
УЧЕБНО-МЕТОДИЧЕСКОЕ
ПОСОБИЕ
ДЛЯ ПОЛЬЗОВАТЕЛЕЙ
КОМПЬЮТЕРОВ
ZX-SPECTRUM

УЧЕБНО-МЕТОДИЧЕСКОЕ ПОСОБИЕ ДЛЯ ПОЛЬЗОВАТЕЛЕЙ КОМПЬЮТЕРОВ ZX-SPECTRUM

НАУЧНО-ПРОИЗВОДСТВЕННЫЙ ЦЕНТР "ТИНОПОС"

НОВОПОЛОЦК 1991г

О Г Л А В Л Е Н И Е

ВВЕДЕНИЕ

Персональный компьютер ZX-SPECTRUM - самый популярный домашний компьютер в мире, и хотя масса фирм уже давно выпускает гораздо более мощные машины - позиций своих он не сдает.

История создания компьютера достаточно интересна. Его создатель Клайв Марло Синклер (1940 г.р.) свою первую фирму "Синклер радионикс" зарегистрировал в 1961 году, а первое изделие - микроусилитель - выпустил в 1963 году.

Во всех разработках он ставил перед собой две сверхзадачи - минимальные габариты и минимальная цена. Успех Синклера всегда основывался на том, что он со своим товаром был первым, причем часто ориентировался на рынок, который еще не существовал.

В 1979 г. фирма "Коммодор" выпустила свой первый бытовой компьютер "PET" ценой 700 фунтов. Газета "Файнэшнл Таймс" тогда предсказывала, что цены на персональные компьютеры опустятся ниже 100 ф.ст. не ранее, чем через 5 лет, а Синклер уже через полгода выпустил ZX-80 ценой 97 фунтов.

Резкому снижению цены содействовала идея использования телевизора в качестве дисплея, а бытового магнитофона в качестве внешней памяти.

В первые 8 месяцев было продано 20 тыс. компьютеров, и в марте 1981 г. была выпущена новая модель ZX-81 ценой 69 фунтов. В эти дни американская фирма "Таймекс" купила право на производство всех разработок Синклера, как сделанных, так и тех, которые будут выпущены впредь. Фирма "Митцуи" купила исключительное право на распространение ZX-81 в Японии. Решительным рывком вперед стал договор с британской книготорговой сетью о реализации компьютеров по их торговым каналам. За один год товарооборот фирмы вырос с 4,6 до 30 млн.ф.ст., а Синклер уже готовил новую модель- SPECTRUM (март 1982 г.). Были разработаны две версии - 16K и 48K. Эта машина сильно отличалась от своих предшественников, и ее популярность превзошла все ожидания. ZX-SPECTRUM продавались по 15 тыс. штук в неделю, за 2,5 года фирма SINCLAIR RESEARCH OF CAMBRIDGE изготовила и продала около 500 тыс. штук компьютеров.

Задумывался этот компьютер как учебный для изучения программирования, но фирмы, выпускающие программное обеспечение, быстро поняли, что программирование на уровне команд процессора позволяет получить неплохую динамичную графику, и для этого компьютера стали выпускаться увлекательные видеоигры. Получилась своего рода положительная обратная связь. Чем больше СПЕКТРУМов покупалось населением, тем активнее выпускались для него программы, а чем больше на рынке высококачественных программ для компьютера, тем активнее он покупается. Такой же процесс охватил и фирмы "третьего рынка", выпускающие периферийные устройства и аксессуары для компьютеров. К 1984 году, когда фирмы "Атари", "Коммодор" и "Амстрад" выпустили компьютеры, превосходящие СПЕКТРУМ 48, рынок уже был смещен в пользу СПЕКТРУМа, что продолжает чувствоваться и по сей день, а сам СПЕКТРУМ уже выпускался более чем в 30 странах мира.

В 1984 году Синклер выпустил модель СПЕКТРУМа, отличающуюся усовершенствованной клавиатурой, а в конце 1985 года СПЕКТРУМ-128 ("ДЕРБИ"), имеющий 128K оперативной памяти и 32K ПЗУ. Кроме того, новая модель имела звуковой процессор.

В дальнейшем, в 1986 году Синклер под давлением финансовых трудностей, вызванных с одной стороны - просчетами при разработке новых моделей машин, а с другой - отсутствием у него серьезной деловой хватки, продал все права на производство СПЕКТРУМ - совместимых моделей французской фирме "Амстрад".

Продав все права на производство и реализацию своих изделий, Синклер оставил за собой исследовательскую лабораторию в Кембридже.

Последующие модели СПЕКТРУМ+2 (1986 г.) со встроенным магнитофоном и СПЕКТРУМ+3 (1987 г.) со встроенным дисководом выпускались уже фирмой "Амстрад". Основным их преимуществом является полноценная клавиатура, в то время как встроенные магнитофон и дисковод воспринимаются скорее как "нагрузка", непропорционально увеличивающая цену. Особенно если принять во внимание нестандартный диаметр дискет 3,0 дюйма, малую их емкость (180K) и практическую сложность переноса имеющихся кассетных версий программ на диск, граничащую с нецелесообразностью.

В заключение вступления несколько слов о перспективной разработанной модели, планировавшейся к выпуску в 1987 г. В основу СУПЕР СПЕКТРУМа ("ЛОККИ") был положен процессор Z-80H, который может работать с частотой 7 МГц. При такой скорости удается организовать и обслужить два банка памяти по 64К и экран емкостью более 51К. Он имел разрешающую способность 192*256 с возможностью одновременного воспроизведения 64-х цветов (для каждой точки). Эта машина была программно совместима со СПЕКТРУМом, стоила менее 200 фунтов и была бы серьезным конкурентом для "АМИГИ". Но фирма "Амстрад", пользуясь своими правами, опасаясь конкуренции для своих машин, отказала в разрешении на его производство.

Компьютер ZX-SPECTRUM - конструктивно достаточно несложное устройство, и при наличии необходимых комплектующих изделий и деталей - для тех, кто немного знаком с электроникой - не представляет трудностей его монтаж и отладка в домашних условиях. Простота его воспроизведения и эксплуатации, большие вычислительные возможности СПЕКТРУМа, большой объем наработанного в мире программного обеспечения, особенно игровых программ - по количеству которых ему нет конкурентов и привели к появлению на внутреннем рынке ряда модификаций компьютера СПЕКТРУМ-48, отличающихся от фирменных машин только конструктивным исполнением и непринципиальными изменениями схемотехнических решений.

Настоящая книга имеет свой целью помочь пользователю изучить этот компьютер, научиться программировать на БЕЙСИКе и в машинных кодах для него, использовать все его вычислительные и интеллектуальные возможности. Отдельно подробно рассмотрен интерфейс гибкого диска. В приложении даны практические рекомендации по подключению и работе компьютера совместно с магнитофоном, принтером.

При чтении настоящего руководства старайтесь использовать компьютер. Если у вас возникнет вопрос, что будет, если выполнить те или иные действия с компьютером, то ответ вы получите на экране телевизора, введя эти фразы в компьютер.

Всякий раз, когда в этой книге вы встретите предложение что-нибудь ввести в компьютер и выполнить на нем те или иные действия, подумайте, чем можно заменить эти действия и попробуйте их проделать. Чем больше собственных программ вы напишите, тем лучше будете понимать, как работает компьютер.

Не бойтесь отдать компьютеру не ту команду, нажать не ту клавишу - этим его не испугаешь.

Если все же в результате ваших манипуляций компьютер "зависнет" и перестанет реагировать на нажатие клавиш - нажмите кнопку "СБРОС" - и все придет в нормальное состояние.

1.ОПИСАНИЕ КОМПЬЮТЕРА.

Сам по себе микрокомпьютер ZX SPECTRUM представляет собой черный пластмассовый корпус размерами 233*144*30 мм. На верхней части его расположено 40 клавиш, которые образуют клавиатуру. Сзади расположены: выходной разъем, куда подключается антенный вход телевизора; входной разъем, который соединяется с выходом магнитофона; выходной разъем, который соединяется со входом магнитофона; разъем для подключения устройств ввода/вывода: принтера, микроприводов и др.; разъем для подключения блока питания.

На основной печатной плате (ПП) расположен микропроцессор Z80A и практически все другие логические элементы, входящие в состав компьютера, в том числе и память. На второй ПП расположена клавиатура. Обе платы находятся в одном корпусе и соединены между собой ленточными кабелями.

1.1 ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ КОМПЬЮТЕРА

Обьем ОЗУ .....48 кБайт (49152 Байт)
Обьем ПЗУ .....16 кБайт (16384 Байт)
Количество внешних портов .....48000 (на системной магистрали)
Скорость записи на магнитофон .....1500 Бод (около 180 Байт/сек)
Емкость кассеты магнитофона .....1 мБайт (МК-90)
Скорость вычисления .....950 000 Оп/сек.
Формат экрана .....256*192 точки
Количество цветов 8
Количество градаций яркости 2
Режим мерцания .....есть (8*8 точек)
Цветов бордера (рамки) 8
Таймер .....кварцованный, дискретность 1/50 С

Компьютер построен на высокоскоростном 8-разрядном процессоре Z80 фирмы ZILOG с тактовой частотой 3,5 мГц,что позволило получить производительность на уровне 950 000 операций в секунду. Это приблизительно в 4 раза лучше, чем у компьютеров "Радио-86" и "Микроша", близко к производительности компьютера ДВК-3 и даже немного выше, чем у ДВК-2.

Система команд процессора Z80 содержит все команды процессора, применяемого в машинах "Радио-86РК" и "Микроша", и плюс к ним еще в два раза больше своих.

Контроллер дисплея позволяет получать на экране обычного цветного телевизора сложные графические изображения, состоящие из 256*192 точек при 8 цветах, а также текстовые изображения в формате 32*24 символа.

При использовании черно-белого телевизора 8 цветов представляются на нем как 8 градаций яркости.

Имеется возможность подключения светового пера, манипулятора "мышь" (все модели с разъемом INTERFASE 2) и графического принтера (только модели с системным разъемом).

При включении компьютера немедленно активируется тестовая система, проверяющая исправность компьютера и пытающаяся при обнаружении неисправности обеспечить нормальное его функционирование. Она настолько мощна, что позволяет нормально работать при 3/4 не работающем ОЗУ компьютера и других серьёзных неисправностях, что значительно повышает надежность системы.

Через 1,5 секунды после включения тестовая система завершит свою работу и компьютер будет полностью готов к действию - операционная система будет активирована и готова к исполнению команд (в том числе команд встроенного БЕЙСИКа). При этом не требуется загрузка каких-либо программ откуда-либо извне. Пуск машины в работу не сложнее, чем включение обычного калькулятора.

Упрощенная блок-схема компьютера представлена на рис. 1

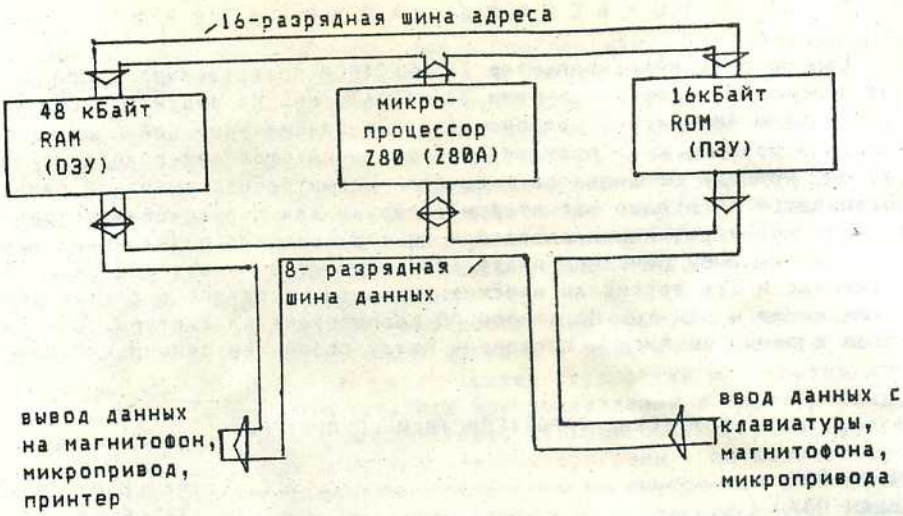


Рис.1

Микропроцессор Z80 (Z80A) - это кремниевый кристалл (ЧИП). Он является самым важным из всех элементов компьютера и предназначен для выполнения программы. Программа для Z80 представляет собой набор кодовых инструкций и согласованных с ними данных.

Назначение выводов микропроцессора Z80 (Z80A) приведено на рис. 2.

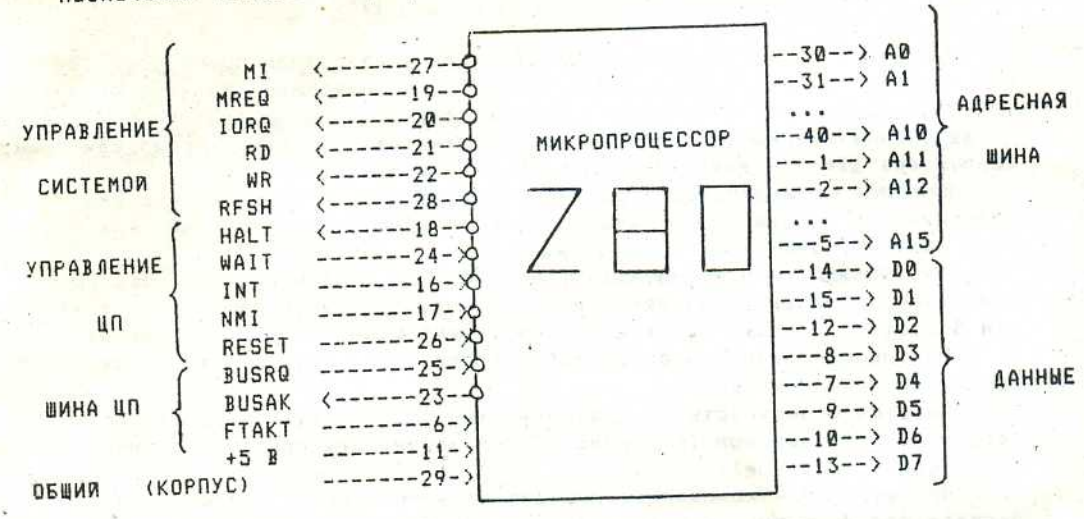


Рис. 2. Назначение выводов микропроцессора Z80 (Z80A) Значок "0" - в обозначении вывода означает, что активный уровень сигнала - низкий (так же как и черта над наименованием сигнала).

1.2 ОПИСАНИЕ СИГНАЛОВ МИКРОПРОЦЕССОРА Z80, Z80A

A0-A15 - (адресная шина). Выходы с тремя устойчивыми состояниями. Активный уровень сигналов - высокий. Адресует ОЗУ или УВВ (до 64К для ОЗУ).

D0-D7 - (шина данных). Входы-выходы с тремя устойчивыми состояниями. Активный уровень - высокий.

MI - (машинный цикл). Выходной активный сигнал - низкий - указывает, что в текущем цикле осуществляется выборка кол (кода операции).

MREQ - (запрос памяти). Выход с тремя устойчивыми состояниями. Активный уровень - низкий. Сигнал указывает, что на адресной шине установлен адрес для операции чтения или записи в память.

IORQ - (запрос ввода-вывода). Выход с тремя устойчивыми состояниями. Активный уровень сигнала - низкий. Сигнал указывает, что младший байт шины адреса содержит адрес УВВ. Кроме того, этот сигнал генерируется после выдачи подтвержения прерывания, тем самым указывая, что вектор прерывания может быть помещен на шину данных.

RD - (чтение из памяти). Выход с тремя устойчивыми состояниями. Активный уровень сигнала - низкий. Сигнал указывает, что процессор (ЦП) готов к чтению данных из памяти,из устройства ВВ. Адресованное УВВ или память использует этот сигнал для стробирования при подаче данных на шины данных ЦП.

WR - (запись в память). Выход с тремя устойчивыми состояниями. Активный уровень сигнала - низкий. Сигнал указывает, что на шине данных содержатся данные, предназначенные для записи в память или вывода на устройство вывода.

RFSH - (восстановление). Выход, активный уровень - низкий. Сигнал указывает, что младшие 7 разрядов шины адреса содержат адрес восстановления для ОЗУ и текущий сигнал MREQ должен использоваться для восстановления динамической памяти.

HALT - (останов). Выход, активный уровень - низкий. Сигнал указывает, что ЦП выполнил команду HALT и ожидает появления либо немаскируемого либо маскируемого прерывания, после которого он продолжит работу. Перед выполнением HALT, ЦП заносит в ОЗУ информацию, которая нужна для восстановления.

WAIT - (ожидание). Вход, активный уровень - низкий. Сигнал указывает микропроцессору, что адресуемые память или устройство не готовы к передаче данных. ЦП ждет пока активен этот сигнал.

INT - (запрос на маскируемое прерывание). Вход, активный уровень - низкий. Запрос будет воспринят ЦП в конце выполнения текущей команды, если триггер разрешения прерывания IFF, управляемый внутренними программными средствами, установлен в определенное состояние.

NMI - (запрос на немаскируемое прерывание). Вход, активный уровень - низкий. Это прерывание имеет более высокий приоритет, чем INT, распознается в конце текущей команды. Сигнал автоматически переводит ЦП к выполнению программы с адреса 0066(HEX).

RESET - (сброс). Вход, активный уровень - низкий. При поступлении сигнала выполняются следующие действия:

а. сброс триггера разрешения прерывания IFF
б. очистка счетчика команд регистров I и R
в. шины адресная и данных - в состоянии с высоким сопротивлением(3-е состояние)
г. для всех управляющих выходных сигналов устанавливаются неактивные уровни.

BUSRQ - (запрос шин). Вход, активный уровень - низкий. Сигнал имеет более высокий приоритет, чем NMI и всегда распознается в конце текущего машинного цикла. Он используется для организации прямого доступа к памяти (ПДП). Переводит в состояние высокого сопротивления все шины и трехстабильные выходы сигналов управления, после чего этими шинами могут управлять другие внешние устройства.

BUSAK - (подтверждение перевода шин в состояние высокого сопротивления). Выход, активный уровень - низкий. Сигнал подается на запрашивающее устройство.

РЕГИСТРЫ ПРОЦЕССОРА Z80 (Z80A)

Набор основн.	альтерн.	название и назначение регистра
A	A'	аккумулятор (8 Бит, со вспомог. 16 Бит)
B	B'	РОН B (8 Бит) - пара BC (16 Бит)
C	C'	РОН C (8 Бит)
D	D'	РОН D (8 Бит) - пара DE (16 Бит)
E	E'	РОН E (8 Бит)
H	H'	РОН H (8 Бит) - пара HL (16 Бит)
L	L'	РОН L (8 Бит)
F	F	регистр флагов (8 Бит)
IX	IX	регистр сегмента (16 Бит)
IY	IY	регистр сегмента Y (16 Бит)
IR	IR	регистр прерывания и регенерации (16 Бит)
SP	SP	указатель стека (16 Бит)
PC	PC	указатель команды (16 Бит)

Особенностью процессора Z80 является наличие у него "альтернативного" набора регистров, обозначающихся так же, как и регистры основного набора, но со штрихом.

Работа с альтернативными регистрами не отличается от работы с использованием основных регистров, и для программиста их подмена может быть незаметна.

Переход на альтернативный набор регистров удобен, например, при переходе с основной на выполнение фоновой программы.

Всего имеется 21 программно-доступный регистр и еще несколько недоступных, которые процессор использует для хранения промежуточных результатов. Большинство регистров процессора или имеет размер 16 Бит, или обьединены в пары по 16 Бит. Какова же разрядность процессора Z80, 8 или 16 Бит? Ответить на этот вопрос очень сложно. Во всяком случае, у процессора имеется и набор команд 8-Битовой арифметики, и набор команд 16-Битовой арифметики. Наличие в составе процессора двойного АЛУ (арифметико-логического устройства) и 16-разрядной схемы сумматора-вычислителя, а также 16-разрядного инкрементора-декрементора позволяет ему оперировать 16-Битовыми данными столь же легко, как и 8-Битовыми. Таким образом, можно сказать, что "внутри" процессор 16-разрядный, но "снаружи" выглядит как 8-разрядный (ведь внешняя шина данных имеет 8 разрядов).

Именно эта особенность процессора Z80 и обусловила наличие у него огромного количества команд (около 680 разновидностей) и регистров (21). Но этого создателям процессора Z80 показалось мало. В дополнение к 16 и 8-Битовым операциям они ввели еще и поразрядные (1-Битовые) операции, что на порядок упростило и, приблизительно, в 2 раза ускорило работу с внешними устройствами (например, с дисководом). Это позволяет в целом ряде случаев обойтись без соответствующих схем контроллеров внешних устройств.

Все эти возможности имеет только процессор Z80, и реализует их только соответствующий ассемблер. Одним из наиболее мощных ассемблеров является программа макроассемблер GENS -4/5 из пакета DEVPAC -4 (кроме макроассемблера GENS -4/5 в пакет входит отладчик-дизассемблер MONS -4). Приставка макро перед названием ассемблера означает, что кроме нормальных команд микропроцессора Z80 вы можете сами создавать и использовать макрокоманды. Например, если вам захочется иметь команду умножения MUL (ее нет в системе команд Z80), ее можно заменить программным эквивалентом и использовать так же свободно, как если бы она всегда была в системе команд.

Загружают пакет командой LOAD ""CODE ..., где ... - адрес, начиная с которого будет находиться программа. Затем программы запускаются командой RANDOMIZE USR ..., где ... - тот же адрес, что и в команде LOAD .

16K ROM - ПОСТОЯННАЯ ПАМЯТЬ (ПЗУ)

Представляет собой один ЧИП объемом 128 кБит (или два ЧИПа по 64 кБит) - или это 16 кБайт памяти, содержащие программу монитора в машинных кодах для микропроцессора Z80.

Программа "монитор 16К" разбита на 3 части:

7 кБайт - операционная система
8 кБайт - интерпретатор БЕЙСИКа
1 кБайт - генератор знаков

48 кБайт RAM - ОПЕРАТИВНАЯ ПАМЯТЬ (ОЗУ)

Вся память компьютера разбита на байты, каждый из которых может содержать число от 0 до 255 (в двоичном коде от 00.000.000 до 11.111.111). Каждый байт может быть записан в память по определенному адресу от 16384 до 65535. Сам адрес может быть записан в память как два байта. Для организации работы операционной системы в оперативной памяти выделена системная область. Остальная память доступна для программ. Распределение памяти приведено на рис. 3

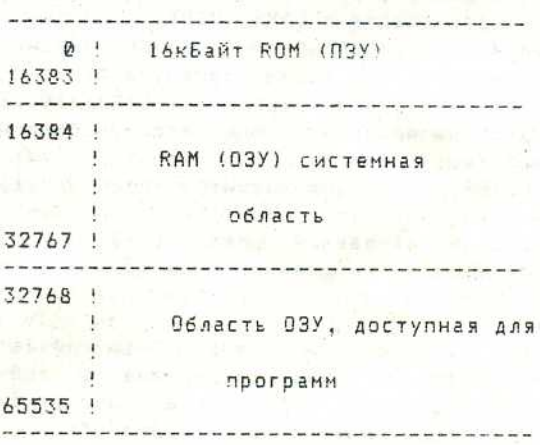


Рис. 3

В большинстве моделей СПЕКТРУМов ОЗУ построено на 8 микросхемах типа 565 РУ5Б(В,Г), имеющих организацию 64К*1 или на 16 микросхемах типа 565РУ5Д1(Д2) - с организацией 32К*1. Память динамического типа.

КЛАВИАТУРА

Клавиатура СПЕКТРУМа имеет 40 клавиш. Каждая клавиша клавиатуры имеет многофункциональное назначение (до 6-7 функций в зависимости от того, в сочетании с какими клавишами она нажимается) и позволяет вводить как отдельные символы, так и целые слова. Более подробно работа с клавиатурой будет рассмотрена в следующей главе.

ЭКРАН ТЕЛЕВИЗОРА

Экран телевизора содержит 24 строки по 32 позиции в каждой и делится на две части. Верхняя часть в 22 строки отображает листинг или вывод из программы.

Нижняя часть используется для ввода команд, строк программы и входных данных, а также для отображения сообщений.

Доступны цвета: черный, голубой, красный, синий, пурпурный (фиолетовый), зеленый, желтый и белый. Края экрана (бордер) могут быть установлены пользователем в необходимый ему цвет. Более подробно описание работы с экраном приведено в разделе 8.

МАГНИТОФОН

Вопросы работы компьютера СПЕКТРУМ с магнитофоном (подключение магнитофона, загрузка компьютера, копирование записей и т. д.) подробно рассмотрены в приложении.

2. К Л А В И А Т У Р А

В СПЕКТРУМе клавиши содержат не только одиночные символы (буквы, цифры и т. д.), но также составные символы (ключевые слова, названия функций и т.д.)

Для того, чтобы реализовать все функции и команды, некоторые клавиши клавиатуры имеют по 5 и более значений, получаемых либо путем выбора соответствующего регистра (т.е. получаемых путем нажатия клавиши CAPS SHIFT или SYMBOL SHIFT одновременно с какой-либо необходимой клавишей), либо путем перевода компьютера в один из возможных режимов работы.

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

Рассмотрим назначение некоторых наиболее характерных клавиш.

ENTER - нажатие этой клавиши обычно завершает ввод в компьютер чего-либо (команды, данных, программной строки). На компьютерах других систем эта клавиша может иметь другое наименование - RETURN, ВК (возврат каретки) и др.

CAPS SHIFT - переключение регистра на печать прописными буквами (аналогично пишущей машинке).

SYMBOL SHIFT - переключение регистра для печати символов (+, -, /, и т.д.), а также некоторых ключевых слов (THEN, TO, OR, AND, STEP и д.р.).

SPASE - пробел. Эта же клавиша выполняет BREAK (прерывание исполнения программы), если ее нажать вместе с CAPS SHIFT.

Остальные клавиши имеют буквенное или цифровое обозначение.

При подключении компьютера на экране должно появиться исходное сообщение "1982 (1988,1989 - в зависимости от версии ОС) SINCLAIR RESEARCH LTD". Это означает, что первичные проверки в компьютере прошли нормально, он исправен и готов к работе.

Нажмите клавишу ENTER, на экране появится черный квадрат с буквенным мерцающим обозначением на нем. Это курсор - место на экране, в котором будет помещаться очередной набираемый символ, а также указывает, в каком режиме находится клавиатура. Этих режимов - четыре, и им соответствуют пять разных курсоров.

