ZX-SPECTRUM

ПРОГРАММИРОВАНИЕ
в машинных кодах

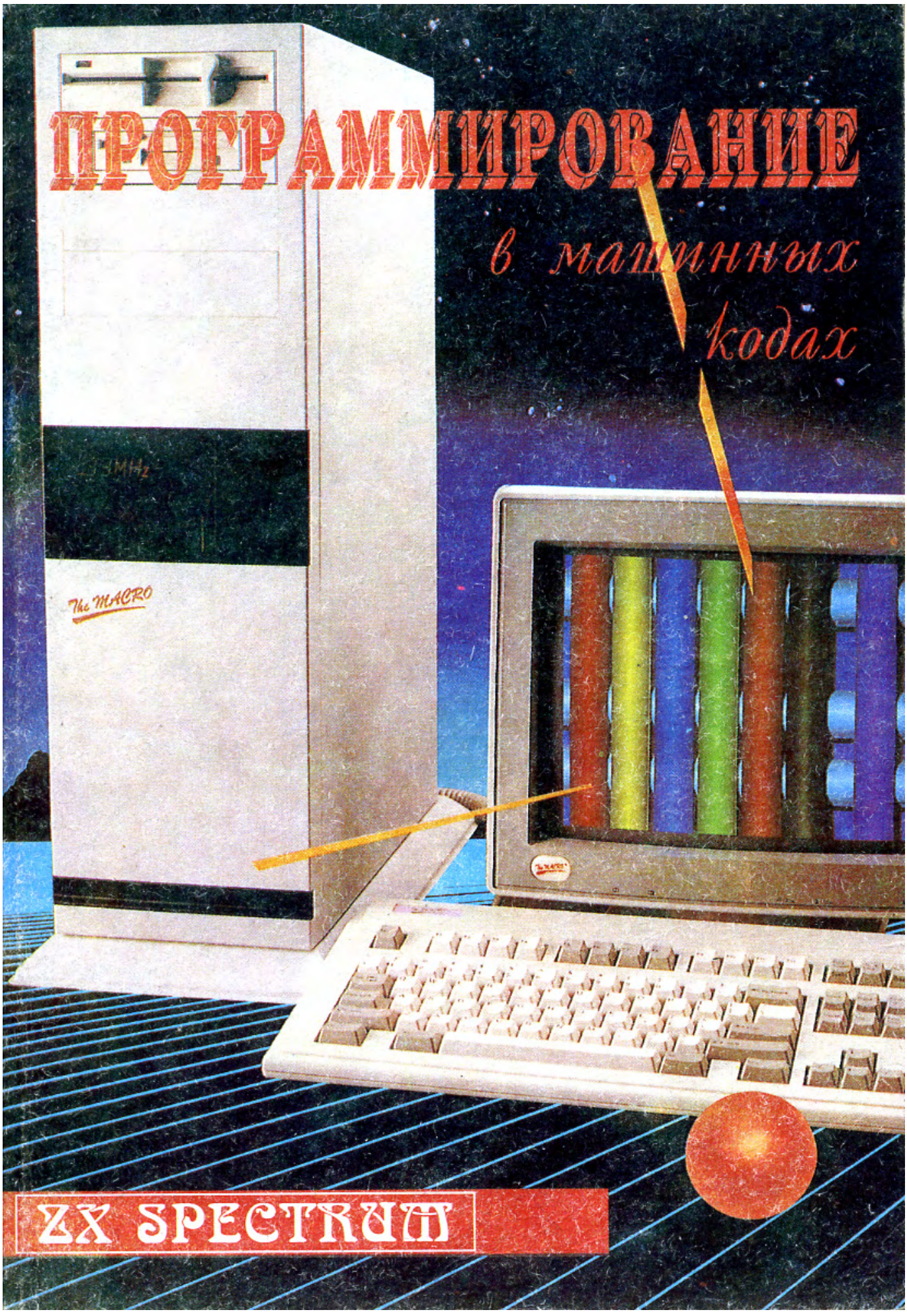
PDF version by Deny (Денисенко Д.А.)
e-mail: DenyDA@mail.ru
2007

ПРЕДИСЛОВИЕ ОТ DENY

Прежде чем читатель приступит к прочтению данной книги, я предупреждаю, что данная книга не является сто процентной копией книги. Почему? Просто я исправил неимоверное количество ошибок (такое ощущение, что редактор книги понятия не имеет о компьютерах) и не полностью соблюдал формат текста, хотя старался.

P.S. Возможно я исправил не все ошибки. Осталось одна известная ошибка – не работает демонстрационная программа INKEY$ из главы 8.

С уважение Deny.


ПРОГРАММИРОВАНИЕ
в машинных кодах
The MACRO
ZX SPECTRUM

ZX SPECTRUM

МАШИННЫЕ КОДЫ

Продукция фирмы VA PRINT

Москва
1993

ББК 32.973.2 – 01
П78

"VA PRINT" предлагает

Широкий выбор разнообразной литературы по Спектрум - совместимым компьютерам как для начинающих, так и для пользователей со стажем. Вот краткий перечень того, что мы предлагаем сейчас:

1. Описание игровых программ:
   - 500 игр, часть 1
   - 500 игр, часть 2
   - 500 игр, часть 3
   - 500 игр, часть 4
2. Справочники пользователя прикладных и системных программ ZX Spectrum.
3. Ассемблер Z-80.
4. Основы программирования на языке BASIC, описание команд, примеры программ.
5. TR-DOS.
6. Описание редакторов, языков и программ для ZX Spectrum
7. Лабиринты (описания секретов игровых экранов).
8. Графика, машинные коды, бессмертие игр и др.
9. Инструкция по пользованию компьютером 48К.
10. Инструкция по пользованию компьютером 128К.
11. Электронные устройства ZX Spectrum.
12. Полный дизассемблер ПЗУ (1982г.).

Планируются к выходу 5 и 6 выпуски описаний игровых программ, а также другая литература. Мы поможем Вам войти в удивительный мир компьютера ZX Spectrum.

Ждем Ваших предложений о сотрудничестве.

По вопросам оптового приобретения продукции фирмы
"ВА ПРИНТ" обращайтесь по телефонам:

(095) * 354 - 94 - 39
495 - 67 - 49
491 - 24 - 06

или по адресу: Москва, 127322, а/я 50.

© Оформление "ВА ПРИНТ"
© Составление "ВА ПРИНТ"
© Перевод "ВА ПРИНТ"

ISBN 5-85734-011-x

© Авторское право и право на
издание принадлежит фирме
"ВА ПРИНТ"

ВВЕДЕНИЕ

Почти невозможно поверить, что всего 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 чёрно-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 я чувствую, что для того, чтобы от неё получить всё, потребуется написание более совершенных программ. Эта книга написана для более глубокого понимания работы со SPECTRUM и компьютером в целом.

1. ПРИНЦИПЫ РАБОТЫ МИКРОКОМПЬЮТЕРНОЙ СИСТЕМЫ

1.1. ПРЕДИСЛОВИЕ

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.

1.2. ОПИСАНИЕ СИСТЕМЫ

Сам по себе SPECTRUM представляет собой чёрный пластмассовый корпус шириной 233 мм, глубиной 144 мм, высотой 30 мм. На верхней части расположены 40 клавиш, которые образуют клавиатуру. Сзади расположен выходной разъём для подключения антенного входа телевизора, входной разъём, который соединяется с выходом магнитофона, разъём для подключения УВВ: принтера, микродрайверов и т.д.; разъём для подключения БП.

На основной плате находится процессор Z80 и другие элементы. Плата находится в том же корпусе, что и клавиатура. Клавиатура и ПП соединены между собой 2-мя ленточными кабелями.

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

А сейчас рассмотрим каждый элемент печатной платы:

1.3. Z80 - МИКРОПРОЦЕССОР

Это кремниевый кристалл (чип). Он является самым важным из всех элементов и предназначен для управления работой компьютера. Программа для Z80 представляет собой набор кодовых инструкций и согласованных с ними данных. Z80 имеет задающую частоту 3.5 МГЦ, что позволяет выполнять 875000 простых кодовых инструкций в секунду. При подаче питания к МП он через определенное время становится работоспособным.

1.4. 16K ROM (ПОСТОЯННАЯ ПАМЯТЬ)

Представляет собой чип объёмом 128 Кбит или 16 Кбайт памяти, содержащей программу в машинных кодах для Z80.

Программа монитор 16К разбита на 3 части:
7 Кбайт - операционная система;
8 Кбайт - интерпретатор Бейсик;
1 Кбайт - генератор знаков.

VA PRINT

1.5. 16K RAM (ОПЕРАТИВНАЯ ПАМЯТЬ - ОЗУ)

В стандартной версии 16К SPECTRUM имеет 8 чипов по 2 Кбайта или 16 Кбит, а версия 48К содержит ещё дополнительно 32 Кбайт памяти.

Три из 8 чипов памяти образуют "ОБЛАСТЬ ЭКРАНА" и обычно используются только для этой цели, а 4 чип предназначен для хранения всех атрибутов 768 знаков экрана (24 по 32 строки) и системных переменных. В версии 16К остается свободным не более 8К памяти.

1.6. ULA

Это большой чип, включающий в себя много небольших чипов. В SPECTRUME ULA больше всего связана с областью экрана и атрибутной областью для получения TV -сигнала.

1.7. PAL - КОДИРОВЩИК

Этот чип получает цветовую информацию от ULA и использует её в формировании требуемого сигнала для UHF-МОДУЛЯТОРА. В английской версии SPECTRUM сигнал, полученный от модулятора, находится на 36-м дециметровом диапазоне.

На печатной плате дополнительно размещены: динамик, радиатор, регулятор напряжения, системные часы, чипы выборки адресов и другие.

Логика работы

Рассмотрим связь между различными элементами микрокомпьютерной системы.

Z80 микропроцессор (МП) может обращаться к 65536 адресам памяти (64K). Ограничением на непосредственную адресацию МП Z80 является 64K. В версии 16K SPECTRUM можно адресоваться только к адресам от 0 до 32767 десятичном, а в версии 48K SPECTRUM - доступны адреса 0-65535. Адрес задается в виде 16-битного сигнала. 0 представляется как 0000 0000 0000 0000, а 65535 как 1111 1111 1111 1111. Адреса, выданные Z80 поступают на адресную шину, состоящую из 16 линий, находящихся в высоком или низком напряжении (0 или 1). Адрес можно описать в двух 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 SPECTRUM, также важно рассмотреть нормальный режим работы системы и обсудить карту памяти. SINCLAIR RESEARCH выпускает SPECTRUM с 16K мониторной программой и обеспечивает пользователя операционной системой (ОС) и интерпретатором Бейсик (ИБ). По существу, возможно вместо программы монитор использовать МП Z80 с программой в машинных кодах.

При обычном использовании ОС не требует со стороны пользователя, каких либо сложных действий. Поэтому при работе SP уже работает ИБ как часть мониторной программы. Пользователь может сразу выводить или выполнять Бейсик программу. Т.о., ИБ является подпрограммой ОС, а Бейсик программа - подпрограммой ИБ.

Заметьте, что Z80 сам по себе не может выполнять Бейсик - программы, а только программу, составленную в машинных кодах.

Карта памяти стандартного 16K SP представлена на рис. И далее кратко описана.

32767	7FFF	Графическая область	PRAMT UDG
32600	7F58	пользователя	
32599	7F57	Стек GOSUB	RAMTOP
		Машинный стек	указатель стека
		Резервная память	STREND
		Стек вычислителя	STKBOT
		Рабочая область	WORKSP
		Область редактора	ELINE
		Область переменных	VAHS
		Область Бейсик программ	PROG
		Область информационных каналов	CHANS
23734	5CB6	Карты микродрайвов	
23552	5C00	Системные переменные	
23296	5B00	Буфер печати	
22528	5800	Область атрибутов	
16384	4000	Область экрана	
0	0000	ROM область	

VA PRINT

1.8. ROM ОБЛАСТЬ

16K ROM содержащие ОС, ИБ и генератор знаков (ГЗ) располагаются в адресах 0-16383 (16-ное 0-3FFF). В адресе 0 хранится стартовый адрес программы в кодах.

1.9. КАРТА ПАМЯТИ ЭКРАННОЙ ОБЛАСТИ

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

В 6K - 1024*8*6=49152 битов.

Количество символов в 24 строках экрана (по 64 точки на символ) равно 32*24*64=49152. Экран TV разбит на 3 части. Линиям с номерами 0-7 соответствуют адреса памяти 16384-18431 (4000-47FF), линиям 8-15 адреса 18432-20479 (4800-4FFF), 16-23 адреса 20480-22527 (5000-57FF).

Рассмотрим каждые 2K памяти, как 8 областей по 1/4 Кбайт. Каждая из областей задает биты 1 линии (256 точек) трети экрана начиная сверху. Это распространяется на все 24 строки экрана.

1.10. ОБЛАСТЬ АТРИБУТОВ

Экран имеет 768 символов, каждый из которых имеет 1 из 8 цветов "бумаги", 8 цветов "чернил", признак мигания, признаки повышенной или пониженной яркости.

Область расположена в адресах 22528-23295 (5800-5AFF), где закодированы биты, последовательно определяющие атрибуты экрана.

Зависимость между площадями символа и байтами атрибута несложная, так как байты просматриваются подряд для каждой линии экрана сверху вниз, слева направо.

Биты 0-2 определяют цвет чернил, 3-5 цвет бумаги, 6 яркость, 7 мигание.

1.11. БУФЕР ПЕЧАТИ

Область, расположенная в адресах 23296-23551 (5B00-5BFF) содержит 256 байт, по 32 байта на каждую печатаемую строку.

Этот буфер, если требуется, можно использовать как рабочую область памяти.

1.12. СИСТЕМНЫЕ ПЕРЕМЕННЫЕ

182 байта в адресах 23552-23733 (5C00-5CB5) отводятся под системные переменные, которые в дальнейшем будут подробно описаны.

1.13. КАРТЫ МИКРОДРАЙВОВ

Эта область памяти начинается с 23734 (5CB6).

1.14. ОБЛАСТЬ ИНФОРМАЦИОННЫХ КАНАЛОВ

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

В стандартном SPECTRUM без подсоединенных микродрайвов имеются 4 основных канала:

А. Канал K - обеспечивает ввод от клавиатуры и вывод на нижнюю часть экрана;

Б. Канал S - только вывод на верхнюю часть экрана;

В. Канал R - только вывод на рабочую область, которая по требованию может быть увеличена;

Г. Канал P - только вывод на принтер.

Информационные каналы состоят для каждого канала из 5 байт данных. Эти байты содержат: адрес программы ввода (2 байта), адрес программы вывода (2 байта) и один символ имени файла (1 байт).

В стандартном SPECTRUM, с учётом 4 каналов и маркера конца, область занимает память 23734-23754 (5CB6-5CAA=21 байт).

1.15. ОБЛАСТЬ БЕЙСИК-ПРОГРАММЫ (БП)

Область содержит последовательные строки программ. Её 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 микродрайв или использован дополнительный канал (скажем, для подключения дисковода).

1.16. ФОРМАТ СТРОК БП

Первые 2 байта любой строки содержат номер строки, и, причём в начале хранится старший байт, а затем - младший. В 3 и 4 байтах храниться длина строки, причем сначала младший байт, а потом - старший. Дальше байты с 5 до символа ENTER 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 идёт NUMBER (код 14(0E)) и плавающая точка или целые числа в диапазоне от -63535 до +63535, которые в любом случае займут 5 байт. Это приводит к тому, что на каждое число отводится 6 байт памяти из RAM дополнительно Следующая демонстрационная программа всё подробно показывает:

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

1.17. ОВЛАСТЬ ПЕРЕМЕННЫ

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 её 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
       RUN
```

VA PRINT

1.18. ОБЛАСТЬ РЕДАКТОРА

В области отражается строка, которую вводят или редактируют Начало области задаётся в переменной ELINE – адреса 3641-23642 (5C59-5C5A).

Когда нижняя часть экрана высвечивает только мигающий курсор, то область редактирования будет иметь 2 определяющих её адреса.

Первый из них, указанный в системной переменной KCUB, хранит адрес символа ENTER 2-ой адрес указывает на маркер конца (код 128). Нижняя часть экрана высвечивает редактируемую строку программы по мере поступления входных символов с клавиатуры область редактирования расширяется.

Аналогичная процедура имеет место, когда используется клавиша EDIT для вызова строки БП на нижнюю часть дисплея. Прежде всего область редактирования расширяется до требуемого размера с тем, чтобы разместилась эта строка. Затем эта строка копируется из области программы в область редактирования и, наконец, строка из области редактирования копируется в нижнюю часть экранной области RAM. Этот последний этап в действительности включает в себя отображение на экране символов с высокой разрешающей способностью.

Так как область редактирования представляет собой динамическую область, т.е. она меняется при использовании, то нецелесообразно давать пример на Бейсике.

1.19. РАБОЧАЯ ОБЛАСТЬ

Эта область используется для выполнения большого числа различных задач, например ввод данных, соединение в цепочку строк и т.д. 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 сводится к 0, чтобы избежать большего расхода памяти, чем это необходимо.

Ещё раз отметим, что поскольку эта область является динамической, то невозможно привести простой пример на Бейсике.

1.20. СТЕК ВЫЧИСЛИТЕЛЯ

Это очень важная область памяти. Она начинается с адреса, определенного системной переменной STKBCT, которая сама находится в адресах 23651 и 23652 (5C63, 5C64). Область распространяется до адреса, заданного в системной переменной STKEND (адреса 23653, 23654 (5C65, 5C66)). Стек вычислителя содержит числа с плавающей точкой, пять байт целых, а при использовании строк, он содержит пять байт, определяющих параметры строки. Стек обслуживается по правилу "первым пришел - последним ушел". Можно рассматривать величину, находящуюся вверху стека, если она существует, как последнюю пришедшую величину.

1.21. РЕЗЕРВНАЯ ПАМЯТЬ

Область памяти между стековым вычислителем и стеком машины представляет собой количество памяти, которым располагает пользователь. В стандартном 16K SPECTRUM номинальный размер области равен 8939 когда система включается. Однако интересно отметить, что приемлемая самая низкая величина для CLEAR равна 23821, которая опускает RAMTOP и расширяет резервную область до 8878 байт.

1.22. МАШИННЫЙ СТЕК

Мп Z80 должен иметь рабочую область для своего использования, и это называется машинный стек. Указатель стека Z80 всегда показывает на последний адрес, который должен заполняться.

Машинный стек будет подробно рассматриваться дальше.

VA PRINT

1.23. GO SUB - СТЕК

Когда имеется какое-либо активное GO SUB, количество строк сохраняется в GO SUB-стеке.

Стек расширяется в памяти вниз и каждая команда GO SUB требует 3 адреса. Самый старший адрес содержит номер операнда в пределах строки Бейсик, к которой должен быть осуществлен возврат. Второй адрес содержит младшую часть номера строки, определяющей цикл и третий адрес содержит её старшую часть.

Демонстрация следующей программы показывает стек GO SUB, который используется для сохранения чисел строки при организации 3-х гнездовых подпрограмм.

Демонстрационная программа GO SUB-стека

```
1 CLEAR 32599
10 GOSUB 20: STOP
20 GOSUB 30: RETURN
30 GOSUB 40: RETURN
40 FOR A-32547 TO 32584 STEP 1: PRINT A.PEEK A: NEXT A: RETURN
     RUN
```

Два адреса выше GO SUB-стека всегда содержат величины 0 и 62 (00 и 3E), и они представляют собой

неправильный номер строк.

Программа Бейсик при выполнении лишней команды RETURN перейдет к неправильному номеру строки и выдаст сообщение "RETURN без GO SUB".

Примечание: фирма SINKLAIR не совсем уверена в написании GO SUB.

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

1.24. ГРАФИЧЕСКАЯ ОБЛАСТЬ, ОПРЕДЕЛЕННАЯ ПОЛЬЗОВАТЕЛЕМ

До тех пор, пока системная область Бейсика расширяется вниз вследствие использования команды CLEAR, старшие 168 ячеек памяти содержат представления 21 графического символа, определенного пользователем.

Самый старший адрес памяти всегда адресуется системной переменной PRAMT, расположенной в адресах 23732 и 23733 (5CB4, 5CB5).

В стандартном 16K SPECTRUM величина, содержащаяся в PRAMT, должна быть равна 32767, поскольку это показывает, что все 16K памяти находятся в рабочем состоянии.

Это, очевидно, не вредит случайному вводу строки: **`PRINT PEEK 23732+256*PEEK 23733`** и видно, что результат является в действительности величиной 32767 (в 48K SPECTRUM - 65535).

VA PRINT

2. ФУНКЦИИ И КОМАНДЫ БЕЙСИКА

2.1. ВВЕДЕНИЕ

Ожидается, что читатели этой книги уже будут иметь некоторые знания о Бейсике "SP".

Эта глава обсуждает команды Бейсика и функции, и постарается прояснить моменты, не освященные детально в двух пособиях, прилагающихся к SP.

Интерпретатор Бейсик для SP 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 2.3.

2.2. КОМАНДЫ БЕЙСИКА

BEEP X,Y

По этой команде подается сигнал в динамик. X-длительность в секундах (0…10), Y-высота звукового тона, от среднего "си", в десятичном диапазоне чисел от -60 до +69. 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 -0.5<=M <=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 даёт красный бордюр). Но OUT и BORDER различны. Цвет, получаемый бордюром по команде OUT - "временный", в то время как BORDER даёт 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 вернётся 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 Бейсика (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 подсветлённой бумаге. Использование BRIGHT 1 и CLS - простой путь сделать всю область экрана яркой. При M=8 "высветленным" будет текст, подлежащий печати. Следующие примеры показывают 4 различных пути, в которых команда цвета, такая как BRIGHT, может быть использована.

```
10 BRIGHT 1: PRINT "BRIGHT-": BRIGHT 0: PRINT "NORMAL"
```

где два изменения яркости имеют постоянный характер.

```
20 PRINT BRIGHT 1;"BRIGHT-"; BRIGHT 0;"NORMAL"
```
– изменения яркости временны в течение оператора.

```
30 PRINT CHR$ 19+CHR$ 1;"BRIGHT-"; CHR$ 19+CHR$ 0;"NORMAL"
```
– команда в CHR$ эквивалентна.

