесь и далее означает шестнадцатиричное представление числа). Сам адрес может быть записан в память как два байта. На диаграмме показано распределение памяти компьютера ZX SPECTRUM:

0	3FFFh	4000h	7FFFh	8000h	FFFFh
ROM(ПЗУ)		RAM(ОЗУ) Системная область		RAM(ОЗУ)Доступно для программ	
0	16383	16384	32767	32768	65535

Для получения содержимого области любой из памятей 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 "ADRESS"; 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 ADRESS, NEW CONTENTS, где "ADRESS" и "NEW CONTENTS" - числовые выражения. Например:

```
POKE 31000, 57
```

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

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

Область экрана	Атрибуты	Буфер принтера	Системные переменные	План дополнительной памяти	
16384	22528	23296	23552	23734	chans

Канальная иформация	80h	Программа на Бейсике	Переменные программы	80h	Редактирование строки программы	NL
chans		PROG	VARS		E LINE	WORKSP

Считанные данные	NL	Рабочая область	Стек калькулятора	Резерв	Аппаратный стек	Стек переходов е подпрограммам

WORKSP STKBOT STKEND SP

7	3Eh	Определяемые пользователем символы

RAMTOP UDG P 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-ми точек). Однако эти восемь байт хранятся в памяти раздельно. Полный экран представляет собой 24 строки по 32 символа. Каждая строка экрана прописывается 8-мью строками развёртки телевизионного экрана. Итого, для записи одного экрана выполняется 192 сканирования и в памяти рядом хранятся байты одноимённых 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 системным 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: клавиатуре (с нижней половиной экрана), верхней половине экрана и принтере.

Каждая строка области Бейсик-программ имеет формат:

2 БАЙТА	2 БАЙТА	ТЕКСТ	0Dh

Номер строки Длина текста + ENTER ENTER

Числовые константы в программе представлены в двоичной форме, используя CHR$14 и следующие за ним 5 байт самого числа.

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

60h	1 БАЙТ порядка	БИТ знака	4 БАЙТА мантиссы

Буква Значение числа

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

60h	N1	...	NN	5 БАЙТ значения числа

1-ая буква 2-ая буква Последняя буква

Формат размещения числового массива:

N	2 БАЙТА	1 БАЙТ	2 БАЙТА	...	2 БАЙТА
Буква	Общая длина элементов +1 на каждое измерение	Номера размерностей	Первая размерность		Последняя размерность

Порядок 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 по следующей размерности и т. д. Например, элементы массива с размерностью (3, 4) расположатся в памяти в следующем порядке:

B(1, 1), B(1, 2), B(1, 3), B(1, 4), B(2, 1), B(2, 2), . . . , B(3, 4)

Формат размещения управляющих переменных для FOR-NEXT операторов:

Мл. байт Ст. байт

	5 БАЙТ	5 БАЙТ	5 БАЙТ	2 БАЙТА	1 БАЙТ
Буква	Значение	Ограничение	Приращение	Строка цикла	N 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	2 БАЙТА	По одному байту на каждый элемент
Буква	Общее число элементов на каждую размерность	Номера размерностей	Первая размерность	Последняя размерность	Элементы

Программируеый 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,

по которому:

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.

Функция RUN также выполняет действия CLEAR, хотя и не изменяет значение RAMTOP.

Используя функцию CLEAR, Вы можете смещать RAMTOP, увеличивая область для Бейсика, уменьшая тем самым область определяемых пользователем символов. Можно несколько увеличить доступную часть RAM (ОЗУ), используя функцию NEW. Например, выполнение NEW, 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, изменив число по адресу 23608. По умолчанию предполагается 64.

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

+ M * 2 ** E,

где M - мантисса в интервале. 5. . . 1 (не может быть 1);

E - экспонента, положительное или отрицательное число.

Допустим, Вы записали "M" в двоичной системе счисления, "M" - дробное и имеет двоичную точку (подобно десятичной точке), то тогда будет: 1/2-->.1 1/4 -->.01 3/4-->.11 и т. д.

Наше число "M" 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 числа 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 т. д. до пятого байта;

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

в) записываем в первый байт экспоненту +128. Например, мы хотим записать число 1/10:

$1/10 = 4/5 \times 2$

мантисса будет. 11001100110011001100110011001100B

(поскольку 33-й бит равен 1, мы должны округлить 32-й бит, записав 1 вместо 0), а экспонента равна -3.

Теперь, применив три наших правила, запишем 5 байт:

01111101	0	1001100	11001100	11001100	11001101
-3+128	Знак	Мантисса (4 байта из 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) 3 и 4 байты содержат младшие и старшие значащие биты числа (или число +131072, если оно отрицательное); г) 5 байт равен 0.

ГЛАВА 25

Системные переменные

Байты памяти с 23552 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 к этим переменным из Бейсик-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 изменяться, 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*REEK(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-х долях секунды
1	23562	REPPER	Задержка в 50-х долях секунды между последовательными опросами клавиш. Начальное значение 5.
N2	23563	DEFADD	Адрес аргументов функции пользователя
N1	23565	K DATA	Второй байт управления цветом с клавиатуры.
N2	23566	TVDATA	Байты цвета
X38	23568	STRMS	Адреса подключенных каналов.
2	23606	CHARS	Адрес символьного набора-256. Обычно этот набор находится в ПЗУ
1	23608	RASP	Продолжительность звукового сигнала.
1	23609	RIP	Длительность задержки
1	23610	ERR NR	Код сообщения-1. Начальное значение 255 (для '-1')
X1	23611	FLAGS	Управляющие флажки Бейсика.

Зн.	Адрес	Имя	Содержание
X1	23612	TV FLAG	Флажок телевизора.
X2	23613	ERR SP	Адрес в аппаратном стеке
N2	23615	LIST SP	Адрес возврата из автоматического листинга.
N1	23617	MODE	Режим спецификации [K]
2	23618	NEW PPC	Номер строки
2	23621	PPC	Номер строки в которой оператор.
1	23623	SUB PPC	Порядковый номер выполняющегося оператора в строке.
1	23624	DDRDCR	Цвет рамки экрана
2	23625	E PPC	Количество текущих строк (с курсором).
X2	23627	VARS	Адреса переменных.
N2	23629	DEST	Адрес переменной в задании
X20	23631	CHANS	Адрес канала данных.
X2	23633	CURCHL	Адрес данных для ввода-вывода.
X2	23635	PROG	Адрес Бейсик-программы.
X2	23637	NXTLIN	Адрес следующей строки в программе.
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
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
1	23658	FLAGS2	Старшие флажки.
X1	23659	DF SZ	Число строк (включая одну чистую) в нижней чпсти экрана.
2	23660	S TOP	Количество верхних строк прграммы в автоматическом листинге.
2	23662	OLDPPC	Номер строки на которую указывает CONTINUE.
1	23664	OSPPC	Номер оператора в строке
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
3	23672	FRAMES	Счетчик кадров - приращение через 20 ms (см. главу 18).

Зн.	Адрес	Имя	Содержание
2	23675	UDG	Адрес первого определяемого пользователем символа.
1	23677	COORDS	X - координата точки грфопостроителя.
1	23678	COORDS	Y - координата точки графопостроителя.
1	23679	P POINTS	33-позиционное число для позиционироанич принтера.
1	23680	PR CC	Младши байт адреса позиции для LPRINT для печати.
1	23681		Не используется.
2	23682	ECHO E	33-позиционное и 24-строковое число (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	23688	S POSN	33-позиционное число для PRINT-позиции.
X1	23689	S POSN	23-строковое число для PRINT-позиции.
X2	23690	S POSNL	Подобно S POSN для ни. ней части.
1	23692	SCR CT	Счетчик сверток: всегда на 1 больше числа сверток
1	23693	ATTR P	Сплошные цвета.
1	23694	MASK P	Используется для высвечивания цветов. Бит
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 калькулчтора. Используется для записи чисел
2	23728		Не используется
2	23730	RAMTOP	Адрес последнего байта области Бейсик-системы.
2	23732	P-RAMT	Адрес последнего байта физического ОЗУ.