1. КОМАНДНЫЙ РЕЖИМ. Курсор - "К". Он означает, что сейчас при нажатии клавиши будет введена команда, которая закреплена за этой клавишей, или цифра. Компьютер сам понимает, что строка может начинаться только либо с номера строки, либо с команды, поэтому дает курсор "К".

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

После того, как команда набрана, и дальше должны пойти ее параметры, режим автоматически переключится на литерный (курсор "L"). Если вы введете ":" (двоеточие), то курсор опять переключится на "К", т. к. двоеточие является разделителем между несколькими командами, если они записываются в одной строке.

2. ЛИТЕРНЫЙ РЕЖИМ. Курсоры - L, C. Курсор "L" соответствует печати строчными буквами, а курсор "С"- прописными. Переключиться на курсор "С" можно командой CAPS LOCK. Это выполняется одновременно нажатием CAPS SHIFT и цифры "2". В этом же режиме набираются символы и служебные слова, связанные с клавишей "SYMBOL SHIFT".

И в [К] и в [L] режимах одновременное нажатие клавиши SYMBOL SHIFT и какой-либо клавиши воспринимается как вспомогательный символ, изображенный на клавише, а в случае CAPS SHIFT с цифровой клавишей - как управляющая функция, написанная на цифровой клавише.

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

3. РАСШИРЕННЫЙ КОМАНДНЫЙ РЕЖИМ. Курсор "E". В этом режиме набираются команды, записанные над клавишами или под клавишами. Переход в режим "E" выполняется одновременным нажатием CAPS SHIFT и SYMBOL SHIFT. Он действует только на одно нажатие. Если в режиме "E" нажать какую-либо клавишу, то появится слово, записанное над клавишей, а если нажать эту клавишу совместно с CAPS SHIFT - то слово, записанное под клавишей. Здесь имеется исключение для цифровых клавиш (верхний ряд). Чтобы набрать слово, записанное под клавишей, надо в режиме "E" нажать не CAPS SHIFT, а SYMBOL SHIFT совместно с клавишей. Такая уникальная способность СПЕКТРУМа вводить операторы и функции не по буквам, а одним нажатием клавиши, называется токенизированной формой записи ключевых слов. Сначала это выглядит несколько сложновато, но опыт приходит быстро, а с ним и удобство работы.

4. ГРАФИЧЕСКИЙ РЕЖИМ. Курсор - "G".В этом режиме набираются символы блочной графики, расположенные на цифровых клавишах, а также символы графики пользователя.

Переход в графический режим выполняется одновременным нажатием клавиш CAPS SHIFT и цифры "9". Если вы в этом режиме нажмете, например, клавишу "5", то на экране появится квадрат, левая половина которого черная, а правая - белая. Нажатие клавиши "5" совместно с CAPS SHIFT изобразит тот же квадрат в инвертированном виде, т. е. левая половина будет белая, а правая - черная.

Графический режим сохраняется до тех пор, пока не будет нажата клавиша CAPS SHIFT одна или совместно с "9".

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

П Р И М Е Р.

Если набрать что-то произвольно на экране, то скорее всего это будет неправильная строка и при нажатии ENTER на экране появится знак "?". Чтобы начать работу, надо эту строку стереть. Это выполняется командой DELETE (одновременное нажатие CAPS SHIFT и цифры 0). Когда все символы будут уничтожены, курсор встанет на исходную позицию и примет вид "K", т. е. компьютер ждет от вас команду.

Нажмите клавишу "R". На экране появится команда RUN. Курсор изменится на "L". Еще раз нажмите ту же клавишу - появится буква r.

Нажатие этой клавиши совместно с CAPS SHIFT даст прописную букву R, а совместно с SYMBOL SHIFT - знак "<".

Перейдите в расширенный командный режим - CAPS SHIFT + SYMBOL SHIFT одновременно. Появится курсор "E". Теперь нажатие той же клавиши даст оператор INT, снова войдите в режим "E" и нажмите клавишу совместно с CAPS SHIFT. Получите команду VERIFY. Если какую-либо клавишу задержать в нажатом положении больше 2-3 секунд, то буква будет повторена многократно со скоростью около 6 символов в секунду. Эта удобная функция называется "автоповтор" (кстати, он действует и при стирании).

Ввод с клавиатуры осуществляется в нижнюю половину экрана, каждый символ (или группа символов для ключевых слов) появляется перед курсором. Сам курсор может перемещаться по экрану клавишами:

влево - CAPS SHIFT и "5"
вправо - CAPS SHIFT и "8" и т. д.

Символ перед курсором может быть удален командой DELETE (CAPS SHIFT и 0) Примечание: целая строка может быть удалена вводом EDIT (CAPS SHIFT и"1") и последующим нажатием клавиши ENTER.

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

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

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

Те, кто работает с компьютерами "СПЕКТРУМ+", "+128", "+2", "+3" могут упростить свою работу благодаря наличию дополнительных клавиш, которые выполняют ряд функций одним нажатием.

DELETE - стирание символа
GRAPH - переход в графический режим
EXTENDED MODE - переход в расширенный командный режим
EDIT - выполнение редактирования строки
CAPS LOCK - переключение регистра на печать прописными буквами
INV. VIDEO - включение инверсного режима (печать белым по черному)
TRUE VIDEO - возвращение из инверсного режима

Кроме того, там имеются дополнительные клавиши для набора знаков препинания: точка, запятая, точка с запятой, кавычки, а также четыре клавиши для управления перемещением курсора при редактировании.

В таблице 1 приведено расписание клавиатуры компьютера СПЕКТРУМ ,т. е. значение клавиш в зависимости от режима работы компьютера.

ТАБЛИЦА 1

КЛАВИША	КУРСОР					
	K		L			E
			CAPS SHIFT	SYMBOL SHIFT		CAPS SHIFT
A	NEW	a	A	STOP	READ	G //
B	BORDER	b	B	*	BIN	BRISHT
C	CONTINUE	c	C	?	LPRINT	PAPER
D	DIM	d	D	STEP	DATA	/ \
E	REM	e	E	>=	TAN	ATN
F	FOR	f	F	TO	SGN	{
G	GO TO	g	G	THEN	ABS	}
H	GO SUB	h	H	^	SQR	CIRCLE
I	INPUT	i	I	AT	CODE	IN
J	LOAD	j	J	-	VAL	VAL$
K	LIST	k	K	+	LEN	SCREEN$
L	LET	l	L	=	USR	ATTR
M	PAUSE	m	M	.	PI	INVERSE
N	NEXT	n	N	,	INKEY$	OVER
O	POKE	o	O	!	PEEK	OUT
P	PRINT	p	P	"	TAB	©
Q	PLOT	q	Q	<=	SIN	ASN
R	RUN	r	R	<	INT	VERIFY
S	SAVE	s	S	NOT	RESTORE	I
T	RANDOMIZE	t	T	>	RND	MERGE
U	IF	u	U	OR	CHR$	[
V	CLS	v	V	/	LLIST	FLASH
W	DRAW	w	W	<>	COS	ACS
X	CLEAR	x	X	;	EXP	INK
Y	RETURN	y	Y	AND	STR$	]
Z	COPY	z	Z	:	LN	BEEP
0	0	0	DELETE	-	BLACK	FORMAT *
			CAPS SHIFT	SYMBOL SHIFT		SYMB SHIFT
1	1	1	EDIT	!		DEF FN *
2	2	2	CAPS LOSK	@		FN *
3	3	3	TRUE VIDEO	#		LINE *
4	4	4	INV. VIDEO	$		OPEN $ *
5	5	5	курсор влево	%		CLOSE $ *
6	6	6	курсор вниз	&		MOVE *
7	7	7	курсор вверх	'		ERASE *
8	8	8	курсор вправо	(		POINT *
9	9	9	GRAPHICS	)		CAT *

Примечание: далее в тексте в отдельных разделах вместо символа "$" применен символ "¤".

3. ОРГАНИЗАЦИЯ ПАМЯТИ КОМПЬЮТЕРА

Для обеспечения работы операционной системы компьютера в его памяти выделено несколько областей. Эти области отделены друг от друга некоторыми границами, которые могут быть фиксированы (постоянно), а могут изменяться в зависимости от конкретных требований.

Фиксированные границы обозначены числом, а переменные - именем системной переменной, в которой они хранятся.

КАРТА ПАМЯТИ КОМПЬЮТЕРА

Адреса с 0 до 16383 (16К) - область ПЗУ

В ПЗУ хранятся:

7кБайт - операционная система (ОС)
8кБайт - интерпретатор БЕЙСИКа
1кБайт - генератор знаков

Операционная система (ОС) - это программная надстройка над компьютером. ОС СПЕКТРУМа относится к типу "твердотельных" ОС, т. к. находится в ПЗУ.

Основная функция ОС - операции над файлами, например, загрузка их в память, запуск, выгрузка, сравнение, компановка.

Всего имеется 5 типов файлов:

BYTE - программа в машинных кодах
PROGRAMM - программа на БЕЙСИКе
SCREEN$ - копия экрана
DATA - значения числовых массивов
DATA$ - значение символьных переменных или массивов

Кроме того, некоторые программы создают свои типы файлов. Однако эти типы файлов не являются стандартными и не обрабатываются ОС.

АДРЕСА С 16384 ДО 22527 (6К), (4000-57FF- в шестнадцатиричном виде) - экранная область - обеспечивает высокое разрешение экрана телевизионного приемника (ТП).

Доступна для операторов PEEK и POKE.

Каждая позиция экрана представима матрицей 8*8 точек (1 байт на каждый ряд из 8 точек). Однако эти 8 байтов хранятся в памяти не вместе. Полный экран представляет собой 24 строки по 32 символа в каждой. Каждая строка прописывается 8-ю строками развертки экрана ТП. Итого для воспроизведения (записи) изображения одного экрана выполняется 24*8=192 сканирования и в памяти рядом хранятся байты одноименных рядов матриц соседних позиций экрана.

Высокое разрешение экрана ТП достигается за счет того, что в отведенных под экранную область 6К памяти можно закодировать изображение любой из 49152 точек экрана ТП (матрица 192*256 точек), что подтверждается расчетом:
1 символ кодируется 8*8=64 точками
количество символов на 1 экран 32*24=768
количество адресуемых точек 768*64=49152

Экран ТП разбит на три части по 2К. Строкам с номерами 0-7 соответствуют адреса 16384-18431 (4000-47FF - шестнадцатиричные). Строкам с 8 по15 соответствуют адреса 18432-20479 (4800-48FF - шестнадцатиричные). Строкам с 16 по 23 соответствуют адреса 20480-22527 (4900-57FF - шестнадцатиричные).

Каждые из этих третей по 2К состоят из 8 массивов по 1/4 кБайта. Каждый массив из 1/4 кБайта (256 байтов) хранит коды символов одной строки экрана. Счет линий начинается сверху экрана.

АДРЕСА С 22528 ПО 23295 (5800-5AFF - шестнадцатиричные) - это область атрибутов - по 1 байту на каждую позицию символа, где хранятся признаки цвета, яркости, мигания. Экран имеет 768 символов, каждый из которых имеет один из восьми цветов "чернил", восьми цветов "бумаги" (фон), признак мигания и признак яркости - повышенной и нормальной.

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

Кодировка байтов атрибутов следующая:

биты 0-2 - код цвета "чернил"
биты 3-5 - код цвета "бумаги" (фон)
бит 6 - признак яркости (0- обычная, 1- повышенная)
бит 7 - признак мигания (0- постоянное свечение, 1 - мигание).

Коды цветов следующие:

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

АДРЕСА С 23296 ПО 23551 - область буфера ZX-принтера - содержит 256 байтов - по 32 байта на каждую печатаемую строку. При отсутствии такого принтера часто используется для хранения небольших процедур в машинных кодах. Здесь они не повреждаются БЕЙСИКом, и, кроме того, размещение их в нижней половине памяти компьютера повышает их быстродействие.

В компьютерах 128К в этой области хранятся дополнительные системные переменные для поддержания повышенных возможностей этих машин, и потому туда ничего нельзя записывать.

АДРЕСА С 23552 ПО 23733 - 182 байта (5000-5CB5 - шестнадцатиричные) область системных переменных.

стр 48

Байты памяти с 23552 до 23733 предназначены для специального использования. В них размещаются так называемые "системные переменные". Не надо путать их имена с именами переменных в программе. Компьютер не распознает ссылки к этим переменным из БЕЙСИК-программы по их именам. Имена используются только для мнемонического обозначения этих переменных в этом описании.

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

X - переменная не должна изменяться, так как это может нарушить работу системы.

N - изменение переменной не приводит к длительному эффекту.

Число - число байтов в переменной (для двухбайтовых переменных младший байт первый).

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

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

Для просмотра нового значения можно использовать оператор:

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

зн.	АДРЕС	ИМЯ	СОДЕРЖАНИЕ
1	2	3	4
N8	23552	KSTATE	используется при чтении с клавиатуры
N1	23560	LAST K	запоминается вновь нажатая клавиша
1	23561	REPDEL	время,в 50-ых долях секунды, в течение которого клавиша должна быть зафиксирована в нажатом состоянии. Начальное значение 35, но может быть изменено
1	23562	REPPER	задержка, в 50-ых долях секунды, между последовательными опросами клавиш. Начальное значение 5.
N2	23563	DEFADD	адрес аргументов функций пользователя если они используются, иначе 0.
N1	23565	K DATA	второй байт управления цветом с клавиатуры.
N2	23566	TVDATA	байты цвета, AT, TAB управления TV
X38	23568	STRMS	адреса подключенных каналов.
2	23606	CHARS	адрес символьного набора - 256, обычно этот набор находится в ПЗУ, но может быть размещаем и в ОЗУ с указанием в CHARS адреса размещения.
1	23608	RASP	продолжительность звукового сигнала
1	23609	PIP	длительность задержки, устраняющей дребезг клавиатуры.
1	23610	ERR NR	код сообщения -1. Начальное значение 255 (для '-1'), т.е. PEEK 23610 = 255.
X1	23611	FLAGS	управляющие флаги БЕЙСИКА.

1	2	3	4
X1	23612	TV FLAG	флаг телевизора.
X2	23613	ERR SP	адрес в аппаратном стеке, используемый как адрес возврата при ошибке.
N2	23615	LIST SP	адрес возврата из автоматического листинга.
N1	23617	MODE	режим. Спецификация [K],[L],[C],[E] или [G] курсора.
2	23618	NEW PPC	номер строки, на которую должен быть сделан переход.
2	23621	PPC	номер строки, оператор в которой выполняется
1	23623	SUB PPC	порядковый номер выполняющегося оператора в строке.
1	23624	DORDCR	цвет рамки экрана, содержит атрибуты.
2	23625	E PPC	количество текущих строк (с курсором)
X2	23627	VARS	адреса переменных
N2	23629	DEST	адрес переменной в задании.
X2	23631	CHANS	адрес канала данных
X2	23633	CURCHL	адрес данных для ввода-вывода
X2	23635	PROG	адрес БЕЙСИК-программы.
X2	23637	NXTLIN	адрес следующей строки в программе
X2	23639	DATADD	адрес терминатора последнего символа в DATA.
X2	23641	E LINE	адрес выведенной команды.
2	23643	K CUR	адрес курсора
X2	23645	CH ADD	адрес следующего интерпретируемого символа: символ аргумента в PEEK, NEWLINE или POKE операторов
2	23647	X PRT	адрес символа следующего за маркером [?].
X2	23649	WORK SP	адрес временной рабочей области
X2	23651	STK BOT	адрес "ДНА" программируемого стека
X2	23653	STK END	адрес начала резервной области памяти
N1	23655	BREG	B-регистр калькулятора
N2	23566	MEM	адрес области, используемой как память калькулятора (обычно MEMBOT, но не всегда)
1	23658	FLAG52	старшие флаги

1	2	3	4
X1	23659	DF SZ	число строк (включая и одну чистую) в нижней части экрана
2	23660	S TOP	количество верхних строк программы в автоматическом листинге
2	23662	OLDPPC	номер строки, на которую указывает CONTINUE
1	23664	OSPPC	номер оператора в строке, на которую указывает CONTINUE
N1	23665	FLAGX	переменные флаги
N2	23666	STR LEN	размер расстояний между строками
N2	23668	T ADDR	адрес следующего символа в синтаксической таблице
2	23670	SEED	начальное значение для RND, изменяется функцией RANDOMIZE
3	23672	FRAMES	счетчик кадров - приращение через каждые 20 mS (см. главу 18)
2	23675	UDG	адрес первого определяемого пользователем символа
1 1	23677 23678	COORDS	X-координата точки графопостроителя Y-координата точки графопостроителя
1	23679	POSN	33-позиционное число для позиционирования принтера
1	23680	PR CC	младший байт адреса позиции для LPRINT для печати
1	23681		не используется
2	23682	ECHO E	33-позиционное и 24-строковое числа (в нижней половине) конца входного буфера
2	23684	DF CC	адрес PRINT-позиции в области экрана
2	23686	DF CCL	подобно DF CC для нижней части экрана
X1 X1	23688 23689	S POSN	33-позиционное число для PRINT позиции 23- строковое число для PRINT позиции
X2	23690	S POSNL	подобно S POSN для нижней части
1	23692	SCR CT	счетчик сверток: всегда на 1 больше числа сверток, которые должны быть проведены перед остановом со сверткой. Если вы установите это число больше, чем на 1 (скажем 255), то экран будет сворачиваться без запроса к вам
1	23693	ATTR P	сплошные цвета

1	2	3	4
1	23694	MASK P	используется для высвечивания цветов. Бит, установленный в 1, показывает, что биты атрибутов берутся не из ATTR P, а из того, что указано на экране
N1	23695	ATTR T	временный указатель цветов
N1	23696	MASK T	временный MASK P
1	23697	P FLAG	старшие флаги
N30	23698	MEMBOT	область памяти для калькулятора. используется для записи чисел, которые не могут быть размещены в программируемом стеке калькулятора
2	23728		не используется
2	23730	RAMTOP	адрес последнего байта области БЕЙСИК-системы
2	23732	P-RAMT	адрес последнего байта физического ОЗУ

Следующая программа выдаст вам первые 22 байта области системных переменных:

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

Теперь замените строку 20 на

```
20 PRINT PEEK(23755+N)
```

и вы дополнительно получите дамп самой программы.

АДРЕСА С 23734 ДО CHANS-1 - карты памяти микродрайва. При работе с микродрайвом здесь хранится информация, например, об испорченных секторах на ленте и т. п. Если микродрайва нет, то эта область не организуется.

ОБЛАСТЬ ИНФОРМАЦИОННЫХ КАНАЛОВ.

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

В стандартном СПЕКТРУМе без подсоединенных микроприводов имеются 4 основных канала:

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

Информационные каналы состоят, для каждого канала, из 5 байтов данных. Эти байты содержат:

- адрес программы ввода - 2 байта
- адрес программы вывода - 2 байта
- один символ имени файла - 1 байт

В стандартном СПЕКТРУМе, с учетом четырех каналов и маркера конца, область занимает память с 23734 по 23754 - 21 байт.

ОБЛАСТЬ БЕЙСИК-ПРОГРАММЫ (БП).

Область содержит последовательные строки текста программы. Размер области определяется количеством строк.

Начало программы задается в переменной PROG (адреса 23635-23636). Отметим, что в стандартном СПЕКТРУМе переменная PROG указывает на адрес 23755 и так до тех пор, пока не будет подключен микродрайв или использован дополнительный канал.

Формат строк БП приведен в разделе 5.

ОБЛАСТЬ ПЕРЕМЕННЫХ - это область, где хранятся все переменные, с которыми оперирует БП.

Начало области задается переменной VARS (адреса 23627 и 23628, или шестнадцатиричные 5C4B и 5C4C). Начало этой области остается постоянным во время работы БП. Однако с появлением новых переменных ее длина может меняться. Граница области задается маркером конца (код 128 или шестнадцатиричный 80).

Следующая программа содержит только переменные оператора FOR NEXT

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

ОБЛАСТЬ БУФЕРА РЕДАКТОРА - сюда помещается строка программы, которую вводят или редактируют.

Начало области задается переменной E-LINE (адрес 23641 и 23642, или, 5C59 и 5C5A шестнадцатиричных).

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 БЕЙСИК на нижнюю часть дисплея. Прежде всего область редактирования расширяется до требуемого размера с тем, чтобы разместилась эта строка БЕЙСИКа. Затем эта строка копируется из области программы на 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.

РАБОЧАЯ ОБЛАСТЬ (буфер инструкции INPUT)

Эта область памяти 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 (шестнадцатиричных 5061 и 5062). Когда же требуется дополнительное место в рабочей области памяти, то эта область памяти расширяется. После использования рабочая область освобождается. Это значит, что она сводится к нулю, чтобы избежать использования большего количества адресов, чем это необходимо.

СТЕК ВЫЧИСЛИТЕЛЯ (калькулятора)

Эта очень важная область памяти начинается с адреса, определенного системной переменной STK BOT, которая сама находится в адресах 23651 и 23652 (шест. 5063-5064). Область распространяется до адреса, заданного в системной переменной STK END (адреса 23653 и 23654, шест. 5065 и 5066). Стек вычислителя может содержать числа с плавающей точкой, целые числа, а при использовании строк - он содержит пять байтов, определяющих параметры строки.

Стек обслуживается по правилу: "первым пришел - последним ушел". Можно рассматривать величину, находящуюся вверху стека, если она существует, как последнюю "пришедшую величину".

РЕЗЕРВНАЯ ПАМЯТЬ. Область памяти между стеком вычислителя и стеком машины представляет собой количество памяти, которым располагает пользователь. Например, в 16К-СПЕКТРУМе 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.

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

GOSUB - СТЕК (стек возвратов). Здесь хранится номер строки, к которой надо вернуться после выполнения подпрограммы по RETURN.

Стек расширяется в памяти вниз и каждая команда GOSUB требует три адреса. Самый старший адрес содержит номер операнда в пределах строки БЕЙСИКа, к которой должен быть осуществлен возврат. Второй адрес содержит младшую часть номера строки, определяющей цикл, а третий адрес содержит ее старшую часть.

Демонстрация следующей программы показывает стек GOSUB, который используется для сохранения чисел строки при организации трех гнездовых программ.

Демонстрационная программа GOSUB - стека

```
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 GOSUB - стека всегда содержат величины 0 и 62 (шестнадцатиричный 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 GOSUB.

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

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 в памяти всегда адресуется системной переменной P-RAMT, расположенной в адресах 23732 и 23733 (шестн. 5CB4-5CB5).

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

Рассматриваемая ниже программа выводит содержание первых 21 байтов с их адресами:

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

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

POKE ADDRESS,NEW CONTENTS

где "ADDRESS" "NEW CONTENS" - числовые выражения

Пример:

```
POKE 31000, 57
```

"NEW CONTENS" может принимать значения от -255 до 255.

4. ЯЗЫК ПРОГРАММИРОВАНИЯ БЕЙСИК

Язык Бейсик-диалоговый алгоритмический язык высокого уровня. Характерной особенностью его является непосредственное взаимодействие пользователя с ЭВМ, когда он сразу же получает ответы на свои вопросы, посланные в ЭВМ.

Язык Бейсик близок к популярному алгоритмическому языку ФОРТРАН.

В языке Бейсик предусмотрены режимы работы программный и непосредственный.

Для работы в программном режиме необходимо, чтобы каждый оператор имел номер (метку). Роль метки выполняют десятичные целые числа от 1 до 9999. Последовательность пронумерованных операций составляет программу.

При работе в непосредственном режиме оператор набирается без номера и немедленно выполняется. Этот режим применяют при простых вычислениях и отладке програми.

Следует различать понятия "язык Бейсик" и "интерпретирующая Бейсик-система". Интерпретирующая Бейсик-система преобразует программу, написанную на алгоритмическом языке Бейсик, во внутреннее представление конкретной ЭВМ и обеспечивает ее выполнение в режиме интерпретации. Как правило, Бейсик-системы ориентированы на определенный тип ЭВМ и могут отличаться одна от другой по своим возможностям в разных ЭВМ.

Ниже мы рассмотрим один из вариантов языка и системы Бейсик, ориентированный на семейство компьютеров, построенных на базе микропроцессоров Z-80 (семейство ZX-СПЕКТРУМ и СПЕКТРУМ-совместимых моделей).

КОНСТАНТЫ.

В языке Бейсик используются целые и вещественные константы. Константы выражаются целыми и десятичными числами. Запись числа осуществляется в форматах с фиксированной и плавающей запятой. Все числа могут иметь точность 9 или 10 знаков. Наибольшее число 10**38, а наименьшее положительное число 4*10**(-39). Числа имеют внутреннее представление, как числа с плавающей (двоичной) точкой, с выделением одного байта на показатель степени E в интервале от 1 до 255, и четырех байтов на мантиссу M в интервале от 0,5 до 1. Это представляется числом M*2**E(-128).

Поскольку 1/2<=M<1, старший значащий бит мантиссы всегда 1. Следовательно, мы можем заменять его на бит, обозначающий знак: 0 - для положительного числа и 1 - для отрицательного.

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

ПЕРЕМЕННЫЕ

Переменная - это величина, значение которой может меняться в процессе выполнения программы. Числовые переменные имеют имя произвольной длины, начинающееся с буквы и продолжающееся буквами или цифрами. Пробелы и символы управления цветом игнорируются и все буквы преобразуются к минимально-упакованному виду.

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

МАССИВЫ

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

Строки символов более гибкие в своей длине. Имя строковой переменной в отличие от простой переменной заканчивается символом $ (доллар).

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

Все строки в массивах имеют фиксированную длину, которая определяется числом, задающим последнюю размерность в операторе DIM. Начальный индекс 1.

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

а) пустым;

б) числовым выражением;

в) некоторое "строковое выражение" TO другое "строковое выражение" и использоваться в:

 *) строковых выражениях (сечениях);

**) строковых массивах переменных
(индекс 1, индекс 2, ... , индекс N, сечение)
или, что то же самое
(индекс 1, индекс 2, ... , сечение).

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

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

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

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

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

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

ФУНКЦИИ

ИМЯ ФУНКЦИИ	ТИП АРГУМЕНТА	ДЕЙСТВИЕ (ВОЗВРАЩАЕМОЕ ЗНАЧЕНИЕ)
1	2	3
ABS	число	Абсолютное значение
ACS	число	Арккосинус в радианах. Выдает сообщение об ошибке A если X не лежит в интервале от -1 до 1
AND	логическая операция. Правый операнд всегда число. Слева может быть: -число, тогда---->	A AND B = / A , если B<>0 \ B , если B=0
	-строка, тогда---->	A$ AND B = / A$, если B<>0 \ " ", если B=0
ASN	число	Арксинус в радианах. Выдает сообщение A , если X не лежит в интервале от -1 до 1
ATN	число	Арктангенс в радианах.
ATTR	два числовых аргумента X и Y заключ. в скобки	Число,двоичный код которого представляет собой атрибуты Y-вой позиции X-вой строки экрана. Бит 7 (старший) равен 1 для мерцающего поля, и 0 для немерцающего. Биты с 5 по 3 - цвет фона. Биты с 2 по 1 - цвет закрашивания. Выдает сообщение B если 0<=X<=23 и 0<=Y<=31
BIN		Это не обычная функция. За BIN записывается последовательность нулей и единиц, представляющая собой двоичное представление числа, которое записывается в память
CHR$	число	Символ, чей код представим числом X, округленным к ближайшему целому.
CODE	строка символов	Код первого символа в строке X (или 0, если X = пустая строка).
COS	число в радианах	косинус X
EXP	число	E в степени X.
FN		FN с последующим именем, определенной пользователем функции (см. DEF). Аргументы должны заключаться в скобки. Даже если нет аргументов, скобки все равно должны записываться.
IN	число	Осуществляется ввод на уровне микропроцессора из порта X (0<=X<=FFFFH). Загружается пара регистров BC и выполняется команда Ассемблера IN A(C).

1	2	3
INKEY$	нет	Чтение с клавиатуры. Возвращает символ, введенный с клавиатуры (в режиме [L] или [C]), если было действительное нажатие клавиши, или пустую строку в противном случае.
INT	число	Округление к ближайшему меньшему целому.
LEN	строка символов	Длина строки.
LN	число	Натуральный логарифм. Выдает сообщение A, если X<=0
NOT	число	0, если X<>0 и 1, если X=0. Операция имеет 4-ый приоритет.
OR	логическая операция. Оба операнда - числа.	A OR B = / 1 , если B<>0 \ A , если B=0 (операция имеет второй приоритет).
PEEK	число	Значение байта в памяти по адресу X, округленному к ближайшему целому.
PI	нет	Число 'пи' (3.14159265...)
POINT	два числовых аргумента X и Y заключ. в скобки	1, если точка экрана с координатами (X,Y) закрашена. 0, если точка имеет цвет фона. Выдает сообщение B, если не выполняется условия 0<=X<=255 и 0<=Y<=175.
RND	нет	Очередное псевдослучайное число из последовательности получаемой возведением в 75-ую степень модуля числа 65537, вычитанием 1 и делением на 65536. Число лежит в интервале 0<=Y<=1.
SCREEN$	два числовых аргумента X и Y заключ. в скобки	Символ (обычный или инверсный), который появляется на экране в строке X, позиции Y. Дает пустую строку, если символ не опознан.
SGN	число	-1 , если X<0 0 , если X=0 1 , если X>0
SIN	число в радианах	Синус X
SQR	число	Корень квадратный. Выдает сообщение A, если X<0.
STR$	число	Строка символов, которая должна быть отображена, если X выводится.
USR	число	Вызывает программу в машинных кодах, начальный адрес которой X. При возврате результатом будет содержимое регистровой пары BC.
USR	строка символов	Адрес группы байтов, задающих определенный пользователем символ для закрепления его за X.
VAL	строка символов	Вычисление X как числового выражения. Выдает сообщение C, если X содержит синтаксические ошибки или дает строковое (не числовое) значение. Возможны и другие ошибки.

1	2	3
VAL$	строка символов	Вычисляет X как строковое выражение. Выдает сообщение C, если X содержит синтаксическую ошибку или дает не строковое (числовое) значение.

ОПЕРАЦИИ

ПРЕФИКСНЫЕ :

- число — отрицательное значение

ИНФИКСНЫЕ (ДВУХОПЕРАНДОВЫЕ) :

- `+` сложение для чисел, конкатенция для строк
- `-` вычитание
- `*` умножение
- `/` деление
- `**` возведение в степень (стрелка вверх). Сообщение B, если левый операнд отрицательный.
- `=` равенство
- `>` больше
- `<` меньше
- `>=` больше или равно
- `<=` меньше или равно
- `<>` не равно

Оба операнда должны быть одного типа. Результат равен 1, если сравнение истинно, и равно 0, если нет.

Функции и операции имеют следующие приоритеты:

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

ОПЕРАТОРЫ

Принятые обозначения :

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

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

Все операторы, кроме INPUT, DEF и DATA могут использоваться и как команды, и в программах. Команда или строка программы может содержать несколько операторов, разделенных двоеточием (':').

Нет ограничений на положение оператора в строке, хотя есть и некоторые ограничения в IF и REM.

Все операторы языка сведены в следующую таблицу :

ОПЕРАТОР	ДЕЙСТВИЕ ОПЕРАТОРА
1	2
BEEP X,Y	Воспроизводит звук длительностью X сек. и высотой Y полутонов вверх от основного тона ДО (или вниз, если Y отрицательное).
BORDER M	Устанавливает цвет рамки (бордюра) экрана. Выдает сообщение об ошибке K, если 0>M>9.
BRIGHT M	Устанавливает яркость выводимого символа: 0 - для обычной яркости; 1 - для повышенной яркости; 8 - сохраняет существующую яркость.
CAT ***	Без MICRODRIVE не работает.
CIRCLE X,Y,Z	Изображает дугу или окружность с центром в точке с координатами (X,Y) и радиусом Z.
CLEAR	Уничтожает все переменные и очищает занимаемую ими память. Выполняет RESTORE и CLS, устанавливает PLOT-позицию в нижнюю левую точку экрана и очищает GO SUB стек.
CLEAR N	Подобно CLEAR, но дополнительно изменяет системную переменную RAMTOP на 'N' и задает новый GO SUB стек.
CLOSE# ***	Без MICRODRIVE не работает. (Выдает каталог дискеты при работе с НГМД).
CLS	(CLEAR SCREEN) Очищает файл экрана.
CONTINUE	Продолжает выполнение программы, начатой ранее и остановленной с сообщением, отличным от 0. Если было сообщение 9 или L, то выполнение продолжается со следующего оператора, в других случаях - с того оператора, где случилась ошибка. Если сообщение возникло в командной строке, то CONTINUE вызовет попытку повторить командную строку и перейдет в цикл, если было сообщение 0:1, дает сообщение 0 , если было 0:2 или дает сообщение N, если было 0:3 или более. В качестве CONTINUE используется ключевое слово CONT на клавиатуре.
COPY	Пересылает копию 22 строк экрана на принтер, если он подключен. Помните, что по COPY нельзя распечатать находящийся на экране автоматический листинг. Выдает сообщение D, если нажать клавишу BREAK. (При работе с НГМД копирует файлы)
DATA E1,E2,E3,..	Часть списка данных. Должна располагаться в программе.
DEF FNA(A1,A2,..AK)=E	Определяемая пользователем функция. Должна располагаться в программе. A,A1,A2 и т.д. единственные буквы или буквы и $ для строковых аргументов, значений. Используется форма DEF FNA(), если нет аргументов.
DELETE F ***	Без MICRODRIVE не работает.

1	2
DIM A(N1,N2,...,NK)	Уничтожает массив с именем 'A' и устанавливает числовой массив 'A' с 'K' измерениями и присваивает всем его элементам значения 0.
DIM A$(N1,N2,...,NK)	Уничтожает массив или строку с именем 'A$' и устанавливает символьный массив с 'K' измерениями и присваивает всем его элементам значение ' '. Массив может быть представлен, как массив строк фиксированной длины NK, с K-1 размерностью. Сообщение 4 выдается, если недостаточно места для размещения массива. Массив не определен до его описания в DIM.
DRAW X,Y	То же самое, что и DRAW X,Y,0. Чертит прямую линию.
DRAW X,Y,Z	Изображает линию от текущей графической позиции в точку с приращениями X,Y по дуге в Z радиан. Выдает сообщение B при выходе за пределы экрана.
ERASE ***	Без MICRODRIVE не работает. (удаление файлов с ГМД)
FLASH N	Определяет, будет ли символ мерцающим или с постоянным свечением. N=0 для постоянного свечения, N=1 для мерцания, N=8 для сохранения предыдущего состояния.
FOR A=X TO Y	То же, что и FOR A=X TO Y STEP 1
FOR A=X TO Y STEP Z	Уничтожает скалярную переменную 'A' и устанавливает управляющую переменную 'X', предел 'Y', шаг приращения 'Z', зацикливает адрес, указанный в утверждении после FOR-оператора. Проверяет, если начальное значение больше (если STEP>=0) или меньше (если STEP<0), чем предел, то происходит переход к утверждению NEXT A или выдача сообщения 1, если нет (см. NEXT). Сообщение 4 выдается, если недостаточно места для размещения управляющей переменной.
FORMAT F ***	Без MICRODRIVE не работает. (форматирование ГМД)
GO SUB N	Записывает строку с оператором GO SUB в стек для использования затем, как GOTO N. Выдается сообщение 4, если не все подпрограммы заверши-лись с RETURN.
GO TO N	Продолжает выполнение программы со строки 'N'. Если 'N' пропущено, то с первой строки после этой.
IF X THEN S	Если 'X' истинно (не равно 0), то выполняется 'S'. 'S' включает все операторы до конца строки. Форма 'IF X THEN НОМЕР СТРОКИ' недопустима
INK N	Устанавливает цвет закрашивания (т.е. цвет, которым будут изображаться символы на цвете фона). 'N' в интервале от 0 до 7 указывает цвет. N=8 - оставить цвет без изменения, N=9 - увеличение контраста. Выдает сообщение K, если 'N' не лежит в интервале от 0 до 9.

1	2
INPUT ...	Где '...' есть последовательность вводимых символов, разделяемых как в операторе PRINT запятыми, точками с запятой или апострофами. Вводимыми символами могут быть: а) некоторый PRINT-символ, начинающийся не с буквы; б) имя переменной; в) строка имен переменных строкового типа. PRINT-символы в случае а) представляются так же, как и в операторе PRINT, за исключением того, что они все выводятся в нижнюю часть экрана. В случае б) компьютер останавливается и ждет ввода некоторого выражения с клавиатуры, значение которого будет присвоено переменной. Ввод осуществляется обычным образом, а синтаксические ошибки выдаются мерцающим [?]. Для строкового выражения вводной буфер устанавливается для размещения двух таких строк (который при необходимости может быть увеличен). Если первый вводимый символ STOP, то программа останавливается с сообщением H. Случай в) подобен случаю б) с той лишь разницей, что вводимая информация представляет собой строковый литерал неограниченной длины, и STOP в этом случае не сработает. Для останова Вы должны нажать клавишу 'КУРСОР ВНИЗ'.
INVERSE N	Символ управления инверсией выводимого символа. Если N=0, символ выводится в обычном виде с прорисовкой цветом закрашивания (INK) на цвете фона (PAPER). Если N=1, то цветовое решение изображения символа меняется на обратное. Выдает сообщение K, если 'N' не 0 или 1.
LET V=E	Присваивает значение 'E' переменной 'V'. Ключевое слово LET не может быть опущено. Скалярная переменная не определена, пока не встретится в операторах LET, READ или INPUT. Если 'V' индексируемая строковая переменная или сечение строкового массива (подстрока), то присваивание осуществляется с усечением справа или дополнением пробелами до фиксированной длины.
LIST	То же, что и LIST 0.
LIST N	Записывает часть программы в верхнюю часть экрана, начиная с первой строки, меньшей, чем 'N', и делает 'N' текущей строкой.
LLIST	То же, что и LLIST 0.
LLIST N	Подобно LIST, но вывод осуществляется на принтер.
LOAD F	Загружает программу и переменные.
LOAD F DATA ( )	Загружает числовой массив.
LOAD F DATA$ ( )	Загружает строковый массив.
LOAD F CODE M,N	Загружает старшие 'N' байтов, начиная с адреса 'M'.

1	2
LOAD F CODE M	Загружает байты, начиная с адреса 'M'.
LOAD F CODE	Загружает байты по тому же адресу, с которого они были разгружены.
LOAD F SCREEN$	Аналогично LOAD F CODE 16384,6912. Очищает файл экрана и загружает его с кассетного магнитофона.
LPRINT	Подобно PRINT, но использует принтер.
MERGE F	Подобно LOAD F, но не затирает всю старую программу в памяти, а заменяет только те строки и переменные, у которых совпадают номера или имена с такими же на ленте.
MOVE F1,F2 ***	Без MICRODRIVE не работает.
NEW	Запускает по новой систему программирования Бейсик, уничтожая старую программу и переменные и используемую память, включая и байт адреса в системной переменной RAMTOP, но сохраняет системные переменные UDG, P RAMT, RASP и PIP.
NEXT A	а) Находит управляющую переменную 'A'; б) Прибавляет к ней значение STEP; в) Если STEP >=0, а значение 'A' стало не больше значения 'ПРЕДЕЛ', или STEP<0, а значение 'A' не меньше, чем 'ПРЕДЕЛ', то происходит переход к оператору цикла. Сообщение 2 выдается, если не найдена переменная 'A'. Сообщение 1 выдается, если 'A' не является управляющей переменной цикла.
OPEN# ***	Без MICRODRIVE не работает.
OUT M,N	Выводит байт 'N' в порт 'M'. Операция выполняется на уровне микропроцессора (загружает в регистровую пару BC адрес 'M', а в регистр A 'N' и выполняет команду ассемблера OUT (C),A). 16384<=M<=65535, -255<=N<=255, иначе выдается сообщение B.
OVER N	Управляющий символ надпечатывается по выведенной строке. Если N=0, то выводимый символ затирает существующий в этой позиции. Если N=1, то новый символ соединяется со старым, образуя цвет закрашивания (INK) — если один из символов (но не оба сразу) имеет цвет закрашивания, или цвет фона, если оба символа имеют один и тот же цвет.
PAPER N	Подобен INK, но управляет цветом фона.
PAUSE N	Останавливает выполнение программы и задерживает изображение на экране на 'N' кадров (50 кадров в секунду- частота кадровой развертки) или до нажатия любой клавиши. 0<=N<=65535, иначе выдается сообщение B. При N=0 время задержки не учитывается и продолжается до нажатия любой клавиши.

5. ОСНОВЫ ПРОГРАММИРОВАНИЯ НА ЯЗЫКЕ БЕЙСИК

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

Наберите эти две строки программы вычисления суммы двух чисел

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

так, чтобы на экране появилось:

```
! 10 [>] LET A=10                          !
! 20     PRINT A                           !
!                                          !
!                                          !
!                                          !
!                                          !
!                                          !
!                                          !
!                                          !
!                                          !
! [ K ]                                    !
```

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

Теперь наберите:

```
15 LET B=15
```

и введите. Подобного невозможно было бы сделать, если бы нумерация начиналась с 1 и 2, а не с 10 и 20, как в нашем случае. (номер может быть в интервале от 1 до 9999).

Допустим, теперь вам понадобилось изменить строку 20 на следующую:

```
20 PRINT A+B
```

это можно сделать используя команду EDIT.

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

```
20 PRINT A+B
```

теперь нажмите ENTER и это вызовет замену старой строки 20 на новую, записанную в нижней части экрана. На экране это будет выглядеть так:

```
! 10     LET A=10                          !
! 15     LET B=15                          !
! 20 [>] PRINT A+B                         !
!                                          !
!                                          !
!                                          !
!                                          !
!                                          !
!                                          !
! [K]                                      !
```

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

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

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

Введите строку:

```
12 LET B=B
```

теперь для удаления этой строки наберите:

```
12          ( и затем ENTER)
```

программный курсор встанет между строками 10 и 15, по клавишами управления курсором вы можете установить его в любую строку. Еще раз выполните: 12 и ENTER, курсор снова установится между строками 10 и 15, теперь нажмите EDIT и строка 15 будет скопирована в нижнюю часть экрана. Оператор EDIT копирует вниз строку, следующую за строкой с новым номером. Нажмите ENTER для очистки нижней части экрана.

Теперь введите:

```
30          ( и затем ENTER)
```

программный курсор установится после конца программы, если вы теперь нажмете EDIT, вниз будет переслана строка 20.

И, наконец, выполните команду:

```
LIST 15
```

теперь вы увидете на экране:

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

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

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

Другое назначение номера строки - он служит именем оператора при ссылке к нему из другого места программы (в GO TO N).

Команда LIST без операндов выдает листинг с первой строки.

Команда NEW очищает память компьютера от старых программ и переменных.

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

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

Вы увидете, что заголовок выводится в строке 20, и у вас возникнет вопрос, что же делает строка 10? Компьютер игнорирует эту строку, это коментарий (REMARK или REMINDER). Все, что следует после REM, компьютером игнорируется до конца строки.

Вычисления доходят до строки 40 и компьютер переходит в ожидание ввода значения переменной F. Вы можете ввести это значение в режиме (L). Наберите число и нажмите ENTER. Компьютер выведет результат и снова перейдет в ожидание следующего числа. Этот переход обеспечивается в строке 60 в операторе GO TO 40.

Если на запрос очередного числа ответить STOP, то компьютер остановится с выдачей сообщения: 'H STOP IN INPUT IN LINE 40:1', которое поясняет причину останова и место останова (первый оператор в строке 40).

Если теперь вы желаете вновь продолжить выполнение порограммы, то введите CONTINUE и компьютер запросит очередное число.

При использовании CONTINUE компьютер запоминает (до выдачи им 'O OK') номер последней выполнявшейся строки и продолжает выполнение именно с этой строки.

Просмотрите внимательно оператор PRINT в строке 50, запятая в нем очень важна. Запятые в операторе PRINT используются для указания того, что вывод следующего после запятой знака должен продолжаться либо с левого края экрана либо с его середины, в зависимости от того, какая это по порядку запятая в данном операторе. Так, в строке 50, запятая предписывает выводить значения температуры в градусах Цельсия с середины экрана.

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

Оператор в строке 30 выводит чистую строку.

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

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

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

Теперь наберем еще несколько программных строк:

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

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

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

```
RUN 100
```

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

```
N$          ( удалив кавычки )
```

это сделает оператор INPUT в строке 110 подобным оператору

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

Если вы решили ввести STOP под строковый ввод, то должны установить курсор в начало строки, используя клавишу управления курсором <влево>.

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

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

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

остановить эту программу можно, если нажать клавиши CAPS SHIFT и SPACE, это вызовет ввод команды BREAK, которая остановит вычисления с выдачей сообщения 'L BREAK INTO PROGRAM'. Команда BREAK может быть использована и во время выполнения операции на магнитофоне или принтере, в этом случае выдается сообщение 'D BREAK--CONT REPEATS'. Команда CONTINUE в этом случае (как и в большинстве других), вызовет повторение оператора, в котором произошел останов. Ввод команды CONTINUE после сообщения 'L BREAK INTO PROGRAM' продолжит выполнение со следующего оператора.

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

```
N$          ( удалив кавычки )
```

поскольку значение 'N$' не определенно, то будет выдано сообщение: '2 VARIABLE NOT FOUND'. Если теперь вы выполните:

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

на что компьютер ответит ' 0 OK,0:1', а затем введете CONTINUE, то увидете, что программа завершится нормально.

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

Все приведенные утверждения PRINT, LET, INPUT, RUN, LIST, GO TO, CONTINUE, NEW и REM могут быть использованы либо как операторы в программе, либо как команды. Хотя RUN, LIST, CONTINUE и NEW чаще используются как команды, но могут быть использованы и в программе.

(3)

5.1 УСЛОВИЯ

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

Последовательность выполнения операторов программы не всегда предсказуема, в определенных местах программы компьютер может принимать решения о дальнейшем ходе вычислений. Оператор, реализующий это, имеет форму:
IF-некоторое истинное или ложное выражение, THEN-некоторое действие.

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

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

ELSE

Здесь IF-оператор имеет форму:

IF условие THEN ...

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

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

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

Например, выражения 1<2 , -2<1 , -3<1 истинны, а выражения 1<0 , 0>2 ложны.

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

Знаки условия набирают на клавиатуре следующим образом:

>	- SYMBOL SHIFT вместе с T	- больше
<	- SYMVOL SHIFT вместе с R	- меньше
<=	- SYMBOL SHIFT вместе с O	- меньше или равно
	(нельзя набирать < и =)	
>=	- SYMBOL SHIFT вместе с E	- больше или равно
<>	- SYMBOL SHIFT вместе с W	- не равно

5.3 ЦИКЛЫ

(4)

Краткое содержание: FOR, NEXT, TO, STEP

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

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

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

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

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

Наша программа при использовании этих операторов будет выглядеть так:

```
10 LET TOTAL=0
20 FOR C=1 TO 5
30 INPUT A
40 REM C=NUMBER OF TIMES THAT A HAS BEEN INPUT SO FAR
50 LET TOTAL=TOTAL+A
60 NEXT C
70 PRINT TOTAL
```

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

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

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

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

то 'C' последовательно примет значения 1, 2.5 и 4.

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

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

следующая программа выводит числа домино:

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

Значение STEP, равное 0, вызовет бесконечное повторение цикла, этого не рекомендуется делать.

(5)

5.4 ПОДПРОГРАММЫ

Краткое содержание: GO SUB, RETURN

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

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

GO SUB N

где "N" номер первой строки в подпрограмме.
Этот оператор подобен GO TO N с той разницей, что при использовании GO SUB компьютер запоминает следующий после GO SUB оператор, которому передается управление после выполнения подпрограммы.
Делатся это посредством помещения номера оператора (адреса возврата) в специальную область памяти, называемую GO SUB-стек.

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

Приведем пример использования подпрограммы:

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

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

5.5 О П Е Р А Т О Р Ы READ, DATA И RESTORE

(6)

Краткое содержание: READ, DATA, RESTORE

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

Например:

```
10 READ A,B,C
20 PRINT A,B,C
30 DATA 10,20,30
40 STOP
```

Оператор READ состоит из слова READ, за которым следует список имен переменных, разделенных запятыми. Он выполняется значительно эффективнее оператора INPUT, особенно когда вводимые значения присваиваются переменным. В этом случае компьютер ищет величины переменных в утверждении DATA.

Каждый оператор DATA - это список значений (числовых или строчных), разделенных запятыми. Вы можете вводить их, где вам угодно в программе, так как компьютер игнорирует их, за исключением тех случаев, когда их использует оператор READ.

Вы должны представлять себе выражения для всех операторов DATA в программе, что позволит вам вводить совместно формы одной длины списка выражений (списка DATA). Сначала компьютер выбирает первое значение из списка DATA для величины из оператора READ, в следующий раз он берет второе значение из списка DATA, и, таким образом, выбираемые последовательно операторы READ обрабатываются с использованием списка DATA (если оказалось, что список просмотрен до конца, то возникает ошибка).

Заметьте для себя, что является бесполезным введение оператора DATA в виде управляющего оператора, так как оператор READ не обнаружит его. Оператор DATA выполняется только в программе.

Можно посмотреть, как эти требования выполнены в весьма характерной программе:

Строка 10 дает указание компьютеру читать три значения данных и присвоить их переменным A, B и C.
Строка 20 говорит о том, что надо вывести (PRINT) три значения,
в строке 30 оператор DATA задает значения для A, B и C,
строка 40 заканчивает программу.
Информация в сператоре DATA может быть частью FOR...NEXT цикла.

Например:

```
10 FOR N=1 TO 6
20 READ D
30 DATA 2,4,6,8,10,12
40 PRINT D
50 NEXT N
60 STOP
```

Когда выполняется эта программа, то можно увидеть, как READ оператор перебирает весь список DATA.

Оператор DATA может содержать также и строчные переменные.

Например:

```
10 READ D$
20 PRINT "THE DATA IS",D$
30 DATA "JUNE 1ST,1982"
40 STOP
```

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

Попробуйте выполнить такую программу:

```
10 READ A,B
20 PRINT A,B
30 RESTORE 10
40 READ X,Y,Z
50 PRINT X,Y,Z
60 DATA 1,2,3
70 STOP
```

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

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

(7)

5.6 АРИФМЕТИЧЕСКИЕ ОПЕРАЦИИ

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

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

Пример:

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

Отсюда видно, что вычисления могут быть комбинированными. Комбинации такого типа, как

```
SUM*15/100
```

называются выражениями. Выражение - это самый короткий путь для указания компьютеру на то, что вычисления надо делать одно за другим. В нашем примере выражение

```
SUM*15/100
```

указывает: возьми значение переменной с именем "SUM", умножь его на 15 и затем раздели на 100. Если вы не можете еще этого сделать, мы рекомендуем посмотреть вводную часть этой книги, чтобы ознакомиться с тем, как ZX SPECTRUM работает с числами и каков порядок, в котором выполняются математические выражения.

КРАТКОЕ ПОВТОРЕНИЕ:

Умножение и деление выполняется первыми. Они имеют более высокий приоритет, чем сложение и вычитание.

Относительно друг друга умнсжение и деление имеют равные приоритеты. Существует правило, по которому умножение и деление выполняются последовательно слева направо. Когда все они выполнятся, будут выполняться сложение и вычитание по порядку (также слева направо).

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

Вы можете использовать операцию сложение ("+")для сцепления строк (конкатенции) в выражениях.

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

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

Примеры допустимых имен переменных:

```
X
T42
THIS NAME IS SO LONG THAT I SHALL NEVER BE ABLE TO TYPE
     IT OUT AGAIN WITHOUT
MAKING A MISTAKE
```

Примеры недопустимых имен переменных:

```
    2001                          ( начинается с цифры)
    3 BEARS                       ( начинается с цифры)
    M*A*S*H                       ( знак "*"-не буква и не цифра)
    FOTHERINGTON-THOMAS           ( содержит знак "-")
```

Числа и выражения могут задаваться в экспоненциальной форме.
Попробуйте выполнить:

```
    PRINT 2,34E0
    PRINT 2,34E1
    PRINT 2,34E2      и т.д. до
    PRINT 2,34E15
```

Помните,что оператор PRINT дает числа лишь в значащих цифрах.
Попробуйте выполнить еще:

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

и вы увидете, что компьютер может воспринять только цифры 4294967295.
Компьютер ZX SPECTRUM использует арифметику с плавающей точкой (запятой), при этом различные части числа (мантисса и порядок) хранятся в отдельных байтах, что приводит к не всегда точным результатам даже для целых чисел. Выполните:

```
    PRINT 1E010+1-1E10,1E10-1E10+1
```

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

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

Погрешность в 5E9 составляет около 1, а с прибавлением единицы фактически округлится до 2.
Числа 5E9+1 и 5E9+2 для компьютера равны. Наибольшее целое, которое может воспринять компьютер, равно 2**32-1 или (4 294 967 295).
Строка "" без единого символа называется пустой или нулевой строкой. Не путайте ее с пробелом. Наберите:

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

Когда вы нажмете клавишу ENTER, вы получите мерцающий знак вопроса, указывающий ошибочное место в строке. Когда, при интерпретации этой строки, компьютер найдет двойную кавычку, открывающую "FINNEGANS WAKE", то сочтет ее закрывающей кавычкой для строки "HAVE YOU FINISHED" и затем не сможет вывести "FINNEGANS WAKE". Здесь надо помнить специальное правило:

== Если вы хотите вывести кавычки внутри строки,они должны удваиваться. ==

Например:

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

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

(8)

5.7 СТРОКИ СИМВОЛОВ

Краткое содержание: сечения, использование TO
Примечание: эти операции отсутствуют в стандартном Бейсике

Пусть имеется строка символов, тогда ее подстрокой будет некоторая последовательность символов из этой строки, так "STRING" является подстрокой от "BIGGER STRING", а "W STRING" и "BIG STRING" не являются. Существует действие, называемое сечением для определения подстрок и которое может применяться к строковым выражениям. Общая его форма:

'СТРОКОВОЕ ВЫРАЖЕНИЕ'('начало' TO 'конец')

следующее выражение истинно:

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

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

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

Последнее выражение можно было бы записать и так:

"ABCDEF"( ),

что тоже верно. Можно опускать и слово TO:

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

'начало' и 'конец' должны находиться в пределах строки, иначе будет выдано сообщение об ошибке. Так выражение:

"ABCDEF"(5 TO 7)

вызывает сообщение '3 SUBSCRIPT', так как 'конец' превышает длину строки (6).

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

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

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

Следующая программа иллюстрирует эти правила:

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

Можно также присваивать значения подстроке. Попробуйте:

```
10 LET A$= "I AM THE ZX SPEKTRUM"
20 PRINT A$
30 LET A$(5 TO 8) = "******"
40 PRINT A$
```

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

Если вы теперь выполните

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

и

```
PRINT A$ ","
```

вы увидете, что будут выведены дополнительные пробелы, так как 'A$( )' считается подстрокой. Для правильного выполнения следует писать:

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

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

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

5.8 ФУНКЦИИ

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

Функции - это зашитые в BASIC-систему подпрограммы, которые, получая на входе одни значения, называемые аргументами, возвращают другие значения - результаты.

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

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

```
PRINT LEN "SINCLAIR"
```

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

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

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

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

аналогично по действию команде

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

или выполните

```
PRINT LEN STR$ 100.0000
```

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

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

```
VAL  "3.5" = 3.5
```

или

```
VAL  "2*3" = 6
```

или даже так

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

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

Можно попасть в затруднительное положение, например:

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

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

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

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

сделайте

```
LET A$ = "99"
```

и затем выведете все следующие значения:

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

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

Функция SGN - это так называемая математическая функция сигнум (знак), и аргумент и результат ее числовые. Результат равен:

```
 1 , если аргумент положителен;
 0 , если аргумент равен 0;
-1 , если аргумент отрицателен.
```

Функция ABS преобразует аргумент в положительное число:

ABS -3.2 = ABS 3.2 = 3.2

Функция INT (от 'INTEGER PART' - целая часть) преобразует дробное число к целому отбрасыванием дробной части:

INT 3.9 = 3

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

INT -3.9 = -4

Функция SQR вычисляет корень квадратный от числа, например:

SQR 4 = 2

SQR 0.25 = 0.5

SQR 2 = 1.4142136 (приближенно).

Если аргумент отрицательный, то выдается сообщение:

'A INVALID ARGUMENT'.

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

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

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

DEF вводится в соответствующем режиме (SIMBOL SHIFT и 1). Теперь функция может использоваться в программе:

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

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

```
20 DEF FN R(X) = INT(X + 0.5); REM GIVES X ROUNDET TO
                                  THE NEAREST INTEGER.
```

и можете затем попробовать ввести:

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

Введите и выполните следующее:

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

Есть одна тонкость в этой программе: во-первых, функция FN Q не использует аргументов, но скобки при этом должны обязательно использоваться. Во-вторых, операторы DEF не выполняемые, компьютер после выполнения строки 10 просто переходит к выполнению строки 40. Помните, что DEF может быть только оператором, но не командой. В третьих, 'X' и 'Y' имена целых переменных в программе и, в то же время, имена аргументов в функции FN P.

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

При вычислении FN Q( ) участвуют только переменные программы, так как аргументов нет, и ответ в этом случае будет 10+0*0=10. Теперь изменим строку 20 на

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

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

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

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

RIGHT$(A$,N) - возвращает подстроку, содержащую 'N' последних символов в строке 'A$';

TL$(A$) - возвращает подстроку, содержащую все символы строки 'A$',кроме первого.

Вы можете определить такие функции на своем компьютере:

```
10 DEF FN T$(A$) = A$(2 TO): REM TL$
20 DEF FN L$(A$,N)=A$(TO N): REM LEFT$
```

проверте их работу со строками длиной 0 и 1.

П Р И М Е Ч А Н И Е : Функция может иметь до 26 числовых аргументов и, в то же время, до 26 строковых.

5.9 МАТЕМАТИЧЕСКИЕ ФУНКЦИИ

(10)

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

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

** и EXP

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

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

Таким образом, запись A**B означает: 'умножь 'A' само на себя 'B' раз. Но это предполагает, что B - положительное целое число.

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

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

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

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

полезно помнить, что:

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

Проэкспериментируйте с этим, попробовав выполнить такую программу:

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

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

Другой типичный пример использования этой операции - это вычисление дохода. Предположим, что вы вложили часть своих денег в общественное строительство, которое приносит вам 15% годовых. После года вы будете иметь уже не точно 100% от того, что имели в начале, а плюс 15%, что составит 115%. Для вычисления другим способом, вы умножаете вашу сумму денег на 1,15 и получаете тот же результат. В конце следующего года вы получите прибыль, что в сумме составит 1,15*1,15=1,15**2=1,3225 от вашей первоначальной суммы. В итоге после Y лет вы будете иметь в 1,15**Y раз больше денег.

Выполнив операторы:

```
FOR Y=0 TO 100:PRINT Y,10*1,15**Y:NEXT Y
```

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

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

Предположим вы записали:

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

здесь A определено в операторе LET, его значение передается для вычисления степени.

Имеется определенное значение A, которое делает функцию FN A иллюстрирующей специальную математическую функцию. Это значение называется 'E'.

ZX SPECTRUM имеет специальную функцию, называемую EXP и определяемую как:

EXP X = E**X .

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

PRINT EXP 1

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

LN

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

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

LOGA X = LN X/LN A

PI

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

3,141592653589...

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

PRINT PI

SIN, COS, TAN и ASN, ACS, ATN

Тригонометрические функции измеряют те случаи, когда точка перемещается вокруг окружности единичного радиуса. Точка стартует с позиции "3-х часов" и перемещается против часовой стрелки. Начало координат находится в центре этой окружности. Тогда SIN угла между радиусом, соединяющим движущуюся по окружности точку с началом координат, будет ордината этой точки, а COS - абсциса. Необходимо помнить, что если точка находится слева от оси Y, то косинус отрицательный, а если точка находится под осью X, то отрицательный синус. Необходимо помнить, что:

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

имеются и другие тригонометрические функции:

TAN - тангенс;
ASN - арксинус;
ACS - арккосинус;

§ 10 СЛУЧАЙНЫЕ ЧИСЛА

(11)

Краткое содержание: RANDOMIZE, RND.

В этой главе описывается функция RND и ключевое слово RANDOMIZE. Их не надо путать, хотя они обе расположены на клавише 'T'. Для RANDOMIZE допустимо сокращение RAND.

При обращении к функции RND, она возвращает случайное число в интервале от 0 до 1 (может принимать значение 0, но никогда 1). Попробуйте выполнить:

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

ENTER ENT RUN ENT продолж. Y Законч BREK

вы увидете как меняется результат.

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

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

```
1.3 + 0.7 * RND
```

даст интервал от 1.3 до 2.

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

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

будет давать числа 1,2,3,4,5,6.

Пусть имеется программа:

```
10 REM THROWING PROGRAM          (выбрасывание кости)
20 CLS
30 FOR N=1 TO 2
40 PRINT 1+ INT(RND*6;' ';)
50 NEXT N
60 INPUT AR:GO TO 20
```

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

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

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

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

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

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

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

Ниже приводится текст программы, моделирующей выбрасывание монеты и подсчета числа выпадений 'орла' и 'решки'.
(перевод имен программы: HEADS - орлы, TAILS - решки, COIN - монета)

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

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

(12)

5.11 МАССИВЫ

Краткое содержание: DIM.

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

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

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

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

Элементы массивов называются индексируемыми переменными. Перед использованием массива необходимо зарезервировать под него память, это делается в операторе DIM (от латинского DIMENSION). В нашем случае это будет оператор DIM B(10), который определяет массив с именем B и размерностью 10 (т.е. 10 индексируемых переменных B(1),B(2),...B(10) и присваивает всем элементам массива значение 0.

Итак, теперь мы можем записать:

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

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

(номер страницы, номер строки, номер столбца).

Объявим двумерный массив 'C' с размерностью 3 и 6:

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

что даст 3*6=18 индексируемых переменных:

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

Могут быть также строковые массивы. Строки в таких массивах отличаются от скалярных тем, что имеют фиксированную длину, а присваивание им значения осуществляется с усечением справа или добавлением до полной длинны пробелами. Имя строкового массива образуется добавлением справа к имени специального символа, перечеркнутой буквы S (знак доллара - в данном описании заменен на '$').

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

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

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

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

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

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

даст '123456789 7'

Можно использовать также сечения массивов:

```
AЙ(2,4 TO 8)=AЙ(2)(4 TO 8)='45678'
```

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

5.12 ЛОГИЧЕСКИЕ ОПЕРАЦИИ

(13)

Краткое содержание: AND, OR, NOT

Если мы взглянем на описанную ранее форму оператора IF:

```
IF условие THEN
```

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

Некоторое выражение 'и' некоторое другое выражение истинны, если истинны оба этих выражения. Например:

```
IF AЙ='YES' AND X>0 THEN PRINT X
```

'X' будет напечатано только тогда, когда

```
AЙ='YES' и X>0
```

Некоторое выражение 'или' некоторое другое выражение истинны, если истинно хотя бы одно из этих выражений. 'не' выражение истинно, если ложно само выражение и наоборот.

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

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

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

Тем, кто боится сложностей, следующие разделы можно опустить.

1. Условия =, <, >, <=, >=, <> дают числовой результат 1 для истины и 0 если ложь. Например, оператор PRINT 1=2, 1<>2 выведет 0 для "1=2", которое ложно, и 1 для "1<>2", которое истинно.
2. В операторе "IF условие THEN ...", само условие может быть числовым выражением. Если значение после вычисления равно 0, то считается что это ложь, если другое значение (включая и 1), то считается, что это истина. Таким образом IF-оператор можно представить:

```
IF условие <> 0 THEN ...
```

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

```
A<>B тоже, что и NOT  A=B
NOT A<>B  тоже, что и A=B
X AND Y имеет значение X, если Y~=0 и 0, если Y=0
X  OR Y имеет значение 1, если Y~=0 и X, если Y=0
  NOT Y имеет значение 0, если Y~=0 и 1, если Y=0
```

например:

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

в каждой операции будет выводиться большее из двух чисел A или B.

Пример использования OR:

```
LET TOTAL PRICE=PRICE LESS TAX*(1.15 OR VЙ'ZERO RATER'
```

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

```
XЙ AND Y имеет значение XЙ, если Y~=0, и " ", если Y=0
```

где " " - пустая строка.

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

```
10 INPUT "TYPE IN TWO STRING"AЙ,BЙ
20 IF AЙ>BЙ THEN LET CЙ=AЙ:LET AЙ=WЙ:LET BЙ=CЙ
30 PRINT AЙ;" ";("<" AND AЙ<BЙ)+("=" AND AЙ=BЙ)
40 PRINT " ";BЙ
50 GO TO 10
```

(15) 5,13

ДОПОЛНИТЕЛЬНЫЕ СВЕДЕНИЯ ОБ ОПЕРАТОРАХ PRINT И INPUT

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

Выражения, значение которых используется в операторе PRINT, называются PRINT-параметрами. Они разделяются запятыми или точкой с запятой, называемыми PRINT-разделителями.

В операторе PRINT возможно отсутствие некоторых параметров. В этом случае ставятся две запятые подряд.

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

выведет '*' в середине экрана.

```
AT 'строка','столбец'
```

этот параметр перемещает позицию вывода в место, определяемое номером строки и столбца. Номер строки меняется от 0 (верхняя) до 21, а номер столбца от 0 (левый) до 31.

Оператор SCREEN$. Его действие противоположно действию оператора PRINT AT, он использует те же параметры, номер строки и столбца, но их значения заключаются в скобки.

Оператор SCREEN$ сообщает вам, какой символ находится на экране в указанной позиции.

Выполнив:

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

мы получим '*' выведенную предыдущим оператором.

В качестве возвращаемого значения могут использоваться: алфавитно-цифровые символы, специальные символы, пробелы. Линии, нарисованные с помощью операторов: PLOT, DRAW, CIRCLE, определяемые пользователем символы и графические символы, возвращаются, как пустая строка.

То же, когда функция OVER используется для получения комбинированных знаков.

```
TAB столбец
```

этот параметр перемещает позицию вывода в указанный столбец на той же строке, или переходит на новую строку, если столбец был последним.

Помните, что компьютер уменьшает номер позиции по модулю 32 (т.е. делит на 32 и использует остаток).

Так 'TAB 33' равнозначно 'TAB 1'.

К примеру:

```
PRINT TAB 30,1;TAB 12;"CONTENTS";AT 3,1;"CHAPTER";
TAB 24;"PAGE"
```

Выведет на экран оглавление для первой страницы книги.

'. Можно использовать ',' (или вообще ничего).

При этом необходимо следить за установкой текушей позиции вывода.

2) нельзя использовать для вывода две нижние строки экрана (22,23), т.к. они используются для получения оператором INPUT данных. Последняя используемая строка - 21.

3) можно использовать параметр AT для установки позиции вывода в то место, где уже имеется выведенная информация, при этом каждый новый символ уничтожает старый.

Еще одним оператором, используемым совместно с PRINT, является CLS, он производит очистку экрана, подобно операторам CLEAR и RUN.

При заполнении всего экрана происходит его свертка. В этом можно убедиться проделав:

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

и далее выполнить 'PRINT 99' некоторое количество раз.
Или вариант с остановом вывода, для просмотра текста. Чтобы убедиться в этом выполним:

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

когда экран заполнится, вывод остановится и в нижней части экрана появится запрос : "SCROLL ?". После просмотра нажмите "Y" (да) и вывод продолжится. Возможен отрицательный ответ "NO" (нет), STOP ("SIMBOL SHIFT"+"A") или SPACE (BREAK). В последнем случае компьютер остановит программу и выдаст сообщение: 'D BREAK-CONT REPEATS'.

Оператор INPUT используется для ввода различных значений. Например:

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

компьютер выведет на экран (в нижней части) вопрос, в ответ на который вы должны ввести свой возраст.
Фактически INPUT содержит те же параметры, что и PRINT, так "HOW OLD ARE YOU?" и 'AGE' оба являются операторами INPUT. Однако существуют и некоторые отличия.

Первое: это дополнительный параметр - переменная, значение которой вы должны присвоить (в нашем примере 'AGE').

Второе: вы можете выводить значение переменной, как часть запроса, заключив ее для этого в скобки.
Пример:

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

Значение 'MY AGE' выдает компьютер, значение 'YOUR AGE' вводите вы сами, по мере выдачи операторов INPUT происходит свертка экрана. Рассмотрим пример использования AT в INPUT-операторе:

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

когда "THIS IS LINE 2" будет выведено, нижние строки станут сдвигаться вверх, освобождая место, но нумерация останется прежней.

Выполним:

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

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

```
INPUT LINE A$
```

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

Управляющие символы CHR$22 и CHR$23 выполняют функции, подобные параметрам TAB и AT. Их преимущество состоит в том, что можно использовать имена переменных, а для TAB и AT это невозможно. Эти управляющие символы обрабатываются как числа. Аналогом AT является управляющий символ CHR$22, первое значение определяет строку, второе столбец.

```
PRINT CHR$22+CHR$1+CHR$0;
```

то же, что и

```
PRINT AT 1,0;
```

как значения параметров рассматриваются только CHR$1 и CHR$0 (CHR$22 не учитывается).

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

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

то же, что и

```
PRINT TAB A+256*B
```

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

```
POKE 23692,255
```

компьютер станет сворачивать экран без запроса 255 раз, прежде чем запросит свертку. Так например запустите:

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

и следите сколько сверток сделает компьютер.

Запустим следующую программу, проверяющую знание таблицы умножения:

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

Можно несколько изменить программу, так чтобы не зная правильного ответа, можно было узнать его. К примеру, компьютер спрашивает, сколько будет 2*3. Не зная ответа, вы вводите 2*3 и получаете его. Для этого замените в 30 строке 'C' на 'C$', в 100 строке на 'VAL C$' и дополнительно введите строку:

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

для исключения подсказки поменяйте 'C$' в строке 30 на 'LINE C$'

(14) S.14

НАБОР СИМВОЛОВ

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

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

Компьютер ZX SPECTRUM использует 256 символов с кодами от 0 до 255. Все они приведены в приложении 1. Для преобразования из символьной формы во внутреннюю кодовую и наоборот служат две функции CODE и CHR$.

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

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

Следующая программа выводит весь отображаемый символьный набор:

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

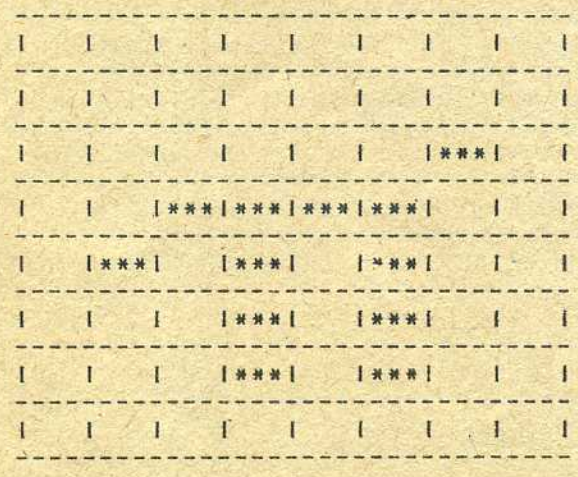
Все эти символы (кроме знака фунта и 'C' в кружочке) образуют код ASCII (American Standart Codes for Information Interchange).

Следующие символы не входят в ASCII, но используются в ZX SPECTRUM. Первые из них это 15 черно-белых значков , называемых графическими символами и используемых для изображения рисунков. Их можно ввести с клавиатуры, используя так называемый "графический режим". Если вы нажмете GRAPHICS (CAPS SHIFT 9), то курсор изменится на <G>. Теперь цифровые клавиши с 1 по 8 выдают графические символы, обозначенные на клавишах, а если при этом удерживать SHIFT, то они будут выдавать инверсные символы, т.е. черное становится белым, а белое черным.

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

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

1. Каждый символ представляется точками в матрице 8*8, поэтому вначале начертим диаграмму , приведенную на рисунке. По периметру символа оставим по одной клетке для отделения его от других знаков.



2. Закрепим этот символ за клавишей "P", так, чтобы при нажатии клавиши в графическом режиме выдавался символ "ПИ".

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

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

Эти восемь двоичных чисел запоминаются в памяти в восьми ячейках, каждая из которых имеет свой адрес, для нашего символа адрес первого из восьми байтов в группе будет USR "P". Второй байт имеет адрес USR "P"+1 и т.д. до USR "P"+7.

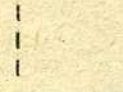
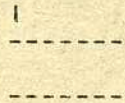
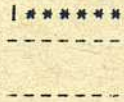
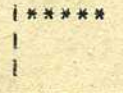
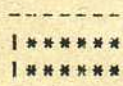
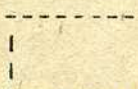
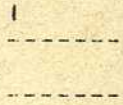
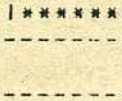
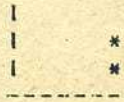
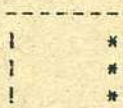
USR - функция преобразования строки символов в адрес первого байта в строке. Строковый аргумент может содержать единственный символ, который будет обозначать символ, определяемый пользователем. Имеются и другие функции применения USR с числовым аргументом, мы рассмотрим их позже. Поясним все сказанное программой:

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

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

Оператор "POKE" записывает данные непосредственно в память, минуя обычный аппарат бейсика. Обратным оператору "POKE" является оператор "PEEK", который служит для отображения области памяти. Более подробно этот оператор будет рассмотрен в разделе "ПАМЯТЬ".

ГРАФИЧЕСКИЕ СИМВОЛЫ:

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

Вернемся к знакам. Первые 32 знака с кодами от 0 до 31 - это управляющие символы. Они не отображаются, вместо них на экране телевизора отображается знак "?". Назначение этих символов описано в приложении 1. Три символа с кодами 6, 8 и 13 имеют специальное назначение при работе с экраном.

CHR$6 печатает пробел, используемый как запятая в операторе PRINT.
PRINT 1;CHR$6;2 даст тот же результат, что и PRINT 1,2.
Но это не совсем корректное использование, правильнее будет написать

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

CHR$8 - символ забоя, обеспечивает возврат на одну позицию назад. Оператор PRINT "1234";CHR$8;"5" даст строку "1235".
CHR$13 - перевод строки, продолжает вывод с новой строки. При работе с экраном используются также символы с кодами 16 и 23, которые подробнее будут рассмотрены ниже.

Все символы расположены в кодовой таблице в алфавитном порядке по возрастанию кодов. Причем все строчные буквы расположены после заглавных.

Существует правило по которому сравниваются две строки. Сначала сравниваются первые символы, если они различаются, то строка, содержащая символ с меньшим кодом считается меньшей, а если они равны, то выбирается следующая пара символов, и так до тех пор, пока не встретятся несовпадающие символы, либо пока одна из строк не кончится - она и будет считаться меньшей. В противном случае строки считаются равными. Отношения =, >, <, =>, <=, <> применяются к строкам символов так же как и к числам. Знак "<" означает "находится впереди в кодовой таблице", а знак ">" - соответственно, позади.

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

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

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

```
P - за пешкой   (POWN)
R - за ладьей   (POOK)
N - за конем    (KNIGHT)
B - за слоном   (BISHOP)
K - за королем  (KING)
Q - за королевой (QUEEN)
```

```
  5 LET B=BIN 01111100:LET S=BIN 00111000:
    LET D=BIN 00010000
 10 FOR N=1 TO 6: READ P$: REM 6 PIECES
 20 FOR F=0 TO 7: REM READ PIECE INTO 8 BYTES
 30 READ A: POKE USR P$+1,A
 40 NEXT F
 50 NEXT N
100 REM BISHOP
110 DATA "B",0,D,BIN 00101999,BIN 01000100
120 DATA BIN 01101100,C,B,0
130 REM KING
140 DATA "K",0,D,C,D
150 DATA C,BIN 01000100,C,0
160 REM POOK
170 DATA "R",0,BIN 01010100,B,C
180 DATA C,B,B,0
190 REM QUEEN
200 DATA "Q",0,BIN 01010100,BIN 00101000,D
210 DATA BIN 01101100,B,B,0
220 REM POWN
230 DATA "P",0,0,D,C
240 DATA C,D,B,0
250 REM KNIGHT
260 DATA "N",0,D,C,BIN 01111000
270 DATA BIN 00011000,C,B,0
```

(16)

5.5 Ц В Е Т А

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

Выполним следующую программу:

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

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

Ниже дана кодировка цветов:

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

Для черно-белого телевизора этот ряд представляет собой последовательность перехода серых полутонов от черного до белого.
Для использования цветов уясним строение графического экрана. Он состоит из 768 позиций (24 строки по 32 знакоместа), каждая из которых представляет из себя матрицу 8 на 8 пикселей.
Вспомним:

0-белая точка
1-черная точка

Позиция символа (знакоместо) также рассматривается с этих позиций: INK-цвет тона, PAPER-цвет фона, т.о. знакоместо состоит из INK и PAPER обычной и повышенной яркости, а так же мерцающих и немерцающих.
Все это имеет следующую кодировку:

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

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

```
PAPER 5
```

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

Формат операторов:

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

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

```
PAPER 8
```

не изменит цвета фона (т.к. такого цвета нет), а поможет выяснить значение предыдущего PAPER. Операторы: INK 8, BRIGHT 8, FLASH 8 выдадут значения этих атрибутов.

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

Выполним:

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

Можно запустить программу, выдающую на экран дисплея цветные полосы:

```
INK 9:PAPER 8:PRINT AT 0,0;:FOR N=1 TO 1000:
PRINT N;:NEXT N
```

цвет фона будет контрастен цвету тона в каждой выводимой позиции.

Цветной телевизор построен на способности человеческого глаза воспронимать только три первичных цвета-синий, красный и зеленый. Другие цвета образуются из их сочетаний. К примеру, фиолетовый цвет образуется, как комбинация синего с красным (код фиолетового цвета '3', он является суммой кодов синего '1' и красного '2'.)

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

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

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

Операторы: INVERSE и OVER не управляют атрибутами, но тем не менее определяют способ вывода на экран. В этих операторах используются значения параметров '0' и '1'. Если вы дадите: INVERSE 1, то выводимый символ изменит свою обычную форму (вывод будет осуществляться в негативном изображении).

В обычном виде мы пишем черным по белому, в инверсном белым по черному.

Оператор: OVER 1

устанавливает режим расширенного вывода. В обычном режиме при выводе символа на знакоместо, там стирается все выведенное ранее, при расширенном выводе можно накладывать символы друг на друга. Это позволяет выводить символы, например стилизованные шрифты.

Программа для вывода готического шрифта:

```
10 OVER 1
20 FOR N=1 TO 32
30 PRINT "0";CHR$8;"""";
40 NEXT N
```

Отметим, что управляющий символ CHR$8 возвращает на одну позицию.

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

В примере:

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

только 'X' будет выведен на желтый фон.

	[K],[L],[C]		[G]			[E]		
	SYMBOL SHIFT	CAPS SHIFT		любой		CAPS SHIFT	SYMBOL SHIFT	
1	!	EDIT	фон голубой	тон голубой	DEF FN			
2	@	CAP LOC	фон красный	тон красный	FN			
3	#		фон фиолето вый	тон фиолето вый	LINE			
4	$	INVERS VIDEO	фон зеленый	тон зеленый	OPEN			
5	%	курсо влево	фон синий	тон синий	CLOSE			
6	&	курс вниз	фон желтый	тон желтый	MOVE			
7	'		фон белый	тон белый	ERASE			
8	(	курсор вправо	нормал. яркость	нормал. мерцан.	POINT			
9	)	графи- ческий режим	графич. выход	графич. выход	повыш. яркость	C мерцан.	CAT	
10	-	DELETE	DELETE	DELETE	фон черный	закраш. черный	FORMAT	

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

```
BORDER 'цвет'
```

Кодировка цветов прежняя. Возможны мерцание и повышенная яркость, для этого используйте соответствующие параметры в INPUT (наподобие PRINT). Эти параметры действуют до конца оператора или до тех пор, пока запрашиваемые данные не будут введены.
Выполним:

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

Возможно изменение цветов и с помощью управляющих символов, подобных управляющим символам для AT и TAB

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

так

```
PRINT CHR$16+CHR$9
```

то же, что и

```
PRINT INK 9;
```

Можно пользоваться либо управляющими символами, либо операторами. Их можно ставить как после номера строки, так и в конце строки. Для удобства можно пользоваться в расширенном режиме [E] цифрами. Цифры от 0 до 7 устанавливают цвет INK, если CAPS SHIFT нажата, и цвет PAPER, если не нажата. Если нажать цифру в [E]-режиме, то будут выведены CHR$17 и CHR$(код цвета). Если в это время была нажата CAPS SHIFT, то будут выведены: CHR$16 и CHR$код цвета. Если вы захотите уничтожить вводимое, нажмите DELETE два раза, после первого раза на экране высветится знак вопроса, или что-нибудь еще, не пугайтесь и нажмите DELETE еще раз. Управление курсором тоже не работает обычным образом до тех пор, пока курсор не выйдет к предыдущему управляющему символу.
Действия в расширенном режиме [E]:

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

В [L]-режиме:

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

Функция ATTR имеет следующий формат:

```
ATTR ('строка','столбец')
```

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

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

Пример: знакоместо мерцающее, обычной яркости, желтый фон, синий тон 128+0+(8*6)+1=177
проверим это, выполнив:

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

Упражнения:
1)

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

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

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

дополнительно к описанному выше утверждению, то мы в результате увидим 'B' не перечеркнутое '/'. Так ли это?

2) выполним :

```
PAPER 0:INK 0
```

Действуют ли эти операторы в нижней части экрана? что мы увидим если добавить

```
BORDER 0 ?
```

3) выполним программу:

```
10 POKE 22527+RND*127
20 GO TO 10
```

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

5.16 ГРАФИКА

(17)

Краткое содержание: PLOT, DRAW, CIRCLE, POINT

Эта глава описывает возможности компьютера ZX SPECTRUM по отображению графической информации. Экран компьютера содержит 22 строки по 32 символа в каждой, что составляет 22*32=704 символьные позиции. Каждая символьная позиция представляется квадратом 8*8 точек, называемых пикселями.

Пиксель задается двумя числами - его координатами. Первое задает координату X, то есть удаление (в пикселях) до левой границы экрана, второе - задает координату Y (удаление от нижней границы экрана). Координаты записываются в скобках, например : (0,0),(255,0),(0,175) и (255,175). Они задают соответственно нижний левый, нижний правый, верхний правый и верхний левый углы экрана.

Оператор PLOT X,Y вызывает высвечивание закрашивающим цветом (INK) пикселя с указанными координатами.

Например программа:

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

будет высвечивать некоторый случайный пиксель при каждом нажатии ENTER.

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

```
10 FOR N=0 TO 255
20 PLOT N,88+80*SIN(N/128*PI)
30 NEXT N
```

Или программа:

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

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

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

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

Оператор DRAW чертит линию, заданную в форме

```
DRAW X,Y
```

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

Поэкспериментируйте с такими командами:

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

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

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

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

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

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

Если 'A' положительно, то дуга вычерчивается влево, а если отрицательно, то вправо. При 'A' равном 2*PI вычерчивается полная окружность. Например:

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

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

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

```
CIRCLE X,Y, радиус
```

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

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

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

выведет:

```
PAPER 7:INK 0
```

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

PLOT INVERSE 1 - устанавливает для заданного пикселя цвет фона,
PLOT OVER 1 - изменяет цвет пикселя на противоположный,если был цвет тона, то становиться цвет фона и наоборот.
PLOT INVERSE 1;OVER 1; - сохраняет цвет пикселя без изменения, но меняет текущую PLOT-позицию.

Другой пример использования OVER с записью черным по белому:

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

вычерчивает линию по диагонали.

Теперь попробуйте:

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

и перечертите ее командой

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

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

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

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

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

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

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

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

5.17 УКАЗАНИЯ

(18)

Краткое содержание: PAUSE, INKEY$, PEEK

Если вы решили задержать выполнение программы на некоторое время, то вам следует использовать оператор

PAUSE N ,

который останавливает выполнение программы и отображает картинку в течение 'N' телевизионных кадров (50 кадров в сек в Европе или 60 - в Америке). 'N' может быть вплоть до 65535, что составляет 22 минуты. Если N=0, то это означает, что оператор PAUSE не имеет ограничений по времени.

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

Пример программы моделирования секундной стрелки часов:

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

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

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

```
(65536*PEEK 23674 + 256*PEEK 23673 + PEEK 26672)/50
```

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

```
 10 REM FIRS WE DRAW THE CLOCK FACE
 20 FOR N=1 TO 12
 30 PRINT AT 10-10*COS(N/6*PI),16+10*SIN(N/6*PI);N
 40 NEXT N
 50 DEF FN T()=INT((65536*PEEK 23674+256*PEEK 23673+
          PEEK 23672)/50):REM NUMBER OF SECOND SINCE
                                 START
100 REM NOW WE START THE CLOCK
110 LET T1=FN T
120 LET A=T1/30:PI:REM A IS THE ANGLE OF THE SECOND
                       HAND IN RADIANS
130 LET SX=72*SIN A:LET SY=72*COS A
140 PLOT 131,91:DRAW OVER 1; SX,SY:REM DRAW HAND
200 LET T=FN T()
210 IF T<=T1 THEN GO TO 200:REM WAIT UNTIL TIME FOR
                                 NEXT HAND
220 PLOT 131,91:DRAW OVER 1; SX,SY:REM RUB OUT OLD HAND
230 LET T1=T:GO TO 120
```

Эти часы обеспечивают точность 0,01% или уход на 10 секунд в день. Однако, это возможно при условии, что вы не использовали оператор BEEP, ввод/вывод на магнитофон и принтер. Все эти операции увеличивают погрешность.

Числа PEEK 23674, PEEK 23673 и PEEK 23672 выделяют адреса ячеек памяти компьютера и используемых для подсчета 1/50 долей секунды. В каждой из ячеек подсчитывается сумма от 0 до 255, после достижения величины 255 в любой из ячеек она сбрасывается в 0. Первой начинает отсчитывать ячейка PEEK 23672. Каждую 1/50 секунды ее содержимое увеличивается на 1. Когда в ячейке накопится величина, равная 255, то она сбрасывается в 0, а значение ячейки PEEK 23673 увеличивается на 1. Через каждые 256/50 сек содержимое этой ячейки переходит из состояния 255 в 0, а содержимое ячейки PEEK 23674 увеличивается на 1.

При значениях 0 для ячейки PEEK 23674 и 255 для ячеек PEEK 23673 и PEEK 23672 (этот момент наступит через 21 минуту) наше выражение примет значение:

```
(65536*0 + 256*255 + 255)/50=1310.7
```

но здесь имеется скрытая опасность. Через следующую 1/50 сек ячейки будут содержать соответственно следующие значения: 1, 0, 0.

Пока производится вычисление выражения, компьютер может оценить значение ячейки PEEK 23674 как 0 до завершения циклического переноса. В результате получим:

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

что безнадежно неверно.

Простое правило позволяет решить эту проблему:

"Следует вычислять выражение дважды в некоторой последовательности и использовать сохраненный ответ".

Пример:

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

Вы можете изменять значения числовых счетчиков так, чтобы получать реальное время того момента, когда компьютер был включен. Например, надо установить 10 часов вечера. Вы посчитали, что это

```
10*60*60*50 = 1800000   50-х долей
```

секунды и значит

```
1800000 = 65536*27 + 256*119 + 64.
```

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

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

Функция INKEY$ (без аргументов) считывает с клавиатуры. Если вы нажали некоторую клавишу (или SHIFT и какую-нибудь клавишу), результатом будет символ, который дает эта клавиша в режиме курсора [L], или пустая строка.

Выполните программу, которая использует эту функцию:

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

Помните, что функция INKEY$ не будет подобно INPUT ждать вас. Если вы не выполните ввод, то считайте, что ваш шанс упущен.

5.5 ПРОГРАММИРОВАНИЕ ЗВУКОВ

Краткое содержание: BEEP.

ZX SPECTRUM может воспроизводить звуки при помощи оператора BEEP:

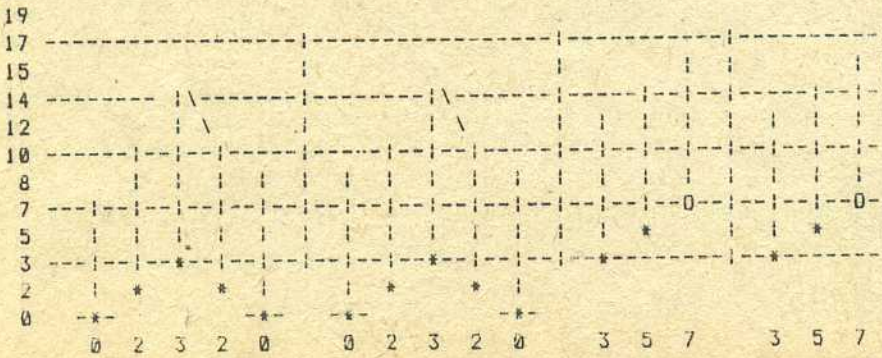
BEEP продолжительность, высота звука.

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

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

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

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



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

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

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

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

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

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

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

Фрагмент программы:

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

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

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

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

Некоторая музыка, например индийская, использует интервалы меньшие, чем полутон. Вы можете без особого труда запрограммировать это в операторе BEEP. Например, для звука на четверть тона выше среднего ДО надо указать значение высоты звука, равное 0.5.

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

```
POKE 23609,255
```

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

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

5.19 ВНЕШНЯЯ ПАМЯТЬ НА МАГНИТНОЙ ЛЕНТЕ (20)

Краткое содержание: LOAD, SAVE, VERIFY, MERGE

Основные команды работы с магнитофоном SAVE, LOAD и VERIFY уже рассматривались во вводном описании. Вы могли видеть, что LOAD затирает старую программу в памяти компьютера при загрузке новой программы с ленты. Есть другая команда MERGE, не делающая этого. Эта команда стирает лишь те строки старой программы или переменные, которые совпадают с номерами строк новой программы или именами новых переменных.

Теперь вы знаете четыре оператора для работы с кассетными магнитофоном:

SAVE - записывает программу и переменные на магнитофон;
VERIFY - проверяет программу и переменные в памяти компьютера по их копии на ленте;
LOAD - очищает память компьютера от всех программ и загружает в нее новые, считанные с магнитофона;
MERGE - подобна LOAD, только не очищает всю память, а лишь заменяет те строки программы или переменные, у которых совпадают номера или имена с такими же на магнитной ленте.

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

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

Вариант 2. С использованием оператора SAVE:

```
SAVE STRING LINE NUMBER
```

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

Кроме текстов программ, на ленту можно записывать также массивы или данные.

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

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

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

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

Нельзя использовать оператор MERGE.

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

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

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

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

LOAD "PICTURE" CODE

после CODE можно указать числа:

LOAD "PICTURE" CODE START, LENGTH

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

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

Выражение CODE 16384,6912 можно заменить на SCREEN$:

SAVE "PICTURE" SCREEN$ и затем

LOAD "PICTURE" SCREEN$

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

В заключение. Везде, где указывается имя файла на ленте, используются только первые 10 символов. Существует четыре типа информации, которые могут быть записаны на ленту:

- программы и переменные (совместно);
- числовые массивы;
- строковые массивы;
- непосредственно байты.

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

"PROGRAM:", "NAMBER ARRAY:", "CHARACTER ARRAY:" или "BYTES:".

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

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

1. Программа и переменные.

SAVE NAME LINE NUMBER

записывает программу на ленту таким образом, что последующая команда LOAD автоматически вставляет в программу

GO TO LINE NUMBER

и начинает ее выполнять.

2. Байты.

SAVE NAME CODE START, LENGTH

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

SAVE NAME SCREEN$

эквивалентно

SAVE NAME CODE 16384,6912

и записывает один телевизионный экран.

3. Массивы.

SAVE NAME DATA LETTER() или

SAVE NAME DATA LETTER$()

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

Команда VERIFY проверяет (сравнивает) информацию в памяти и на ленте. Может выдавать сообщение

"R TAPE LOADING ERROR".

Запись SAVE „A" ENTER

Ввод LOAD „" ENT
RUN ENT

1. Программа и переменные.

VERIFY NAME

2. Байты.

VERIFY NAME CODE START, LENGTH

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

VERIFY NAME CODE START

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

VERIFY NAME CODE

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

VERIFY NAME SCREEN$

или эквивалентно

VERIFY NAME CODE 16384, 6912

однако это будет проверка уже проверенного файла.

3. Массивы.

VERIFY NAME DATA LETTER()

VERIFY NAME DATA LETTER$()

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

1. Программа и переменные.

LOAD NAME

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

2.Байты.

LOAD NAME CODE START, LENGTH

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

LOAD NAME CODE START

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

LOAD NAME CODE

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

3. Массивы.

LOAD NAME DATA LETTER()

или

LOAD NAME DATA LETTER$()

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

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

1. Программа и переменные.

MERGE NAME

дописывает программу "NAME" к некоторой программе, находящейся в памяти. Может выдать сообщение

"4 OUT OF MEMORY".

2. Байты.

Не поддерживается.

3. Массивы.

Не поддерживается.

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

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

обратная загрузка

LOAD "CHESS" CODE

LOAD "CHESS" CODE USR "A" .

5.20

(21)

УСТРОЙСТВО ПЕЧАТИ

Краткое содержание: LPRINT, LLIST, COPY.

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

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

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

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

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

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

Теперь попробуйте выполнить такую программу:

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

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

Вообще, при печати, перевод строки осуществляется в следующих случаях:

а) При заполнении буфера строки;
б) После LPRINT, если это не конец оператора и в нем встретилась запятая или точка с запятой;
в) Если запятая, апостроф или TAB требуют новой строки;
г) При окончании программы, если остались невыведенные данные.

5.22 ВВОД И ВЫВОД (23)

Краткое содержание: OUT, IN

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

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

IN аналогичен оператору PEEK:

```
IN ADDRESS
```

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

OUT подобен оператору POKE:

```
OUT ADDRESS,VALUE
```

и записывает указанные данные в заданный порт вывода.

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

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

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

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

Биты A8, A9 используются иногда для получения дополнительной информации.

Информационный байт мы будем обозначать буквой D:

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

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

```
IN 65278 считывает ряд от CAPS SHIFT до V,
IN 65022 считывает ряд от          A до G,
IN 64510 считывает ряд от          Q до T,
IN 63486 считывает ряд от          1 до 5,
IN 61438 считывает ряд от          0 до 6,
IN 57342 считывает ряд от          P до 7,
IN 49150 считывает ряд от      ENTER до H,
IN 32766 считывает ряд от      SPACE до B.
```

Эти адреса могут быть вычислены из выражения:

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

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

Выходной порт 254 обслуживает громкоговоритель (D4) и разъем 'MIC' (D3), а так же установку цвета (D2,D1,D0).

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

Порты 254,247 и 239 используются для связи с дополнительными устройствами.

Запустите следующую программу:

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

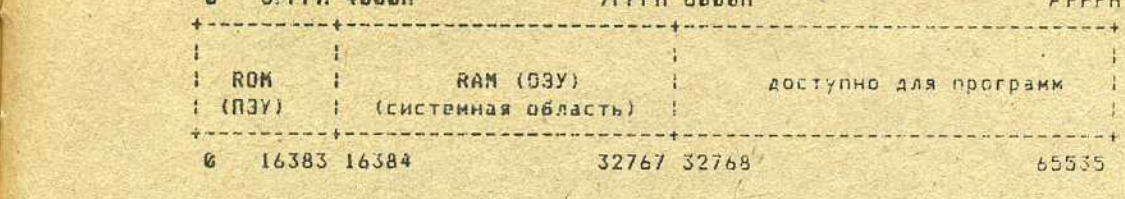
и понажимайте по одной клавише в каждом полуряду. После нажатия очередной клавиши введите BREAK, а затем NEXT N.

(24)

5.23 ПАМЯТЬ

Краткое содержание: CLEAR

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



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

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

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

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

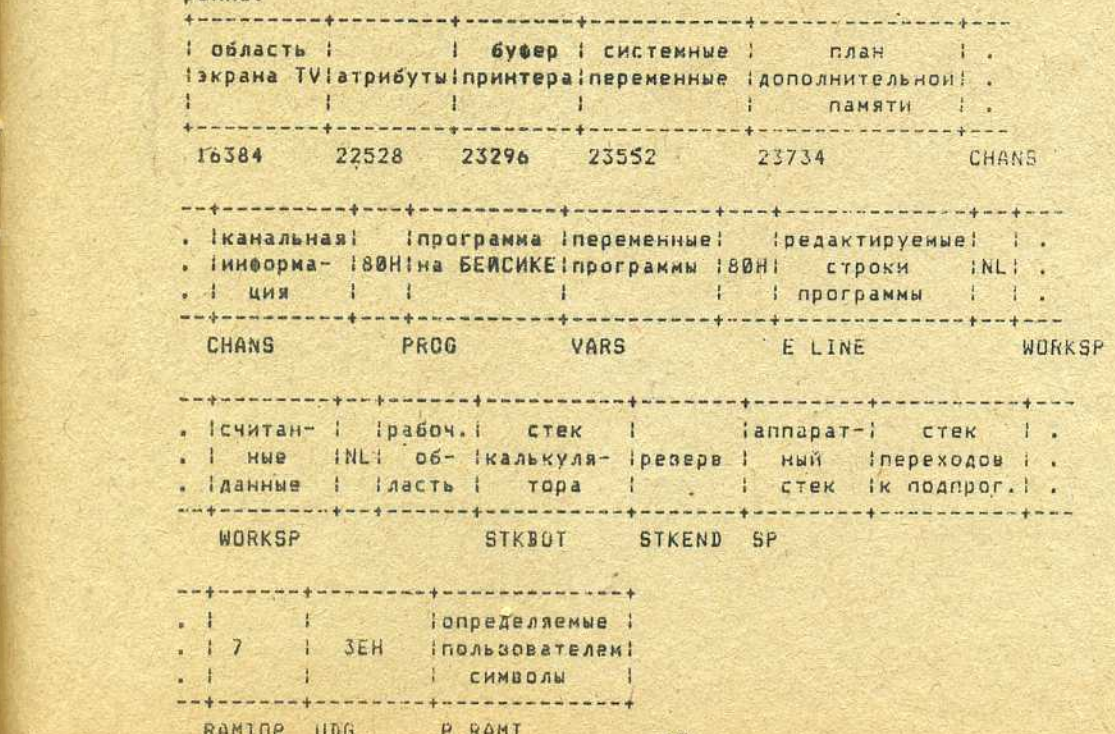
POKE ADDRESS, NEW CONTENTS ,

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

POKE 31000,57

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

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



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

Полный экран представляет собой 24 строки по 32 символа. Каждая строка экрана прописывается 8-мью строками развертки телевизионного экрана. Итого для записи одного экрана выполняется 172 сканирования, и в памяти рядом хранятся байты одноименных рядов, матриц соседних позиций экрана.

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

Буфер принтера содержит символы, передаваемые на печать.

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

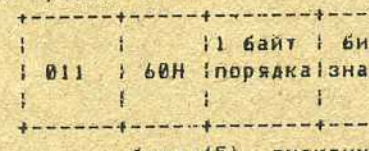
Область планов дополнительной памяти используется только с MICRODRIVE.

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

Каждая строка области БЕЙСИК-программ имеет формат:

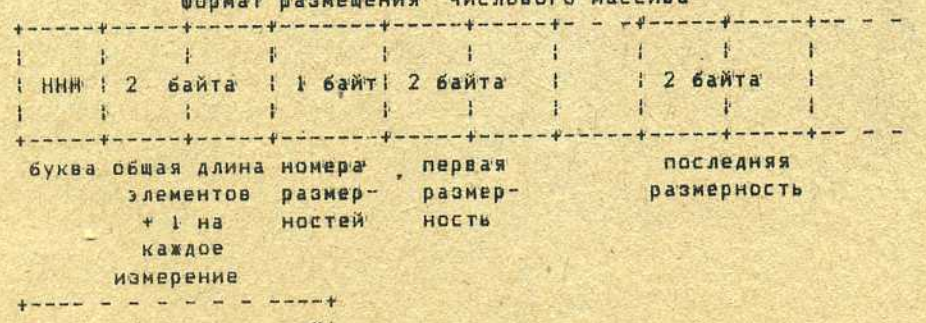
Числовые константы в программе представлены в двоичной форме, используя CHR$14 и следующие за ним 5 байт самого числа.

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



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

Формат размещения числового массива



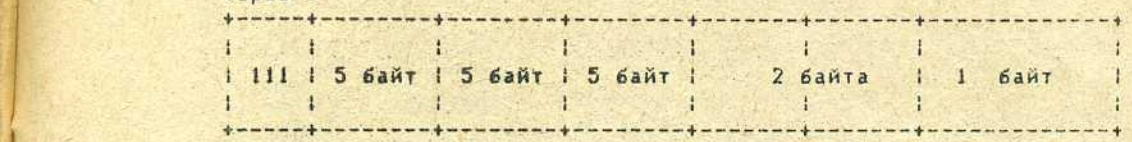
Порядок элементов следующий:

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

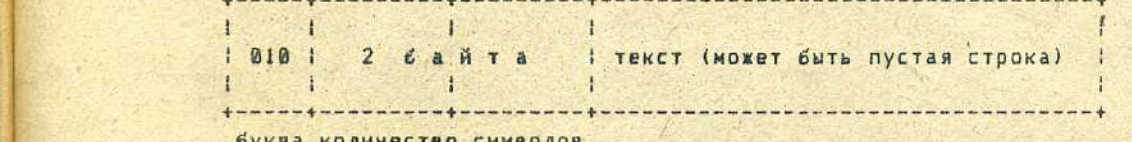
Затем в том же порядке по следующей размерности и т.д. Например, элементы массива с размерностью (3,6) расположатся в памяти в следующем порядке:

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

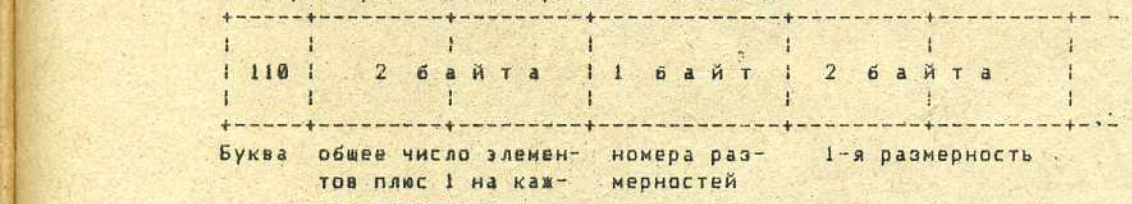
Формат размещения управляющих переменных для FOR-NEXT операторов :



Формат размещения строки символов:



Формат размещения строкового массива:



Программируемый стек - есть часть интерпретатора БЕЙСИКА. Аппаратный стек используется микропроцессором Z80 для запоминания адресов возврата.

Резерв в данной версии не используется.

Назначение стека переходов к подпрограммам описано в главе 5.

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

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

Вы можете изменить адрес RAMTOP в операторе CLEAR :

CLEAR NEW RAMTOP ,

по которому:

а) очищаются все области переменных;

б) очищается область экрана (подобно CLS);

в) переустанавливается позиция PLOT в левый нижний угол экрана;

г) выполняется функция RESTORE;

д) очищается стек переходов и устанавливается новое значение RAMTOP.

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

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

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

"4 MEMORY FULL" (переполнение памяти);

"6 NO ROOM FOR LINE" (нет места для строки программы).

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

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

+ M * 2 ** E ,

где M - мантисса в интервале 0.5 ... 1 (не может быть 1);

E - экспонента, положительное или отрицательное число.

Допустим вы записали "M" в двоичной системе счисления, "M" - дробное и имеет двоичную точку (подобно десятичной точке), то тогда будет:

1/2 ---> .1

1/4 ---> .01

3/4 ---> .11 и т.д.

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

Для записи числа мы используем 5 байтов в следующем порядке:

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

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

в) записываем в первый байт экспоненту +128.

Например, мы хотим записать число 1/10.

$$1/10 = 4/5 \times 2^{-3}$$

мантисса будет .1100110011001100110011001100110011001100B (поскольку 33-й бит равен 1, мы должны округлить 32-й бит, записав 1 вместо 0), а экспонента равна -3.

Теперь применив наших три правила, запишем 5 байт:

