ПРОГРАММИРОВАНИЕ В МАШИННЫХ КОДАХ

ZX-SPECTRUM

Программирование в машинных кодах.

Москва, 1994

Подписано к печати 12.04.94
Формат 60×84/16 Печать офсетная
Печ. л. 10,0 Тираж 3000 Зак. 305

Отпечатано на фабрике офсетной печати, г. Обнинск

Предисловие.

Почти невозможно поверить, что всего лишь за 2.5 года фирма SINCLAIR RESEARCH OF CAMBRIDGE изготовила и продала около 500 тысяч микрокомпьютеров.

Весной 1980 года был разработан микрокомпьютер ZX-80. разработка этой машины явилась большим успехом, поскольку это был первый сравнительно дешевый микрокомпьютер для любителей. Однако, Клив Синклер и его лаборатория изготовила машину ZX-81 всего за 1 год. Эта модель была более совершенна по сравнению с ZX-80 и явилась развитием этого компьютера с низким разрешением и черно-белым экраном.

Но теперь мы имеем ZX-SPECTRUM. Эта машина была разработана по принципу ZX-80, ZX-81. Сделав это фирма разработала микрокомпьютер с высоким разрешением и цветным экраном. Однако с сожалением надо отметить, что ZX-80, ZX-81 были вытеснены, хотя обе являются прекрасными машинами. Они просты в работе, что приятно для программиста, но это не означает, что SPECTRUM это трудная машина для использования, но для того, чтобы от нее получить все, потребуется написание более совершенных программ. Эта книга написана для более глубокого понимания работы с компьютером ZX-SPECTRUM в целом.

1. Принцип работы микрокомпьютерной системы.

Всегда очень любопытно погрузиться в чтение, открывая книгу то здесь, то там; но компьютер является самой логической машиной и каждый, кто попытается совершенствовать свои познания должен начать с самого начала.

1.1. Описание системы.

Сам по себе SPECTRUM представляет собой пластмассовый корпус. На верхней части расположены 40 клавиш, которые образуют клавиатуру. Сзади расположен выходной разъем для подключения антенного входа телевизора, входной разъем, который соединяется с выходом магнитофона, разъем для подключения УВВ, принтера, микроприводов и так далее; разъем для подключения блока питания.

На основной плате находиться микропроцессор Z-80 и другие элементы. Плата находиться в одном корпусе, что и клавиатура. Клавиатура и основная плата соединены между собой ленточным кабелем.

Система разработана таким образом, что она может отображать информацию на экране TV приемника или выводить ее на принтер или другое периферийное устройство.

1.2. Логика работы.

Рассмотрим связь между различными элементами микрокомпьютерной системы.

Z-80 микропроцессор (МП) может обращаться к отдельным 65536 байтам памяти (64к). Ограничением на непосредственную адресацию МП Z-80 является 64к. В версии 16к SPECTRUM можно адресоваться только по адресам от 0 до 32767, а в версии 48к SPECTRUM - доступны адреса 0-65535. Адрес задается в виде 16-битного числа. 0 представляется как 0000 0000 0000 0000, а 65535 как 1111 1111 1111 1111. адреса, выданные Z-80 поступают на адресную шину, состоящую из 16 линий, находящихся в высоком или низком напряжении (1 или 0) . Адрес можно описать в двухъ байтах по 8 бит. В отличие от адресной шины, шина данных имеет 8 бит, поэтому любые данные могут представляться как 0-255 в десятичном виде или 0000 0000-1111 1111 в двоичном.

Как часть логической концепции ZX-SPECTRUM, также важно рассмотреть нормальный режим работы системы и обсудить использование памяти компьютера. SINCLAIR RESEARCH выпускает SPECTRUM с 16к мониторной программой и обеспечивает пользователя операционной системой (ОС) и интерпретатором БЕЙСИК (ИБ). По существу возможно вместо программы монитор использовать МП Z-80 с программой в машинных кодах.

При обычном использовании, ОС не требует со стороны пользователя каких либо сложных действий. Поэтому при работе SPECTRUM уже работает ИБ, как часть мониторной программы. Пользователь может сразу вводить или выполнять БЕЙСИК программу. Таким образом ИБ является подпрограммой ОС, а БЕЙСИК программа подпрограммой ИБ.

Заметьте, что Z-80 сам по себе не может выполнять БЕЙСИК программы, а только программу составленную в машинных кодах.

Карта памяти стандартного 16к SPECTRUM представлена ниже и далее кратко описана.

32767	7FFF	графическая область	P-RAMT
32600	7F58	пользователя	UDG
32599	7F57	стек GOSUB	RAMTOP
		машинный стек	указатель стека
		резервная память	STKEND
		стек калькулятора	STKBOT
		рабочая область	WORKS
		область редактора	E-LINE
		область переменных	VARS

		область БЕЙСИК программ	PROG
		область информационных каналов	CHANS
23734	5CB6	карты микроприводов	
23552	5C00	системные переменные	
23296	5B00	буфер принтера	
22528	5800	область атрибутов	
16384	4000	область экрана	
0	0000	ROM область	

ROM область.

16к ROM содержащие ОС, ИБ и генератор знаков (ГЗ) располагаются в адресах 0-16383 (16-ное 0-3FFF) .

Карта памяти экранной области.

6к памяти с 16384 до 22527 (4000-57FF) обеспечивают высокое разрешение на TV. Важно, что расположение экранной области зафиксировано в аппаратуре SPECTRUM и не может меняться программно. Имеется однозначное соответствие всех битов области и точек экрана TV, и следующие вычисления показывают, что число битов в 6к памяти равно числу точек экрана.

В 6к - 1024*8*6=49152 битов.

Количество символов в 24 строках экрана (каждый символ представлен в виде матрицы 8*8 точек) равно 32*24=768, следовательно всего будет 768*64=49152 точки. Экран TV разбит на 3 части. Строкам с номерами 0-7 соответствуют байты памяти 16384-18431 (4000-477F), строкам 8-15 - байты 18432-20479 (4800-4FFF), строкам 16-23 - байты 20480-22527 (5000-57FF). Из этого следует, что каждым 8-ми строкам экрана соответствуют 2к экранной области.

Рассмотрим эти 2к памяти для верхней части экрана (строки 1-8). Первые 32 байта экранной области соответствуют первой линии первой строки. Следующие 32 байта соответствуют первой линии второй строки и так далее до восьмой строки. После этого очередные 32 байта соответствуют второй линии первой строки. Следующие 32 байта соответствуют второй линии второй строки и так далее. После первых восьми строк аналогичным образом устанавливается соответствие для следующих восьми строк и так далее.

Следующий пример все наглядно показывает:

```
10 FOR A=16384 TO 22527 STEP 1
20 POKE A,255: PAUSE 1
30 NEXT A
40 PAUSE 0
```

Область атрибутов.

Экран имеет 768 символов, каждый из которых имеет 8 цветов "бумаги", 8 цветов "чернил", признак мигания, признаки повышенной или пониженной яркости. Каждому символу экрана соответствует один байт из области атрибутов. Область расположена по адресу 22528-23295 (5800-5AFF).

Зависимость между символами и байтами атрибутов несложная, так как байты просматриваются подряд для каждой строки экрана слева направо.

Байт из области атрибутов имеет следующий формат:

7	6	5	4	3	2	1	0
мерцание	яркость	цвет бумаги (3 бита)			цвет чернил (3 бита)		

следующий пример демонстрирует вышесказанное:

```
10 FOR A=16384 TO 22527 STEP 1
20 POKE A,15
30 NEXT A
40 FOR A=22528 TO 23295 STEP 1
50 POKE A,161
60 NEXT A
70 PAUSE 0
```

Буфер принтера.

Буфер принтера расположен по адресу 23296-23551.

Системные переменные.

182 байта по адресу 23552-23733 (5C00-5CB5) отводятся под системные переменные, которые используются для организации работы компьютера.

Карты микроприводов.

Эта область памяти начинается с 23734 (5CB6) и в стандартном SPECTRUM представлена теоретически, то есть область не используется до тех пор, пока микропривод не будет подключен.

Область информации о каналах.

Специальная область памяти, расположенная с адреса, указанного в переменной CHANS. CHANS находится по адресу 23631-23632 (5C4F-5C50). Область переменной длины. Концом области служит маркер со значением 128 (80 16-ричное).

В системе SPECTRUM обмен данными с периферийными устройствами (дисплей, клавиатура, принтер и так далее) осуществляется посредством системы ввода-вывода, которая включает в себя каналы и потоки.

Канал - совокупность программных и аппаратных средств, обеспечивающих обмен данными с периферийным устройством. В стандартном SPECTRUM без подсоединенных микроприводов имеются 4 основных канала:

- ⇨ канал K - обеспечивает ввод от клавиатуры и вывод на нижнюю часть экрана;
- ⇨ канал S - только вывод на верхнюю часть экрана;
- ⇨ канал R - только вывод в рабочую область памяти, которая по требованию может быть увеличена;
- ⇨ канал P - только вывод на принтер.

Кроме перечисленных каналов, могут быть дополнительные каналы ("M" - канал микродрайва, "N" - канал локальной сети, "T" и "B" дополнительные каналы для принтера) и программист может создать канал самостоятельно.

Информация о каналах находится в области, адрес которой содержится в системной переменной CHANS.

Стандартные каналы описываются блоками по пять байт, структура каждого блока следующая:

байты 0,1 - адрес процедуры вывода (PRINT #);
байты 2,3 - адрес процедуры ввода (INPUT #);
байт 4 - имя канала ("K","S","P").

Канал создаваемый пользователем (программистом) описывается блоком произвольной длины, но не менее чем 11 байт:

байты 0,1 - адрес процедуры вывода (PRINT #);
байты 2,3 - адрес процедуры ввода (INPUT #);
байты 4 - имя канала;
байты 5,6 - признак пользовательского канала (число "1234");
байты 7,8 - адрес закрывающей процедуры (CLOSE #);
байты 9,10 - длина данного блока (не менее 0B 16-ричное);
байты 11... - любая дополнительная информация.

В стандартном SPECTRUM, с учетом 4 каналов и маркера конца, область занимает память 23734 - 23754 (21 байт).

Обращение к каналу производится через поток. К одному потоку может быть подключен только один канал, хотя к одному каналу может быть подключено несколько потоков. В БЕЙСИКе это осуществляется с помощью команд "OPEN #" и "CLOSE #".

Например:

OPEN #6, "K" - подключить к потоку 6 канал "K";

CLOSE #2 - отключить от потока 2 канал, который был к нему подключен.

В системе SPECTRUM к потокам 0, 1, 2, 3 подключены стандартные каналы:

- ⇨ поток 0 - к каналу "K";
- ⇨ поток 1 - к каналу "K";
- ⇨ поток 2 - к каналу "S";
- ⇨ поток 3 - к каналу "P".

При вводе или выводе данных с помощью команд БЕЙСИКа INPUT и PRINT обычно поток не задается (умалчивается), это соответствует (по умолчанию) тому, что команда PRINT работает с потоком 2, а команда INPUT - с потоком 0, но номер потока можно задать следующим образам: PRINT #3;"PROG" - это приведет к печати слова "PROG" на принтере, то есть аналогично команде LPRINT "PROG".

Информация о подключении потоков к каналам находится в системной переменной STRMS. Всего в системе 16 потоков с номерами от 0 до 15.

Область Бейсик-программ (БП).

Область содержит последовательные строки программ. Ее размер определяется количеством строк.

Начало программы задается в переменной PROG (адрес 23635- 23636 (5C53-5C54)). Заметьте, что в стандартном SPECTRUM переменная PROG указывает на адреса 23755 (5CCB) и так до тех пор, пока не будет подключен микропривод или использован дополнительный канал.

Формат строк БП:

Первые 2 байта любой строки содержат номер строки и причем в начале храниться старший байт, а затем - младший. В 3 и 4 байтах храниться длина строки, причем сначала младший байт, а потом старший. Байты с 5-го до символа ENTER (0D 16-ричное) являются информационной частью строки в кодах SINCLAIR и ASCII. внутри строки операторы разделены символом 58(3A). Заметьте, что десятичные числа накапливаются в строках БП, т.к. представляют из себя последовательности ASCII символов, за которыми идет NUMBER (код 14 (0E)).

Следующая демонстрационная программа все подробно показывает:

```
10 FOR A=23755 TO 23817: PRINT A; TAB 9; PEEK A; TAB 15; CHR$
PEEK A: NEXT A
```

Область переменных.

Начало области, описывающей все переменные, задается в переменной VARS - адреса 23627-23628 (5C4B-5C4C). Начало этой области остается постоянным во время работы БП, однако в связи с появлением новых переменных ее длина будет меняться.

Граница области задается маркером конца (код 128).

Следующая программа содержит только переменные оператора FOR NEXT:

```
10 FOR A=23804 TO 23823: PRINT A; TAB 9; PEEK A: NEXT A
```

Область редактора.

В области отражается строка, которую вводят или редактируют. Начало области задается в переменной E-LINE - байты 23641-23642 (5C59-5C5A).

Так как область редактирования представляет собой динамическую область, то есть она меняется при использовании, то не целесообразно давать пример на БЕЙСИКе.

Рабочая область.

Эта область используется для выполнения большого числа различных задач, например ввод данных, соединение в цепочку строк и так далее. Начальный адрес области задается величиной, содержащейся в системной переменной WORKS , которая сама находится по адресу 23649 и 23650 (5C61,5C62). Когда же требуется дополнительное место в рабочей области, то эта область памяти расширяется. После использования рабочая область освобождается. Это значит, что она сводиться к 0, чтобы избежать большего количества байтов, чем это необходимо.

Еще раз отметим, что поскольку эта область является динамической, невозможно привести простой пример на БЕЙСИКе.

Стек калькулятора.

Это очень важная область памяти. Она начинается с байта, определенного системной переменной STKBOT , которая сама находится по адресу 23651-23652. область распространяется до байта, заданного в системной переменной STKEND (23653-23654).

Стек калькулятора используется подпрограммой ос (находящейся в ROM-области), называемой "калькулятор". Эта подпрограмма выполняет различные арифметические операции. Использование стека калькулятора и самого калькулятора будет описано далее.

Резервная память.

Область памяти между стеком калькулятора и машинным стеком представляет собой количество памяти, которым располагает пользователь.

Машинный стек.

МП Z-80 должен иметь рабочую область для своего использования и это называется машинный стек. Указатель стека Z-80 всегда показывает на последний байт, который должен заполняться.

Машинный стек будет подробно рассматриваться дальше.

GOSUB - стек.

Стек расширяется в памяти вниз и каждая команда GOSUB требует памяти в GOSUB-стеке. Самый старший байт содержит номер операнда в пределах строки Бейсик-программы, к которой должен быть осуществлен возврат. Второй байт содержит младшую часть номера строки и третий байт содержит ее старшую часть.

Демонстрация следующей программы показывает стек GOSUB, который используется для сохранения данных возврата при организации вызова подпрограмм.

Демонстрационная программа GOSUB-стека для SPECTRUM 48к:

```
10 GOSUB
20: STOP 20 GOSUB
30: RETURN 30 GOSUB
40: RETURN 40 FOR A=65368 TO 65357 STEP -1: PRINT A, PEEK A:
NEXT A: RETURN
```

Два байта выше GOSUB-стека всегда содержат величины 0 и 62 (00 и 3E) и они представляют собой неправильный номер строки.

Программа БЕЙСИК при выполнении лишней команды RETURN перейдет к неправильному номеру строки и выдаст сообщение "RETURN без GOSUB".

Примечание: фирма SINCLAIR не совсем уверена в написании GOSUB.

Системная переменная RAMTOP , которая занимает байты с адресом 23730 и 23731 (5CB2,5CB3) содержит адрес ячейки, в которой храниться код 62. Эта ячейка рассматривается как последняя ячейка системной области БЕЙСИК.

Графическая область, определенная пользователем.

До тех пор, пока системная область БЕЙСИКа не изменяется вследствие использования команды CLEAR , последние 168 ячеек памяти содержат представления 21 графического символа, определенного пользователем.

Адрес самого старшего байта памяти всегда находится в системной переменной P-RAMT , расположенной по адресу 23732,23733 (5CB4,5CB5).

В стандартном 16к SPECTRUM величина, содержащаяся в P-RAMT должна быть равна 32767 , поскольку это показывает, что все 16к памяти находятся в рабочем состоянии.

Результатом выполнения следующей строки:

```
PRINT PEEK 23732+256*PEEK 23733
```

является величина 32767 (в 48к SPECTRUM - 65535).

2. Функции и команды БЕЙСИК.

2.1. Введение

Ожидается, что читатели этой книги уже будут иметь некоторые знания о БЕЙСИКе SPECTRUM .

Эта глава обсуждает команды БЕЙСИКа и функции, и постарается прояснить моменты, не освещенные детально в других пособиях по БЕЙСИКу.

Интерпретатор БЕЙСИК для SPECTRUM распознает 50 различных команд и 33 функции. Каждая из них здесь будет обсуждена подробно. Они будут рассмотрены в алфавитном порядке, так чтобы ссылки на них были проще. Управляющие символы обсуждены в пункте 2.4.

2.2. Команды БЕЙСИКа.

BEEP X,Y.

По этой команде подается сигнал в динамик. X-длительность в сек., Y-высота звукового тона, от среднего "до", в десятичном диапазоне чисел -60 +69,8. X или Y, или оба могут быть выражениями.

Интересно отметить, что BEEP не может быть прерван, поскольку программа, содержащая команду BEEP не прерывается по клавише BREAK. BREAK возможен только в конце оператора, содержащего команду BEEP.

BORDER M.

Имеется 8 возможных цветов, которые могут быть переданы на граничную область (бордюр TV-экрана. Используются целые числа от 0 до 7, однако допускается значения от 0.5 до 7.5 , как результат выражения, поскольку M округляется до целого.

По команде BORDER выходной сигнал поступает на порт 254 и это может быть выражено оператором:

OUT 254,M

(где M=2 дает красный бордюр).

Но OUT и BORDER различны. Цвет получаемый бордюром по команде OUT - "временный", в то время как BORDER дает постоянный цвет, хранящийся в системной переменной BORDER (5C48 16-ное). BORDER 2 и ENTER делают бордюр красным. OUT 254, 1 и ENTER - бордюр будет голубой временно, а потом опять вернется красный цвет.

Отметим что BORDER хранит цвет бумаги нижней части TV экрана.

BRIGHT M.

Это первая из команд элемента цвета, все эти команды могут быть использованы, либо только как команды в операторе БЕЙСИК (в этом случае действие постоянно), либо включены в оператор печати и тогда их действие временно. M - может быть выражением, но результат должен быть целым числом 0,1 или 8. при M=0 экран имеет нормальную яркость, при M=1 любая последующая надпись будет сделана на подсвеченной бумаге. Использование BRIGHT 1 и CLS - простой путь сделать всю область экрана яркой.

При M=8 будет использовано постоянное значение из соответствующих системных переменных.

Если BRIGHT используется самостоятельно, то ее действие постоянно, а если в другом операторе, то ее действие временно.

Например :

```
10 BRIGHT 0
20 PRINT "BRIGHT - NORMAL"
30 PRINT BRIGHT 1; "GROG"
40 PRINT "BRIGHT - NORMAL"
```

CIRCLE X,Y,Z.

Эта команда рисует окружность с радиусом Z и с координатами центра X, Y. Z - берется как абсолютное целое значение а X, Y используются как плавающие величины. Самая большая окружность при радиусе 88 единиц получается оператором CIRCLE 127.5, 87.5, 88, а окружность 0-вым радиусом представляет точку. Любая команда цвета может быть включена в оператор CIRCLE и ее влияние всегда будет временным. По некоторым параметрам команда медленна и не точна, но очень полезна.

CLEAR, CLEAR N.

В системе SPECTRUM имеется такое большое количество памяти RAM, что использование команды CLEAR само по себе вряд ли будет полезным. Однако расширение команды возможностью перемещать указатель RAMTOP делает ЕЕ очень сильной . RAMTOP указывает на верхнюю границу системы БЕЙСИК и содержимое памяти ниже RAMTOP может изменяться, в то время как любое содержимое выше RAMTOP остается неизменным, даже после выполнения команды NEW. Нижняя граница для N возможна не менее чем 23821. Иначе SPECTRUM выдает звуковой сигнал. Это говорит о том что разрешенного RAM не хватает на задачу. Верхней границей N является для системы 16к 32767, а для систем 48к 65535. Использование CLEAR N с соответствующим значением вызывает перемещение стека GOSUB в область, отведенную для графики пользователя. Поучительно выполнить следующее упражнение:

Ввести CLEAR 32767 или (65535) изменить курсор на G и вывести литералы с L по U. нажать на клавишу SPACE и несколько секунд наблюдать изменение графики пользователя, поскольку каждое нажатие клавиши приводит к использованию машинного стека.

CLOSE #N.

Используется с микроприводом. Предназначена для закрытия потока с номером N (N может принимать значения от 0 до 15).

CLS.

Это очень простая команда, которая меньше чем за 1/10 секунды выполняет много работы. Команда CLS очищает дисплейный файл. Она обнуляет адреса с 16384 по 22527 (40000- 5800) и сбрасывает байты атрибутов по адресам 22528-23295 (5800-5AFF). Байты атрибутов не обнуляются, а копируются из системной переменной ATTR-P, которая содержит постоянные атрибуты. По команде POKE значение переменной ATTR-P, которая помещается по адресу 23693 (5C8D) может быть изменено. В переменной ATTR-P бит 7 отвечает за мигание, бит 6 за яркость, биты 3-5 за цвет бумаги, биты 0-2 за цвет чернил.

CONTINUE.

В большинстве требуемых случаев эта команда работает хорошо, но когда пользователь выполнит несколько команд непосредственно, SPECTRUM зациклится и сможет работать только по клавише BREAK.

Существует 2 основных аспекта команды. 1-й позволяет пользователю использовать в программе БЕЙСИК оператор STOP и тогда не требуется использовать клавишу BREAK. пользователь в этом случае может проверить переменные, установить переменные и изменить программу БЕЙСИК. Использование операторов STOP и CONTINUE очень облегчает отладку программ.

Второй аспект позволяет повторять выполнение после исправления ошибок. Например если программа остановилась по ошибке "VARIABLE NOT FOUND", то переменная может быть определена в непосредственном режиме и оператор заново запущен по команде CONTINUE.

6 системных переменных NEWPPS, NSPPS, PPS, SUBPPS, OLDPPS и OSPPS включаются некоторым образом в выполнение команды CONTINUE.

COPY.

Это очень простая команда. Верхние 22 строки TV экрана посылаются на принтер. Команда COPY одна из немногих команд, которые останавливают часы реального времени на время выполнения команды. Это можно видеть по системной переменной FRAMES до и после использования команды.

DATA E,E...

Эта команда, которая может использоваться только в программных строках, определяет список данных.

Хотя это и упоминается в различных описаниях, но становится очень ясно, что рассмотрение элементов в операторе DATA как выражений является особенностью, делающей эту команду очень полезной. Более детально см. READ, RESTORE.

DEF FN A(A....Z)=E, DEF FN A$(A....Z)=E.

Команда DEF FN очень мощная. Пользователь может определять до 52-х функций - 26 числовых и 26 строковых. Имена используемые для функции, должны быть одинарными литерами (+$ для строк) и они могут быть именами, используемыми где-нибудь как простые переменные. Существует небольшое ограничение на имена аргументов, так как они тоже должны быть одинарными литерами, (+$ для строк). Поэтому может быть не более 52-х аргументов для одной функции. Выражение определяемое функцией, может быть любым дающим числовой или строковый результат.

DIM A(E1.....Ek), DIM A$(E1.....Ek).

Команда DIM удаляет существующий массив с тем же именем и затем устанавливает новый. Числовые массивы обнуляются, а в строковые заносится литера пробел. В системе SPECTRUM вся индексация начинается с 1. Достижение же индексом 0-го значения является ошибкой. Необходимо знать, что в символьных массивах размер строки фиксирован и он определяется значением последнего элемента размерности массива. Например : DIM A$(5,10) означает создание строкового массива A$ из пяти строк, длина каждой из которых равна десяти символам. Возможно обращение как к целой строке (LET A$(5)="PRINT", пятой строке присваивается указанное значение), так и к отдельному элементу строки (LET A$(4,5)="R").

DRAW X,Y; DRAW X,Y,Z.

Эта команда рисует линию от текущей позиции (но не включая ее) на расстояние X по горизонтали и Y по вертикали. Если аргумент Z определен то рисуется дуга вместо прямой линии. Z должно быть определено в радианах, при Z=PI (3.14) получается полуокружность. При положительных значениях дуги рисуются справа от того места где бы была прямая линия, а при отрицательных слева, любая из команд цвета может быть включена в оператор DRAW, где ее влияние будет временным.

ERASE.

Используется с микроприводом.

FLASH M.

Это вторая команда управления цветом. Когда она используется одна в программе на БЕЙСИКе, ее влияние постоянно. Но когда используется включением в оператор печати, ее влияние временно.

Как и BRIGHT может принимать значения 0, 1, 8. при M=0 область литер не мигает, при M=1 они мигают, при M=8 будет использовано постоянное значение.

FOR A=X TO Y STEP Z.

Команда FOR наиболее интересная команда и обычно она плохо понимается, хотя и широко используется.

Задачи, стоящие перед интерпретатором, когда он имеет дело с командой FOR таковы:

1. Удалить любые переменные с тем же именем (A) и любые управляющие переменные с тем же именем (A).
2. Добавить к существующим переменным новую управляющую переменную (A). Эта переменная требует 19 байтов памяти. Первый байт содержит имя переменной. Следующие 5 содержат начальное значение цикла (X). Следующие 5 байтов конечное значение цикла (Y). Следующие 5 байтов значение приращения (Z). Последние 3 байта содержат информацию о цикле: первые 2 указывают номер строки в которой записан оператор FOR, а последний - номер оператора в строке, увеличенный на 1. таким образом после окончания цикла будет выполняться следующий оператор, вне зависимости находится он в той же строке или нет. Если начальное значение, предельное значение и шаг целые числа от -65535 до +65535 то операция выполняется на 20% быстрее. Следующая программа демонстрирует содержимое управляющей переменной команды FOR . Заметьте что оператор FOR в строке 10 формирует переменную под именем A, которая в дальнейшем в программе не используется.

```
10 FOR A=1.6 TO 2.1 STEP 0.1
20 LET V=PEEK 23627+256*PEEK 23628-1
30 FOR B=1 TO 19
40 PRINT B, PEEK (V+B)
50 NEXT B
```

добавьте 25 LET V=V+25 и увидите контрольную переменную B.

3. Третье действие предпринимаемое оператором, является очень специфичным. В SPECTRUM не существует ошибок, вызванных установлением шага в неправильном направлении. Это значит, что строка, подобная FOR A=2 TO 0 STEP 1 допустима. Однако это действие приводит к полному игнорированию FOR NEXT-цикла. Это может быть показано изменением продемонстрированной выше программы следующим образом:

```
10 FOR A=1.6 TO 2.1 STEP -0.1
50 NEXT A
```

и программа будет успешно работать, но не будет никакой печати. Эта возможность перепрыгивать через целые циклы, может быть и преимуществом (это позволяет распространить ограничения на 0) и недостатком (ожидаемый результат не появляется вообще).

FORMAT F.

Используется с микроприводом.

GO SUB N.

Эта команда приводит к выполнению подпрограммы, которая начинается со строки с номером N. Если строки с номером N нет, то будет выполнятся подпрограмма начиная со строки следующей в порядке возрастания за N. после выполнения подпрограммы управление возвращается оператору следующему после оператора GO SUB. Следующая программа показывает, что в SPECTRUM 48к GO SUB стек начинается с адресов 65363-65365 :

```
10 GO SUB 20: STOP
20 FOR A=1 TO 3: PRINT A, PEEK (65362+a): NEXT A
```

важным свойством БЕЙСИКа является то, что он позволяет командам GO SUB и GO TO делать управляемые переходы. Это определяется тем, что в качестве параметра в командах GO SUB и GO TO может использоваться переменная, например :

```
10 LET A=30
20 GO SUB A: STOP
30 PRINT A
40 RETURN
```

GO TO N.

Это очень простая команда перехода к строке N. следующей строкой является строка N или первая строка после N. команда GO TO помещает адрес перехода (номер строки) в системные переменные NEWPPS и NSPPS, расположенные в 23618-23620 (5C42-5C44). Переход к операторам не являющимися первыми в строке, в обычной практике не допустим.

IF X THEN S.

В системе SPECTRUM значение 0 соответствует логическому значению "ложь", а любое значение отличное от 0 логическому значению "истина". Команда выполняет действие S и другие операторы в этой строке, если значение X истинно. Эта команда работает хорошо, но следующая программа написана, чтобы показать, используя повторное деление что происходит, когда значения никогда не становятся ложью .

```
10 LET A=1
20 IF NOT A THEN PRINT A: STOP
30 PRINT A
40 LET A=A/2
50 GO TO 20
```

значение в команде IF никогда не станет "истиной" (значение а будет распечатываться бесконечно). Если строку 40 поменять на LET A=A*0.5 то значение "ложь" будет достигнуто после того как распечатается 6 экранов.

INK N.

Это 3-я из команд элементов цвета. Когда она используется самостоятельно ее действие постоянно, а когда добавляется в оператор печати - временно. Значение N может быть от 0 до 9. 8 основных цветов в SPECTRUM закодированы от 0 до 7 и это ясно видно на клавиатуре. Использование INK 8 при печати дает цвет, хранившийся в атрибутах (постоянное значение). Использование INK 9 однако немного более сложно : чернила становятся только черными или белыми в зависимости от цвета бумаги. Когда цвет бумаги черный, красный, синий, фиолетовый - цвет чернил белый, когда зеленый, голубой, желтый или белый - цвет чернил черный. Действие постоянной команды INK состоит : для N=0-7 устанавливаются биты 0-2 переменной ATTR-P; для N=8 биты 0-2 переменной MASK-P, для N=9 бит 5 P-FLAG . Если команда INK имеет временное действие то устанавливаются соответствующие биты временных системных переменных.

INPUT...

Это очень мощная команда, она позволяет совместно с элементами печати добиваться желаемых результатов.

Пример:

```
10 INPUT "NAME PLEASE";A$;CHR$ 13;"AGE PLEASE";B$
20 PRINT AT 5,0;"NAME";TAB 7;A$;CHR$ 13;"AGE";TAB 7;B$
```

использование INPUT позволяет помещать в строковую переменную всю введенную строку.

INVERSE N.

Команда управления цветом. Эффект может быть временным или постоянным. При INVERSE 1 цвета чернил и бумаги меняются местами а при INVERSE 0 - нормальное значение цвета чернил и бумаги. При постоянном режиме по команде устанавливается бит 5 P-FLAG, а при временном бит 4.

LET V=E.

Значение E присваивается переменной V. ИБ ищет среди существующих переменных переменную V, преобразует ее, если это необходимо, или создает заново, если переменная V не найдена. Далее ИБ вычисляет значение E и

присваивает его переменной V. Переменная V может быть различной : числовая переменная, строковая переменная, элемент числового или строкового массива.

LIST N.

Выводит листинг программы начиная со строки с номером N. допускает что величина N=0 до тех пор пока она не будет определена другим способом. Если строка существует тогда она становится текущей строкой и обозначается курсором. Особенностью системы является то, что LIST 49172 например будет такой же как LIST 20.

LLIST N.

Посылает листинг программы на принтер начиная со строки с номером N. N становится текущей строкой.

LOAD.

Имеет много различных оттенков и позволяет загружать программу БЕЙСИК, массивы переменных и блоки кодов. Информация (код или программа) записана на магнитной ленте в 2-х частях. 1-я представляет 17-байтовый заголовок, 2-я блок кодов. Каждая из этих частей использует одинаковый формат единичных битов, который в 2 раза длиннее 0-вых битов.

В SPECTRUM используется головной маркер (+00 для заголовка и +FF для блока кодов) и завершающий байт четности после заголовка и блока кодов. Блок заголовка разбит на 5 частей:

1. Одиночный байт информации который равен 0 для программы БЕЙСИК, 1 для числового массива, 2 для массивов символов, 3 - для блока кодов.
2. Следующие 10 байтов содержат название файла. Название больше 10 символов не применяется.
3. 2 байта которые содержат общую длину блока кодов.
4. 2 байта, которые для программы содержат начальный номер строки или адрес загрузки блока кодов.
5. 2 байта, которые для программы содержат длину области программы.

Блок кодов загружается в требуемую область RAM как указано в заголовке.

LPRINT.

Вызывает посылку элементов печати на принтер (аналогично PRINT).

MERGE F.

Загружает программу и ее переменные с магнитной ленты и соединяет с существующей программой и ее переменными. Команда выполняет обработку новой программы как блока данных, который должен быть загружен в рабочую область. Далее области программы сравниваются строка за строкой и новые строки копируются из рабочей области в основную область програм-

мы. Область программы переопределяется и расширяется если необходимо. Поскольку строки соединены подобная операция соединяет и область переменных.

NEW.

Это команда повторного запуска системы без изменения системных переменных RAMTOP, P-RAMT, RASP, PIP, UDG, графика, определенная пользователем при этом не сбрасывается.

NEXT A.

Команда должна рассматриваться как команда, которая выполняет работу в цикле FOR-NEXT В котором управляемая переменная установлена (см. FOR выше). Этапы выполнения команды следующие:

- к начальному значению переменной цикла прибавляется значение шага;
- получившаяся величина записывается на место начального значения и сравнивается с конечным значением;
- если начальное значение больше конечного при положительном шаге или меньше конечного при отрицательном шаге, то выполняется оператор, следующий за командой NEXT;
- в противном случае производится переход на оператор, следующий за командой FOR, которая создала эту переменную цикла.

OPEN #N,X$.