Следующая программа выдаст 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

```
20 PRINT PEEK(23755+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.

Приведём 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:

	Определяемые пользователем символы.	
RAMTOP=23599	UDG=32600	PRAMT=32767

Если Вы теперь выполните:

CLEAR 32499

то получите дополнительно 100 байтов памяти, начиная с адреса 32500.

	100 БАЙТОВ		Определяемые пользователем символы	
RAMTOP=23599	32500		UDG=32600	PRAMT=32767

Для загрузки программы в машинных ходах 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 с 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 I.

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 чего надо сделать:

SAVE NAME LINE

а затем:

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

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

Полный набор символов

DEC код	HEX код	Символ	Ассемблерная мнемоника	CBH...	EDH...
0	00	не использ.	NOP	RLC B	
1	01	не использ.	LD BC,NN	RLC C	
2	02	не использ.	LD (BC),NN	RLC D	
3	03	не использ.	INK BC	RLC E	
4	04	не использ.	INK B	RLC H	
5	05	не использ.	DEC B	RLC L	
6	06	PRINT упр.	LD B,N	RLC (HL)	
7	07	EDIT	RLCA	RLC A	
8	08	курс. влево	EX AF,AF'	RRC B	
9	09	курс. вправо	ADD HL,BC	RRC C	
10	0A	курс. вниз	LD A,(BC)	RRC D'	
11	0B	курс .вверх	DEC BC	RRC E	
12	0C	DELETE	INC C	RRC H	
13	0D	ENTER	DEC C	RRC L	
14	0E	число	LD C,N	RRC (HL)	
15	0F	не использ.	RRCA	RRC A	
16	10	не использ.	DJNZ DIS	RL B	
17	11	PAPERупр.	LD DE,NN	RL C	
18	12	FLASHупр.	LD(DE), A	RL D	
19	13	BRIGHTупр.	INC DE	RL E	
20	14	INVERSEупр.	INC D	RL H	
21	15	OVERупр.	DEC D	RL L	
22	16	ATупр.	LD D,N	RL (HL)	
23	17	TABупр.	RLA	RL A	
24	18	не использ.	JP DIS	RR B	
25	19	не использ.	ADD HL,DE	RR C	
26	1A	не использ.	L A,(DE)	RR D	
27	1B	не использ.	DEC DE	RR E	
28	1C	не использ.	INC E	RR H	
29	1D	не использ.	DEC E	RR L	
30	1E	не использ.	LD E,N	RR (HL)	
31	1F	не использ.	RRA	RR A	
32	20	пробел	JP NZ,DIS	SLA B	
33	21	!	LD HL,NN	SLA C	
34	22	"	LD (N),HL	SLA D	
35	23	#	INC HL	SLA E	
36	24	$	INC H	SLA H	
37	25	%	DEC H	SLA L	
38	26	&	LD H,N	SLA (HL)	
39	27	'	DAA	SLA A	
40	28	(	JP 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 L	
46	2E	.	LD L,N	SRA (HL)	
47	2F	/	CPL	SRA A	
48	30	0	JP NC,DIS		
49	31	1	LD SP,NN		
50	32	2	LD (NN),A		
51	33	3	INC SP		
52	34	4	INC (HL)		
53	35	5	DEC (HL)		
54	36	6	LD (HL),N		

55	37	7
56	38	8
57	39	9
58	3A	:
59	3B	;
60	3C	<
61	3D	=
62	3E	>
63	3F	?
64	40	@
65	41	A
66	42	B
67	43	C
68	44	D
69	45	E
70	46	F
71	47	G
72	48	H
73	49	I
74	4A	J
75	4B	K
76	4C	L
77	4D	M
78	4E	N
79	4F	O
80	50	P
81	51	Q
82	52	R
83	53	S
84	54	T
85	55	U
86	56	V
87	57	W
88	58	X
89	59	Y
90	5A	Z
91	5B	[
92	5C	/
93	5D	]
94	5E	стрелка врх.
95	5F	подчерк.
96	60	фунт-стерл.
97	61	A(строчн)
98	62	B(строчн)
99	63	C(строчн)
100	64	D(строчн)
101	65	E(строчн)
102	66	F(строчн)
103	67	G(строчн)
104	68	H(строчн)
105	69	I(строчн)
106	6A	J(строчн)
107	6B	K(строчн)
108	6C	L(строчн)
109	6D	M(строчн)
110	6E	N(строчн)
111	6F	O(строчн)
112	70	P(строчн)
113	71	Q(строчн)
114	72	R(строчн)
115	73	S(строчн)

```
SCF
JP C,DIS
ADD HL,SP
LD A,(NN)
DEC SP
INC A
DEC A
LD A,N
CCF
LD B,B
LD B,C
LD B,D
LD B,E
LD B,H
LD B,L
LD B,(HL)
LD B,A
LD C,B
LD C,C
LD C,D
LD C,E
LD C,H
LD C,L
LD C,(HL)
LD C,A
LD D,B
LD D,C
LD D,D
LD D,E
LD D,H
LD D,L
LD D,(HL)
LD D,A
LD E,B
LD E,C
LD E,D
LD E,E
LD E,H
LD E,L
LD E,(HL)
LD E,A
LD H,B
LD H,C
LD H,D
LD H,E
LD H,H
LD H,L
LD H,(HL)
LD H,A
LD L,B
LD L,C
LD L,D
LD L,E
LD L,H
LD L,L
LD L,(HL)
LD L,A
LD (HL),B
LD (HL),C
LD (HL),D
LD (HL),E
```