```
40 LET A$=CHR$ 19+CHR$ 1: LET B$=CHR$ 19+CHR$ 0: PRINT A$;"BRIGHT-";B$;"NORMAL"
```
– помещает команду цвета в строку переменных.

Также возможно использовать последовательность набора:

- ✓ режим курсора "E и клавиша 9 для получения BRIGHT 1
- ✓ режим курсора "E" и клавиша 8 для получения BRIGHT 0

Этот набор может быть размещен внутри скобок, т.е. между открывающей скобкой и первым символом, или как единственный символ строки.

Например: LET A$="": REM внутри кавычек набрана вышеприведенная последовательность с клавишей "$". Строка A$ будет напечатана как при BRIGHT 1.

CIRCLE X,Y,Z

Эта команда рисует окружность с радиусом Z и с координатами центра X, Y. Z - берётся 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 её влияние всегда будет временным. По некоторым параметрам команда медленна и неточна, но очень полезна.

CLEAR; CLEAR N

В системе SP 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 её очень сильной. RAMTOP указывает на верхнюю границу системы Бейсик, и содержимое адресов ниже RAMTOP

© VA PRINT

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 SP выдает звуковой сигнал. Это говорит о том, что разрешенного RAM не хватает на задачу. Верхней границей N является для системы 16K 32767, а для систем 48K 65535. Использование CLEAR N с соответствующим значением вызывает перемещение стека GO SUB 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

Используется с микродрайвом или дисководом.

CLS

Это очень простая команда, которая меньше чем за 1/10 секунды выполняет много работы. Команда CLS очищает дисплейный файл. Она зануляет адреса с 16384 по 22527 (4000-57FF) и сбрасывает байты атрибутов по адресам 22528-23295 (5800-5AFF). Байты атрибутов не зануляются, а копируются из системной переменной ATTRP, которая содержит постоянные атрибуты. По команде POKE значение переменной ATTRP, которая помещается по адресу 23693 (5C8D), может быть изменено. В переменной ATTRP 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, SP 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 Бейсика. 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. Поскольку принтер не управляется SP, 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 руководстве, 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-ти функций - 26 числовых и 26 строковых. Имена, используемые для функции, должны быть единичного характера (+$ 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 ординарными литерами, (+$ для строк). Поэтому может быть не более 52 аргументов.

Выражение, определяемое функцией, может быть любым дающим числовой или строковый результат. Однако рекурсивное использование приводит к неудаче.

DIM (E1…Ek), DIM$ (E1…Ek)

Команда DIM удаляет существующий массив с тем же именем и затем устанавливает новый. Числовые массивы зануляются, а в строковые заносится литера пробел. В системе SP вся индексация начинается с 1. Достижение же индексом 0-го значения является ошибкой. Использование в системе SP в строковых массивах строк фиксированной длины, тогда как простая строковая переменная меняет длину, может привести к неудаче. Наиболее просто, используя в определении строкового массива последний индекс, всегда фиксировать длину строки. Эта особенность может быть полезна при установлении формата TV - экрана.

DRAW X,Y, DRAW X,Y,Z

Эта команда рисует линию от текущей позиции (но, не включая её) на расстояние X по горизонтали и Y по

VA PRINT

вертикали. Если аргумент Z определен, то рисуется дуга вместо прямой линии. Z должно быть определено в радианах, при Z=PI получается полуокружность. При положительных значениях дуги рисуются справа от того места, где была прямая линия, а при отрицательных слева. Любая из команд цвета может быть включена в оператор DRAW, где её влияние будет временным.

ERASE

Используется с микродрайвом или дисководом.

FLASH M

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

M, как и в BRIGHT, может принимать значения 0,1,8. При M=0 область литер не мигает; при M=1 они мигают, при M=8 сохранится исходная ситуация.

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 и любые управляющие переменные с тем же именем.
2. Добавить к существующим переменным новую управляющую переменную. Эта переменная требует 19 адресов памяти. Первый адрес содержит имя переменной. Следующие 5 содержат начальное значение цикла. Следующие 5 байтов конечное значение цикла. Следующие 5 байтов значение приращения. 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. Т.о., 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 устанавливает пустую 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
RUN
```

Добавьте **25 LET V=V+25** и увидите контрольную переменную B. Третье действие, предпринимаемое оператором, является очень специфичным. В SP 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. Это возможность перепрыгивать через целые циклы, может быть и преимуществом. Это позволяет снять ограничения на 0 и недостатком - ожидаемый результат не появляется вообще.

FORMAT

Используется с микродрайвом или дисководом.

GO SUB N

Эта команда приводит к выполнению подпрограммы, которая начинается с номера N или с первого номера этой строки. После выполнения подпрограммы управление возвращается к 1-му оператору в программе после оператора содержащего GO SUB. Следующая программа показывает, что в 16K SP GO SUB стек начинается с адресов 32595-32597.

```
10 GO SUB 20: STOP
20 FOR A=1 TO 3: PRINT A, PEEK (32594+A): NEXT A
RUN
```

Важным свойством Бейсика является то, что он позволяет командам GO SUB и GO TO делать управляемые переходы. Эта черта недоступна на многих других машинах, хотя обычные команды ON X GO SUB и ON X GO TO действительно позволяют сделать некоторое количество управляемых переходов.

GO TO N

Это очень простая команда. Следующей строкой является строка N или первая строка после N. Переходы могут быть сделаны и к REM операторам, хотя делать это, вероятно, не корректно. 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 недопустим.

VA PRINT

IF THEN S

В системе SP значение 0 соответствует логическому значению ложь, а любое значение, отличное от 0 - логическому значению истинности. Команда выполняет S и другие операторы в строках, 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. Проблема, появляющаяся в этом случае, связана с получением в SP величины 2

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

И она будет распечатываться больше 5-ти экранов, до тех пор пока она не достигнет значения 2""™. Если строку 40 поменять на LET A-A*0.5, то значение ложь будет достигнуто.

INK N

Это 3-я из команд элементов цвета. Когда используется самостоятельно, то её действие постоянно, а когда добавляется в оператор печати - временное. Значение N может быть от 0 до 9. 8 основных цветов в SP закодированы от 0 до 7, и это ясно видно на клавиатуре. Использование INK 8 при печати дает цвет, хранившийся в атрибутах. 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 ATTRP; для N=8 биты 0-2 переменной MASKP, для N=9 бит 5 PFLAG. Если команда INK имеет временное действие, то устанавливаются соответствующие биты временных системных переменных.

INPUT

Это очень мощная команда, она позволяет набором операторов INPUT 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-LINE 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. При постоянном режиме по команде устанавливается бит 5 PFLAG, а при временном бит 4.

LET V=E

Переменная выбрана пользователем, и затем интерпретатор должен определить, является ли переменная простой, должна быть найдена как управляющая переменная FOR…NEXT цикла или образует часть массива. Простая переменная может иметь любые существующие значения для определения и распределения нового места. Управляющая переменная и переменная массива будет размещена, но не определена. Далее определяется выражение, которое должно дать величину переменной, вычисляется и копируется в определенное место. Все численные переменные имеют 5 адресов, распределенные для величины. Эта величина затем сохраняется как число с плавающей запятой или целое число. Простые строки имеют динамическую длину, она представляет число адресов, отведенных на простую строку и зависит от количества символов в строке на текущий момент. Все массивы строк имеют фиксированную длину и строка, предназначенная для массива, будет уменьшаться, если она очень длинная и заполняться знаками пробела, если слишком коротка для фиксированной области, отведенной для нее. Пример показывает простой путь, как можно проследить за областью переменной. Пользователю надо ввести ряд переменных для наблюдения.

```
10 REM ENTER YOUR VARIABLE.
20 LET V=PEEK 23627+256*PEEK 23628
30 PRINT V; PEEK V: LET V=V+1: GO TO 30
```

следует сделать несколько предложений для строки 10

```
LET A=0
LET A=9E4
LET A$="AAA"
DIM A(2): LET A(1)=9E4
DIM A$(5): LET A$="1234567"
DIM A$(2,5): LET A$(1)="AAA"
```

LIST N

Допускает, что величина N=0 до тех пор, пока она не будет определена другим способом. Если строка

WARNING

существует, то она становится текущей строкой и обозначается курсором. Особенность системы: LIST 49172, например, будет такой же, как LIST 20.

Посылает листинг программы на принтер. N становится текущей строкой.

LOAD

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

В SP 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. В случае использования область программы и область переменной содержаться на МЛ.
4. 2 байта, которые для программы содержат начальный номер строки или начало для 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.

MERGE

Загружает программу и её переменные с МЛ. И соединяет с существующей программой и её переменными. Команды выполняют 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 программы. 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, PRAMT, 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, включающие в себя выполнения команды, следующие:

- ✓ управляемая переменная расположена в области переменных, и величина STEP добавляется к VALUE, вне зависимости от того, является STEP положительным или отрицательным.
- ✓ далее новая VALUE проверяется относительно LIMIT, но здесь следует обратить внимание на знак STEP. Если LIMIT не достигнут, тогда выполняется дальнейшее прохождение. В SP VALUE всегда увеличивается до того как выполняется проверка ограничения, следовательно, конечная VALUE после окончания цикла будет всегда больше, чем LIMIT для положительной STEP и меньше чем LIMIT для отрицательной.

OPEN#

Для использования с микродрайвом.

Однако возможно открывать и закрывать потоки данных следующим образом: используя строку **PRINT #5;"WORKS"** и она не будет работать до тех пор, пока поток данных не 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 от 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 то же время порт 254 может использоваться для управления звуковым динамиком, и следующая программа показывает, как это сделать.

С помощью этой 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. (Оператор вставьте вместо черточек.)

```
10 OUT 254,23:--------------------: OUT 254,71: GO TO 10
```

© YA PRINT

В программе оператор может быть, например, таким: PRINT, RANDOMIZE, LET A=B или POKE 0,0 в программе OUT 254,23 не используется динамик, а в OUT 254,71 используется. Операция повторяется очень быстро, что даст звук.

OVER N

Команда цвета.

При OVER 1 последовательность печати будет выполняться объединением по команде XOR с уже существующим символом. При выполнении XOR бит перебрасывается - установленный бит сбрасывается и не будет печати, при установке бита будет печать.

При OVER 0 все биты в данной области символа будут напечатаны.

При OVER 1, если символ или строка напечатаны дважды, то они исчезнут, т.е.

```
10 OVER 1: PRINT "A"; CHR$8;"A"
```

где CHR$8 - шаг назад.

Бит 1 в переменной PFLAG - управляющий флаг OVER.

PAPER N

Команда цвета. Может быть постоянной или временной. Команда может использоваться с N=0-9.

Использование N от 0 до 7 очень простое.

При N=8 устанавливается режим высвечивания, и цвет бумаги области передаваемого символа остается неизменным.

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 – зелёный, голубой, желтый, белый - бумага будет чёрной, а при тёмных чернилах - чёрный, синий, красный, фиолетовый - бумага будет белой.

Фактическим действием PAPER является:

для N=8 устанавливается бит 3,4,5 переменной MASKP и для N=9 бит 7 переменной PFLAG, а для N=0-7 биты 3,4,5 переменной ATTRP.

PAUSE N

Эта команда HALT останавливает Z80 на N прерываний. Единственной работой, которая выполняется за время паузы, является опрос клавиатуры. Период паузы закончится после N прерываний или при нажатии клавиш.

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, а использование OVER 1 может привести к UPLOT, INVERSE 1 - к NOPLOT.

POKE N,M

Команда позволяет вводить значения по адресам 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. Элементы отделяются друг от друга разделителями : ";" - элементы выводятся подряд, "," - выполняется табуляция к следующей половине строки. Позицию экрана можно также набирать с помощью TAB и AT.

Элементы могут быть элементами цвета, действие которых будет временно, числовыми выражениями или строковыми.

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 берётся из двух 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 см. ниже 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.

VA PRINT

Заметьте, что в оператор READ можно включать неописанные ранее переменные. Системная переменная DATADD, 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 адрес перед областью программы. Если N определено, то указывает на адрес строки Бейсик с номером N, если она существует или ближайшую после неё. Если N больше чем 9999 (самый большой разрешенный номер строки) или "ушел" за последний номер программы, то DATADD указывает на последний адрес в области программы.

RETURN

По команде извлекается верхнее значение из стека GO SUB. Если обращение корректно, тогда интерпретатор будет выполнять это утверждение следующим. Иначе будет выдано сообщение об ошибке. "RETURN без GO SUB".

RUN N

Позволяет выполнять программы Бейсик. Если N не задано, то будет принято начало с первой строки. Когда N определено, то интерпретатор находит область программы строки с этим номером или первую строку после неё, если строка N не существует. Команда инициирует требуемые указатели путём 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 передаёт номер ошибки в системную переменную ERRNR, которая располагается по адресу 23610 (5C3A). Номер ошибки всегда меньше на 1, чем код ошибки, который появляется перед сообщением об ошибке.

Поскольку ошибка имеет место, интерпретатор останавливает выполнение программы и переходит на выполнение программы обработки ошибки. Затем номер ошибки передается в ERRNR, выбирается адрес возврата в командный режим (адрес расположен в 4867 (1303)). Этот адрес возврата всегда существует и расположен в 2-х адресах ниже стека GO SUB. Если отсутствуют текущие циклы GO SUB, тогда стек GO SUB будет пустым.

Но так как различные подпрограммы вызываются, адрес возврата опускается в стек и подымается опять, когда программа завершается.

Адрес возврата всегда указывается системной переменной ERRSP, которая расположена в адресах, 23613 и 23614 (5C3D-5C3E).

Заметьте, что команды RUN и GO TO не очищают стек GO SUB, и если программа останавливается без завершения выполнения подпрограммы, тогда ERRSP постепенно указывает все ниже в памяти (на 3 байта для каждой подпрограммы). Однако команда CLEAR восстанавливает ERRSP.

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 при загрузке с ленты" появляется, если запись на ленте не точно соответствует оригиналу.

VA PRINT

ABS

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

ACS

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

AND

Это двоичная операция и поэтому она требует 2 операнда. Если операнды логически истинны, то операция полностью истинная. Однако, если один или оба операнда ложны, то операция полностью ложна. Числовое выражение, которое является ложным, будет иметь 0 значение, т.е. 0. Когда результат полностью истинный, именно 1-ый операнд возвращается пользователю.

ASN

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

ATN

Аргумент X берётся как тангенс, и функция возвращает величину в радианах. ATN является первой из 4-х

функций, определенных в SP 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

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 BIN 00000001: PRINT A: NEXT A
```

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

CHR$ N

Эта функция возвращает пользователю символ, как строку для данного кода. Следующая строка показывает, как распечатать набор символов. Клавиши CONTINUE и ENTER должны будут использоваться 6 раз, чтобы установить коды управления цветом.

```
10 FOR A=0 TO 255: PRINT CHR$ A: NEXT A
```

CODE

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

COS

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

EXP

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

FN

В SP около 26 числовых и 26 строчных функций. За знаком функции должна следовать 1 буква или 1 буква и знак $, а затем требуемые аргументы в скобках.

IN

Аргумент X используется как адрес порта, и считываемое значение будет в диапазоне 0-255. Если порт не используется, величина будет 255.

INKEY$

Функция опрашивает клавиатуру без обращения к INPUT. Если нажата 1 клавиша, то длина входной строки определяется этой клавишей. Команда распознает как символ на клавише CAPS LOCK или без неё. Другие комбинации дают пробел или ?. Если ни одна клавиша не нажата, или нажата более чем одна, то длина входной строки - (0).

INT X

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

LEN A$

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 логарифм e для величины X. Представляет собой другую функцию вычисленную в SP использованием многочлена Чебышева.

NOT

Дает логическое состояние значений в SP. Однако результат инвертируется, и логический результат даётся просмотром "NOT NOT X".

Для X=2 NOT X даст ложь, а NOT NOT X истинно.

Для X=0 NOT X истинно, а NOT NOT X ложно.

© VA PRINT

OR

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

Истинный результат равен 1, если 2-ой операнд не равен 0 и величине 1 -го операнда, если второй операнд равен 0.

PEEK X

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

PI

Представление с плавающей точкой. PI/2 является константой в таблице постоянных в ROM памяти. Эта величина выбирается, удваивается и передается пользователю. Величина e=3.14…

POINT X,Y

Имеет форму /X-ТОЧКА,Y- ТОЧКА/ и возвращает величину 1, если элемент изображения в этом положении установлен, и величину 0 если сброшен.

RND

В SP случайные числа выдаются генератором псевдослучайных чисел. Значение системной переменной SEED, находящейся в адресах 23670 и 23671, выбирается, модифицируется и возвращается при каждом обращении к RND. Случайное число, возвращенное пользователю, является новой величиной SEED, деленной на 65535. SEED сбрасывается в 0 при инициализации SP. С этого времени она является последовательностью целых чисел 0-65535. Величина не будет повторяться до тех пор, пока все числа не использовались.

SCREEN$

Формат функции SCREEN$(строка, столбец). Функция возвращает строку, содержащую символ для заданной позиции. Распознает символы с кодами в диапазоне 32-127, когда они нормальны или инвертированные.

Работа заключается в сравнении содержимого 8 байтов для области данного символа относительно 8-ми байтового представления, имеющего место в генераторе символов. Функция возвращает 0 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 "a"; 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 корень аргумента X.

STR$

Аргумент для этой функции рассматривается как числовое выражение.

Операция включает в себя числовое выражение, печатает его в рабочей области так, как оно будет выведено на TV - экране, если требуется, я затем возвращает пользователю параметры строки.

TAN X

Вычисление тангенса угла. Аргумент - в радианах.

USR-число

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

Когда команда вызывается, то Z80 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.

VA PRINT

В SP числовые значения передаются через пару регистров BC.

USR-строка

Возвращает адрес требуемого символа из графики. Аргумент состоит из 1 -го символа, и он должен быть в диапазоне "A-U".

Графическая область будет начинаться с адреса 32600, но лучше рассматривать её по адресу UDG (по адресам 23675 и 23676). Если область графики задается в адресе 32600, тогда этот адрес задается с помощью USR "A" (SPECTRUM-16).

8 адресов 32600-32607 содержат представление символа A. USR "B" возвращает адрес 32608, USR "C" даёт 32616 и т.д., до 21 графического символа (для SPECTRUM-16).

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

Позволяет вычислять выражения, которые содержатся в строках.

VAL$

Требует строку в качестве аргумента, и эта строка должна быть заключена в кавычки. Функция затем возвращает пользователю результат в виде строки.

2.3. 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, следующими двумя символами. Первый символ даёт номер строки, второй номер столбца.

CHR$ 23 - TAB

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

© VA PRINT

3. МИКРОПРОЦЕССОР Z-80

3.1. ВВЕДЕНИЕ

Микропроцессор Z-80 - микросхема с 40 выводами, пронумерованными от 1 до 40. Рассмотрим функции выводов.

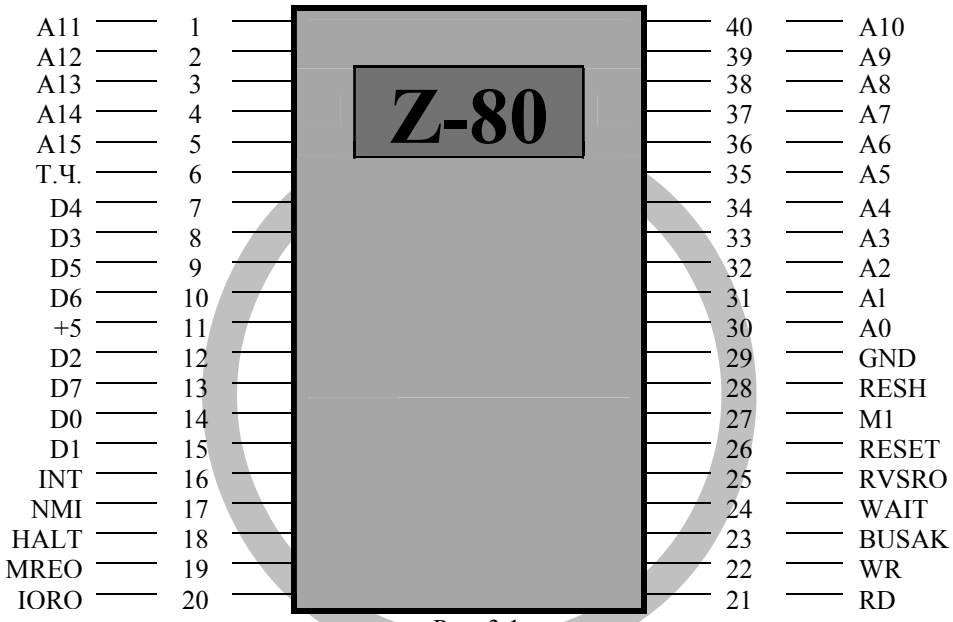


Рис. 3.1

Выв. 11 - напряжение питания +5В.

Выв. 29 - общий.

Выв. 06 - тактовый вход. В SP используется тактовая частота 3.5 МГц.

Выв. 07-10, 12-15 - информационная шина.

Выв. 1-5, 30-40 - адресные шины от МП к памяти.

Оставшиеся 13 выводов присоединены к линиям, которые несут контрольные сигналы.

Выв. 21 - линия считывания RD. Эта линия становится активной, когда байт информации должен быть считан из памяти.

Выв. 22 - линия записи WR. Эта линия активна, когда байт информации должен быть записан от МП в память.

Выв. 19 - запрос памяти MREQ. Эта линия активна в тех случаях, когда байт информации должен следовать как из процессора, так и в него.

Байт информации следует из памяти в соответствии с адресом, помещенным на адресной шине. Далее, в соответствии с откликом на сигналы RD и MREQ соответствующая микросхема памяти помещает байт информации на информационную шину, с которой эта информация в дальнейшем считывается МП. Байт данных записывается в память МП, который помещает требуемые адреса на адресную шину. Шины MREQ и WR активизируются, и байт данных выставляется на шину данных. В дальнейшем, если к микросхеме памяти продолжается обращение, байт данных будет скопирован в память.

Выв. 28 - линия регенерации RESH. Она используется для регенерации динамической памяти. Частично используется для регенерации TV сканирующих сигналов.

Выв. 27 - выборка памяти M1 наиболее важная линия, которая активизируется или по машинному коду, или по соответствующему байту информации, выбираемому из памяти.

Выборка инструкции или байта данных требует, чтобы все 3 линии M1, MREQ и RD были активизированы. В то же время выборка байта из ячейки памяти, которая является другой частью памяти, требует, чтобы только MREQ и RD были активизированы. Время, необходимое для выборки инструкции, составляет 1.14 мкс, что является 4 тактовыми циклами или Т-состояниями.

Выв. 20 - линия вход/выход IORQ. Эта линия активна при выполнении команд IN или OUT.

Выв. 18 - останов HALT. Линия активизируется при выполнении команды HALT.

Выв. 25 - линия запроса BUSRQ. Z-80 позволяет внешним устройствам использовать адресную и информационную шину в режиме пропуска цикла. Запрос МП пропустить следующий цикл выполняется внешними устройствами путем активизации этой линии.

Выв. 23 - линия подтверждения BUSAK. МП подтверждает запрос остановки после выполнения дальнейших команд и активизирует эту линию.

Оставшиеся 4 вывода находятся под контролем пользователя.

Выв. 26 - линия сброса RESET. Используется для инициализации МП. Она активизируется при включении питания. Сброс может быть использован в SP соединением линий RESET и GND.

VA PRINT

Выв. 24 - линия ожидания WAIT. "Медленная" память может требовать большего времени для цикла считывания или записи и об этом сообщает МП путем активизации линии WAIT.

Выв. 17 - "немаскируемое прерывание" NMI. Активизация этой линии приводит к остановке выполнения МП текущей программы, вместо этого МП выполняет программу прерывания, записанную специально для этой цели. В SP немаскируемое прерывание требует системного сброса, который выполняется записыванием 0 по адресу 23728.

Выв. 16 - "маскируемые прерывания" INT. В SP сканирование клавиатуры и обмен в режиме реального времени называется "управляемым прерыванием". Это означает, что электроника системы каждые 1/50 секунды активизирует ШТ, вызывая остановку выполнения МП основной программы и, вместо неё, выполнение программы сканирования клавиатуры. Способность Z80 реагировать на 1NT может управляться программистом специальными машинными командами.

3.2. ЛОГИЧЕСКИЙ ВЗГЛЯД НА Z80

Z80 состоит: устройство управления, регистр команд, программный счетчик, 24 регистра пользователя и АЛУ.

Устройство управления

В функции УУ входит пересылка байта данных в Z80, который, в свою очередь, результаты (байты данных) посылает из процессора в правильном направлении и убеждается, что операция успешно завершена.

В Z80 УУ выдаёт огромное число управляющих сигналов, а также управляющие сигналы на линии RD, WR, MREQ.

Регистр команд (РК)

Термин "регистр" используется для описания простой ячейки внутри Z80. Регистр содержит 8 бит (1 байт). В Z80 имеется большой блок регистров, и пересылка байтов данных к регистрам и от них является простейшей и наиболее важной чертой программирования в машинных кодах.

РК - специальный регистр, где МП содержит копию выполняемой текущей команды. Одной из черт набора команд Z80 является то, что определенные инструкции содержатся в 2-х байтах данных. В этих случаях регистр команд содержит каждую команду по очереди.

Программный счётчик (ПС)

ПС - это пара регистров, которые используются совместно. ПС может содержать 16-ти битные значения. Программный счётчик имеет специфическую цепь в сохранении адресов, либо текущей выполняемой команды, в зависимости от того, был ли изменен ПС или нет.

Когда инструкции выбираются, УУ использует адреса в ПС как адрес памяти, содержащей команду, которая должна выполняться следующей, и затем УУ наращивает ПС.

Действие ПС очень похоже на переменную PPC интерпретатора Бейсик, которая содержит номер текущей строки Бейсик и также наращивается.

Регистры пользователя /основные регистры/ (РП)

Имеется 24 регистра пользователя. Все регистры однобайтовые, обычно используются парами.

24 РП МП Z80 как 12 регистровых пар. Номера битов показаны.

Основной набор		Альтернативный набор	
76543210	76543210	76543210	76543210
A	F	A'	F'
H	L	H'	L'
B	C	B'	C'
D	E	D'	E'
15……0			
IX			
IY			
SP			
I	R		

Регистр A называется аккумулятором.

Регистр F - это флаговый регистр. Он часто рассматривается, как набор 8-ми флаговых битов, связанных вместе, а не как отдельный регистр. Флаг установлен - 1, флаг сброшен - 0. Программист обычно имеет дело с 4-мя основными флагами. Это флаг нуля, флаг знака, флаг переноса и флаг чётности переполнения. Дополнительные флаги используются УУ и не могут быть использованы программистом непосредственно.

Регистровая пара HL. При записи регистровой пары HL сначала указывается младшая часть (LOW), а затем старшая (HIGH).

Память 64K может быть рассмотрена как 256 страниц по 256 адресов в каждой. В этом случае значение старшего байта указывает на используемую страницу.

В МП регистровая пара HL является одной из 3-х регистровых пар, которая используется в качестве адресных

VA PRINT

регистров. Однако регистровая пара HL является наиболее важной. HL-пара также может быть использована для хранения 16-ти байтного числа и существует определенный ряд арифметических операций, которые могут быть выполнены с этими числами. H-регистр и L-регистр могут быть использованы и как отдельные регистры, хотя с ними может быть выполнено ограниченное число операций.

Регистровые пары BC и DE.

Эти пары используются главным образом как адресные регистры. Регистры могут быть использованы как одинарные. Регистры BC рекомендуется использовать в качестве счётчика циклов.

Набор альтернативных регистров.

Z80 имеет альтернативный набор регистров для A,F,H,L,B,C,D,F. Они обозначаются A',F', и т.д.

Существуют две специальные команды, которые позволяют обменивать основной и альтернативный набор регистров. После обмена для Z80 альтернативный набор становиться основным, а основной - альтернативным.

Альтернативные наборы регистров часто используются для сохранения среды, когда запускается независимая задача.

Особенным примером для программы SP с монитором 16K является ситуация, когда при вычислениях с плавающей точкой используется H и L регистры для сохранения адреса возврата. Если эти регистры испорчены, то возвращение в Бейсик невозможно.

Регистровые пары IX и IY - используются для выполнения операции, которая включает индексацию. Это даёт возможность работать со списком или таблицей. Начальный адрес списка или таблицы должен быть первоначально занесен в подходящую пару регистров IX или IY. В программе SP с монитором 16K IY пара содержится по 10-му адресу 23610 (5C3A), что является начальным адресом таблицы системных переменных. IX пара широко используется как указатель в программах обработки команд LOAD, SAVE, VERIFY, MERGE.

Указатель стека - это адресный регистр. Он используется для указания в памяти области машинного стека и всегда рассматривается как одинарный двухбайтовый регистр. Z80 использует стек, заполняемый в памяти сверху вниз. Аналогией является магазин автомата - по принципу: "последний пришел - первым ушел".

Указатель стека используется для указания различных размещений в области стека в каждом случае. Указатель стека всегда содержит адрес, куда последний раз была произведена запись. Поэтому УУ сначала уменьшает значение указателя стека, а затем помещает туда значение. Пересылка в стек 2-х байтовая и поэтому указатель стека должен быть дважды уменьшен при помещении в стек, и дважды увеличен при выборке из стека. Машинный стек обычно используется как место сохранения адресов возврата, но можно использовать его и как рабочую область.

Регистр I - регистр вектора прерываний, используется для размещения адресов устройств ввода-вывода. Однако в SP эта возможность не используется и I - регистр используется для генерации светового TV - сигнала.

Регистр R - регистр регенерации памяти. Он является простым счётчиком, который увеличивается каждый раз при выполнении цикла регенерации. Значение в регистре циклически изменяется от 0 до 127.

АЛУ

В АЛУ выполняются арифметические и логические операции. Возможно выполнение сложения и вычитания, а также АЛУ способно выполнять большое количество битовых операций и устанавливать флаги, чтобы показать результат.

3.3. СТРУКТУРА ПРОГРАММ В МАШИННЫХ КОДАХ

Программа должна существовать как набор машинных команд и данных, размещенных в последовательных адресах памяти. В SP 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
```