Для использования с микроприводом. Однако возможно открывать и закрывать потоки данных следующим образом:

- строка PRINT #5;"WORKS" не будет работать до тех пор, пока поток данных номер 5 не будет открыт использованием строки OPEN #5,"S". Строка CLOSE #5 - закрывает поток. Следующие строки показывают ввод с клавиатуры и вывод на экран:

```
10 OPEN #5, "K"
20 INPUT #5; A$
30 OPEN #5, "S"
40 PRINT #5; A$
```

OUT M,N.

Команда позволяет посылать сигналы на выходной порт из БЕЙСИКа.

В пункте BORDER было показано как порт 254 управляет цветом бордюра: разные цвета были даны для случаев от N=0 до N=7, в тоже время порт 254 может использоваться для управления звуковым динамиком и следующая программа показывает как это сделать.

С помощью программы можно получить оценку времени которое требуется для того чтобы интерпретировать оператор (-------), так как быстрый

оператор даст высокий звук, а медленная интерпретация оператора даст ряд щелчков.

```
10 OUT 254,23:(-----------): OUT 254,7: GO TO 10
```

в программе оператор может быть например таким

```
PRINT RAND, LET A=B; или POKE 0,0
```

OVER N.

Команда управления выводом на экран. При N=1 содержимое позиции экрана объединяется с выводимым на экран символом по правилу "исключающее или". При N=0 выводимый символ затирает символ, находящийся на экране в позиции печати.

PAPER N.

Команда цвета. Может быть постоянной или временной. N может принимать значения от 0 до 9.

Использование N от 0 до 7 очень простое.

При N=8 будет использовано постоянное значение.

Использование PAPER 9 очень полезно при оформлении заголовков. Команда устанавливает, что цвет бумаги в области будет контрастным по отношению с используемым цветом чернил, следовательно, при светлых чернилах - зеленый, голубой, желтый, белый - бумага будет черной; а при темных чернилах - черный, синий, красный, фиолетовый - бумага будет белой.

Фактическим действием PAPER является:

- ⇨ для N=8 устанавливается бит 3, 4, 5 переменной MASK-P
- ⇨ для N=9 бит 7 переменной P-FLAG , а для N=0-7 биты 3,4,5 переменной ATTR-P.

PAUSE N.

Эта команда останавливает Z-80 на N прерываний. Единственной работой, которая выполняется за время паузы, является опрос клавиатуры. Период паузы закончится после N прерываний или при нажатии клавиш. Если N=0 то пауза будет длиться до первого нажатия на любую клавишу.

PLOT X,Y.

Экран представляет собой 256*176 элементов изображения, каждый из которых может управляться отдельно командой PLOT. В обычной работе команда PLOT установит бит RAM соответствующий биту изображения. Команда делает также этот элемент изображения текущим в системной переменной COORDS, который записывает координаты последнего элемента изображения.

Любой из цветовых элементов может использоваться в команде PLOT.

POKE N,M.

Команда позволяет вводить значение M по адресу N непосредственно в память. M может принимать значения от -255 до +255. Значения от 0 до 255 вводится непосредственно, а к значениям от -255 до -1 прибавляется +256.

PRINT.....

Команда позволяет выводить на TV-экран различные элементы. Элементы отделяются друг от друга разделителями; ";" - элементы выводятся подряд; "," - выполняется табуляция к следующей половине строки. Позицию экрана можно также указать с помощью TAB и AT.

Элементы могут быть элементами цвета, действие которых будет временно, числовыми выражениями или строковыми (строковое выражение объединение двухъ или более строк символов в одну путем стыковки, например: "PROG"+"NAME").

Все элементы управления цвета и управления табуляцией могут быть выполнены использованием соответствующих значений CHR$ и это может быть полезно.

RANDOMIZE N.

Команда устанавливает значение системной переменной SEED, расположенной в адресах 23670 и 23671 (5C76-5C77). Если N не задано то значение для SEED , берется из двухъ младших байтов FRAMES и может рассматриваться как случайное число. Если N задано тогда это значение копируется в SEED. Для дальнейшего уточнения работы генератора псевдослучайных чисел смотри функцию RND.

READ V1,V2...

Команда используется вместе со списком DATA и может рассматриваться как несколько команд LET . Элементы в DATA являются выражениями, а элементы в READ - переменными, которым эти выражения присваиваются. Сообщения об ошибке появляются, если нет соответствия между переменными и выражениями.

Заметьте что в оператор READ можно включать неописанные ранее переменные. Системная переменная DATA, расположенная в адресах 23639 и 23640 (5C57-5C58) используется для указания в списке DATA . Первоначально указатель устанавливается на адрес начала области программы и сбрасывается в это значение по RUN.

REM.

Полезная команда т.к. позволяет вносить комментарии. Вся строка БЕЙСИКа после REM , рассматривается как комментарий и не интерпретируется.

RESTORE N.

Эта команда используется вместе со списком DATA и относится к одному или нескольким DATA. Если N=0 то переменная DATADD указывает на байт перед областью программы. Если N определено то указывает на адрес строки БЕЙСИК с номером N если она существует или ближайшую после нее. Если N больше чем 9999 (самый большой разрешенный номер строки) или "удел" за последний номер программы, то DATADD указывает на последний байт в области программы.

RETURN.

По команде извлекается верхнее значение из стека GOSUB. Если обращение в стек корректно тогда интерпретатор будет выполнять эту команду. Иначе будет выдано сообщение об ошибке.

RUN N.

Позволяет выполнять программы БЕЙСИК. Если N не задано то выполнение начнется с первой строки. Когда N определено то выполнение начинается со строки программы с номером N или со следующей строки, после нее если строка N не существует. Команда инициирует требуемые указатели путем выполнения RESTORE и CLEAR до интерпретации какой либо строки.

STOP.

По этой команде появляется сообщение об ошибке "STOP" использование CONTINUE как прямой команды позволит затем продолжать программу от следующего оператора. Как и в ошибочной ситуации команда передает номер ошибки в системную переменную ERR-NR, которая располагается по адресу 23610 (5C3A) номер ошибки всегда меньше на 1, чем код ошибки, который появляется перед сообщением об ошибке.

VERIFY.

Команда позволяет сверить с оригиналом любую программу, выведенную на магнитную ленту с помощью SAVE. процесс проверки осуществляется для обеих частей программы - заголовка и блока данных. Ошибка "R TAPE LOADING ERROR" появляется если запись на ленте не точно соответствует оригиналу.

2.3. Функции БЕЙСИКа.

ABS X.

Все отрицательные числа становятся положительными в то время как положительные не меняются.

ACS X.

Аргумент "X" берется как косинус и функция возвращает величину угла в радианах.

AND (X AND Y).

Операция требует 2 операнда. Если операнды логически истинны, то операция полностью истинная. Однако, если один или оба операнда ложны, то операция полностью ложна.

Например :

```
IF AB AND BC THEN PRINT "YES"
```

PRINT "YES" будет выполнено, если выполнятся оба условия одновременно (AB и BC).

ASN X.

Аргумент X берется как синус и функция возвращает величину данного угла в радианах.

ATN X.

Аргумент X берется как тангенс и функция возвращает величину угла в радианах.

ATN является первой из 4-х функций, определенных в SPECTRUM использованием многочленов Чебышева.

ATTR (X,Y).

Эта функция имеет форму ATTR (строка, колонка). Она возвращает пользователю значение атрибутного байта.

Функция эквивалентна PEEK 22528+строка*32+колонка. Величина признака может рассматриваться следующим образом INK+PAPER*8+BRIGHT*64+FLASH*128.

BIN X.

Позволяет число в диапазоне от 0 до 65535 вводить в двоичной форме. Разрешается использовать любое число двоичных цифр до 16.

Следующая строка поэтому законна и даст числа от 0 до 10:

```
10 FOR A=BIN 1 TO BIN 1010 STEP 00000001: PRINT A: NEXT A
```

хотя SPECTRUM позволяет ввести числа в двоичной форме, они не могут печататься в этой форме без использования программ.

CHR$ X.

Эта функция возвращает пользователю символ, как строку для данного кода. Следующая строка показывает как распечатать набор символов. Первые 33 (0-32) значения используются в SPECTRUM для служебных целей и не имеют графического эквивалента.

```
10 FOR A=33 TO 255 : PRINT CHR$ A: NEXT A
```

CODE X$.

Функция возвращает пользователю код для первого символа в аргументе X$, код будет равен 0 , если X$ 0-ая строка.

Например :

```
10 PRINT CODE "A"
20 LET B$="A"
30 PRINT CODE B$ COS X.
```

Аргумент X рассматривается в радианах и функция возвращает соответствующий косинус для этого угла.

EXP X.

Для аргумента X функция возвращает величину E в степени X, где E=2,7182818. Вычисления производятся по многочлену Чебышева.

FN X(A,B$,C...), FN X$(A,B$,C...).

Вызывает функцию X или X$, возвращает числовой или строковый результат выполнения функции, соответственно. В SPECTRUM 26 числовых и 26 строчных функций. За знаком функции (FN) должна следовать 1 буква или 1 буква и знак $, а затем требуемые аргументы в скобках.

IN X.

Аргумент X используется как адрес порта и считываемое значение будет в диапазоне 0-255.

INKEY$.

Функция опрашивает клавиатуру и возвращает строку, если нажата клавиша. Значение строки соответствует нажатой клавише в режиме курсора L. Функция распознает символы при простом нажатии клавиши или при нажатии клавиши одновременно с клавишами CAPS SHIFT или SYMBOL SHIFT. Другие комбинации дают пробел или ?. Если не одна клавиша не нажата, или нажата более чем одна, то длина входной строки равна нулю.

Например :

```
10 PRINT AT 10,15;"
20 LET A$=INKEY$
30 PRINT AT 10,15;A$
40 IF CODE A$=0 THEN GOTO 10
50 GOTO 20
```

INT X.

Для данной величины X эта функция возвращает только целую часть. У отрицательного числа прежде всего отбрасывается дробная часть, затем вычитается 1, чтобы округлить результат.

LEN X$.

Функция определяет длину заданной строки. 0-ая строка даст 0-ой результат.

Однако элементы цвета, полученные нажатием клавиши могут включится в строку. Эти элементы будут иметь длину 2.

LN X.

Для заданного X функция возвращает натуральный логарифм для величины X (логарифм по основанию E=2,7182818).

Представляет собой другую функцию вычисленную в SPECTRUM использованием многочлена Чебышева.

NOT X.

Дает логическое состояние значений в SPECTRUM. Если X=0, то значение NOT X равно 1, если X=1 (не равно нулю), то значение NOT X равно 0 (напоминаем, что значение 1 является "истиной", а 0 "ложью").

Например :

```
10 FOR A=0 TO 1
20 IF NOT A THEN PRINT "YES": GOTO 40
30 PRINT "NO"
40 NEXT A
```

OR (X OR Y).

Операция требует два операнда. Если любой из операндов логически истинный, тогда операция истинна. Однако если ни один из операндов не является истинным, тогда операция ложна.

Например :

```
IF AB OR BC THEN PRINT "YES"
```

PRINT "YES" будет выполнено, если любое одно или оба условия будут выполнены (AB) или (BC).

PEEK X.

Возвращает величину, находящуюся по адресу X. результат всегда целое число в диапазоне от 0 до 255.

PI.

PI/2 является константой в таблице постоянных в ROM памяти. Эта величина выбирается, удваивается и передается пользователю. Величина PI=3.14...

POINT X,Y.

Возвращает 1, если элемент изображения в этом положении установлен, и 0 если сброшен (X и Y - координаты точки на экране TV).

RND.

В SPECTRUM случайные числа выдаются генератором псевдослучайных чисел (от 0 до 1). Значение системной переменной SEED находящейся в

адресах 23670 и 23671 выбирается, модифицируется и возвращается при каждом обращении к RND.

SEED сбрасывается в 0 при инициализации SPECTRUM. С этого времени она является последовательностью целых чисел 0 - 65535. Величина не будет повторятся до тех пор пока все числа не использовались.

SCREEN$ (X,Y).

Формат функции SCREEN$ (строка, столбец) и функция возвращает строку, содержащую символ для заданной позиции. Распознает символы с кодами в диапазоне 32-127 , когда они нормальны или инвертированы. Работа заключается в сравнении содержимого 8 байтов для области данного символа с 8-ми байтовым представлением в генераторе знаков. Функция возвращает 0-ю строку, если символ не является ни одним из 96 символов генератора. Нежелательно, чтобы функция исключала из сравнения символы определенные пользователем, т.к. они могут появиться. Однако весьма трудно проверить обычные графические символы, которые не хранятся в 8-ми байтовой форме, а создаются когда требуются.

Пример :

```
10 PRINT "@";SCREEN$ (0,0)
```

SGN X.

Положительные величины превращаются в +1, все отрицательные - в -1, 0 дает 0.

SIN X.

Это обычная функция SIN в радианах.

SQR X.

Вычисляется квадратный корень аргумента X.

STR$ X.

Аргумент для этой функции рассматривается как числовое выражение. Операция вычисляет это числовое выражение, печатает его в рабочей области так, как оно будет выведено на TV-экране, и затем возвращает пользователю полученную строку.

TAN X.

Вычисление тангенса угла в радианах.

USR X.

Аргумент X, который является числовым выражением, принимается как адрес программы машинного кода, написанного пользователем.

Когда команда вызывается, то Z-80 останавливает выполнение программы монитора и вместо этого выполняет команды с адресом X и далее.

Команда машинного кода RET вызовет возврат к программе монитора. В программе написанной в машинном коде, можно модифицировать регистр

IY, но его необходимо восстановить перед возвратом в программу монитора (но маскируемое прерывание должно быть запрещено пока пара регистров IY содержит любую величину отличную от +5C3A). Однако значение в этой паре регистров должно быть сохранено или восстановлено если требуется успешный возврат к БЕЙСИКу. Функция возвращает пользователю содержимое пары регистров BC.

В SPECTRUM числовые значения передаются через пару регистров BC.

USR X$.

Возвращает адрес требуемого символа из графики определяемой пользователем.

Аргумент состоит из 1-го символа и он должен быть в диапазоне "A-U". Адрес графической области пользователя (в ней находятся символы, определяемые пользователем) определен в системной переменной UDG (23675, 23676). Этот адрес соответствует значению USR "A".

Пример определения символа:

```
10 DATA BIN 00000001
11 DATA BIN 00000010
12 DATA BIN 00000100
13 DATA BIN 00000100
14 DATA BIN 00001000
15 DATA BIN 01001000
16 DATA BIN 00101000
17 DATA BIN 00010000
18 FOR A=0 TO 7; READ B; POKE USR "A"+A,B: NEXT A
19 PRINT CHR$ 144: REM="A"
```

VAL X$.

Позволяет вычислять выражения, которые содержатся в строках.

Например :

```
10 PRINT VAL "100+7*4"
```

VAL$ "X$".

Эта функция требует строку в качестве аргумента и эта строка должна быть заключена в кавычки. Строка, заключенная в кавычки считается строковой переменной, если это не так, то возникает ошибка. Функция возвращает пользователю значение этой строковой переменной.

Например :

```
10 LET A$="100+7*4"
20 PRINT VAL$ "A$+""6*""+A$"
```

2.4. Управляющие символы.

CHR$ 6- печать запятой.

CHR$ 8- перемещение обратно.

CHR$ 9- перемещение вправо.

CHR$ 13- следующая строка экрана.

CHR$ 16 до CHR$ 21

Эти управляющие знаки дают временные цвета.

Они соответствуют операторам INK, PAPER, FLASH, BRIGHT, INVERSE, OVER.

Пример - установить INK в RED.

```
10 PRINT "XXXX";CHR$ 16;CHR$ 2;"YYYY"
```

CHR$ 22 - AT.

Печатаемая позиция видоизменяется в позицию данными следующих 2-х символов. Первый символ дает номер строки, второй - номер столбца.

CHR$ 23 - TAB.

Печатаемая позиция устанавливается в столбец, заданный следующим символом. Если текущий номер столбца равен или превышает новое значение столбца, тогда печатаемая позиция будет на следующей строке экрана.

3. Микропроцессор Z-80.

3.1. Введение.

В компьютере ZX SPECTRUM использован микропроцессор (МП) Z-80. Он имеет шестнадцатиразрядную адресную шину, восьмиразрядную шину данных и шину управления, которая включает в себя ряд управляющих сигналов. Управляющую шину мы не будем рассматривать подробно, т.к. для программирования не требуются такие знания.

Шина данных предназначена для передачи данных между микропроцессором и внешними абонентами, например, такими как ОЗУ (память) или клавиатура. Шина является двунаправленной, то есть данные могут передаваться как в микропроцессор, так и из него.

Шина адреса предназначена для передачи адреса к внешним абонентам и является однонаправленной. Микропроцессор выставляет адрес на адресную шину и сопровождает его управляющими сигналами, которые определя-

ют для кого предназначен этот адрес - для ОЗУ или для других абонентов (клавиатура, принтер и так далее). Управляющие сигналы формируются автоматически в зависимости от требуемых действий и не требуют вмешательства программиста.

Для успешного программирования необходимо знать некоторые особенности компьютера. В частности рассмотрим два управляющих сигнала, которые оказывают на это влияние. Это сигналы INT и NMI.

Управляющий сигнал NMI предназначен для немаскированного прерывания, то есть при возникновении этого сигнала прерывание выполняющейся программы произойдет в любом случае. При этом управление передается подпрограмме монитора, которая в свою очередь передает управление программе, находящейся по адресу указанному в системной переменной, которая расположена по адресу 23728 (адрес системной переменной, содержащей адрес программы, которой нужно передать управление). К сожалению в мониторе ZX SPECTRUM допущена ошибка и при возникновении прерывания этого типа происходит сброс, аналогичный выключению компьютера.

Управляющий сигнал INT предназначен для маскируемого прерывания программы. В ZX SPECTRUM этот сигнал формируется электронной схемой через каждые 50 миллисекунд. Возникновение этого сигнала приводит к прерыванию выполняющейся программы и передаче управления программе сканирования клавиатуры. После сканирования клавиатуры управление передается прерванной программе. Необходимо отметить, что это маскируемое прерывание, а это означает, что его можно запретить или разрешить с помощью определенных команд микропроцессора.

3.2. Логический взгляд на Z-80.

Рассмотрим МП Z-80 с точки зрения программиста. Он состоит из устройства управления, арифметико-логического устройства (АЛУ) и ряда ячеек памяти, называемых регистрами.

Устройство управления предназначено для управления работой микропроцессора.

АЛУ предназначено для выполнения различных арифметических и логических команд.

Регистры микропроцессора предназначены для хранения информации различного рода, например: данных (числа), адресов данных, команды, адреса команды. Программисту доступны не все регистры, а только часть. Другая часть регистров используется микропроцессором для организации своей работы.

Рассмотрим назначение регистров Z-80. Регистры МП Z-80 своеобразны и имеют свои имена. Они обозначаются буквами латинского алфавита: F, A, B, C, D, E, H, L, SP, PC, IX, IY, I, R. Это основной набор регистров. Есть еще один дополнительный набор, его называют альтернативным, его мы рассмотрим ниже.

Некоторые из этих регистров - восьмиразрядные (однобайтные) и могут содержать целые числа от 0 до 255. Это регистры F, A, B, C, D, E, H, L, I, R.

Другие - шестнадцатиразрядные (двухъбайтные). Они могут содержать целые числа от 0 до 65535. Это регистры SP, PC, IX, IY.

Некоторые однобайтные регистры могут объединятся в двухъбайтовые регистровые пары BC, DE, HL. В этом случае о такой паре можно говорить как об отдельном шестнадцатиразрядном регистре.

Микропроцессор Z-80 является весьма асимметричным микропроцессором, то есть каждый регистр обладает специфическими чертами, делающими его непохожим на остальные, и различные регистры предназначены для выполнения различных функций. Рассмотрим регистры основного набора.

Регистр A

Аккумулятор.

Этот регистр называется аккумулятором, т.к. во многих случаях результат исполнения операции остается в нем. В этом регистре выполняется наибольшее количество арифметических и логических команд. Это основной однобайтный регистр микропроцессора. Во многих командах даже не указывается к какому регистру они относятся; предполагается, что они относятся к регистру A.

Регистры B, C, D, E, H, L

Регистры общего назначения.

Эти регистры довольно широко используются МП при выполнении различных команд. Их характерная особенность состоит в том, что они могут использоваться и как однобайтные одиночные регистры и как двухъбайтные шестнадцатиразрядные регистровые пары BC, DE, HL. В последнем случае наибольшее количество команд имеет регистр HL. При работе с двухъбайтными целыми числами он имеет такое же важное значение, что и регистр A при работе с однобайтными.

Все эти пары используются, как правило, для хранения адресов. При этом в регистровой паре (рег. паре) HL часто хранится адрес того байта с которым производится операция. В регистровой паре DE как правило находится адрес места назначения. Во многих операциях, связанных с перемещением чего-либо куда-либо, регистровая пара HL указывает откуда взять, а регистровая пара DE - куда отправить. В регистровой паре BC при этом обычно содержится длина пересылаемой информации.

При объединении одиночных регистров в пары, в первом хранится старший байт, а во втором - младший. Итак B, D, H - старший; C, E, L младший.

Регистр SP

Указатель стека.

В тех случаях, когда необходимо на некоторое время освободить какой-либо регистр (или регистры) и запомнить его содержимое впредь до дальнейшего использования, компьютер может использовать в качестве места временного хранения данных особый участок оперативной памяти, называемый стеком (машинным). Стек - это такая форма организации памяти, при которой загрузка и выгрузка данных выполняется по принципу "последним пришел - первым уйдешь". Он может быть организован почти в любом месте оперативной памяти (даже в экранной области).

Стек "растет" сверху вниз. То есть если вершина стека находится по адресу 60000, то следующее число, помещенное в стек, будет записано в ячейку 59999, а следующее за ним - в 59998. При этом перемещается и его вершина. В регистре SP содержится адрес вершины стека. При помещении очередного числа в стек, регистр SP автоматически уменьшает свое значение, а при извлечении числа из стека - увеличивает. Заслав свой адрес в указатель стека, вы можете организовать стек в другом месте оперативной памяти.

В принципе, для временного хранения промежуточных результатов вычислений можно использовать обычные методы типа LD (ADDR),A, но при хранении данных на стеке доступ к ним проще.

Еще одно назначение стека связано с использованием подпрограмм. Так, при изменении нормального порядка исполнения программы, (когда команды выполняются последовательно одна за другой) и при переходе на выполнение подпрограммы необходимо запомнить адрес возврата, чтобы продолжить вычисления с точки, находящейся после места вызова подпрограммы. Этот адрес (в виде двухъ байтов) и помещается в стек. По окончании выполнения подпрограммы по команде RET адрес возврата извлекается из стека и помещается в программный счетчик PC, что и обеспечивает возврат. Разумеется, если во время работы подпрограммы вы что-то будете помещать на стек, то при возврате будет выбрана из стека помещенная туда величина, что неизбежно приведет к нарушению работы программы. Значит, если при работе в подпрограмме вам необходимо поместить сколько-то чисел на стек, то столько же надо оттуда снять до того, как встретится команда RET.

Регистр PC

Программный счетчик.

Этот регистр также, как и регистр SP - шестнадцатиразрядный. Он служит для хранения адреса той команды, которая должна выполнятся следующей. Когда микропроцессор (МП) начинает выполнять эту команду, программный счетчик увеличивается на единицу, если команда однобайтная, на два - если команда двухъбайтная и так далее. Таким образом, по окончании выполнения этой команды МП знает какую команду выполнять следующей. Так обеспечивается последовательное выполнение команд из которых состоит программа.

Если в программе имеется переход в иное место (аналогично GO TO БЕЙСИКа) или вызов подпрограммы (аналогично GO SUB БЕЙСИКа), то именно программный счетчик и обеспечивает такой переход. В него автоматически помещается адрес перехода.

Регистр F

Флаговый регистр.

Этот регистр отличается от всех остальных и играет весьма своеобразную роль. Его рассматривают не как отдельный регистр хранящий восьмиразрядное число, а как восемь отдельных флаговых битов. Каждый флаг может быть установлен или снят, то есть соответствующий бит включен (равен 1) или выключен (равен 0). Структура флагового регистра следующая :

бит 0 - флаг переноса C;
бит 1 - флаг сложения/вычитания N;
бит 2 - флаг четности/переполнения P/O;
бит 3 - не используется;
бит 4 - флаг полупереноса H;
бит 5 - не используется;
бит 6 - флаг нуля Z;
бит 7 - флаг знака S.

Флаги C, P/O, Z, S - активно используются программистом. Мы рассмотрим работу с этими флагами далее.

Флаги N и H программистом не используются. Они участвуют при выполнении операций с десятичными числами, выраженными в двоичнодесятичной форме.

Регистры IX, IY

Индексные регистры.

Это шестнадцатиразрядные регистры. Используются они, как правило, для запоминания адресов. Характерной особенностью этих регистров является, что благодаря им можно организовать так называемую индексную адресацию. Если вам надо работать с какими либо таблицами данных, то вы можете при выборке нужного элемента таблицы (например 55-го) не задавать его адрес, а хранить в индексном регистре адрес начала таблицы. Обращение к нужному (55-ому) элементу выполняется путем обращения к этому индексному регистру с указанием величины "смещения" нужного вам элемента от начала таблицы, то есть в данном случае числа 55.

В SPECTRUM регистр IY хранит адрес 23610, который является начальным адресом таблицы системных переменных. Это позволяет легко обращаться к любой системной переменной. Первоначальную установку содержимого регистра IY на этот адрес выполняет после включения компьютера в сеть программа инициализации, находящаяся в ПЗУ, но вы, конечно можете изменить эту установку, поместив туда какой-либо свой адрес.

Регистр IX широко используется компьютером при исполнении команд загрузки/выгрузки программ: LOAD, SAVE, VERIFY, MERGE. В нем хранится адрес начала места загрузки (выгрузки) блока. Когда вы выполняете загрузку/выгрузку из БЕЙСИКа, вам об этом думать не надо. Процедуры, расположенные в ПЗУ, все за вас сделают, но если вы выполняете эти операции из программы в машинных кодах, то надо установить требуемое значение в этом индексном регистре.

Регистр R

Регистр регенерации.

Это внутренний системный регистр, который используется МП автоматически, без участия программиста. Электрические заряды в микросхемах оперативной памяти имеют тенденцию "утекать", поэтому для нормальной работы компьютера необходимо производить подзарядку (регенерацию) оперативной памяти. В этих целях и используется регистр R.

Регистр I

Вектор прерываний.

Для компьютеров иных систем, собранных на базе МП Z-80 , это регистр используется для организации обмена с периферийными устройствами. В SPECTRUM эта возможность не используется и этот регистр участвует только в формировании телевизионного сигнала.

Альтернативный набор регистров.

Одной из отличительных черт МП Z-80 является наличие альтернативного набора регистров. Это регистры F', A', B', C', D', E', H', L'. Они полностью идентичны регистрам F, A, B, C, D, E, H, L.

Наличие такого альтернативного набора позволяет быстро переключаться с одного набора на другой. Когда нужно произвести какие-то операции с регистрами общего назначения, а они уже заняты и потерять содержащиеся в них данные нельзя, перед вами есть три пути:

1. Записать содержимое регистра в память, а потом вернуть его назад.
2. Поместить содержимое регистра в стек, а потом вернуть его оттуда.
3. Переключиться на альтернативный набор регистров и работать с ним, а потом переключиться назад на основной набор.

Третий путь самый оперативный.

Благодаря наличию альтернативного набора МП может обслуживать две задачи. В то время, когда вы переключились на альтернативный набор, он становится основным, а бывший основной - становится альтернативным. Физически эти наборы совершенно идентичны и поэтому нет никаких программных средств, чтобы определить с каким именно набором вы в данный момент работаете. Программа может получаться довольно хитросплетенной и от программиста требуется определенная внимательность.

3.3. Структура программ в машинных кодах.

Программа должна существовать как набор машинных команд и данных, размещенных в памяти. В SPECTRUM единицей памяти является один байт. Поэтому программа в машинных кодах является последовательностью 8-ми битовых значений.

Пример:

```
LOCATION    0          11110011
            1          11001100
            2          11001100
            3          10101010
```

это представление является иллюстрацией программы в машинных кодах. Но очень громоздко и приводит к ошибкам. Поэтому лучше использовать 16-тиричную запись кодов или программу на БЕЙСИКе, которая печатает байты в 10-ричной форме.

```
10 FOR A=0 TO 7
20 PRINT "LOCATION"; TAB 10; A; TAB 15; PEEK A
30 NEXT A
```

программа показывающая 16-ричное представление рассматривается на следующем примере:

```
10 FOR A=0 TO 7
20 LET H=INT(PEEK a/16)
30 LET L=PEEK A-H*16
40 PRINT "LOCATION"; TAB 10;A; TAB 15;CHR$(48+H+7*(H9));
CHR$(48+L+7*(L9))
50 NEXT A
```

эти программы включены, чтобы показать способ получения 10-ного или 16-ричного листинга программы в машинных кодах. Рассмотрим шаги получения документированного листинга. Используя таблицу машинных кодов можно представить эти машинные команды в следующем виде:

```
LOCATION 0 - DI запрещение маскируемого прерывания;
LOCATION 1 -  XOR A сложение по модулю 2 регистра A с
регистром A;
LOCATION 2 - 4 LD DE,+FFFF загрузка регистровой пары DE
константой FFFF (десятичное 65535);
LOCATION 5 - 7 JP +11CB переход по абсолютному адресу.
```

Все машинные команды Z-80 имеют свою мнемонику и программы в машинных кодах записываются с использованием мнемоники, а не двоичного

или 16-ричного представления. Такая мнемоническая форма представления называется ассемблерной. Программист вначале обычно составляет программу в мнемоническом виде на языке ассемблера, а затем, в процессе трансляции, это представление преобразуется в шестнадцатиричное (16-ричное).

Поэтому такое написание очень удобно при записи программ.

4. Математика машинного программирования.

В МП Z-80 все данные передаются 8-ми битовыми комбинациями, но такое представление трудно для работы и поэтому программисты обычно используют 16-ричное представление.

4.1. Шестнадцатиричное представление.

В этой системе счисления первые 10 знаков записываются цифрами от 0 до 9 и к ним добавляются 6 знаков, записываемых буквами от a до F. Таблица показывает соответствие чисел 0 - 15.

2-ное	10-ное	16-ное
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	10	A
1011	11	B
1100	12	C
1101	13	D
1110	14	E
1111	15	F

из таблицы видно что одна 16-ричная цифра записывается 4-х битовыми двоичными числами. Поэтому 8-ми битовое число записывается парой 16-ричных знаков, а 16-битовое 4-мя 16-ричными знаками.

Например:

0000 0000 (2-ное)=00(16-ное);

0100 1111 (2-ное)=4F(16-ное);

0000 0000 0000 0000 (2-ное)=0000(16-ное);

0100 1100 1010 1111 (2-ное)=4CAF(16-ное).

Следующая программа переводит числа из 10-ной в 16-ричную систему.

```
10  INPUT "DECIMAL NUMBER",D
20  IF D65535 THEN GOTO 170
30  PRINT "DECIMAL",D
40  DIM H(4)
50  DIM H$(4)
60  LET H(1)=INT(D/4096)
70  LET D=D-H(1)*4096
80  LET H(2)=INT (D/256)
90  LET D=D-H(2)*256
100 LET H(3)=INT(D/16)
110 LET D=D-H(3)*16
120 LET H(4)=D
130 FOR A=1 TO 4
140 LET H$(A)=CHR$ (H(A)+48+7*(H(A)9))
150 NEXT A
160 PRINT "HEXADECIMAL",H$
170 PRINT
180 GO TO 10
```

В программе десятичное значение последовательно уменьшается для поиска 16-ричного эквивалента. Строки 130- 150 эти значения переводят в код ASCII.

Следующая программа показывает 16-ричное в 10-ное преобразование.

```
10  DIM H$(4)
20  INPUT "HEX.CHARACTERS",H$
30  IF CODE H$=32 THEN GO TO 20
40  LET D=0: LET K=1
50  FOR A=4 TO 1 STEP -1
60  IF H$(A)=CHR$ 32 THEN GO TO 90
70 IF (H$(A)"0" OR H$(A)"9") AND (H$(A)"A" OR H$(A)"F") THEN
GO TO 120
80  LET D=D+K*(CODE H$(A)-48-7*(CODE H$(A)57))
85  LET K=K*16
90  NEXT A
100 PRINT "HEXADECIMAL",H$
```

```
110 PRINT "DECIMAL",D
120 PRINT
130 GO TO  20
```

в программе строка 16-ричных знаков H$ всегда имеет 4 знака. Если значение не задано пользователем, требование всегда повторяется. Последний пример в этом разделе показывает как преобразование 16-ного в 10-ное может быть выполнено вручную.

```
16-ричный член = 789A
десятичный эквивалент = 7*4096=28672
8*256 = 2048
+ 9*16  =  144
A*1   =   10
----------
789A  =30874
или попарно:
десятичный эквивалент = 78*256=30720
9A*1  =  154
--------------
789A  =30874
```

4.2. Абсолютная двоичная арифметика.

Одинарный регистр или байт памяти содержит 8-ми битовое двоичное число. Это число может быть в диапазоне 0000 0000-1111 1111. Ни в одном из этих случаев число не может быть ни отрицательным ни дробным и это образует основу абсолютной двоичной арифметики - значение в памяти или в регистре всегда положительное и целое.

Также важно представлять себе, что значение, содержащееся в памяти или в регистре изменяется циклически в пределах от 0 до 255. Это означает, что если в операции сложения полученное значение превышает число 255 , то окончательный результат записывается после вычитания 256, а при вычитании к значению меньше 0 прибавляется 256.

Это показано на следующих примерах.

```
254+44 дает 42
или
FC+2C дает 28 87-200 дает 143
или
57-C8 дает 8F
```

в системах, основанных на Z-80 , все числа представлены в абсолютной двоичной арифметике, но программист часто нуждается в различной интерпретации чисел, таких как положительное или отрицательное, целое или дробное.