```
SRL B
SRL C
SRL D
SRL E
SRL H
SRL L
SRL (HL)
SRL A
BIT 0,B      IN B,(C)
BIT 0,C      OUT (C),B
BIT 0,D      SRC HL,BC
BIT 0,E      LD (NN),BC
BIT 0,H      NEG
BIT 0,L      RETN
BIT 0,(HL)   IM 0
BIT 0,A      LD I,A
BIT 1,B      IN C,(C)
BIT 1,C      OUT C,(C)
BIT 1,D      ADD HL,BC
BIT 1,E      LD RC,(NN)
BIT 1,H
BIT 1,L      RETT
BIT 1,(HL)
BIT 1,A      LD R,A
BIT 2,B      IN D,(C)
BIT 2,C      OUT (C),D
BIT 2,D      SBC HL,DE
BIT 2,E      LD (NN),DE
BIT 2,H
BIT 2,L
BIT 2,(HL)   IM 1
BIT 2,A      LD A,I
BIT 3,B      IN E,(C)
BIT 3,C      OUT (C),E
BIT 3,D      ADC HL,DE
BIT 3,E      LD DE,(NN)
BIT 3,H
BIT 3,L
BIT 3,(HL)   IM 2
BIT 3,A      LD A,R
BIT 4,B      IN H,(C)
BIT 4,C      OUT (C),H
BIT 4,D      SBC HL,HL
BIT 4,E      LD (NN),HL
BIT 4,H
BIT 4,L
BIT4,(HL)
BIT4,A       RRD
BIT5,B       IN L,(C)
BIT5,C       OUT (C),L
BIT5,D       ADC HL,HL
BIT5,E       LD HL,(NN)
BIT5,H
BIT5,L
BIT5,(HL)
BIT5,A       RID
BIT6,B       IN F,(C)
BIT6,C
BIT 6,D      SBC HL,SP
BIT 6,E      LD (NN),SP
```

116	74	T(строчн)	LD (HL),H	BIT 6,H	
117	75	U(строчн)	LD (HL),L	BIT 6,L	
118	76	V(строчн)	HALT	BIT 6,(HL)	
119	77	W(строчн)	LD (HL),A	BIT 6,A	
120	78	X(строчн)	LD A,B	BIT 7,B	IN A,(C)
121	79	Y(строчн)	LD A,C	BIT 7,C	OUT (C),A
122	7A	Z(строчн)	LD A,D	BIT 7,D	ADC HL,SP
123	7B	фигур.ск.лев.	LD A,E	BIT 7,E	LD SP,(NN)
124	7C	верт.черта	LD A,H	BIT 7,H	
125	7D	фигур.ск.прв.	LD A,L	BIT 7,L	
126	7E	дефис	LD A,(HL)	BIT 7,(HL)	
127	7F	с вокр	LD A,A	BIT 7,A	
128	80	оо оо	ADD A,B	RES 0,B	
129	81	ож оо	ADD A,C	RES 0,C	
130	82	жо оо	ADD A,D	RES 0,D	
131	83	жо оо	ADD A,E	RES 0,E	
132	84	оо ож	ADD A,H	RES 0,H	
133	85	ож ож	ADD A,L	RES 0,L	
134	86	жо	ADD A,(HL)	RES 0,(HL)	
135	87	жж ож	ADD A,A	RES 0,A	
136	88	о о жо	ADC A,B	RES 1,B	
137	89	ож жо	ADC A,C	RES 1,C	
138	8A	жо жо	ADC A,D	RES 1,D	
139	8B	жж жо	ADC A,E	RES 1,E	
140	8C	оо жж	ADC A,H	RES 1,H	
141	8D	ож жж	ADC A,L	RES 1,L	
142	8E	жо жж	ADC A,(HL)	RES 1,(HL)	
143	8F	жж жж	ADC A,A	RES 1,A	
144	90	(A)	SUB B	RES 2,B	
145	91	(B)	SUB C	RES 2,C	
146	92	(C)	SUB D	RES 2,D	
147	93	(D)	SUB E	RES 2,E	
148	94	(E)	SUB H	RES 2,H	
149	95	(F)	SUB L	RES 2,L	
150	96	(G)	SUB (HL)	RES 2,(HL)	
151	97	(H)	SUB A	RES 2,A	
152	98	(I)	SBC B	RES 3,B	
153	99	(J)	SBC C	RES 3,C	
154	9A	(K)	SBC D	RES 3,D	
155	9B	(L)	SBC E	RES 3,E	
156	9C	(M)	SBC H	RES 3,H	
157	9D	(N)	SBC L	RES 3,L	
158	9E	(O)	SBC (HL)	RES 3,(HL)	
159	9F	(P)	SBC A	RES 3,A	
160	A0	(Q)	AND B	RES 4,BLDI	
161	A1	(R)	AND C	RES 4,CCPI	

162	A2	(S)
163	A3	(T)
164	A4	(U)
165	A5	RND
166	A6	INKEY$
167	A7	PI
168	A8	FN
169	A9	POINT
170	AA	SCREEN$
171	AB	ATTR
172	AC	AT
173	AD	TAB
174	AE	VAL$
175	AF	CODE
176	B0	VAL
177	B1	LEN
178	B2	SIN
179	B3	COS
180	B4	TAN
181	B5	ASN
182	B6	ACS
183	B7	ATN
184	B8	LN
185	B9	EXP
186	BA	INT
187	BB	SQR
188	BC	SGN
189	BD	ABS
190	BE	PEEK
191	BF	IN
192	C0	USR
193	C1	STR$
194	C2	CHR$
195	C3	NOT
196	C4	BIN
197	C5	OR
198	C6	AND
199	C7	<=
200	C8	>=
201	C9	<>
202	CA	LINE
203	CB	THEN
204	CC	TO
205	CD	STEP
206	CE	DEF FN
207	CF	CAT
208	D0	FORMAT
209	D1	MOVE
210	D2	ERASE
211	D3	OPEN
212	D4	CLOSE
213	D5	MERGE
214	D6	VERIFY
215	D7	BEEP
216	D8	CIRCLE
217	D9	INK
218	DA	PAPER
219	DB	FLASH
220	DC	BRIGHT

AND D
AND E
AND H
AND L
AND (HL)
AND A
XOR B
XOR C
XOR D
XOR E
XOR H
XOR L
XOR (HL)
XOR A
OR B
OR C
OR D
OR E
OR H
ORL
OR(HL)
OR A
CP B
CP C
CP D
CP E
CP H

CP L
CP (HL)
CP A
RET NZ
POP BC
JP NZ,NN
JP NN
CALL NZ,NN
PUSH BC
ADD A,N
RST 0
RET Z
RET
JP Z,NN

CALL Z,NN
CALL NN
ADC A,N
RST 8
RET NC
POP DE
JPN C,NN
OUT (N),A
CALL NC,NN
PUSH DE
SUB N
RST 16
RET C
EXX
JP C,NN
IN A,(N)
CALL C,NN

RES 4,D	INI
RES 4,E	OUTI
RES 4,H	
RES 4,L	
RES 4,(HL)	
RES 4,A	
RES 5,B	LDD
RES 5,C	CPD
RES 5,D	IND
RES 5,E	OUTD
RES 5,H	
RES 5,L	
RES 5,(HL)	
RES 5,A	
RES 6,B	LDIR
RES 6,C	CPIR
RES 6,D	INIR
RES 6,E	OTIR
RES 6,H	
RES 6,L	
RES 6,(HL)	
RES 6,A	
RES 7,B	LDDR
RES 7,C	CPDR
RES 7,D	INDR
RES 7,E	OTDR
RES 7,H	

RES 7,L
RES 7,(HL)
RES 7,A
SET 0,B
SET 0,C
SET 0,D
SET 0,E
SET 0,H
SET 0,L
SET 0,(HL)
SET 0,A
SET 1,B
SET 1,C
SET 1,D
SET 1,E
SET 1,H
SET 1,L
SET 1,(HL)
SET 1,A
SET 2,B
SET 2,C
SET 2,D
SET 2,E
SET 2,H
SET 2,L
SET 2,(HL)
SET 2,A
SET 3,B
SET 3,C
SET 3,D
SET 3,E
SET 3,H

221	DD	INVERSE	PREFIXES IN-STRUCTIONS USING IX	SET 3,L
222	DE	OVER	SBC A,N	SET 3,(HL)
223	DF	OUT	RST 24	SET 3,A
224	E0	LPRINT	RET PC	SET 4,B
225	E1	LLIST	POP HL	SET 4,C
226	E2	STOP	JP PC,NN	SET 4,D
227	E3	READ	EX (SP),HL	SET 4,E
228	E4	DATA	CALL PC,NN	SET 4,H
229	E5	RESTORE	PUSH HL	SET 4,L
230	E6	NEW	AND N	SET 4,(HL)
231	E7	BORDER	RST 32	SET 4,A
232	E8	CONTINUE	RET PE	SET 5,B
233	E9	DIM	JP (HL)	SET 5,C
234	EA	REM	JP PE,NN	SET 5,D
235	EB	FOR	EX DE,HL	SET 5,E
236	EC	GOTO	CALL PE,NN	SET 5,H
237	ED	GO SUB		SET 5,L
238	EE	INPUT	XOR N	SET 5,(HL)
239	EF	LOAD	RST 40	SET 5,A
240	F0	LIST	RET P	SET 6,B
241	F1	LET	POP AF	SET 6,C
242	F2	PAUSE	JP P,NN	SET 6,D
243	F3	NEXTE	DI	SET 6,E
244	F4	POKE	CALL P,NN	SET 6,H
245	F5	PRINT	PUSH AF	SET 6,L
246	F6	PLOT	OR N	SET 6,(HL)
247	F7	RUN	RST 48	SET 6,A
249	F9	RANDOMIZE	LD SP,HL	SET 7,C
250	FA	IF	JP M,NN	SET 7,D
251	FB	CLS	EI	SET 7,E
252	FC	DRAW	CALL M,NN	SET 7,H
253	FD	CLEAR	PREFIXES INSTRUCTIONS USINGIY	SET 7,L
254	FE	RETURN	CP N	SET 7,(HL)
255	FF	COPY	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 сообщение помогает найти ошибочную строку и ошибочный оператор в этой строке (команда указывается как строка 0, оператор 1 располагается в строке первым, оператор 2 после первого или THEN и т.д.).

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

Перечень сообщений представлен в формате:

код - значение (перевод)
ситуация
комментарий.

0 - OK (О'кей! Порядок!)
Разное

Успешное завершение или переход на строку с номером большим чем имеется всего. Это сообщение не меняет строки или оператора, определённого для CONTINUE.

1 - NEXT WITHOUT FOR (NEXT без FOR)
NEXT

Управляющей переменной нет(не была определена в юператоре FOR), но есть обычная переменная с тем же именем.

2 - VARIABLE NOT FOUND (переменная не найдена)
Разное

Для простой переменной выдаётся, если она используется без предварительного определения в операторах LET, READ или INPUT, или загружается с ленты или устанавливается в операторе FOR. Для индексируемой переменной сообщение выдаётся, если она не была предварительно определена в операторе DIM перед использованием или загрузкой с ленты.

3 - SUBSCRIPT WRONG (ошибочный индекс)
В индексной переменной или подстроке

Индекс превышает размерность массива, либо ошибочное число задает индекс. если индекс отрицательный или больше 65535, то выдаётся сообщение 8.

4 - OUT OF MEMORY (вне памяти)
LET, INPUT, FOR, DIM, GO SUB, LOAD, MERGE

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 впоследствии), получить дополнительную память маневрируя оператором CLEAR.

5 - OUT OF SCREEN (вне экрана)
INPUT, PRINT AT

Если INPUT оператор генерирует больше чем 23 строки в нижней полвине экрана. Также встречается с PRINT AT 22, ...

6 - NUMBER TOO BIG (число больше максимально допустимого)
Арифметические операции

В результате вычислений получилось число больше 10^{38}

7 - RETURN WITHOUT GO SUB (RETURN без GO SUB)

RETURN

Встретилось больше операторов RETURN, чем было операторов GO SUB.

8 - END OF FILE (конец файла)
Операции с внешней памятью

9 - STOP STATEMENT (оператор STOP)
STOP

После этого сообщения CONTINUE не может повторить STOP, но может передать управление на следующий оператор.

A - INVALID ARGUMENT (ошибочный аргумент)
SQR, LN, ASN, ACS, USR (со строковым аргументом), RUN, DIM,
Аргумент функции не допустим в данной версии

B - INTEGER OUT OF RANGE (переполнение целого)
RANDOMIZE, GO SUB, GO TO, PLOT, CHR$, LIST, LLIST, PEEK, PAUSE, USR (с числовым аргументом)
Выдаётся, когда аргумент с плавающей точкой округляется к целому. Для случая массивов смотри также сообщение 3.

C - NONSENSE IN BASIC (выражение не Бейсика)
VAL, VAL$
Текст (строка) не распознается Бейсиком как допустимое выражение.

D - BREAK-CONT REPEATS