VA PRINT

Это представление является иллюстрацией программы в машинных кодах. Но очень громоздко и приводит к ошибкам. Поэтому лучше использовать шестнадцатеричную запись кодов или программу на Бейсике, которая печатает адреса в десятичной 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$(48+H+7*(H>9)); CHR$ (48+L+7*(L>9))
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 из приложения, можно представить эти машинные команды в следующем виде:

LOCATION 0 - DI	запрещение маскированного прерывания
LOCATION 1 - XOR A	сложение по модулю 2 регистра A
LOCATION 2 - 4 LD DE,+FFFF	загрузка регистровой пары DE
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 ассемблеровской. Программист вначале обычно составляет программу в мнемоническом виде на языке ассемблера, а затем, в процессе трансляции, это представление преобразуется в 10 или 16-ричное.

Поэтому такое написание очень удобно при записи программ.

©

VA PRINT

4. МАТЕМАТИКА МАШИННОГО ПРОГРАММИРОВАНИЯ

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

4.1. ШЕСТНАДЦАТИРИЧНОЕ ПРЕДСТАВЛЕНИЕ

Принцип представления состоит в описании чисел по основанию 16. 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
0111	7	7
1000	8	S
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-ми знаками.

Примеры:

0000 0000 (2-ное) - 00(16)
0100 1111 (2-ное) - 4F(16)
0000 0000 0000 0000 (2-ное) - 0000(16)
0100 1100 1010 1111 (2-ное) - 4CAF(16)

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

```
10 INPUT "DECIMAL NUMBER",D
20 IF D>65535 THEN GO TO 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$(H(A)+48+7*(H(A)>9))
150 NEXT A
160 PRINT "HEX A DECIMAL",H$
170 PRINT
180 GOTO 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$=32 THEN GO TO 20
40 LET D=0
50 FOR A=1 TO 4
60 IF H$(A)=CHR$ 32 THEN GO TO 100
70 IF H$(A)<"0" OR H$(A)>"9" AND H$(A)<"A" OR H$(A)>"F" THEN GO TO 120
```

VA PRINT

```
80 LET D=D+16^(4-A)*(CODE H$(A)-48-7* (CODE H$(A)>57))
90 NEXT A
100 PRINT "HEX A DECIMAL",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=30,874

или попарно:

десятичный эквивалент - 78*256=30720
9A*1=154
789A=30,874

4.3. АБСОЛЮТНАЯ ДВОИЧНАЯ АРИФМЕТИКА

Одинарный регистр или адрес 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 вычитаниях к значению меньше 0 прибавляется 256.

Это показано на следующих примерах.

252+44 даёт 40 или FC+2C даёт 28; 87-200 даёт 143 или 57-C8 даёт 8F.

Флаг переноса действует в большинстве операций, требующих переноса. Для дальнейшей детализации см. главу 5.

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.

4.4. ДОПОЛНИТЕЛЬНАЯ АРИФМЕТИКА

Принцип дополнительной арифметики очень прост, но при использовании её программы результат может быть очень неожидан.

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.

Заметим, что возможно распространить принципы дополнительной арифметики на 16-битовые числа. В этом случае интервал возможных значений чисел будет от -32678 до 32767 десятичных.

Преобразование отрицательных 10-ных чисел в дополнительную двоичную или 16-ричную форму просто.

Порядок следующий:

1. Найдите двоичную форму для абсолютного двоичного значения, например: 54 - 00110110.
2. Получите обратный код - изменить 0 на 1 и наоборот 0011 0110 - 1100 1001.
3. Прибавьте 1 – 1100 1001 + 1 = 1100 1010.
4. Используйте двоичную форму или переведите в 16-ричную. -54 в дополнительном коде - 1100 1010, или CA.

Операцию можно провести в обратном порядке. Для перевода числа в дополнительном коде в 10-ный вид.

4.5. ИНТЕГРАЛЬНОЕ ПРЕДСТАВЛЕНИЕ

Интерпретатор Бейсик в системе SP использует 5 байт для представления чисел. Любое целое в интервале -65535 - +65535 включительно записывают в интегральной форме, в то время как любые дробные или целые числа вне этого диапазона записываются в 5 байт плавающего числа.

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

© VA PRINT

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 N<>INT N OR N<-65535 OR N>65535 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
```

2-ное	10-ное	16-ное
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. Крайний левый бит в двоичном знаковый бит.

В приведенной выше программе указатель 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 -65535. Это значение не соответствует интегральной форме 0,255,0,0,0

VA PRINT

4.6. ПРЕДСТАВЛЕНИЕ С ПЛАВАЮЩЕЙ ТОЧКОЙ

5 байт, используемых в системе SP для представления чисел с плавающей точкой, позволяют хранить числа в диапазоне 0.29e-38 - 1.7e38 (приблизительно).

Нулевое значение всегда хранится как 5 нулевых байтов. Все остальные значения хранятся: экспонента в первом байте, мантисса в остальных 4-х. Теория описания чисел в терминах экспонент и мантисс первоначально появилась в десятичной записи, поскольку в ней лучше работать.

Нахождение экспоненты и мантиссы десятичного числа - это, в сущности, перевод числа в е-формат. Рассмотрим число 12345, которое в EXP-формате может быть представлено как .12345 E+5. Для получения мантиссы 10-ная точка двигается налево до тех пор, пока она не становится перед значащей старшей цифрой. Экспонента - число необходимых сдвигов, мантисса - дробное десятичное число. Экспонента - +5; мантисса - 0.12345.

Поскольку система SP имеет дело скорее с двоичными числами, чем с десятичными, подобные операции могут быть рассмотрены для двоичного числа.

Рассмотрим двоичное число 0001 1111, что эквивалентно +31. Двоичная точка лежит справа от 8-го бита и требует +5 сдвигов, чтобы перевести её перед старшим значащим битом. Поэтому экспонента =+5; мантисса =.1111 1000 ……

В SP производятся ещё некоторые действия для экспоненты: - положительная экспонента всегда увеличивается на 10-ное значение +128 (16-ное 80), в нашем примере +5+128=133.

Для мантиссы: первый бит мантиссы всегда будет единичным, поэтому его можно использовать как знаковый - для положительного числа он будет 0, а для отрицательного останется 1.

В нашем примере мантисса становится .0111 1000… И теперь остается поместить число в 5 байтов. Неиспользуемые байты мантиссы зануляются. В десятичном плавающем виде +31 записывается как 133,120,000, в шестнадцатеричном 85,78,00,00,00 и, если хотите, в двоичном 1000 0101 0111 1000 0000.

Подобная операция позволяет переводить и отрицательные, и дробные числа, хотя и не так просто. Следующая программа показывает 10-ную форму плавающих чисел. Включение строки 30 гарантирует, что всегда получится плавающая форма числа.

```
10 INPUT N
20 IF N=0 THEN GO TO 40
30 LET N=N-.2E-38
40 PRINT "NUMBER CHOSEN";N
50 PRINT
60 PRINT "EXP";TAB 9;"MANTISSA"
70 LET V=PEEK 23627+256*PEEK 23628
80 PRINT PEEK (V+1);TAB 9;
90 FOR A=2 TO 5
100 PRINT PEEK (V+A); CHR$32;
110 NEXT A
120 PRINT
130 GO TO 10
```

Приведенная выше программа даёт следующие результаты.

	EXP	MANTISSA
1	=129	0 0 0 0
1	=130	0 0 0 0
35456	=144	10 128 0 0
-1	=129	128 0 0 0
-35456	=144	138 128 0 0
6.333	=131	74 167 239 158

Интересно заметить, что 0.5 и 1/2 дадут различные значения.

.5	=127	127 255 255 255
1/2	=128	0 0 0 0

© VA PRINT

5. НАБОР СИСТЕМЫ 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 A 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. Двухбайтовая константа +DDDD, то есть число A в диапазоне 0000-FFFF 16-ричное и 0-65535 10-ное. Те команды, которые требуют за собой двухбайтовую константу, имеют мнемоническую приставку '+DDDD'. Например: LD HL,+DDDD.

3. Двухбайтовый адрес, т.е. число A в диапазоне 0000-FFFF 16-ричное, 0-65535 10-ное, т.е.число, которое используется как адрес памяти. Те команды, которые требуют за собой двухбайтовый адрес, имеют приставку 'ADDV'. Например: JP ADDV.

4. Однобайтовая константа смещения E, т.е. число A в диапазоне 00-FF 16-ричное -128 - +127 10-ное. Число всегда представлено в дополнительной арифметике. Те команды, которые требуют за собой однобайтовую константу, имеют мнемоническую приставку 'E'. Например: JR E.

5. Однобайтовая индексирующая константа смещения +D. Т.е. число A 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, т.е. 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 байта данных, сопровождаются мнемониками '+D' и '+DD'. Например: LD (IX+D),+DD.

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.

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

Группа 1

Команда: нет операции.

Мнемоника	*16-ный код*
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.

Программа Бейсика, демонстрирующая эту команду, приведена на странице 56.

Группа 2

Команда загрузки регистра константами.

Мнемоника	*16-ный код*
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 включают загрузку пары регистров двухбайтовыми константами.

Мнемоника	*16-ный код*
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, т.е. D,C,E,X,Y,P, во второй

VA PRINT

байт - старший регистр, т.е. 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 также и двухбайтовым числовым значением и 2-я отдельными 1-байтовыми числовыми значениями.

Команды этой группы не изменяют флагов. Программа Бейсик для демонстрации этих команд приведена на странице 57.

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 A,R	LD H,R	LD L,R	LD B,R	LD C,R	LD D,R	LD 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.

Мнемоника	*16-ный код*
`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 Б. Команды копирования регистровая пара - регистровая пара

Существует только 3 команды этой подгруппы. И они копируют значения в указатель стека.

Мнемоника	*16-ный код*
`LD SP,HL`	`F9`
`LD SP,IX`	`DD F9`
`LD SP,IY`	`FD F9`

Эти команды не изменяют содержимое флага. 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,DE и её заменяют, используя LD H,D и LD L,E. Или содержимое 1-ой регистровой пары может быть сохранено в стеке и впоследствии переписано во 2-ю регистровую пару (см. команды стека на странице 46).

Подгруппа В. Команды EX DE,HL

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

Мнемоника	*16-ный код*
`EX DE,HL`	`EB`

Эта очень полезная команда позволяет обменивать значения в паре DE с содержимым пары HL. Значение флагов не изменяется. Команда используется, когда необходимо, чтобы адрес или 2-х байтовая числовая константа из пары DE была записана в пару HL, но содержимое последней не было потеряно.

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

Мнемоника	*16-ный код*
`EXX`	`D9`
`EX AF,A'F'`	`08`

Команда EXX вызывает переключение регистров H,L,B,C,D,E на регистры H',L',B',C',D',E'. Команды EX AF.A'F', как следует из мнемоники, переключает регистр A и F на A' и 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.

Программа Бейсик, демонстрирующая эти команды, находится на странице 58.

Группа 4. Команды загрузки регистров из памяти

В систему Z80 входит много команд, позволяющих находить данные в памяти и затем загружать их в основной регистр. Все эти команды требуют, чтобы программист указал адрес или пару адресов байтов, откуда данные должны быть скопированы в регистр получателей данных.

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

YA PRINT

Виды адресации:

- ✓ косвенная - 2-х байтовый адрес уже размещен в адресной регистровой паре.
- ✓ непосредственная - действительный адрес указан в двух байтах за кодом операции.
- ✓ индексная - адрес данных должен быть высчитан путём сложения значения совмещения D с основным адресом, уже содержащимся в регистровой паре IX или IY.

Подгруппа А. Команды, использующие непосредственную адресацию

Мнемоника	*16-ный код*
LD A,(ADDR)	3A ADDR
LD HL,(ADDR)	2A ADDR - обычная форма
	ED 6B ADDR - необычная форма
LD BC,(ADDR)	ED 4B ADDR
LD DE,(ADDR)	ED 5B ADDR
LD D,(IX+D)	DD 56 D
LD E,(IX+D)	DD 5E D

Для команд, использующих пару IY, надо изменить IX на IY и DD на FD. Ни одна из команд этой группы не меняет флаги. Программа Бейсик этих команд приведена на странице 58.

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

Мнемоника	*16-ный код*
LD (ADDR),A	32 ADDR
LD (ADDR),HL	22 ADDR - обычная форма
	ED 63 ADDR - необычная форма
LD (ADDR),BC	ED 43 ADDR
LD (ADDR),DE	ED 53 ADDR
LD (ADDR),IX	ED 22 ADDR
LD (ADDR),IY	ED 22 ADDR
LD (ADDR),SP	ED 73 ADDR

Приведенные команды приводят только непосредственную адресацию и важно заметить, что нет команд для записи в память констант. Если это необходимо, константа должна быть загружена в регистр A или каким-либо другим путем. Затем команды типа LD (ADDR),HL - фактически двойные команды: LD (ADDR),L и LD (ADDR+1),H. 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 метка пары последних адресов памяти, используемой для хранения текущего значения верхней границы памяти. Выборка текущего значения может быть позднее выполнена командой 4 группы, например, LD HL,(RAMTOP).

Подгруппа Б. 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.

Мнемоника	*16-ный код*
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 В. Команды индексной адресации

Мнемоника	*16-ный код*
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),E	DD 73 D
LD (IX+D),+DD	DD 36 D DD

Для команд, использующих IY регистровую пару, надо изменить IX на IY и DD на FD. Программа иллюстрирует эти команды на странице 60.

© VA PRINT

Группа 6. Команды сложения

Эта группа команд выполняет арифметические действия. Команды сложения позволяют программисту прибавить (в абсолютной 2-й арифметике) заданное число к регистровой паре, регистру или индексному адресу памяти.

Команды этой группы могут быть поделены на 3 группы, каждая из которых содержит 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 2-ное переполнение регистра или байта памяти. Команды ADD и ADC меняют флаг переноса, а INC - не меняет.

Подгруппа А. Команда ADD.

Мнемоника	*16-ный код*
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 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 A,(HL)	86
ADD A,(IX+D)	DD 86 D
ADD IX,SP	DD 39

Для команд, использующих IX регистр, надо, поменять IX на IY и DD на FD. Команды ADD, приведенные выше, очень просты. Команда устанавливает или сбрасывает флаг переноса в зависимости от того было или нет 2-ное переполнение регистра. Или пары регистров.

Это показано на следующих примерах:

1. Регистр 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 Б. Команда INC

Команды этой группы позволяют прибавить 1 к 8-ми битовому регистру, ячейке памяти или 16-ной паре регистров. Во всех случаях флаг переноса игнорируется, он не изменяется.

Мнемоника	*16-ный код*
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 HL	23
INC BC	03
INC DE	13
INC SP	33
INC IX	DD 23
INC IY	FD 23

Подгруппа В. Команда ADC

Мнемоника	*16-ный код*
ADC A,+DD	CE DD
ADC A,A	8F
ADC A,H	8C
ADC A,L	8D
ADC A,B	88

© VA PRINT

```
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 вместе с текущим значением флага переноса. Все команды этой группы меняют флаг переноса. Он сбрасывается, если операция ADC не дает 2-ного переполнения и устанавливается, если даёт. Примеры показывают это:

1. регистр A=F0

регистр A=60 16-ной ADC A,B регистр B=90 флаг=0

2. регистр A=27

регистр A=A8 флаг=1 ADC A,B регистр B=7E флаг=1

Программа иллюстрирует эти команды на странице 61.

Группа 7. Команды вычитания

Позволяют вычесть заданное число из одинарного регистра, пары регистров или индексного адреса 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 2-ный заём. Команда DEC оставляет флаг переноса без изменения.

Подгруппа А. Команды SUB

Мнемоника	*16-ный код*
SUB +DD	D6 DD
SUB A	97
SUB H	94
SUB L	95
SUB B	90
SUB C	91
SUB D	92
SUB E	93
SUB (HL)	06
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 т.д., поскольку все команды используют регистр A.

В Z80 команда SUB выполняет "истинное" абсолютное 2-ное вычитание, как показано в примерах. Флаг переноса сбрасывается, если первоначальное значение регистра A больше или равно вычитаемому, и устанавливается если меньше.

Регистр A=DC, B=2A SUB B - регистр A=B2 флаг=0

Регистр A=AA, B=DC SUB B регистр A=CE флаг=C

Подгруппа Б. Команды DEC

Команды этой группы позволяют вычесть 1 из содержимого 8 битового регистра, ячейки памяти или 16-битной пары регистров.

Во всех этих случаях флаг переноса не изменяется.

Мнемоника	*16-ный код*
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

© VA PRINT

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

Подгруппа B. Команды SBC

Мнемоника	*16-ный код*
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,(IY+D)	FD 93 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. Программа, демонстрирующая команды этой группы, находится на странице 62.

Группа 8. Команды сравнения

Команды этой группы используются очень часто во всех программах. Они позволяют программисту сравнить значение, находящееся в регистре A, с константой, значением в регистре и области памяти. Команды выполняют операцию вычитания без переноса, без запоминания результата вычитания и только устанавливают флаги в регистре F. Первоначальное значение в регистре A не изменяется. Флаг переноса устанавливается, как и при операциях вычитания. Сравнение при больше или равно сбрасывает флаг переноса, а при меньше - устанавливает. Команды этой группы - команды одинарного сравнения, а блочное сравнение рассмотрено на странице 48. Команды:

Мнемоника	*16-ный код*
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.

1. Регистр A=31 CP B оставляет регистр без изменения
   B=30 и сбрасывает флаг переноса 31>00
2. Регистр A=30 CP B оставляет регистр без изменения
   B=30 и устанавливается флаг нуля 30=30
3. Регистр A=01 CP B оставляет регистр без изменения
   B=30 и устанавливает флаг переноса 01<30

Программа, демонстрирующая эти команды, находится на странице 63.

Группа 9. Команды логики

Это команды AND, OR, XOR. Они применяются к регистру A и другой заданной переменной. Операция выполняется побитно, и 8-битовый результат возвращается в регистре A.

Подгруппа A. Команда AND

Логическая операция выполняется над двумя двоичными цифрами и рез.-1 только, если оба тестируемых бита установлены. В противном случае результирующий бит 0. Пример показывает, как команда выполняется над восемью

© VA PRINT

отдельными битами.

Двоичн.	10101010	16-ный AA
	AND	AND
	11000000	C0
Результат	10000000	80

В этой подгруппе следующие команды:

Мнемоника	*16-ный код*
AND +DD	A6
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 SP биты 0-2 системной переменной ATTRP содержат информацию о постоянной цвета чернил. Когда цвет чернил изменяется, старый цвет удаляется использованием команды AND и новый цвет вводится по команде ADD, т.е.

LD A,(ATTRP)	находится системная переменная
AND +F1	демаскируется старый цвет
ADD +NEWCOLOR	добавляется новый цвет
LD (ATTRP),A	сохраняется системная переменная

Подгруппа Б. Команда OR

Логическая операция или выполняется над 2-мя двоичными цифрами, и результат =-1, если ни одна или обе цифры установлены. В противном случае бит результата сброшен. Пример показывает, как команда выполняет операцию над 8 отдельными битами:

В 2-ной	10101010	16-ный AA
	OR	OR
	11000000	C0
	11101010	EA

В подгруппе следующие команды:

Мнемоника	*16-ный код*
OR +DD	F6 DD
OR A	B7
OR H	B4
OR L	B5
OR B	B0
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 использований команды OR. В SP биты 5,4,3 системной переменной ATTRP определяют постоянный цвет бумаги. Поэтому можно сделать цвет бумаги белым использованием команды OR +DD.

LD A,(ATTRP)	найти системную переменную
OR +38	установить биты 5,4.3
LD (ATTRP),A	сохранить системную переменную

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

В 2-ной	10101010	16-ный AA
	XOR	XOR
	11000000	C0
	01101010	6A

Команды в этой подгруппе:

Мнемоника	*16-ный код*
XOR +DD	EE DD

© VARRINT

```
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 A. Это, вероятно, вначале трудно понять, но рассмотрите пример, приведенный выше, ещё раз, а именно: AA XOR C0 дает CA.