4.3. Дополнительная арифметика.

Принцип дополнительной арифметики очень прост. Но при использовании ее результат может быть очень неожиданным.

Метод позволяет рассматривать программисту двоичные значения в интервале 0000 0000-0111 1111 , как эквивалент 10-ных значений от 0 до 127, а двоичные значения в интервале 1000 0000-1111 1111, как эквивалент 10-ных значений от -128 до -1.

В результате такой интерпретации бит 7 (левый крайний бит 8-ми битового числа) рассматривается как знаковый. Он будет 0 для положительного числа и 1 для отрицательного. На диаграмме 4.1 показан принцип дополнительной арифметики для 8-ми битных чисел.

двоичное	десятичное	16-ное	крайний левый бит - знаковый
0111 1111	+127	7F	+
0111 1110	+126	7E	+
.....	.....	....	+
.....	.....	....	+
0000 0010	+2	02	+
0000 0001	+1	01	+
0000 0000	0	00	
1111 1111	-1	FF	-
1111 1110	-2	FE	-
.....	.....	....	-
.....	.....	....	-
1000 0001	-127	81	-
1000 0000	-128	80	-

Диаграмма 4.1.

Заметим, что возможно распространить принципы дополнительной арифметики на 16-битовые числа. В этом случае интервал возможных значений чисел будет от -32768 до 32767 десятичных.

Таким образом положительные числа в двоичной дополнительной арифметике записываются так, как они есть на самом деле, например:

+127 (десятичное) - 0111 1111 (двоичное);

+3 (десятичное) - 0000 0011 (двоичное).

Отрицательные числа для записи их в двоичной дополнительной форме требуют преобразований. Рассмотрим эти преобразования:

1. Отрицательное число записывают в абсолютном двоичном виде.

2. Полученный двоичный код инвертируют (меняют "0" на "1" и наоборот).

3. К результату вышеизложенных действий прибавляют 1. например:

номер действия	1	2	3
- 127 (десятичное)	0111 1111	1000 0000	1000 0001
- 128 (десятичное)	1000 0000	0111 1111	1000 0000
- 3 (десятичное)	0000 0011	1111 1100	1111 1101

4.4. Двоично-десятичная арифметика (BCD-арифметика).

Это особый вид представления десятичных чисел в двоичной форме. Каждая цифра десятичного числа представляется в виде четырех двоичных разрядов, например:

1563 - 0001 0101 0110 0011

9701 - 1001 0111 0000 0001.

Таким образом в двоично-десятичной арифметике цифры от 0 до 9 записываются в виде четырехразрядных двоичных чисел от 0000 до 1001.

Двоичные комбинации от 1010 до 1111 (от 10 до 15) считаются запрещенными. Арифметические операции с числами, представленными в двоично-десятичной форме необходимо выполнять учитывая вышеизложенные правила. В каждом байте памяти можно записать две десятичные цифры числа, представленного в двоично-десятичной форме, одну в правом полубайте, другую в левом полубайте. В системе команд микропроцессора Z-80 есть 3 команды работающие с BCD-арифметикой.

4.5. Интегральное представление.

Интерпретатор БЕЙСИК в системе SPECTRUM использует 5 байт для представления чисел. Любое целое число в интервале от -65535 до +65535 включительно записывают в интегральной форме, в то время как любые дробные или целые числа вне этого диапазона записываются в плавающей форме.

В интегральной форме первый байт всегда 0. второй байт содержит 0, если число положительное и 255 (FF), если оно отрицательное. 3-ий и 4-ый байты содержат целое значение как 16-битовое беззнаковое число в дополнительном коде. Но 3-й байт содержит всегда младшую часть, а 4-ый старшую. 5-й байт не используется, но всегда содержит 0.

Следующая программа демонстрирует интегральную форму (в десятичном виде) для любых целых чисел вводимых пользователем. Строка 20 программы гарантирует получение интегральной формы.

```
10 INPUT N
20 IF NINT N OR N65535 OR N65535 THEN GO TO  10
30 PRINT "NUMBER CHOSEN =",N
40 LET V=PEEK 23627+256*PEEK 23628
50 FOR A=1 TO 5
60 PRINT A; ". "; TAB 5;PEEK(a+V)
70 NEXT A
80 GO TO 10
```

В приведенной выше программе переменная V указывает на начало поля переменных и адреса с +1 по +5 будут содержать 5 байт введенного числа.

Программа дает результаты в виде:

выбрано значение 0

1.0
2.0
3.0
4.0
5.0

так как 0 - положителен и представляется как 0*1+0*256 выбрано значение 1516

1.0
2.0
3.236
4.5
5.0

так как 1516-положительно и представляется как 236*1+5*256 выбрано значение -1.

1.0
2.255
3.255
4.255
5.0

так как -1 отрицательно и имеет форму FFFF.

Замечание : ошибкой программиста будет перевод -65536. это значение не соответствует интегральной форме 0, 255, 0, 0, 0.

4.6. Представление с плавающей точкой.

В системе SPECTRUM минимальное значение которое можно записать в плавающей форме равно 2.9387359E-39. Максимальное значение, записанное в плавающей форме, равно 1.7014118E+38. Причем эти числа могут быть как положительными, так и отрицательными. Такие числа хранятся в экспоненциальном представлении.

Под числа с плавающей точкой выделяют пять байт. В старшем (пятом) байте хранится экспонента числа, увеличенная на 128, а в остальных четырех байтах - мантисса. Причем старший бит четвертого байта указывает на знак мантиссы. Если он включен (равен 1) - число отрицательное, если же выключен (равен 0) - положительное.

байт 5	байт 4	байт 3	байт 2	байт 1
76543210	7 1 6543210	76543210	76543210	76543210
экспонента+128	мантисса первый бит - знак мантиссы			

для тех, кто не помнит, что такое экспоненциальная форма, кратко поясним.

Предположим, что вам надо записать какое-то очень большое число. Если вы будете его делить раз за разом на два до тех пор, пока оно не станет по абсолютной величине меньше единицы, то сколько раз вам пришлось его делить это и есть экспонента, а то, что осталось после последнего деления - мантисса. Обратите внимание на то, что мантисса всегда больше 0.5, но меньше 1. Мантисса хранится в четырех младших байтах, а экспонента - в пятом (старшем). Обратная операция выполняется так: мантиссу надо умножить на двойку в степени экспоненты.

Если же число было малой десятичной дробью, например 0.0000375, то делить его на два, конечно бессмысленно. В этом случае его наоборот надо умножать на два до тех пор, пока оно не станет больше, чем 0.5, но меньше, чем 1. Только экспонента в этом случае считается отрицательной. На знак экспоненты указывает ее величина. Если в старшем байте стоит число, большее 128, то экспонента положительная, а если меньше, то отрицательная.

Следующая программа показывает как хранится число с плавающей точкой.

```
10 INPUT N
20 IF N=0 THEN GOTO 40
30 LET N=N+.2E-38
40 PRINT "NUMBER ";N
50 PRINT
```

```
60 PRINT "EXP"; TAB 10; "MANTISSA"
70 LET V=PEEK 23627+256*PEEK 23628
80 PRINT PEEK (V+1); TAB 10;
90 FOR A=2 TO 5
100 PRINT PEEK (V+a);" ";
110 NEXT A
120 PRINT
130 GOTO 10
```

сравните результаты ввода чисел .5 и 1/2 (задавать именно в указанном виде).

4.7. Представление символьных переменных.

Символьная переменная или строка символов (STRING) в системе SPECTRUM представляется путем задания начального адреса строки в регистровой паре DE и длины строки в регистровой паре BC.

5. Система команд.

5.1. Команды и данные.

Команды поделены на 18 групп, каждая из которых состоит из тех команд, которые сильно похожи друг на друга. Рассмотрим 6 классов данных, которые могут следовать за командой.

1. Однобайтовая константа (+DD).

То есть число в диапазоне 00-FF 16-ное , 0-255 10-ное. те команды, которые требуют за собой однобайтовую константу, имеют мнемоническую приставку '+DD'.

```
Например: LD A,+DD
```

2. Двухъбайтовая константа (+DDDD), то есть число в диапазоне 0000-FFFF 16-ричное и 0-65535 10-ное. Те команды, которые требуют за собой двухъбайтовую константу, имеют мнемоническую приставку '+DDDD'.

```
Например: LD HL,+DDDD
```

3. Двухъбайтовый адрес (ADDR).

То есть число в диапазоне 0000-FFFF 16-ричное,0-65535 10-ное, которое используется как адрес памяти. Те команды, которые требуют за собой двухъбайтовый адрес, имеют приставку 'ADDR'.

Например: JP ADDR

4. Однобайтовая константа смещения (E) . то есть число в диапазоне 00-FF 16-ричное, от -128 до +127 10-ное. Число всегда представлено в дополнительной арифметике. Те команды, которые требуют за собой однобайтовую константу смещения, имеют мнемоническую приставку 'E'.

```
Например: JR E
```

5. Однобайтовая индексирующая константа смещения (+D).

То есть число в диапазоне 00 - FF 16-ричное , -128 - +127 10-ное. Число представлено в дополнительной арифметике. Те команды, которые требуют за собой однобайтовую индексирующую константу смещения, имеют мнемоническую приставку '+D'.

```
Например: LD A,(IX+D)
```

6. Однобайтовая индексирующая константа смещения и однобайтовая константа (+D,+DD).

То есть два числа в диапазоне 00-FF 16-ричное, первое из которых рассматривается как 10-ное -128 - +127, а второе как десятичное 0-255.

Команды, требующие за собой 2-х байтов данных, сопровождаются мнемониками '+D' и '+DD'.

```
Например: LD (IX+D),+DD
```

важно отметить следующее: если в команде указывается адрес данных или пара регистров, содержащих адрес данных, то он записывается в круглых скобках.

Например:

```
LD (HL),+DD - загрузить константу +DD в ячейку памяти, адрес
которой находится в паре регистров HL.
```

5.2. Группы команд.

Существует много путей для разделения на группы сотен различных машинных команд. Однако метод, выбранный здесь, разделяет команды на 18 функциональных групп.

При описании команд, сначала записывается мнемоническое обозначение, затем операнды, если они есть и затем эквивалент машинного кода в 16-ричном виде.

После изучения команд по группам читатель выполнит программы БЕЙСИК из следующей главы, которые иллюстрируют эти команды.

Группа 1. команда 'нет операции'.

NOP 00

выполнение команды требует 1,14 мкс. ни один из регистров или флагов не изменяется. Команда NOP используется программистом для организации задержек, но чаще для удаления ненужных команд из программы.

Группа 2. Команда загрузки регистра константами.

```
LD A,+DD    3E DD
LD H,+DD    26 DD
LD L,+DD    2E DD
LD B,+DD    06 DD
LD C,+DD    0E DD
LD D,+DD    16 DD
LD E,+DD    1E DD
```

каждая строка этих команд требует 2-х байтов памяти: один для кода операции, второй для константы. Команды записывают в регистр соответствующие значения. Старые значения регистра пропадают.

Следующие команды выполняют загрузку пары регистров двухъбайтовыми константами.

```
LD HL,+DDDD 21 DD DD
LD BC,+DDDD 01 DD DD
LD DE,+DDDD 11 DD DD
LD IX,+DDDD DD 21 DD DD
LD IY,+DDDD FD 21 DD DD
LD SP,+DDDD 31 DD DD
```

строка команды требует 3 или 4 байта в памяти. Код операции занимает 1 или 2 байта и 2 байта занимает константа. Первый байт константы загружается в младший регистр регистровой пары то есть L,C,E,X,Y,P, а второй байт - в старший регистр, то есть H,D,B,I,S.

Эти команды записывают в регистровые пары данные значения. Эти значения часто рассматриваются программистом, как 2-х байтовый адрес, но могут быть так же и двухъбайтовым числовым значением и 2-я отдельными 1-байтовыми числовыми значениями.

Команды этой группы не изменяют флагов.

Группа 3. Команды копирования регистров и обмена.

Существует 59 команд, которые выполняют копирование регистров и регистровых пар. Эти команды лучше делить на 4 подгруппы.

Подгруппа А. Команды копирования регистр-регистр.

Следующая таблица дает коды операций команд, выполняющих копирование содержимого одного регистра в другой.

регистр	LD	LD	LD	LD	LD	LD	LD
R	A,R	H,R	L,R	B,R	C,R	D,R	E,R
A	7F	67	6F	47	4F	57	5F
H	7C	64	6C	44	4C	54	5C
L	7D	65	6D	45	4D	55	5D
B	78	60	68	40	48	50	58
C	79	61	69	41	49	51	59
D	7A	62	6A	42	4A	52	5A
E	7B	63	6B	43	4B	53	5B

ни одна из команд, приведенных в таблице не изменяет флаги.
Существует кроме того 4 команды для I и R регистров.

```
LD A, I      ED 57
LD A, R      ED 5F
LD I, A      ED 47
LD R, A      ED 4F
```

эти последние команды влияют на флаг переполнения/четности.

Подгруппа B. Команды копирования регистровая пара - регистровая пара. Существует только 3 команды этой подгруппы. И они копируют значения в указатель стека.

```
LD SP, HL    F9
LD SP, IX    DD F9
LD SP, IY    FD F9
```

эти команды не изменяют флагов.

Заметьте, что если содержимое регистровой пары надо копировать в другую регистровую пару и эти команды не подходят, необходимо выполнить 2 команды копирования регистр-регистр. Например, нет команды LD HL,BC и ее заменяют, используя LD H,B и LD L,C. иначе содержимое 1-ой регистровой пары может быть сохранено в стеке и в последствии переписано во 2-ю регистровую пару.

Подгруппа C. Команда EX DE,HL.

Существует только одна команда, которая позволяет обменивать содержимое регистров в пределах основного набора регистров.

EX DE,HL (EB)

это очень полезная команда. Она производит обмен содержимого пары DE с содержимым пары HL. Значение флагов не изменяется. Команда используется когда необходимо, чтобы адрес или 2-х байтовая числовая кон-

станта из пары DE была записана в пару HL, но содержимое последней не было потеряно.

Подгруппа D. Команды альтернативного набора регистров.
В этой группе 2 команды.

EXX (D9), EX AF,A'F' (E8)

команда EXX вызывает переключение регистров H,L,B,C,D,E на регистры H',L',B',C',D',E'.

Команда EX AF,A'F', как следует из мнемоники, переключает регистры A и F на A' F'.

Альтернативные регистры часто используются для хранения адресов и данных. Помещенные в альтернативные регистры, эти значения сохраняются от искажения и могут быть легко и просто восстановлены.

Группа 4. Команды загрузки регистров из памяти.

В систему Z-80 входит много команд, позволяющих находить данные в памяти и затем загружать их в основной регистр. Все эти команды требуют, чтобы программист указал адрес, откуда данные должны быть скопированы в регистр.

Команды этой группы лучше рассматривать как команды трех подгрупп в соответствии с методом адресации.

Виды адресации:

- ⇨ Косвенная - 2-х байтовый адрес уже размещен в адресной регистровой паре.
- ⇨ Непосредственная - адрес указан в двухъ байтах вслед за кодом операции.
- ⇨ Индексная - адрес данных должен быть высчитан путем сложения значения смещения +D с основным адресом уже содержащимся в регистровой паре IX или IY.

Подгруппа A. Команды, использующие непосредственную адресацию. Эти команды не влияют на значения флагов.

LD A, (ADDR)	3A ADDR
LD SP, (ADDR)	ED 7B ADDR
LD HL, (ADDR)	2A ADDR или ED 6B ADDR
LD BC, (ADDR)	ED 4B ADDR
LD DE, (ADDR)	ED 5B ADDR
LD IX, (ADDR)	DD 2A ADDR
LD IY, (ADDR)	FD 2A ADDR

Подгруппа B. Команды, использующие косвенную адресацию.
Эти команды так же не изменяют значения флагов.

LD A, (HL)	7E
LD A, (BC)	0A

```
LD A,(DE)    1A
LD H,(HL)    66
LD L,(HL)    6E
LD B,(HL)    46
LD C,(HL)    4E
LD D,(HL)    56
LD E,(HL)    5E
```

Подгруппа С. Команды загрузки регистров из памяти, использующие индексную адресацию. Состояние флагов не меняется.

```
LD A,(IX+D) DD 7E D
LD H,(IX+D) DD 66 D
LD L,(IX+D) DD 6E D
LD B,(IX+D) DD 46 D
LD C,(IX+D) DD 4E D
LD D,(IX+D) DD 56 D
LD E,(IX+D) DD 5E D
LD A,(IY+D) FD 7E D
LD H,(IY+D) FD 66 D
LD L,(IY+D) FD 6E D
LD B,(IY+D) FD 46 D
LD C,(IY+D) FD 4E D
LD D,(IY+D) FD 56 D
LD E,(IY+D) FD 5E D
```

Группа 5. Команды записи в память содержимого регистров или константы.

Обычно команды этой группы выполняют операции противоположные тем, которые делают команды группы 4.

Команды позволяют содержимое регистров переписывать в память или записывать туда константы. Эти команды лучше рассмотреть по трем группам.

Подгруппа А. Команды непосредственной адресации.

```
LD (ADDR),A  32 ADDR
LD (ADDR),HL 22 ADDR или ED 63 ADDR
LD (ADDR),BC ED 43 ADDR
LD (ADDR),DE ED 53 ADDR
LD (ADDR),IX DD 22 ADDR
LD (ADDR),IY FD 22 ADDR
LD (ADDR),SP ED 73 ADDR
```

приведенные команды выполняют только непосредственную адресацию и важно заметить, что нет команд для записи в память констант. Если нео-

бходимо записать в память константу, то ее нужно загрузить в регистр, например в A, а затем содержимое регистра A записать в память.

Команды записи в память содержимого пары регистров можно рассматривать как две команды. Например: LD (ADDR),HL эквивалентна двум командам - LD (ADDR),L и LD (ADDR+1),H; хотя таких команд на самом деле нет.

Команды подгруппы часто используются для сохранения адресов и значений в памяти, когда эти значения рассматриваются как переменные. Например, часто используется LD (RAMTOP),HL, где RAMTOP системная переменная, используемая для хранения текущего значения верхней границы памяти. Выборка текущего значения может быть позднее выполнена командой 4 группы, например, LD HL,(RAMTOP).

Подгруппа B. Команды косвенной адресации.

Команды этой подгруппы позволяют копировать содержимое регистров в память, адрес которой содержится в регистровой паре HL,BC или DE. Существует также команда записи однобайтовой константы по адресу, указанному в паре HL.

```
LD (HL),A   77
LD (BC),A   02
LD (DE),A   12
LD (HL),H   74
LD (HL),L   75
LD (HL),B   70
LD (HL),C   71
LD (HL),D   72
LD (HL),E   73
LD (HL),+DD 36 DD
```

Подгруппа C. Команды индексной адресации.

```
LD (IX+D),A DD 77 D
LD (IX+D),H DD 74 D
LD (IX+D),L DD 75 D
LD (IX+D),B DD 70 D
LD (IX+D),C DD 71 D
LD (IX+D),D DD 72 D
LD (IX+D),E DD 73 D
LD (IX+D),+DD       DD 36 DD D
LD (IY+D),A FD 77 D
LD (IY+D),H FD 74 D
LD (IY+D),L FD 75 D
LD (IY+D),B FD 70 D
LD (IY+D),C FD 71 D
```

```
LD (IY+D),D FD 72 D
LD (IY+D),E FD 73 D
LD (IY+D),+DD                FD 36 DD D
```

команды пятой группы не изменяют значения флагов.

Группа 6. Команды сложения.

Эта группа команд выполняет арифметические действия. Команды сложения позволяют программисту прибавить (в абсолютной двоичной арифметике) заданное число к регистровой паре, регистру, байту в памяти.

Команды этой группы могут быть поделены на 3 подгруппы, каждая из которых имеет собственное имя.

Команды ADD.

Команды INC.

Специальный случай сложения, когда к числу прибавляется 1.

Команда ADC.

Значение флага переноса прибавляется к результату.

Флаг переноса - это один из битов регистра флагов и используется для сигнализации о том, было ли при выполнении последней арифметической операции двоичное переполнение регистра или байта памяти. Команды ADD и ADC меняют флаг переноса, а INC - не меняет.

Подгруппа А. Команда ADD (простое сложение).

```
ADD A,+DD    C6 DD
ADD A,A      87
ADD A,H      84
ADD A,L      85
ADD A,B      80
ADD A,C      81
ADD A,D      82
ADD A,E      83
ADD A,(HL)   86
ADD A,(IX+D)                 DD 86 D
ADD A,(IY+D)                 FD 86 D
ADD HL,HL    29
ADD HL,BC    09
ADD HL,DE    19
ADD HL,SP    39
ADD IX,IX    DD 29
ADD IX,BC    DD 09
ADD IX,DE    DD 19
ADD IX,SP    DD 39
ADD IY,IY    FD 29
```

```
ADD IY,BC    FD 09
ADD IY,DE    FD 19
ADD IY,SP    FD 39
```

команды ADD, приведенные выше очень просты. Команда устанавливает или сбрасывает флаг переноса в зависимости от того, было или нет двоичное переполнение регистра или пары регистров.

Это показано на следующих примерах:

```
1. рег.      A=60          16-ных
             B=90

выполняется ADD A,B
             A=F0
             B=90          флаг переноса сброшен
2.           A=A8
             B=7E

выполняется ADD A,B
             A=26
             B=7E          флаг переноса установлен
```

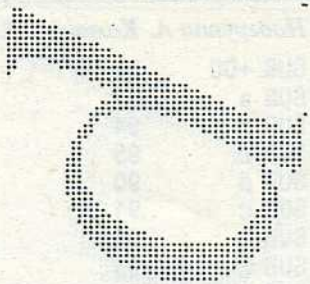
Подгруппа B. Команда INC.

Команды этой подгруппы позволяют прибавить единицу к 8-ми битовому регистру, ячейке памяти или 16-разрядной паре регистров. Во всех случаях флаг переноса не изменяется.

```
INC A        3C
INC H        24
INC L        2C
INC B        04
INC c        0C
INC D        14
INC E        1C
INC (HL)     34
INC (IX+D)   DD 34 D
INC (IY+D)   FD 34 D
INC HL       23
INC BC       03
INC DE       13
INC SP       33
INC IX       DD 23
INC IY       FD 23
```

Подгруппа С. Команда ADC

```
ADC A,+DD    CE DD
ADC A,A      8F
ADC A,H      8C
ADC A,L      8D
ADC A,B      88
ADC A,C      89
ADC A,D      8A
ADC A,E      8B
ADC A,(HL)   8E
ADC A,(IX+D)          DD 8E D
ADC A,(IY+D)          FD 8E D
ADC HL,HL    ED 6A
ADC HL,BC    ED 4A
ADC HL,DE    ED 5A
ADC HL,SP    ED 7A
```



команды этой группы позволяют сложить два числа с учетом текущего значения флага переноса.

Все команды этой подгруппы меняют флаг переноса. Он сбрасывается если операция ADC не дает двоичного переполнения и устанавливается, если дает.

Примеры показывают это:

```
рег. A=60   флаг=1        ADC A,B      рег. A=F1   флаг=0
рег. B=90                              рег. B=90
рег. A=A8   флаг=1        ADC A,B      рег. A=27   флаг=1
рег. B=7E                              рег. B=7E
```

Группа 7. Команды вычитания.

Позволяют вычесть заданное число из одинарного регистра, пары регистров, или байта памяти.

Команды этой группы полезно разделить на 3 подгруппы, каждая из которых имеет свою мнемонику.

Команда SUB.

Команда DEC,
специальный случай вычитания, когда из числа вычитается 1.

Команда SBC,
значение флага переноса вычитается из результата.

Все команды SUB и SBC изменяют флаг переноса в зависимости от того, требовался ли двоичный заем. Команда DEC оставляет флаг переноса без изменения.

Подгруппа A. Команды SUB.

```
SUB +DD       D6 DD
SUB a         97
SUB H         94
SUB L         95
SUB в         90
SUB c         91
SUB D         92
SUB E         93
SUB (HL)      96
SUB (IX+D)    DD 96 D
SUB (IY+D)    FD 96 D
```

замечание: мнемоника команд хотя и пишется, как указано выше, но SUB L подразумевается SUB A,L и так далее поскольку все команды используют регистр A.

В Z-80 команда SUB выполняет "истинное" абсолютное двоичное вычитание, как показано в примерах. Флаг переноса сбрасывается если первоначальное значение регистра A больше или равно вычитаемому и устанавливается если меньше. Например:

```
1.    рег. A=DC     SUB B     рег. A=32 флаг=0
      рег. B=AA               рег. B=AA
2.    рег. A=AA     SUB B     рег. A=ce флаг=1
      рег. B=DC               рег. B=DC
```

Подгруппа B. Команды DEC.

Команды этой подгруппы позволяют вычесть 1 из содержимого 8 битового регистра, ячейки памяти или 16-битовой пары регистров. Во всех этих случаях флаг переноса не изменяется.

```
DEC A         3D
DEC H         25
DEC L         2D
DEC B         05
DEC c         0D
DEC D         15
DEC E         1D
DEC (HL)      35
DEC (IX+D)    DD 35 D
DEC (IY+D)    FD 35 D
DEC HL        2B
DEC BC        0B
DEC DE        1B
DEC SP        3B
```

```
DEC IX          DD 2B
DEC IY          FD 2B
```

Подгруппа С. Команды SBC.

```
SBC A,+DD       DE DD
SBC A,A         9F
SBC A,H         9C
SBC A,L         9D
SBC A,B         98
SBC A,C         99
SBC A,D         9A
SBC A,E         9B
SBC A,(HL)      9E
SBC HL,HL       ED 62
SBC HL,BC       ED 42
SBC HL,DE       ED 52
SBC HL,SP       ED 72
SBC A,(IX+D)                DD 9E D
SBC A,(IY+D)                FD 9E D
```

команда SBC выполняет "истинное" вычитание, если флаг переноса сброшен и вычитание "с займом", если флаг переноса установлен. Это может быть полезно, если используются значения повышенной точности. Например 4-х байтовое число содержащееся в регистрах H',L',H,L может быть вычтено из другого числа, хранящегося в регистрах D',E',D,E следующим образом:

```
AND A       сброс флага переноса
SBC HL,DE   вычитание младшей части
EXX         переключение набора
SBC HL,DE   вычитание старшей части
EXX         переключение набора
```

и соответствующий заем между младшей и старшей частями учитывается.

Группа 8. Команды сравнения.

Команды этой группы используются очень часто во всех программах. Они позволяют программисту сравнить значение находящиеся в регистре A с константой, значением в регистре и области памяти. Команды выполняет операцию вычитания без переноса, без запоминания результата вычитания и только устанавливают флаги в регистре F. Первоначальное значение в регистра A не изменяется. Флаг переноса устанавливается, как и при операциях вычитания. Сравнение, которое больше или равно сбрасывает флаг переноса, а меньше - устанавливает.

```
CP+DD       FE DD
CP A        BF
CP H        BC
```

```
CP L          BD
CP B          B8
CP C          B9
CP D          BA
CP E          BB
CP (HL)       BE
CP (IX+D)     DD BE D
CP (IY+D)     FD BE D
```

следующие примеры показывают использование команды CP B.

```
рег. A=31   CP B      оставляет регистры без изменений
            B=30      и сбрасывает флаг переноса (3130)
рег. A=30   CP B      оставляет регистры без изменений
            B=30      и сбрасывает флаг переноса (30=30)
рег. A=01   CP B      оставляет регистры без изменений
            B=30 и устанавливает флаг переноса (01).
```

Группа 9. Команды логики (AND, OR, XOR).

Эти команды предполагают, что первый операнд находится в регистре A. операция выполняется побитно и 8-битовый результат возвращается в регистр A.

Подгруппа A. Команда AND.

Логическая операция выполняется над каждыми двумя битами и результат равен 1 только, если оба используемых бита равны 1. В противном случае результирующий бит равен 0.

Следующий пример показывает как команда выполняется над восемью отдельными битами.

```
            В двоичной      в 16-ричной
            10101010        AA
            AND             AND
            11000000        C0
результат ------------------------
            10000000        80
```

в этой подгруппе следующие команды:

```
AND+DD      E6 DD
AND A       A7
AND H       A4
AND L       A5
AND B       A0
AND C       A1
AND D       A2
AND E       A3
AND (HL)    A6
```

```
AND (IX+D)  DD A6 D
AND (IY+D)  FD A6 D
```

командой AND удобно сбрасывать биты 0-8 регистра A.

Этот процесс называется демаскированием и позволяет проверять определенные биты байта данных. Следующий пример показывает как это можно сделать.

В системе SPECTRUM биты 0-2 системной переменной ATTR-P содержат информацию о постоянном цвете чернил. Когда цвет чернил изменяется, старый цвет удаляется использованием команды AND и новый цвет вводится по команде ADD:

```
LD A,(ATTR-P)          находится системная переменная
AND +F8                демаскируется старый цвет
ADD A,+NEW COLOR       добавляется новый цвет
LD (ATTR-P),A          сохраняется системная переменная
```

Подгруппа B. Команда OR.

Логическая операция "или" выполняется над каждыми 2-мя битами, и результат равен 1, если один или оба проверяемых бита равны 1. В противном случае бит результата равен 0.

Пример показывает как команда выполняет операцию над 8 отдельными битами:

```
            в 2-ной      в 16-ной
            10101010     AA
            OR           OR
            11000000     C0
результат   ----------------
            11101010     EA
```

в подгруппе следующие команды

```
OR +DD      F6 DD
OR A        B7
OR H        B4
OR L        B5
OR B
OR C        B1
OR D        B2
OR E        B3
OR (HL)     B6
OR (IX+D)   DD B6 D
OR (IY+D)   FD B6 D
```

при работе команды OR будут установлены или проверены, что они установлены от 0 до 8 битов регистра A.

Пример показывает одно из использования команды OR.

В SPECTRUM биты 5,4,3 системной переменной ATTR-P определяют постоянный цвет бумаги. Поэтому можно сделать цвет бумаги белым использованием команды OR +DD.

```
LD A,(ATTR-P)          найти системную переменную
OR +38      установить биты 5,4,3
LD (ATTR-P),A          сохранить системную переменную
```

Подгруппа С. Команда XOR.

При работе команды проверяются два разряда. Если хотя бы 1 из них, но не вместе, установлен в единицу, то результирующий бит устанавливается также в 1; иначе результирующий бит сбрасывается в ноль ("исключающее или" или сложение по модулю два).

Пример показывает каким образом команда XOR выполняет восемь отдельных операций над битами.

```
          в 2-ном      в 16-ном
          10101010     AA
          XOR          XOR
          11000000     C0
результат ----------------
          01101010     6A
```

команды в этой подгруппе:

```
XOR+DD    EE DD
XOR A     AF
XOR H     AC
XOR L     AD
XOR B     A8
XOR C     A9
XOR D     AA
XOR E     AB
XOR (HL)  AE
XOR (IX+D) DD AE D
XOR (IY+D) FD AE D
```

при работе этой команды будут установлены или сброшены с 0 до 8 битов регистра А. Это, вероятно вначале трудно понять, но рассмотрите пример, приведенный выше еще раз; а именно:

```
AA XOR C0 дает 6A
```

в этом примере 2-ым операндом является байт C0 , в котором все биты, за исключением 6 и 7 равны нулю.

Поэтому эффект работы операции XOR заключается в переключении битов 6 и 7 первого операнда и изменении AA в 6A. использование команды XOR в программах машинного кода часто является сложным, но команда "XOR a", однако часто используется как альтернативная, для LD A,+00. Обе

эти команды очищают регистр A, но XOR использует только один байт, в то время как LD A,+00 использует два байта.

Все команды AND,OR,XOR сбрасывают флаг переноса когда они используются.

Группа 10. Команды перехода.

Имеется 17 команд, которые позволяют выполнять переходы внутри программы. Команды этой группы лучше всего рассматривать в 8 подгруппах.

4 из этих подгрупп содержат команды, которые зависят от состояния одного из основных флагов, и если требуется, флаги будут подробно рассмотрены.

Подгруппа A. Команда абсолютного перехода.

```
JP ADDR      C3 ADDR
```

при выполнении команды значение ADDR загружается в счетчик команд и выполнение программы машинного кода продолжается с этого адреса.

Подгруппа B. Команды переходов, использующие косвенную адресацию.

```
JP (HL)      E9
JP (IX)      DD E9
JP (IY)      FD E9
```

эти команды приводят к тому, что 16-битная величина, хранящаяся в соответствующем регистре загружается в счетчик команд. Команды этой группы часто используются, когда переход должен осуществляться на команду, адрес которой находится в соответствующей паре регистров.

Подгруппа C. Команда относительного перехода.

```
JR E         18 E
```

команда добавляет к счетчику команд число E и после этого начинает выполнятся команда, на которую указывает счетчик команд. Заметьте, что текущий адрес фактически является адресом, идущим после JR E, поскольку счетчик команд уже получил приращение. Число E всегда рассматривается как число, представленное в дополнительной двоичной форме и может принимать значения от -128 до +127. Следовательно переход возможен как вперед, так и назад.

Подгруппа D. Команда перехода по флагу переноса.

Имеется 4 команды, позволяющие делать такой переход.

Флаг переноса.

Этому флагу соответствует бит 0 регистра F и он является флагом, который показывает имело ли место или нет двоичное переполнение. Но этот флаг может при определенных условиях сбрасываться или устанавливаться.

Имеется много примеров, когда на флаг переноса не влияет выполнение команд.

Кратко можно описать следующее правило:

1. Все команды ADD и ADC Влияют на флаг переноса. Если нет переполнения флаг сбрасывается, но если есть переполнение, то флаг устанавливается.
2. Все команды SUB, SBC, CP влияют на флаг переноса. Если был 2-ный заем, тогда флаг устанавливается; в противном случае сбрасывается.
3. Все команды AND, OR, XOR сбрасывают флаг переноса.
4. Команды сдвига влияют на флаг переноса.