LOAD, SAVE, VERIFY, MERGE, LPRINT, LLIST, COPY (только когда компьютер запросил свёртку, а Вы ответили N, SPACE или STOP).
Клавиша BREAK нажата во время действия периферийной операции. Действия CONTINUE после этого оператора обычные, те что указаны в операторе. Сравните с сообщением L.

E - OUT OF DATA (вне данных)
READ
Попытка выдать READ когда список данных в DATA кончился.

F - INVALID FILE NAME (неверное имя файла)
SAVE
Оператор SAVE с пустой строкой вместо имени или с именем длиннее 10 символов.

G - NO ROOM FOR LINE (нет места для строки)
Ввод строки в программу
Недостаточно места в памяти для записи очередной строки программы.

H - STOP IN INPUT
INPUT
Некоторые введённые данные начинаются с оператора STOP, или была нажата INPUT LINE. действия CONTINUE - обычное.

I - FOR WITHOUT NEXT (FOR без NEXT)
FOR
Цикл FOR ни разу не выполнялся, не найден NEXT оператор.

J - INVALID I/O DEVICE (неверное устройство ввода - вывода)
В операциях с внешними устройствами

K - INVALID COLOUR (неверный цвет)

PAPER, BORDER, FLASH, BRIGHT, INVERSE, OVER, а также после одной из передач управляющих символов.

Специфицированное число имеет неверное значение.

L - BREAK INTO PROGRAM (BREAK во время выполнения программы)
Разное

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.

M - RAMTOP NO GOOD (адрес RAMTOP не годен)
CLEAR, возможно RUN

Число, указанное для RAMTOP, слишком велико или слишком мало.

N - STATEMENT LOST (оператор отсутствует)
RETURN, NEXT, CONTINUE

Переход к оператору, которого уже нет.

O - INVALID STREAM (ошибочный поток данных)
В операциях ввода-вывода

P - FN WITHOUT DEF (FN без DEF)
FN

Определяемая пользователем функция не определена в операторе DEF FN.

Q - PARAMETER ERROR (ошибка в параметре)
FN

Ошибочное число аргументов или один из них не того типа, какой был описан.

R - TAPE LOADING ERROR (ошибка загрузки с ленты)
VERIFY, LOAD, MERGE

Файл на ленте найден, но не может быть считан.

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 клавиши определяется частично переключающими клавишами (CAPS SHIFT и SYMBOLSHIFT), а частью режимом, в котором находится компьютер.

Режим отображается курсором, мерцающей буквой, указывающей позицию, в которую будет вводится 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, нанесённый на эту клавишу. Для обоих режимов L и K при нажатии с клавишей одновременно и клавиши SYMBOLSHIFT, 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 даёт один символ или знак (изображённый на ЗЕЛЕНОМ ПОЛЕ клавиши), если НЕ НАЖАТА переключающая клавиша, или знак, изображённый на КРАСНОМ ПОЛЕ, если УДЕРЖИВАЕТСЯ переключающая 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 DELET, а каждая алфавитная клавиша, кроме V, W, X, Y и Z, будет выдавать определённый пользователем графический символ.

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

Ввод с клавиатуры производится в нижнюю часть экрана, каждый символ (или составной знак) вставляются перед курсором. Курсор может быть перенесен влево действием CAPS SHIFT и 5, а вправо CAPS SHIFT и 8 , символ перед курсором можно удалить командой DELETE (CAPS SHIFT и 0), целая строка может быть удалена вводом EDIT (CAPS SHIFT и 1) и последующим ENTER. Когда нажимается ENTER, выполняется набранная строка, либо она вводится в программу, либо она используется как входные данные для INPUT оператора, либо 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 введённая строка называется текущей и отмечается символом [>], и её можно изменить, используя клавиши перемещения курсора вверх и вниз (CAPS SHIFT и 6 ; CAPS SHIFT и 7), если введён 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 результаты отображаются в верхней половине экрана и сохраняются до ввода строки программы, ввода пустой строки или нажатия клавиши управления курсором вверх или вниз. В нижней части выдаются сообщения и коды, приведённые в приложении B. В сообщении указывается номер ошибочной строки (0 для команды) и позиция оператора в этой строке, сообщение сохраняется на экране до нажатия любой клавиши (отображается переходом в режим K).

В определённых обстоятельствах CAPS SHIFT и SPACE действуют как BREAK, останавливая компьютер с сообщениями D и L, и при этом до останова:

а) завершается выполняющийся оператор или

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

Пример использования клавиатуры:

при курсоре [K] :
1.простое нажатие клавиши- на экране - COPY
2.SIMBOL SHIFT и клавиша- на экране - Z

при курсоре [L]:
3.CAPS SHIFT и клавиша- на экране - :
4.BREAK/SPACE и клавиша- на экране - Z
5.SYMBOL SHIFT и CAPS SHIFT - курсор - [E]

при курсоре [E]:
6.простое нажатие клавиши- на экране - LN
7.CAPS SHIFT и клавиша- на экране - BEEP

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 свёртка 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 REPORTS'. Нажатие других клавиш разрешает свёртку 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 свёртка.

Каждая позиция имеет атрибуты, определяющие её как чистую (цвет фона), либо как закрашенную (основной цвет), с повышенной или пониженной яркостью, мерцающую или нет. Доступные цвета: чёрный, голубой, красный, фиолетовый, зелёный, синий, жёлтый и белый. Края экрана могут быть установлены в определённый 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 чёрный цвет для закрашенной точки (INK) и белый для чистой (PAPER), нормальную яркость, не мерцающие, не инверсные. Постоянные параметры для нижней части экрана используют цвет рамки (BORDER COLOUR) как цвет фона (незакрашенный), с чёрным 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.

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

Параметры PAPER и INK могут принимать значения от 0 до 9.

Параметры от 0 до 7 определяют цвета выводимого символа:

0 - чёрный(BLACK)
1 - синий(BLUE)
2 - красный(RED)
3 - фиолетовый(MAGENTA)
4 - зелёный(GREEN)
5 - голубой(CYAN)
6 - жёлтый(YELLOW)
7 - белый(WHITE)

Параметр 8 определяет, что цвет должен остаться при выводе без изменения. Параметр 9 (контрастность) определяет, что цвет должен стать либо белым, либо чёрным для выделения его от других цветов.

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

Параметры OVER и INVERSE могут принимать значения 0 или 1:

OVER 0 - новый символ затирает старый;

OVER 1 - ход старого символа и нового символа соединяются операцией 'исключающего ИЛИ', образуя новый символ (OVERPRINTING)

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' (указанным в TAB) и затем выводом нужного количества пробелов для смещения текущей позиции вывода в колонку 'N'.

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

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

Принтер

Вывод на принтер осуществляется через буфер длиной в 32 символа. Очередная строка выдаётся 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 ещё остались другие не выведенные 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}, а наименьшее положительное число $4*10^{-39}$. Числа имеют внутреннее представление как числа с плавающей (двоичной) точкой, с выделением одного байта на показатель степени 'E' (экспоненты) в интервале от 1 до 255, и четырёх байтов на мантиссу 'M' в интервале от 0.5 до 1 (M =1). Это представляется числом $M*2^{E(-128)}$.

Поскольку 1/2<=M<1, старший значащий бит мантиссы всегда 1. Следовательно, мы можем заменять его на бит, обозначающий знак 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 четвёртый 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 1.

Строки символов более гибкие в своей длине. Имя строковой переменной в отличие от простой переменной заканчивается символом 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, задающим последнюю размерность в операторе DIM. Начальный индекс 1.

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

а) пустым;

б) числовым выражением;

в) некоторым 'числовым выражением' 'TO' другим 'числовым выражением' и использоваться в:

*) строковых выражениях (сечениях);

**) строковых массивах переменных

(индекс 1, индекс 2, ..., индекс N, сечение)

или, что тоже самое

(индекс 1, индекс 2, ..., индекс N)(сечение).

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

Если сечение представляется числовым выражением со значением 'M' то результатом будет M-ый символ от S$ (подстрока длиной 1).

Если сечение представлено в форме в) и первое числовое выражение имеет значение 'M' (умалчиваемое значение 1), а второе 'N' (умалчиваемое значение S$), и если 1<=M<=H чем длина S$, то результатом будет подстрока от S$ cM-ым начальным символом и N-ым конечным.

Если 0<=H<M, то результатом будет пустая строка. В любом другом случае выдаётся сообщение по 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

ФУНКЦИИ

Функции представлены в формате

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 интервале от -1 до 1.

AND
Логическая операция. Правый операнд всегда число. Слева может быть число или строка.
если левый операнд число, тогда A AND B = A (при B<>0) или A AND B = B (при B = 0);
если левый операнд строка, тогда A$ AND B = A$ (при B<>0) или A$ AND B = " " (при B = 0);

ASN
Число
Арксинус в радианах, выдаёт сообщение A, если X не лежит в интервале от -1 до 1.

ATN
Число
Арктангенс в радианах

ATTR
Два числовых аргумента X и Y, заключаемые 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 с 2 по 1 - цвет закрашивания. Выдаёт 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, чей код представим числом X, округлённым к ближайшему целому.

CODE
Строка
Код первого символа в строке X (или 0, символов если X - пустая строка).

COS
Число в радианах
Косинус X

EXP
Число
E в степени X

FN
FN с последующим именем, определённой пользователем функции (см. DEF). Аргументы должны заключаться в скобки. Даже если нет аргументов, скобки все равно должны записываться.

IN
Число
Осуществляется ввод на уровне микропро- цессора из порта X (0<=X<=FFFFh). Загружается пара регистров BC и выполняется команда IN A (C).

INKEY$
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
Округление к ближайшему меньшему целому

LEN
Строка
Длина строки символов

LN
Число
Натуральный логарифм. Выдаёт сообщение A, если X<=0.

NOT
Число
0, если X<>0;
1, если X=0.
Операция имеет четвёртый приоритет.

OR
Оба операнда числа
A OR B = 1, если B<>0;
A OR B = A, если B=0.
Операция имеет второй приоритет.

PEEK
Число
Значение байта в памяти по адресу X, округлённому к ближайшему целому.

PI
Число Пи (3.14159265...)

POINT

Два числовых аргумента X и Y, заключённых в скобки

1, если точка экрана с координатами (X, Y) закрашена.
0, если эта точка имеет цвет фона.
Выдаёт сообщение B, если не выполняются условия 0<=X<=255 и 0<=Y<=175.

RND

Очередное псевдослучайное число из последовательности, получаемой возведением в 75 степень модуля числа 65537, вычитанием 1 и делением на 65536, число лежит в интервале 0<=Y<=1.

SCREEN$

Два числовых аргумента X и Y, заключённых 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 X выводится.

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 даёт 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 даёт не строковое (числовое) значение.

ОПЕРАЦИИ

Префиксные:

-<*число*>: отрицательное значение
Инфиксные (двухоперандовые):
+: сложение
-: вычитание
*: умножение
/: деление
**: возведение в степень (стрелка вверх).
Сообщение B, если левый операнд
отрицательный.
=: равенство
>: больше;
<: меньше;
>=: больше или равно;
<=: меньше или равно
<>: Оба операнда должны быть одного типа. Результат равен 1, если сравнение истинно и 0, если нет.

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

ОПЕРАТОРЫ

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 оператора в строке, хотя есть некоторые ограничения в IF и REM. Все операторы языка сведены в следующую таблицу:

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 0 м 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

COPY

Пересылает копию 22 строк экрана на принтер, если он подключен. Помните, что по COPY нельзя распечатать находящийся на экране автоматический листинг. Выдаёт сообщение D, если нажать клавишу BREAK.

CONTINUE

Продолжает выполнение программы, начатой ранее и остановленной с сообщением, отличным от 0. Если было сообщение 9 или L, 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 перейдёт в цикл, если было сообщение 0:1, даёт сообщение 0, если было 0:2, или даёт сообщение N, если было 0:3 или более, в качестве CONTINUE используется ключевое слово CONT на клавиатуре.

DATA E1, E2, E3, ...

Часть списка данных. Должна располагаться в прграмме.

DEF FNA(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, значений. Используется форма DEFFNA() , если нет аргументов.

DELETE F

Без MICRODRIVE не работает

DIM A(N1, N2, ..., NK)

Уничтожает массив с именем 'A' и 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$' и 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

То же самое , что и DRAW X, Y, Z чертит прямую линию.

DRAW X, Y, Z

Изображает линию от текущей графической позиции в точку с приращениями X, Y по дуге в Z радиан.Выдаёт 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, указанный в утверждении после FOR оператора. Проверяет, если начальное значение больше (если STEP=0) или меньше (если STEP), 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 управляющей переменной.

GO SUB N

Проталкивает строку с оператором GO SUB в стек для использования затем как GO TO N. Выдаёт сообщение 4, если не все подпрограммы завершились с RETURN.

GO TO N

Продолжает выполнение программы со строки 'N', если 'N' опущено, то с первой строки после этой.

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. Выдаёт сообщение K, если 'N' не ле- жит в интервале от 0 до 9.

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 B. выдает сообщение K, если 'N' не 0 или 1.

INPUT ...

Где'...' есть последовательность вводимых символов, разделяемых как в операторе PRINT запятыми, точками с запятой или апострофами. Вводимы- ми символами могут быть:

а) некоторый PRINT-символ, начинающийся не с буквы;

б) имя переменной;

в) строка имен переменных строкового типа.

PRINT- символы в случае а)представляются так же, 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 выдаются мерцающим [?]. для строкового выражения вводной буфер устанавливается для размещения двух таких строк (который при необходимости может быть увеличен). Если первый вводимый символ STOP, то программа 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 сработает. Для останова Вы должны нажать клавишу 'курсор вниз'.

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 осуществляется с усечением справа 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 LIST 0

LLIST N

Подобно 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 'N' байтов, начиная с адреса 'M'.

LOAD F CODE M

Загружает байты начиная с адреса 'M'.

LOAD F CODE

Загружает байты по тому же адресу, с которого они были загружены.

LOAD F SCREEN$

Аналогично LOAD F CODE 16384, 6912 . Очищает файл экрана и загружает его с кассетного магнитофона, смотри главу 20.

LPRINT

Подобно PRINT , но использует принтер.

MERGE F

Подобно LOAD F, но не затирает всю старую 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 и переменные и используемую память, включая и байт адреса в системной переменной RAMBOT, 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 , а значение 'A' меньше, чем значение 'предел', то происходит переход к оператору цикла.