В этом примере 2-ным операндом является байт C0, который является байтом только из битов 6 и 7.

Поэтому эффект работы операции XOR заключается в переключении битов 6 и 7 первого операнда и изменении AA в CA. 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 адрес, в то время как LD A,+00 использует два адреса. Все команды AND, OR, XOR сбрасывают флаг переноса, когда они используются. Программа, демонстрирующая эти команды, на странице 63.

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 условно зависят от состояния одного из основных флагов. И если требуется, флаги будут подробно рассмотрены.

Подгруппа А. Команды абсолютного перехода

Мнемоника	*16-ный код*
JP ADDV	C3 ADDV

Это команда перехода, при выполнении команды значение ADDV загружается в счётчик команд, и выполнение программы машинного кода продолжается с этого адреса.

Подгруппа Б. Команды переходов, которые используют косвенную адресацию

Мнемоника	*16-ный код*
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 счётчик команд. Команды этой группы часто используются, когда переход должен осуществляться на команду, адрес которой находится в таблице адресов.

Подгруппа В. Команда относительного перехода

Мнемоника	*16-ный код*
JR E	18 E

Команда позволяет сделать переход к адресам вперед на 128 байт и назад на 128 байт от текущего адреса. Заметьте, что текущий адрес фактически является адресом, идущим после E, поскольку счётчик команд уже выполнил приращение.

E всегда рассматривается как арифметическое дополнение до 2 и положительное E даёт число байт, на которое должен быть осуществлен переход, в то время как отрицательная вели на E показывает на сколько байт надо уменьшить счётчик команд.

Подгруппа Г. 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, SP влияют на флаг переноса. Если был 2-ный заём, тогда флаг устанавливается; в противном случае сбрасывается.
3. Все команды AND, OR, XOR сбрасывают флаг переноса.
4. Команды сдвига влияют на флаг переноса (см. страницу 49) команды перехода по флагу являются:

Мнемоника	*16-ный код*	
JP NC,ADDR	D2 ADDR	переход если
JR NC,E	30 E	C флаг сброшен
JP C,ADDR	DA ADDR	переход если
JR C,E	38 E	C флаг установлен

VA PRINT

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

```
.............       введите регистр A, хранящий код
CP +30              код для цифры 0
JP C,ERROR          переход, если он находится
                    за границами диапазона
CP +3A              код для":"
JP NC,ERROR         опять переход, если он находится
                    за пределами диапазона
.............       продолжение только с цифрами, т.е. диапазон 30-39
```

Подгруппа Д. 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 - 0, в противном случае он сбрасывается. Например:

6C ADD 5A даёт C6, и флаг нуля сбрасывается.

6D ADD 93 даёт 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.
2. Команды сдвига (страница 49), проверки битов, поиска блоков влияют на флаг.
3. Команды LD за исключением LD A,I и LD A,R не влияют на флаг нуля. Команды перехода:

```
Мнемоника           16-ный код
JP NZ,ADDR          C2 ADDR         переход только тогда,
JR NZ,E             20 E            когда 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";" ?
JR Z,S-COLON        переход, если это так
CP +2C              это символ"," ?
JR NZ,ELSE          переход, если нет
.............       продолжение при ","
```

Подгруппа Е. 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 виде арифметического дополнения до 2, 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 правило:

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 (стр. 49) 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 положителен
JP M,ADDR           FA ADDR переход, если результат отрицателен
```

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 действуют отдельно
.............       продолжение только с графическими символами.
```

Подгруппа Ж. Команды перехода по флагу переполнение/чётности.

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

Флаг переполнение/чётности. Этому флагу соответствует 2 бит регистра F и он 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 не относится к двоичному переполнению, а относится к арифметическому

VA PRINT

дополнению до четного числа, как показано следующими примерами.

Рассмотрите: 16-ный 0A ADD 5C даёт 66.

10-ный 10 ADD 92 даёт 102, которое правильно - нет переполнения.

16-ный 6A ADD 32 даёт 9C.

10-ный 206 ADD 50 даёт 256, которое неправильно - даёт переполнение.

Переполнение может также возникнуть при вычитаниях, а именно:

16-ный 83 SUB 14 дает 6F.

10-ный 125 SUB 20 дает 105, которое неправильно, т.е. имеет переполнение.

Флаг переполнение/четность устанавливается, когда имеет место переполнение.

Понятие четности касается числа единичных битов в заданном байте. Это показано примером:

Байт 01010101 имеет чётное равенство, и флаг устанавливается.

Байт 00000001 имеет нечётное равенство, и флаг сбрасывается.

Кратко сформулируем правило:

1. Все команды AND, OR, XOR и большинство команд сдвига дают результат, который может быть проверен на четность.

2. Все команды ADD, ADC, SBC, CP, использующие одиночные регистры и команды ADC, SBC, использующие 2 регистра, дают результат, который может быть проверен на переполнение.

3. Команда INC установит флаг, если результат будет равен 16-ным 80, и команда DEC, если результат равен 16-ным 7F.

4. Различные другие команды, а именно LD A,I и LD A,R и многие из команд блока отработки также влияют на флаг переполнение/чётности.

Команды перехода этой группы:

```
JP PO,ADDR          E2 ADDR      переход, если равенство нечётное
                                 или нет переполнения
JP PE,ADDR          EA ADDR      переход, равенство чётное
                                 или имеет место переполнение
```

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

Программы, которые используют эти команды, представлены на странице 64.

Группа 11. Команды DJNZ

Одиночная команда в этой группе является одной из самых полезных и самой широко используемой в Z80. 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 "B" 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 её в регистр B. Затем идёт 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
LD A,+5B            16-я 5B-1A=41
SUB B               печать символа (см. букву)
RST 0010            "сдвиньте" для "B", "C", и т.д.
DJ NZ,LOOP          аналогично
```

16-м кодом для этого примера является: 06, lA, 3E, 5B, 90, 17, 10, FA, 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.

Программа Бейсик, показывающая команду этой группы, представлена на странице 66.

Группа 12. Команды стека

В большинстве программ машинных кодов осуществляется активное использование машинного стека программистом в качестве места, где можно хранить данные и 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. Данные должны копироваться в строго определенную пару регистров, но важно знать, что никакой записи не производится, которая указывала бы, к какому из двух регистров какой байт данных принадлежит.

Команды *** PUSH ***

```
PUSH AF             F5
PUSH HL             E5
```

VA PRINT

```
PUSH BC             C5
PUSH DE             D5
PUSH IX             DDE5
PUSH IY             FDE5
```

Команды *** POP ***

```
POP AF              F1
POP HL              E1
POP BC              C1
POP DE              D1
POP IX              DD E1
POP IY              FD E1
```

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 адрес. Затем указатель стека уменьшается вторично, и величина из младшего регистра стека 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, C "без вреда", но она должна оставить указатель стека прежней величины, если вся программа должна выполняться, как требуется.

Подгруппа Б. Команды обмена со стеком

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

Мнемоника	*16-ный код*
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 альтернативные командам PUSH и POP в особых случаях. Например:

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

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.

Использовать дополнительный RG для временного хранения "первого", т.е.:

```
POP BC              сохраните "первый"
PUSH HL             "второй" передать в стек
PUSH BC             переведите "первый" в HL
POP HL              ?
```

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

Подгруппа В. Команды CALL

Команды машинного кода CALL эквивалентны команде Бейсик "GO SUB". Команды включаются в эту группу, т.к. микропроцессор использует стек машины, т.е. 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:

Текущее значение счётчика команд, т.е. адрес первой ячейки после ADDR, сохраняется в стеке. Указатель стека изменяется как в случае команды PUSH. Старший байт счётчика команд сохраняется в стеке над младшим

© YA PRINT

байтом.

Адрес затем записывается в счётчик команд, и выполнение программы продолжается.

Подгруппа Г. 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 счётчике 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 GO SUB, 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               дублирование входного значения
LD E,A              и пересылка в регистр E
LD HL,+TBL-BAS      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 Д. 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.

Команды подгруппы:

```
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.

Программа Бейсик, которая показывает команды этой, группы находится на странице 67.

VA PRINT

Группа 13. Команды сдвига

Набор команд Z80 имеет большое число команд для сдвигов битов определенного байта. Эти команды очень часто полезны. Тем более что все они сдвигают, биты через флаг переноса, который затем может проверяться.

Сдвиг байта влево удваивает величину, записанную в этом байте (регистре), старшие биты при этом не теряются. Вовремя сдвига вправо величина делится пополам.

В таблице продемонстрированы команды этой группы:

	RCL	RL	SLA	ARC	RR	SRA	SRL
A	CB 07	CB 17	CB 27	CB 0F	CB 1F	CB 2F	CB 3F
H	CB 04	CB 14	CB 24	CB 0E	CB 1C	CB 2C	CB 3C
L	CB 05	CB 15	CB 25	CB 0D	CB 1D	CB 2D	CB 3D
B	CB 00	CB 10	CB 20	CB 08	CB 18	CB 28	CB 38
C	CB 01	CB 11	CB 21	CB 09	CB 19	CB 29	CB 39

D	CB 02	CB 12	CB 22	CB 0A	CB 1A	CB 2A	CB 3A
Б	CB 03	CB 13	CB 23	CB 0B	CB 1B	CB 2B	CB 3B
(HL)	CB 06	CB 16	CB 26	CB 0E	CB 1E	CB 2B	CB 3E
IX+ +D	DD CB D 06	DD CB D 16	DD CB D 26	DD CB D 0E	DD CB D 1E	DD CB D 2E	DD CB D 3E
IY+ +D	FD CB D 06	FD CB D 16	FDCB D 26	FD CB D 0E	FD CB D 1E	FD CB D 2E	FD CB D 3E

Имеются также четыре однобайтовые команды для сдвига регистра А и две специальные команды, работающие с полубайтами.

Мнемоника	*16-ный код*
RLCA	07
RLA	17
RRCA	0F
RRA	1F
RRD	ED 67
RLD	ED 6F

Суммируя влияние этих команд на основные флаги, можно сделать следующие выводы:

- ✓ все команды, за исключением RLD и RRD, влияют на флаг переноса;
- ✓ все команды, за исключением четырех команд с одиночными байтами, влияют на флаги знака и переполнение четности.

Программа, которая иллюстрирует работу этих команд, представлена на странице 68.

Группа 14. Команды обработки битов

Набор команд Z80 имеет команды, которые позволяют определять, устанавливать и сбрасывать отдельные биты в пределах байта, помещенного в регистр или находящегося в памяти.

Подгруппа А. Команды бит

Они позволяют определять состояние определенного бита. При использовании одной из этих команд обычным является процесс проверки, например, путём использования команды JP Z.

Команды бит устанавливают флаг 0, если используемый бит сброшен, и наоборот.

Подгруппа Б. Команды SET

Эти команды позволяют устанавливать определенный бит, 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 - ничего не изменит.

Подгруппа В. Команда RES

Данная команда сбрасывает определенный бит. Флаги не меняются. Команды этих трёх подгрупп приведены в таблице:

	Ком	Биты							
		0	1	2	3	4	5	6	7
A	BIT	47	4F	57	5F	67	6F	77	7F
Рег.	RES	87	8F	97	9F	A7	AF	B7	BF
Св**	SET	C7	CF	D7	DF	E7	EF	F7	FF
H	BIT	44	4C	54	5C	64	6C	74	7C
Реп	RES	84	8C	94	9C	A4	AC	B4	BC
CB**	SET	C4	CC	D4	DC	E4	EC	F4	FC
L	BIT	45	4D	55	5D	65	6D	75	7D
Рег.	RES	85	8D	95	9D	A5	AD	B5	BD
CB**	SET	C5	CD	D5	DD	E5	ED	F5	FD
B	BIT	40	48	50	58	60	68	70	78
Рег,	RES	80	88	90	98	A0	A8	B0	B8
CB**	SET	C0	C8	D0	D8	E0	E8	F0	F8
C	BIT	41	49	51	59	61	69	71	79
Рег.	RES	81	89	91	99	A1	A9	B1	B9
CB**	SET	C1	C9	D1	D9	E1	E9	F1	F9
D	BIT	42	4A	52	5A	62	6A	72	7A
Рег.	RES	82	8A	92	9A	A2	AA	B2	BA
CB**	SET	C2	CA	D2	DA	E2	EA	F2	FA
E	BIT	43	4B	53	5B	63	6B	73	7B
Рег.	RES	83	8B	93	9B	A3	AB	B3	BB
CB**	SET	C3	CB	D3	DB	E3	EB	F3	FB
(HL)	BIT	46	4E	56	5E	66	6E	76	7E
CB**	RES	86	8E	96	9E	A6	AE	B6	BE

VAPRINT

	SET	C6	CE	D6	DE	E6	EE	F6	FE

Программа, которая демонстрирует эти команды, показана на странице 70.

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 область памяти.

Чтобы перемещать блок данных, базовый адрес должен помещаться в пару регистров HL, адрес места назначения информации - в пару регистров DE, и число байтов в блоке - в пару регистров BC. Для того, чтобы искать область первого присутствия в памяти определенной величины, базовый адрес может опять быть в паре регистров BC, а регистр A должен содержать искомую величину.

Команды подгруппы:

Мнемоника	*16-ный код*
AUTOMATIC	
LDIR	ED B0
LDDR	ED B8
CPIR	ED B1
CPDR	ED B9
NON-AUTOMATIC	
LDI	ED A0
LDD	ED A8
CPI	ED A1
CPD	ED A9

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

Автоматическая команда будет выполнять свою задачу без другой команды, составленной программистом. Когда команда LDIR выполняется МП, тогда блок данных перемещается. Отсюда следует, что выполнение автоматической команды может длиться разное время. Это время зависит от количества байтов, которые буду перемещаться.

Однако неавтоматическая команда передает только по одному байту, и требует, чтобы программист использовал команду поочередно, если блок данных перемещается или находится в стадии поиска.

Время неавтоматической команды, поэтому определено. Автоматическая команда представляет собой команду обработки блока, при выполнении которой адреса источника, места назначения информации и счётчика приращения - уменьшаются или увеличиваются МП. 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 адресом, находящимся в паре регистров DE с числом байтов, определенных в паре регистров BC. При работе одиночный байт перемещается из (HL) в (DE). Величина в паре регистров BC затем уменьшается, а величина в парах HL и DE увеличивается. Если счётчик в регистрах BC ещё не достиг 0, тогда будет пересылаться другой байт данных. Этот процесс продолжается, пока счётчик не достигнет 0, но заметьте, что в этот момент пары регистров HL и DE достигают значения адресов после блока. Флаг переполнения/чётности сбрасывается этой командой.

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 базовый адрес блока соответствовал последнему адресу блока. Место назначения информации также должно относиться к последнему адресу области памяти.

CPIR

Эта команда просматривает определенную область памяти для поиска первого наличия эталонной величины. Регистр HL должен содержать базовый адрес, а BC -число байтов для исследования, регистр A - эталонную величину. При работе байт из HL сравнивается с байтом (A). Если они не совпали, то BC уменьшается, а HL увеличивается и сравниваются следующие. Так до тех пор, пока не совпадут байты, либо BC не станет равным 0. Если совпали, тогда флаг нуля устанавливается, а флаг знака сбрасывается, а регистр HL указывает адрес выбранного байта. Величина в BC указывает, как далеко от конца блока определено совпадение. Если не совпали, то регистр BC=0, а флаги 0, знака и переполнение/чётности сбрасываются.

CPDR

Работа этой команды похожа на CPIR, но блок данных просматривается сверху вниз. Далее рассматриваются неавтоматические команды.

LDI

Выполнение этой команды даст перемещение одиночного байта данных из адреса в регистровой паре HL по

© VA PRINT

адресу в регистровой паре DE. Величина в BC уменьшается. Флаг переполнение/чётности устанавливается до тех пор, пока регистровая пара BC не станет равной 0. Значения в парах HL и DE уменьшаются. Это сделано для программиста, который решает, перемещать или не перемещать байты данных. Программист, вероятно, проверит байт, который надо перемещать, и затем осуществит его перемещение.

LDD

Как LDI, за исключением того, величина в HL и DE уменьшаются.

CPI

Выполнение этой команды приведет к тому, что байт, определенный в паре HL, копируется в МП и сохраняется там до тех пор, пока величина в HL увеличивается, а в BC уменьшается. Сохраненный байт затем сравнивается с байтом в регистре A. Флаг 0 устанавливается, если они совпали, в противном случае - сбрасывается. Флаг знака сбрасывается, флаг переполнение/чётности сбрасывается до тех пор, пока BC не станет равным 0, при BC=0 он устанавливается.

CPD

Эта команда подобна CPI, за исключением того, что регистровая пара HL уменьшается.
Программа, демонстрирующая эту группу команд, приведена на странице 71.

Группа 16. Команды ввода/вывода

В Z80 имеется исчерпывающий набор команд, позволяющих программисту получать данные от внешнего источника (IN) или выдавать данные на периферию (OUT). Имеются простые, неавтоматические и автоматические команды в этой группе, хотя в SP используются только простые команды. Во всех случаях данные, которые передаются через IN и OUT, представляют форму 8-ми битовых данных. По команде IN Z80 принимает байты данных с шины данных и копирует их в определенный регистр. Во время выполнения команды IN контрольная линия IORQ активна, так же как и RD. По команде OUT МП передает данные с определенного регистра на шину данных, оттуда они поступают на периферийное устройство. Во время работы команды OUT линии IORQ и WR будут активны. В дополнение к состоянию RD, WR и IORQ периферийное устройство будет активизироваться с использованием адреса, помещенного на адресной шине во время выполнения либо IN, либо OUT команды (по требованию), этот адрес опознает адрес порта и в Z80 представляет собой 16-битный адрес. В МП Z80 используется только несколько из 65535 возможных адресов порта, позволяя периферийному устройству идентифицировать состояние определенной линии адреса. Принтер ZX выбирается линией A2, которая активна. Командами этой группы являются:

		Рег. I/O	Старший	Младший
IN A,(+DD)	DB 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 (+DD),A	D3 DD	A	A	DD
IN (C),A	ED 79	A	B	C
IN (C),H	ED 61	H	B	C
IN (C),L	ED 69	L	B	C
IN (C),B	ED 41	B	B	C
IN (C),C	ED 49	C	B	C
IN (C),D	ED 51	D	B	C
IN (C),E	ED 59	E	B	C

Неавтоматические и автоматические команды;

```
INI             ED A2    неавтоматическое увеличение
INIR            ED B2    автоматическое увеличение
IND             ED AA    неавтоматическое уменьшение
INDR            ED BA    автоматическое уменьшение
OUTI            ED A3    неавтоматическое увеличение
OTIR            ED B3    автоматическое увеличение
OUTD            ED AB    неавтоматическое уменьшение
OUTR            ED BB    автоматическое уменьшение
```

Группа 17. Команды прерывания

Имеется 7 команд, которые позволяют прерывать систему выполнения команд Z80. Командами этой группы являются:

```
EI              FB
```

VAPRINT

```
DI                F3
IM 0              ED 46
IM 1              ED 56
IM 2              ED 5E
RETI              ED 5D
RETN              ED 45
```

Каждая из команд будет рассмотрена поочередно.

EI

При включении Z80 система маскированного прерывания не может прервать выполнение команд. Эта ситуация существует до тех пор, пока прерывания не будут разрешены программистом по команде EI. В SP система маскирующих прерываний используется для отсчёта текущего времени и опроса клавиатуры, и эти прерывания передаются с синхронизирующей частотой 50 Гц.

DI

В любом месте программы программист может запретить прерывания по команде DI, которая не позволяет МП принимать сигналы с линии INT. В SP маскируемое прерывание невозможно во время выполнения команд SAVE, VERIFY, MERGE.

IM 0

Имеется три типа прерывания. Режим 0 выбирается автоматически МП при включении или по команде IM 0. Этот режим позволяет периферийным устройствам обмениваться информацией с МП, программа перезапуска которого должна следовать за приёмом маскированного прерывания на линии INT. Этот режим в SP не используется.

IM 1

Режим прерывания 1 выполняется по команде IM 1. При выполнении 16K монитора необходимо включить IM 1 в начало программы. В этом режиме перезапуск с адреса 0038 будет всегда выбираться при приёме сигнала на линии INT, длительность которого соответствует возможности обработки системы маскированного прерывания. В SP программа в кодах с 0038 корректирует данные, полученные в реальном времени, и опрашивает клавиатуру.

IM 2

Режим прерывания 2 не используется в SP, но является самым мощным из трёх режимов. В нем периферийное устройство может указать МП, какая из 128 подпрограмм должна обрабатывать маскированное прерывание. Содержимое регистра I и байта, который поступает на внешнее устройство, используются вместе, чтобы образовать 16-ричный адрес, который затем используется для выбора таблицы векторов, размещенной заранее в памяти.

RETI

Это специальная команда возврата для использования с программой маскированного прерывания. По этой команде происходит возврат с сохранением маскированного прерывания, принятого раньше.

RETN

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

Группа 18. Дополнительные команды

Имеется 6 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, т.е. она устанавливает сброшенный бит и сбрасывает установленный. Это называется дополнением до 1. Основные флаги не изменяются.

NEG

Это команда дополнения до 2-х содержимого регистр A, т.е. инвертирование с увеличением. Эта команда влияет на основные флаги. Флаги знака и 0 зависят от результата. Флаг переноса будет установлен, если регистр A первоначально был 0, в противном случае - сброшен.

SCF

Установка флага переноса.

VA PRINT

CCF

Сброс флага переноса.

HALT

Это специальная команда, заставляющая МП останавливать выполнение команд, пока не возникнет прерывание. В SP единственными прерываниями могут быть маскируемые прерывания. Следовательно, при поступлении прерываний команда HALT заканчивается следующим сигналом маскируемого прерывания. Команда "PAUSE" использует это, чтобы подсчитать 1/50 сек.

DAA

Это 10-ная переупаковка регистра A в 2-10-ной арифметике (BCD), 10-ные цифры от 0 до 9 представлены 2-ными тетрадами 0000-1001, а тетрады 1010- 1111 не используются. Поэтому:

Байт 0000 0000 представляет число 0

Байт 0011 1001 представляет число 9

Эта команда превращает байты в 2-10-ную систему (BCD). Флаги 0 и знака зависят от результата, и флаг переполнения/чётности устанавливается. Флаг переноса зависит от того, было ли переполнение или заём при операциях BCD.