К командам перехода по флагу переноса относятся следующие команды:

```
JP NC,ADDR  D2 ADDR        переход, если
JR NC,E     30 E           флаг сброшен
JP C,ADDR   DA ADDR        переход, если
JR C,E      38 E           флаг установлен
```

в качестве примера этих команд рассмотрите следующую программу, которая распознает цифры в коде ASCII.

```
.....                      введите в регистр A, распознаваемый к
CP +30      код для цифры 0
JP C,ERROR  переход, если он находится за границей диапазона
CP +3A      код для ":"
JP NC,ERROR опять переход, если он находится
за пределами диапазона
.....                      продолжение только с цифрами
```

то есть диапазон 30-39.

первые две команды в вышеприведенном примере определяют нижнюю границу диапазона цифр в коде ASCII (30). Вторые две команды определяют верхнюю границу диапазона цифр в коде ASCII (39). Если код, находящийся в регистре A не является цифровым символом кода ASCII, то будет переход по адресу ERROR, а если код, находящийся в регистре A является символом цифры, то выполнение программы продолжится.

Подгруппа E. Команды перехода по флагу 0.

Имеется 4 команды, которые позволяют выполнить переход только в том случае, если флаг нуля будет таким же, как требуется командой.

Флаг 0.

Этому флагу соответствует 6-й бит регистра F и в большинстве случаев он устанавливается, если результат равен 0, в противном случае он сбрасывается.

Например:

```
6C ADD 5A дает C6 и флаг нуля сбрасывается
6C ADD 94 дает 00 и флаг нуля устанавливается.
```

Можно привести правило:

1. Все команды ADD, INC, ADC, SUB, DEC, SBC, CP, AND, OR, XOR, использующие одиночные регистры и команды ADC, SBC использующие пару регистров, установят флаг 0 , если результат равен 0, в противном случае флаг нуля сбрасывается.
2. Команды сдвига, команды проверки битов и команды поиска блоков влияют на флаг.
3. Команды LD за исключением LD A,I и LD A,R не влияют на флаг нуля.

Команды перехода по флагу нуля:

```
JP NZ,ADDR   C2 ADDR        переход только когда
JR NZ,E      20 E           сброшен флаг 0
JP Z,ADDR    CA ADDR        переход только когда
JR Z,E       28 E           установлен флаг 0
```

команды этой группы обычно широко используются и пример показывает, как распознать символы ASCII.

```
.........  введите в регистр A код символа
CP +3B     это символ ";" ?
JP Z,S-COLON                переход, если это так
CP +2C     это символ "," ?
JP NZ,ELSE переход, если нет.
.........  продолжение при ","
```

Подгруппа F. Команды перехода по флагу знака.

Имеется две команды, которые позволяют сделать переход только в том случае, если флаг знака будет таким же как требуется командой.

Флаг знака.

Этому знаку соответствует 7 бит регистра F и в большинстве случаев он представляет собой копию старшего (левого) бита результата.

Когда 8 битовое или 16 битовое двоичное число представлено в форме дополнительной двоичной арифметики, тогда левый бит (бит 7 или бит 15) принимает бит знака. Он сбрасывается для положительных чисел, и устанавливается для отрицательных.

Поэтому флаг знака может сбрасываться с помощью положительных результатов и устанавливаться с помощью отрицательных.

Кратко опишем правила:

1. Все команды ADD, INC, ADC, SUB, DEC, SBC, CP, AND, OR, XOR использующие 1 регистр, а команды ADC и SBC использующие 2 регистра, будут влиять на флаг знака, как указано выше.
2. Команды поиска блока и большинство команд сдвига также влияют на флаг знака.
3. Команды LD, за исключением LD A,I и LD A,R не влияют на флаг знака.

Команды перехода по состоянию флага :

```
JP P,ADDR    F2 ADDR - переход, если результат "+"
JP M,ADDR    FA ADDR - переход, если результат "-"
```

команды в этой группе обычно не используются. Частично потому, что требуют абсолютного адреса и частично потому, что бит знака может считываться некоторыми другими методами.

Пример показывает использование этих команд.

```
.......... введите в регистр A код символа.
AND a      это влияет на флаг знака
JP P,ELSE  коды 00-7F вызывают переход
.......... продолжение только с графическими символами (коды
80-FF).
```

Подгруппа G. Команды перехода по флагу переполнения/четности.

Имеются две команды, которые позволяют выполнить переход, если флаг переполнение/четности будет таким как требуется.

Флаг переполнения/четности.

Этому флагу соответствует 2-ой бит регистра F и является флагом двойного назначения. Некоторые команды используют флаг, чтобы указать на переполнение, в то время как другие команды используют его, чтобы хранить результат проверки на четность.

Понятие переполнение относится к арифметическим операциям с числами в дополнительном двоичном коде, как показано следующими примерами:

```
16-ный 0A ADD 5C дает 66
10-ный 10 ADD 92 дает 102, которое правильно - нет
переполнения.
16-ный 6A ADD 32  дает 9C
10-ный 106 ADD 50 дает 156, которое неправильно - дает
переполнение.
```

Переполнение может также возникнуть при вычитании, а именно:

```
16-ный 83 SUB 14 дает 6F
10-ный 131 SUB 20 дает 111, которое неправильно, то есть
имеет переполнение.
```

Таким образом, если мы при выполнении арифметических операций рассматриваем исходные данные как числа, представленные в дополнительном двоичном коде, то флаг переполнения/четности указывает на наличие переполнения (он устанавливается, если результат операции, представленный в дополнительном двоичном коде больше +127 (десятичное) или меньше -128 (десятичное)).

Понятие четности касается числа единичных битов в заданном байте.

Это показано примером:

байт 01010101 имеет четное число единиц и флаг
устанавливается
байт 00000001 имеет нечетное число единиц и флаг
сбрасывается.

Кратко сформулируем правила:

1. Все команды AND, OR, XOR и большинство команд сдвига дают результат, который может быть проверен на четность.
2. Все команды ADD, ADC, SBC, SUB, CP использующие одиночные регистры и команды ADC, SBC использующие два регистра дают результат который может быть проверен на переполнение.
3. Команда INC установит флаг, если результат будет равен 80 (16-ричное) и команда DEC так же, установит флаг, если результат равен 7F (16-ричное).
4. Различные другие команды, а именно LD A,I и LD A,R и команды отработки блоков также влияют на флаг переполнение/четности.

Команды перехода этой группы:

JP PO,ADDR E2 ADDR - переход, если равенство нечетное или нет переполнения

JP PE,ADDR EA ADDR - переход, равенство четное или имеет место переполнение.

Команды этой подгруппы используются только в тех случаях, когда невозможно использование других команд.

Группа 11. Команда DJNZ.

Одиночная команда в этой группе является одной из самых полезных и самой широко используемой в Z-80.

DJNZ E 10 E

работа этой команды может быть уподоблена циклу FOR-NEXT программы БЕЙСИК следующего вида:

```
FOR B=X TO 0 STEP -1: NEXT B
```

в этом цикле переменной "в" присваивается значение X. затем с каждым проходом цикла она уменьшается до тех пор, пока не достигнет значения 0.

Команда DJNZ используется подобным же образом, прежде всего программист должен определить значение переменной цикла и занести ее в регистр B. Затем идет цикл. Наконец, используется команда DJNZ, но с большим вниманием, чтобы убедиться, что величина смещения подходит.

Следующий пример показывает как эта команда должна использоваться для печати алфавита.

```
LD        B,+1A      26 букв в алфавите  "A" - первая
LOOP LD   A,+5B      в регистр A загрузим код последней
```

```
                        буквы алфавита
SUB B                   уменьшим значение регистра A
RST 0010                печатаем символ, чей код
                        находится в регистре A
DJNZ LOOP               сравниваем регистр B с нулем, если в регистре B
                        ноль, то выполняем следующую команду, а если
                        в регистре B не ноль, то вычитаем из регистра B
                        единицу и переходим на метку LOOP
RET                     возврат в БЕЙСИК.
```

16-м кодом для этого примера является:

06, 1A, 3E, 5B, 90, D7, 10, FA, C9

где байт "3E" является меткой цикла.

Автор обнаружил, что лучшие методики для вычисления правильной величины "E" в примерах коротких машинных кодов, подобных приведенному выше, заключаются в представлении байта "E" величиной "FF" и затем считать в обратном направлении (в 16-ой форме) до тех пор, пока не будет достигнут байт цикла.

В приведенном выше примере имеется 5 шагов назад и соответственная величина для "E" равна FA.

Группа 12. Команды стека.

В большинстве программ машинных кодов осуществляется активное использование машинного стека: программистом в качестве места, где можно хранить данные; микропроцессором - для сохранения адресов возврата.

Команды, которые образуют эту группу можно разделить на две подгруппы для пользователя и три подгруппы для микропроцессора.

Подгруппа A. PUSH и POP.

Эти команды позволяют программисту по PUSH сохранить 2 байта данных в машинном стеке, а по POP копировать 2 байта из машинного стека.

Эти 2 байта данных должны копироваться в строго определенную пару регистров, но важно знать, что не осуществляется никаких пометок, указывающих к какому из двухъ регистров какой байт принадлежит.

```
PUSH AF     F5
PUSH HL     E5
PUSH BC     C5
PUSH DE     D5
PUSH IX     DDE5
PUSH IY     FDE5
POP AF      F1
POP HL      E1
POP BC      C1
POP DE      D1
```

```
POP IX      DDE1
POP IY      FDE1
```

Нужно понимать работу этих команд, если в программе машинного кода собираются использовать машинный стек не в прямом смысле.

Когда выполняется команда PUSH , указатель стека предварительно уменьшается так, чтобы указать на свободный байт стека. В этот байт записывается содержимое старшего регистра. Затем указатель стека уменьшается вторично и величина из младшего регистра копируется в стек.

Противоположные действия имеют место во время выполнения команды POP.

Важно помнить, что указатель стека после выполнения одной из этих команд указывает текущий адрес стека, использованный в "последний раз".

Команды этой подгруппы часто используются одновременно и следующий пример подтверждает это:

```
PUSH AF     сохранить AF
PUSH BC     сохранить BC
.......     "другая работа"
POP BC      восстановить BC
POP AF      восстановить AF
```

в примере под словами "другая работа" подразумевается использование регистров A, F, B, с "без вреда", но она ("другая работа") не должна изменять содержимое указателя стека.

Подгруппа B. команды обмена со стеком.

Команды этой подгруппы не часто используются, но они могут оказаться полезными:

```
EX (SP),HL  E3
EX (SP),IY  DD E3
EX (SP),IX  FD E3
```

эти инструкции позволяют программисту обменивать текущую величину, имеющуюся в указанной паре регистров с последним значением машинного стека. Значение указателя машинного стека не меняется.

Использование этих команд может привести к путанице и лучше всего их использовать как альтернативу командам POP и PUSH в особых случаях. Например:

величина "первая" находится в стеке, величина "вторая" в паре регистров HL, и если программист намерен обменять их друг с другом, то имеется два способа:

1. Использовать команду EX (SP),HL тогда данные из стека по текущему значению указателя стека будут размещены в паре HL, а из HL передадутся в стек.

2. Использовать дополнительный регистр для временного хранения "первого", то есть:

```
POP BC      сохраните "первый" в регистровой паре BC
PUSH HL     "второй" передать в стек
PUSH BC     "первый" передать в стек
POP HL      "первый" передать из стека в HL
```

кроме того команды этой подгруппы могут использоваться для изменения адреса возврата (см. ниже).

Подгруппа C. Команды вызова подпрограмм (CALL).

Команды машинного кода CALL эквивалентны команде БЕЙСИК "GOSUB". Команды включаются в эту группу, т.к. микропроцессор использует стек машины, то есть область, в которой хранятся адреса "возврата".

Имеются 9 команд в этой группе и они позволяют перейти на подпрограмму без условий или в зависимости от состояний 4-х флагов.

Командами в этой подгруппе являются:

CALL ADDR	CD ADDR
CALL C,ADDR	DC ADDR
CALL NC,ADDR	D4 ADDR
CALL Z,ADDR	CC ADDR
CALL NZ,ADDR	C4 ADDR
CALL M,ADDR	FC ADDR
CALL P,ADDR	F4 ADDR
CALL PE,ADDR	EC ADDR
CALL PO,ADDR	E4 ADDR

действие команд CALL будут следующими:

1. Текущее значение счетчика команд, то есть адрес первой ячейки после ADDR, сохраняются в стеке. Указатель стека изменяется как в случае команды PUSH. Старший байт счетчика команд сохраняется в стеке над младшим байтом.
2. Адрес вызываемой подпрограммы записывается в счетчик команд и выполнение программы продолжается.

Подгруппа D. Команды RET.

Команды машинного кода RET эквивалентны команде БЕЙСИК "RETURN". Имеются 9 команд в этой подгруппе, которые позволяют осуществить выход из подпрограммы без условий или в зависимости от состояния 4-х флагов.

Командами в этой подгруппе являются:

RET	C9
RET C	D8
RET NC	D0
RET Z	C8

```
RET NZ      C0
RET M       F8
RET P       F0
RET PE      E8
RET PO      E0
```

по команде RET адрес возврата из машинного стека побайтно восстанавливается в счетчике команд. При этом указатель стека дважды модифицируется в сторону увеличения.

Важно понимать, что восстановленный адрес возврата из машинного стека не обязательно тот, что был вначале помещен туда командой CALL.

Не совсем обычно при программировании на БЕЙСИКе модифицировать стек GOSUB , но при программировании в машинном коде это обычно имеет место.

Следующий пример показывает как это делается. Адрес программы выбирается из таблицы адресов и на эту программу осуществляется переход.

```
..... регистр A содержит входное значение
LD          D,+00           очистка регистра D
ADD         A,A             удвоение входного значения
LD          E,A             и пересылка в регистр E
LD          HL,+TABLE       HL указывает на таблицу
ADD         HL,DE           указатель на точку входа
LD          D,(HL)          получение старшего байта адреса
INC         HL
LD          E,(HL)          получение младшего байта адреса
PUSH DE     сохранение адреса в стеке
.....
RET         возврат
```

Подгруппа E. команды RST.

Последняя подгруппа команд в этой группе содержит специальную команду RST или команду "повторный пуск". Эти команды являются аналогичными командам CALL, но не требуют, чтобы адрес определялся.

```
RST 0000 C7 CALL 0000
RST         0008            CF              CALL 0008
RST         0010            D7              CALL 0010
RST         0018            DF              CALL 0018
RST         0020            E7              CALL 0020
RST         0028            EF              CALL 0028
RST         0030            F7              CALL 0030
RST         0038            FF              CALL 0038
```

в системе SPECTRUM восемь адресов "перезапуска" являются точками входа подпрограмм монитора и они будут рассмотрены в главе 7.

Группа 13. Команды сдвига.

Эти команды позволяют сдвигать на один бит вправо или влево содержимое регистров или байтов памяти. Разделим эти команды на 4 подгруппы:

- ⇨ команды логического сдвига
- ⇨ команды арифметического сдвига
- ⇨ команды ротации (циклического сдвига)
- ⇨ команды ротации полубайтов.

Подгруппа А. Команды логического сдвига.

Эти команды сдвигают содержимое указанной ячейки на один бит вправо. При этом в крайний левый бит записывается ноль, а "вытесненный" бит поступает во флаг переноса (если "вытесненный" бит равен 1, то флаг переноса устанавливается в 1; если "вытесненный бит равен 0, то флаг переноса сбрасывается в 0).

```
SRL A       CB 3F
SRL H       CB 3C
SRL L       CB 3D
SRL B       CB 38
SRL C       CB 39
SRL D       CB 3A
SRL E       CB 3B
SRL (HL)    CB 3E
SRL (IX+D)  DD CB      D 3E
SRL (IY+D)  FD CB      D 3E
```

Подгруппа В. Команды арифметического сдвига.

Команды арифметического сдвига сдвигают содержимое указанной ячейки на один бит вправо или влево, при этом "вытесненный" бит поступает во флаг переноса.

Рассмотрим команды арифметического сдвига вправо. Эти команды сдвигают содержимое указанной ячейки (регистра, байта в памяти) на один бит вправо, при этом крайний левый бит (бит знака для чисел в дополнительном двоичном коде) остается прежним, а крайний правый бит поступает во флаг переноса. Например, число 1100 1100 (CC 16-ричное) после арифметического сдвига вправо будет следующим: 11100110 (E6 16-ричное). Важно отметить, что арифметический сдвиг вправо эквивалентен делению на два (для чисел представленных в дополнительной двоичной форме), при этом флаг переноса свидетельствует о наличии остатка.

```
SRA A       CB 2F
SRA H       CB 2C
SRA L       CB 2D
SRA B       CB 28
SRA C       CB 29
```

```
SRA D          CB 2A
SRA E          CB 2B
SRA (HL)       CB 2E
SRA (IX+D)     DD CB          D 2E
SRA (IY+D)     FD CB          D 2E
```

рассмотрим команды арифметического сдвига влево. Эти команды сдвигают содержимое указанной ячейки (регистра, байта в памяти) на один бит влево. При этом в крайний правый бит записывается ноль, а "вытесненный" бит поступает во флаг переноса. Команда арифметического сдвига влево соответствует умножению на два, при этом флаг переноса свидетельствует о том, что результат больше 255.

Поскольку арифметический и логический сдвиги влево совпадают по результатам действия, то для логического сдвига влево не выделено своей команды.

```
SLA A          CB 27
SLA H          CB 24
SLA L          CB 25
SLA B          CB 20
SLA C          CB 21
SLA D          CB 22
SLA E          CB 23
SLA (HL)       CB 26
SLA (IX+D)     DD CB          D 26
SLA (IY+D)     FD CB          D 26
```

Подгруппа С. Команды ротации.

Команды ротации - это команды циклического сдвига вправо или влево. Рассмотрим эти команды.

Команды RL - ротация влево. В результате действия этих команд содержимое указанной ячейки сдвигается на один бит влево, "вытесненный" бит поступает во флаг переноса, а флаг переноса поступает в крайний правый бит ячейки. Таким образом, происходит циклический сдвиг влево через флаг переноса.

```
RL             A              CB 17
RL             H              CB 14
RL             L              CB 15
RL             B              CB 10
RL             C              CB 11
RL             D              CB 12
RL             E              CB 13
RL             (HL)           CB 16
RL             (IX+D)         DD CB          D 16
RL             (IY+D)         FD CB          D 16
```

команды RR - команды циклического сдвига вправо. В результате действия этих команд содержимое указанной ячейки сдвигается вправо, "вытесненный" бит (крайний правый бит) поступает во флаг переноса, а флаг переноса поступает в крайний левый бит указанной ячейки. Таким образом содержимое регистра или байта памяти циклически сдвигается вправо через флаг переноса.

```
RR      A       CB 1F
RR      H       CB 1C
RR      L       CB 1D
RR      B       CB 18
RR      C       CB 19
RR      D       CB 1A
RR      E       CB 1B
RR      (HL)    CB 1E
RR      (IX+D)  DD CB       D 1E
RR      (IY+D)  FD CB       D 1E
```

необходимо отметить, что при использовании команд RR и RL бит, "вытесненный" во флаг переноса, хранится там до тех пор, пока не будет изменен следующими командами, влияющими на флаг переноса.

Команда RRC и RLC - команды циклического сдвига вправо и влево соответственно. Эти команды, в отличии от команд RR и RL, сдвигают содержимое указанной ячейки на один бит вправо или влево минуя флаг переноса, хотя "вытесненный" бит поступает и в него. Например, команда RRC (HL) сдвигает содержимое ячейки памяти, адрес которой находится в паре регистров HL, на один бит вправо, при этом "вытесненный" бит (крайний правый) поступает в "освободившийся" бит (крайний левый). "Вытесненный" бит поступает также и во флаг переноса.

```
RLC A       CB 07
RLC H       CB 04
RLC L       CB 05
RLC B       CB 00
RLC C       CB 01
RLC D       CB 02
RLC E       CB 03
RLC (HL)    CB 06
RLC (IX+D)  DD CB       D 06
RLC (IY+D)  FD CB       D 06
RRC A       CB 0F
RRC H       CB 0C
RRC L       CB 0D
RRC B       CB 08
RRC C       CB 09
```

```
RRC D          CB 0A
RRC E          CB 0B
RRC (HL)       CB 0E
RRC (IX+D)     DD CB          D 0E
RRC (IY+D)     FD CB          D 0E
```

Подгруппа D. Команды ротации полубайтов.

В этой подгруппе всего две команды RRD и RLD. Указанные операции производятся над байтом, адрес которого находится в паре регистров HL, также в этих операциях участвует правый полубайт (4 разряда) регистра A (аккумулятора). Рассмотрим действия вышеуказанных команд.

Команда RRD производит следующие действия:

1. Левый полубайт ячейки, адрес которой находится в HL, поступает на место правого полубайта этой ячейки памяти.
2. "Вытесненный" правый полубайт ячейки памяти поступает в правый полубайт регистра A.
3. "Вытолкнутый" правый полубайт регистра A поступает на место левого полубайта ячейки памяти. Левый полубайт регистра A остается без изменений.

Команда RLD производит следующие действия:

1. Правый полубайт ячейки памяти, адрес которой находится в паре регистров HL, поступает на место левого полубайта этой ячейки.
2. "Вытесненный" левый полубайт ячейки памяти поступает в правый полубайт регистра A.
3. "Вытолкнутый" правый полубайт регистра A поступает на место правого полубайта ячейки памяти. Левый полубайт регистра A остается без изменений.

```
RRD            ED 67
RLD            ED 6F
```

кроме перечисленных 4-х групп есть еще одна дополнительная группа (пятая) команд ротаций аккумулятора (регистра A):

```
RLA            17          действия аналогичны команде RL a
RRA            1F          действия аналогичны команде RR a
RLCA           07          действия аналогичны команде RLC A
RRCA           0F          действия аналогичны команде RRC A
```

важно отметить, что все команды, кроме RRD и RLD влияют на флаг переноса. И все команды, кроме RLA, RRA, RLCA, RRCA влияют на флаг четности и знака.

Группа 14. Команды обработки битов.

Набор команд Z-80 имеет команды, которые позволяют определять, устанавливать и сбрасывать отдельные биты в пределах байта, помещенного в регистр или находящегося в памяти.

Подгруппа A. Команды BIT.

Они позволяют определять состояние определенного бита. Например: BIT 0,A - проверяет нулевой (крайний правый) бит регистра A. при использовании одной из этих команд обычным является процесс проверки, например, путем использования команды JP Z,ADDR.

Команды бит устанавливают флаг нуля, если проверяемый бит равен нулю, и наоборот.

Подгруппа B. Команды SET.

Эти команды позволяют устанавливать определенный бит в 1, при этом ни один из флагов не изменяется.

На практике команды используются для установки битов, но во многих случаях их использование лучше описывать как подтверждение, что определенный бит установлен. Команда SET, работающая с установленным битом, не будет иметь демонстрационного эффекта.

Подгруппа C. Команда RES.

Данная команда позволяет сбрасывать определенный бит в ноль. Флаги не меняются.

Команды этих трех подгрупп приведены в таблицах.

Команды BIT:

рег.	BIT 0	BIT 1	BIT 2	BIT 3	BIT 4	BIT 5	BIT 6	BIT 7
A	CB 47	CB 4F	CB 57	CB 5F	CB 67	CB 6F	CB 77	CB 7F
H	CB 44	CB 4C	CB 54	CB 5C	CB 64	CB 6C	CB 74	CB 7C
L	CB 45	CB 4D	CB 55	CB 5D	CB 65	CB 6D	CB 75	CB 7D
B	CB 40	CB 48	CB 50	CB 58	CB 60	CB 68	CB 70	CB 78
C	CB 41	CB 49	CB 51	CB 59	CB 61	CB 69	CB 71	CB 79
D	CB 42	CB 4A	CB 52	CB 5A	CB 62	CB 6A	CB 72	CB 7A
E	CB 43	CB 4B	CB 53	CB 5B	CB 63	CB 6B	CB 73	CB 7B
(HL)	CB 46	CB 4E	CB 56	CB 5E	CB 66	CB 6E	CB 74	CB 7E
(IX+D)	DD CB D 46	DD CB D 4E	DD CB D 56	DD CB D 5E	DD CB D 66	DD CB D 6E	DD CB D 76	DD CB D 7E
(IY+D)	FD CB D 46	FD CB D 4E	FD CB D 56	FD CB D 5E	FD CB D 66	FD CB D 6E	FD CB D 76	FD CB D 7E

Команды SET:

рег.	SET 0	SET 0	SET 0	SET 0	SET 0	SET 0	SET 0	SET 0
A	CB C7	CB CF	CB D7	CB DF	CB E7	CB EF	CB F7	CB FF
H	CB C4	CB CC	CB D4	CB DC	CB E4	CB EC	CB F4	CB FC
L	CB C5	CB CD	CB D5	CB DD	CB E5	CB ED	CB F5	CB FD
B	CB C0	CB C8	CB D0	CB D8	CB E0	CB E8	CB F0	CB F8
C	CB C1	CB C9	CB D1	CB D9	CB E1	CB E9	CB F1	CB F9
D	CB C2	CB CA	CB D2	CB DA	CB E2	CB EA	CB F2	CB FA
E	CB C3	CB CB	CB D3	CB DB	CB E3	CB EB	CB F3	CB FB
(HL)	CB C6	CB CE	CB D6	CB DE	CB E6	CB EE	CB F6	CB FE
(IX+D)	DD CB D C6	DD CB D CE	DD CB D D6	DD CB D DE	DD CB D E6	DD CB D EE	DD CB D F6	DD CB D FE
(IY+D)	FD CB D C6	FD CB D CE	FD CB D D6	FD CB D DE	FD CB D E6	FD CB D EE	FD CB D F6	FD CB D FE

Команды RES:

рег.	RES 0	RES 1	RES 2	RES 3	RES 4	RES 5	RES 6	RES 7
A	CB 87	CB 8F	CB 97	CB 9F	CB A7	CB AF	CB B7	CB BF
H	CB 84	CB 8C	CB 94	CB 9C	CB A4	CB AC	CB B4	CB BC
L	CB 85	CB 8D	CB 95	CB 9D	CB A5	CB AD	CB B5	CB BD
B	CB 80	CB 88	CB 90	CB 98	CB A0	CB A8	CB B0	CB B8
C	CB 81	CB 89	CB 91	CB 99	CB A1	CB A9	CB B1	CB B9
D	CB 82	CB 8A	CB 92	CB 9A	CB A2	CB AA	CB B2	CB BA
E	CB 83	CB 8B	CB 93	CB 9B	CB A3	CB AB	CB B3	CB BB
(HL)	CB 86	CB 8E	CB 96	CB 9E	CB A6	CB AE	CB B6	CB BE
(IX+D)	DD CB D 86	DD CB D 8E	DD CB D 96	DD CB D 9E	DD CB D A6	DD CB D AE	DD CB D B6	DD CB D BE
(IY+D)	FD CB D 86	FD CB D 8E	FD CB D 96	FD CB D 9E	FD CB D A6	FD CB D AE	FD CB D B6	FD CB D BE

Группа 15. Команды обработки блоков.

Имеется 8 команд обработки блоков. Эти команды очень интересны и очень полезны. Они позволяют перемещать блок данных из одной области памяти в другую или искать данные в области памяти.

Для перемещения блока данных перед выдачей соответствующей команды необходимо подготовить следующую информацию:

1. Адрес начала перемещаемого блока в паре регистров HL.
2. Адрес места назначения в паре регистров DE.
3. Количество перемещаемых байтов в паре регистров BC. для поиска заданной величины в блоке данных необходимо подготовить следующую информацию:

1. Заданная величина должна быть помещена в регистр A.
2. Адрес начала блока, в котором осуществляется поиск заданной величины в паре регистров HL.
3. Длина блока, в котором осуществляется поиск в паре регистров BC.

Команды группы:

автоматические

LDIR	ED B0
LDDR	ED B8
CPIR	ED B1
CPDR	ED B9

неавтоматические

LDI	ED A0
LDD	ED A8
CPI	ED A1
CPD	ED A9

как можно увидеть из выше приведенной таблицы имеются автоматические и неавтоматические команды. Автоматические команды более просты и боле полезны.

Автоматическая команда будет выполнять свою задачу до тех пор пока не будут выполнены все действия по поиску или пересылке. Выполнение автоматической команды может длиться разное время. Это время зависит от количества байтов, которые будут перемещаться.

Неавтоматическая команда передает только по одному байту, и требует, чтобы программист использовал команду повторно для дальнейшей пересылки или поиска. Время выполнения неавтоматической команды поэтому фиксировано.

Автоматическая команда представляет собой команду обработки блока, при выполнении которой адреса источника, места назначения информации

и значение счетчика байт - уменьшаются или увеличиваются микропроцессором.

Каждая команда этой группы будет обсуждаться по очереди.

LDIR.

Это самая распространенная команда из этой группы. Команда будет перемещать данные, адрес источника которых содержится в паре регистров HL, в область памяти с адреса, находящегося в паре регистров DE. Количество пересылаемых байт определено в паре регистров BC.

Команда работает следующим образом. Байт памяти, адрес которого содержится в паре регистров HL, пересылается в память по адресу, находящемуся в паре регистров DE.

Величина в паре регистров BC затем уменьшается, а величина в парах HL и DE увеличивается.

Если счетчик в регистровой паре BC еще не достиг 0, тогда будет пересылаться другой байт данных. Этот процесс продолжается пока счетчик не достигнет 0, но заметьте, что в этот момент пары регистров HL и DE достигают значения адресов после блока. Флаг переполнения/четности сбрасывается.

LDDR.

Эта команда является такой же как команда LDIR , за исключением того, что после пересылки каждого байта значения в парах регистров HL и DE уменьшаются. Поэтому команда требует, чтобы начальный (базовый) адрес блока соответствовал последнему байту блока. Место назначения информации также должно относиться к последнему байту выбранной области памяти.

CPIR.

Эта команда просматривает определенную область памяти для поиска первого наличия эталонной величины. Регистровая пара HL должна содержать базовый адрес, а BC - число байтов для исследования, регистр A - эталонную величину.

Команда работает следующим образом. Байт данных из памяти по адресу, находящемуся в паре регистров HL сравнивается с регистром A. Если они не совпали, то содержимое пары регистров HL увеличивается, а BC уменьшается и сравнивается следующий байт памяти с регистром A. Так происходит до тех пор, пока не произойдет сравнение байта памяти с содержимым регистра A или пока пара регистров BC не станет равной нулю.

Если содержимое байта и регистра A совпали, тогда флаг нуля устанавливается, а флаг знака сбрасывается, а регистровая пара HL содержит адрес выбранного байта. Величина в BC указывает как далеко от конца блока определено совпадение.

Если не совпали, то регистровая пара BC=0 , а флаги нуля, знака и переполнения/четности сбрасываются.

CPDR.

Работа этой команды похожа на CPIR, но блок данных просматривается сверху вниз (то есть содержимое HL уменьшается, а не увеличивается).

Далее рассматриваются неавтоматические команды.

LDI.

Действие этой команды такое же, как и у команды LDIR, но если команда LDIR выполняет перемещение блока целиком байт за байтом до тех пор, пока в регистре BC не будет достигнут ноль, то команда LDI перемещает только один байт. Если при этом в регистровой паре BC не достигнут ноль, то флаг переполнения/четности равен 1, если ноль достигнут, то флаг переполнения/четности равен 0 (сброшен). По результатам проверки этого флага программист может принять решение о продолжении действия команды LDI или прекращении. Одним словом, если команда LDIR выполняет перемещение каждого следующего байта автоматически, то LDI - не автоматически.

LDD.

Работает как LDI за исключением того, что величины в HL и DE уменьшаются.

CPI.

Команда аналогична CPIR, но в отличии от нее не является автоматической. Здесь после каждого сравнения очередного байта программист может принять решение о прекращении или продолжении поиска в зависимости от состояния флагов регистра F.

Если содержимое проверяемой ячейки и аккумулятора (регистра A) совпадут - включается флаг нуля, в противном случае он выключен. Если в процессе поиска нужный байт еще не найден, а содержимое пары регистров BC уменьшилось до нуля, то включается флаг переполнения/четности, в противном случае он выключен (сброшен).

CPD.

Эта команда подобна CPI, за исключением того, что регистровая пара HL уменьшается.

Группа 16. Команды ввода/вывода.

Периферийное устройство (принтер, клавиатура, магнитофон и так далее) подключается к компьютеру через специальное устройство, называемое "порт ввода/вывода" или просто "порт". Максимальное количество портов 65535, но реально их гораздо меньше. Для обмена информацией с периферийным устройством через порт существуют специальные команды.

Рассмотрим эти команды.

Подгруппа A. Команды простого ввода.

Эти команды принимают байт данных из указанного порта и помещают его в указанный регистр. Важно отметить, что адрес порта двухъбайтовая величина, но во многих ситуациях старший байт можно проигнорировать и не задавать вообще, так как он не влияет на выбор порта.

Командами этой подгруппы являются:

	адрес порта	регистр приемник	старший байт	младший байт
IN A,(+DD)	EB DD	A	A	DD
IN A,(C)	ED 78	A	B	C
IN H,(C)	ED 60	H	B	C
IN L,(C)	ED 68	L	B	C
IN B,(C)	ED 40	B	B	C
IN C,(C)	ED 48	C	B	C
IN D,(C)	ED 50	D	B	C
IN E,(C)	ED 58	E	B	C
IN F,(C)	ED 70	F	B	C

последняя команда служит для выставления флагов в регистре F без изменения содержимого остальных регистров.

Подгруппа B. Команды простого вывода.