Сообщение 2 выдаются, если не найдена переменная 'A'. Сообщение 1 выдаются, 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 BC адрес 'M', а в регистр a - 'N' 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, то выводимый символ затирает существующий в данной позиции, если N=1, то новый символ соединяется со старым, образуя закрашивающий цвет, при условии, что старый символ имел указание цвета, отличное от старого, или цвет фона, если оба указывают на один и тот же цвет (либо фона, либо закрашивания. Сложение по модулю 2). Смотри 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 время задержки не учитывается и продолжается до первого нажатия клавиши.

PLOT C;M, N

Выводит точку закрашивающего цвета (обработанная OVER и INVERSE) с координатами (ABS(M), ABS(N)), смещает графическую (PLOTPOZITION) позицию, если цветовой символ'C' не специфицирован иначе, то цвет закрашивания в позиции, где расположена эта точка изменяется на текущий сплошной закрашивающий цвет, и другие указания (цвет фона, мерцание, яркость) остаются без изменения. 0<=ABS(M)<=255, 0<=ABS(N)<=175, иначе-сообщение B.

POKE M, N

Записывает значение 'N' в байт памяти по адресу 'M'. 0<= M <= 65535, -255 <= N<= 255, иначе сообщение B.

PRINT ...

Где '...' последовательность PRINT-символов, разделённых запятыми, точками с запятой или апострафами, которые выводятся в экранный файл для отображения на экране телевизора. Точка с запятой сама действия не вызывает, а используется для разграничения символов. Запятая порождает управляющий символ 'запятая' и апостроф порождает символ ENTER.В конце оператора PRINT, если он не заканчивается точкой с запятой, запятой или апострофом, автоматически выводится символ ENTER.PRINT-символом может быть:

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 записывается после буквы 'E' с последующим знаком и двумя цифрами порядка. Иначе X выводится как обычное десятичное число 8-мью значащими цифрами.

в) Строковое выражение. В строке возможны пробелы до и после символов. Управляющие символы вызывают определяемое ими действие. Не отображаемые на экране символы выводятся как '?'.

г) AT M, N Вывод в строку 'M', позицию

д) TAB N Вывод управляющего символа TAB с последующими двумя байтами 'N' (первый байт старший), вызывает TAB-останов.

е) Цветовой символ в форме PAPER, INK, FLASH, BRIGHT, INVERSE или OVER оператора.

RANDOMIZE

То же, что и RANDOMIZE 0

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), выдаёт сообщение B, если 'N' не лежит в интервале 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 с 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 начнёт считывание отсюда.

RETURN

Ссылается на оператор GO SUB в стеке и передает управление на строку после него. Выдаёт сообщение 7, если нет указываемого оператора в стеке. Характерная ошибка, когда операторы GO SUB 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 CODE $ ()

Запись на ленту строкового массива

SAVE F CODE M, N

Записывает на ленту 'N' 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 не совпадающий байт.

Приложение D
Система команд микропроцессора ZILOG Z80

Обозначения:

[R], [R'] регистр, 8 бит
A=111 / B=000 / C=001 / D=010 / E=011 / H=100 / L=101
DD, QQ, PP, RR, регистровая пара, 16 бит
DD : BC=00 / DE=01 / HL=10 / SP=11
QQ : BC=00 / DE=01 / HL=10 / AF=11
PP : 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
(-128..+127 при базовой адресации,-126..+129 при относительной)
[B] позиция бита в байте D0=000 .. D7=111
CCC условие ветвления NZ=000 Z=001
NC=010 C=011
PO=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 00110110	19	11111101 - DIS - - N -	(IY+DIS):=N	.	.	.	.	.	.
LD A,(BC)	7	0001010	A:=(BC)	.	.	.	.	.	.
LD A,(DE)	7	0011010	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	?	0	0
LD A,R	9	11101101 01011111	A:=R	.	?	IF	?	0	0
LD I,A	9	11101101 01000111	I:=A	.	.	.	.	.	.
LD R,A	9	11101101 01001111	R:=A	.	.	.	.	.	.
		Пересылки 16 бит							
LD DD,NN	10	00 DD0001 - N - - N -	DD:=NN	.	.	.	.	.	.
LD IX,NN	14	11011101 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 01DD0011 - N - - N -	(NN):=DD
LD (NN),IX	20	11111101 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(SP)-:=IX 11100101	
PUSH IY	15	11111101 11100101	(SP)-:=IY
POP QQ	10	11 QQ0001	QQ:=(SP)+
POP IX 11100001	14	11011101	IX:=(SP)+
POP IY 11100001	14	11111101	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 BC0 P=1
LDR	21 /16	11101101 10110000	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)<>O then begin	. ? ? ? 1 ? A-(HL)=>Z
CPD	16	11101101 10101001	A-(HL)=? HL:=HL-1 BC:=BC-1 BC = P	. ? ? ? 1 ? A-(HL)=>Z
CPDR	21 /16	11101101 10111001	A-(HL)=? HL:=HL-1 BC:=BC-1 if BC<>0 and A-(HL)<>O 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	A:=A+(IX+DIS)?	? V ? 0 ?

ADD A,(IY+DIS)	19	- DIS - 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 00110100	11111101	(IY+DIS):= (IY+DIS)+1	.	?	V	?	0	?	
DEC T		101	T:=T-1	.	?	V	?	1	?	
DAA	4	00100111	Десятичная коррекция	?	?	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	00DD1 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=A7	?	.	.	.	0	0	
RRCA	4	00001111	CY=A7,A7=A6... ...A0=A7	?	.	.	.	0	0	
RRA	4	00011111	CY=A7,A7=A6...							

			...A0=A7	?	.	.	.	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	смотри RLCA	?	?	P	?	0	0	
RLC (IY+DIS)	23	11111101 11001011 - DIS - 00 000110	смотри RLCA	?	?	P	?	0	0	
RLT		010	смотри RLA	?	?	P	?	0	0	
RRC T		001	смотри RRCA	?	?	P	?	0	0	
RR T		011	смотри 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 1101111	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 в [R]	.	?	X	X	0	1	
BIT [B],(HL)	12	11001011 01 [B]110	Z:=значение бита в (HL)	.	?	X	X	0	1	
BIT [B],(IX+DIS)	20	11011101 11001011 - DIS - 01 [B]110	Z:=значение бита в (IX+DIS)	.	?	X	X	0	1	
BIT [B],(IY+DIS)	20	11111101 11001011 - DIS - 01 [B]110	Z:=значение бита в (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	10	11 CCC010 - N - - N -	PC:=NN,если условие CCC истинно	.	.	.	.	.	.	
JR DIS	12 - DIS-2 -	00011000	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+2 Z=1	.	.	.	.	.	.	
	/7	- DIS-2 -	PC:=PC+DIS Z=0							

JR NC,DIS	12	00100000	PC:=PC+2 Z=0	. . . . .
	/7	- DIS-2 -	PC:=PC+DIS Z=1	
JP (HL)	4	11101001	PC:=HL	. . . . .
JP (IX)	8	11011101	PC:=IX	. . . . .
		11101001		
JP (IY)	8	11111101	PC:=IY	. . . . .
		11101001		
DJNZ DIS	13	00010000	B:=B-1	. . . . .
	/8	- DIS-2 -	B=0:PC:=PC+2	
			B<>0:PC:=PC+DIS	

Команды организации подпрограмм

CALL NN	17	11001001	-(SP):=PC	. . . . .
	- N -	- N -	PC:=NN	
CALL CCC,NN	17	11 CCC100	call NN, если	. . . . .
	/10		CCC истинно	
		- N -		
		- N -		
RET	10	11001101	PC:=(SP)+	. . . . .
RET CCC	11	11 CCC000	ret, если	
	/5		CCC истинно	
RETI	14	11101101	возврат из INT	. . . . .
		01001101		
RETN	14	11101101	возврат из NMI	. . . . .
		01000101		
RST P	11	11 P 111	call P	. . . . .

Команды ввода/вывода

IN A,(N)	11	11011011	A:=(N),	
		- N -	ADR:=A/N	. . . . .
IN [R],(C)	12	11011011	[R]:=(C),	
		01 [R]000	ADR:=B/C	. ? P ? 0 ?
INI	16	11011011	(HL):=(C),	
		10100010	ADR:=B/C	. ? X X 1 X
			HL:=HL+1	B-1=0 Z=1
INIR	21	11011011	B:=B-1	B-1=1 Z=0
	/16	10110010	(HL):=(C),	
			ADR:=B/C	. 1 X X 1 X
			HL:=HL+1	
			BC:=BC-1	
			if BC<>0 then	
IND	16	11101101	begin	
			(HL):=(C),	
		10101010	ADR:=B/C	. ? X X 1 X
			HL:=HL-1	B-1=0 Z=1
INDR	21	11101101	B:=B-1	B-1=1 Z=0
	/16	10111010	(HL):=(C),	
			ADR:=B/C	. 1 X X 1 X
			HL:=HL-1	
			BC:=BC-1	
			if BC<>0 then	
OUT (N),A	11	11010011	begin	
		- N -	(N):=A,	
			ADR:=A/N	. . . . .

OUT (C),[R]	12	11101101 01 [R]001	(C):=[R], ADR:=B/C	. ? P ? 0 ?
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

OTIR	21
	/16
		11101101
		10110011
			(C):=(HL),
			ADR:=B/C
			HL:=HL+1
			BC:=BC-1
			if B<>C0 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

OUDR	21
	/16
		11101101
		10111011
			(C):=(HL),
			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 прерываний 0	. . . . .
IM 1	8	11101101 01010110	режим прерываний 1	. . . . .
IM 2	8	11101101 01011110	режим прерываний 2	. . . . .

Приложение Е

Примеры программ

Это приложение содержит некоторые примеры программ, демонстрирующие возможности ZX SPECTRUM. Первая из этих программ требует ввести дату и даёт день недели, который соответствует этой дате.