```
          ¦---знак числа
+---------+-+-------+--------+--------+--------+
¦         ¦ ¦       ¦        ¦        ¦        ¦
¦0111 1101¦0¦100 1100¦1100 1100¦1100 1100¦1100 1100¦
¦         ¦ ¦       ¦        ¦        ¦        ¦
+---------+-+-------+--------+--------+--------+
  -3+128     мантисса 4/5, исключая левый знаковый бит
```

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

а) первый байт равен 0;

б) второй байт равен 0 для положительного числа и FFH для отрицательного;

в) 3 и 4 байты содержат младшие и старшие значащие биты числа (или число +131072, если оно отрицательное);

г) 5 байт равен 0.

6. ИСПОЛЬЗОВАНИЕ ПРОГРАММ В МАШИННЫХ КОДАХ

ОС СПЕКТРУМА (как впрочем и бейсик) позволяет загружать, выгружать и запускать на выполнение программы в машинных кодах процессоров Z80 и К580ВМ80. При загрузке и выгрузке такие программы представленны файлами типа BYTE, являющимися полной аналогией файлов типа .COM компьютера IBM PC. Эти программы полностью резидентны (т.е. при работе целиком находятся в ОЗУ компьютера), что повышает скорость их работы практически до скорости работы процессора (950 000 операций в секунду). Это самые быстрые программы из всех возможных. Такие программы-результат компиляции программ на языках высокого уровня (Паскаль, СИ, Фортран, Бейсик, специальные языки); результат ассемблирования программ на языке ассемблера Z80; непосредственное написание программ в кодах с помощью специальных отладчиков или просто вручную. (Перечень мнемокодов команд микропроцессора Z80 приведен в приложении 1).

Эти программы запускаются на исполнение командой RANDOMISE USR ... где ... -адрес (десятичный), с которого программа запускается. Такая же команда выполняется и в бейсике.

Кроме того, для запуска таких программ в бейсике имеется функция USR, возвращающая содержимое регистровой пары BC процессора Z80 в виде целого положительного числа от 0 до 65 535. Все программы такого типа должны заканчиваться кодом процессора RET (возврат) 201 (шестнадцатеричный C9).

Примеры:

RANDOMISE USR 0 ..... запуск программы в кодах с адреса 0000 т.е. в ПЗУ. Такая команда полностью аналогична нажатию кнопки "сброс"

RANDOMISE USR 55000..... запуск программы в кодах с адреса 55000

LET A=USR 55000..... запуск программы в кодах с адреса 55000 и присвоение переменной "A" значения, оказавшегося в регистровой паре BC процессора в момент возврата.

PRINT USR 55000..... запуск программы в кодах и печать значения регистровой пары BC процессора.

RANDOMISE USR USR 55000..... очень интересная комбинация, запускается программа в кодах с адреса 55000, а затем запускается программа в кодах с адреса, возвращенного первой программой в регистровой паре BC процессора.

Приведем пример программы:

```
ID BC,99
RET
```

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

байты 1, 99, 0 для ID BC,99 и
201 для RET

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

```
- - ---------------+-------------------------+---------------+
                   |      определяемые       |               |
                   | пользователем символы    |               |
- - ---------------+-------------------------+---------------+
       RAMTOP=32599  UDG=32600                 PRAMT=32767
```

Если вы теперь выполните:

```
CLEAR 32499
```

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

```
- --+---------+-----------+-------------------------+---------+
    |   100   |           |       определяемые      |         |
    | байтов  |           | пользователем символы   |         |
- --+---------+-----------+-------------------------+---------+
RAMTOP=32499   32500       UDG=32600                 PRAMT=32767
```

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

```
10 LET A=32500
20 READ N: POKE A,N
30 LET A=A+1: GOTO 20
40 DATA 1,99,0,201
```

(программа может завершиться с сообщением 'E OUT OF DATA', если переполняются отведенные вами 4 байта).

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

Если вы выполните:

```
PRINT USR 32500
```

то получите ответ: 99

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

Вы можете записать вашу программу на ленту:

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

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

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

для чего надо сделать:

```
SAVE NAME LINE
```

а затем:

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

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

Программы в кодах не имеют атрибута самозапуска, из-за чего они не могут сами запуститься после загрузки, если не применять для этого особых программных ухищрений. Поэтому для их загрузки и запуска применяют маленькие программки-загрузчики на бейсике.

При использовании программ в кодах надо учитывать, что при их выполнении не производится контроль ошибок и может произойти зависание компьютера, устраняемое только кнопкой <сброс>

Написание программ в кодах - занятие весьма кропотливое, так как приходится помнить не только коды команд,но и их длину (1,2,3,4 байта), внимательно следить за адресами,в которых размещается программа и данные

Использование ассемблера позволяет получить столь же эффективные и быстрые программы,как и при написании их в кодах,но с гораздо меньшими затратами сил и времени.Одновременно обеспечивается возможность работы программы в любых адресах (при перетрансляции) и легкость ее модификации.

Принцип программирования на ассемблере состоит в том,что вместо кодов машинных команд вы пишите их условные обозначения -мнемоники,а вместо адресов -именные метки.Программа ассемблер превращает мнемоники в коды команд,а метки -в конкретные адреса этих команд.Такая операция называется ассемблированием,и в результате ее получается программа в кодах,используемая также,как и обычные программы в кодах.

Ассемблеры очень сильно зависят от типа процессора машины. В нашем случае необходим ассемблер для процессора Z80. Вот как выглядят одни и те же команды на ассемблерах для К580ВМ80А и для Z80:

К580ВМ80А	Z80	ДЕЙСТВИЕ
MOV B,C	LD B,C	B=C
LXI B,#4F	LD B,#4F	B=4F (HEX)
LDAX B	LDA,(BC)	A=содержимое яч.памяти BC
LALD #4FBF	LD HL,(#4FBF)	HL=содержимое яч. памяти #4FBF
SPHL	LD SP,HL	SP=HL
PUSH PSW	PUSH AF	спасение аккум.и флагов
PUSH B	PUSH BC	спасение рег.пары BC
ANI #4F	AND #4F	A=A & 4F (HEX)
ANA B	AND B	A=A & B
RPO	RET PO	возврат,если не плюс
RNC	RET NC	возврат,если не перенос
RET	RET	возврат безусловный
XCHG	EX DE,HL	обмен рег. пар DE и HL
ZTHL	EX (SP),HL	обмен рег. пары HL и стека

Простое сравнение показывает,что мнемоники Z80 при меньшем их количестве более понятны и наглядны.Однако,если вы привыкли програмировать на ассемблере К580ВМ80А,можете на нем и продолжать,так как программы в кодах К580ВМ80А всегда будут работать на процессоре Z80(но не наоборот) Правда при этом вы не используете всех возможностей вашего процессора.

Ниже приводятся 78 команд микропроцессора 8085 совместимых с микропроцессором Z80 (158 команд)./U880-ГДР,1810ВМ-СССР/

МНЕМОНИКА	ДЕЙСТВИЕ	КОД	ЦИКЛЫ
1	2		3
MOV R,R	ПЕРЕСЛАТЬ РЕГИСТР В РЕГИСТР	0.RR RRRR	4[7]
MVI R,D	ПЕРЕСЛАТЬ НЕПОСР. В РЕГИСТР	00RR R110	7[10]
LXI RP,DD	ЗАГРУЗИТЬ НЕПОСР. В ДВА РЕГ	00PP 0001	10
STAX B	ЗАПОМНИТЬ А КОСВЕННО ПО BC	0000 0010	7
STAX D	ЗАПОМНИТЬ А КОСВЕННО ПО DE	0001 0010	7
LDAX B	ЗАГРУЗИТЬ А КОСВЕННО ПО BC	0000 1010	7
LDAX D	ЗАГРУЗИТЬ А КОСВЕННО ПО DE	0001 1010	7
STA DD	ЗАПОМНИТЬ А ПО АДРЕСУ DD	0011 0010	13
LDA DD	ЗАГРУЗИТЬ А ПО АДРЕСУ DD	0011 1010	13
SHLD DD	ЗАПОМНИТЬ H,L ПО АДРЕСУ DD	0010 0010	16
LHLD DD	ЗАГРУЗИТЬ H,L ПО АДРЕСУ DD	0010 1010	16
XCHG	ОБМЕНЯТЬ DE И HL	1110 1011	4
ПРИРАЩЕНИЕ И УМЕНЬШЕНИЕ			
INR R	ПРИРАЩЕНИЕ РЕГИСТРА	00RR R100	4[11]
DCR R	УМЕНЬШЕНИЕ РЕГИСТРА	00RR R101	4[11]
INX RP	ПРИРАЩЕНИЕ ПАРЫ РЕГИСТРОВ	00PP 0011	6
DCX RP	УМЕНЬШЕНИЕ ПАРЫ РЕГИСТРОВ	00PP 1011	6

АРИФМЕТИЧЕСКИЕ И ЛОГИЧЕСКИЕ			
ADD R	ПРИБАВИТЬ РЕГИСТР К А	1000 0RRR	4[7]
ADC R	ПРИБАВИТЬ РЕГ. К А С ПЕРЕН.	1000 1RRR	4[7]
SUB R	ВЫЧЕСТЬ РЕГИСТР ИЗ А	1001 0RRR	4[7]
SBB R	ВЫЧЕСТЬ С ЗАЕМОМ	1001 1RRR	4[7]
ANA R	РЕГИСТР А	1010 0RRR	4[7]
XRA R	ИСКЛ. ИЛИ РЕИСТРА И А	1010 1RRR	4[7]
ORA R	РЕГИСТР А	1011 0RRR	4[7]
CMA	СРАВНИТЬ РЕГИСТР И А	1011 1RRR	4[7]
ADI D	ПРИБАВИТЬ НЕПОСР.ДАННЫЕ К А	1100 0110	7
ACI D	ПРИБАВИТЬ НЕПОСР. С ПЕРЕН.	1100 1110	7
SUI D	ВЫЧЕСТЬ НЕПОСР. ИЗ А	1101 0110	7
SBI D	ВЫЧЕСТЬ НЕПОСР. С ЗАЕМОМ	1101 1110	7
ANI D	НЕПОСР. А	1110 0110	7
XRI D	ИСКЛ. ИЛИ НЕПОСР. И А	1110 1110	7
ORI D	НЕПОСР. А	1111 0110	7
CPI D	СРАВНИТЬ НЕПОСР. С А	1111 1110	7
DAD RP	ПРИБАВИТЬ ПАРУ РЕГИСТРОВ К HL	00PP 1001	11
ОПЕРАЦИИ С НАКОПИТЕЛЯМИ И ФЛАГАМИ			
RLC	СДВИНУТЬ А ВЛЕВО	0000 0111	4
RRC	СДВИНУТЬ А ВПРАВО	0000 1111	4
RAL	СДВ-ТЬ А ВЛЕВО ЧЕР.РАЗР.ПЕР	0001 0111	4
RAR	СДВ-ТЬ А ВПРАВО ЧЕР.РАЗР.ПЕР	0001 1111	4
DAA	ДЕСЯТ.КОРРЕКЦИЯ НАКОПИТЕЛЯ	0010 0111	4
SMA	ДОПОЛНЕНИЕ К НАКОПИТЕЛЮ	0010 1111	4
STC	УСТАНОВИТЬ БИТ ПЕРЕНОСА	0011 0111	4
CMC	ОБРАТИТЬ БИТ ПЕРЕНОСА	0011 1111	4
I/O, УПРАВЛЕНИЕ И ОПЕРАЦИИ СО СТЕКОМ			
IN D	ВВОД ИЗ ПОРТА D	1101 1011	10
OUT D	ВЫВОД В ПОРТ D	1101 0011	11
EI	РАЗРЕШЕНИЕ ПРЕРЫВАНИЯ	1111 1011	4
DI	ЗАПРЕЩЕНИЕ ПРЕРЫВАНИЯ	1111 0011	4
NOP	НЕТ ОПЕРАЦИИ	0000 0000	4
HLT	ОСТАНОВ	0111 0110	4
PUCH RP	ЗАНЕСТИ ПАРУ РЕГИСТРОВ В СТЕК	11PP 0101	11
POP RP	ВЗЯТЬ ПАРУ РЕГИСТРОВ ИЗ СТЕКА	11PP 0101	10
XTHL	ОБМЕНЯТЬ HL С ВЕРХОМ СТЕКА	1110 0011	19
SPHL	ПЕРЕСЛАТЬ HL В SP	1111 1001	6
ПЕРЕДАЧИ УПРАВЛЕНИЯ			
JMP DD	БЕЗУСЛОВНЫЙ ПЕРЕХОД	1100 0011	10
JCC DD	ПЕРЕЙТИ ПО УСЛОВИЮ С	11CC C010	10
CALL DD	БЕЗУСЛОВНЫЙ ВЫЗОВ	1100 1101	17
CCC DD	ВЫЗОВ ПО УСЛОВИЮ СС	11CC C100	17(10)
RET	ВОЗВРАТ ПОСЛЕ ВЫЗОВА	1100 1001	10
RCC	ВОЗВРАТ ПО УСЛОВИЮ СС	11CC C000	11(5)
RST N	ВОЗОБНОВЛЕНИЕ В ЯЧЕЙКЕ 8*N	11NN N111	11
PCHL	ПЕРЕСЛАТЬ HL В PC	1110 1001	4

О Б О З Н А Ч Е Н И Я:

1. П О Л Я Д А Н Н Ы Х.

D - один байт непоср.данных (длина команды 2 байта)
DD- двухбитовый адрес (длина команды 3 байта)

2. Ц И К Л Ы.

N - число тактов, нужное для выполнения команды
[N] - число тактов, когда R=M (доступ в память)
(N) - число тактов, если условие не выполнено.

3. П О Л Я Р Е Г И С Г Р О В.

"R"	RRR
B	000
C	001
D	010
E	011
H	100
L	101
M	110
A	111

"RP"	PP
BC	00
DE	01
HL	10
PS	11
PSW	11

["M"=(HL)]

4. К О Д Ы У С Л О В И Й.

"CC"	CCC	УСЛОВИЕ
NZ	000	НЕ НУЛЬ
Z	001	НУЛЬ
NC	010	НЕТ ПЕРЕНОСА
C	011	ПЕРЕНОС
PO	100	НЕЧЕТНЫЙ ПАРИТЕТ
PE	101	ЧЕТНЫЙ ПАРИТЕТ
P	110	ПОЛОЖИТЕЛЬНОЕ
M	111	ОТРИЦАТЕЛЬНОЕ

7. ЗАЩИТА ПРОГРАММ

Защита программ состоит в исключении возможности просмотра текста программы и внесения в него изменений лицом, не уполномоченным на это автором программы.

Методов защиты программ существует очень много, и здесь мы рассмотрим только основные принципы и обозначим подходы к этой проблеме.

Можно выделить три основных направления:

1. Исключить возможность остановки (прерывания работы программы).
2. Исключить возможность подачи команды LIST для распечатывания текста программы.
3. Сделать листинг программы нечитаемым.

ИСКЛЮЧЕНИЕ ВОЗМОЖНОСТИ ОСТАНОВКИ ПРОГРАММЫ

Самое первое, что вы должны сделать - это выполнить программу автостартующей со строки номер N. Это выполняется при загрузке программы командой SAVE "имя" LINE N. Теперь программа после загрузки будет сама стартовать с указанной строки. Тем не менее она может быть остановлена командой BREAK или по сообщению об ошибке.

Отключить клавишу BREAK можно, если в той строке,с которой происходит автостарт, поместить команду POKE 23613, PEEK 23730-5.

Если программа вами хорошо отлажена, то остановку по ошибке можно исключить. Надо помнить, что чаще всего при взломе программы сознательно вносят ошибку во время исполнения оператора INPUT. Так, если программа просит от пользователя ввести какое-либо число, он сознательно нажимает на клавишу с буквой. Программа останавливается с сообщением VARIABLE NOT FOUND. Рекомендуем не использовать оператор INPUT, а организовывать опрос клавиатуры на основе функции INKEY$.

Если в строке автостарта поместить нижеприведенные команды, то при попытке сделать BREAK или при появлении ошибки программа будет сбрасываться:

```
LET ERR = 256*PEEK 23614 + PEEK 23613: POKE ERR,0: POKE ERR+1,0
```

Примерно тот же эффект дает помещение в стартовой строке команды POKE 23659,0.

Одним из способов остановки автостартующей программы является использование команды MERGE "" вместо LOAD "" при загрузке.

Если ваша программа имеет размер больше 7К, то защитить ее от этого приема можно, если перед выгрузкой подать следующие прямые команды:

```
LET X = 256*PEEK 23636 + PEEK 23635: POKE X,60: POKE X+1,0
```

Другой способ защиты от MERGE состоит в том, чтобы перед выгрузкой вашей программы на ленту дать ложную информацию о том, что якобы первая строка вашей программы имеет чрезмерно большую длину. Это делается прямой командой:

```
POKE (PEEK 23635 + 256*PEEK 23636 +3),255.
```

ОТКЛЮЧЕНИЕ КОМАНДЫ LIST

Введите первой строкой программы 1 REM. Перед выгрузкой программы на ленту дайте прямую команду:

```
POKE (PEEK 23635 + 256*PEEK 23636),100
```

Теперь программу можно запускать (RUN), но читать нельзя.

КАК СДЕЛАТЬ ТЕКСТ ПРОГРАММЫ НЕЧИТАЕМЫМ

Простейший прием, применяемый во многих программах состоит в том, что устанавливают одинаковым цвета символов (INK) и фона (PAPER).
Например: 10 INK 7: PAPER 7
В тех местах, где программа должна сделать вывод на экран в операторе PRINT вставляют в качестве временных правильные цвета:

```
20 PRINT INK 0; "ZX SPECTRUM"
```

Другой способ состоит в искажении набора символов путем задания "фальшивого" значения системной переменной CHARS. Попробуйте, например:

```
POKE 23606,8: PRINT "ZX SPECTRUM":POKE 23606,0
```

Если адресовать системную переменную CHARS в те области памяти, где вообще ничего нет, например POKE 23607,200, то все символы будут выглядеть как пробелы и на экране вообще ничего не будет. Вам же нужно перед всяким оператором PRINT включать его после PRINT.
Еще один регулярно используемый прием состоит в организации "нулевой" строки, которая реально исполняется программой, но не может быть удалена при редактировании. Чтобы первая строка стала нулевой, дайте команду POKE 23755,0: POKE 23756,0.

8. ОПЕРАЦИИ С ЭКРАНОМ

Типичной задачей для тех, кто начинает создавать собственные программы на "СПЕКТРУМе", является создание красочных изображений в графике высокого разрешения, хранение их в памяти компьютера и вывод на экран.
Мы не будем рассматривать графические операции с операторами DRAW, CIRCLE, PLOT, а также печать символов графики пользователя, а рассмотрим операции с экраном, созданном в графическом редакторе. Наиболее удобным в работе является графический редактор ARTSTUDIO, но можно использовать и какой-либо другой доступный.
Созданное в редакторе изображение вы можете выгрузить на ленту в виде блока кодов, длина которого 6912 байтов. Если теперь загрузить его командой

```
LOAD "" CODE 16384, 6912
```

то изображение сразу будет загружено на экран,но хранить его там, естественно, нельзя. Можно загрузить изображение в произвольное место и вызывать его оттуда на экран по мере необходимости.
Примем адрес для хранения картинки 30000.

```
LOAD "" CODE 30000, 6912
```

Переброску на экран можно организовать в БЕЙСИКе.

```
500 FOR I = 1 TO 6912
510 POKE 16383+I, PEEK 29999+I
520 NEXT I
530 RETURN
```

Всякий раз, когда вам надо вызвать изображение на экран, вы можете делать это командой GO SUB 500.

К сожалению этот метод очень медленно работает, что связано с ограниченными возможностями БЕЙСИКа. Построение экрана происходит более 20 секунд.

Возможность быстрой переброски экрана (менее 1 сек.) предоставляет программирование в машинном коде. Процессор Z-80 для этой цели имеет мощную команду перемещения блоков LDIR (ее код 237,176). Для ее работы необходимо, чтобы в регистре HL процессора находился адрес начала блока, подлежащего переброске (в нашем случае - 30000), в регистре BC - длина блока (6912), а в регистре DE - адрес места назначения (16384). Тогда программа в машинном коде будет выглядеть так:

МАШИННЫЙ КОД	АССЕМБЛЕР	КОММЕНТАРИЙ
33 48 117	LD HL, 48 117	Загрузить в регистр HL адрес 30000. Обратите внимание на то, что 48 + 117*256 = 30000.
1 0 27	LD BC, 0 27	Загрузить в регистр BC 6912.
17 0 64	LD DE, 0 64	Загрузить в регистр DE 16384.
237 176	LDIR	Выполнить переброску блока в новое место.
201	RET	Возврат в вызывающую программу.

Таким образом, программа в машинном коде выглядит так:
33, 48, 117, 1, 0, 27, 17, 0, 64, 237, 176, 201.

Поместим ее в произвольное место оперативной памяти, например начиная с адреса 65000 следующей программой, написанной на БЕЙСИКе.

```
10 FOR I=1 TO 12
20 READ A
30 POKE 59999+I,A
40 NEXT I
50 DATA 33,48,117,1,0,27,17,0,64,237,176,201
```

Теперь всякий раз, когда вам нужно поместить картинку на экран, вам надо стартовать эту процедуру, что выполняется командой RANDOMIZE USR 65000.

Если у вас в памяти компьютера хранится не одно, а несколько графических изображений, то их можно также перебрасывать на экран этой процедурой, надо только изменить адрес, поступающий в регисто процессора HL, что можно сделать командой POKE по адресу 65001 и по адресу 65002.

Предположим, что у вас есть вторая картинка, помещенная начиная с адреса 37000. Тогда INT (37000/256) = 144, а 37000-256*144 = 136. Выполним POKE 65001,136 и POKE 65002,144. Команда RANDOMIZE USR 65000 выполнит переброску второй картинки и т.д.

Очевидно, что поскольку каждый экран (включая атрибуты) занимает 6912 байтов , то одновременно в памяти нельзя хранить более 6 изображений, что во многих случаях бывает недостаточно. Если учесть, что каждое изображение имеет длительные последовательности нулей, то имеется возможность компрессировать каждое изображение.
Это позволяет во многих случаях вдвое сократить объем памяти, занимаемой графикой, а иногда даже больше.

Мы предлагаем вашему вниманию две процедуры в машинном коде. Первая процедура вызывается FNs и служит для компрессирования изображения с экрана и размещения его в заданном пользователем адресе. Она имеет два параметра h и l, которые задают адрес для хранения экрана. h - старший байт адреса, а l - младший байт.

Сама процедура FNs расположена начиная с адреса 56600. Длина - 60 байтов.

Вторая процедура FNt служит для декомпрессирования изображения и переброски его на экран. Она имеет два тех же параметра h,l Адрес размещения этой процедуры - 56500. Длина - 40 байтов.

Чтобы создать процедуру FNs, вам надо набрать и запустить следующую БЕЙСИК-программу. После того, как она отработает, выгрузите созданную процедуру на ленту командой SAVE "FNs" CODE 56600,60 и в дальнейшем можете использовать в своих программах. Точно так же исполняется копия процедуры FNt: SAVE "FNt" CODE 56500,40.

ПРОЦЕДУРА FNs

```
8300 LET b=56600: LET l=60: LET z=0: RESTORE 8310
8301 FOR i=0 TO l-1: READ a
8302 POKE (b+i),a: LET z=z+a
8303 NEXT i
8304 LET z=INT(((z/l)-INT(z/l))*l)
8305 READ a: IF a<>z THEN PRINT "???", STOP
8310 DATA 42, 11, 92, 1, 4
8311 DATA 0, 9, 86, 14, 8
8312 DATA 9, 94, 237, 83, 82
8313 DATA 221, 33, 0, 64, 6
8314 DATA 1, 126, 44, 32, 8
8315 DATA 36, 245, 124, 254, 91
8316 DATA 40, 16, 241, 78, 185
8317 DATA 32, 4, 4, 32, 238
8318 DATA 5, 18, 19, 120, 18
8319 DATA 19, 24, 227, 241, 18
8320 DATA 19, 120, 18, 237, 83
8321 DATA 22, 221, 201, 0, 0
8322 DATA 56, 0, 0, 0, 0
```

После того, как процедура выполнит компрессирование экрана и перебросит его в место, отведенное для длительного хранения, вы можете узнать адрес, в который можно помещать следующий экран. PRINT PEEK 23296 + 256*PEEK 232297. Но помните, что делать это можно только немедленно после окончания компрессирования и переброски блока.

ПРОЦЕДУРА FNt

```
8350 LET b = 56500: LET l=35: LET z=0: RESTORE 8360
8351 FOR i=0 TO l-1 : READ a
8352 POKE (b+i),a: LET z=z+a
8353 NEXT i
8354 LET z=INT(((z/l)-INT(z/l))*l)
8355 READ a: IF a<>z THEN PRINT "???": STOP
8360 DATA 42, 11, 92, 1, 4
8361 DATA 0, 9, 86, 14, 8
8362 DATA 9, 94, 33, 0, 64
8363 DATA 26, 245, 19, 26, 19
8364 DATA 71, 241, 119, 35, 16
8365 DATA 252, 124, 254, 91, 32
8366 DATA 240, 201, 0, 0, 0
8367 DATA 28, 0, 0, 0, 0
```

ПРИМЕР ПРИМЕНЕНИЯ ПРОЦЕДУР КОМПРЕССИИ И ДЕКОМПРЕССИИ

Предположим, что вам надо компрессировать и отправить на хранение три экрана scr1, scr2, scr3, причем первый из них идет в адрес 30000 (48, 117).

Тогда основная программа может выглядеть так:

```
10  DEF FN s(h,l) = USR 56600
20  DEF FN t(h,l) = USR 56500
100 LET a=23296: LET b=23297
110 LET h1=117: LET l1=4897
120 LET ad=l1+(h1*256)
130 PRINT ab: LOAD "scr1"CODE 16384,6912
140 RANDOMIZE FNs(h1,l1)
150 LET l2 = PEEK a: LET h2 = PEEK b
155 LET ab = l2+h2*256
160 PRINT ad: LOAD "scr2"CODE 16384,6912
170 RANDOMIZE FNs(h2,l2)
180 LET l3=PEEK a: LET h3=PEEK b
185 LET ad=l3+256*h3
190 PRINT ad: LOAD "scr3"CODE 16384,6912
200 RANDOMIZE FNs(h3,l3)
210 PAUSE 50
300 RANDOMIZE FNt(h1,l1)
310 PAUSE 50
320 RANDOMIZE FNt(h2,l2)
330 PAUSE 50
340 RANDOMIZE FNt(h3,l3)
350 PAUSE 50: GO TO 300
```

9. МЕТОДЫ ВВЕДЕНИЯ РУССКОГО ШРИФТА

Обеспечение возможности работы компьютера с символами русского алфавита является, наверное, одной из первейших задач, которые возникают перед тем, кто хочет сам писать программы на СПЕКТРУМе.

Существуют два принципиально различных подхода для решения этой задачи. Первый подход состоит в использовании символов графики пользователя, а второй - в создании нового набора символов, размещении его в оперативной памяти и назначении его в качестве действующего с помощью системной переменной CHARS.

ИСПОЛЬЗОВАНИЕ ГРАФИКИ ПОЛЬЗОВАТЕЛЯ

В графическом режиме вы можете, во-первых, набирать графические символы блочной графики, расположенные на цифровых клавишах, а также можете воспользоваться символами графики пользователя. Их может быть до 21; они берутся нажатием на клавиши от "A" до "U" в графическом режиме (курсор "G"). Изображение символов графики пользователя должны быть сформированы в специальной области памяти, на которую указывает системная переменная UDG. Адрес, в котором расположено изображение символа (8 байтов), может быть получен с помощью функции USR. Таким образом, заслав, начиная с адреса USR "A", 8 байтов, формирующих изображение скажем буквы Я, мы будем иметь возможность нажатием клавиши "A" в графическом режиме печатать на экране букву "Я".

Учитывая, что многие буквы латинского и русского алфавита имеют сходное начертание (например O, A, X, H, P, и др.), можно задать прописные буквы, не имеющие аналогов и при этом уложиться в разрешенный 21 символ.

Изображение буквы "Я" формируется из точек экрана (пикселей) например так:

```
00000000   0
00xxxxx0   0*128+0*64+1*32+1*16+1*8+1*4+1*2+0*1 = 62
0x0000x0   0*128+1*64+0*32+0*16+0*8+0*4+1*2+0*1 = 66
0x0000x0   0*128+1*64+0*32+0*16+0*8+0*4+1*2+0*1 = 66
00xxxxx0   0*128+0*64+1*32+1*16+1*8+1*4+1*2+0*1 = 62
00x000x0   0*128+0*64+1*32+0*16+0*8+0*4+1*2+0*1 = 34
0x0000x0   0*128+1*64+0*32+0*16+0*8+0*4+1*2+0*1 = 66
00000000   0
```

Таким образом, в результате такого синтеза шаблона буквы "Я", мы можем представить ее последовательностью из 8 байтов, разместив их в соответствующем месте-0, 62, 66, 66, 62, 34, 66,0.

Точно так же можно изобразить и остальные желаемые символы. Для тех, кто не хочет глубоко вдаваться в суть процесса, мы предлагаем небольшую готовую программу, которую вы можете набрать и выгрузить на ленту. Всякий раз, когда вы хотите составить какую-либо программу, в которой возможна необходимость в русских буквах, загрузите эту программу и стартуйте ее (RUN). После остановки программы дайте LIST и продолжайте со строки 100 набирать свою программу.