Эти команды посылают байт из указанного регистра в выбранный порт. Адрес порта задается аналогично командам IN:

- ⇨ младшая часть адреса порта задается числом +DD или в регистре C.
- ⇨ старшая часть адреса задается в регистре A (для команды IN A,+DD) или в регистре B (для остальных команд).

Командами этой подгруппы являются:

	адрес порта	регистр приемник	старший байт	младший байт
OUT (+DD),A	D3 DD	A	A	DD
OUT (C),A	ED 79	A	B	C
OUT (C),H	ED 61	H	B	C
OUT (C),L	ED 69	L	B	C
OUT (C),B	ED 41	B	B	C
OUT (C),C	ED 49	C	B	C
OUT (C),D	ED 51	D	B	C
OUT (C),E	ED 59	E	B	C

Подгруппа C. Команды блочного ввода/вывода.

Эти команды требуют для своего выполнения адрес порта в регистре C (если нужно, то в регистре B содержится старшая часть адреса), количество передаваемых байт в регистре B, адрес области памяти в паре регистров HL.

среди этих команд есть автоматические и неавтоматические (аналогично командам LDIR, LDDR и LDI, LDD соответственно). Автоматические команды принимают или передают данные до тех пор пока регистр B не станет равным нулю. При этом адрес данных в HL увеличивается или уменьшается. Неавтоматические команды передают по одному байту аналогично командам LDI, LDD. При этом программист может принимать решение о продолжении обмена данными по состоянию флага нуля (он равен 1, если в регистре B окажется ноль).

```
INIR    ED B2 автомат. ввод, увеличивает HL
INDR    ED BA автомат. ввод, уменьшает HL
OTIR    ED B3 автомат. вывод, увеличивает HL
OTDR    ED BB автомат. вывод, уменьшает HL
INI     ED A2 неавтомат. ввод, увеличивает HL
IND     ED AA неавтомат. ввод, уменьшает HL
OUTI    ED A3 неавтомат. вывод, увеличивает HL
OUTD    ED AB неавтомат. вывод, уменьшает HL
```

Группа 17. команды прерывания.

Имеется 7 команд, связанных с прерываниями микропроцессора Z-80 . командами этой группы являются:

```
EI      FB
DI      F3
IM0     ED 46
IM1     ED 56
IM2     ED 5E
RETI    ED 5D
RETN    ED 45
```

каждая из команд будет рассмотрена поочередно.

EI.

При включении Z-80 система маскированного прерывания не может прервать выполнение команд. Эта ситуация существует до тех пор, пока прерывания не будут разрешены программистом по команде EI.

В SPECTRUM система маскируемых прерываний используется для отсчета текущего времени и опроса клавиатуры и эти прерывания происходят с частотой 50 гц.

DI.

В любом месте программы программист может запретить маскируемые прерывания по команде DI.

В SPECTRUM маскируемое прерывание не возможно во время выполнения команд SAVE, VERIFY, MERGE.

IM0.

Имеется три типа маскируемого прерывания. Режим 0 выбирается автоматически при включении или по команде IM0. Этот режим в SPECTRUM не используется.

IM1.

Режим 1 устанавливается по команде IM1 при инициализации компьютера (после включения питания). В этом режиме при возникновении маскируемого прерывания, управление передается подпрограмме по адресу 0038 (16-ричное). Это подпрограмма сканирования клавиатуры и изменения системного счетчика кадров (через каждые 20 миллисекунд значение системной переменной FRAMES (23762) увеличивается на единицу).

IM2.

Режим 2 не используется в SP, но является самым мощным из трех режимов. В нем периферийное устройство может указать, какая из 128 подпрограмм должна обрабатывать маскируемое прерывание. Содержимое регистра I и байт, который поступает из внешнего устройства используются вместе, чтобы образовать 16-битный адрес, который затем используется для выбора элемента из таблицы адресов, размещенной заранее в памяти. Двухъбайтовое число, выбранное из таблицы является начальным адресом подпрограммы обработки прерывания.

RETI.

Это специальная команда возврата для использования в программе обработки маскируемого прерывания. По этой команде происходит возврат с разрешением маскируемого прерывания.

RETN.

Эта команда подобна RETI, но она применима в конце программы обработки немаскируемого прерывания.

Группа 18. Дополнительные команды.

Имеется 7 команд, которые рассмотрены ниже, а именно:

```
CPL     2F
NEG     ED 44
SCF     37
CCF     3F
HALT    76
DAA     27
```

каждая из этих команд будет рассмотрена по очереди.

CPL.

Представляет собой простую команду, которая инвертирует регистр A, то есть она устанавливает сброшенный бит и сбрасывает установленный. Это называется дополнением до 1. основные флаги не изменяются.

NEG.

Эта команда инвертирует содержимое регистра A и прибавляет к нему единицу, что эквивалентно преобразованию положительного числа в отрицательное, записанное в двоичной дополнительной форме. Эта команда влияет на основные флаги. Флаги знака и 0 зависят от результата. Флаг переноса будет установлен, если регистр A первоначально был равен 0, в противном случае флаг переноса сбрасывается.

SCF.

Установка флага переноса.

CCF.

Переключение флага переноса на противоположенное значение.

HALT.

Это специальная команда, заставляющая микропроцессор останавливать выполнение команд, пока не возникнет маскируемое прерывание.

DAA.

Эта команда используется при работе с числами представленными в двоично-десятичном коде. В системе команд Z-80 нет специальных команд, которые учитывали бы правила сложения и вычитания чисел представленных в двоично-десятичном коде (BCD-арифметика). Поэтому двоично-десятичные числа складывают и вычитают с помощью имеющихся команд сложения и вычитания.(ADD, ADC, SUB, SBC), но после них результат корректируют с помощью команды DAA. Например:

```
LD          A,+DD          (+DD=77 16-ричное)
ADD A,+DD   (+DD=14 16-ричное)
```

DAA

после выполнения команды ADD в вышеприведенном примере, в регистре A будет находится число 8B (16-ричное). После выполнения команды DAA в регистре A будет находится число 91 (16-ричное), что соответствует правилам BCD-арифметики.

```
DAA         27
```

Важно отметить, что команда DAA преобразует только регистр A. флаги 0 и знака зависят от результата. Флаг переполнения/четности устанавливается как флаг четности. Флаг переноса зависит от того было ли переполнение или заем при операциях BCD.

6. Демонстрационные примеры программ в кодах.

Основная цель главы дать читателю сведения о начальных проблемах, связанных с написанием программ в машинных кодах. Команды Z-80 пере-

числены в главе 5, но не указывается как ими пользоваться на SPECTRUM. Настоящая глава содержит серию демонстрационных программ.

6.1. Запись машинных команд в память компьютера.

Имеется три точки зрения, подлежащие обсуждению.

1. Выбор области RAM.

Программист должен определить требуемый объем памяти. В SPECTRUM имеется несколько областей памяти, которые могут использоваться, но демонстрация программ в этой главе будет представлена в области RAM с адреса 32000 и дальше. Программа в кодах в этой области, если требуется, может сохраняться на ленте (SAVE) или загружаться с ленты (LOAD) как блок данных.

2. Ввод кодов программы.

В SPECTRUM можно вводить коды только по команде POKE. Однако операнд этой команды может быть задан как 10-ное или 2-ное число или как выражение. Поэтому следующие строки все равносильны:

```
10 POKE 32000,201
10 POKE 32000,BIN 1100 1001
10 LET       A=201: POKE 32000,A
```

и каждая может оказаться полезной в определенном случае.

Рекомендуемый метод заключается в написании команд в 16-ных символах. Следующий 16-ный загрузчик будет использоваться во всей главе !!!

```
1 REM HEXADECIMAL LOADER
5 CLEAR 31999
10 LET D=32000
20 DEF FN A(A$,B)=CODE A$(B)-48-7*(CODE A$(B)>57)
30 DEF FN C(A$)=16*FN A(A$,1)+FN A(A$,2)
40 READ A$: IF A$"**" THEN POKE D,FN C(A$): LET D=D+1: GO TO
40
```

загрузчик будет считывать данные из DATA , в котором каждая пара 16-ных символов заключена в кавычки и отделена от других пар запятыми. Пара символов "**" используется как символ конца.

Пример:

```
50 DATA "00","01","02","**"
```

если теперь получившуюся программу выполнить с помощью команды RUN, то это приведет к тому, что байт по адресу 32000 будет содержать 0, 32001 - 1, 32002 - 2.

3. Выполнение программы.

Команда БЕЙСИКа "USR число" позволяет остановить выполнение программы монитора и начать выполнять программу в кодах. Важно убедиться, что операнд "USR" установлен на требуемую команду, что программа пользователя кончается командой "RE- TURN", если необходимо вернуться в БЕЙСИК, и что величина возвращенной командой "USR" обработана соответствующим образом.

Примеры USR :

```
PRINT USR N - печатает содержимое регистровой пары BC
(10-тичное).
RANDOMIZE USR N - влияет на генератор случайных чисел.
LET A=USR N - изменяет переменную A.
```

Каждая из этих форм полезна при определенных условиях.

Следующие программы представлены в формате ассемблера и оператор DATA используется совместно с 16-ным загрузчиком. Команды представлены в том же порядке как и в главе 5.

6.2. Программы.

Группа 1. Команда "нет операции" (NOP).

Следующая программа ассемблера показывает простое использование команды NOP.

```
адрес     Маш. Код    обозначение    комментарии
7D00      00          NOP            NOP
7D01      C9          RET            возврат
```

следующая программа БЕЙСИК показывает строки, которые необходимо добавить к 16-ричному загрузчику для демонстрации команды NOP.

Программа 1. "NOP".

```
50 DATA "00","C9","**"
60 LET A=USR 32000
70 PRINT "YES"
```

Группа 2. Программы для непосредственной загрузки регистра.

Команда "USR число" возвращает значение регистровой пары BC как абсолютное 16-битовое число и команда PRINT отображает в 10-ном виде от 0 до 65535.

В следующем примере показаны команды "LD B,+DD" и "LD C,+DD". В регистры B и C загружаются константы.

```
7D00      06 00       LD B,+00       регистр B=0
7D02      0E 00       LD C,+XX       пользователь
```

```
                                вводит различные значения
7D04        C9                  RET
```

пользователь может задавать различные значения для регистра С.

Программа 2.

```
50 DATA "06","00","0E","00","C9","**"
60 INPUT "INPUT THE VALUE FOR REG. C (0-255)",N
70 CLS
80 POKE 32003,N
90 PRINT AT 10,0; "REG. C = "; USR 32000
100 GOTO 60
```

следующая программа показывает команду LD BC,+DDDD

```
7D00        01 00 00            LD BC,+0000     пользователь вводит
7D03        C9                  RET             различные значения
```

В программе 3 можно задавать различные величины "+DDDD". Введенная величина должна быть разбита на младший и старший байты прежде, чем она может быть заслана по POKE.

Программа 3. 'LD BC,+DDDD'

```
50 DATA "01","00","00"
51 DATA "C9","**"
60 INPUT "INPUT VALUE FOR REG. BC (0-65535)",N
70 CLS
80 POKE 32001,N-256*INT (N/256): POKE 32002,INT (N/256)
90 PRINT AT 10,0,"REG. BC = "; USR 32000
100 GOTO 60
```

Группа 3. Команды копирования и обмена.

Команды копирования регистр-регистр могут быть показаны загрузкой регистров, отличных от в и С, постоянными, которые затем копируются в регистры в и с для возврата их пользователю.

```
7D00        06 00               LD B,+00        регистр B=0
7D02        2E 00               LD L,+XX        вводится значение в
                                                регистр L
7D04        4D                  LD C,L          регистр C = регистр L
7D05        C9                  RET
```

В программе 4 переменная величина заносится в регистр L и пересылается в регистр С.

Программа 4. 'LD C,L'.

```
50 DATA "06","00","2E","00"
51 DATA "4D","C9","**"
60 INPUT "INPUT THE VALUE FOR REG. L (0-255)",N
70 CLS
80 POKE 32003,N
90 PRINT AT 10,0;"REG. C = ";USR 32000
100 GOTO 60
```

Группа 4. Команды загрузки регистров из памяти.

Команды в этой группе позволяют пользователю загрузить регистр содержимым памяти. Адресация может быть абсолютной, косвенной и индексной.

Первая программа показывает абсолютную адресацию. Задается адрес 60000, имеющий имя STORE и команда LD A, (ADDR) используется для выборки содержимого STORE в следующей программе.

```
7D00        06 00          LD B,+00          регистр B=0
7D02        3A 60 EA       LD A,(STORE)      выборка текущего
                                             значения
7D05        4F             LD C,A
7D06        C9             RET
```

В программе 5 пользователь может вводить различные значения, сохраняемые в STORE.

Программа 5. "LD A,(ADDR)".

```
50 DATA "06","00"
51 DATA "3A","60","EA"
52 DATA "4F","C9","**"
60 INPUT "ENTER a VALUE FOR LOCATION STORE (0-255 ONLY)",N
70 CLS
80 POKE 60000,N
90 PRINT AT 10,0;"THE LOCATION STORE NOW HOLDS"; CHR$ 32;
USR 32000
100 GOTO 60
```

Вторая программа показывает косвенную адресацию. В ней адрес STORE загружается в регистр HL до использования команды 'LD C,(HL)'.

```
STORE=EA60
7D00        06 00          LD B,+00          регистр B=0
7D02        21 60 EA       LD HL,+STORE      HL указывает на STORE
```

```
7D05        4E          LD C,(HL)     выборка текущего
                                      значения
7D06        C9          RET
```

В программе 6 пользователь может вводить различные величины сохраняемые в STORE.

Программа 6. 'LD C,(HL)'

```
50 DATA "06","00"
51 DATA "21","60","EA"
52 DATA "4E","C9","**"
```

строки 60-100 как в программе 5.

Третья программа показывает индексную адресацию. В этой программе адрес 59937 (EA21) называется BASE, а адрес 60000 (EA60) - STORE.

```
BASE=EA21
STORE=BASE+3F
```

```
7D00        06          LD B,+00      регистр B=0
7D02        DD 21 21 EA LD IX,BASE    установка IX
7D06        DD 4E 3F    LD C,(IX+3F)  выборка текущего
                                      значения
7D09        C9          RET
```

В программе 7 пользователь опять вводит различные значения, которые помещаются в STORE.

Программа 7. 'LD C,(IX+D)'.

```
50 DATA "06","00"
51 DATA "DD","21","21","EA"
52 DATA "DD","4E","3F"
53 DATA "C9","**"
```

строки 60-100 как в программе 5.

В программе используется регистр IX, но вполне возможно использование регистра IY, но при этом машинный код надо изменить заменив DD на FD.

Заметьте, что маскируемое прерывание потребует блокирования, в то время, пока IY содержит новую величину.

Группа 5. Команды загрузки памяти константами или данными, хранящимися в регистре.

Команды этой группы позволяют пользователю загрузить байты памяти константами или данными из регистров. Опять возможны абсолютная, косвенная и индексная адресация.

Первая программа показывает абсолютную адресацию. В программе абсолютный адрес для команды 60000, 16-ной EA60 названный 'STORE'. Используется команда 'LD (ADDR),A'.

```
STORE='EA60'
7D00        3E 00          LD A,+XX        ввод различных
                                           значений
7D02        32 60 EA       LD (STORE),A    ввод текущего
                                           значения в STORE
7D05        C9 RET
```

В программе 8 пользователь опять вводит различные значения в STORE

Программа 8. 'LD (ADDR),A'

```
50 DATA     "3E","00"
51 DATA     "32","60","EA"
52 DATA     "C9","**"
60 INPUT "ENTER a VALUE FOR LOCATION STORE (0-255 ONLY)",N
70 CLS
80 POKE     32001,N: RANDOMIZE USR 32000
90 PRINT AT 10,0;"THE LOCATION STORE NOW HOLDS"; CHR$ 32;
PEEK 60000
100 GOTO    50
```

Вторая программа показывает косвенную адресацию. В этой программе пара регистров HL указывает на STORE, а команда 'LD (HL),E' используется для записи в память.

```
STORE=EA60
7D00        1E 00          LD E,+XX        вводится значение
7D02        21 60 EA       7C LD HL,STORE  HL-указывает на STORE
7D05        73             LD (HL),E       пересылка
7D06        C9 RET
```

В программе 9 пользователь вводит величину, которая должна передаваться в STORE и обратно считываться с помощью PEEK 60000.

Программа 9. 'LD (HL),E'

```
50 DATA "1E","99"
51 DATA "21","60","EA"
52 DATA "73","C9","**"
```

строки 60-100 как в программе 8.

В 3-ей программе используется индексная адресация. В этом случае адрес 60063 (EA9F) называется BASE и STORE поэтому рассматривается как BASE-3F.

```
BASE=EA9F
STORE=BASE-3F
7D00        3E 00          LD A,+XX        вводится значение
7D02        DD 21 9F EA    LD IX,+BASE     IX указывает на BASE
7D06        DD 77 C1       LD (IX-3F),A    пересылка
(-3F представлено  в дополнительном двоичном коде (C1))
7D09        C9             RET
```

В программе 10 пользователь опять вводит величины, которые должны быть переданы в STORE и считывает обратно с помощью PEEK 60000.

Программа 10. 'LD (IX+D),A'

```
50 DATA "3E","00"
51 DATA "DD","21","9F","EA"
52 DATA "DD","77","C1"
53 DATA "C9","**"
```

Группа 6. Команды сложения.

Команды этой группы позволяют складывать величины вместе (ADD), увеличивать значение на единицу (INC) и складывать величины с учетом флага переноса (ADC).

Первая программа показывает использование команды 'ADD A,B'

```
7D00        00             NOP
7D01        3E 00          LD A,+XX        вводятся значения
7D03        06 00          LD B,+XX
7D05        80             ADD A,B         суммирование
7D06        06 00          LD B,+00        регистр B=0
7D08        4F             LD C,A          пересылка результата
7D09        C9             RET             в регистр C.
```

В программе 11 пользователь вводит 2 числа. Эти числа передаются в регистры а и в и складываются вместе в двоичной арифметике. Результат возвращается функцией USR.

Программа 11. 'ADD A,B'

```
50 DATA "00","3E","00"
51 DATA "06","00","80"
52 DATA "06","00","4F"
53 DATA "C9","**"
60 INPUT "ENTER a FIRST VALUE (0-255)";CHR$ 32;F
```

```
70 INPUT "ENTER a SECOND VALUE (0-255)";CHR$ 32;S
80 CLS
90 POKE 32002,F: POKE 32004,S
100 PRINT AT 10,5;F;CHR$ 32;"ADD";CHR$ 32;S;CHR$
32;"=";CHR$ 32;USR 32000
110 GOTO 60
```

Вторая программа для этой группы использует команду 'INC BC'

```
7D00 01 00 00          LD BC,+XXXX     вводятся значения
7D03 03      INC BC    значение увеличивается
7D04 C9      RET
```

В программе 12 пользователь вводит число в диапазоне 0-65535. это число затем делится на старшую и младшую части и помещается командой POKE в байты по адресам 7D01 и 7D02.

Регистровая пара BC затем увеличивается и значение возвращается функцией USR. Заметьте эффект ввода значения 65535.

Программа 12. "INC BC"

```
50 DATA     "01","00","00"
51 DATA     "03","C9","**"
60 INPUT "ENTER a VALUE (0-65535)";CHR$ 32;N
70 POKE     32002,INT (N/256)
80 POKE     32001,N-256*INT (N/256)
90 CLS
100 PRINT AT 10,0; CHR$ 32;"INCREMENTS TO GIVE"; CHR$ 32;
USR 32000
110 GOTO 60
```

Третья программа показывает использование команды "ADC", это та же программа, которая используется в программе 11, но изменена для включения команды "ADC A,B" а не "ADD A,B".

```
7D00      37          SCF  установка флага переноса
7D01      3E 00       LD A,+XX        вводятся значения
7D03      06 00       LD B,+XX
7D05      88          ADC A,B  суммирование с переносом
7D06      06 00       LD B,+00        регистр B=0
7D08      4F          LD C,A          регистр C=регистр A
7D09      C9          RET
```

Эта программа используется в программе 13, в которой пользователь складывает 2 числа вместе с битом переноса. Флаг переноса всегда устанавливается.

Используйте эффект замены команды "SCF" на "AND A" (16-ное A7) которая даст сброс флага переноса.

Программа 13. "ADC A,B".

```
50 DATA "37","3E","00"
51 DATA "06","00","88"
52 DATA "06","00","4F"
53 DATA "C9","**"
60 INPUT "ENTER a FIRST VALUE (0-255)";CHR$ 32;F
70 INPUT "ENTER a SECOND VALUE (0-255)";CHR$ 32;S
80 CLS
90 POKE 32002,F:POKE 32004,S
100 PRINT AT 10,0;"WITH GARRY SET";F;CHR$ 32;"ADC";CHR$ 32;
S;CHR$ 32;"=",CHR$ 32;USR 32000
110 GOTO 60
```

Группа 7. Команды вычитания.

Команды в этой группе позволяют вычесть одну величину из другой (SUB), уменьшить значение на единицу (DEC) и вычесть одну величину из другой с учетом флага переноса (SBC).

Первая программа для этой группы команд использует команду 'SUB B' . Состояние флага переноса определяется после вычитания и величины 0 или 1 сохраняются в STORE.

STORE=EA60

```
7D00    00        NOP будет использоваться дальше
7D01    3E 00     LD A,+XX вводятся значения
7D03    06 00     LD B,+XX
7D05    90        SUB B вычитание
7D06    06 00     LD B,+00 регистр B=0
7D08    4F        LD C,A пересылка результата
7D09    3E 00     LD A,+00 регистр A=0
7D0B    CE 00     ADC A,+00 сложение с переносом
7D0D    32 60 EA  LD (STORE),A передача значения
                  флага переноса в STORE
7D10    C9        RET
```

В программе 14 вышеприведенная программа вызывается после того, как пользователь вводит значения для регистров а и в. значение флага переноса получено использованием PEEK 60000.

Программа 14 'SUB B'

```
50 DATA      "00","3E","00"
51 DATA      "06","00","90"
52 DATA      "06","00","4F"
53 DATA      "3E","00","CE","00"
54 DATA      "32","60","EA"
55 DATA      "C9","**"
60 INPUT "ENTER a FIRST VALUE (0-255)";CHR$ 32;F
70 INPUT "ENTER a SECOND VALUE (0-255)";CHR$ 32;S
80 CLS
90 POKE 32002,F: POKE 32004,S
100 PRINT AT 10,0;F;CHR$ 32;"SUB";CHR$ 32;S;CHR$ 32;
"=";CHR$ 32;USR 32000;CHR$ 32;"WITH GARRY";CHR$ 32; "SET"
AND PEEK 60000;"RESET" AND NOT PEEK 60000
110 GOTO 60
```

Используйте приведенную программу с числами, которые меньше, больше и равны одно другому (GARRY - флаг переноса).

Вторая группа команд содержит команду DEC. команду "DEC BC" можно продемонстрировать, сделав соответствующие исправления в программе 12:

```
51 DATA "0B","C9","**"
100 PRINT AT 10,0;N;CHR$ 32;"DECREMENTS TO GIVE";CHR$ 32;
USR 32000
```

Третья подгруппа команд содержит команду SBC.

Команду SBC можно продемонстрировать, сделав соответствующие исправление в программе 14:

```
50 DATA "37","3E","00" для установки С-флага
  или
50 DATA "A7","3E","00"      для сброса С-флага
51 DATA "06","00","98" 'SBC A,B'  изменение SUB на SBC
```

Опять используйте эффект вычитания чисел, которые меньше, больше и равны друг другу.

Следует отметить, что нет разницы в результатах между SUB и SBC при сброшенном флаге переноса.

Группа 8. Команды сравнения.

Команды сравнения по существу такие же, как команды вычитания, исключая тот факт, что регистр а не изменяется. Флаг переноса изменяется и в последствие может проверяться.

Действие команда "CP B" может быть показано, сделав соответствующие изменения в программе 14:

```
51 DATA "06","00","B8"
100 PRINT AT 10,0;F;CHR$ 32;"CP";CHR$ 32;S;CHR$ 32;"GIVE
GARRY";CHR$ 32; и добавить строки
101 RANDOMIZE USR 32000
102 PRINT "RESET" AND NOT PEEK 60000; "SET" AND PEEK 60000
```

Группа 9. Логические команды.

Команды в этой группе позволяют осуществлять логические операции AND,OR,XOR над двумя 8-битовыми числами.

Следующая программа показывает команду "AND B", которая используется в логических операциях над двумя значениями, вводимыми пользователем.

```
7D00 3E 00   LD A,+XX
7D02 06 00   LD B,+XX      вводятся значения
7D04 A0      AND B         логическое "и"
7D05 32 60 EA
7D08 C9      RET           LD (STORE),A   результат в STORE
```

В программе 15 используется вышеприведенная программа. Пользователю предлагается ввести 2 значения.

При вводе каждого значения оно печатается в двоичной форме. Результат считывается из STORE с использованием PEEK 60000, и также печатается в двоичной форме.

Программа 15. 'AND B'

```
50 DATA      "3E","00"
51 DATA      "06","00"
52 DATA      "A0"
53 DATA      "32","60","EA"
54 DATA      "C9"
60 INPUT "ENTER a FIRST VALUE (0-255)";CHR$ 32;F
70 CLS
80 POKE 32001,F
90 PRINT AT 8,4;: GO SUB 300
```

```
100 PRINT AT 10,8;"AND"
110 INPUT "ENTER a SECOND VALUE (0-255)";CHR$ 32;S
120 POKE 32003,6
130 PRINT AT 12,4;: LET F=S: GO SUB 300
140 PRINT AT 14,7;"GIVES"
150 RANDOMIZE USR 32000
160 PRINT AT 16,4;:LET F=PEEK 60000: GO SUB 300
170 GOTO 60
300 LET P=256
310 FOR N=7 TO 1 STEP -1
320 LET P=P/2
330 PRINT CHR$(48+INT (F/P));
340 LET F=F-p*INT (F/P)
350 NEXT N
360 PRINT INT F
370 RETURN
```

Программа 15 может быть применена для демонстрации команд 'OR B' и 'XOR B' для 'OR B' требуется внести изменения в строки 52 и 100 следующим образом:

```
52 DATA "B0"
100 PRINT AT 10,8;"OR"
```

и для 'XOR B'

```
52 DATA "A8"
100 PRINT AT 10,8;"XOR"
```

Группа 10 . Команды переходов.

Семнадцать команд этой группы позволяют сделать переходы из одной части программы в другую. Переходы могут быть относительными от -128 до +127 от значения счетчика команд, или абсолютными т. Е. задается адрес перехода. Переход может быть сделан безусловно или условно в зависимости от состояния основных флагов, хотя только абсолютные команды перехода имеют полный диапазон имеющихся возможностей.

Следующая программа будет использоваться в программе 16, чтобы показать команды этой группы.

```
NEXT=7D0E

7D00        3E 00        LD A,+XX        вводятся 2 значения
                                         и сравниваются
```

```
7D02        FE 00          CP          +XX между собой
7D04        01 00 00       LD BC,+0000 BC=0
7D07        18 06          JR NEXT     переход на NEXT
7D09        00             NOP         для дальнейшего
                                       использования
7D0A        C9             RET         выход если
                                       не было перехода
7D0B        00             NOP         не используется
7D0C        00             NOP         не используется
7D0D        00             NOP         не используется
7D0E        00             NOP         не используется
7D0F        03             INC BC      увеличение
регистровой пары BC, если был переход
7D10        C9             RET         выход
```

В приведенной программе в паре регистров BC будет возвращено значение 0, если не делается переход и 1 если делается переход.

Программа 16 приведена для демонстрации команды 'JR E'. Эта команда является безусловной, так что переход будет вне зависимости от результата сравнения величин введенных пользователем.

Программа 16. 'JR E'

```
50 DATA      "3E","00"
51 DATA      "FE","00"
52 DATA      "01","00","00"
53 DATA      "18","06","00"
54 DATA      "C9"
55 DATA      "00","00","00","00"
56 DATA      "03","C9","**"
60 PRINT AT 4,0;"COMMAND";
70 PRINT "JR E"
80 PRINT AT 8,0;
90 INPUT "ENTER a FIRST VALUE (0-255)";CHR$ 32;F
100 PRINT F;CHR$ 32;"CP";CHR$ 32;
110 INPUT "ENTER a SECOND VALUE (0-255)";CHR$ 32;S
120 PRINT S
130 PRINT AT 12,0;"JUMP - ";
140 POKE 32001,F: POKE 32003,S
150 LET R=USR 32000: PRINT "NO" AND NOT R;"YES" AND R
```

```
160 PRINT AT 21,0;"PRESS ANY KEY TO CONTINUE"
170 LET A=CODE INKEY$
180 IF A=0 THEN GOTO 170
190 CLS: GOTO 60
```

В программе 16 вводятся две величины, выполняется операция сравнения, а затем, если необходимо, переход. Переменная R будет равна 0, если перехода не было и 1, если был переход.

Программа написана так, что пользователь может путем замены строк 53 и 70 продемонстрировать 14 из 17 команд переходов.

Для команд относительного перехода изменениями будут следующими:

```
53 DATA "20","06","00"      - JR NZ,E
70 PRINT "JR NZ,E"
53 DATA "28","06","00"      - JR Z,E
70 PRINT "JR Z,E"
53 DATA "30","06","00"      - JR NC,E
70 PRINT "JR NC,E"
53 DATA "38","06","00"      - JR C,E
70 PRINT "JR C,E"
```

Команды абсолютного перехода показаны изменением программы так, что абсолютный переход осуществляется по адресу NEXT. Строка команды по адресу 7D07 будет теперь читаться следующим образом:

```
7D07 C3 0F 7D JP NEXT
```

Изменения в программе 16:

```
53 DATA      "C3","0F","7D"        - JP ADDR
70 PRINT "JP ADDR"
53 DATA "C2","0F","7D"     - JP NZ,ADDR
70 PRINT "JP NZ,ADDR"
53 DATA "CA","0F","7D"     - JP Z,ADDR
70 PRINT "JP Z,ADDR"
53 DATA "D2","0F","7D"     - JP NC,ADDR
70 PRINT "JP NC,ADDR"
53 DATA "DA","0F","7D"     - JP C,ADDR
70 PRINT "JP C,ADDR"
53 DATA "E2","0F","7D"     - JP PO,ADDR
70 PRINT "JP PO,ADDR"
53 DATA "EA","0F","7D"     - JP PE,ADDR
70 PRINT "JP PE,ADDR"
```

```
53 DATA "F2","0F","7D"      - JP P,ADDR
70 PRINT "JP P,ADDR"
53 DATA "FA","0F","7D"      - JP M,ADDR
70 PRINT "JP M,ADDR"
```

остальные 3 команды используют косвенную адресацию и они остаются как упражнение для заинтересованного читателя.

Группа 11. Команда DJNZ E.

Эта команда является очень полезной командой и может легко использоваться для получения простых циклов в программе, написанной в машинных кодах.

Чтобы использовать команду программист сначала должен определить число требуемых циклов и это число должно быть занесено в регистр B.

Следующие 3 строки программы БЕЙСИК будут печатать алфавит заглавными буквами.

```
10 FOR A=65 TO 90
20 PRINT CHR$ A;
30 NEXT A
```

Следующая программа машинного кода показывает выполнение этой же программы. Обратите внимание, что программист может использовать только уменьшение на 1.

```
LOOP=7D02

7D00 06 1A   LD B,+1A      задание счетчика циклов
7D02 3E 5B   LD A,+5B
7D04 90      SUB B         регистр A содержит значение
7D05 D7      RST 0010      печать символа
7D06 10 FA   DJNZ LOOP     следующая буква
7D08 C9      RET
```

Программа 17 использует выше приведенную программу для печати алфавита на экране телевизора.

Программа 17.

```
50 DATA "06","1A"
51 DATA "3E","5B"
52 DATA "90","D7"
53 DATA "10","FA"
55 DATA "C9","**"
```

```
60 PRINT
70 RANDOMIZE USR 32000
```

примечание: включение строки 60 важно, т.к. она открывает канал для основной области экрана.

Если эта команда или подобная ей опускается тогда печать произойдет в нижней части экрана.

Группа 12. Команды стека.

Имеется пять подгрупп в этой группе. Первая содержит команды PUSH и POP, а вторая команды обмена со стеком. Следующая программа используется в программе 18 чтобы показать команды из первой подгруппы.

```
7D00    21 00 00 LD HL,+XXXX    вводятся значения
7D03    E5       PUSH HL        сохраняются в стеке
7D04    C1       POP BC         пересылаются в BC
7D05    C9       RET
```

В программе 18 пользователю будет предложено ввести величину. Эта величина затем передается в регистровую пару HL. потом она сохраняется в машинном стеке и далее данные из стека поступают в регистровую пару BC.

Программа 18. "PUSH HL" и "POP BC".

```
50 DATA "21","00","00"
51 DATA "E5","C1"
52 DATA "C9","**"
60 INPUT "ENTER a VALUE (0-65535)";CHR$ 32;F
70 POKE 32001,F-INT (F/256)*256: POKE 32002,INT (F/256)
80 CLS
90 PRINT AT 10,0;"VALUE TAKEN OFF STACK =";CHR$ 32;USR 32000
100 GOTO 60
```

читателю предлагается использовать альтернативные процедуры в этой программе. Но следует быть уверенным что число команд PUSH совпадает с POP.