VA PRINT

6. ДЕМОНСТРАЦИОННЫЕ ПРИМЕРЫ ПРОГРАММ В КОДАХ

Основная цель главы - дать читателю сведения о начальных проблемах, связанных с написанием программ в машинных кодах. Команды Z80 перечислены в главе 5, но не указывается, как ими пользоваться на SP. Настоящая глава содержит серию демонстрационных программ. В главе 8 будут даны советы, как программа машинного кода может быть написана, чтобы иметь полное представление о преимуществе цветного SP и дисплея с высоким разрешением. Имеется три точки зрения, подлежащие обсуждению.

А. Выбор области RAM.

Программист должен определить требуемый объём памяти. В SP 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.

Б. Ввод кодов программы.

В SP 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
```

Каждая может оказаться полезной в определенном случае. Рекомендуемый метод заключается в описании команд машинного кода в их соответствующих 16-ных символах. Следующий 16-ный загрузчик будет использоваться во всей главе!!!

```
10 D=32000: REM HEX LOADER
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","**"
RUN
```

Это приведет к тому, что адрес 32000 содержит 0, адрес 32001 содержит 1, адрес 32002 содержит 2.

В. Выполнение программы.

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

Примеры USR:

PRINT USR N	- печатает 10-ное значение содержимое регистровой пары BC.
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.

6.1. ПРОГРАММЫ

Группа 1

Команда "нет операции" (NOP) (см. стр. 33). Эта команда очень простая, и программа машинного кода, содержащая одну или более строк "нет операции", заканчивается "RET".

Следующая программа ассемблера показывает простое использование команды NOP.

Адрес Код	*Обозначение*	*Комментарии*
7D00 00	NOP	NOP
7D01 C9	RET	возврат

Следующая программа Бейсик показывает, как изменить вышеприведенную программу на ассемблере (ASM) для SP.

Программа 1. "NOPE"

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

Группа 2

Программы для непосредственной загрузки регистров (см. стр. 33). Команда "USR число" возвращает

значение регистра BC как абсолютное 16-битовое число, и команда PRINT отображает в 10-ном виде от 0 до 65535. В следующем примере B и C регистры загружаются постоянными, используя команды из этой группы, и программа 2 Бейсик использует программу кода.

```
7D00 06 00          LD B,00         регистр B=0
7D02 0E 00          LD C,+XX        пользователь вводит различные значения
7D04 C9             RET
```

В программе 2 команды 'LD B,+ADD' И LD C+ADD' обе используются. Пользователь может задавать различные значения в качестве входных операндов.

Программа 2.

```
50 DATA "06","00","0E","00"
51 DATA "C9","**"
60 INPUT "введите значения для регистра C (только 0-255) ";N
70 CLS
80 POKE 32003,N
90 PRINT AT 10,0; "регистр C теперь содержит "; CHR$ 32; USR 32000
100 GO TO 60
```

Следующая программа показывает команду LD BC,+DDDD

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

В программе 3 можно задавать различные величины "LD BC,+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 "введите значение для регистра пары BC (только 0-255)"; CHR$ 32;N
70 CLS
80 POKE 32001,N-256*INT(N/256): POKE 32002,INT(N/256)
90 PRINT AT 10,0, "Регистр BC теперь содержат"; CHR$ 32; USR 32000
100 GO TO 60
```

Группа 3

Команды копирования и обмена регистров (см. стр. 33).

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

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

В программе 4 переменная заводится в регистр B и пересылается в регистр C.

Программа 4. "LD C,L"

```
50 DATA"06","00","2E","00"
51 DATA "4D","C9","**"
60 INPUT "введите значения для регистра L (только 0-255)"; CHR$ 32; N
70 CLS
80 POKE 32003,N
90 PRINT AT 10,0; "C регистр теперь содержит"; CHR$ 32; USR 32000
100 GO TO 50
```

Нижеследующие программы воодушевят читателя.

```
50 DATA "06","00","OE","00"
51 DATA "65","7C","57","5F"
52 DATA "4B","C9","**"
```

Команда 'EX DE,HL' может также быть включена в программу 4.

Пример.

```
7D00 26 00          LD H,+00        регистр H=0
7D02 2E00           LD L,+00        вводятся значения
7D04 EB             EX DE,HL        пересылается в DE
7D05 42             LD B,D          регистр B = регистр D
7D06 4B             LD C,E          регистр C = регистр E
7D07 C9             RET
```

Это задается следующим оператором DATA.

```
50 DATA "26","00","2E","00"
51 DATA "EB","42","4B"
52 DATA "C9","**"
```

и может использоваться с программой 4.

Последние две программы этой группы посвящены командам 'EX AF.'F" и 'EXX'. Пользователю можно

© YA PRINT

использовать альтернативные регистры, кроме H' и L', содержащие адрес возврата в Бейсик. Читатель может включить эти команды в программу 4.

Пример.

```
50 DATA "06","00","2E","00"
51 DATA "08","D9","D9","06"
52 DATA "4D","C9","**"
```

Строка 51 переключает все основные регистры, а потом переключает их обратно, не изменяя.

Группа 4

Команды загрузки регистров из адресов памяти (см. стр. 35). Команды в этой группе позволяют пользователю загрузить регистр содержимым памяти. Эта адресация может быть абсолютной, косвенной и индексной.

Первая программа показывает абсолютную индексацию. Задается адрес 31999, имеющий имя STORE, и команда LD A,(ADDR) используется для выборки содержимого STORE по следующей программе. STORE в строке 1 присваивается значение 31999 (16-ное 7CFF).

```
7D00 06 00          LD B,+00        регистр B
7D02 3A FF 7C       LD A,(STORE)    выборка текущего значения
7D05 4F             LD C,A
7D06 09             RET
```

В программе 5 пользователь может вводить различные значения, сохраняемые в STORE, и вышеприведенная программа на ASM возвращает их пользователю.

Программа 5. "LD A,(ADDR)"

```
50 DATA "06","00"
51 DATA "3A","FF","7C"
52 DATA "4F","C9","**"
60 INPUT "INTER A VALUE FOR LOCATION STORE(0-255 ONLY)"; CHR$ 32; N
70 CLS
80 POKE 31999,N
90 PRINT AT 10,0; "THE LOCATION STORE NOW HOLDS"; CHR$ 32; USR 32000
100 GO TO 60
```

Вторая программа показывает косвенную адресацию. В ней адрес STORE загружается в регистр HL до использования команды "LD C,(HL)" STORE=7CFF.

```
7D00 06 00          LD B,+00        регистр B=0
7D02 21 FF 7C       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","FF","7C"
52 DATA "4E","C9","**"
```

строки 60-100 как в программе 5.

Третья программа показывает индексную адресаций. В этой программе адрес 31936 (7CC0) называется BASE, а адрес 31999 (7CFF) - STORE,

```
BASE=7CC0
STORE=BASE+3F
7D00 06             LD B,+00        регистр B=0
7D02 DD 21 C0 6C    LD IX,+BASE     установка IX
7D06 DD 4E 3E       LD C,(BASE+3F)  выборка текущего значения
7D09 C9             RET
```

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

Программа 7.

```
50 DATA "06","00"
51 DATA "DD","21","C0","7C"
52 DATA "DD","4E","3F"
53 DATA "C9","**"
```

строки 60-100 как в программе 5.

В программе используется регистр IX, но вполне возможно использование регистр IY, но при этом байты машинного кода надо заменить 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 регистрах (см. стр. 35).

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

© YA PRINT

Первая программа показывает абсолютную адресацию. В программе абсолютный адрес для команды 31999, 16-ной 7CFF, названной 'STORE', используется команда 'LD (ADDR),A'.

```
STORE
7D00 3E 00          LD A,+XX        ввод различных значений
7D02 32 FF 7C       LD (STORE),A    ввод текущего значения в STORE
7D05 C9             RET
```

В программе 8 пользователь опять вводит различные значения в STORE. Программа в машинных кодах выполняется использованием через RANDOMIZE USR 32000, и величина STORE задается использованием PEEK 31999.

Программа 8. "LD (ADDR),A".

```
50 DATA "3E","00"
51 DATA "32","FF","7C"
52 DATA "C9","**"
60 INPUT "INTER 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 31999
100 GO TO 50
```

Вторая программа показывает косвенную адресацию. В этой программе регистровая пара HL указывает на STORE, а команда 'LD (HD),E' используется для передачи.

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

В программе 9 пользователь опять вводит величину, которая должна передаваться в STORE и обратно считываться с помощью PEEK 31999.

Программа 9. "LD (HL),E"

```
50 DATA "1E","99"
51 DATA "21","FF","7C"
52 DATA "73","C9","**"
```

Строки 60-100 как в программе 8.

В 3-ей программе используется индексная адресация. В этом случае адрес 32061 (7D3D) называется BASE, и STORE поэтому рассматривается как ('BASE-3E').

```
BASE=7D3D
STORE=BASE-3E
7D00 3E 00          LD A,+XX        вводится значение
7D02 DD 21 3D 7D    LD IX,+BASE     IX указывает на BASE
7D06 DD 77 C2       LD (IX-3E),A    IX-3E рассматривается как IX+CR
7D09 C9             RET
```

В программе 10 пользователь опять вводит величины, которые должны быть переданы в STORE, и считывает обратно с помощью PEEK 31999.

Программа 10. "LD (IX+D),A"

```
50 DATA "3E","00"
51 DATA "DD","21","3D","7D","DD"
52 DATA "77","C2"
53 DATA "C9","**"
```

Программы 5-10 все использовали 1-байтовые числа и соответствующие команды регистров.

VA PRINT

Группа 6.

Команды сложения (см. стр. 36). Команды этой группы позволяют складывать величины вместе (ADD) и увеличивать значение (INC) и складывать с битом переноса.

Первая программа показывает использование команды 'ADD A,B'.

```
7D00 00             NOP             будет использоваться позже
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 программе 11 пользователь вводит 2 числа. Эти числа передаются в регистры A и B и складываются вместе в двоичной арифметике. Результат возвращается функцией USR.

Программа 11. "ADDA.B".

```
50 DATA "00","3E","00"
51 DATA "06","00","80"
```

```
52 DATA "06","00","4F"
53 DATA "C9","**"
60 INPUT "INTER A FIRST VALUE (0-255) ";CHR$ 32,F
70 INPUT "INTER A SECOND VALUE (0-255)";CHR$ 32;S
80 CLS
90 POKE 32000,F: POKE 32004,S
100 PRINT AT 10,5;F;CHR$ 32;"ADD";CHR$ 32;S;CHR$ 32;"=";CHR$ 32;U SR 32000
110 GO TO 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 09             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 адреса 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;"IN CREMENTS TO GIVE"; CHR$ 32; USR 32000
110 GO TO 60
```

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

```
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 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 даёт 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"; CHR$ 32;F; CHR$ 32;"ADC"; CHR$ 32;S; CHR$ 32;"=";
  CHR$ 32; USR 32000
```

Группа 7. Команды вычитания.

Команды в этой группе позволяют вычесть одну величину из другой (SUB), декрементировать значение (DEC) и вычесть бит 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=7CFF
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 FF 7C       LD (STORE),A    передать значение флага переноса в STORE
7D10 C9             RET
```

В программе 14 вышеприведенная программа вызывается после того, как пользователь вводит значения для регистров A и B. Значение флага переноса получено использованием PEEK 31999.

© VA PRINT

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","FF","7C"
55 DATA "C9","**"
60 INPUT "INTER A FIRST VALUE (0-255)";CHR$ 32;F
70 INPUT "INTER A SECOND VALUE (0-255)";CHR$ 32;S
80 CLS
90 POKE 32002,F: POKE 32QO4,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 31999;"RESET" AND NOT PEEK 31999
110 GO TO 60
```

Используйте приведенную программу с числами, которые меньше, больше и равны одно другому.

Вторая группа команд содержит команду DEC. Команда DEC BC может быть показана, сделав соответствующие исправления в программе 12. Ими являются.

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

Третья подгруппа команд содержит команду SBC. Команда SBC может быть показана, сделав соответствующие исправление в программе 14. Ими являются:

`50 DATA "37","3E","00"` для установки С-флага или

`51 DATA "A7","3E","00"` для сброса С-флага

`52 DATA "06","00","98"` 'SBC A,B'

`100` изменение SUB на SBC

Опять используйте эффект чисел, которые меньше, больше и равны один другому. Нет разницы в результатах между SUB и SBC при сброшенном флаге переноса.

Группа 8

Команды сравнения (см. стр. 40). Команды сравнения по Существу такие же, как команды сложения, исключая тот факт, что регистр А не изменяется. Флаг переноса изменяется и в последствии может проверяться.

Команда CP B может быть показана ещё раз, сделав соответствующие изменения в программе 12.

```
51 DATA "06","00","B8"
100 PRINT AT 10,0;F;CHR$ 32;"CP";CHR$ 32;S,CHR$ 32;"GIVE CARRY";CHR$ 32;
```

и добавить строки

```
101 RANDOMIZE USR 32000
102 PRINT "RE" AND NOT PEEK 31999;"SET"
```

Группа 9

Логические команды (см. стр. 40).

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.

```
7D0O 3E 00        LD A,+XX
7D02 06 00        LD B,+XX       вводятся значения
7D04 A0           AND B          логическое И
7D05 32 FF 7C     LD (STORE),A   результат в STORE
7D08 C9           RET
```

В программе 15 используется вышеприведенная программа. Пользователю предлагается ввести 2 значения поочередно, при вводе каждого значения оно показывает в двоичной форме. Результат считывается из STORE с использованием PEEK 31999 и также даётся в двоичной форме.

Программа 15. "AND B".

```
50 DATA "3E","00"
51 DATA "06","00"
52 DATA "A0"
53 DATA "32","FF","7C"
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 PRINT "ENTER A SECOND VALUE (0-255) "; CHR$ 32; S
```

VA PRINT

```
120 POKE 32003,S
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 31999: GO SUB 300
170 GO TO 60
300 REM BINARY OFF
310 FOR N=7 TO 1 STEP -1
320 LET P=2^N
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 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 (см. стр. 42). Семнадцать команд этой группы позволяют сделать переходы из одной части программы машинного кода к другой. Переходы могут быть относительными от -128 до +127 от значения счётчика команд, или абсолютными, т. е. задаётся адрес следующей команды. Переход может быть сделан условно в зависимости от состояния основных флагов, хотя только абсолютные команды перехода имеют полный диапазон имеющихся команд.

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.-ЮООО     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 00 00 00                    не используется
7D0F 03             INC BC          увеличение C 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. "MR 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;"КОМАНДА"
70 PRINT "JR E"
80 PRINT AT 8,0;
90 INPUT "INTER A FIRST VALUE (0-255) ",CHP$ 32; F
100 PRINT F; CHR$ 32;"CP"; CHR$ 32;
110 INPUT "INTER 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;"ANY KEY TO CONTINUE"
170 PAVSE 50
180 IF INKEY$="" THEN GO TO 180
190 CLS: GO TO60
```

© VA PRINT

В программе пользователю предложено ввести две величины, выполняется операция сравнения, а затем, если необходимо, переход.

Переменная R будет = 0, если перехода не было и 1, если необходим переход.

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

Для команд перехода изменениями являются:

```
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 "JRNC.E
53 DATA "38","00","00" JR C,E
70 PRINT "JR C,E"
```

Команды абсолютного перехода показаны изменением программы машинного кода, так что абсолютный переход осуществляется к адресу NEXT. Строка команды по адресу NEXT (7D0F) будет теперь читаться -JP ADDR.

```
7D07 C3 0F7D 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 NZ,ADDR
70 PRINT "JP NZ,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

Команды DJNZ E. Эта команда является очень полезной командой и может легко использоваться для получения простых циклов в программе машинного кода. Чтобы использовать команду, программист сначала должен определить число требуемых циклов, и это число должно быть занесено в регистр B. Цикл может быть выполнен по команде 'DJNZ E' аналогично NEXT.

Следующие 4 строки программы Бейсик будут без конца печатать алфавит заглавными буквами. Когда экран будет заполнен, появится вопрос "SCROLL ?".

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

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 счётчика циклов
7D02 3E 5B          LD A,+5B
7D04 90             SUB B           регистр A содержит значение
7D05 D7             RST 0010        печать символа
7D06 10 FA          DJ NZ,LOOP      следующая буква
7D08 C9             RET
```

Программа 17 использует вышеприведенную программу для печати алфавита на экране TV-приемника. Каждое обращение USR 32000 будет повторять операцию печати еще и еще раз.

Программа 17.

```
50 DATA "06","1A"
51 DATA "3E","5B"
```

© VA PRINT

```
52 DATA "90","D7"
53 DATA "10","FA"
55 DATA "C9","**"
60 PRINT
70 RANDOMIZE USR 32000
80 GO TO 70
```

Примечание: включение строки 60 важно, т к. она открывает канал для основной области экрана.
Если эта команда или подобная ей опускается, тогда будет сделана печать в области редактирования

Группа 12

Команды стека (см. стр. 46). Имеется пять подгрупп команд в этой группе Первая содержит команды PUSH и POP, а вторая - команды обмена со стеком. Следующая программа использует программу 18, чтобы показать команды из первой подпрограммы.

```
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, в итоге 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 STASK="; CHR$ 32; USR 32000
100 GO TO 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 00 00       ……
7D0F 03             INC BC          увеличение BC
7D10 C9             RET             возврат из подпрограммы
```

Эта процедура может использоваться в программе 18 с последующими изменениями для различных команд.

```
130 PRINT AT 12,0;"CALL"
CALL NZ,ADDR      53 DATA "C4","0F","7D"
                  70 PRINT "CALL NZ,ADDR"
CALL ADDR         53 DATA "CD","0F","7D"
                  70 PRINT "CALL ADDR"
CALL Z,ADDR       53 DATA "CC","0F","7D"
                  70 PRINT "CALL Z,ADDR"
CALL NZ,ADDR      53 DATA "D4","0F","7D"
                  70 PRINT "CALL NZ,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      53DATA "EC","0F","7D"
                  70 PRINT "CALL PE,ADDR"
CALL P,ADDR       53 DATA "E4","0F","7D"
                  70 PRINT "CALL P,ADDR"
CALL M,ADDR       53 DATA "FC","0F","7D"
                  70 PRINT "CALL M,ADDR"
```

Следующая программа 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
```

VAPRINT

```
7D02 FE 00          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 (см. стр. 49).

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

```
STORE=7CFF
7D00 AF             XOR A           очистка регистра A
7D01 FE 00          CP +XX          сравнен, со значением 0
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 C-флага
7D0D 32 FF 7C       LD (STORE),A    в STORE
7D10 C9             RET
```

В программе 19 пользователю предлагается ввести 0 или 1, чтобы сбросить или установить флаг переноса. Следующая подсказка просит ввести значения. Она 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","FF","7C"
55 DATA "C9","**"
60 PRINT AT 2,0;"INSTRUCTION......"
70 PRINT "RLC C"
80 INPUT "CARRY RESET OR SET? (0/1)"; CHR$ 32; C
90 POKE 32002,C
100 PRINT AT 6,0;"CARRY-"; C
110 INPUT "ENTER A VALUE (0-255) "; CHR$ 32; F
120 POKE 32006,F
130 PRINT AT 10,0;"INITIAL VOLUE";: GO SUB 300
140 LET F=USR 32000
150 PRINT AT 14,0;"FINAL VALUE": GO SUB 300
160 PRINT AT 18,0;"CARRY-"; PEEK 31999
170 PRINT AT 21,0;"ANY KEY TO CONTINUE"
180 PAUSE 0
190 IF INKEY$="" THEN GO TO 190
200 CLS: GO TO 60
300 REM BINARY OFF
310 FOR N=7 TO 1 STEP -1
320 LET P=2^N
330 PRINT CHR$(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 команд, которые используют регистр C.

```
RLC C               52 DATA "CB","01"
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"
```

© VA PRINT

```
SRL C              52 DATA "CB","39"
                   70 PRINT "SRL C"
```

Если желательно, процедура в программе 19 может быть изменена для демонстрации 4-х команд с одиночными битами.

Группа 14

Команды обработки битов (см. стр. 50).

Команды этой группы могут делиться на три подгруппы. Команды RES 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 в большей степени являются самыми полезными в этой группе, и следующая программа показывает, как BIT 7,H используется.

Программа 20 является программой 2-ной печати. Пользователю предложено ввести целое число в диапазоне 0-65535. Программа машинного кода затем вызывается, чтобы отпечатать 2-ное представление.

Значение помещается пользователем в регистровую пару HL, и 7-ой бит регистра H считывается и печатается. Регистры H и L затем сдвигаются влево, перенос из регистра L поступает в регистр H, опять 7-ой бит регистра H считывается и печатается. Операция чтения и сдвига выполняется 17 раз, чтобы получить полное двойное число.

Программа представлена следующим образом:

```
LOOP=7D05
PRINT=7D0C
7D00 21 00 00       LD HL,+XXXX     вводятся значения
7D03 06 10          LD B,+10        счётчик цикла на 10 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 1
7D0B 3C             INC A           символ 1
7D0C D7             RST 0010        печать 0 или 1
7D0D 3E 20          LD A,"SP"       печать пробела
7D0F D7             RST 0010
7D10 CB 15          RL L            сдвиг L
7D12 CB 14          RL H            сдвиг H
7D14 10 EF          DJ NZ,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 14,0;
130 RANDOMIZE USR 32000
140 GO TO 60
```

VA PRINT

Группа 15

Команды обработки блока (см. стр. 51).

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 в области адресов 4000-47FF копируются командами в адреса 4800-4FFF Пользователь может увидеть, что это произошло, так как все символы, которые были в строках 0-7, были 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     начало строки
7D06 01 00 08       LD BC,+0800     2048 адресов
```

```
7D09 ED B0          LDIR            пересылка блока
7D0B C9             RET
```

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 CHARACTEERS-8 LINES"; C$
70 PRINT C$
80 RANDOMIZE USR 32000
```

Заметьте, что LDIR или LDDR могут использоваться для дублирования трети экрана, но физическое расположение позволяет использовать эти команды для других типов копирования, которые включают всю символьную область.

Нижеследующая программа демонстрирует команду CPIR. Программа 22, которая использует стандартную программу, даёт адрес первого байта в 16K SP, который хранит специфицированный байт. Когда программа выполняется, можно видеть, что все значения 0-255 м.б. найдены в ROM, но значение 154 впервые встречаются в адресе 11728 и поэтому является редко используемой величиной в этой части ROM.

Программа имеет вид:

```
7DO0 3E 00          LD A,+XX        ввод образца
7D02 01 FF 3F       LD BC,+3FFF     старший адрес ROM
7D05 21 00 00       LD HL,+0000     первый адрес ROM
7D08 ED B1          CPIR            поиск в ROM
7D0A 44             LD B,H          пересылка адреса байта
7D0B 4D             LD C,L
7D0C 0B             DEC BC          указатель на искомый адрес
7D0D C9             RET
```

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;"OCCURS FIRST AT LOC."; CHR$ 32; USR 32000
90 NEXT F
```

Читатель может переписать эти программы, используя неавтоматические команды. Здесь не приводятся программы для команд ввода-вывода, команд прерываний и небольшой группы смешанных команд, но читатель может написать их сам.

VA PRINT

7. МОНИТОР 16К

Система SP снабжена 16К постоянной памяти, включающей в себя:

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

ROM содержит программу монитора, которая содержит адреса 0000-3FFF (0-16383) и в стандартной системе SP не может быть перемещена из этих адресов. Команда, расположенная по 0-му адресу, является 1 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 в центре системы работает по программе в машинных кодах, содержащейся в 16K ROM с того момента когда, питание подаётся к системе. Единственным исключением из этого бывает выполнение программы в кодах, написанных пользователем.