```
10 FOR z=1 TO 20
20 READ A$
30 FOR X=0 TO 7
40 READ Y
50 POKE USR A$ + X,Y
60 NEXT X
70 NEXT Z
80 DATA "B",0,126,64,124,66,66,124,0
81 DATA "G",0,126,64,64,64,64,64,0
```

```
82 DATA "D",0,28,36,36,36,36,126,66
83 DATA "J",0,65,73,62,73,73,65,0
84 DATA "I",0,66,70,74,82,98,66,0
85 DATA "L",0,30,34,34,34,34,98,0
86 DATA "P",0,126,66,66,66,66,66,0
87 DATA "O",0,66,66,36,24,16,96,0
88 DATA "C",0,68,68,68,68,68,126,2
89 DATA "H",0,66,66,66,126,2,2,0
90 DATA "N",0,65,73,73,73,73,127,0
91 DATA "M",0,65,73,73,73,73,127,1
92 DATA "E",0,60,66,30,2,66,60,0
93 DATA "U",0,76,82,114,82,82,76,0
94 DATA "A",0,62,66,66,62,34,66,0
95 DATA "S",24,66,70,74,82,98,66,0
96 DATA "R",0,64,64,124,66,66,124,0
97 DATA "T",0,192,64,124,66,66,124,0
98 DATA "F",0,127,73,73,73,127,8,0
99 DATA "Q",0,66,66,114,74,74,114,0
```

В результате работы этой программы устанавливается следующее соответствие между клавишами в графическом режиме и символами русского алфавита:

"Q" - "Ы"	"R" - "Ь"	"U" - "Ю"	"O" - "У"	"A" - "Я"
"D" - "Д"	"G" - "Г"	"J" - "О"	"C" - "Ц"	"N" - "Ш"
"E" - "Э"	"T" - тв.знак	"I" - "И"	"P" - "З"	"B" - "И"
"F" - "Ф"	"H" - "Ч"	"L" - "Л"	"D" - "Б"	"M" - "Щ"

ИЗМЕНЕНИЕ НАБОРА ЗНАКОВ

Набор знаков компьютеров "СПЕКТРУМ" состоит из 96 символов и занимает 96*8=768 байтов. Стандартный набор находится в ПЗУ по адресу 15616. Этот адрес можно определить с помощью системной переменной CHARS.

PRINT PEEK 23606 +256*PEEK23607+256 <ENTER> 15616 O.K.

При этом по адресу 23606 находится 0, а по адресу 23607 - 60. Изменение набора знаков состоит в том, чтобы сформировать с помощью какой-либо специализированной программы или вручную желаемый набор знаков, поместить его в любое свободное место оперативной памяти и изменить содержимое ячеек 23606 и 23607, чтобы оно указывало на адрес места расположения нового набора. Рассмотрим эти процедуры на конкретном примере.

1. Для формирования набора знаков очень удобно использовать графический редактор "ARTSTUDIO". Эта программа не только является идеальным графическим редактором, позволяющим легко выполнять графические изображения, но и позволяет конструировать шрифты. Выгрузим сформированный шрифт на ленту.

2. Допустим, мы хотим разместить новый шрифт, начиная с адреса 30000. Загрузим его с ленты LOAD ""CODE 30000,768.

3. Системная переменная CHARS должна указывать на адрес, находящийся на 256 байтов ниже, т.е. на 30000-256=29744.

4. Определим старший байт двухбайтной переменной CHARS.
D2 - INT (29744/256)=116.

5. Определим младший байт: B1 - 29744 - 1169*256 = 48.

6. Переключение на печать сформированным вами шрифтом выполняется командой POKE 23606,48: POKE 23607,116. Обратное переключение на латинский шрифт - командой POKE 23606,0: POKE 23607,60.

10 СОСТАВЛЕНИЕ И РЕДАКТИРОВАНИЕ ПРОГРАММ

Компьютер "СПЕКТРУМ" может работать в двух режимах. Это режим непосредственного исполнения команд и программный режим.

В режиме непосредственного исполнения команды выполняются после нажатия клавиши ENTER и после исполнения в памяти не хранятся. Если вам надо повторить команду, ее придется снова набрать с клавиатуры.

Например: PRINT 2*2 <ENTER> 4 O.K. - лишний раз убедитесь, что дважды два - четыре. Сообщение O.K. (о'кей) говорит о том, что команда выполнена и все в порядке. Команды, подаваемые в режиме непосредственного исполнения, называют прямыми командами. Прямая команда может состоять и из нескольких команд. В этом случае они должны разделяться двоеточием. Например, для создания таблицы квадратов целых чисел от 1 до 20 дайте прямую команду:

```
FOR i=1 TO 20 : PRINT i, i*i: NEXT i
```

В программном режиме команды записываются в виде строк программы. Каждая строка начинается с номера. Строки в программе размещаются в порядке возрастания номеров, поэтому если вы в программе после строки номер 10 наберете строку номер 5, то она встанет в программе до строки номер 10. Ввод строки в программу завершается нажатием ENTER, а начать работу программы можно прямой командой RUN или RUN N, где N - номер строки, с которой вы хотите чтобы программа начала работу. Можно запустить программу с нужной строки и прямой командой GO TO N.

Обычно строки программы нумеруют через десяток - 10, 20, 30... Это делают для того, чтобы в случае необходимости можно было бы легко вставить пропущенную строку. Так, если вам понадобилось бы вставить строку между строкой 1 и строкой 2, вам пришлось бы их переделать, а чтобы вставить строку между строкой 10 и строкой 20 вам достаточно присвоить ей номер от 11 до 19, например 15.

В программе для "СПЕКТРУМа" вы можете использовать любые номера строк от 1 до 9999. В одной строке могут быть записаны несколько команд или операторов, но они должны разделяться двоеточием.

Если вы наберете строку с некоторым номером N, а строка с таким номером у вас уже есть, то после нажатия ENTER новая строка встанет на свое место в программе, а старая будет уничтожена полностью. Это дает эффективный метод стирания ненужных строк. Для этого достаточно набрать номер строки, подлежащей уничтожению, и нажать ENTER.

Программа для печати квадратов целых чисел от 1 до 20 будет выглядеть так:

```
10 FOR i=1 TO 20
20 PRINT i, i*i
30 NEXT i
```

Запускается программа командой RUN. После того как программа отработает, она, в отличие от прямой команды, остается в памяти и может быть повторена сколько угодно раз. Прервать исполнение работающей программы можно командой BREAK. Если вам надо внести изменения в программу, то надо ее остановить, затем командой LIST или LIST N вывести на экран текст программы, а затем выполнить редактирование.

СПЕКТРУМ имеет встроенный строчный программный редактор. Это означает, что редактировать можно каждую строку отдельно.

При редактировании используют специальные клавиши - клавиши перемещения курсора и некоторые другие:

стрелка вверх	(CAPS + 6)	- программный курсор на строку вверх
стрелка вниз	(CAPS + 7)	- программный курсор на строку вниз
стрелка влево	(CAPS + 5)	- курсор редактирования на символ влево
стрелка вправо	(CAPS + 8)	- курсор редактирования на символ вправо
EDIT	(CAPS + 1)	- вызвать строку, на которую указывает программный курсор, на редактирование
DELETE	(CAPS + 0)	- стереть символ слева от курсора редактирования

Все эти операции выполняются сразу при нажатии на клавишу.

Для того, чтобы редактировать (переделывать) строку, надо сначала вызвать ее в позицию редактирования, которая расположена в нижней части экрана. Это делается так. Предположим, что вам надо устранить ошибку в работающей программе. Прервите ее нажатием BREAK. Дайте команду LIST для печати текста на экране. Если программа достаточно длинная и на экране целиком не помещается, то после того, как экран будет заполнен, распечатка прекратится и появится запрос SCROLL? Если строка, которую вы хотите переделать, есть на экране, нажмите в ответ на запрос N (нет) или BREAK, а если ее нет, то любую другую клавишу и распечатка текста программы на экране будет продолжена до получения очередного запроса SCROLL?

Когда нужная строка найдена и вы увидели ее номер, вы должны подвести к ней курсор. Этот курсор имеет вид стрелки ">" и перемещается курсорными клавишами. CAPS SHIFT 6 - вниз и CAPS SHIFT 7 - вверх. Если программа длинная, то перегонять курсор из одного места в другое довольно утомительно. В этом случае лучше сразу обратиться по номеру строки N командой LIST N, а в ответ на запрос SCROLL? нажать BREAK. После этого курсор ">"сразу будет установлен против нужной строки.

Теперь надо эту строку вызвать в позицию редактирования. Это делается ключевым словом EDIT (CAPS SHIFT 1). Строка будет повторена в нижней части экрана. Здесь можно вносить в строку изменения. Чтобы удалить из строки ненужные символы, установите курсор в строке с помощью курсорных клавиш справа от символа, подлежащего удалению, и нажмите DELETE (CAPS SHIFT 0). Курсорные клавиши в данном случае - CAPS SHIFT 5 - влево; CAPS SHIFT 8 - курсор вправо. Ввод новых символов и ключевых слов производится по месту установки курсора. После того как редактирование строки закончено, нажмите ENTER и, если отредактированная строка не содержит синтаксических ошибок, она займет свое место в программе.

Процесс редактирования упрощается несколько за счет того, что если после остановки программы нажать не LIST, а сразу ENTER, то на экране появляется текст программы, причем именно в том фрагменте, который последним редактировался или вводился. Это сокращает время на поиск нужной строки.

Конечно, стандартный редактор СПЕКТРУМа очень удобным не назовешь, но к нему быстро привыкнуть. Кроме этого, существуют другие версии БЕЙСИКа, например БЕТА БЕЙСИК (версии 1.0;1.8;3.0) Эта программа загружается с кассеты. Она предоставляет неплохие возможности для эффективного редактирования программ. Например, она имеет автонумерацию строк, возможность перенумерации строк, команду EDIT N для немедленного редактирования строки N, возможность быстрой переброски курсора, отладочные операторы, дополнительные операторы организации цикла (как в ПАСКАЛе), возможность работы с "окнами", развитые средства структурного про-

граммирования (процедуры) и еще несколько десятков удобных функций и команд. Опыт показывает, что при программировании на БЕЙСИКе эта программа сокращает время примерно в два раза на малых программах (за счет средств редактирования) и примерно в три-четыре раза на больших (более 200 строк) программах за счет средств структурного программирования и отладочных операторов.

Программа BETA BASIC занимает определенное место в оперативной памяти, но во-первых это критично только для очень больших программ, а во-вторых именно для них эта программа и предоставляет наибольшие возможности по экономии памяти за счет структурного программирования. Поэтому занятая память на больших программах с лихвой окупается, а на малых не имеет значения.

В качестве примера наберите небольшую программу, предназначенную для копирования программ с одного магнитофона на другой через компьютер.