3 и 4 подгруппы содержат команды CALL и RET. следующая программа в машинных кодах может использоваться для демонстрации различных команд CALL.

```
NEXT=7D0F
7D00 3E 00   LD A,+XX      вводятся 2 значения и сравниваются
7D02 FE 00   CP +XX
7D04 01 00 00              LD BC,+0000     BC=0
7D07 CD 0F 7D              CALL NEXT       вызов подпрограммы
7D0A C9      RET           возврат в БЕЙСИК
7D0B 00      NOP
```

```
7D0C 00     NOP
7D0D 00     NOP
7D0E 00     NOP
7D0F 03     INC BC          увеличение BC
7D10 C9     RET             возврат из подпрограммы
```

эта процедура может использоваться в программе 16 с последующими изменениями для различных команд

для всех команд - 130 PRINT AT 12,0;"CALL - ";

```
CALL ADDR   -               53 DATA "CD","0F","7D"
70 PRINT "CALL ADDR"
CALL NZ,ADDR                - 53 DATA "C4","0F","7D"
70 PRINT "CALL NZ,ADDR"
CALL Z,ADDR -               53 DATA "CC","0F","7D"
70 PRINT "CALL Z,ADDR"
CALL NC,ADDR                - 53 DATA "D4","0F","7D"
70 PRINT "CALL NC,ADDR"
CALL C,ADDR -               53 DATA "DC","0F","7D"
70 PRINT "CALL C,ADDR"
CALL PO,ADDR                - 53 DATA "E4","0F","7D"
70 PRINT "CALL PO,ADDR"
CALL PE,ADDR                - 53 DATA "EC","0F","7D"
70 PRINT "CALL PE,ADDR"
CALL P,ADDR -               53 DATA "F4","0F","7D"
70 PRINT "CALL P,ADDR"
CALL M,ADDR -               53 DATA "FC","0F","7D"
70 PRINT "CALL M,ADDR"
```

Следующая программ машинного кода может использоваться для демонстрации команд RET. однако она составлена для тренировки читателя, чтобы он сделал необходимые исправления в программе 16.

```
7D00    3E 00       LD A,+XX        вводятся 2 значения и
7D02    FE 0D       CP +XX          сравниваются
7D04    01 01 00    LD BC,+0001     BC=1
7D07    D8          RET C           возврат, если флаг C=1
7D08    0B          DEC BC          уменьшение BC
7D09    C9          RET
```

Группа 13. Команды сдвига.

Имеется большое количество команд сдвига. В программе 19 используется следующая процедура, чтобы продемонстрировать семь типов сдвига в качестве примера выбран регистр C.

STORE=EA60

```
7D00    AF          XOR             очистка регистра A
7D01    FE 00       CP +XX          сравнение с нулем
```

```
7D03    06 00       LD B,+00        регистр B=0
7D05    0E 00       LD C,+XX        вводятся значения
7D07    CB 01       RLC             C   сдвиг
7D09    3E 00       LD A,+00        регистр A=0
7D0B    ce 00       ADC A,+00       пересылка состояния
7D0D    32 60 EA    LD (STORE),A    флага C в STORE
7D10    C9          RET
```

В программе 19 пользователю предлагается ввести 0 или 1 чтобы сбросить или установить флаг переноса.

Следующая вводимая величина передается в регистр C и сдвигается как требуется. Результат печатается совместно с текущим значением флага переноса.

Программа 19. 'RLC C'

```
50 DATA "AF","FE","00"
51 DATA "06","00","0E","00"
52 DATA "CB","01"
53 DATA "3E","00","CE","00"
54 DATA "32","60","EA"
55 DATA "C9","**"
60 PRINT AT 2,0;"INSTRUCTION      ";
70 PRINT "RLC C"
80 INPUT "GARRY RESET OR SET ? (0/1)";CHR$ 32;C
90 POKE 32002,C
100 PRINT AT 6,0;"GARRY - ";C
110 INPUT "ENTER a VALUE (0-255)";CHR$ 32;F
120 POKE 32006,F
130 PRINT AT 10,0;"INITIAL VALUE ";:GO SUB 300
140 LET F=USR 32000
150 PRINT AT 14,0;"FINAL VALUE ";: GO SUB 300
160 PRINT AT 18,0;"GARRY - ",PEEK 60000
170 PRINT AT 21,0;"PRESS ANY KEY TO CONTINUE"
180 PAUSE 0
200 CLS: GOTO 60
300 LET P=256
310 FOR N=7 TO 1 STEP -1
320 LET P=P/2
```

```
330 PRINT CHR$ (48+INT(F/P));CHR$ 32;
340 LET F=F-INT(F/P)*P
350 NEXT N
360 PRINT INT F
370 RETURN
```

Программа 19 может быть использована для демонстрации 7-ми команд, которые используют регистр C со следующими изменениями:

```
RLC C - 52 DATA "CB","01"
70 PRINT "RLC C"
RRC C       - 52 DATA "CB","09"
70 PRINT "RRC C"
RL          C              - 52 DATA "CB","11"
70 PRINT "RL               C"
RR          C              - 52 DATA "CB","19"
70 PRINT "RR               C"
SLA C       - 52 DATA "CB","21"
70 PRINT "SLA C"
SRA C       - 52 DATA "CB","29"
70 PRINT "SRA C"
SRL C       - 52 DATA "CB","39"
70 PRINT "SRL C"
```

Группа 14. Команды обработки битов.

Команды этой группы могут делиться на три подгруппы. Команды RET и SET не очень часто используются, и здесь не приводятся программы БЕЙСИКа, чтобы показать их использование.

Читатель однако, может попробовать написать простые программы, которые используют эти команды.

Команды BIT являются самыми полезными в этой группе и следующая программа показывает как используется команда 'BIT 7,H'.

Программа 20 является программой печати чисел в двоичной форме.

Программа представлена следующим образом:

```
LOOP=7D05
PRINT=7D0C
7D00 21 00 00      LD HL,+XXXX    вводятся значения
7D03 06 10   LD B,+10       счетчик цикла на 16 бит
7D05 CB 7C   BIT 7,H проверяется левый байт
7D07 3E 30   LD A,'0'       подготовка для печати 0
7D09 28 01   JR Z,PRINT     переход, если символ 0
7D0B 3C      INC A   символ 1
```

```
7D0C D7      RST 0010      печать 0 или 1
7D0D 3E 20   LD A,*20      печать пробела
7D0F D7      RST 0010
7D10 CB 15   RL L          сдвиг L
7D12 CB 14   RL H          сдвиг H
7D14 10 EF   DJNZ LOOP     повторить
7D16 C9      RET
```

Программа 20 показывает используемую процедуру.

Программа 20. 'BIT 7,H'

```
50 DATA "21","00","00"
51 DATA "06","10","CB","7C"
52 DATA "3E","30","28","01"
53 DATA "3C","D7"
54 DATA "3E","20","D7"
55 DATA "CB","15","CB","14"
56 DATA "10","EF","C9","**"
60 INPUT "ENTER a VALUE (0-65535)"; CHR$ 32;F
70 POKE 32001,F-INT(F/256)*256
80 POKE 32002,INT(F/256)
90 CLS
100 PRINT AT 10,8;"BINARY OF";CHR$ 32;F
110 PRINT AT 12,14;"IS"
120 PRINT AT 14,10;
130 RANDOMIZE USR 32000
140 GOTO 60
```

Группа 15. Команды обработки блоков.

Команды этой группы позволяют перемещать блоки данных или осуществлять поиск в блоке данных.

Из всех команд в группе LDIR будет несомненно самой распространенной.

Следующая программа использует команду LDIR для копирования верхней трети области экрана в область средней трети. Это означает, что когда программа вызывается все байты из экранной области по адресу 4000-47FF копируются командой LDIR в экранную область по адресу 4800-4FFF.

Пользователь может увидеть, что это произошло, так как все символы, которые были в строках 0-7 будут продублированы в строки 8-15.

Программа представлена следующим образом:

```
7D00    21 00 40    LD HL,+4000    первая строка экрана
7D03    11 00 48    LD DE,+4800    восьмая строка экрана
7D06    01 00 08    LD BC,+0800    2048 байт
7D09    ED B0       LDIR           пересылка блока
7D0B    C9          RET            возврат
```

Программа 21 использует эту процедуру.

Программа 21. 'LDIR'.

```
50 DATA "21","00","40"
51 DATA "11","00","48"
52 DATA "01","00","08"
53 DATA "ED","B0"
54 DATA "C9","**"
60 INPUT "ENTER YOUR CHARACTERS - 8 LINES";C$
70 PRINT C$
80 RANDOMIZE USR 32000
```

Нижеследующая программа демонстрирует команду CPIR.

Эта программа ищет заданную величину в ROM-области. Заданная величина изменяется в от 0 до 255. Например число 154 впервые встречается в ROM-области по адресу 11728.

Программа имеет вид:

```
7D00    3E 00       LD A,+XX       ввод образца
7D02    01 FF 3F    LD BC,+3FFF    старший байт ROM
7D05    21 00 00    LD HL,+0000    первый байт ROM
7D08    ED B1       CPIR           поиск в ROM
7D0A    44          LD B,H         пересылка адреса
7D0B    4D          LD C,L         байта
7D0C    0B          DEC BC         указатель на искомый
7D0D    C9          RET            байт
```

Программа 22. 'CPIR'.

```
50 DATA "3E","00"
51 DATA "01","FF","3F"
52 DATA "21","00","00"
53 DATA "ED","B1","44","4D"
54 DATA "0B","C9","**"
```

```
60 FOR F=0 TO 255
70 POKE 32001,F
80 PRINT F; TAB 4; "OCCUSR FIRST AT LOC."; CHR$ 32; USR 32000
90 NEXT F
```

Читатель может переписать эти программы, используя не автоматические команды.

Здесь не приводятся программы для команд ввода-вывода, команд прерываний и небольшой группы смешанных команд, но читатель может написать их сам.

7. Монитор 16к.

Система SPECTRUM снабжена 16к постоянной памяти (ROM) включающей в себя:

1. Операционную систему.
2. Интерпретатор БЕЙСИК.
3. Набор 96 литер.

Содержимое ROM-области, включающее в себя вышеперечисленные компоненты называют монитором.

Монитор занимает область памяти начиная с адреса 0000 до 3FFF (0-16384) и в стандартной системе SPECTRUM не может быть перемещен из этих адресов. Команда расположенная по 0-му адресу является первой командой, выполняемой после включения питания.

Программа монитора заслуживает изучения по следующим причинам:

1. Подпрограммы монитора всегда доступны для вызова из программ в машинных кодах, написанных пользователем. Это позволяет создавать короткие программы.
2. Программа монитора показывает, как фирма решила определенные проблемы.
3. Программа монитора - это программа в машинных кодах больших размеров и поучительно посмотреть, как можно структурировать большую программу.

Разные части монитора теперь будут обсуждены по очереди, сперва используя системный взгляд, а затем каждую в отдельности.

7.1. Системный взгляд на монитор.

Пользователь системы обычно не подозревает, что МП (микропроцессор) в центре системы работает по программе в машинных кодах содержащейся в 16 к ROM с того момента когда питание подается к системе. Единственным исключением из этого бывает выполнение программы в кодах, написанных пользователем.

Пользователю SPECTRUM представляется как машина, которая ожидает программную строку БЕЙСИКа либо в непосредственном режиме, либо с номером для построения БЕЙСИК программы. Строки БЕЙСИК без номера выполняются немедленно и могут использовать интерпретацию строк из заранее введенной программы. Система SPECTRUM немного сложна, поскольку интерпретатор вызывает проверку синтаксиса строки прежде чем записывает ее в программную область - вне зависимости от того выполняется она непосредственно или представляет элемент программы.

Операционная система (ОС).

Изучение программы монитора начинается с программы INITIALIZATION, которая запускается либо после включения питания, либо по переходу к 0-му адресу.

INITIALIZATION (инициализация).

Эта программа занимает адрес 0000-0007 и 11CB-12A1. основными задачами являются проверка наличия памяти и установка системных переменных. Более подробно это будет рассмотрено позднее. Программа инициализации заканчивается появлением сообщения SINCLAIR. За этим стартует программа MAIN EXECUTION. Эта программа занимает адреса 12A2-15AE. в системе SPECTRUM эта программа, как следует из названия выполняет основные функции в мониторе.

Эта программа вызывает (если необходимо) программы LIST EDITOR и SYNTAX CHECKER строка БЕЙСИК добавляется в программную область, в случае непосредственного выполнения строки вызывается программа LINE-RUN и эта строка интерпретируется - это может включать в себя интерпретацию других строк БЕЙСИК. Затем по возвращению из программы LINE-RUN выдается сообщение. Таблица сообщений об ошибках находится по адресам 1391-1536. возврат к MAIN EXECUTION приводит к новому вызову программы EDITOR.

EDITOR (редактор).

Программа занимает адреса 0F2C-10A7.

Программа позволяет пользователю поместить строку БЕЙСИКа в нижней части экрана. В действительности строка формируется в области редактирования и затем копируется в дисплейную область.

Пользователь может вводить символы или управлять курсором и соответствующие подпрограммы вызываются по мере необходимости.

Управление курсором и ENTER вызывают возврат в программу MAIN EXECUTION.

В стандартной системе SPECTRUM войти в программу EDITOR можно только через клавиатуру и поэтому программа KEYBOARD-INPUT вызывает программу EDITOR .

KEYBOARD-INPUT.

Программа занимает адреса 10A8-111C.

Эта программа дает код последней набранной на клавиатуре клавиши, читая его из системной переменной LAST-K.

Эта программа выполняет некоторые операции, например, устанавливает флаги CAPS LOCK и другие и получает 2-й байт управляющей клавиши цвета для системной переменной.

KEYBOARD.

Каждые 1/50 сек. прерываются вычисления и производится сканирование клавиатуры. Существует 5 отдельных программ связанных с основной программой KEYBOARD, занимающей адреса 02BF-030F. Собственно сканирование клавиатуры выполняет подпрограмма KEY-SCAN, которая занимает адреса 028E-02BE. Подпрограмма возвращает соответствующее ключевое значение в регистровой паре DE , которое другие программы клавиатуры используют для получения кода литеры.

PRINT-OUTPUT.

В добавление к программам, обсужденным выше, имеется много других, которые также составляют часть ОС.

Программа PRINT-OUTPUT располагается в адресах 09F4-0D4C и возможно является самой важной среди других программ. Эта программа вызывается командой RST 0010.

В системе SPECTRUM программа PRINT-OUTPUT очень мощная, поскольку и символы и управляющие коды выводятся с помощью этой же программы.

В системе SPECTRUM каждый символ, который печатается на TV-экране или принтере, формируется программой PRINT-OUTPUT. тот факт, что программа применяется во многих различных случаях, делает программу довольно медленной, но несмотря на это и очень полезной.

Интерпретатор БЕЙСИК.

Интерпретатор вызывается как для проверки синтаксиса, так и для выполнения строки.

Например, в строке LET A=1 сначала проверяется синтаксис, а затем переменной а присваивается значение 1. поэтому в таких случаях может быть удобно рассматривать синтаксический анализатор как сумму отдельных функций исполнителя строк, несмотря на то, что реально этого нет.

Интерпретатор состоит из нескольких частей:

1. Таблица команд:

располагается в области 1A48-1B16. в системе SPECTRUM существует 50 команд. Таблица команд содержит класс команд, символы разделители и адреса программ выполнения команды;

2. Управляющая программа:

располагается в адресах 1B17-1CEE и содержит команды, позволяющие рассматривать операторы БЕЙСИК один за другим, как требуется в программе.

Точкой входа в синтаксический анализатор является адрес 1B17, а в исполнитель строки 1B0A;

3. Программы класса команды:

располагаются в адресах 1C01-1CDD и главным образом занимаются анализом параметров команды БЕЙСИК. Например, команда NEXT относится к классу 4, как требующая за собой простую переменную. Команда POKE относится к классу 8, как требующая за собой 2-а числовых выражения, разделенных запятой;

4. Программы команд:

большинство программ команд располагаются в адресах 1CDE-24FA. программы команд, связанные с процедурами ввода-вывода, обычно входят в ос как часть программы монитора.

Программы команд существуют для каждой из 50 команд БЕЙСИК и выполнение этих программ и есть собственно работа интерпретатора. Интерпретация операторов БЕЙСИК может быть проиллюстрирована следующими примерами:

рассмотрим оператор 10 CLS, интерпретация которого проста. Сначала рассматривается команда не имеющая операндов, по таблице команд находится адрес (0D6B).

Программа команды выполняется, вызывая очистку экрана дисплея и установку атрибутов.

Программа управления затем переходит к интерпретации следующего оператора.

Преобразователь выражений:

эта программа занимает адреса 24F8-28B1. в SPECTRUM результат преобразования выражения может быть или числовой или строковой.

Числовой результат возвращается как плавающее значение в 5-ти верхних байтах калькуляторного стека, в случае строкового результата эти же 5 байтов представляют собой набор параметров определяющих строку.

Выражение преобразуется слева направо. Операция, имеющая высший приоритет, выполняется перед операцией с подобным или низшим приоритетом. Определенные операции такие как FN,RND,PI,INKEY$,BIN,SCREEN$, ATTR,POINT выполняются без использования преобразователя выражений, но для всех других он используется.

Когда переменная используется в выражении, преобразователь выражений получает соответствующее значение для переменной, вызывая программу идентификации этой переменной.

Программы переменных.

Этот набор программ занимает адреса 28B2-2ACB.

Эти программы возвращают текущее значение или параметры заданной переменной в область переменной.

Смешанные арифметические программы.

Часть программы монитора, которая занимает адреса 2C88- 2F9A, содержит серию арифметических программ. Самое важная из них STACK-BC (по адресу 2D2B), которая преобразует текущее значение, содержащееся в регистровой паре BC, в плавающее значение наверху калькуляторного стека, и PRINT-FP (по адресу 2DE3) которая берет верхнее значение калькуляторного стека, преобразует его в 10-ную форму и печатает на TV-экране или принтере.

Калькулятор.

Это большая и сложная программа, занимающая адреса 2F9B-2FCD . Обычно выполняется по команде RST 0028 , в свою очередь выполняющую переход по адресу 3358. по существу калькулятор состоит из 66 подпрограмм, каждая из которых выполняет различные действия. Вызов этих подпрограмм осуществляется не по команде 'CALL', а использованием литералов с 16-ным значением 00-41 (16-ричное).

Например литерал 04 эквивалентен CALL 30CA и приводит к тому, что два верхних значения калькуляторного стека заменяют одиночным значением, которое является результатом умножения этих значений. Следовательно 04 является литералом, обеспечивающим умножение. Эти литералы включают в программу как байты данных (DEFB-определение байтов), следующие за командой RST 0028. последний DEFB всегда имеет значение 38, которое выполняет завершающие операции и таким образом является возвратом из калькулятора.

Рассмотрим для примера подпрограмму вычисления тангенса (TAN X).

```
37DA EF RST 0028 - вызов калькулятора. Значение X уже в
калькуляторном стеке.
37DB 31 DEFB+31     - дублирование X,X
37DC 1F DEFB+1F     - SIN X
37DD 01 DEFB+01     - обмен SIN X и X
37DE 20 DEFB+20     - COS X
37DF 05 DEFB+05     - деление - TAN X
37E0 38 DEFB+38     - конец
```

37E1 C9 RET - возврат к обычным машинным кодам и верхнее значение калькуляторного стека поменялось с X на TAN X.

Причина включения специальной системы использования команды RST 0028 со следующими за ней литералами состоит в том, что это сокращает программу в машинных кодах. Пример приведенный выше использует только 8 байтов памяти, в то время как использование 5-ти команд CALL потребовало бы большего количества байт.

Набор литер.

Существует "неиспользуемая" область 386E-3CFF, которая предшествует набору литер, занимающих адреса 3D00-3FFF.

Набор литер содержит 96 форм символов, каждый из которых использует 64 бита. Например знак '+' хранится в форме представленной ниже.

```
3D58 DEFB+00 '0 0 0 0 0 0 0 0'
3D59 DEFB+00 '0 0 0 0 0 0 0 0'
3D5A DEFB+08 '0 0 0 0 1 0 0 0'
3D5B DEFB+08 '0 0 0 0 1 0 0 0'
3D5C DEFB+3E '0 0 1 1 1 1 1 0'
3D5D DEFB+08 '0 0 0 0 1 0 0 0'
3D5E DEFB+08 '0 0 0 0 1 0 0 0'
3D5F DEFB+00 '0 0 0 0 0 0 0 0'
```

далее представлена программа, показывающая содержимое литерной области и соответствующего ей символа.

```
20 FOR A=15616 TO 16376 STEP 8
30 FOR B=0 TO 7: LET F=PEEK (A+B): GO SUB 300: NEXT B
40 PRINT
50 PRINT TAB 2; """"; CHR$ (32+(A-15615)/8); """"
60 PRINT AT 20,0;"PRESS ANY KEY TO CONTINUE": PAUSE 0
70 CLS
80 NEXT A
90 STOP
300 LET P=256
310 FOR N=7 TO 1 STEP-1
320 LET P=P/2
330 PRINT CHR$ (48+INT(F/P));
340 LET F=F-INT(F/P)*P
```

```
350 NEXT N
360 PRINT INT F
370 RETURN
```

строка 60 этой программы позволяет пользователю последовательно проследить формы всех 96 литер одна за одной.

7.2. Различные части программы монитора.

Программа монитора теперь будет обсуждена секция за секцией в том порядке в каком она занимает ROM.

0000-0007. RST 0000.

Первоначальный пуск. Запрещаются маскируемые прерывания, очищается регистр А, регистровая пара DE загружается значением +FFFF (верхнее значение адреса RAM) и происходит переход по адресу 11CB.

0008-000F. RST 0008.

Программа выхода по ошибке. Машинный стек очищается и выдается соответствующее сообщение.

0010-0012. RST 0010.

Точка входа программы PRINT-A (печать символа, код которого находится в регистре А). Осуществляется переход по адресу 15F2.

0018-0024. RST 0018 и RST 0020.

Получает текущий символ на который указывает CH-ADD как следующий.

0028-002F. RST 0028.

Переход по адресу 335B, который является стартовым адресом калькулятора.

0030-0037. RST 0030 BK-SPACE.

Переход по адресу 169E для выделения места в рабочей области.

0038-0052.

Программа обработки маскируемых прерываний. Часы реального времени обновляются и осуществляется сканирование клавиатуры вызовом по адресу 02BF.

0066-0070.

Программа обработки немаскируемых прерываний, которая перезапускает систему, если по адресу 5CB0 записан 0.

0095-0204.

Таблица ключевых слов. Программа показывает эту таблицу:

```
20 FOR A=149 TO 516
30 LET B=PEEK A
40 IF B3 THEN PRINT CHR$ B: GOTO 60
50 PRINT CHR$ (B-128)
60 NEXT A
```

0205-028D. Таблица клавиш.

Существует 6 таблиц, по 1 на каждый из возможных режимов. Наиболее важна 1-ая таблица (0205-022B), которая содержит значения в коде ASCII для больших букв и цифр.

028E-02BE. Программа KEY-SCAN.

Значение клавиши возвращается в регистровую пару DE. Флаг 0 сбрасывается, если одновременно нажато слишком много клавиш. Обычно регистр E указывает номер клавиши (16-ное 00-27), а регистр D указывает какой регистр клавиши был набран.

02BF-03B4. Подпрограмма KEYBOARD.

Набор подпрограмм, которые осуществляют возможность повторения и декодируют значения клавиш для получения требуемого кода литеры. Если клавиша нажата, то ее код копируется в системную переменную LAST-K и бит 5 системной переменной FLAGS устанавливается, чтобы сигнализировать о наличии нового кода.

03B5-03F7. Подпрограмма BEEPER.

При входе регистровая пара HL содержит высоту тона требуемой ноты и регистровая пара DE содержит ее длительность.

03F8-046D. Программа команды BEEP.

Эта программа широко использует калькулятор для вычисления длительности и высоты тона в соответствии со значением регистровых пар HL и DE. существует таблица 12 плавающих чисел (016E-04A8) для получения правильных значений полутонов.

04AA-04C1.

Эта программа была написана для ZX-81 и не изменена для использования в SPECTRUM, поэтому она не используется.

04C2-09F3.

Программа команд SAVE, LOAD, VERIFY и MERGE.

04C2-053E. Программа SAVE-BYTES.

Передает количество байт указанных в регистровой паре DE, начиная с адреса, хранящегося в регистре IX на магнитофон вместе с байтом четности.

053F-0555.

Подпрограмма окончания SAVE/LOAD.

0556-0604. Подпрограмма LOAD-BYTES.

Загружает или сопоставляет данные, количество которых указано в регистровой паре DE с магнитофона. Регистровая пара IX содержит адрес первого байта. Флаг переноса сброшен для операции верификации и установлен для операции загрузки или добавления.

SAVE-BYTES и LOAD-BYTES.

Используется как для работы с заголовком так и с блоком данных.

0605-075F.

Точка входа во все подпрограммы находится по адресу 0605, и эта часть имеет дело с конструкцией деталей заголовка в рабочей области и является общей для всех 4-х команд БЕЙСИКа.

0760-096F.

Эта часть связана с загрузкой, верификацией (сравнением) и добавлением и вызывает LOAD-BYTES.

0970-09A0.

Эта часть связана с сохранением и совсем проста. Канал предназначенный для печати в нижней части экрана открывается и появляется сообщение "пустите ленту". Затем появляется сообщение "нажмите любую клавишу" и ожидается нажатие. После этого выводится заголовок и еще через секунду - блок данных.

09A1-09F3.

Сообщения, связанные с работой магнитофона.

09F4-0D4C. Программа PRINT-OUTPUT - печать символа.

Бит 1 системной переменной FLAGS должен быть установлен для вывода на принтер и сброшен для вывода на TV-экран, важные части этой секции монитора описаны ниже.

09F4-0A10.

Символы, которые могут быть напечатаны, отделяются от управляющих символов. В любом случае ищется текущая позиция для печати.

0A11-0A22. Таблица управляющих символов.

0A23-0ADB.

Различные программы, имеющие дело с управляющими символами.

0ADC-0B02. Важная подпрограмма STORE.

Текущая позиция сохраняется в соответствующих системных переменных. Позиция печати может указывать на основную часть экрана, нижнюю часть экрана или буфер принтера.

0B03-0B23. Программа определения текущей позиции.

Текущая позиция выбирается из соответствующих системных переменных.

0B24-0BDA. Подпрограмма печати литеры PRINT-ANY.

Эта особенная подпрограмма представляет собой весьма важную часть программы печати символов в системе SPECTRUM. в регистровой паре HL содержится начальный адрес места где должен быть напечатан символ (системная переменная DF-CC). Регистровая пара BC содержит текущее значение строки и колонки (системная переменная S-POSN) и регистр A содержит код символа. Начальный адрес формы литеры находится или в наборе литер, или в области памяти, где находятся специально созданные графические литеры. P-FLAG также анализируется, чтобы определить требуемый режим (OVER или INVERSE или оба сразу).

0BB7-0BC4.

Форма символа копируется из памяти или в дисплейную область или в буфер принтера.

0BDB-0C09. Программа установки атрибутов.

После того, как символ напечатан, устанавливается байт атрибутов для этого символа. Это включает в себя поиск и формирование значения атрибутов и системных переменных ATTR-T, MASK-T, P-FLAG.

0C0A-0C54. Подпрограмма печати сообщений и ключевых слов.

Входной точкой для печати сообщений является 0C0A, а для печати ключевых слов - 0C10. В случае печати сообщений регистровая пара DE содержит начальный адрес таблицы сообщений (значение должно быть более чем 7F), а регистр A содержит номер сообщения (начиная с 0). Все символы печатаются до тех пор, пока не встретится "инверсный" символ. В случае печати ключевых слов команда по адресу 0C10 загружает в регистровую пару DE начальный адрес таблицы ключевых слов 0095.

0C55-0D4C. Подпрограмма SCROLL.

Всякий раз, когда символ печатается на TV-экране, каждая текущая позиция проверяется для определения необходимости продвижения экрана. Если продвижение необходимо, то печатается сообщение-подсказка и ожидается нажатие клавиши.

0D4D-0D6A.

Программа установки временных цветовых характеристик.

0D6B-0EAB. Программа команды CLS.

Операция очистки экрана в SP включает в себя заполнение всех байтов экранной области нулевыми значениями, байтов в области атрибутов специфицированными значениями.

0EAC-0F2B. Программы для принтера.

0EAC-0ECC. Программа команды COPY.

176 точечных линий непосредственно на принтер.

0ECD-0EF0. Подпрограмма COPY-BUFF.

Содержимое буфера принтера передается на принтер.

0EF4-0F2B. Собственно программа принтер.

0F2C-10A7. EDITOR.

Программа позволяет пользователю создавать строку на БЕЙСИКе в редакторской области памяти. При каждом нажатии клавиши печатаемый символ или ключевое слово по программе ADD-CHAR (0F81-0F91) добавляется в редактируемую строку. Адреса 0FAC-0FA8 содержат таблицу для кодов (16-ое 07-0F) для клавиш редактирования и секция 0FA9-10A7 содержит различные программы обработки этих клавиш. EDITOR также вызывается из программы команды INPUT и позволяет пользователю создавать входные строки в рабочей области.

10A8-111C. KEYBOARD-INPUT.

Подпрограмма собирает значения из LAST-K до тех пор, пока бит 5 FLAGS показывает, что нажата новая клавиша.

111D-11B6. Подпрограмма ED-COPY.

Редакторская строка или входная строка, построенные в редакторской или рабочих областях этой подпрограммой высвечиваются на TV-экране.

11B7-11CA. Программа команды NEW.

Эта команда выполняет операции системного рестарта, но оставляет без изменения переменные RAMTOP, P-RAMT, RASP, PIP, UDG. Она продолжается в INITIALISATION.

11CB-12A1. Программа INITIALISATION.

При входе в регистр A содержит значение 0 для полного системного рестарта или FF для генерации NEW.

Далее описаны основные части программы INITIALISATION.

11CC-11CF. TV-экран становится белым.

11D0-11D9.

В регистр I записывается +3F. Регистр используется для генерации TV-сигналов.

11DA-11EF. Программа RAM-CHECK.

Тестируются байты, начиная с 4000 до RAMTOP. При завершении регистровая пара HL содержит адрес последней пригодной для использования ячейки памяти.

11F0-11FF.

В случае операции NEW эта секция используется для перезаписи значений P-RAMT, RASP, PIP, UDG.

1200-1218.

Эта часть используется при системном рестарте. Графика пользователя от A до U устанавливается копированием форм из набора литер в область UDG. переменные PIP, RASP, P-RAMT инициализируются.

1219-1234.

Системная переменная CHARS устанавливается равной +3000 и организуется машинный стек. Устанавливается режим 1 системы прерываний, в регистровую пару IY загружается +5C3A и разрешаются маскируемые прерывания. С этого момента клавиатура опрашивается через каждые 20 миллисекунд (50 раз в секунду).

1235-1243.

Начальная информация каналов копируется в область каналов.

1244-127B.

Инициализируется ряд системных переменных. Например, переменные постоянных характеристик цвета устанавливаются так, чтобы были черные чернила, белая бумага и белый бордюр.

127C-1285.

Начальные данные потока копируются в первые 14 байтов STRMS (потоки 0, 1, 2, 3 и еще 3 дополнительных потока, используемых монитором для внутренней работы).

1286-12A1.

Очищается буфер принтера, очищается экран и фирменное сообщение печатается в нижней части экрана.

12A2-15AE.

Программа MAIN EXECUTION. Различные части этой программы будут обсуждены ниже.

12A2-12E1.

Основной цикл для построения и последовательного вывода программы БЕЙСИК. Синтаксис каждой строки проверяется и, если синтаксис правильный, строка копируется в область программ.

12E2-1302.

Непосредственная строка БЕЙСИКа, которая прошла проверку на синтаксис, должна быть интерпретирована. Возврат по адресу 1303 завершает интерпретацию независимо от причин.

1303-1390.

Соответствующие сообщения появляются на экране и цикл повторяется с адреса 12AC.

1391-155A. Таблица сообщений об ошибках.

Фирменное сообщение находится также в этой таблице.

1555-15AE. Заключительная часть программы MAIN EXECUTION.

Используется для копирования строки из редакторской обл. в соответствующее место в программной области. При копировании строки с номером, который уже существует, происходит замещение.

15AF-15C5. Начальная таблица данных канала.

15C6-15D3. Начальная таблица данных потока.

15D4-1651. Программа обслуживания канала "K".

Входом в этот набор программ является адрес 15D4. Программа ожидает нажатия клавиши на клавиатуре повторным вызовом подпрограммы KEYBOARD-INPUT до тех пор, пока установлен флаг переноса. Вход 15EF или 15F2 вызывают печать символа. Выход стандартно осуществляется через PRINT-OUTPUT. Первая точка входа используется для печати цифр, вторая для одинаковых символов и ключевых слов.

1652-1654. Подпрограмма ONE-SPACE.

Одиночное место выделяется в редакторской области или рабочей области для строки БЕЙСИКа.

1655-1663.

Текущее значение регистровой пары BC показывает как много места должно быть выделено, а регистровая пара HL указывает после какого места в памяти должна быть выделена эта область.

1664-168E. Подпрограмма POINTERS.

Все указатели от VARS до STREND при необходимости изменяются.

168F-169D. Подпрограмма сбора номера строки.

Для данного адреса начала строки номер строки собирается в регистровой паре DE.

16DE-16AF. Подпрограмма RESERVE.

Требуемый номер области делается доступен в рабочей области.

16B0-16D8. Набор программ очистки.

По коду 16B0 очищается редакторская строка, временная рабочая область и калькуляторный стек, в то время как по коду 16C5 очищается только калькуляторный стек.

16DB-16E5.

Подпрограмма индексации используется в различных применениях поиска по таблицам.

16E5-1792. Программа команд CLOSE и OPEN.

Открытие потока состоит в помещении отличного от 0 значения в соответствующую переменную STRMS для этого потока.

Например: PRINT PEEK 23584 содержит нормальный 0, который показывает что поток 5 закрыт. Но после OPEN #5,"K" команда PRINT PEEK 23584 даст 1 и поток теперь открыт и к нему подключен канал "K". После CLOSE #5 команда PRINT 23584 даст 0. эти программы проверяют, что потоки открываются и закрываются правильно.

1793-1794.