```
10 REM CONVERT DATE TO DAY
20 DIM D$(7, 6); REM DAYS OF WEEK
30FOR N=1 TO 7: READ $D(N): NEXT N
40 DIM M(129: REM LENGTHS OF MONTHS
50 FOR N=1 TO 12: READ M(N): NEXT N
100 REM INPUT DATE
110 INPUT "DAY?";DAY
120 INPUT "MONTH?";MONTH
130 INPUT "YEAR? (20TH CENTERURY ONLY)?";YEAR
140 IF YEAR <= 1901 THEN PRINT "20TH CENTURY STARTS AT 1901":
GO TO 100
150 IF YEAR <= 2000 THEN PRINT "20TH CENTURY ENDS AT 2000":
GO TO 100
160 IF MONTH <= 1 THEN GO TO 210
170 IF MONTH <= 12 THEN GO TO 210
180 IF YEAR/4-INT(YEAR/4)=0 THEN LET M(2)=29:
REM LEAP YEAR
190 IF DAY > M(MGNTH) THEN PRINT "THIS MONTH HAS ONLY";
M(MONTH);"DAYS.": GO TO 500
200 IF DAY > 0 THEN GO TO 300
210 PRINT "STUFF AND NONSENSE, GIVE ME A REAL DATE"
220 GO TO 500
300 REM: CONVERT DATE TO NUMBER OF DAYS SINCE START
OF CENTURY
310 LET Y=YEAR-1901
320 LET B=365*Y+INT(Y/4):REM NUMBER OF DAYS TO START OF YEAR
320 LET B=365*Y+INT(Y/4):REM NUMBER OF DAYS TO START OF YEAR
330 FOR N=1 TO MONTH-1: REM ADD ON PREVIOUS MONTHS
340 LET B=B+MM(N):NEXT N
350 LET B=B+DAY
400 REM CONVERT TO DAY OF WEEK
410 LET B=B-7*INT(B/7)+1
420 PRINT DAY;"/";MONTH;"/";YEAR
430 FOR N=6 TO 3 STEP=-1: REM REMOVE TRAILING SPACES
450 NEXT N
460 LET E$=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", "WEDNES"
1020 DATA "THURS", "FRI", "SATUR", "SUN"
1100 REM LENGTHS OF MONTHS
1110 DATA 31, 28, 31, 30, 31, 30
1120 DATA 31, 31, 30, 31, 30, 31
```

Данная программа обрабатывает ярды, футы, дюймы:

```
10 INPUT "YARDS?", YD, "FOOT?", FT, "INCHES?", IN
20 GO SUB 2000 : REM PRINT THO VALUES
50 PRINT " - ";
```

```
70 GO SUB 1000: REM THE ADJUSTMENT
80 GO SUB 2000: REM PRINT THE ADJUSTED VELUES
90 PRINT
100 GO TO 10
1000 REM SUBROUTINE TO ADJUST YD, FT, IN FO THE NORMAL
FORM FOR YARDS, FOOT AND INCHES
1010 LET IN=36*YD+12*FT+IN: REM NOW EVERYTHING IS IN INCHE
1030 LET S=SGN IN: LET IN-ABS IN: REM WE WORK WITH IN
POSITIVE, HOLDING ITS SIGN IN S
1050 LET FT=INT(IN/12): LET IN=(IN-12*(T)*S): REM NOW IN I
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;"IN";:RETURN
```

А вот программа, имитирующая броски монет (игра и-чинг) :

```
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 COIN
40 LET C=C+2+INT(2*RND)
50 NEXT N
60 PRINT " ";
70 FOR N=1 TO 2: REM 1-ST FOR THE THROWN HEXAGRAM, 2-ND FOR
THE 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 "9";: LET C=8
130 PRINT "— "
140 NEXT N
150 PRINT
160 INPUT A$
170 NEXT M: NEW
```

Для запуска программы введите её в компьютер, запустите на выполнение, а затем нажмите клавишу ENTER пять раз для получения двух гексаграмм. Посмотрите 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 результатов.

Многие пользователи найдут тексты всегда более вероятными, нежели они сами могут это предполагать.

Следующая программа - игра "ящеры". Вы задумываете название некоторого животного, а компьютер пытается его отгадать, задавая Вам вопросы, на которые Вы отвечаете Да или Нет.Если компьютер не был ранее знаком с таким животным, то он попросит Вас задавать ему наводящие вопросы, которые помогут ему найти правильные ответ или попросит 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): NEXT N
100 REM START PLAING
110 PRINT "THINK OF AN ANIMAL.", "PRESS ANY KEY TO CONTINUE."
```

```
120 PAUSE 0
130 LET C=1: REM START WITH 1ST QUESTION
140 IF A(C, 1)=0 THEN GO TO 300
150 LET 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 "ANSER ME PROPERLY WHEN I'M", "TALKING TO YOU."
GO TO 300
400 REM QUESSED IT
410 PRINT "I THOUGHT AS MUCH,": GO TO 800
500 REM NEW ANIMAL
510 IF QF > NQ-1 THEN PRINT "I'M SURE YOER ANIMAL IS VERY",
"INTRESTING, BUT I DON'T HAVE",
"ROOM FOR IT JUST NOW": GO TO 800
520 LET Q$(QF)=Q$(C): REM MOVE OLD ANIMAL
530 PRINT "WHAT IS IT, THEN?": INPUT Q$(QF+1)
540 PRINT "TELL ME A QUESTION WHICH DIST-", "INQUISHES
BETWEEN"
550 LET P$=Q$(QF): GO SUB 900: PRINT "AND"
560 LET P$=Q$(QF+1): GO SUB 900: PRINT " "
570 INPUT S$: LET B=LEN S$
580 IF S$(B)="?" THEN LET B=B-1
590 LET Q$(C)=S$(TO B): REM INSERT QUESTION
600 PRINT "WHAT IS THE ANSER FOR"
610 LET P$=Q$(QF+10): GO SUB 900: PRINT "?"
620 GO SUB 1000
630 LET IN=1: LET IO=2: REM ANSWERS FOR NEW AND OLD ANIMALS
640 IF R$="Y" THEN GO TO 700
650 IF R$="y" THEN GO TO 700
660 LET IN2; LET IO=1
670 IF R$="N" THEN GO TO 700
680 IF R$="n" THEN GO TO 700
690 PRINT "THAT'S NO GOOD.";GO TO 600
700 REM UPDATE ANSEWERS
710 LET A(C, IN)=QF+1: LET A(C, IO)=QF
720 LET QF=QF+2: PEM NEXT FREE ANIMAL SPACE
730 PRINT "THAT FOOLED ME."
800 REM AGAIN ?
810 PRINT "DO YOU WANT ANOTHER GO?": GO SUB 1000
820 IF R$="Y" THEN GO TO 100
830 IF R$="y" THEN GO TO 100
840 STOP
900 REM PRINT WITHOUT TRALING SPACES
905 PRINT " ";
910 FOR N=50 TO 1 STEP -1
920 IF P$(N)" " THEN GO TO 940
930 NEXT N
```

```
940 PRINT P$(TO N): RETURN
1000 REM GET REPLAY
1010 INPUT R$: IF R$="" THEN RETURN
1020 LET R$=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 EAT ANTS", 6, 7
2040 DATA "A WHALE", "A BLANCMANGE", "A PANAGOLIN", "AN ANT"
```

Приложение F

Использование кириллицы

MOVE, ERASE

В оригинальном исполнении ROM микрокомпьютера ZX SPECTRUM не содержит знакогенератора русских букв.Однако в предлагаемом описании речь идет о доработке стандартной прошивки ПЗУ 573РФ4 в части, касающейся использования букв русского алфавита. Для перехода на "русский регистр" и обратно используются не задействованные в стандартном SPECTRUM функции MOVE и ERASE.Кроме того, использование кириллицы в SPECTRUM подчиняется обычным для такого применения ограничениям: русские буквы могут появляться в программе только в строковых константах.Это значит, что букв русского алфавита вне областей ограниченных строковыми двойными кавычками, интерпретируется Бейсиком как синтаксическая ошибка.

Применение для переключения языкового регистра не используемых но стандартных функций MOVE и ERASE делает программно совместимыми стандартный и модифицированный ROM.Программы, написанные с использованием русского алфавита (только в области констант) будут нормально выполняться и на стандартном SPECTRUM (без изображения кириллицы на экране).

Для перехода к русскому алфавиту необходимо ввести функцию MOVE (в режиме E курсора нажать SYMBOL SHIFT и 6).После набора текста на русском языке необходимо переключить регистр обратно - ввести ERASE (в режиме E курсора нажать SYMBOL SHIFT и 7).

Таблица соответствия
букв русского алфавита клавишам SPECTRUM

A	Нажатием клавиши	А	T	Нажатием клавиши	Т
B	Нажатием клавиши	Б	U	Нажатием клавиши	У
W	Нажатием клавиши	В	F	Нажатием клавиши	Ф
G	Нажатием клавиши	Г	H	Нажатием клавиши	Х
D	Нажатием клавиши	Д	C	Нажатием клавиши	Ц
E	Нажатием клавиши	Е	A	E - SYMBOL SHIFT	Ч
V	Нажатием клавиши	Ж	H	SYMBOL SHIFT	ч
Z	Нажатием клавиши	З	F	E - SYMBOL SHIFT	Ш
I	Нажатием клавиши	И	Y	E - SYMBOL SHIFT	ш
J	Нажатием клавиши	Й	G	E - SYMBOL SHIFT	Щ
K	Нажатием клавиши	К	U	Нажатием клавиши	щ
L	Нажатием клавиши	Л	Y	Нажатием клавиши	Ы
M	Нажатием клавиши	М	X	E - SYMBOL SHIFT	Ь
N	Нажатием клавиши	Н	S	E - SYMBOL SHIFT	Э
O	Нажатием клавиши	О	D	E - SYMBOL SHIFT	э
P	Нажатием клавиши	П	X	E - SYMBOL SHIFT	Ю
R	Нажатием клавиши	Р	2	E - SYMBOL SHIFT	ю
S	Нажатием клавиши	С	Q	Нажатием клавиши	я

Переход от прописных букв к строчным и наоборот осуществляется обычным образом (CAPS SHIFT и 2) для букв, вводимых простым нажатием клавиши.Для дополнительных букв, которым

не хватает клавиш SPECTRUMа используются разные клавиши для строчных и прописных букв, а также режим E курсора и клавиша SYMBOL SHIFT (см. таблицу).

Приложение G.
Команды прошивки TURBO-90

Прошивка TURBO-90:
-экономит время при работе с магнитофоном
-не требует доработок схемы SINCLAIR
-подсчитывает свободный объем памяти
-имеет встроенный монитор на уровне машинных кодов
Для перехода в требуемый режим используйте:
NEW [ENTER] - сбрасывает программы (сброс это не делает)
- вызов на редактирование строки с номером N
* [ENTER] - выход в монитор
, [ENTER] - подсчитывает объем свободной памяти ОЗУ в десятичном или шестнадцатиричном виде в зависимости от выбора в мониторе
- [ENTER] - переключает на стандартную скорость считывания/записи
+ [ENTER] - переключает на TURBO скорость считывания/записи
Для работы в мониторе используйте:
[1] - взаимный переход десятичной и шестнадцатиричной системы счисления
[2] - взаимный переход символьного и цифрового режима чтения
[6] - сдвиг адреса по памяти назад
[7] - сдвиг адреса по памяти вперед
[5] - сдвиг адреса на 64 байта назад
[8] - сдвиг адреса на 64 байта вперед
[SPACE] - выдает подсказку "]"
nnnn - переход на начальный адрес nnnn
[4] - запись в ячейку памяти байта
[0] - переход по регистрам вперед
[9] - переход по регистрам назад
[3] - занесение в регистр числа
[СБРОС] - выход из монитора с сохранением регистров и памяти

СОДЕРЖАНИЕ