Пользователю SP представляется как машина, которая ожидает программную строку Бейсика либо в непосредственном режиме, либо с номером для построения Бейсик программы. Строки Бейсика без номера выполняются немедленно и могут использовать интерпретацию строк из заранее введенной программы. Система SP немного сложна, поскольку интерпретатор вызывает проверку синтаксиса строки, прежде чем записывает её 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 (RANDOMIZE USR 0).

Инициализация (INITIALIZATION)

Эта программа занимает адреса 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 SP эта программа, как следует из названия, выполняет основные функции в мониторе.

Эта программа вызывает (если необходимо) программы LIST EDITOR и SYNTAX CHEKER, если строка Бейсика 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 Бейсика. Затем по возвращению из программы LINE-RUN выдается сообщение.

Таблица сообщений об ошибках находится по адресам 1391-1536. Возврат к началу MAIN EXECUTION приводит к новому вызову программы EDITOR.

Редактор (EDITOR)

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.

Управляющие курсоры и ENTER вызывают возврат в программу MAIN EXECUTION.

В стандартной системе SP войти в программу EDITOR можно единственно через клавиатуру, и поэтому Программа KEYBOARD-INPUT вызывает программу EDITOR. Этот вызов направляется через информационную область канала.

KEYBOARD-INPUT

Программа занимает адреса 10A8-111C. Эта программа даёт код последней набранной на клавиатуре клавиши, читая его из системной переменной LASTK.

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

VA PRINT

KEYBOARD

Каждые 1/50 секунды 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 (занимает адреса 028E-02BE). Подпрограмма возвращает соответствующее ключевое значение в регистровую пару DE, которую другие программы клавиатуры используют для получения кода литеры.

PRINT-OUTPUT

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

Программа PRINT-OUTPUT располагается в адресах 09F4-0D4C и, возможно, является самой важной среди других программ.

Эта программа в сущности вызывается инструкцией RST 0010. Адрес, помещенный в 09F4, берётся из информационной области канала. RST 0010 определяет, быть ли коду, находящемуся в регистре A, помещенным на TV-экран, или быть напечатанным на принтере. Флаг, определяющий это, в программе повторно тестируется.

В системе SP программа PRINT-OUTPUT очень мощная, поскольку и символы, и управляющие коды выводятся по этой же программе.

В случае вывода на экран телевизора, текущая позиция печати берётся из соответствующих системных переменных, модифицируется и затем снова сохраняется. Позиция печати определяет номер строки и колонки, а также соответствующий адрес в дисплейном файле (верхняя левая точка в поле литеры). Программа PRINT-OUTPUT имеет подпрограммы, которые по необходимости вызываются для передачи. Другие подпрограммы меняют байт атрибутов для той области символов, чье значение содержится в соответствующих системных переменных. Заметьте, что стандартные литеры копируются из набора литер, а символы, определенные пользователем, - из области UDG, в то время как графические вычисляются по мере необходимости.

Подобная операция приводит к передаче формы символа в буфер принтера для последующего вывода на принтер; управляющие символы игнорируются.

В системе SP 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 Бейсика

Интерпретатор вызывается как для проверки синтаксиса, так и для выполнения строки. По большей части эти программы используют для обеих функций флаг синтаксис/выполнение, чтобы определить какая из операций должна быть выполнена.

Например, в строке LET A=1 сначала проверяется синтаксис, а затем переменной A 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 SP существует 50 команд и таблица команд. Она содержит класс команд, символы разделители и адреса программ выполнения команды;
2. Управляющая программа располагается в адресах 1B17-1CEE и содержит команды, позволяющие рассматривать операторы Бейсика один за другим, как требуется в программе. Точкой входа в синтаксический анализатор является адрес 1B17, а в исполнитель строки – 1B0A;
3. Программы класса команд располагаются в адресах 1C01-1CDD и главным образом занимаются анализом параметров команды Бейсика. Например, команда NEXT относится к классу 4, как требующая за собой простую переменную. Команда POKE относится к классу 8, как требующая за собой 2 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 Бейсика, и выполнение этих программ и есть собственно работа интерпретатора. Интерпретация операторов Бейсика может быть проиллюстрирована следующими примерами.

Рассмотрим оператор 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 байтов. Программа управления затем переходит к интерпретации следующего оператора. Следующий оператор 20 GO TO 50. В этом случае переменная GO TO программы располагается по адресу 1E67. В этой программе операнд оператора специфицируется и помещается в системную переменную NEWPPC так, что управляющая программа рассматривает оператор строки 50 как следующий оператор для интерпретации.

В строке 20 GO TO 50 символы между командой Бейсика GO TO и возвратом каретки (:) рассматриваются как выражение. В рассматриваемом случае преобразование выражения просто, так как значение, полученное преобразованием литер 5 и 0 даёт десятичный результат 50. Однако в SP интерпретатор Бейсика имеет и преобразователь выражений, который будет рассмотрен ниже.

Преобразователь выражений

Эта программа занимает адреса 24F8-28B1. В SP результат преобразования выражения может быть числовой

VA PRINT

или строковый. Числовой результат возвращается как плавающее значение в 5-ти верхних байтах калькуляторного стека, в случае строкового результата 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. В случае массива правильный элемент или элементы должны быть определены прежде, чем правильные значения могут быть возвращены, и в случае, если используется часть строковой переменной, её параметры должны быть соответствующим образом подготовлены.

Смешанные арифметические программы

Часть программы монитора, которая занимает адреса 2C88-2F9A, содержит серию арифметических программ. Самые важные 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 берёт 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. Например, литерал 04 эквивалентен CALL 30CA и приводит к тому, что 2 верхних значения калькуляторного стека замещаются одиночным значением, являющимся их результатом. Следовательно, 04 является литералом, обеспечивающим умножение. Эти литералы включаются в программу как байты данных (DEFB - определение байтов), следующие за командой RST 0028. Последний DEFB всегда имеет значение 38, которое выполняет завершающие операции и т.о. является возвратом из калькулятора.

Подпрограмма TAN

```
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, 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 этой 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 адресов памяти, в то время как использование 5-ти команд CALL и команд JP потребовало бы 32 адресов.

Можно использовать калькулятор в программах, написанных пользователем, но надо беспокоиться о том, чтобы правильно работать с калькуляторным стеком.

Набор литер

Существует "неиспользуемая" область 386E-3CFF, которая предшествует набору литер, занимающих адреса 3D0C-3FFF.

Набор литер содержит 96 форм символов, каждый из которых использует 64 бита.

```
3D58                DEFB +00        '00000000'
3D59                DEFB +00        '00000000'
3D5A                DEFB +08        '00001000'
3D5B                DEFB +08        '00001000'
3D5C                DEFB +3E        '00111110'
3D5D                DEFB +08        '00001000'
3D5E                DEFB +08        '00001000'
3D5F                DEFB +00        '00000000'
```

т.е. '+'.

Программа Бейсика, которая показывает 2-ные формы, дана следом:

```
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
```

VA PRINT

```
50 PRINT TAB 2;"'"; CHR$(32+(A-15615)/8);"'"
60 INPUT A$
70 CLS
80 NEXT A
90 STOP
300 REM BINARY OFF F
310 FOR N=7 TO 1 STEP -1
320 LET P=2^N
330 PRINT CHR$(48+INT(F/P))
340 LET F=F-INT(F/P)
350 NEXT N
360 PRINT INT F
370 RETURN
```

Строка 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 A, регистровые пары, DE загружаются значением +FFFF (верхнее значение адреса RAM) и происходит переход по адресу 11CB.

0008-000F RST 0008. Программа ошибки. Машинный стек очищается и выдается соответствующее сообщение.

0010-0012 RST 0010. Точка входа программы PRINT-A. Осуществляется переход по адресу 15F3.

0018-0024 RST 0018 и RST 0020. Получает текущий символ, на который указывает CH-ADD как следующий.

0028-0029 RST 0028. Переход по адресу 335B, который является стартовым адресом калькулятора.

0030-0037 RST 0030 BC-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:

```
10 RET POKE TABLE PRINTER
20 FOR A=149 TO 516
30 LET B=PEEK A
40 IF B<128 THEN PRINT CHR$ B: GO TO 60
50 PRINT CHR$(B-128)
60 NEXT B
```

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.

028F-03B4. Подпрограмма KEYBOARD, Набор подпрограмм, которые осуществляют возможность повторения и декодируют значения клавиш для получения требуемого кода литеры. Если клавиша нажата, то её код копируется в системную переменную LIST-K и бит 5 FLAG 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 её. Значение длительности абсолютно и должно быть увеличено так же, как и значение высоты тона уменьшено, для того, чтобы иметь повсюду одинаковую длительность.

03F8-046D. Программа команды BEEP. Эта программа широко использует калькулятор для изменения 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. По ошибке эти адреса содержат подпрограмму для ZX-81.

04C2-09F3. Программа команд SAVE, LOAD, VERIFY и MERGE. Важные части этой секции программы монитора следующие:

04C2-053E. Программа SAVE-BYTES, передает количество байт, указанных в регистровой паре DE, начиная с адреса, хранящегося в регистре IX на магнитофон вместе с начальным адресом и 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 указывает 1-ый адрес загрузки. 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.

VA PRINT

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 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 ещё через секунду - блок данных.

09A1-09F3. Сообщения, связанные с работой магнитофона.

09F4-0D4C. Программа PRINT-OUTPUT для печати на принтер. Бит 1 FLAGS должен быть установлен и сброшен для вывода на TV-экран. Важными частями этой секции монитора являются:

09F4-0A0E. Символы, которые могут быть напечатаны, отделяются от управляющих символов. В любом случае ищется текущая позиция для печати - смотри 0BC3.

0A11-0A22. Таблица управляющих символов.

0A23-0AD8. Различные программы, имеющие дело с управляющими кодами.

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. Такая же важная подпрограмма FETCH.

0B24-0BDA. Подпрограмма печати литеры PRINT-ANY. Эта особенная подпрограмма представляет собой весьма важную часть программы печати символов в системе SP. На выводе регистровой пары HL содержится начальный адрес места, где должен быть напечатан символ (DF-CC или эквивалент), регистровая пара BC содержит текущее значение строки и колонки (5-FCSN или эквивалент), и регистр A содержит код символа. Начальный адрес формы литеры находится или в наборе литер, или в области их, или в области калькуляторной памяти, где находятся специально созданные графические литеры. P-FLAG также анализируется, чтобы увидеть в режиме OVER или INVERSE (или в обоих сразу) должен быть напечатан символ.

0BB7-0BC4. В этой подпрограмме форма символа копируется из памяти или в дисплейную область, или в буфер принтера.

0BDB-0C09. Программа установки атрибутов. После того, как символ напечатан, байт атрибутов для символов должен быть установлен. Это включает в себя поиск и формирование значения атрибута и системных переменных ATTR-T, MASK-T, P-FLAG. Все эти значения обрабатываются совместно, и результат значения хранится как новое значение атрибута.

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 экране TV, 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. Эта важная маленькая подпрограмма используется во многих случаях. Если основная часть экранной области используется, то системные переменные ATTR-P и MASK-P, в системной переменной P-FLAG копируются нечётные биты (постоянные значения) в чётные (временные).

Однако когда используется нижняя часть экрана, системная переменная ATTR-T копируется из переменной BORDER и обнуляется переменная MASK-T. Все биты переменной P-FLAG, хранящие временные характеристики цвета, сбрасываются.

0D6B-0EAB. Программа команды CLS. Операция очистки экрана в SP включает в себя заполнение всех адресов экранной области нулевыми значениями адресов в области атрибутов специфицированными значениями. Программа команды использует подпрограмму CL-LINE (0E44-0E67) для очистки 24 строк экранной области. Продвижение экрана реализуется подпрограммой CL-SCROLL (0E00-0E43).

0EAC-0F2B. Программы для принтера.

0EAC-0BCC. Программа команды COPY. 176 точечных линий непосредственно на принтер (так называемый ZX-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.

0FAC-0FA8. Содержат таблицу для кодов (16-ое 070F) 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 LIST-K до тех пор, пока бит 5 FLAGS показывает, что нажата новая клавиша. Если код "печатаемый", то устанавливается флаг переносе и флаг 0.

© VA PRINT

Установка флага CAPS LOCK обрабатывается в адресах 10DD-10E5. Бит 3 FLAGS 2 дополняется каждый раз, когда эта, секция выполняется.

Установка MODE, использующая клавишу GRAPHICS и регистр, или SIMBOL SHIFT и регистр, обрабатывается в адресах 10E9-10F3.

Секция 10FA-111C Связана с установкой K-DATA, если была нажата цифровая 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, PRAMT, RASF, 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.

Основные части программы:

11CC-11CF. TV-экран становится белым.

11D0-11D9. Регистр I устанавливается на +3F. Регистр используется для генерации TV-сигналов.

11DA-11EF. Программа RAM-CHECK. Тестируются адреса, начиная с 4000 до RAMTOP. При выходе регистр, 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. При системном рестарте выполнение этой секции бессмысленно.

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-REMT инициализируются.

1219-1234. Системная переменная CHARS устанавливается равной +3000, и организуется машинный стек. Устанавливается режим 1 системы прерываний, регистровая пара IY заполняется 5C3A и разрешаются маскируемые прерывания. С этого момента клавиатура опрашивается каждые 1/50 секунд.

1235-1243. Начальная информация каналов копируется в область каналов.

1244-127B. Инициализируется ряд системных переменных. Например, переменные постоянных характеристик цвета устанавливаются так, чтобы были чёрные чернила, белая бумага и белый бордюр.

127C-1285. Начальные данные потока копируются в первые 14 адресов STRMS. Они представляют потоки от -3 до +3.

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:

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.

1391-1554 - таблица сообщений об ошибках. Фирменное сообщение находится также в этой таблице.

1555-15AE - заключительная часть программы MAIN EXECUTION. Используется для копирования строки из редакторской области 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. 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 ON E-SPACE. Одиночное место выдается 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 размер 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.

16E6-1792. Программа команд CLOSE и OPEN. Открытие канала состоит в помещении отличного от 0 значения в соответствующую переменную STRMS для этого канала. Например: PRINT PEEK 23584 содержит нормальный 0, который показывает, что канал 5 закрыт. Но после OPEN #5,"K" PRINT PEEK 23584 даст 1, и канал теперь открыт. После CLOSE #5,"K" PRINT PEEK 23584 даст 0. Эти программы проверяют, что каналы открываются и закрываются правильно.

1793-1794. В стандартном SP использование команд CAT, ERASE, FORMAT, MOVE приводит к выдаче

VA PRINT

сообщения об ошибке - "неверный поток". Эти команды используются с дисководом.

1795-1A47. Программа листинга. Вход 1795 использует MAIN EXECUTION для выработки автоматического листинга, вход 17F5 для LLIST и выход 17F9 собственно для LIST. Различные подпрограммы будут обсуждены:

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.

18C1-18BC. Эта подпрограмма 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.

19B8-19D4. Подпрограмма NEXT-ONE. Используя её, 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	Команда	Класс команды	Адрес программы, разделители
1A7A	LET	01-02	2AFF
1A7D	GO TO	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	00	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	0905	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)
1AEEE	BRIGHT	07	(1C96)
1AEF	INVERSE	07	(1C96)
1AF0	OVER	07	(1C96)
1AF1	OUT	08 00	1E7A

VA PRINT

Адрес	Команда	Класс команды	Адрес программы, разделители
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, GO TO, CONTINUE или соответствующее RETURN или NEXT возможна интерпретация других строк Бейсика прежде, чем возврат будет выполнен после достижения конца программы.

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. Программа класса команд. Требования, определяемые различными классами команд, могут быть суммированы так:

класс 00	нет других операторов;
класс 01	LET. Требуется переменная;
класс 02	LET. 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, которые 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 SPECTRUME. Вычисление 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 наборе параметров строки для каждого строкового значения. В любом случае эти 5 байт памяти содержат последнее значение, и программа помещает их

VARINT

наверх калькуляторного стека, т.е. при выполнении вычислителя выражений калькуляторный стек 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 SP значения приоритетов операций, которые ещё будут выполнены, сохраняются в машинном стеке и по частям передаются в калькуляторный стек. В машинном стеке также сохраняются литералы, определяющие какая программа калькулятора будет использоваться для различных операций. Частями вычислителя выражений являются:

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 SYNTAZ-Z. Во время проверки синтаксиса флаг нуля должен быть установлен.

2535-257F. Программа SCREENS.

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.

2795-27 AF. Таблица для преобразования кодов операций в литералы калькулятора.

27B0-27BC. Таблица приоритетов операций.

27BD-28B1. Программа FN.

28B2-2995. Подпрограмма LOD-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 калькуляторный стек.

2A52-2AB0. Подпрограмма SLIKING. Любая подстрока может быть выделена из строки, и эта подпрограмма вызывается всякий раз, когда специфицировано выделение.

2AB1-2ACB. Подпрограмма STK-STORE. Очень полезная программа, которая передает параметры текущей строки (регистры A, B, C, D, E) в калькуляторный стек. Стек расширяется на одно значение. В наборе параметров строки регистровая пара BC содержит длину строки, регистровая пара HL - начальный адрес строки и регистр A не используется и содержит нуль. Иногда регистр A содержит единицу и индицирует наличие элементов массива.

2AFF-2BF0. Программа команды LET. Эта команда 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 калькуляторного стека в регистры A, B, C, D, E. Калькуляторный стек уменьшается на одно значение.

В системе SP 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 приведёт 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 важными из них являются:

2D22-2D27. Подпрограмма STK-DIGIT. Цифровой код ASCII (шестнадцатеричное 3039) передаётся 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 позаботиться об этом, 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 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 нет действительно числовыми.

© YA PRINT

При своей работе эта подпрограмма использует специальный буфер печати в 10 адресах MEM3, MEM4 (системные переменные 5CA1-5CAA).

2F9B-386D. Подпрограмма калькулятор. Различные части этой подпрограммы будут обсуждены по очереди:

2F9B-300E. Набор различных арифметических подпрограмм.

300F-3013. Литерал 03-вычитание. Первая из 4 основных арифметических программ.

В программе вычитаний: операция выполняется между двумя верхними значениями калькуляторного стека. Самое верхнее значение калькуляторного стека вычитается из следующего, и результатом всегда является одинарное значение на самом верху калькуляторного стека. Поэтому стек уменьшается в размере на 1 значение.

Вычитание рассматривается как сложение С отрицательным слагаемым, т.е. A-B выполняется как A+(-B).

3014-30A8. Литерал 0F-СЛОЖЕНИЕ, два значения наверху калькуляторного стека суммируются и замещаются результатом.

30CA-31AE. Литерал 04-умножение. Два значения наверху калькуляторного стека умножаются.

31AE-3210. Литерал 05-деление. Два значения наверху калькуляторного стека делятся - верхнее значение на значение под ним.

32C5-32D6. Таблица констант для калькулятора. Пять значений могут быть найдены в этой таблице. Значения хранятся в упакованном виде. Это значения 0, 1, 1/2, K/2, 10. Эти константы используются литералами A0-A5 и помещаются наверх калькуляторного стека. Стек увеличивается на одно значение.

32D7-335A. Калькуляторная таблица адресов. Эта таблица содержит адреса 66 программ, которые вызываются использованием калькуляторных литералов. Информация, содержащаяся я таблице, показана ниже.

Литерал	Адрес	Название	Литерал	Адрес	Название
00	368F	JUMP-TRUE	16	353B	STR-GTR
01	343C	EXCBANGE	17	359C	STRS-ADD
02	33A1	DELETE	18	35DE	VAL$
03	300F	SUBSTRAON	19	34BC	USR-S
04	30CA	MULTIPLY	1A	3645	READ-IN
05	31AF	DIVISION	1B	346E	NEGATE
06	3851	TO-POWER	1C	3669	CODE
07	351B	OR	1D	35DE	VAL
08	3524	NO-&-NO	1E	3674	LEN
09	353B	NO-L-EQ	1F	37B5	SIN
0A	353B	NO-GR-EQ	20	37AA	COS
0B	353B	NOS-NEQL	21	37DA	TAN
0C	353B	NO-GTR	22	3833	ASN
0D	353B	NO-LESS	23	3843	ACS
0E	353B	NOS-EQL	24	37E2	ATN
0F	3014	ADDITION	25	3713	IN
10	352D	STR-&-NO	26	36C4	EXP
11	353B	STR-L-EQ	27	36AF	INT
12	353B	STR-GR-EQ	28	384A	SQR
13	353B	STRS-NEQL	29	3492	SGN
14	353B	STR-GTR	2A	346A	ABS
15	353B	STR-LESS	2B	34AC	PEEK
2C	34A5	IN	38	369B	END-CALC
2D	34B3	USR-NO	39	3783	GET-ARGT
2E	361F	STRS	3A	3214	TRUNCATE
2F	35C9	CHR$	3B	33A2	FP-CALC-2
30	3501	NOT	3C	2D4F	E-TO-FP
31	33C0	DUPLICATE	3D	3297	RESTACK
32	36Л0	N-MOD-M	86	3449	SERIES-06 ETC
33	3686	JUMP	AD	341B	CONSTANTS
34	33C6	STK-DATA	C0	342D	ST-MEM-0 ETC
35	367A	DEC-JR-NZ	F0	340E	GET-MEM-0 ETC
36	3506	LESS-0			
37	34F9	GREATER-0			

VAPRINT

3351-33A1. Управляющая программа калькулятора.

33A2-33A8. Литерал 33 - 'FP-CALC-2'. Эта важная программа используется вычислителем выражений при выполнении арифметических операций. Для того, чтобы использовать эту подпрограмму, литерал арифметической операции должен быть помещен в регистр B, прежде чем используется команда RST 0028.

35DE-361E. Литерал ID-VAL. В основном подпрограммы в калькуляторе несколько сложны и выходят за рамки этой книги. Однако эту частную программу стоит рассмотреть несколько детально.

Она выполняет:

- ✓ параметры строки получает из калькуляторного стека;
- ✓ в рабочей области отводится достаточное место для строки, включая и символ возврата каретки;
- ✓ строка копируется в рабочую область и дополняется символом возврата каретки;

- ✓ бит 7 FLAGS сбрасывается, и вызывается вычислитель выражений;
- ✓ далее, если выражение было числовым, флаг синтаксиса будет установлен, и вычислитель выражений вызывается еще раз;
- ✓ последнее значение в калькуляторном стеке будет содержать требуемый результат.

36AF-386D. Подпрограммы функций. В системе SP для вычисления функций EXP, LN, SIN, ATN используются полиномы Чебышева. Таблицы констант для этих подпрограмм находятся:

36D6-36F6	EXP	- 8 констант;
3752-377E	LN	-12 констант;
37BF-37D0	SIN	- 6 констант;
3803-382E	ATN	- 2 константы.

Эти константы хранятся в упакованном формате и могут быть распакованы подпрограммой STK-DATA (регистровая пара DE загружается адресом, куда надо поместить результат, регистровая пара HL - основным адресом константы и используется CALL 33C6).

Объясним теперь упакованный вид:

1. Первый байт делится на 40 (16-ное), и экспонента будет равна:
   - ✓ если остаток существует, то остаток - 16-ное +50;
   - ✓ если остатка нет, то 2-ой байт +50;
2. Частное, которое может быть 0, 1, 2, 3 и показывает, как много ещё байтов будет специфицировано. Во всех случаях число байт будет частное +1. Поскольку мантисса имеет 4 байта, то любые не специфицированные байты зануляются.

Следующие примеры иллюстрируют метод, используемый в SP. 0-упакованная форма 00BC00 - байт 00 делится на 40. Остатка нет, так что экспонента B0+50, которая равна 00, и мантисса 3-й байт 00 и 3 неспецифицированных байта.

1 - упакованная форма 40 B0 00 00, как и выше, экспонента будет 00. Частное 1 и два последующих байта специфицируются 00 00 01 00 00.

½ - упакованный вид 30 00, распакованный вид 80 00 00 00 00.

N/2 - упакованный вид F1 49 0F DA A2, распакованный вид 81 49 0F DA A2.

10 - упакованный вид 40 B0 00 0A, распакованный вид 00 00 0A 00 00.

При использовании; упакованной формы как внешней, любые пять байтов могут быть переданы в калькуляторный стек. Байты в упакованном виде помещаются после литерала 34 - STK-DATA.

Например, чтобы сделать значение 10 последним значением калькуляторного стека надо:

RST 0028	использовать калькулятор;
34	использовать STK-DATA;
40 B0 00 0A	упакованный вид;
38	использовать END-CALE;

Литерал 3D - RESTACK также стоит рассмотреть. Эта подпрограмма изменяет последнее значение из интегральной формы в плавающую.

Например, для значения 10 00 00 0A 00 00 будет изменено на 84 20 00 00 00.

Эта подпрограмма не оказывает действия на значения, которые уже находятся в плавающей форме.

386E-3CFF. Набор литер (см. стр. 67 для детализации).

VA PRINT

8. ИСПОЛЬЗОВАНИЕ ПОДПРОГРАММ МОНИТОРА

8.1. ВВЕДЕНИЕ

Цель этой главы показать, что программы машинного кода могут быть написаны для SP относительно просто, используя большое число подпрограмм, которые всегда имеются в программе монитора.

Во многих случаях метод основан на Бейсике, т.к. соответствующие программы машинного кода разработаны, чтобы заменить простые предписания Бейсика. Эти программы могут быть вызваны в функции USR-ЧИСЛО. Основная программа кода может быть сформирована, объединив вместе ряд небольших программ машинного кода.

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. ШЕСТНАДЦАТЕРИЧНЫЙ ВВОД

В главе 6 программа HEX.LOADER (16-ный загрузчик) была так представлена, что читатель может составить программу машинного кода в списке данных DATA и потом выполнить её. Однако HEX.LOADER не совсем подходит для больших программ и предпочтительна следующая программа:

```
10 LET D=32000: REM HEX INPUT
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))+FNC$(E-16*INT(E/16))
60 DIM A$(2)
70 PRINT FN H$(PEEK D); TAB7; FN H$(INT(D/256)); FN H$(D-256*INT(D/256))
80 INPUT A$
90 LET L=1
100 IF A$(1)="U" THEN LET L=-1: GO TO 130
110 IF A$(1)="*" THEN GO TO 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 GO TO 70
160 INPUT "PRESS ANY KEY TO RUN THE ROUTINE"; A$
170 RANDOMIZE USR 32000
```

Примечание: используйте программу CAPS LOCK SET:

- ✓ введите значение как пару 16-ных символов;
- ✓ используйте ENTER для выполнения шага вперед;
- ✓ используйте "U" и ENTER, выполняя шаг вперед;
- ✓ 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 осведомлён о программе SAVE перед тем, как использовать её впервые, так как имеют место частые аварийные ситуации в системе.

Читатель будет рад изменить вышеприведенную программу, чтобы выполнить какое-либо определенное желание, например, 16-ные символы в "нижнем регистре" и т.д.

VA PRINT

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, в которых программа BEEP может быть выполнена. По первому методу необходимо вызвать подпрограмму BEEPER с соответствующими величинами в двух 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 секунда
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 регистров 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 F*T. Это задается в регистровой паре DE.
- ✓ - величина для регистровой пары HL представлена как 437.500/F-30.125.

Заметьте, что нет ограничения длительности 10 секундами при использовании этого метода.

Метод 2

Используйте программу HEX.INPUT, чтобы ввести программу:

```
7D00 3E 01          LD A,+01          длительность 1 секунда
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, но подпрограмма STK-DATA должна использоваться для других величин, т.е. для BEEP 1.3,1.12 она будет:

```
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 CALE
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 - счётчик байтов, например, переслать байты с 7E00 по 7EFF включительно. Регистровая пара IX будет содержать 7E00, а DE - 0100

В регистре 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 блока 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
```

VA PRINT

8.5. ЭЛЕМЕНТЫ ЦВЕТА

В системе SP все байты атрибутов имеют следующий вид:

бит 7 - 1 для FLASH

бит 6 - 1 для BRIGHT

биты 5-3 цвет PAPER

биты 2-0 цвет 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.

Чётные биты P-FLAG (биты 6, 4, 2, 0) являются временными флагами, а нечётные биты (7, 5, 3, 1) являются постоянными флагами.

В большинстве случаев SP использует временные величины, когда заполняется значение для атрибутной области. 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 изменяться путём 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, а BORDER заполняться новым значением.

```
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 регистр A
7D06 07             RLCA
7D07 CB 6F          BIT 5,A           цвет чернил для нижней
7D09 20 02          JR NZ,BORD-2      части экрана - контрастен
7D0B EE 07          XOR +07
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 задаётся битами 2-0 ATTR-P, Следующая программа показывает, как эти биты могут изменяться для INK 0 - INK

```
7D00 3E 8F          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 IY+53, 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 3 состояния FLASH следующим образом: установка и сброс бита 7 в ATTR-P и MASK-P:

FLASH 0 0 для7-го бита ATTR-P
FLASH 1 1 для 7-го бита ATTR-P
FLASH 8 1 для 7-го бита MASK-P

BRIGHT

© VA PRINT

Подобным образом задаются 3 состояния яркости: установка и сброс бита 6 в ATTR-P и MASK-P:

BRIGHT 0 0 для 6-го бита ATTR-P 'RES 6,(ATTR-P)'
BRIGHT 1 1 для 6-го бита ATTR-P 'SET 6,(ATTR-P)'
BRIGHT 8 1 для 6-го бита MASK-P 'SET 6,(MASK-P)'

Во всех случаях, когда 8-ой режим требует исключения соответствующие биты MASK-P нужно сбросить опять.

OVER

Два состояния OVER могут быть представлены следующим образом: установка и сброс 1 -го бита в P-FLAG:

OVER 1 0 для 1 -го бита P-FLAG 'SET 1,(P-FLAG)'
OVER 0 1 для 1 -го бита P-FLAG 'RES 1,(P-FLAG)

INVERSE

Два состояния INVERSE могут быть представлены следующим образом: установка и сброс бита 3 в P-FLAG аналогично OVER.

Постоянные и переменные.

Большинство операций в SP используют переменные величины. Например, операция RST 0010 является важной программой для использования постоянных цветов.

Чтобы копировать текущее постоянное значение во временные системные, переменные используется CALL TEMPS (CALL 0D4D) и если переменные величины будут использоваться как новые постоянные величины, то 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. КОМАНДЫ CLS И SCROLL

В программе монитора SP имеются подпрограммы для очистки части экранной области и для сдвига ее части. Они вызываются только из программ машинного кода.

CLS

Полная очистка экрана и установка всех атрибутных байтов в постоянные величины получена путём использования: CALL CLS (CALL 0D6B), но важно убедиться, что канал S 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
```

Вышеприведённая программа имеет преимущество при использовании постоянных цветов, но проще во многих случаях использовать подпрограмму CL-LINE.

CL-LINE

Эта подпрограмма используется для очистки определенного числа строк экрана. Эти строки считаются снизу. Постоянные величины используются, когда установлены атрибутные байты.

До того, как сделан вызов подпрограммы, в регистр B должна быть помещена величина в диапазоне 01-18 16-ной, где 18-очистить весь экран.

Следующая программа показывает, как используется эта подпрограмма:

```
7D00 06 17          LD B,+17          очистить 23 (10-ое) строки
7D02 CD 44 0E       CALL CL-LINE      оставить одну верхнюю
7D05 C9             RET
```

CL-SCROLL

Эта интересная подпрограмма, т.к. она даёт возможность пользователю сдвигать определенное число строк дисплея. Подпрограмма кончается использованием подпрограммы CL-LINE для очистки нижней строки и, следовательно, атрибутные байты для этой строки дают постоянные значения.

Регистр B опять используется для хранения величины, которая обуславливает число строк, которые должны передаваться, но в этом случае она является "действительным числом строк без 1", т.е. 16-ный диапазон равен 01-17 (минимально требуется число 2, т.к. это число строк, которое должно быть сдвинуто).

Следующая подпрограмма 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
```

VA PRINT

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 строка `LD A,+02  CALL CHAN-OPEN` должна использоваться до того, как вызываются любые подпрограммы.

RST 0010

В системе SP вся печать символов на экране выполняется при использовании этой команды. С открытым каналом 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"
7D0C D7             RST 0010          оператор
7D0D 3E 08          LD A,+05          PRINT AT 5,0
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 10          LD A,"INK"        оператор INK 6
7D27 D7             RST 0010          (временное)
7D28 3E 06          LD A,"YELLOW"
7D2A D7             RST 0010
7D2B 3E 42          LD A,"B"          оператор
7D2D D7             RST 0010          PRINT "B"
7D2E 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 устанавливает).

Когда используется инструкция RST 0010 для печати цветного элемента, тогда потребуется два отдельных вызова, а когда используется символ контроля AT или TAB, 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 и "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 требует

```
LD BC,+1321
CALL 0DD9
```

Печать строк

Строка символов в SP всегда задаётся парами регистров DE и BC, где в регистре DE хранится адрес ячейки, содержащей первый символ строки, а в паре регистров BC содержится величина, равная числу символов в строке.

В программе команды PRINT используется подпрограмма PR-STRING для печати любой строки.

Даётся подробное рассмотрение подпрограммы:

```
PR-STRING=203C
    LD A,B          выбор старшего байта
```

VA PRINT

```
        OR C            операция OR с младшим
        DEC BC          уменьшение счётчика
        RET Z
        LD A, (DE)      поиск символа
        INC DE          печать символа
        RST 0010
        JR PR-STRING    переход к следующему
```

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, то строка может содержать коды символов, цветные элементы, атрибуты положения печати и кодов ключевых слов.

Как пример использования подпрограммы PR-STRING выполните следующую программу.

```
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       CAAL CL-LINE
7D0A 11 14 7D       LD DE,+STRING       начало строки
7D0D 01 0F 00       LD BC,+LENGTH       длина строки
7D10 CD 3C 20       CAAL PR-STRING      печать строки
7D13 C9             RET
7D14 11 01 10 09                        строка длиной 0F (16-ное)
7D18 16 0A 05 53                        начинающаяся в 7D14
7D1C 70 65 63 74
7D20 72 75 6D
```

Вышеприведенная программа подобна **`CLS: PRINT PAPER 1;INC 9;AT 10,5;"SP"`**

Печать чисел

Очень мощная подпрограмма PRINT-FP используется для печати в десятичной форме любых 5-ти байтов 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-CALE'
7D10 CD E3 2D       CALL PRINT-FP       печатать верхнее значение
7D13 C9             RET                 в десятичном формате
```

Использование строки RANDOMIZE USR 32000

RANDOMIZE USR 32000: PRINT PI/2 показывает идентичность результата. Но интересно рассмотреть, что при получении PI/2 постоянная PI/2 выбиралась из таблицы констант, удвоенных, а затем деленных 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 путём использования подпрограмм OUT-NUM, которая обычно используется для воспроизведения номера строки в листинге или сообщении. Когда используется программа OUT-NUM-1 (16-ная) 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".

Заметьте, что, пользуясь этой подпрограммой, ведущие пробелы печатаются, и это может быть вполне полезно.

Следует обратить внимание: если эти подпрограммы используются с числами, это может привести к превышению десятичного предела 9999.

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
```

© VA PRINT

```
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 точек в десятичном 256×176, 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, которые вызывают программу CLS.

Заметьте, что в Бейсике три команды обычно используют 8: FLACH 8, BRIGHT 8, PAPER 8, то есть MASK-T содержит 16-ное F8.

PLOT

Эта самая простая из трёх 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 наверху калькуляторного стека. Величина X находится ниже величины Y. Использование STACK-A для помещения значений в стек очень важно.

2. CALL PLOT-1 - "CALL 22DF".

Вход в этой точке требует, чтобы в регистре B хранилась величина Y, а регистре C - величина X.

После того, как точка выводится, постоянные цветовые величины копируются автоматически в (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
```

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.

DRAW X,Y

Имеются две соответствующие точки входа для этой команды:

Точка входа DRAW-1 (2477) с величинами X и Y, которые являются верхними значениями калькуляторного стека. Постоянные цветовые величины переписываются во временные системные переменные после того, как линия уже нарисована.

Точка входа DRAW-3 (24BA) с регистром B, содержащим ABS Y, регистром C, содержащим ABS X, регистром D, содержащим SGN X и регистром E, содержащим SGN Y.

Следующая программа показывает строку, которая должна быть изображена. 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 значение H'L'
7D0E E5             PUSH HL
7D0F D9             EXX
7D10 FD 36 43 64    LD (COORDS-X),+64 установить координаты
```

VA PRINT

```
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 выбирает строку +50,+50.

DRAW X,Y,A

Для того, чтобы изобразить дугу, 3 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 последующим 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 HL
7D0E E5              PUSH HL
7D0F D9              EXX
7D10 FD 36 43 64     LD (COORDS-X),+64 установить r
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-CALE
7D21 CD 94 23        CALL DRAW-ARC изображение
7D24 D9              EXX
7D25 E1              POP HL            восстановить HL
7D26 D9              EXX
7D27 C9              RET
```

В программе величины, полученные для X, Y, A будут десятичными +50,+50,+1.

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 калькулятор
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-CALE
7D1D CD 2D 23        CALL CIRCLE-1     изображение окружности
7D20 D9              EXX
7D21 E1              POP HL            восстановить HL
7D22 D9              EXX
7D23 C9              RET
```

Вышеприведенная программа дает тот же результат, как **`CIRCLE 100,100,48`**. Заметьте, что при

© VA PRINT

использовании вышеприведенной программы постоянные цветовые величины копируются во временные системные, одновременно с изображением дуги или окружности.

8.9. POINT, ATTR, SCREEN$

Эти три функции могут вызываться из программы машинного кода очень просто, так как 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 регистр BC, и вызывается подпрограмма POINT-1 (22CE). Как обычно, регистр B хранит координату Y, регистр C хранит координату X.

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 41          LD A,"A"          печать 'A'
7D0F D7             RST 0010
7D10 3E 0D          LD A,"CR"         переход к, следующей строке
7D12 D7             RST 0010
7D13 01 04 AE       LD BC,+AE04       оператор POINT (4,174)
7D16 CD CE 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 C, а номер колонки в регистр B, и вызывается подпрограмма как 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 слова
7D15 D7             RST 0010          COPY
7D16 3E 0D          LD A,"CR"         переход к следующей строке
7D18 D7             RST 0010
7D19 01 00 04       LD BC,-Ю400       оператор ATTR (0,4)
7D1C CD 83 25       CALL ATTR-1
7D1F CD E3 2D       CALL PRINT-FP
7D22 C9             RET
```

Вышеприведённая программа аналогична команде **PRINT PAPER 5 CHR$ 255;CHR$ 13;ATTR (0,4)**.

SCREEN$

Опять номер строки засылается в регистр C, а номер колонки в регистр B. Подпрограмма тогда вызывается путем использования CALL SCREENS-1 (2538). Результат возвращается как входное значение калькуляторного стека, 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 строке
7D12 D7             RST 0010
```

© VA PRINT

```
7D13 01 00 00       LD BC,+0000       оператор SCREENS (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 набор символов и, следовательно, ищет коды в диапазоне 16-ном 20-7F.

Ошибка SCREEN$. Двойное запоминание результата не будет иметь место, когда используется входная точка SCREEN$-1.

8.10. PI, RND, INKEY$

Эти функции группируются вместе, потому что их программные вычисления включают вычислитель выражений и невозможно вычислить их непосредственно.

Однако, с 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

Фактическая процедура вычисления для PI будет:

```
PI=262C
262C                RST 0028          использование калькулятора
262D A3             "PI/2"
262E 3B             "END-CODE"
262F                INC (HL)          удвоение увеличением экспоненты
```

Процедура занимает 4 адреса и легко копируется. Её можно вызвать 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"         кодP1
7D0F CD 28 2D       CALL STACK-A
7D12 EF             RST 0028          использовать калькулятор
7D13 2F             DEFB +2F
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-COLE
7D1A CD E3 2D       CALL PRINT-FP
7D1D C9             RET
```

RND

При включении питания или NEW величина в SEED=0, затем, с помощью 1-го обращения к RND величина SEED становится 74 (10-ное).

Это даёт первое случайное число: 74/65535=0,0011291676 последовательность чисел для SEED 0, 74, 5624, 28652…0, 74, т.е. представлены все числа от 0 до 65535.

Правило для изменения величины SEED следующее:

1. Прибавьте 1.
2. Умножьте на 75.
3. Возьмите по модулю 65535, т.е. остаток после деления на 65535.
4. Вычтите 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-65535*INT(SEED/65535)
60 LET SEED=SEED-1
70 PRINT "NEW VALUE="; CHR$ 32; SEED
80 GO TO 30
```

В программе монитора практическая процедура вычисления RND будет следующей.

```
RND=25FD
```

© VA PRINT

```
RND-END=2625
25FD                LD BC, (SEED)     выборка значения SEED
2601                CALL STACK-BC
2604                RST 0028          использование калькулятора
2605 A1                               стек "1".
2606 0F                               SEED-SEED*75
260A 04
260B 34
260C 80 41 00 00 80                   стек "65535"
2611 32                               "N-MOD-M"
2612 02                               "DELETE"
2613 A1                               стек "1"
2614 03                               SEED=SEED-1
2615 31                               "DUPLICATE"
2616 38                               "END-CALE"
2617                CALL FP-TO-BC     оператор "CALL 2DA2"
261A                LD (SEED),BC      новое значение SEED
261E                LD A,(HL)         выбор экспоненты
261F                AND A             проверка на 0
2620                JR Z,RND-END
2622                SUB +10           деление на 65535
2624                LD (HL),A         сохранение экспоненты
2625 ……
```

Величина для RND теперь представляется верхним значением в калькуляторном стеке. Программа RND может вызываться из программы в машинных кодах рассмотрением результата вычисления.

VAL CHR$ 165

Программа для вычисления PI, данная на странице 203, может использоваться с единственным изменением:

```
7D0D 3E A5          LD A,+A5
```

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 значение клавиши находится по известным таблицам.
4. Действие выполняется по программе INKEY$ (2646 16-ное) могут быть повторены, т.е.

```
CALL KEY-SCAN:      CALL 028E
LD C,+00            обеспечение режимов K, L, C
JR NZ,NO-KEY:       не была нажата клавиша
CALL KEY-TEST       регистр или более 2 клавиши
JR NC,NO-KEY:
DEC D               обеспечение режима L или
LD E,A              получение номера
CALL KEY-CODE       CALL 0333
```

И код символа помещается в регистр A.

5. Программа INKEY$ может использоваться вычислением выражения VAL$ CHR$ 166. В следующей программе демонстрируется эта возможность.

```
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 11 00 08       LD DE,+0800       установка счётчика на 2048
7D10 D5             PUSH DE
7D11 3E A6          LD A,"INKEY$"
7D13 CD 2B 2D       CALL STACK-A
7D16 EF             RST 0028          использование калькулятора
7D17 2F             DEFB +2F          DAVZ
7D18 18             DEFB +18          VAL$
7D19 38             DEFB +38          END CALE
7D1A CD F1 2B       CALL STK-FETCH    выбор параметров
7D1D CD 3C 20       CALL PR-STRING    печать строки
7D20 CD BF 16       CALL X-TEMP       очистка рабочей области
7D23 D1             POP DE            построение цикла
```

VA PRINT

```
7D24 7B             LD A,E
7D25 B2             OR D
7D26 C8             RET Z
7D27 1B             DEC DE            уменьшение счётчика
7D28 18 E6          JR LOOP
```

Когда эта программа выполняется, то она аналогична выражению: **FOR A=1 TO 2048: PRINT INKEY$: NEXT A**

Возможно, адреса 7D19-7D1F изменить на `1C, 38, CD, E3, 2D, 00, 00`. Что даст **FOR A=1 TO 2048: PRINT CODE INKEY$: NEXT A**

8.11. BREAK

Во многих случаях полезно, чтобы выход был сделан из программы машинного кода по клавише BREAK.

Пример:

```
BREAK=7D0D
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 7F          LD A.+7F          вводной порт 7FFF
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
```

В программе половина ряда клавиатуры от BREAK до B неоднократно просматривается. Выход будет осуществлен только тогда, когда установится бит, который представляет ключ 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". Вся операция может затем повторятся для другой задачи, и, в конце концов, программа Бейсик будет состоять только из одной строки: **10 RANDOMIZE USR** ....., так как никаких возвратов не будет сделано до тех пор, пока программа машинного кода не будет выполнена.

VA PRINT

ПРИЛОЖЕНИЕ I

ТАБЛИЦА СИМВОЛОВ

10	16	СИМВОЛ	МНЕМОНИКА Z80		
			без префикса	с префиксом #CB	с префиксом #ED
0	00		NOP	RLC B	
1	01		LD BC,nn	RLC C	
2	02		LD (BC),A	RLC D	
3	03	У	INC BC	RLC E	
4	04	П	INC B	RLC H	
5	05	Р	DEC B	RLC L	
6	06	А	LD B,n	RLC (HL)	
7	07	В	RLCA	RLC A	
8	08	Л	EX AF,AP'	RRC B	
9	09	Я	ADD HL,BC	RRC C	
10	0A	Ю	LD A,(BC)	RRC D	
11	0B	Щ	DEC C	RRC E	
12	0C	И	INC C	RRC H	
13	0D	Е	DEC C	RRC L	
14	0E		LD C,n	RRC (HL)	
15	0F		RRCA	RRC A	
16	10		DJNZ $+e	RL B	
17	11		LD DE,nn	RL C	
18	12		LD (DE),A	RL D	
19	13	К	INC DE	RL E	
20	14	О	INC D	RL H	
21	15	Д	DEC D	RL L	
22	16	Ы	LD D,n	RL (HL)	
23	17		RLA	RL A	
24	18		JR $+e	RR B	
25	19		ADD HL,DE	RR C	
26	1A		LD A,(DE)	RR D	
27	1B		DEC DE	RR E	
28	1C		INC E	RR H	
29	1D		DEC E	RR L	
30	1E		LD E,n	RR (HL)	
31	1F		RRA	RR A	
32	20	пробел	JR NZ,$+Ъ	SLA B	
33	21	!	LD HL,nn	SLA C	
34	22	"	LD (nn),HL	SLA D	
35	23	#	INC HL	SLA E	
36	24	$	INC H	SLA H	
37	25	%	DEC H	SLA L	
38	26	&	LD H,n	SLA (HL)	
39	27	‘	DAA	SLA A	
40	28	(	JR Z,$+e	SRA B	
41	29	)	ADD HL,HL	SRA C	
42	2A	*	LD HL,(nn)	SRA D	
43	2B	+	DEC HL	SRA E	
44	2C	,	INC L	SRA H	
45	2D	-	DEC L	SRA L	
46	2E	.	LD L,n	SRA (HL)	
47	2F	/	CPL	SRA A	
48	30	0	JR NC,$+e	[SLI B]	
49	31	1	LD SP,nn	[SLI C]	
50	32	2	LD (nn),A	[SLI DJ	
51	33	3	INC SP	[SLI E]	
52	34	4	INC (HL)	[SLI H]	
53	35	5	DEC (HL)	[SLI L]	
54	36	6	LD (HL),n	[SLI (HL)]	
55	37	7	SCF	[SLI A]	
56	38	8	JR C,$+Ъ	SRL B	

© VA PRINT

Таблица символов

10	16	СИМВОЛ	МНЕМОНИКА Z80		
			без префикса	с префиксом #CB	с префиксом #ED
57	39	9	ADD HL,SP	SRL C	
58	3A	:	LD A,(nn)	SRL D	
59	3B	;	DEC SP	SRL E	
60	3C	<	INC A	SRL H	
61	3D	=	DEC A	SRL L	
62	3E	>	LD A,n	SRL (HL)	
63	3F	?	CCF	SRL A	
64	40	@	LD B,B	BIT 0,B	IN B,C
65	41	A	LD B,C	BIT 0,C	OUT (C),B
66	42	B	LD B,D	BIT 0,D	SBC HL,BC
67	43	C	LD B,E	BIT 0,E	LD (nn),BC
68	44	D	LD B,H	BIT 0,H	NEG
69	45	E	LD B,L	BIT 0,L	RETN
70	46	F	LD B,(HL)	BIT 0,(HL)	IM 0
71	47	G	LD B,A	BIT 0,A	LD I,A
72	48	H	LD C,B	BIT 1,B	IN C,(C)
73	49	I	LD C,C	BIT 1,C	OUT (C),C
74	4A	J	LD C,D	BIT 1,D	ADC HL,BC
75	4B	K	LD C,E	BIT 1,E	LD BC,(nn)
76	4C	L	LD C,H	BIT 1,H	
77	4D	M	LD C,L	BIT 1,L	RETI
78	4E	N	LD C,(HL)	BIT 1,(HL)	
79	5F	O	LD C,A	BIT 1,A	LD R,A
80	50	P	LD D,B	BIT 2,B	IN D,(C)
81	51	Q	LD D,C	BIT 2,C	OUT (C),D
82	52	R	LD D,D	BIT 2,D	SBC HL,DE
83	53	S	LD D,E	BIT 2,E	LD (nn),DE
84	54	T	LD D,H	BIT 2,H	
85	55	U	LD D,L	BIT 2,L	
86	56	V	LD D,(HL)	BIT 2,(HL)	IM 1
87	57	W	LD D,A	BIT 2,A	LD A,I
88	58	X	LDE,B	BIT 3,B	INE,(C)
89	59	Y	LD E,C	BIT 3,C	OUT (C),E
90	5A	Z	LD E,D	BIT 3,D	ADC HL,DE
91	5B	[	LD E,E	BIT 3,E	LD DE,(nn)
92	5C	\	LD E,H	BIT 3,H	
93	5D	]	LD E,L	BIT 3,L	
94	5E	^	LD E,(HL)	BIT 3,(HL)	IM 2
95	5F	_	LD E,A	BIT 3,A	LD A,R
96	60	£	LD H,B	BIT 4,B	IN H,(C)
97	61	a	LD H,C	BIT 4,C	OUT (C),H
98	62	b	LD H,D	BIT 4,D	SBC HL,HL
99	63	c	LD H,E	BIT 4,E	LD (nn),HL
100	64	d	LD H,H	BIT 4,H	
101	65	e	LD H,L	BIT 4,L	
102	66	f	LD H,(HL)	BIT 4,(HL)	
103	67	g	LD H,A	BIT 4,A	RRD
104	68	h	LD L,B	BIT 5,B	INL,(C)
105	69	i	LD L,C	BIT 5,C	OUT (C),L
106	6A	j	LD L,D	BIT 5,D	ADC HL,HL
107	6B	k	LD L,E	BIT 5,E	LD HL,(nn)
108	6C	l	LD L,H	BIT 5,H	
109	6D	m	LD L,L	BIT 5,L	
110	6E	n	LD L,(HL)	BIT 5,(HL)	
111	6F	o	LD L,A	BIT 5,A	RLD
112	70	p	LD (HL),B	BIT 6,B	IN (HL),(C)
113	71	q	LD (HL),C	BIT 6,C	
114	72	r	LD (HL),D	BIT 6,D	SBC HL,SP
115	73	s	LD (HL),E	BIT 6,E	LD (nn),SP
116	74	t	LD (HL),H	BIT 6,H	
117	75	u	LD (HL),L	BIT 6,L	
118	76	v	HALT	BIT 6,(HL)	

© VA PRINT

10	16	СИМВОЛ	МНЕМОНИКА Z80		
			без префикса	с префиксом #CB	с префиксом #ED
119	77	w	LD (HL),A	BIT 6,A	
120	78	x	LD A,B	BIT 7,B	IN A,(C)
121	79	y	LD A,C	BIT 7,C	OUT (C),A
122	7A	z	LD A,D	BIT 7,D	ADC HL,SP
123	7B	{	LD A,E	BIT 7,E	LD SP,(nn)
124	7C	|	LD A,H	BIT 7,H	
125	7D	}	LD A,L	BIT 7,L	
126	7E	~	LD A,(HL)	BIT 7,(HL)	
127	7F	©	LD A,A	BIT 7,A	
128	80	□	ADD A,B	RES 0,B	
129	81	▗	ADD A,C	RES 0,C	
130	82	▖	ADD A,D	RES 0,D	
131	83	▄	ADD A,E	RES 0,E	
132	84	▐	ADD A,H	RES 0,H	
133	85	▌	ADD A,L	RES 0,L	
134	86	▚	ADD A,(HL)	RES 0,(HL)	
135	87	▙	ADD A,A	RES 0,A	
136	88	▝	ADC A,B	RES 1,B	
137	89	▚	ADC A,C	RES 1,C	
138	8A	▌	ADC A,D	RES 1,D	
139	8B	▟	ADC A,E	RES 1,E	
140	8C	▀	ADC A,H	RES 1,H	
141	8D	▛	ADC A,L	RES 1,L	
142	8E	▜	ADC A,(HL)	RES 1,(HL)	
143	8F	■	ADC A,A	RES 1,A	
144	90	[A]	SUB B	RES 2,B	
145	91	[B]	SUB C	RES 2,C	
146	92	[C]	SUB D	RES 2,D	
147	93	[D]	SUB E	RES 2,E	
148	94	[E]	SUB H	RES 2,H	
149	95	[F]	SUB L	RES 2,L	
150	96	[G]	SUB (HL)	RES 2,(HL)	
151	97	[H]	SUB A	RES 2,A	
152	98	[I]	SBC A,B	RES 3,B	
153	99	[J]	SBC A,C	RES 3,C	
154	9A	[K]	SBC A,D	RES 3,D	
155	9B	[L]	SBC A,E	RES 3,E	
156	9C	[M]	SBC A,H	RES 3,H	
157	9D	[N]	SBC A,L	RES 3,L	
158	9E	[O]	SBC A,(HL)	RES 3,(HL)	
159	9F	[P]	SBC A,A	RES 3,A	
160	A0	[Q]	AND B	RES 4,B	LDI
161	A1	[R]	AND C	RES 4,C	CPI
162	A2	[S]	AND D	RES 4,D	INI
163	A3	[T]	AND E	RES 4,E	OUTI
164	A4	[U]	AND H	RES 4,H	
165	A5	RND	AND L	RES 4,L	
166	A6	INKEY$	AND (HL)	RES 4,(HL)	
167	A7	PI	AND A	RES 4,A	
168	A8	FN	XOR B	RES 5,B	LDD
169	A9	POINT	XOR C	RES 5,C	CPD
170	AA	SCREEN$	XOR D	RES 5,D	IND
171	AB	ATTR	XOR E	RES 5,E	OUTD
172	AC	AT	XOR B	RES 5,H	
173	AD	TAB	XOR L	RES 5,L	
174	AE	VAL$	XOR (HL)	RES 5,(HL)	
175	AF	CODE	XOR A	RES 5,A	
176	B0	VAL	OR B	RES 6,B	LDIR
177	B1	LEN	OR C	RES 6,C	CPIR
178	B2	SIN	OR D	RES 6,D	INIR
179	B3	COS	OR E	RES 6,E	OTIR
180	B4	TAN	OR H	RES 6,H	

© VA PRINT

10	16	СИМВОЛ	МНЕМОНИКА Z80		
			без префикса	с префиксом #CB	с префиксом #ED
181	B5	ASN	OR L	RES 6,L	
182	B6	ACS	OR (HL)	RES 6,(HL)	
183	B7	ATN	OR A	RES 6,A	
184	B8	LN	CP B	RES 7,B	LDDR
185	B9	EXP	CP C	RES 7,C	CPDR
186	BA	INT	CP D	RES 7,D	INDR
187	BB	SQR	CP E	RES 7,E	OTDR
188	BC	SGN	CP H	RES 7,H	
189	BD	ABS	CP L	RES 7,L	
190	BE	PEEK	CP (HL)	RES 7,(HL)	
191	BF	IN	CP A	RES 7,A	
192	C0	USR	RET NZ	SET 0,B	
193	C1	STR$	POP BC	SET 0,C	
194	C2	CHR$	JP NZ,nn	SET 0,D	
195	C3	NOT	JP nn	SET 0,E	
196	C4	BIN	CALL NZ,nn	SET 0,H	
197	C5	OR	PUSH BC	SET 0,L	
198	C6	AND	ADD A,n	SET 0,(HL)	
199	C7	<=	RST 0	SET 0,A	
200	C8	>=	RET Z	SET 1,B	
201	C9	<>	RET	SET 1,C	
202	CA	LINE	JP Z,nn	SET 1,D	
203	CB	THEN	префикс #CB	SET 1,E	
204	CC	TO	CALL Z,nn	SET 1,H	
205	CD	STEP	CALL nn	SET 1,L	
206	CE	DEF FN	ADC A,n	SET 1,(HL)	
207	CF	CAT	RST 8	SET 1,A	
208	D0	FORMAT	RET NC	SET 2,B	
209	D1	MOVE	POP DE	SET 2,C	
210	D2	ERASE	JP NC,nn	SET 2,D	
211	D3	OPEN	OUT (N),A	SET 2,E	
212	D4	CLOSE	CALL NC,nn	SET 2,H	
213	D5	MERGE	PUSH DE	SET 2,L	
214	D6	VERIFY	SUB n	SET 2,(HL)	
215	D7	BEEP	RST 16	SET 2,A	
216	D8	CIRCLE	RET C	SET 3,B	
217	D9	INK	EXX	SET 3,C	
218	DA	PAPER	JP C,nn	SET 3,D	
219	DB	FLASH	IN A,(N)	SET 3,E	
220	DC	BRIGHT	CALL C,nn	SET 3,H	
221	DD	INVERSE	префикс для IX	SET 3,L	
222	DE	OVER	SBC A,n	SET 3,(HL)	
223	DF	OUT	RST 24	SET 3,A	
224	E0	LPRINT	RET PO	SET 4,B	
225	E1	LLIST	POP HL	SET 4,C	
226	E2	STOP	JP PO,nn	SET 4,D	
227	E3	READ	EX (SP),HL	SET 4,E	
228	E4	DATA	CALL PO,nn	SET 4,H	
229	E5	RESTORE	PUSH HL	SET 4,L	
230	E6	NEW	AND n	SET 4,(HL)	
231	E7	BORDER	RST 32	SET 4,A	
232	E8	CONTINUE	RET PE	SET 5,B	
233	E9	DIM	JP (HL)	SET 5,C	
234	EA	REM	JP PE,nn	SET 5,D	
235	EB	FOR	EX DE,HL	SET 5,E	
236	EC	GO TO	CALL PE,nn	SET 5,H	
237	ED	GO SUB	префикс #ED	SET 5,L	
238	EE	INPUT	XOR n	SET 5,(HL)	
239	EF	LOAD	RST 40	SET 5,A	
240	F0	LIST	RET P	SET 6,B	
241	F1	LET	POP AF	SET 6,C	
242	F2	PAUSE	JP P,nn	SET 6,D	

© VA PRINT

10	16	СИМВОЛ	МНЕМОНИКА Z80		
			без префикса	с префиксом #CB	с префиксом #ED
243	F3	NEXT	DI	SET 6,E	
244	F4	POKE	CALL P,nn	SET 6,H	
245	F5	PRINT	PUSH AF	SET 6,L	
246	F6	PLOT	OR n	SET 6,(HL)	
247	F7	RUN	RST 48	SET 6,A	
248	F8	SAVE	RET M	SET 7,B	
249	F9	RANDOMIZE	LD SP,HL	SET 7,C	
250	FA	IF	JP M,nn	SET 7,D	
251	FB	CLS	EI	SET 7,E	
252	FC	DRAW	CALL M,nn	SET 7,H	
253	FD	CLEAR	префикс для IY	SET 7,L	
254	FE	RETURN	CP n	SET 7,(HL)	
255	FF	COPY	RST 56	SET 7,A	

ПОЯСНЕНИЯ К ТАБЛИЦЕ СИМВОЛОВ

В первых двух столбцах таблицы приведены коды десятичный (10) и шестнадцатеричный (16).

Условные обозначения, встречающиеся в таблице:

n 1 байт данных, следующий сразу за кодом команды. Он всегда является последним байтом команды.

nn 16-разрядный операнд (2 байта данных, следующие сразу за кодом команды). Порядок расположения: вначале младший, затем старший.

(nn) 16-разрядный адрес, указанный сразу за кодом команды (сначала младший, затем старший байт).

e 8-разрядное смещение относительно адреса в счетчике PC. Располагается сразу за кодом команды.

ПРАВИЛА ПОСТРОЕНИЯ КОДОВ КОМАНД

8-разрядная шина позволяет реализовать непосредственно не более 256 команд. Чтобы снять это ограничение, зарезервированы 4 служебных кода (203 #CB, 221 #DD, 237 #ED и 253 #FD), которые используются как префиксы к кодам. Так, команда LD H,A имеет код #67, команда BIT 4,A - код #CB #67, а команда RRD - код #ED #67.

Байты #DD и #FD используются как префиксы к командам, работающим соответственно с индексными регистрами IX и IY. Коды команд, использующих в качестве операндов индексные регистры IX и IY, образуются путём добавления префикса к кодам команд, работающих с HL (но без префикса #ED).

Коды команд, использующих индексную адресацию, образуются точно также, только префиксироваться будут команды, работающие с ячейкой памяти, адресуемой (HL). При этом код смещения d всегда является третьим байтом получающегося кода.

Коды недокументированных команд, работающих с частями индексных регистров, образуются из префикса #DD или #FD и кода команды, работающей с регистрами H и L. При этом регистру H соответствуют старшие байты регистров IX и IY (IXh и IYh), а регистру L - младшие (IXl и IYl).

В следующей таблице приведены примеры построения кодов команд, использующих индексные регистры.

ИСХОДНАЯ КОМАНДА		ПОЛУЧАЕМАЯ КОМАНДА	
Мнемоника	КОД	Мнемоника	Код
ADD HL,BC	#09	ADD IX,BC	#DD #09
JP (HL)	#E9	JP (IY)	#FD #E9
LD B,(HL)	#46	LD B,(IX+1)	#DD #46 #01
BIT 0,(HL)	#CB #46	BIT 0,(IY-1)	#FD #CB #FF #46
LDA,H	#7C	LD A,IXh	#DD #7C
LD H,L	#65	LD IXh,IXl	#DD #65
LD L,1	#2E #01	LD IYl,1	#FD #2E #01
XOR H	#AC	XOR IXh	#DD #AC

VA PRINT

УПРАВЛЯЮЩИЕ КОДЫ ZX SPECTRUM

10	16	Вводится с клавиатуры	Выводится на экран
0	00	CS/SS + CS/8	Не используется
1	01	CS/SS + CS/9	Не используется
2	02	CS/SS + 8	Не используется
3	03	CS/SS+ 9	Не используется
4	04	CS/3 (True Video)	Не используется
5	05	CS/4 (Inverse Video)	Не используется
6	06	CS/2 (Caps Lock)	Табуляция (аналогично использованию запятой в операторе Бейсика "PRINT")
7	07	CS/1 (Edit)	Не используется
8	08	CS/5	Backspace (перемещение позиции печати на 1 символ влево)
9	09	CS/8	Не используется
10	0A	CS/6	Не используется

10	16	Вводится с клавиатуры	Выводится на экран
11	0B	CS/7	Не используется
12	0C	CS/0 (Delete)	Не используется
13	0B	Enter	CR + LF (Возврат каретки + Перевод строки)
14	0E	CS/SS (Extend Mode)	Не используется
15	0F	CS/9 (Graphics Mode)	Не используется
16	10	CS/SS + 0	Код управления цветом тона (INK control)
17	11	CS/SS + 1	Код управления цветом фона (PAPER control)
18	12	CS/SS + 2	Код управления мерцанием (FLASH control)
19	13	CS/SS + 3	Код управления яркостью (BRIGHT control)
20	14	CS/SS + 4	Код управления инверсией (INVERSE control)
21	15	CS/SS + 5	код управления наложением (OVER control)
22	16	CS/SS + 6	Код управления позицией печати (AT control)
23	17	CS/SS + 7	Код управления табуляцией (TAB control)
24	18	CS/SS + CS/0	Не используется
25	19	CS/SS + CS/1	Не используется
26	1A	CS/SS + CS/2	Не используется
27	1B	CS/SS + CS/3	Не используется
28	1C	CS/SS + CS/4	Не используется
29	1D	CS/SS + CS/5	Не используется
30	1E	CS/SS + CS/6	Не используется
31	1F	CS/SS + CS/7	Не используется

©
VA PRINT

СОДЕРЖАНИЕ

ВВЕДЕНИЕ....6
1. ПРИНЦИПЫ РАБОТЫ МИКРОКОМПЬЮТЕРНОЙ СИСТЕМЫ....6
1.1. ПРЕДИСЛОВИЕ....6
1.2. ОПИСАНИЕ СИСТЕМЫ....6
1.3. Z80 - МИКРОПРОЦЕССОР....6
1.4. 16K ROM (ПОСТОЯННАЯ ПАМЯТЬ)....6
1.5. 16K RAM (ОПЕРАТИВНАЯ ПАМЯТЬ - ОЗУ)....6
1.6. ULA....6
1.7. PAL - КОДИРОВЩИК....7
Логика работы....7
1.8. ROM ОБЛАСТЬ....7
1.9. КАРТА ПАМЯТИ ЭКРАННОЙ ОБЛАСТИ....7
1.10. ОБЛАСТЬ АТРИБУТОВ....8
1.11. БУФЕР ПЕЧАТИ....8
1.12. СИСТЕМНЫЕ ПЕРЕМЕННЫЕ....8
1.13. КАРТЫ МИКРОДРАЙВОВ....8
1.14. ОБЛАСТЬ ИНФОРМАЦИОННЫХ КАНАЛОВ....8
1.15. ОБЛАСТЬ БЕЙСИК-ПРОГРАММЫ (БП)....8
1.16. ФОРМАТ СТРОК БП....8
1.17. ОВЛАСТЬ ПЕРЕМЕННЫ....8
1.18. ОБЛАСТЬ РЕДАКТОРА....9
1.19. РАБОЧАЯ ОБЛАСТЬ....9
1.20. СТЕК ВЫЧИСЛИТЕЛЯ....9
1.21. РЕЗЕРВНАЯ ПАМЯТЬ....9
1.22. МАШИННЫЙ СТЕК....9
1.23. GO SUB - СТЕК....9
Демонстрационная программа GO SUB-стека....9
1.24. ГРАФИЧЕСКАЯ ОБЛАСТЬ, ОПРЕДЕЛЕННАЯ ПОЛЬЗОВАТЕЛЕМ....10
2. ФУНКЦИИ И КОМАНДЫ БЕЙСИКА....11
2.1. ВВЕДЕНИЕ....11
2.2. КОМАНДЫ БЕЙСИКА....11
2.3. УПРАВЛЯЮЩИЕ СИМВОЛЫ....20
3. МИКРОПРОЦЕССОР Z-80....21
3.1. ВВЕДЕНИЕ....21
3.2. ЛОГИЧЕСКИЙ ВЗГЛЯД НА Z80....22
Устройство управления....22
Регистр команд (РК)....22
Программный счётчик (ПС)....22
Регистры пользователя /основные регистры/ (РП)....22
АЛУ....23
3.3. СТРУКТУРА ПРОГРАММ В МАШИННЫХ КОДАХ....23
4. МАТЕМАТИКА МАШИННОГО ПРОГРАММИРОВАНИЯ....25
4.1. ШЕСТНАДЦАТИРИЧНОЕ ПРЕДСТАВЛЕНИЕ....25
4.3. АБСОЛЮТНАЯ ДВОИЧНАЯ АРИФМЕТИКА....26
4.4. ДОПОЛНИТЕЛЬНАЯ АРИФМЕТИКА....26
4.5. ИНТЕГРАЛЬНОЕ ПРЕДСТАВЛЕНИЕ....26
4.6. ПРЕДСТАВЛЕНИЕ С ПЛАВАЮЩЕЙ ТОЧКОЙ....27
5. НАБОР СИСТЕМЫ КОМАНД....29
5.1. КОМАНДЫ И ДАННЫЕ....29
5.2. ГРУППЫ КОМАНД....29
Группа 1....29
Группа 2....29
Группа 3....30
Группа 4. Команды загрузки регистров из памяти....30
Группа 5. Команды записи в память содержимого регистров или константы....31
Группа 6. Команды сложения....32
Группа 7. Команды вычитания....33
Группа 8. Команды сравнения....34
Группа 9. Команды логики....34
Группа 10. Команды перехода....36
Группа 11. Команды DJNZ....38
Группа 12. Команды стека....38
Группа 13. Команды сдвига....40

© ZX PRINT

Группа 14. Команды обработки битов 41
Группа 15. Команды обработки блоков 42
Группа 16. Команды ввода/вывода 43
Группа 17. Команды прерывания 43
Группа 18. Дополнительные команды 44
6. ДЕМОНСТРАЦИОННЫЕ ПРИМЕРЫ ПРОГРАММ В КОДАХ 46
6.1. ПРОГРАММЫ 46
Группа 1 46
Группа 2 46
Группа 3 47
Группа 4 48
Группа 5 48
Группа 6. 49
Группа 7. Команды вычитания. 50
Группа 8 51
Группа 9 51
Группа 10 52
Группа 11 53
Группа 12 54
Группа 13 55
Группа 14 56
Группа 15 56
7. МОНИТОР 16К 58
7.1. СИСТЕМНЫЙ ВЗГЛЯД НА МОНИТОР 58
Операционная система (ОС) 58
7.2. РАЗЛИЧНЫЕ ЧАСТИ ПРОГРАММЫ МОНИТОРА 61
8. ИСПОЛЬЗОВАНИЕ ПОДПРОГРАММ МОНИТОРА 69
8.1. ВВЕДЕНИЕ 69
8.2. ШЕСТНАДЦАТЕРИЧНЫЙ ВВОД 69
8.3. КОМАНДА BEEP 69
8.4. СОХРАНЕНИЕ И ЗАГРУЗКА 70
8.5. ЭЛЕМЕНТЫ ЦВЕТА 70
Постоянные и переменные. 72
8.6. КОМАНДЫ CLS И SCROLL 72
8.7. ПОДПРОГРАММА ПЕЧАТИ 73
Печать строк 73
Печать чисел 74
Использование строки RANDOMIZE USR 32000 74
8.8. КОМАНДЫ PLOT, DRAW И CIRCLE 75
8.9. POINT, ATTR, SCREEN$ 77
8.10. PI, RND, INKEY$ 78
8.11. BREAK 80
8.12. ЗАКЛЮЧЕНИЕ 80
ПРИЛОЖЕНИЕ I 81
ТАБЛИЦА СИМВОЛОВ 81
УПРАВЛЯЮЩИЕ КОДЫ ZX SPECTRUM 85

VA PRINT

Подписано в печать 18.02.93 г. Формат 60x88 1/16. Зак. 6662. Тираж 25 000 экз. Государственное ордена Октябрьской Революции, ордена Трудового Красного Знамени Московское предприятие "Первая Образцовая типография" Министерства печати и информации Российской федерации. 113054, Москва, Валовая, 28.

© VA PRINT

MOSCOW 93