В стандартном SPECTRUM использование команд CAT, ERASE, FORMAT, MOVE приводит к выдаче сообщения об ошибке - "неверный поток"

1795-1A47. Программа листинга.

Вход 1795 используется MAIN EXECUTION для выработки автоматического листинга, вход 17F5 для LLIST и вход 17F9 собственно для LIST. различные части этой программы будут обсуждены ниже.

1855-18B5. Программа печати строки БЕЙСИКа.

Она вызывается повторно управляющими программами для печати каждой строки БЕЙСИКа. Сначала печатается номер строки, затем, если требуется курсор строки. Подпрограмма 18C1-18BC вызывается в случае необходимости мигающих символов. Окончательно печатаются все символы и ключевые слова.

190E-197F. Подпрограмма LINE-ADDR.

Она используется для поиска начального адреса строки БЕЙСИКа в области программы.

19B0-19D4. Подпрограмма NEXT-ONE.

Используя ее, ищется следующая строка БЕЙСИКа или следующая переменная.

19D5-19FA. Подпрограмма исправления.

Любые исправления которые необходимо произвести, выполняются этой подпрограммой, сначала исправляется указатель от VARS до STREND, а затем требуемое данное добавляется в конец калькуляторного стека.

1A1B. Подпрограмма печати номера.

Используется для печати номера оператора БЕЙСИКа, или при листинге, или при выводе сообщения.

1A48-1B16. Таблица команд.

Существует две таблицы. Первая таблица ссылок для таблицы параметров. Полная таблица параметров приведена ниже.

Таблица параметров:

Адрес и разделители	команда	класс команды	адрес программы
		команды	
1A7A	LET	01 = 02	2AFF
1A7D	GOTO	06 00	1E67
1A81	IF	06 THEN 05	1CF0
1A86	GO SUB	06 00	1EED
1A8A	STOP	00	1CEE
1A8D	RETURN	00	1F23
1A90	FOR	04 = 06 TO 06 05	1D03
1A98	NEXT	04 00	1DAB
1A9C	PRINT	05	1ECD
1A9F	INPUT	05	2089
1AA2	DIM	05	2C02
1AA5	REM	05	1BB2
1AA8	NEW	00	11B7
1AAB	RUN	03	1EA1
1AAE	LIST	05	17F9
1AB1	POKE	08 00	1E80
1AB5	RANDOMIZE	03	1E4F
1AB8	CONTINUE	00	1E5F
1ABB	CLEAR	03	1EAC
1ABE	CLS	00	0D6B
1AC1	PLOT	09 00	22DC
1AC5	PAUSE	06 00	1F3A
1AC9	READ	05	1DED
1ACC	DATA	05	1E27

1ACF	RESTORE	03	1E42
1AD2	DRAW	09 05	2382
1AD6	COPY	00	0EAC
1AD9	LPRINT	05	1FC9
1ADC	LLIST	05	17F5
1ADF	SAVE	0B	(0605)
1AE0	LOAD	0B	(0605)
1AE1	VERIFY	0B	(0605)
1AE2	MERGE	0B	(0605)
1AE3	BEEP	08 00	03F8
1AE7	CIRCLE	09 05	2320
1AEB	INK	07	(1C96)
1AEC	PAPER	07	(1C96)
1AED	FLASH	07	(1C96)
1AEE	BRIGHT	07	(1C96)
1AEF	INVERSE	07	(1C96)
1AF0	OVER	07	(1C96)
1AF1	OUT	08 00	1E7A
1AF5	BORDER	06 00	2294
1AF9	DEF FN	05	1F60
1AFC	OPEN	06 0A 00	1736
1B02	CLOSE	06 00	16E5
1B06	FORMAT	0A 00	1793
1B0A	MOVE	0A 0A 00	1793
1B10	ERASE	0A 00	1793
1B14	CAT	00	1793

замечание: некоторые адреса программ не находятся в таблице. Такие адреса даны в скобках.

1B17-1C00. Управляющая программа интерпретатора БЕЙСИК.

В случае проверки редактируемой строки на ошибку, вход в программу осуществляется по адресу 1B17 и выполняются следующие шаги:

1. Сбрасывается флаг синтаксиса - бит 7 FLAGS.
2. Любой номер строки проверяется на корректность - используя подпрограмму 'E-LINE NUMBER' по адресу 19FB.
3. Системная переменная, которая считает количество операторов в строке (SUBPPC) устанавливается в 0.
4. Системная переменная ERR-NR устанавливается в FF, затем по очереди (адреса 1B28-1B3C) на синтаксис проверяется каждый оператор строки. В случае отсутствия синтаксических ошибок, выход осуществляется через программу по адресу 1BB3-1BB6.

В случае непосредственной интерпретации строки БЕЙСИКа входом служит адрес 1B8A. Флаг синтаксиса (бит 7 FLAGS) должен быть всегда установлен. Затем операторы строки рассматриваются по очереди. Прямой возврат выполняется, если нет больше строк БЕЙСИКа для интерпретации. Однако, в том случае когда непосредственный оператор БЕЙСИКа содержит такие команды как RUN, GOTO, CONTINUE или соответствующее RETURN или NEXT возможна интерпретация других строк БЕЙСИКа прежде, чем возврат будет выполнен.

В случае выполнения интерпретации оператора выполняются следующие действия:

1. Определяется команда БЕЙСИКа и вычисляется адрес в таблице команд.
2. В таблице параметров определяется первая программа класса команды и затем выполняется.
3. Выполняются следующие программы класса команд или подбираются соответствующие разделители, пока не достигается состояние, при котором адрес программы команды находится и программа выполняется.
4. Когда интерпретирован последний оператор в строке, рассматривается следующая строка.

1C01-1C0C. Таблица класса команд.

Эта таблица используется для поиска основного адреса различных программ класса команд.

1C0D-1CDD. Программы классов команд.

Требования определяемые различными классами команд, могут быть суммированы так:

- ⇨ класс 00 - нет других операторов;
- ⇨ класс 01 - требуется переменная;
- ⇨ класс 02 - выражение числовое или строковое;
- ⇨ класс 03 - числовое выражение, по умолчанию используется 0;
- ⇨ класс 04 - следующая должна быть простая символьная переменная;
- ⇨ класс 05 - может быть набор элементов;
- ⇨ класс 06 - должно следовать числовое выражение;
- ⇨ класс 07 - элементы цвета;
- ⇨ класс 08 - должны следовать 2 числовых выражения, разделенных запятой;
- ⇨ класс 09 - как для класса 08, но элементам цвета могут предшествовать выражения;
- ⇨ класс 0A - должно следовать строковое выражение;
- ⇨ класс 0B - программы управления магнитофоном.

Программы классов команд довольно сложны и дальше обсуждаться не будут. Однако адреса 1CAD-1CBD, которые содержат часть программы класса 07, представляют собой интерес. Эти адреса содержат программы, кото-

рые копируют текущие временные системные переменные в соответствующие постоянные и которые могут вызываться по мере надобности.

1CDE-24FA. Программы команд.

Программы этой секции вновь достаточно сложны и не будут детально обсуждаться. В главе 8, однако, сделаны ссылки на различные программы печати в том виде, в котором они могут вызываться из программ пользователя.

24FB-28B1. Вычислитель выражений.

Наиболее интересная программа в SPECTRUM. Вычислитель функций, не требующих аргументов, выполняется внутри вычислителя выражений, не выделяясь в отдельные подпрограммы. В результате использования этого метода программист, желающий использовать эти программы - функции, должен обратиться за помощью к вычислителю выражений из программы VAL/VAL$ калькулятора (см. главу 8 для детализации).

Существует 2 основных момента, связанных с вычислителем функций. Первый из них связан с тем, что программа вычисляет следующее выражение и формирует одиночный результат. Этот результат занимает 5 байтов памяти, которые хранят число в плавающей форме для числового значения или набор параметров строки для каждого строкового значения. В любом случае эти 5 байт памяти содержат последнее значение и программа помещает их на верх стека калькулятора, то есть при выполнении вычислителя выражений стек калькулятора всегда увеличивается на одно значение.

Второй момент связан с тем, что вычислитель выражений использует значение приоритетов и операции с более высоким приоритетом должны выполняться раньше операций с низким приоритетом. Операции с равными приоритетами должны выполняться в порядке просмотра. В системе SPECTRUM значения приоритетов операций, которые еще будут выполнены, сохраняются в машинном стеке и по частям передаются в стек калькулятора. В машинном стеке также сохраняются литералы, определяющие какая программа калькулятора будет использоваться для различных операций.

Различные части вычислителя выражений описаны ниже.

24FB-24FE.

Сохранение в машинном стеке 0 значения маркера приоритетов.

24FF-2794.

Основной цикл вычислителя выражений. Через этот цикл проходит каждый элемент выражения.

2530-2534. Подпрограмма SYNTAX-Z.

Во время проверки синтаксиса флаг нуля должен быть установлен.

2535-257F. Программа SCREEN$.

2580-2595. Программа ATTR.

2596-25AE. Таблица функций, не требующих аргументов.

25F8-2626. Программа RND.

2627-2634. Программа PI.

2634-2667. Программа INKEY$.

2756-2758. Используется калькулятор для выполнения специфических операций с одним или двумя операндами.

2795-27AF. Таблица для преобразования кодов операций в литералы калькулятора.

27B0-27BC. Таблица приоритетов операций.

27BD-28B1. Программа FN.

28B2-299F. Подпрограмма LOOK-VARS.

Эта подпрограмма вызывается всякий раз, когда ищется поле переменной. Для специфицированной переменной возвращается адрес текущего значения если переменная уже использовалась, или устанавливаются соответствующие флаги в противном случае.

2996-2A51. Подпрограмма STK-VARS.

Сложная подпрограмма, используемая для поиска простых строковых переменных или переменных массива. Параметры строки или элемента массива возвращаются в стек калькулятора.

2A52-2AB0. Подпрограмма SLICING.

Любая подстрока может быть выделена из строки и эта подпрограмма вызывается всякий раз, когда нужно выделение.

2AB1-2ACB. Подпрограмма STK-STORE.

Очень полезная программа, которая передает параметры текущей строки (регистры A, B, C, D, E) в стек калькулятора. Стек расширяется на одно значение. В наборе параметров строки регистровая пара BC содержит длину строки, регистровая пара DE - начальный адрес строки и регистр A не используется и содержит ноль.

Иногда регистр a содержит единицу и индицирует наличие элементов массива.

2AFF-2BF0. Программа команды LET.

Она специально предназначена для команд LET и INPUT.

В случае простой числовой переменной, или старое значение переписывается, или новая переменная добавляется в конец текущего поля переменных. В случае простой строковой переменной любое старое значение будет

уничтожено или построена новая переменная в конце поля переменных. И, наконец, в случае переменного массива старое значение всегда переписывается.

2BF1-2C01. Подпрограмма STK-FETCH.

Вызов этой подпрограммы приводит к перезаписи верхнего значения стека калькулятора в регистры A,B,C,D,E. Стек калькулятора уменьшается на одно значение.

В системе SPECTRUM подпрограммы STK-STORE и STK-FETCH обычно используются для установки параметров строки, хотя это может быть и 5-байтовое плавающее значение (но использование точки входа 2AB2 для STK-STORE приведет к потере пятого байта).

2C02-2C87. Программа команды DIM.

Простая программа которая устанавливает специфицированный массив. Если заданный массив уже существует, то старый массив уничтожается прежде, чем новый добавляется в конец области переменных.

2C88-2F9A.

Разнообразные арифметические программы: наиболее важные из них описаны далее.

2D22-2D27. Подпрограмма STK-DIGIT.

Цифровой код ASCII (шестнадцатиричное 30-39) передается из регистра A в калькуляторный стек в плавающем виде.

2D28-2D2A. Подпрограмма STACK-A.

Значение содержащееся в регистре A, помещается в калькуляторный стек.

2D2B-2D3A. Подпрограмма STACK-BC.

Значение регистровой пары BC передается в калькуляторный стек.

Интересно, что эта подпрограмма используется как программа выхода из функции USR. Следовательно текущее значение в регистровой паре BC становится последним значением выражения "USR число".

Заметим особо, что регистровая пара IY снова устанавливается на +5C3A и поэтому всегда выполняется корректный возврат после вызова функции USR. в программах, написанных пользователем, необходимо учитывать эту особенность, если регистровая пара IY содержит модифицированное значение и используется STACK-BC или STACK-A.

2DA2-2DC0. Подпрограмма FP-to-BC.

Верхнее значение калькуляторного стека упаковывается в регистровую пару BC.

2DD5-2DE2. Подпрограмма FP-to-A.

Верхнее значение калькуляторного стека упаковывается в регистр A.

2DE3-2F9A. Подпрограмма PRINT-FP.

Длинная и очень сложная подпрограмма, которая получает верхнее значение калькуляторного стека, уменьшая его на 1, и печатает требуемое число в целом или плавающем формате. Эта подпрограмма печатает значение любых 5 байтов, независимо от того, являются они или не являются действительно числовыми.

При своей работе эта подпрограмма использует специальный буфер печати в 10 байтах MEM3 MEM4 (системные переменные 5CA1-5CAA).

7.3. Калькулятор.

В системе команд Z-80 нет команд умножения и деления, а имеющиеся команды сложения и вычитания позволяют производить операции только с целыми числами в диапазоне 0-65535. Поэтому в состав монитора включена специальная программа "калькулятор".

Эта программа позволяет выполнять большое количество арифметических и логических операций. Все операции производятся над числами, которые находятся в калькуляторном стеке. Стек калькулятора аналогичен машинному стеку. Но он отличается от него тем, что одна ячейка калькуляторного стека занимает 5 байт памяти и положение стека в памяти определяется системными переменными STKBOT и STKEND. Кроме этого существует еще шесть пятибайтовых ячеек памяти калькулятора: M0,M1,M2,M3,M4 и M5. Программист может копировать содержимое вершины калькуляторного стека в эти ячейки и помещать содержимое этих ячеек на стек. При программировании следует учитывать тот факт, что выполнение некоторых функций разрушает содержимое ячеек памяти калькулятора M0-M5:

- ⇨ SIN, COS, TAN, ASN, ACS, ATN, LN нарушают содержимое M0, M1 и M2;
- ⇨ EXP нарушает содержимое M0, M1, M2, M3;
- ⇨ INT, MOD DIV, GET ARGT нарушают содержимое M0;
- ⇨ STR$ нарушает содержимое всех шести ячеек от M0 до M5;
- ⇨ USR N может нарушить содержимое любой ячейки;
- ⇨ печать графических символов с кодом от 128 до 145 нарушает содержимое ячеек M0 и M1;
- ⇨ процедура печати десятичных чисел с плавающей точкой разрушает содержимое всех шести ячеек памяти калькулятора.

Запуск программы "калькулятор" происходит по команде RST 0028. После этой команды должны следовать однобайтовые инструкции калькулятора, которые определяют какие действия необходимо выполнить. Последней инструкцией в калькуляторной программе (калькуляторная программа - несколько инструкций калькулятора следующих друг за другом) должна быть инструкция с кодом 38 (16-ричное). Программа "калькулятор" после запуска последовательно просматривает инструкции, следующие после команды за-

пуска (RST 0028) и выполняет их. Когда встречается инструкция с кодом 38 (конец работы калькулятора) программа "калькулятор" завершается, и управление передается программе машинного кода, следующей за инструкцией 38 (16-ричное).

Необходимо отметить, что в системе инструкций калькулятора есть инструкции перехода (безусловного и условного). За инструкцией перехода должен следовать байт смещения (все переходы относительные) в дополнительной двоичной форме. Величина смещения отсчитывается от байта в котором находится инструкция перехода. Переход возможен как вперед, так и назад. Всего таких инструкций две:

- ⇨ инструкция безусловного перехода;
- ⇨ инструкция перехода по условию.

Результат выполнения всех операций заносится на вершину калькуляторного стека. При выполнении логических операций на вершину стека заносится значение "истина" или "ложь" в зависимости от результата. Значению "истина" соответствует ненулевое число, а значению "ложь" - ноль на вершине калькуляторного стека. Поэтому, когда встречается инструкция перехода по условию (код 00 16-ричное), то с вершины калькуляторного стека снимается значение. Если это значение равно нулю, то переход не выполняется и байт смещения игнорируется. А если это значение не равно нулю, то выполняется переход.

Необходимо помнить, что числа в калькуляторном стеке, над которыми выполняются операции, замещаются результатом выполнения операции. Поэтому, если эти числа нужны для дальнейших вычислений, необходимо их дублировать в калькуляторном стеке или копировать в ячейки памяти калькулятора.

Так же необходимо помнить, что содержимое регистра B микропроцессора может нарушиться при работе программы "калькулятор".

Таблица инструкций калькулятора.

Код	Условное обозначение	Содержимое вершины стека		нарушения регистров	назначение инструкции
		До	После		
00	JUMP TRUE	X	-	-	выполняется относительный переход на S шагов, если X имеет значение "истина"
01	EXCHANGE	X,Y	Y,X	-	меняются местами два числа на вершине стека

Код	Условное обозначение	Содержимое вершины стека		нарушения регистров	назначение инструкции
		До	После		
02	DELETE	X	-	-	верхнее число удаляется из стека
03	SUBTRACT	X,Y	X-Y	-	вычитание
04	MULTIPLAY	X,Y	X*Y	-	умножение
05	DIVIDE	X,Y	X/Y	-	деление
06	POWER	X,Y	X**Y	B,M0-M3	возведение в степень
07	OR	X,Y	"истина" или "ложь"	-	логическое "или"
08	N AND	X,Y	"истина" или "ложь"	-	логическое "и" (для чисел)
09	N LE	X,Y	"истина" или "ложь"	-	"меньше или равно", регистр B должен содержать 09 (для чисел)
0A	N GE	X,Y	"истина" или "ложь"	-	"больше или равно", регистр B должен содержать 0A (для чисел)
0B	N NE	X,Y	"истина" или "ложь"	-	"не равно", регистр B должен содержать 0B (для чисел)
0C	N GT	X,Y	"истина" или "ложь"	-	"больше", регистр B должен содержать 0C (для чисел)
0D	N LT	X,Y	"истина" или "ложь"	-	"меньше", регистр B должен содержать 0D (для чисел)
0E	N EQ	X,Y	"истина" или "ложь"	-	"равно", регистр B должен содержать 0E (для чисел)
OF	ADD	X,Y	X+Y	-	сложение (для чисел)

Код	Условное обозначение	Содержимое вершины стека		нарушения регистров	назначение инструкции
		До	После		
10	S AND	X$,Y	"истина" или "ложь"	-	логическое "и" для стринга и числа
11	S LE	X$,Y$	"истина" или "ложь"	-	"меньше или равно", регистр B должен содержать 11 (для стрингов)
12	S GE	X$,Y$	"истина" или "ложь"	-	"больше или ", регистр B должен содержать 12 (для стрингов)
13	S NE	X$,Y$	"истина" или "ложь"	-	"не равно", регистр B должен содержать 13 (для стрингов)
14	S GT	X$,Y$	"истина" или "ложь"	-	"больше", регистр B должен содержать 14 (для стрингов)
15	S LT	X$,Y$	"истина" или "ложь"	-	"меньше", регистр B должен содержать 15 (для стрингов)
16	S EQ	X$,Y$	"истина" или "ложь"	-	"равно", регистр B должен содержать 16 (для стрингов)
17	S ADD	X$,Y$	X$+Y$	-	слияние стрингов
18	VAL$	X$	VAL$ X$	-	расчет X$ без ограничительных расчет X$ без ограничительных кавычек как стрингового выражения, регистр B должен содержать 18

Код	Условное обозначение	Содержимое вершины стека		нарушения регистров	назначение инструкции
		До	После		
19	USR S	X$	USR X$	-	определение адреса графического символа пользователя, установленного для клавиши с символом X$
1A	READ IN	X	INKEY$ #X	-	определение символа, поступающего от канала, подключенного к потоку X (процедура содержит ошибку)
1B	NEGATE	X	-X	-	изменение знака на противоположный
1C	CODE	X$	CODE X$	-	определение кода символа
1D	VAL	X$	VAL X$	-	расчет X$ без ограничительных кавычек как числового выражения, регистр B должен содержать 1D
1E	LEN	X$	LEN X$	-	определение длины стринга
1F	SIN	X	SIN X	B,M0-M2	вычисление синуса
20	COS	X	COS X	B,M0-M2	вычисление косинуса
21	TAN	X	TAN X	B,M0-M2	вычисление тангенса
22	ASN	X	ASN X	B,M0-M2	вычисление арксинуса
23	ACS	X	ACS X	B,M0-M2	вычисление арккосинуса
24	ATN	X	ATN X	B,M0-M2	вычисление арктангенса
25	LN	X	LN X	B,M0-M2	вычисление натурального логарифма

Код	Условное обозначение	Содержимое вершины стека		нарушения регистров	назначение инструкции
		До	После		
26	EXP	X	EXP X	B,M0-M3	вычисление "E" в степени X, E - основание натурального логарифма
27	INT	X	INT X	M0	преобразование действительного числа в целое, округление выполняется вниз
28	SQR	X	SQR X	B,M0-M3	вычисление квадратного корня
29	SGN	X	SGN X	-	определение знака числа
2A	ABS	X	ABS X	-	определение абсолютной величины числа
2B	PEEK	X	PEEK X	-	определение содержимого ячейки памяти по ее адресу
2C	IN	X	IN X	-	ввод числа с внешнего порта X
2D	USR N	X	USR X	?	выполнение программы машинного кода начиная с адреса X
2E	STR$	X	STR$ X	M0-M5	преобразование числа в соответствующий ему стринг, в системной переменной MEM должен хранится адрес MEMBOT
2F	CHR$	X	CHR$ X	-	определение символа числа по его коду

Код	Условное обозначение	Содержимое вершины стека		нарушения регистров	назначение инструкции
		До	После		
30	NOT	X	"истина" или "ложь"	-	условие "не"
31	DUPLICATE	X	X,X	-	дублирование (повторение) числа на вершине стека
32	MOD DIV	X,Y	X-Y*INT(X/Y), INT(X/Y)	M0	выделение остатка от деления двух чисел и целого частного (для отрицательных чисел результат неверный)
33	JUMP	-	-	-	безусловный переход на S шагов
34	STK DATA	-	X	-	помещает на стек число X; оно следует за инструкцией и должно быть в упакованной форме
35	DJNZ	-	-	-	если в регистре B не 0, то выполняется переход на S байтов, а содержимое регистра B уменьшается на 1
36	LT Z	X	"истина" или "ложь"	-	условие "меньше нуля"
37	GT Z	X	"истина" или "ложь"	-	условие "больше нуля"
38	END CALC	-	-	-	выключение калькулятора

Код	Условное обозначение	Содержимое вершины стека		нарушения регистров	назначение инструкции
		До	После		
39	GET ARGT	X	2/PI*ASN (SIN X)	M0	вычисление 2/PI*ASN(SIN X)
3A	TRUNCATE	X	(SGN X) * INT(ABS X)	-	выделение целой части числа (округление "к нулю")
3B	EXECUTE B	-	-	-	выполнение инструкции калькулятора, код которой содержится в регистре B
3C	E TO FP	X	?	-	не работает по причине ошибки в ПЗУ
3D	RESTACK	X	X	-	перевод целого числа в действительное (в плавающую форму)
8n	SERIES N	X	Pn(X)	B,M0-M2	вычисление многочлена Чебышева n-го порядка
9n	SERIES N+10	X	Pn+10(X)	B,M0-M2	вычисление многочлена Чебышева (n+10)-го порядка
A0	CONST ZERO	-	0	-	помещает на стек 0
A1	CONST ONE	-	1	-	помещает на стек 1
A2	CONST HALF	-	1/2	-	помещает на стек 0.5
A3	CONST PI/2	-	1.570796	-	помещает на стек PI/2
A4	CONST TEN	-	10	-	помещает на стек 10

Код	Условное обозначение	Содержимое вершины стека		нарушения регистров	назначение инструкции
		До	После		
Cn	STORE Mn	X	X	Mn	копирование числа с вершины стека в Mn-ю ячейку памяти калькулятора, нормально n=0...5, но можно увеличить до 1F путем изменения системной переменной MEM
Dn	STORE M(n+10)	X	X	M(n+10)	то же, но Mn=M10...M1F
En	RECALL Mn	-	Mn	-	вызов на стек данных из M0-M5 (M0-MF)
Fn	RECALL M(n+10)	-	M(n+10)	-	то же, но для M10...M1F

Упакованная форма целых чисел от 0 до 65535.

Если число представить в двухъбайтовой форме MM NN, то упакованная форма будет следующей:

34 80 B0 00 NN MM

код 34 (16-ричное) здесь и далее означает инструкцию STK DATA.

Упакованная форма отрицательных целых чисел от -65535 до -1.

Если число представить в двухъбайтовой дополнительной двоичной форме MM NN, то упакованная форма будет следующей:

34 80 B0 FF NN MM

Упакованная форма стрингов.

Если адрес PP QQ, а длина MM NN, то упакованная форма стринга будет следующая:

34 B0 QQ PP NN MM

Упакованная форма действительных чисел.

Представим число в пятибайтной интегральной форме:

AA EE DD CC BB

здесь "AA" - экспонента. Если экспонента находится в диапазоне от 51 до 8F то пользуемся таблицей 1, в противном случае пользуемся таблицей 2.

Таблица 1.

интегральная форма	упакованная форма	значение AA"
AA EE 00 00 00	34 AA" EE	AA-50
AA EE DD 00 00	34 AA" EE DD	AA-10
AA EE DD CC 00	34 AA" EE DD CC	AA+30
AA EE DD CC BB	34 AA" EE DD CC BB	AA+70

Таблица 2.

интегральная форма	упакованная форма	значение AA"
AA EE 00 00 00	34 00 AA" EE	AA-50
AA EE DD 00 00	34 40 AA" EE DD	AA-50
AA EE DD CC 00	34 80 AA" EE DD CC	AA-50
AA EE DD CC BB	34 C0 AA" EE DD CC BB	AA-50

8. Использование подпрограмм монитора.

8.1. Введение.

Цель этой главы показать, что программы машинного кода могут быть написаны для SPECTRUM относительно просто, используя большое число подпрограмм, которые всегда имеются в программе монитора.

Автор не имеет намерения в этой книге обсуждать структурное программирование, но хотел бы отметить, что хорошая программа всегда приносит удовлетворение как пользователю так и самому программисту.

8.2. Шестнадцатиричный ввод.

В гл.6 программа HEX. LOADER (16-ный загрузчик) была так представлена, что читатель может составить программу машинного кода в списке данных DATA и потом выполнить ее. Однако HEX. LOADER не совсем подходит для больших программ и предпочтительна следующая программа :

```
1 REM HEX INPUT
5 CLEAR 31999
10 LET D=32000
20 DEF FN A(A$,B)=CODE A$(B)-48-7*(CODE A$(B)>57)
30 DEF FN C(A$)=16*FN A(A$,1)+FN A(A$,2)
40 DEF FN G$(F)=CHR$ (F+48+7*(F>9))
```

```
50 DEF FN H$(E)=FN G$(INT(E/16))+FN G$(e-16*INT(E/16))
60 DIM A$(2)
70 PRINT FN H$(PEEK D); TAB 7; FN H$(INT(D/256)); FN
H$(D-256*INT(D/256));
80 INPUT A$
90 LET=1
100 IF A$(1)="U" THEN LET L=-1: GOTO 130
110 IF A$(1)="*" THEN GOTO 160
120 IF A$(1) CHR$ 32 THEN POKE D, FN C(A$)
130 PRINT TAB 16; FN H$(PEEK D)
140 LET D=D+L
150 GOTO 70
160 INPUT "PRESS ANY KEY TO RUN THE ROUTINE"; A$
170 RANDOMIZE USR 32000
```

Примечание:

- ⇨ используйте режим CAPS LOCK;
- ⇨ введите значение как пару 16-ных символов;
- ⇨ используйте ENTER для выполнения шага вперед;
- ⇨ используйте "U" и ENTER, выполняя шаг назад;
- ⇨ используйте "*" и ENTER для выполнения программы;
- ⇨ строка 160 дает пользователю вторую возможность.

Используя вышеприведенную программу, читатель может просто ввести и проверить программу машинного кода.

Читатель должен быть хорошо осведомлен о программе SAVE перед тем, как использовать ее впервые, так как имеют место частые аварийные ситуации в системе.

8.3. Команда BEEP.

В БЕЙСИКе команда BEEP имеет форму: BEEP длительность, высота тона, где длительность должна быть положительной величиной, не превышающей 10 и высота тона - положительным или отрицательным числом, указывающим как далеко находится нота от среднего "до".

В программе машинного кода имеются два особых метода, используя которые программа BEEP может быть выполнена. По первому методу необходимо вызвать подпрограмму BEEPER с соответствующими величинами в двухъ регистровых парах DE и HL; в то время как по второму методу надо вызвать команду BEEP с помощью величин для длительности и высоты тона в калькуляторном стеке.

Метод 1.

Используя программу HEX INPUT введите:

```
7D00    11 05 01    LD DE,+0105    длительность 1 сек.
7D03    21 66 06    LD HL,+0666    высота тона
7D06    CD B5 03    CALL BEEPER
7D09    C9          RET
```

когда вышеприведенная программа будет выполнена, она даст тот же эффект, как BEEP 1,0.

Величины в регистровых парах DE и HL будут следующими:

- ⇨ рассмотрите ноту заданной частоты "F", например, среднее "до" принято как 261,63 гц в SP. затем длительность требуемого периода "T" будет просто равна F*T. Это задается в паре регистров DE.
- ⇨ величина для пары регистров HL представлена как 437.500/F-30.125

заметьте, что нет ограничения длительности 10 секундами при использовании метода.

Метод 2.

Используйте программу HEX. INPUT, чтобы ввести программу.

```
7D00    3E 01       LD A,+01       длительность 1 сек.
7D02    CD 28 2D    CALL STACK-A   перевод в стек
7D05    3E 00       LD A,+00       высота тона 0
7D07    CD 28 2D    CALL STACK-A   перевод в стек
7D0A    CD F8 03    CALL BEEP
7D0D    C9          RET
```

когда вышеприведенная программа будет выполнена, она даст тот же эффект, как BEEP 1,0.

Можно использовать программу STACK-A для целых значений, для дробных значений необходимо использовать подпрограмму STK-DATA.

То есть для BEEP 1.3,1.12 программа будет выглядеть следующим образом:

```
7D00    EF                  RST 0028       использование
                                           калькулятора
7D01    34                  DEFB +34       STK-DATA
7D02    F1                  DEFB           экспонента 81
7D03    26 66 66 66 DEFBS                  мантисса (= дес. 1.3)
7D07    34                  DEFB +34       STK-DATA
7D08    F1                  DEFB           экспонента 81
7D09    8F 5C 28 F6 DEFBS                  мантисса (= дес. 1.12)
7D0D    38                  DEFB +38       END CALC
7D0E    CD F8 03            CALL BEEP
7D11    C9                  RET
```

чтобы получить ряд звуков, величины могут помещаться в таблицу и выбираться как требуется.

8.4. Сохранение и загрузка.

Подпрограммы SAVE-BYTES и LOAD-BYTES вызываются из машинного кода простым способом.

В любом случае регистровая пара IX должна содержать адрес места назначения и регистровая пара DE - количество байтов, например, переслать байты с 7E00 по 7EFF включительно. Регистровая пара IX будет содержать 7E00, а DE 0100.

В регистр A записывается величина +FF, чтобы показать что переписывается блок данных.

Перед выполнением LOAD должен быть установлен флаг переноса.

Следующая программа сохранит байты.

```
7D00    3E FF          LD A,+FF          указатель бл. данных
7D02    DD 21 00 7E    LD IX,+START      установка пары IX
7D06    11 00 01       LD DE,+COUNT      установка пары DE
7D09    CD C2 04       CALL SAVE-BYTES   выполнить сохранение
7D0C    C9             RET
```

заметьте что байты сохранены без "заголовка", но загрузка будет в том случае, если известно количество байтов.

Программа загрузки будет следующая:

```
7D00    37             SCF               установка флага
                       переноса (сбрасывается для VERIFY)
7D01    3E FF          LD A,+FF указатель блока данных
7D03    DD 21 00 7E LD IX,+START         установка пары IX
7D07    11 00 01       LD DE,+COUNT      установка пары DE
7D0A    CD 56 05       CALL LOAD-BYTES   выполнить загрузку
7D0D    C9             RET
```

8.5. Элементы цвета.

В системе SPECTRUM все байты атрибутов имеют следующий вид:

бит 7 - FLASH;
бит 6 - BRIGHT;
биты 5-3 - PAPER;
биты 2-0 - INK.

этот вид относится к 768 байтам в области атрибутов и к системным переменным BORDER, ATTR-P, MASK-P, MASK-T.

В добавление к атрибутным байтам системная переменная P-FLAG используется для хранения постоянных и временных флагов для PAPER 9, INK 9, INVERSE и OVER.

Четные биты P-FLAG - биты 6,4,2,0 - являются временными флагами, а нечетные биты - 7,5,3,1 - являются постоянными флагами.

В большинстве случаев SPECTRUM использует временные величины, когда заполняется значение для атрибутной области, но при определенных

условиях более важных, когда выполняется программа CLS, используются постоянные величины.

Команда BORDER и 6 элементов цвета будут рассматриваться теперь по очереди.

BORDER.

В любое время цвет бордюра может изменятся путем использования инструкций OUT (+FE),A. Но эта операция обычно связана с сохранением нового значения цвета в битах 5-3 BORDER. заметьте, что другие биты контролируют мигание, яркость и цвет чернил нижней части экрана.

Следующая программа показывает, каким образом цвет бордюра может изменяться.

```
7D00    3E 02       LD A,'RED'      красный бордюр
7D02    D3 FE       OUT (+FE),A     бордюр изменяется
7D04    07          RLCA            цвет бордюра пересылается
7D05    07          RLCA            в биты 5-3 регистра A
7D06    07          RLCA
7D07    CB 6F       BIT 5,A         цвет чернил для нижней
7D09    20 02       JR NZ,BORD-2    части экрана -
7D0B    EE 07       XOR +07         контрастен
7D0D    32 48 5C    LD (BORDER),A   установить BORDER
7D10    C9          RET
```

или проще:

```
7D00 3E 02 LD A,'RED' красный бордюр
7D02 CD 9B 22 CALL BORD-1 установка бордюра
7D05 C9 RET
```

интересно увидеть, как в вышеприведенной программе цвет INK делается контрастным по сравнению с цветом бордюра.

PAPER.

Постоянный цвет бумаги задается битами 5-3 ATTR-P.

Следующая программа показывает для PAPER 0 - PAPER 7 только один метод, с помощью которого эти биты могут изменяться без изменения других атрибутов.

```
7D00    3A 8D 5C    LD A,(ATTR-P) поиск ATTR-P
7D03    0F          RRCA          пересылка битов 5-3
7D04    0F          RRCA          в биты      2-0
7D05    0F          RRCA
7D06    E6 F8       AND,+F8
7D08    C6 02       ADD A,+RED    ввести новый цвет
7D0A    07          RLCA          сдвиг байта
7D0B    07          RLCA
7D0C    07          RLCA
```

```
7D0D    32 8D 5C        LD (ATTR-P),A сохранить ATTR-P
7D10    C9              RET
```

Вышеприведенный метод не очень эффективный, но это "общий метод", который может использоваться в других ситуациях.

PAPER 8 задается отдельно и включает в себя установку битов 5-3 MASK-P, в то время как PAPER 9 включает в себя установку бита 7 P-FLAG.

INK.

Постоянный цвет чернил задается битами 2-0 ATTR-P.

Следующая программа показывает, как эти биты могут изменяться для INK 0 - INK 7.

```
7D00    3E F8           LD A,+F8        приготовить MASK
7D02    FD A6 53        AND (ATTR-P)    выделить биты 7-3
7D05    C6 02           ADD A,+RED      красные чернила
7D07    32 8D 5C        LD (ATTR-P),A сохранить ATTR-P
7D0A    C9              RET
```

заметьте, что ATTR-P в одном случае рассматривается как адрес 53A6, а в другом - как адрес 5C8D.

INK 8 включается установкой битов 2-0 MASK-P в то время как INK 9 - установкой бита 5 P-FLAG.

FLASH.

Можно задать три состояния FLASH установкой или сбросом бита 7 в ATTR-P и MASK-P следующим образом:

```
FLASH 0     0 для 7-го бита ATTR-P
FLASH 1     1 для 7-го бита ATTR-P
FLASH 8     1 для 7-го бита MASK-P.
BRIGHT.
```

Подобным образом задаются три состояния яркости используя установку и сброс бита 6 в ATTR-P и MASK-P:

```
BRIGHT 0    0 для 6-го бита ATTR-P
BRIGHT 1    1 для 6-го бита ATTR-P
BRIGHT 8    1 для 6-го бита MASK-P
```

во всех случаях, когда 8-ой режим требует исключения соответствующие биты MASK-P нужно сбросить опять.

OVER.

Два состояния OVER могут быть заданы установкой или сбросом 1-го бита в P-FLAG:

```
OVER 1      0 для 1-го бита P-FLAG
OVER 0      1 для 1-го бита P-FLAG
```

INVERSE.

Два состояния INVERSE могут быть представлены следующим образом:

установка и сброс бита 3 в P-FLAG аналогично OVER.

Постоянные и временные элементы цвета.

Большинство операций в SPECTRUM используют временные элементы цвета. Например, операция RST 0010. CLS является важной программой для использования постоянных цветов.

Для того, чтобы копировать текущее постоянное значение во временные системные переменные используется CALL TEMPS - CALL 0D4D и если временные величины будут использоваться как новые постоянные величины, используется CALL PERMS - "CALL 1CAD".

Читатель может написать эквивалентные программы машинного кода:

```
PAPER 4: INK 3: BEEP 2,0
```

или

```
BEEP 1,0: BORDER 5: BEEP 2,12: BORDER 2
```

8.6. Команда CLS и SCROLL.

В программе монитора SPECTRUM имеются подпрограммы для очистки части экранной области и для сдвига ее части. Эти подпрограммы могут вызываться только из программ машинного кода.

CLS.

Полная очистка экрана и установка всех атрибутных байтов в постоянные величины получена путем использования:

```
CALL CLS - "CALL 0D6B"
```

но важно убедиться, что канал S (канал "S" подключен к потоку номер 2) открыт до того, как вызывается программа. Каналу потребуется потом вторичное открытие, если будет производиться дальнейший вывод на печать. Следующая программа очищает весь экран:

```
7D00    3E 02       LD A,#2         открыть канал S
7D02    CD 01 16    CALL CHAN-OPEN
7D05    CD 6B 0D    CALL CLS        очистить экран
7D08    C9          RET
```

вышеприведенная программа имеет преимущество при использовании постоянных цветов, но проще во многих случаях использовать подпрограмму CL-LINE.

Подпрограмма CL-LINE.

Используется для очистки определенного числа строк экрана. Эти строки считаются снизу.

До того, как сделан вызов подпрограммы в регистр B должна быть помещена величина в диапазоне 01-18 (16-ричное), где 18 - очистить весь экран.

Следующая программа показывает, как используется эта подпрограмма:

```
7D00    06 17       LD B,+17        очистить
                                    23 (10-ное) строки,
7D02    CD 44 0E    CALL CL-LINE    оставить одну верхнюю
7D05    C9          RET
```

CL-SCROLL.

Эта интересная подпрограмма, т.к. она дает возможность пользователю сдвигать определенное число строк дисплея на одну строку вверх.

Подпрограмма кончается использованием подпрограммы CL-LINE для очистки нижней строки.

Регистр в используется для указания номера строки, начиная с которой и ниже все строки сдвигаются. Строки выше указанной остаются неподвижными. Значение в регистре B должно быть в диапазоне 01-17 (16-ричное).

Следующая программа показывает как подпрограмма будет использоваться:

```
7D00    06 16       LD B,+16        оставить верхнюю
                                    строку не сдвинутой
7D02    CD 00 0E    CALL CL-SCROLL
7D05    C9          RET
```

заметьте, что ни подпрограмма CL-LINE, ни подпрограмма CL-SCROLL не влияют на использование текущего канала.

8.7. Подпрограмма печати.

Для того, чтобы использовать любую из подпрограмм печати для изображения символов, подпрограмма PRINT-OUTPUT должна быть сделана текущей программой ввода. Это можно легко получить, открыв канал S. Поэтому строки:

```
LD A,+02
CALL CHAN-OPEN
```

должны использоваться до того, как вызываются любые подпрограммы.

RST 0010.

В системе SPECTRUM вся печать символов на экране выполняется при использовании этой команды. С открытым каналом S подпрограмма PRINT-OUTPUT по адресу 09F4 действует как программа вывода.

Инструкция RST 0010 очень мощная и может использоваться для печати любого символа, изменения текущего положения печати использованием AT и TAB, печати расширенных ключевых слов и временных цветовых элементов.

Следующая программа показывает это:

```
7D00    3E 02       LD A,+02        открыть канал S
7D02    CD 01 16    CALL CHAN-OPEN
7D05    06 18       LD B,+18
```

```
7D07    CD 44 0E    CALL CL-LINE    очистить экран
7D0A    3E 16       LD A,+'AT'      оператор
7D0C    D7          RST 0010        PRINT AT 5,0
7D0D    3E 05       LD A,+05
7D0F    D7          RST 0010
7D10    3E 00       LD A,+00
7D12    D7          RST 0010
7D13    3E 41       LD A,'A'
7D15    D7          RST 0010
7D16    3E 0D       LD A,'CR'       оператор PRINT
7D18    D7          RST 0010
7D19    3E F9       LD A,+F9        оператор
7D1B    D7          RST 0010        PRINT CHR$ 249
7D1C    3E 0D       LD A,'CR'
7D1E    D7          RST 0010        оператор PRINT
7D1F    3E 11       LD A,'PAPER'
7D21    D7          RST 0010        оператор PAPER 2
7D22    3E 02       LD A,'RED'      (временное)
7D24    D7          RST 0010
7D25    3E 06       LD A,','        оператор PRINT
7D27    D7          RST 0010
7D28    3E 42       LD A,'B'        оператор
7D2A    D7          RST 0010        PRINT "в"
7D2B    C9          RET
```

следует отметить, что в вышеприведенной программе вызов подпрограммы CL-LINE очищает экран, но не устанавливает положение печати "00" (в то время как CALL CLS устанавливают).

Когда используется инструкция RST 0010 для печати цветного элемента, тогда потребуется два отдельных вызова, а когда используется символ управления AT, тогда требуется 3 отдельных вызова.

Альтернативный метод изменения положения печати будет следующим:

загрузите пару регистров BC соответствующими величинами для нового положения печати - CALL CL-SET (CALL 0DD9), которые помещают требуемые величины в S-POSN и DF-C.

Величины в паре регистров BC определяют AT A,B; где регистр B содержит 16-ное 18-A, а регистр C 16-ное 21-B. например:

```
PRINT AT 5,0
LD BC,+1321
CALL 0DD9
```

Печать строк.

Строка символов в SPECTRUM всегда задается парами регистров DE и BC, где в паре регистров DE хранится адрес ячейки, содержащей первый символ строки, а в паре регистров BC содержится величина, равная числу символов в строке.

При печати строк заданных в этой форме используется подпрограмма PR-STRING.

Рассмотрим эту подпрограмму подробнее (PR-STRING - 203C):

```
203C      LD A,B          выбор старшего байта
OR C      операция OR с младшим
DEC BC    уменьшение счетчика
RET Z
LD A,(DE) поиск символа
INC DE    печать символа
RST 0010
JR PR-STRING              переход к следующему символу
```

любая строка символов может быть поэтому напечатана следующим образом:

загрузите стартовый адрес в DE;
загрузите длину в BC;
CALL PR-STRING - "CALL 203C".

Поскольку подпрограмма PR-STRING использует инструкцию RST 0010 для действительной печати символа, строка может содержать коды символов, цветные элементы, атрибуты положения печати и кодов ключевых слов.

Как пример использования подпрограммы PR-STRING выполните следующую программу.

```
7CF0      11 01 10 09     строка длиной 0F (16-ное)
7CF4      16 0A 05 53     начинающаяся в 7CF0
7CF8      70 65 63 74
7CFC      72 75 6D
.....................
7D00      3E 02           LD A,+02        открыть канал S
7D02      CD 01 16        CALL CHAN-OPEN
7D05      06 18           LD B,+18        очистить экран
7D07      CD 44 0E        CALL CL-LINE
7D0A      11 F0 7C        LD DE,+STRING   начало строки
7D0D      01 0F 00        LD BC,+LENGTH   длина строки
```

```
7D10    CD 3C 20    CALL PR-STRING печать строки
7D13    C9          RET
```

вышеприведенная программа подобна

```
CLS: PRINT PAPER 1; INC 9;AT 10,5;"SPECTRUM"
```

Печать чисел.

Очень мощная подпрограмма PRINT-FP используется для печати в десятичной форме любых чисел с плавающей запятой. Эта подпрограмма в качестве операнда использует верхнее значение стека калькулятора, действительное число выбирается из стека и теряется.

Следующая программа использует подпрограмму PRINT-FP для печати постоянной "PI/2" из таблицы констант калькулятора.

```
7D00    3E 02       LD A,+02        открыть канал S
7D02    CD 01 16    CALL CHAN-OPEN
7D05    CD 6B 0D    CALL CLS        очистить экран
7D08    3E 02       LD A,+02        снова открыть
7D0A    CD 01 16    CALL CHAN-OPEN  канал S
7D0D    EF          RST 0028        использовать
                                    калькулятор
7D0E    A3          DEFB +A3        выбрать константу
7D0F    38          DEFB +38        'END-CALC'
7D10    CD E3 2D    CALL PRINT-FP   печатать верхнее
                                    значение
7D13    C9          RET             в десятичном формате.
```

Использование строки RANDOMIZE USR 32000: PRINT PI/2 показывает идентичность результата.

Но интересно рассмотреть, что при получении PI/2, постоянная PI/2 выбиралась из таблицы констант, удваивалась, а затем делились на 2, чтобы в конечном счете получить результат.

Во многих случаях однако, использование аппарата плавающей точки может не потребоваться и по существу возможно печатать 10-ные числа в диапазоне 0-9999 путем использования подпрограммы OUT-NUM, которые обычно используются для воспроизведения номера строки в листинге или сообщении. Когда используется программа OUT-NUM-1 (16-ричный адрес 1A1B) число должно находится в паре регистров BC в обычном порядке "старший - младший", но когда используется подпрограмма OUT-NUM-2 (1A28) пара регистров HL используется для непосредственной адресации числа, которое в это время должно быть в порядке "младший - старший".

Следующий пример показывает, как используется подпрограмма OUT-NUM-1 для печати числа.

```
7D00    3E 02       LD A,+02        открыть канал S
7D02    CD 01 16    CALL CHAN-OPEN
```

```
7D05    CD 6B 0D    CALL CLS          очистить экран
7D08    3E 02       LD A,+02          снова открыть
7D0A    CD 01 16    CALL CHAN-OPEN    канал S
7D0D    01 0F 27    LD BC,+270F       десятичное 9999
7D10    CD 1B 1A    CALL OUT-NUM-1    печать числа
7D13    C9          RET
```

8.8. Команды PLOT, DRAW и CIRCLE.

Эти три команды имеют дело с изображением точек в экранной области. Весь экран состоит из 256*176 точек, что образует 22 верхние строки. Координаты нижней левой точки принимаются как 0,0; а верхней правой точки 255,175. системные переменные COORDS-x и COORDS-Y содержат координаты последней точки, которая должна использоваться. Эти системные переменные устанавливаются в 00 командой CLS, или другими командами, такими как RUN, которая вызывает программу CLS.

Команда PLOT.

Эта самая простая из трех команд, т.к. она включает в себя идентификацию только одного бита в области экрана и его последующую установку или сброс.

Имеется три вполне подходящих точки входа в команду, но третья из них, возможно, самая простая для использования.

1. CALL PLOT - "CALL 22DC". Для входа в команду PLOT необходимо иметь величины X и Y на верху калькуляторного стека. Величина X находиться ниже величины Y. Необходимо использование STACK-A для помещения значений в стек калькулятор.
2. CALL PLOT-1 - "CALL 22DF". Вход в этой точке требует, чтобы в регистре B хранилась величина Y, а в регистре C - величина X. после того, как точка выводится, постоянные цветовые величины копируются автоматически во временные системные переменные.
3. CALL PLOT-BC - "CALL 22E5". Опять регистр B должен содержать величину Y, а регистр C - величину X. Подпрограмма выполняет фактическую операцию PLOT. требуемый бит идентифицируется вызовом FIXED-ADD (22AA) и затем будет установлен или сброшен в соответствии с результатом его состояния и величиной P-FLAG (бит 0 P-FLAG даст постоянное состояние OVER и бит 2 даст состояние INVERSE).

Следующая программа показывает изображение одиночной точки с помощью PLOT.

```
7D00    3E 02       LD A,+02          открыть канал S
7D02    CD 01 16    CALL CHAN-OPEN
7D05    CD 6B 0D    CALL CLS          очистить экран
7D08    3E 02       LD A,+02          снова открыть
```

```
7D0A    CD 01 16    CALL CHAN-OPEN канал S
7D0D    01 64 32    LD BC,+3264    PLOT 100,50
7D10    CD E5 22    CALL PLOT-BC
7D13    C9          RET
```

Команда DRAW.

Использование DRAW X,Y по существу является расширением PLOT X,Y, описывающей ряд точек, которые составляют прямую линию.

Команда DRAW X,Y,A включает в себя изображение дуги и подобна по многим аспектам CIRCLE X,Y,R и DRAW X,Y.

Имеются две соответствующие точки входа для этой команды:

1. Точка входа DRAW-1 (2477) с величинами X и Y, которые являются верхними значениями стека калькулятора. Постоянные цветовые величины переписываются во временные системные переменные после того, как линия уже нарисована.

2. Точка входа DRAW-3 (24BA). При этом регистр B содержит ABS Y, регистр C содержит ABS X, регистр D содержит SGN X и регистр E, содержит SGN Y.

Следующая программа рисует линию.

Заметьте, что пара регистров H'L' должна быть сохранена, а позднее восстановлена для выполнения успешного возврата в БЕЙСИК.

```
7D00    3E 02          LD A,+02          открыть канал S
7D02    CD 01 16       CALL CHAN-OPEN
7D05    CD 6B 0D       CALL CLS          очистить экран
7D08    3E 02          LD A,+02          снова открыть
7D0A    CD 01 16       CALL CHAN-OPEN    канал S
7D0D    D9             EXX               сохранить
7D0E    E5             PUSH HL           значение H'L'
7D0F    D9             EXX
7D10    FD 36 43 64 LD (COORDS-x),+64    установить координаты
7D14    FD 36 44 64 LD (COORDS-Y),+64
7D18    01 32 32       LD BC,+3232       оператор
7D1B    11 01 01       LD DE,+0101       DRAW 50,50
7D1E    CD BA 24       CALL DRAW-3
7D21    D9             EXX               восстановить H'L'
7D22    E1             POP HL
7D23    D9             EXX
7D24    C9             RET
```

вышеприведенная программа устанавливает последнее положение графики в координате 100,100 и затем рисует линию +50,+50.

II. DRAW X,Y,A.

Для того, чтобы изобразить дугу величины X,Y и A должны быть первыми помещены в стек калькулятора либо в их длинной или короткой форме из пяти байтов. Затем можно вызвать DRAW-ARC (2394).

Следующая программа дает тот же результат, например DRAW 50,50,1 с начальным положением графики в координатах 100,100.

```
7D00    3E 02           LD A,+02        открыть канал S
7D02    CD 01 16        CALL CHAN-OPEN
7D05    CD 6B 0D        CALL CLS        очистить экран
7D08    3E 02           LD A,+02        снова открыть
7D0A    CD 01 16        CALL CHAN-OPEN  канал S
7D0D    D9              EXX             сохранить H'L'
7D0E    E5              PUSH HL
7D0F    D9              EXX
7D10    FD 36 43 64 LD(COORDS-X),+64 установить
7D14    FD 36 44 64 LD(COORDS-Y),+64 координаты
7D18    EF              RST 0028        использовать
                                        калькулятор
7D19    34              DEFB +34        STK-DATA
7D1A    40 B0 00 32 DEFBS               10-ное +50
7D1E    31              DEFB +31        DUPLICATE
7D1F    A1              DEFB +A1        STK-ONE
7D20    38              DEFB +38        END-CALC
7D21    CD 94 23        CALL DRAW-ARC   изображение
7D24    D9              EXX
7D25    E1              POP HL          восстановить H'L'
7D26    D9              EXX
7D27    C9              RET
```

в программе величины, полученные для X, Y, а будут десятичными +50,+50,+1.

Команда CIRCLE.

Опять три операнда команды должны быть помещены в стек калькулятора до того, как будет сделано обращение к CIRCLE-1 (232D).

Следующая программа показывает выполнение этой команды. Заметьте, что нет заметной экономии времени изображением окружности из машинного кода, т.к. программа команды CIRCLE медленная.

```
7D00    3E 02           LD A,+02        открыть канал S
7D02    CD 01 16        CALL CHAN-OPEN
7D05    CD 6B 0D        CALL CLS        очистить экран
7D08    3E 02           LD A,+02        снова открыть
7D0A    CD 01 16        CALL CHAN-OPEN  канал S
```

```
7D0D    D9              EXX             сохранить HL
7D0E    E5              PUSH HL
7D0F    D9              EXX
7D10    EF              RST 0028        использовать кальк.
7D11    34              DEFB +34        STK-DATA
7D12    40 B0 00 64 DEFBS               +100 10-ное
7D16    31              DEFB +31        DUPLICATE
7D17    34              DEFB +34        STK-DATA
7D18    40 B0 00 30 DEFBS               +48 (10-ное)
7D1C    38              DEFB +38        END-CALC
7D1D    CD 2D 23        CALL CIRCLE-1   изображение
7D20    D9              EXX             окружности
7D21    E1              POP HL          восстановить HL
7D22    D9              EXX
7D23    C9              RET
```

выше приведенная программа дает тот же результат, как CIRCLE 100,100,48.

Заметьте, что при использовании вышеприведенной программы постоянные цветовые величины копируются во временные системные переменные, одновременно с изображением дуги или окружности.

8.9. POINT, ATTR и SCREEN$.

Эти три функции могут вызываться из программы машинного кода очень просто, т.к. каждая имеет отдельную вычислительную подпрограмму. Они могут также вызываться как функции, используя VAL, но этот метод не заслуживает внимания.

POINT.

Координаты точки, которые должны быть переданы, вводятся в пару регистров BC и вызывается подпрограмма POINT-1 (22CE). Как обычно регистр B хранит координату Y, регистр C хранит координату X.

Результат выполнения возвращается верхним значением стека калькулятора.

Следующая программа показывает выполнение команды POINT.

```
7D00    3E 02           LD A,+02        открыть канал S
7D02    CD 01 16        CALL CHAN-OPEN
7D05    CD 6B 0D        CALL CLS        очистить экран
7D08    3E 02           LD A,+02        снова открыть
7D0A    CD 01 16        CALL CHAN-OPEN  канал S
7D0D    3E 41           LD A,'A'        печать 'A'
7D0F    D7              RST 0010
7D10    3E 0D           LD A,'CR'       переход к следующей
7D12    D7              RST 0010        строке
```

```
7D13    01 04 AE    LD BC,+ae04    оператор POINT (4,174)
7D16    CD c6 22    CALL POINT-1
7D19    CD E3 2D    CALL PRINT-FP  печать результата
7D1C    C9          RET
```

вышеприведенная программа аналогична команде PRINT "A"; CHR$ 13; POINT (4,174).

ATTR.

Номер строки вводится в регистр С, а номер колонки в регистр В, и вызывается подпрограмма ATTR-1 (2583). Результат выполнения возвращается верхним значением калькуляторного стека.

Следующая программа показывает выполнение.

```
7D00    3E 02       LD A,+02       открыть канал S
7D02    CD 01 16    CALL CHAN-OPEN
7D05    CD 6B 0D    CALL CLS       очистить экран
7D08    3E 02       LD A,+02       снова открыть
7D0A    CD 01 16    CALL CHAN-OPEN канал S
7D0D    3E 11       LD A,'PAPER'   бумага временно
7D0F    D7          RST 0010       становится голубой
7D10    3E 05       LD A,'CYAN'
7D12    D7          RST 0010
7D13    3E FF       LD A,'COPY'    печать ключевого
7D15    D7          RST 0010       слова COPY
7D16    3E 0D       LD A,'CR'      переход к следующей
7D18    D7          RST 0010       строке
7D19    01 00 04    LD BC,+0400    оператор ATTR (0,4)
7D1C    CD 83 25    CALL ATTR-1
7D1F    CD E3 2D    CALL PRINT-FP
7D22    C9          RET
```

вышеприведенная программа аналогична команде:

```
PRINT PAPER 5;CHR$ 255;CHR$ 13;ATTR (0,4)
SCREEN$.
```

Опять номер строки засылается в регистр С, а номер колонки в регистр В. Затем вызывается подпрограмма SCREEN$-1 (2538). Результат возвращается как верхнее значение стека калькулятора, но следует отметить, что это набор параметров строки.

Пример:

```
7D00    3E 02       LD A,+02       открыть канал S
7D02    CD 01 16    CALL CHAN-OPEN
7D05    CD 6B 0D    CALL CLS       очистить экран
7D08    3E 02       LD A,+02       снова открыть
7D0A    CD 01 16    CALL CHAN-OPEN канал S
```

```
7D0D    3E 7A       LD A,'Z'        печать символа
7D0F    D7          RST 0010
7D10    3E 0D       LD A,'CR'       переход к следующей
7D12    D7          RST 0010        строке
7D13    01 00 00    LD BC,+0000     оператор SCREEN$(0,0)
7D16    CD 38 25    CALL SCREEN$-1
7D19    CD F1 2B    CALL FETCH      печать строки
7D1C    CD 3C 20    CALL PR-STRING
7D1F    C9          RET
```

программа аналогична команде PRINT "Z"; CHR$ 13; SCREEN$ (0,0). Помните, что SCREEN$ будет искать только в наборе символов и следовательно, определяет коды в диапазоне 20-7F (16-ричное).

8.10. PI, RND, INKEY$.

Эти функции группируются вместе, потому что их программные вычисления включают в вычислитель выражений и невозможно вычислить их непосредственно.

Однако использованием программы VAL/VAL$ калькулятора, который вызывается вычислителем выражений, следует рассмотреть преобразование.

```
VAL         CHR$    167         для PI
VAL         CHR$    165         для RND
VAL$ CHR$   166     для INKEY$
```

PI.

Процедура вычисления для PI находится по адресу 26C6 и выглядит следующим образом:

```
262C    RST 0028    использование калькулятора
262D    A3          'PI/2'
262E    3B          'END-CALC'
262F    INC (HL)    удвоение увеличением экспоненты
```

процедура занимает 4 байта и легко копируется. Ее можно вызвать с помощью VAL CHR$ 167.

Пример:

```
7D00    3E 02       LD A,+02        открыть канал S
7D02    CD 01 16    CALL CHAN-OPEN
7D05    CD 6B 0D    CALL CLS        очистить экран
7D08    3E 02       LD A,+02        снова открыть
7D0A    CD 01 16    CALL CHAN-OPEN  канал S
7D0D    3E A7       LD A,'PI'       код PI
7D0F    CD 28 2D    CALL STACK-A
7D12    EF          RST 0028        использовать
7D13    2F          DEFB +2F        калькулятор
```

```
7D14    38          DEFB +38        строка
7D15    06 1D       LD B,+1D        литерал
7D17    EF          RST 0028
7D18    3B          DEFB +3B
7D19    38          DEFB +38        END-CALC
7D1A    CD E3 2D    CALL PRINT-FP
7D1D    C9          RET
```

RND.

Функция RND вычисляется следующим образом: содержимое системной переменной SEED делится на 65535.

При включении питания или NEW величина SEED равна 0, затем, с помощью 1-го обращения к RND величина SEED становится 74 (10-ное). Это дает первое случайное число: 74/65535=0,0011291504 последовательность чисел для SEED 0,74,5624,28652...0,74 то есть представлены все числа от 0 до 65535.

Правило для изменения величины SEED следующее:

1. Прибавьте 1 (SEED=SEED+1);
2. Умножьте на 75 (SEED=SEED*75);
3. Возьмите по модулю 65537 (SEED=SEED-65537*INT(SEED/65537));
4. Вычтите 1 (SEED=SEED+1).

Следующая программа БЕЙСИК показывает эти изменения.

```
10 INPUT "SEED ?";CHR$ 32;SEED
20 PRINT "OLD VALUE =";CHR$ 32;SEED
30 LET SEED=SEED+1
40 LET SEED=SEED*75
50 LET SEED=SEED-65537*INT(SEED/65537)
60 LET SEED=SEED-1
70 PRINT "NEW VALUE =";CHR$ 32;SEED
80 GOTO 30
```

Программа RND может вызываться из программы в машинных кодах как результата вычисления функции VAL CHR$ 165.

Пример:

```
7D00    3E 02       LD A,+02        открыть канал S
7D02    CD 01 16    CALL CHAN-OPEN
7D05    CD 6B 0D    CALL CLS        очистить экран
7D08    3E 02       LD A,+02        снова открыть
7D0A    CD 01 16    CALL CHAN-OPEN  канал S
7D0D    3E A5       LD A,'RND'      код RND
7D0F    CD 28 2D    CALL STACK-A
```

```
7D12    EF          RST 0028    использовать
7D13    2F          DEFB +2F    калькулятор
7D14    38          DEFB +38    строка
7D15    06 1D       LD B,+1D    литерал
7D17    EF          RST 0028
7D18    3B          DEFB +3B
7D19    38          DEFB +38    END-CALC
7D1A    CD E3 2D    CALL PRINT-FP
7D1D    C9          RET
```

INKEY$.

Имеется ряд путей, с помощью которых опрос клавиатуры может осуществляться из программы в машинных кодах:

1. Непосредственным чтением системной переменной LAST-K.
2. Клавиатура может опрашиваться путем использования инструкции IN аналогично KEY-SCAN (028E).
3. Подпрограмма KEY-SCAN может вызываться сама и результат значения клавиши находится по известным таблицам.
4. Действие выполняемые по программе INKEY$ (2646 16-ное) могут быть повторены.
5. Программа INKEY$ может использоваться вычислением выражения VAL$ CHR$ 166.

В следующей программе демонстрируется эти возможность.

```
7D00    3E 02       LD A,+02
7D02    CD 01 16    CALL CHAN-OPEN
7D05    CD 6B 0D    CALL CLS
7D08    3E 02       LD A,+02
7D0A    CD 01 16    CALL CHAN-OPEN
7D0D    11 00 08    LD DE,+0800
7D10    D5          PUSH DE
7D11    3E A6       LD A,'INKEY$'
7D13    CD 28 2D    CALL STACK-A
7D16    EF          RST 0028
7D17    2F          DEFB +2F
7D18    38          DEFB +38
7D19    06 18       LD B,+18
7D1B    EF          RST 0028
7D1C    3B          DEFB +3B
7D1D    38          DEFB +38
7D1E    CD F1 2B    CALL STK-FETCH выбор параметров
                                   строки
7D21    CD 3C 20    CALL PR-STRING печать строки
```

```
7D24    CD BF 16    CALL X-TEMP    очистка рабочей
7D27    D1          POP DE         области
7D28    7B          LD A,E
7D29    B2          OR D
7D2A    C8          RET Z
7D2B    1B          DEC DE
7D2C    18 E2       JR LOOP
7D2D    00
```

когда эта программа выполняется, то она аналогична выражению:

```
FOR A=1 TO 2048: PRINT INKEY$;: NEXT A
```

8.11. BREAK (работа с клавиатурой).

Рассмотрим принцип ввода данных с клавиатуры. Клавиатура ZX SPECTRUM разбита на 8 полурядов по пять клавиш. Данные с клавиатуры поступают через порт с адресом 254 (FE 16-ричное). Пять младших разрядов (0-4) этого порта определяют какая из клавиш полуряда была нажата. Шестой разряд порта указывает на наличие сигнала с магнитофона (1 - сигнал есть, 0 - сигнал отсутствует). Остальные разряды не используются.

Обращение к клавиатуре происходит следующим образом. Формируется адрес порта:

- ⇨ младший байт адреса равен 254 (FE 16-ричное);
- ⇨ старший байт адреса определяет опрашиваемый полуряд клавиатуры в соответствии с таблицей.

Затем производится ввод данных с порта по сформированному адресу. Нулевое значение любого из пяти младших битов в полученном байте означает нажатие соответствующей клавиши.

Нужно отметить, что назначение разрядов (битов) порта 254 при вводе и выводе различно (смотри таблицу). При этом при вводе старшая часть адреса определяет опрашиваемый полуряд, а при выводе старшая часть адреса порта игнорируется (младшая часть адреса всегда 254).

Таблица разбивки клавиатуры.

номер полуряда	разряд порта 254 (FE)					старшая часть адреса порта
	4	3	2	1	0	
1	5	4	3	2	1	F7 (247)
2	6	7	8	9	0	EF (239)
3	T	R	E	W	Q	FB (251)
4	Y	U	I	O	P	DF (223)

номер полуряда	разряд порта 254 (FE)					старшая часть адреса порта
	4	3	2	1	0	
5	G	F	D	S	A	FD (253)
6	H	J	K	L	ENTER	BF (191)
7	V	C	X	Z	C.S.	FE (254)
8	B	N	M	S.S.	SPACE	7F (127)

Структура порта с адресом 254 (FE -16-ричное).

номер бита	назначение бита при операциях ввода (IN)	назначение бита при операциях вывода (OUT)
7	не используется	не используется
6	ввод с магнитофона	не используется
5	не используется	не используется
4	определяют нажатую клавишу полуряда, если бит равен нулю, то клавиша нажата. Полуряд определяется старшим байтом адреса в соответствии с таблицей разбивки клавиатуры, младший байт 254	выход на динамик
3		выход на магнитофон
2		определяют цвет бордюра (один из возможных восьми цветов)
1		
0		

напоминаем, что 7-ой разряд - это старший (крайний левый) разряд в байте, а 0-ой - младший.

Во многих случаях полезно, чтобы был сделан выход из программы машинного кода по клавише BREAK.

Пример:

```
7D00    3E 02       LD A,+02
7D02    CD 01 16    CALL CHAN-OPEN
7D05    CD 6B 0D    CALL CLS
7D08    3E 02       LD A,+02
7D0A    CD 01 16    CALL CHAN-OPEN
```

```
7D0D    3E 7F     LD A,+7F      вводной порт 7FFE
7D0F    DB FE     IN A,(+FE)    бит 0 регистра A
7D11    1F        RRA           сброшен, если
7D12    D0        RET NC        нажато BREAK.
7D13    18 F8     JR BREAK      переход адресу 7D0D
```

в программе половина ряда клавиатуры от BREAK до B неоднократно просматривается. Выход будет осуществлен только тогда, когда сбросится бит, который соответствует клавише BREAK.

8.12. Заключение.

Теперь читатель готов к написанию программ в машинных кодах. Программы в машинных кодах могут использоваться в программе на БЕЙСИКе. Они вызываются в программе по команде "USR-число".

Разумеется, что программы необходимо писать в мнемонических обозначениях, а затем переводить их в машинный код, используя специальные программы (например EDITAS, ZEUS, GENS4 и так далее). Используемый в этой книге загрузчик машинных кодов применим только для написания простых и коротких программ.

СОДЕРЖАНИЕ