```
10 DIM a(12)
20 FOR i=1 TO 12
30 READ a(i)
40 POKE (29999+i), a(i)
50 NEXT i
60 DATA 243, 175, 219, 254, 31, 31, 31, 211, 254, 195, 50, 117
70 RANDOMIZE USR 30000
```

В принципе эта программа написана в машинном коде. БЕЙСИК служит только для того, чтобы организовать этот код по адресу 30000 и стартовать его.

Некоторые советы:

Если вам надо создать несколько одинаковых (или похожих) строк программы, не спешите их набирать на клавишах. Гораздо проще набить одну строку, вызвать ее на редактирование, поправить; если нужно, и, стерев старый номер строки, поставить новый и нажать ENTER. При этом старая строка остается без изменения и появится новая строка с новым номером. Повторив это действие несколько раз, вы получите нужное количество строк.

Если вы хотите стереть строку программы целиком, достаточно набрать ее номер и нажать ENTER.

Если вы, редактируя строку, ее испортили, пока вы не нажали ENTER, нажмите клавишу EDIT.

Если ваша программа стала очень большой и долго двигать по ней курсором, воспользуйтесь директивой LIST...

СИНТЕЗАТОР ЗВУКОВЫХ ЭФФЕКТОВ

Ваш компьютер обладает встроенной системой синтезатора звуковых эффектов, позволяющей получать различные звуки. В языке БЕЙСИК этим синтезатором управляет всего одна команда BEEP, позволяющая подавать звуковые сигналы и исполнять несложные мелодии.

Если вы увлекаетесь музыкой, есть множество различных музыкальных программ. Например, программа WHAM, MUSIC BOX позволяет исполнять различные произведения в разложении на два канала тонального сигнала и ритм-бокс, с изображением партитуры на нотном стане. Каждый из каналов имеет диапазон 4,5 октавы и свою нотную партитуру. Ритм-бокс имеет три перестраиваемых эффекта и автоподстраивающийся барабан. Партитура запоминается, редактируется, записывается на магнитофон и загружается с него.

После отладки партитуру встроенным компилятором можно откомпилировать, превратив в программу в кодах, которую можно уже использовать как составную часть любой вашей программы.

Если возможности ритм-бокса WHAM! вам недостаточны, имеется более мощный специализированный ритм-бокс фирмы EINSTEIN SOFTWARE. Эта программа имеет уже 10 ударных инструментов, используемых в различных комбинациях, и запоминает до 10 ритмов произвольного размера и длины. Имеется 10 готовых ритмов. Ритмы можно переключать прямо в процессе исполнения.

Есть синтезаторы речи -LMOWA или TUKER, позволяющие компьютеру произносить слова и фразы на русском, польском или английском языках.

12. В С Т Р О Е Н Н Ы Й Т А Й М Е Р

Ваш компьютер обладает широкими возможностями по управлению различными процессами и объектами в реальном масштабе времени.

Скорость реакции вашего компьютера весьма велика и позволяет, например, управлять небольшим промышленным роботом с электроприводом или же программно имитировать работу контроллеров таких устройств, как графический принтер или графопостроитель. В случае графического принтера компьютер одновременно управляет двумя электродвигателями, одним шаговым двигателем и игольчатой головкой и имеет еще значительный запас по скорости, что позволяет ему одновременно с печатью еще и оптимизировать процесс печати, программно корректировать люфты печатающего устройства и регулировать контрастность печати.

Для работы в реальном масштабе времени часто необходим датчик времени -таймер, позволяющий отсчитывать временные интервалы.

Техническая характеристика таймера:

Дискретность отсчета времени .....0,02 сек. (50 Гц)
Режим работы .....маскируемое прерывание
Стабилизация частоты .....кварцевым генератором
Максимальный фиксируемый интервал .....328 000 секунд
Программная установка .....есть
Программная блокировка .....есть

Этот таймер используется ОС для обслуживания клавиатуры, что не мешает использовать его вам -только помните, что блокировка таймера приводит к отключению клавиатуры от ОС. Однако сама клавиатура остается работоспособной и может использоваться вашими программами (при этом к ней надо обращаться как к портам ввода). При блокировке таймера его показания не сбрасываются, но фиксируются -фактически таймер продолжает работать, но подсчет временного интервала прекращается. При снятии блокировки это позволит вам продолжить подсчет временного интервала с того места, на котором вы его прервали.

Для блокировки таймера достаточно всего лишь запретить процессору прерывания - и таймер остановится. Разрешив прерывания, вы разблокируете таймер.

13 С И С Т Е М Н Ы Е П Р О Г Р А М М Ы

С некоторыми из таких программ вы уже познакомились в разделах синтезатор звуковых эффектов и програмирование на ассемблере.На самом деле таких программ очень много,как и возможностей для их применения.

Языки програмирования :

Бейсик-интерпретатор	BETA-B.48 , BETA 3.1 , LASER-BASIC
Бейсик-компиляторы	F-COMPILER , FP 48 K V1.7 ,TOBOS-FP
Паскаль	PASCAL HP-80 ,HP-4 TM 8 ,HP-4 TM 16
Си	HISOFY C
Форт	FIG-FORTH, ED 50 , EP 50
Лого	LOGO
Пролог	PROLOG
Ассемблер	GENS-4 , ZEUS , ASSEMBLER , EDI 1 GS
Отладчик кодов	MONS-4 , MONITOR 48 ,MONITOR 16 , IR 48K

Базы данных :

Картотеки	MASTERFILE V 09 , TOOLKIT
Электронные тарлицы	OMNYCALK

Текстовые экранные редакторы :

TASWORD TWO , TASWORD 3 , TASWORD 3 , TASWORD CS , TAS.RUS , FILES THE LAST word 2

Графические редакторы :

ART STUDIO , ARTIST , DINAMIC-3 , M.DRAW , LP MK 2.3

Все графические редакторы ,кроме последнего,работают с клавиатурой, джойстиком или манипулятором "мышь". Последний графический редактор работает со световым пером.

Имитаторы дисковой операционной системы:

RAM DOS 2

Тесты системы :

TEST-PROG ,TEST BAS , MEMORY TEST , TST 80.V3

Трассировщики печатных плат :

PLATA V.3 , ТРАССА-3

Спектроанализаторы и цифровые осцилографы :

USCILLATOR ,TAPE'R , DIAG , TAPE HEAD

Математические программы :

FUN F(X,Y) , STATIST ,FOURIER

Редакторы знакогенератора пользователя :

UDG DEFINER , PAINTBOX ,UDG

Синтез речи :

FONGEN , LMOWA , SPEAKER , SPEAKEASY

Пакеты для Бейсика:

ZXED , COMPRESSOR , 64*32 , RUS , RENUM , 64 COLOMNY

14 С Т Р У К Т У Р А К О М П Ь Ю Т Е Р Н О Й И Г Р Ы

С точки зрения пользователя ,компьютерная игра представляет собой несколько файлов,записанных на кассете для магнитофона.Типы,размеры и количество файлов варьируется,однако можно выделить три основных типа структур:

загрузчик на Бейсике - блок (блоки) кодов

загрузчик на Бейсике - загрузчик в кодах -блок (блоки) кодов

программа на Бейсике со встроенными кодами

Примером первой структуры может служить игра LUNA CRABS ,второй - SILENT SERVICE ,третьей -игра WHEELIE 2.

Наиболее общим является второй тип структуры -его и рассмотрим подробнее.Начнем с первого файла типа PROGRAMM.Если его загрузить не командой LOAD "",а командой MERGE "" ,можно увидеть примерно следующую программу -загрузчик на Бейсике :

0 ... : LOAD "..." SCREEN$: LOAD "..." CODE : RANDOMISE USR ...

Здесь команда LOAD "..." SCREEN$ загружает на экран картинку-заставку, рассматривая которую вам будет легче ждать конца загрузки всего остального.Следующая команда LOAD "..." CODE загружает загрузчик в кодах, и последняя команда RANDOMISE USR ... его запускает.

Как видете,достаточно загрузить такую программу (она,как правило, имеет атрибут самозапуска и сама начинает работать),и все остальные файлы будут загружены и запущены без всякого вмешательства,аналогично тому,как на компьютере IBM PC при запуске пакетного файла (.BAT-файлы) он сам начинает загружать и запускать нужные вам программы в нужном порядке.

Обратите внимание на номер строки.Набрать такую строку с клавиатуры невозможно,как,впрочем,и отредактировать."Нулевая" строка формируется специальным образом и служит для того,чтобы не дать вам забраться в программу.Обычно кроме зануления строки применяются еще и специальные меры,чтобы вы не увидели на экране текст программы.

П Е Р Е Ч Е Н Ь И С П О Л Ь З О В А Н Н О Й Л И Т Е Р А Т У Р Ы

1. П.Хоровиц, У.Хилл "Искусство схемотехники" том 2
   М. Мир, 1986 г.

2. С.Викерс под редакцией Р.Бредбира "Программирование на языке БЕЙСИК"
   Лондон, 1983 г., (русский перевод 1987 г.).

3. Справочник по персональным ЭВМ под редакцией Б.Н.Малиновского
   Киев, Тэхника, 1990 г.

4. Инструкция для пользователей
   (приложение к компьютеру SPECTRUM-48)

5. А.Эндрю "Искусственный интеллект"
   М.Мир, 1985 г.

6. Микропроцессоры под редакцией Л.Н.Преснухина
   Книга 1. Минск. Вышэйшая школа, 1987 г.

ПРИЛОЖЕНИЕ 1

ПОЛНЫЙ НАБОР СИМВОЛОВ

ДЕС КОД	СИМВОЛ	ШЕСТН КОД	АССЕМБЛЕР МНЕМОНИКА	CBH...	EDH...
1	2	3	4	5	6
0	не использ.	00	NOP	RLC B	
1	не использ.	01	LD BC,NN	RLC C	
2	не использ.	02	LD (BC),A	RLC D	
3	не использ.	03	INC BC	RLC E	
4	не использ.	04	INC B	RLC H	
5	не использ.	05	DEC B	RLC L	
6	PRINT упр.	06	LD B,N	RLC (HL)	
7	EDIT	07	RLCA	RLC A	
8	курс.влево	08	EX AF,AF	RRC B	
9	курс.вправо	09	ADD HL,BC	RRC C	
10	курс.вниз	0A	LD A,(BC)	RRC D	
11	курс.вверх	0B	DEC BC	RRC E	
12	DELETE	0C	INC C	RRC H	
13	ENTER	0D	DEC C	RRC L	
14	число	0E	LD C,N	RRC (HL)	
15	не использ.	0F	RRCA	RRC A	
16	INK.упр.	10	DJNZ DIS	RL B	
17	PAPER упр.	11	LD DE,NN	RL C	
18	FLASH упр.	12	LD (DE),A	RL D	
19	BRIGHT упр.	13	INC DE	RL E	
20	INVERSE упр.	14	INC D	RL H	
21	OVER упр.	15	DEC D	RL L	
22	AT упр.	16	LD D,N	RL (HL)	
23	TAB упр.	17	RLA	RL A	
24	не использ.	18	JR DIS	RR B	
25	не использ.	19	ADD HL,DE	RR C	
26	не использ.	1A	LD A,(DE)	RR D	
27	не использ.	1B	DEC DE	RR E	
28	не использ.	1C	INC E	RR H	
29	не использ.	1D	DEC E	RR L	
30	не использ.	1E	LD E,N	RR (HL)	
31	не использ.	1F	RRA	RR A	
32	пробел	20	JR NZ,DIS	SLA B	
33	!	21	LD HL,NN	SLA C	
34	"	22	LD (NN),HL	SLA D	
35	#	23	INC HL	SLA E	
36	$(доллар)	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,DIS	SRA B	
41	)	29	ADD HL,HL	SRA C	
42	*	2A	LD HL,NN	SRA D	
43	+	2B	DEC HL	SRA E	
44	,	2C	INC L	SRA H	
45	-	2D	DEC L	SRA L	
46	.	2E	LD L,N	SRA (HL)	
47	/	2F	CPL	SRA A	
48	0	30	JR NC,DIS		
49	1	31	LD SP,NN		
50	2	32	LD (NN),A		
51	3	33	INC SP		
52	4	34	INC (HL)		
53	5	35	DEC (HL)		
54	6	36	LD (HL),N		
55	7	37	SCF		
56	8	38	JR C,DIS	SRL B	

1	2	3	4	5	6
57	9	39	ADD HL,SP	SRL C	
58	:	3A	LD A,(NN)	SRL D	
59	;	3B	DEC SP	SRL E	
60	<	3C	INC A	SRL H	
61	=	3D	DEC A	SRL L	
62	>	3E	LD A,N	SRC (HL)	
63	?	3F	CCF	SRL A	
64	@	40	LD B,B	BIT 0,B	IN B,(C)
65	A	41	LD B,C	BIT 0,C	OUT (C),B
66	B	42	LD B,D	BIT 0,D	SBC HL,BC
67	C	43	LD B,E	BIT 0,E	LD (NN),BC
68	D	44	LD B,H	BIT 0,H	NEG
69	E	45	LD B,L	BIT 0,L	RETN
70	F	46	LD B,(HL)	BIT 0,(HL)	IM 0
71	G	47	LD B,A	BIT 0,A	LD L,A
72	H	48	LD C,B	BIT 1,B	IN C,(C)
73	I	49	LD C,C	BIT 1,C	OUT (C),C
74	J	4A	LD C,D	BIT 1,D	ADD HL,BC
75	K	4B	LD C,E	BIT 1,E	LD BC ,(NN)
76	L	4C	LD C,H	BIT 1,H	
77	M	4D	LD C,L	BIT 1,L	RET I
78	N	4E	LD C,(HL)	BIT 1,(HL)	
79	O	4F	LD C,A	BIT 1,A	LD R,A
80	P	50	LD D,B	BIT 2,B	IN D,(C)
81	Q	51	LD D,C	BIT 2,C	OUT (C),D
82	R	52	LD D,D	BIT 2,D	SBC HL,DE
83	S	53	LD D,E	BIT 2,E	LD (NN),DE
84	T	54	LD D,H	BIT 2,H	
85	U	55	LD D,L	BIT 2,L	
86	V	56	LD D,(HL)	BIT 2,(HL)	IM 1
87	W	57	LD D,A	BIT 2,A	LD A,L
88	X	58	LD E,B	BIT 3,B	IN E,(C)
89	Y	59	LD E,C	BIT 3,C	OUT (C),E
90	Z	5A	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	стрелка врх	5E	LD E,(HL)	BIT 3,(HL)	IM 2
95	_	5F	LD E,A	BIT 3,A	LD A,R
96	фунт-стерл.	60	LD H,B	BIT 4,B	IN H,(C)
97	a (строчн)	61	LD H,C	BIT 4,C	OUT (C),H
98	b (строчн)	62	LD H,D	BIT 4,D	SBC HL,HL
99	c (строчн)	63	LD H,E	BIT 4,E	LD (NN),HL
100	d (строчн)	64	LD H,H	BIT 4,H	
101	e (строчн)	65	LD H,L	BIT 4,L	
102	f (строчн)	66	LD H,(HL)	BIT 4,(HL)	
103	g (строчн)	67	LD H,A	BIT 4,A	RRD
104	h (строчн)	68	LD L,B	BIT 5,B	IN L,(C)
105	i (строчн)	69	LD L,C	BIT 5,C	OUT (C),L
106	j (строчн)	6A	LD L,D	BIT 5,D	ADC HL,HL
107	k (строчн)	6B	LD L,E	BIT 5,E	LD HL,(NN)
108	l (строчн)	6C	LD L,H	BIT 5,H	
109	m (строчн)	6D	LD L,L	BIT 5,L	
110	n (строчн)	6E	LD L,(HL)	BIT 5,(HL)	
111	o (строчн)	6F	LD L,A	BIT 5,A	RID
112	p (строчн)	70	LD (HL),B	BIT 6,B	IN F,(C)
113	q (строчн)	71	LD (HL),C	BIT 6,C	
114	r (строчн)	72	LD (HL),D	BIT 6,D	SBC HL,SP
115	s (строчн)	73	LD (HL),E	BIT 6,E	LD (NN),SP
116	t (строчн)	74	LD (HL),H	BIT 6,H	
117	u (строчн)	75	LD (HL),L	BIT 6,L	
118	v (строчн)	76	LD (HL),(HL)	BIT 6,(HL)	
119	w (строчн)	77	LD (HL),A	BIT 6,A	
120	x (строчн)	78	LD A,B	BIT 7,B	IN A,(C)
121	y (строчн)	79	LD A,C	BIT 7,C	OUT (C),A
122	z (строчн)	7A	LD A,D	BIT 7,D	ADC HL,SP

1	2	3	4	5	6
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	C в окр.	7F	LD A,A	BIT 7,A	
128	0 0 0 0	80	ADD A,B	RES 0,B	
129	0 X 0 0	81	ADD A,C	RES 0,C	
130	X 0 0 0	82	ADD A,D	RES 0,D	
131	X X 0 0	83	ADD A,E	RES 0,E	
132	0 0 0 X	84	ADD A,H	RES 0,H	
133	0 X 0 X	85	ADD A,L	RES 0,L	
134	X 0 0 X	86	ADD A,(HL)	RES 0,(HL)	
135	X X 0 X	87	ADD A,A	RES 0,A	
136	0 0 X 0	88	ADC A,B	RES 1,B	
137	0 X X 0	89	ADC A,C	RES 1,C	
138	X 0 X 0	8A	ADC A,D	RES 1,D	
139	X X X 0	8B	ADC A,E	RES 1,E	
140	0 0 X X	8C	ADC A,H	RES 1,H	
141	0 X X X	8D	ADC A,L	RES 1,L	
142	X 0 X X	8E	ADC A,(HL)	RES 1,(HL)	
143	X X X X	8F	ADC A,A	RES 1,A	

1	2	3	4	5	6
144	(A) \	90	SUB B	RES 2,B	
145	(B)	91	SUB C	RES 2,C	
146	(C) /	92	SUB D	RES 2,D	
147	(D)	93	SUB E	RES 2,E	
148	(E)	94	SUB H	RES 2,H	
149	(F)	95	SUB L	RES 2,L	
150	(G)	96	SUB (HL)	RES 2,(HL)	
151	(H)	97	SUB A	RES 2,A	
152	(I) Графика	98	SBC A,B	RES 3,B	
153	(J) пользо-	99	SBC A,C	RES 3,C	
154	(K) теля.	9A	SBC A,D	RES 3,D	
155	(L)	9B	SBC A,E	RES 3,E	
156	(M)	9C	SBC A,H	RES 3,H	
157	(N)	9D	SBC A,L	RES 3,L	
158	(O)	9E	SBC A,(HL)	RES 3,(HL)	
159	(P)	9F	SBC A,A	RES 3,A	
160	(Q)	A0	AND B	RES 4,B	LDI
161	(R)	A1	AND C	RES 4,C	CPI
162	(S)	A2	AND D	RES 4,D	INI
163	(T)	A3	AND E	RES 4,E	OUTI
164	(U) /	A4	AND H	RES 4,H	
165	RND	A5	AND L	RES 4,L	
166	INKEY$	A6	AND (HL)	RES 4,(HL)	
167	PI	A7	AND A	RES 4,A	
168	FN	A8	XOR B	RES 5,B	LDD
169	POINT	A9	XOR C	RES 5,C	CPD
170	SCREEN$	AA	XOR D	RES 5,D	IND
171	ATTR	AB	XOR E	RES 5,E	OUTD
172	AT	AC	XOR H	RES 5,H	
173	TAB	AD	XOR L	RES 5,L	
174	VAL$	AE	XOR (HL)	RES 5,(HL)	
175	CODE	AF	XOR A	RES 5,A	
176	VAL	B0	OR B	RES 6,B	LDIR
177	LEN	B1	OR C	RES 6,C	CPIR
178	SIN	B2	OR D	RES 6,D	INIR
179	COS	B3	OR E	RES 6,E	OTIR
180	TAN	B4	OR H	RES 6,H	
181	ASN	B5	OR L	RES 6,L	
182	ACS	B6	OR (HL)	RES 6,(HL)	
183	ATN	B7	OR A	RES 6,A	
184	LN	B8	CP B	RES 7,B	LDDR
185	EXP	B9	CP C	RES 7,C	CPDR
186	INT	BA	CP D	RES 7,D	INDR
187	SQR	BB	CP E	RES 7,E	OTDR
188	SGN	BC	CP H	RES 7,H	
190	ABS	BD	CP L	RES 7,L	
191	PEEK	BE	CP (HL)	RES 7,(HL)	
192	IN	BF	CP A	RES 7,A	
193	USR	C0	RET NZ	SET 0,B	
194	STR$	C1	POP BC	SET 0,C	
195	CHR$	C2	JP NZ,NN	SET 0,D	
196	NOT	C3	JP NN	SET 0,E	
197	BIN	C4	CALL NZ,NN	SET 0,H	
198	OR	C5	PUSH BC	SET 0,L	
199	AND	C6	ADD A,N	SET 0,(HL)	
200	<=	C7	RST 0	SET 0,A	
201	>=	C8	RET Z	SET 1,B	
202	<>	C9	RET	SET 1,C	
202	LINE	CA	JP Z,NN	SET 1,D	
203	THEN	CB		SET 1,E	
204	TO	CC	CALL Z,NN	SET 1,H	
205	STEP	CD	CALL NN	SET 1,L	
206	DEF FN	CE	ADC A,N	SET 1,(HL)	
207	CAT	CF	RST 8	SET 1,A	
208	FORMAT	D0	RET NC	SET 2,B	
209	MOVE	D1	POP DE	SET 2,C	

1	2	3	4	5	6
210	ERASE	D2	JP NC,NN	SET 2,D	
211	OPEN#	D3	OUT (N),A	SET 2,E	
212	CLOSE#	D4	CALL NC,NN	SET 2,H	
213	MERGE	D5	PUSH DE	SET 2,L	
214	VERIFY	D6	SUB N	SET 2,(HL)	
215	BEEP	D7	RST 16	SET 2,A	
216	CIRCLE	D8	RET C	SET 3,B	
217	INK B4	OR H	RES 6,H		
181	ASN	B5	OR L	RES 6,L	
182	ACS	B6	OR (HL)	RES 6,(HL)	
183	ATN	B7	OR A	RES 6,A	
184	LN	B8	CP B	RES 7,B	LDDR
185	EXP	B9	CP C	RES 7,C	CPDR
186	INT	BA	CP D	RES 7,D	INDR
187	SQR	BB	CP E	RES 7,E	OTDR
188	SGN	BC	CP H	RES 7,H	
190	ABS	BD	CP L	RES 7,L	
191	PEEK	BE	CP (HL)	RES 7,(HL)	
192	IN	BF	CP A	RES 7,A	
193	USR	C0	RET NZ	SET 0,B	
194	STR$	C1	POP BC	SET 0,C	
195	CHR$	C2	JP NZ,NN	SET 0,D	
196	NOT	C3	JP NN	SET 0,E	
197	BIN	C4	CALL NZ,NN	SET 0,H	
198	OR	C5	PUSH BC	SET 0,L	
199	AND	C6	ADD A,N	SET 0,(HL)	
200	<=	C7	RST 0	SET 0,A	
201	>=	C8	RET Z	SET 1,B	
202	<>	C9	RET	SET 1,C	
202	LINE	CA	JP Z,NN	SET 1,D	
203	THEN	CB		SET 1,E	
204	TO	CC	CALL Z,NN	SET 1,H	
205	STEP	CD	CALL NN	SET 1,L	
241	LET	F1	POP AF	SET 6,C	
242	PAUSE	F2	JP P,NN	SET 6,D	
243	NEXT	F3	DI	SET 6,E	
244	POKE	F4	CALL P,NN	SET 6,H	
245	PRINT	F5	PUSH AF	SET 6,L	
246	PLOT	F6	OR N	SET 6,(HL)	
247	RUN	F7	RST 48	SET 6,A	
248	SAVE	F8	RET M	SET 7,B	
249	RANDOMIZE	F9	LD SP,HL	SET 7,C	
250	IF	FA	JP M,NN	SET 7,D	
251	CLS	FB	EI	SET 7,E	
252	DRAW	FC	CALL M,NN	SET 7,H	
253	CLEAR	FD	PREFIX IN- STRUKTIONS, USING IY	SET 7,L	
254	RETURN	FE	CP N	SET 7,(HL)	
255	COPY	FF	RST 56	SET 7,A	

ПРИЛОЖЕНИЕ 2

ШЕСТНАДЦАТЕРИЧНАЯ И ДВОИЧНАЯ СИСТЕМЫ СЧИСЛЕНИЯ

В компьтере ZX SPECTRUM используется шестнадцатеричная система счисления. При этом каждая шестнадцатеричная цифра записывается четырьмя двоичными (тетрада). Таким образом, в одном байте может быть записано 2 шестнадцатеричных числа.

Десятичное	16-ричное	Двоичное (байт)
0	00	0000 0000
1	01	0000 0001
2	02	0000 0010
3	03	0000 0011
4	04	0000 0100
5	05	0000 0101
6	06	0000 0110
7	07	0000 0111
8	08	0000 1000
9	09	0000 1001
10	0A	0000 1010
11	0B	0000 1011
12	0C	0000 1100
13	0D	0000 1101
14	0E	0000 1110
15	0F	0000 1111
16	10	0001 0000
17	11	0001 0001
18	12	0001 0010
19	13	0001 0011
20	14	0001 0100
21	15	0001 0101
22	16	0001 0110
23	17	0001 0111
24	18	0001 1000
25	19	0001 1001
26	1A	0001 1010
27	1B	0001 1011
28	1C	0001 1100
29	1D	0001 1101
30	1E	0001 1110
31	1F	0001 1111
32	20	0010 0000
. . .	. . .	. . .
65535	FF	1111 1111

Два байта образуют машинное слово. Для записи двоичных кодов служит функция BIN. Например " BIN 0 " запишет в память двоичный 0, " BIN 10 " записывает число 2 и т.д. Для записи "-3" необходимо указать " -BIN 11 ", но не " BIN -11 ".

Число не может превышать 65535, т.е. занимать более 16 двоичных разрядов (бит).

ПРИЛОЖЕНИЕ 3

ПРВ стр 62

СООБЩЕНИЯ

Они появляются в нижней части экрана, если компьютер остановился при выполнении некоторого оператора БЕЙСИКА, и указывает причину, по которой произошел останов.

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

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

КОД	ЗНАЧЕНИЕ	СИТУАЦИЯ
1	2	3
0	OK (О'кей! Порядок!) Успешное завершение или переход на строку с номером, большим, чем имеется всего. Это сообщение не меняет строки или оператора, определенного для CONTINUE	Разное
1	NEXT WITHOUT FOR (NEXT без FOR) Управляющей переменной нет (не была определена в операторе FOR), но есть обычная переменная с тем же именем.	NEXT
2	VARIABLE NOT FOUND (Переменная не найдена) Для простой переменной выдается, если она используется без предварительного определения в операторах LET,READ или INPUT, или загружается с ленты или устанавливается в операторе FOR. Для индексируемой переменной сообщение выдается, если она не была предварительно определена в операторе DIM перед использованием или загрузкой с ленты.	Разное
3	SUBSCRIPT WRONG (Ошибочный индекс) Индекс превышает размерность массива, либо ошибочное число задает индекс. Если индекс отрицательный или больше 65535, то выдается сообщение B.	В индексной переменной или подстроке
4	OUT OF MEMORY (Вне памяти) В памяти недостаточно места для ваших действий. Вы можете освободить себе память, удалив командные строки, используя DELETE, затем удалить одну или две строки программы (с целью возврата их впоследствии), получить дополнительную память, маневрируя оператором CLEAR.	LET, INPUT, FOR, DIM, GO SUB, LOAD, MERGE
5	OUT OF SCREEN (Вне экрана) Если INPUT-оператор генерирует больше, чем 23 строки в нижней половине экрана. Также встречается с PRINT AT 22,...	INPUT PRINT AT

1	2	3
6	NUMBER TOO BIG (Число больше макс.-допустимого) В результате вычислений получилось число, большее, чем 10**38	Арифметические операции
7	RETURN WITHOUT GO SUB (RETURN без GO SUB) Встретилось больше операторов RETURN, чем было GO SUB.	RETURN
8	END OF FILE (Конец файла)	Операции с внешней памятью
9	STOP STATEMENT (Оператор STOP) После этого сообщения CONTINUE не может повторить STOP, но может передать управление на следующий оператор.	STOP
A	INVALID ARGUMENT (Ошибочный аргумент) Аргумент функции недопустим в данной версии	SQR, LN, ASN, ACS, USR со строковым аргументом
B	INTEGER OUT OF RANGE (Переполнение целого) Выдается, когда аргумент с плавающей точкой округляется к целому. Для случая массивов смотри также сообщение 3.	GO TO, GO SUB, LIST, LLIST, PAUSE, PLOT, CHR$, PEEK, RUN, RANDOMIZE, POKE, DIM, USR(с числовым аргументом)
C	NONSENSE IN BASIC (Выражение не БЕЙСИКа) Текст (строка) не распознается БЕЙСИКом как допустимое выражение.	VAL, VAL$
D	BREAK-CONT REPEATS (Повторяется BREAK) Клавиша BREAK нажата во время действия периферийной операции. Действие CONTINUE после этого оператора обычные, те что указаны в операторе. Сравните с сообщением L.	LOAD, SAVE, VERIFY, MERGE, LPRINT, LLIST, COPY только когда компьютер запросил свертку,а вы ответили N, SPACE или STOP
E	OUT OF DATA (Вне данных) Попытка выдать READ, когда список данных в DATA кончился.	READ

1	2	3
F	INVALID FILE NAME (Неверное имя файла) Оператор SAVE с пустой строкой вместо имени файла или с именем длиннее 10 символов	SAVE
G	NO ROOM FOR LINE (Нет места для строки) Недостаточно места в памяти для записи очередной строки программы.	Ввод строки в программу
H	STOP IN INPUT Некоторые введенные данные начинаются с оператора STOP, или была нажата INPUT LINE. Действие CONTINUE обычное.	INPUT
I	FOR WITHOUT NEXT (FOR без NEXT) Цикл FOR ни разу не выполнялся, не найден оператор NEXT	FOR
J	INVALID I/O DEVICE (Неверное устройство ввода-вывода)	В операциях с внешними устройствами
K	INVALID COLOUR (Неверный символ) Специфицированное число имеет неверное значение	INK, PAPER, FLASH, BRIGHT, INVERSE, OVER, А также после передачи одного из управляющих символов.
L	BREAK INTO PROGRAM (BREAK во время выполнения программы) Нажата клавиша BREAK; это обнаруживается между двумя операторами. Строка и номер оператора в строке указывают на оператор, выполняемый перед нажатием BREAK, но CONTINUE переходит к следующему оператору	РАЗНОЕ
M	RAMTOP NO GOOD (Адрес RAMTOP не годен) Число, указанное для RAMTOP слишком велико или слишком мало	CLEAR Возможно RUN
N	STATMENT LOST (Оператор отсутствует) Переход к оператору, которого уже нет.	RETURN, NEXT, CONTINUE
O	INVALID STREAM (Ошибочный поток данных)	В операциях ввода-вывода
P	FN WITHOUT DEF (FN без DEF) Определяемая пользователем функция не определена в операторе DEF FN.	FN
Q	PARAMETER ERROR (Ошибка в параметре) Ошибочное число аргументов или один из них не того типа, который был описан.	FN
R	TAPE LOADING ERROR (Ошибка загрузки с ленты) Файл на ленте найден, но не считывается	VERIFY, LOAD, MERGE

FOREVER (OK 0:1)

ПРИЛОЖЕНИЕ 4.

стр 38

ПРИМЕРЫ ПРОГРАММ

Это приложение содержит некоторые примеры программ, демонстрирующие возможности ZX SPECTRUM.

Первая из этих программ требует ввести дату и дает день недели, который соответствует этой дате.

```
 10 REM convert date to day
 20 DIM d$(7,6): REM days of week
 30 FOR n=1 TO 7: READ d$(n): NEXT n
 40 DIM m(12): REM lengths of months
 50 FOR n=1 TO 12: READ m(n): NEXT n
100 REM input date
110 INPUT "day?";day
120 INPUT "month?";month
130 INPUT "year (20th century only)?";year
140 IF year<1901 THEN PRINT "20th century starts at 1901":
    GO TO 100
150 IF year>2000 THEN PRINT "20th century ends at 2000":
    GO TO 100
160 IF month<1 THEN GO TO 210
170 IF month>12 THEN GO TO 210
180 IF year/4-INT(year/4)=0 THEN LET m(2)=29:
    REM LEAP YEAR
190 IF day>m(month) THEN PRINT "This month has only";
    m(month);"days.": GO TO 500
200 IF day>0 THEN GO TO 300
210 PRINT "Stuff and nonsense. Give me a real date."
220 GO TO 500
300 REM convert date to number of days since start of
    century
310 LET y=year-1901
320 LET b=365*y+INT(y/4): REM number of days to start
                              of year
330 FOR n=1 TO month-1: REM add on previous month
340 LET b=b+m(n): NEXT n
350 LET b=b+day
400 REM convert to day of week
410 LET b=b-7*INT(b/7)+1
420 PRINT day;"/";month,"/",year
430 FOR n=6 TO 3 STEP -1:REM remove trailing spaces
440 IF d$(b,n)<>" " THEN GO TO 460
450 NEXT n
460 LET e$=d$(b,TO n)
470 PRINT"is a ";e$;"day"
500 LET m(2)=28: REM restore February
510 INPUT "again?",a$
520 IF a$="n" THEN GO TO 540
530 IF a$<>"N" THEN GO TO 100
1000 REM days of week
     1010 DATA "Mon","Tues","Wednes"
     1020 DATA "Thurs","Fri","Satur","Sun"
     1100 REM iengths of months
     1110 DATA 31,28,31,30,31,30
     1120 DATA 31,31,30,31,30,31
```

1	2	3
F	INVALID FILE NAME (Неверное имя файла) Оператор SAVE с пустой строкой вместо имени файла или с именем длиннее 10 символов	SAVE
G	NO ROOM FOR LINE (Нет места для строки) Недостаточно места в памяти для записи очередной строки программы.	Ввод строки в программу
H	STOP IN INPUT Некоторые введенные данные начинаются с оператора STOP, или была нажата INPUT LINE. Действие CONTINUE обычное.	INPUT
I	FOR WITHOUT NEXT (FOR без NEXT) Цикл FOR ни разу не выполнялся, не найден оператор NEXT	FOR
J	INVALID I/O DEVICE (Неверное устройство ввода-вывода)	В операциях с внешними устройствами
K	INVALID COLOUR (Неверный символ) Специфицированное число имеет неверное значение	INK, PAPER, FLASH, BRIGHT, INVERSE, OVER, А также после передачи одного из управляющих символов.
L	BREAK INTO PROGRAM (BREAK во время выполнения программы) Нажата клавиша BREAK; это обнаруживается между двумя операторами. Строка и номер оператора в строке указывают на оператор, выполняемый перед нажатием BREAK, но CONTINUE переходит к следующему оператору	РАЗНОЕ
M	RAMTOP NO GOOD (Адрес RAMTOP не годен) Число, указанное для RAMTOP слишком велико или слишком мало	CLEAR Возможно RUN
N	STATMENT LOST (Оператор отсутствует) Переход к оператору, которого уже нет.	RETURN, NEXT, CONTINUE
O	INVALID STREAM (Ошибочный поток данных)	В операциях ввода-вывода
P	FN WITHOUT DEF (FN без DEF) Определяемая пользователем функция не определена в операторе DEF FN.	FN
Q	PARAMETER ERROR (Ошибка в параметре) Ошибочное число аргументов или один из них не того типа, который был описан.	FN
R	TAPE LOADING ERROR (Ошибка загрузки с ленты) Файл на ленте найден, но не считывается	VERIFY, LOAD, MERGE

FOREVER (ok 0:1)

ПРИЛОЖЕНИЕ 4.

ПРИМЕРЫ ПРОГРАММ

Это приложение содержит некоторые примеры программ, демонстрирующие возможности ZX SPECTRUM.

Первая из этих программ требует ввести дату и дает день недели, который соответствует этой дате.

```
 10 REM convert date to day
 20 DIM d$(7,6): REM days of week
 30 FOR n=1 TO 7: READ d$(n): NEXT n
 40 DIM m(12): REM lengths of months
 50 FOR n=1 TO 12: READ m(n): NEXT n
100 REM input date
110 INPUT "day?";day
120 INPUT "month?";month
130 INPUT "year (20th century only)?";year
140 IF year<1901 THEN PRINT "20th century starts at 1901":
    GO TO 100
150 IF year>2000 THEN PRINT "20th century ends at 2000":
    GO TO 100
160 IF month<1 THEN GO TO 210
170 IF month>12 THEN GO TO 210
180 IF year/4-INT(year/4)=0 THEN LET m(2)=29:
    REM LEAP YEAR
190 IF day>m(month) THEN PRINT "This month has only";
    m(month);"days.": GO TO 500
200 IF day>0 THEN GO TO 300
210 PRINT "Stuff and nonsense. Give me a real date."
220 GO TO 500
300 REM convert date to number of days since start of
    century
310 LET y=year-1901
320 LET b=365*y+INT(y/4): REM number of days to start
                              of year
330 FOR n=1 TO month-1: REM add on previous month
340 LET b=b+m(n): NEXT n
350 LET b=b+day
400 REM convert to day of week
410 LET b=b-7*INT(b/7)+1
420 PRINT day;"/";month,"/",year
430 FOR n=6 TO 3 STEP -1:REM remove trailing spaces
440 IF d$(b,n)<>" " THEN GO TO 460
450 NEXT n
460 LET e$=d$(b,TO n)
470 PRINT"is a ";e$;"day"
500 LET m(2)=28: REM restore February
510 INPUT "again?",a$
520 IF a$="n" THEN GO TO 540
530 IF a$<>"N" THEN GO TO 100
1000 REM days of week
     1010 DATA "Mon","Tues","Wednes"
     1020 DATA "Thurs","Fri","Satur","Sun"
     1100 REM iengths of months
     1110 DATA 31,28,31,30,31,30
     1120 DATA 31,31,30,31,30,31
```

Эта программа устанавливает соответствие между ярдом, футом и дюймом.

```
 10 INPUT "yards?",yd,"feet?",ft,"inches?",in
 40 GO SUB 2000: REM print the values
 50 PRINT " = ";
 70 GO SUB 1000: REM the adjustement
 80 GO SUB 2000: REM print the adjusted values
 90 PRINT
100 GO TO 10
1000 REM subroutine to adjust yd, ft, in to the normal form for
       yards, feet and inches
1010 LET in=36*yd+12*ft+in: REM now everything is in inches
1030 LET s=SGN in: LET in=ABS in: REM we work with in
     positive, holding its sign in s
1060 LET ft=INT(in/12): LET in=(in-12*ft)*s: REM now in is ok
1080 LET yd=INT(ft/3)*s: LET ft=ft*s-3*yd: RETURN
2000 REM subroutine to print yd, ft and in
     PRINT yd;"yd";ft;"ft";in;"in";: RETURN
```

Эта программа моделирует выбрасывание монеты для игры в 'китайку'.

```
  5 RANDOMIZE
 10 FOR m=1 TO 6: REM for 6 throws
 20 LET c=0: REM initialize coin total to 0
 30 FOR n=1 to 3: REM for 3 coins
 40 LET c=c+2+INT(2*RND)
 50 NEXT n
 60 PRINT "   ";
 70 FOR n=1 TO 2: REM 1st for the thrown hexagram, 2nd for
    the changes
 80 PRINT "___";
 90 IF c=7 THEN PRINT "_";
100 IF c=8 THEN PRINT " ";
110 IF c=6 THEN PRINT "X";: LET c=7
120 IF c=9 THEN PRINT "0";: LET c=8
130 PRINT "___ ";
140 NEXT N
150 PRINT
160 INPUT a$
170 NEXT m: NEW
```

Для запуска программы введите ее в компьютер, запустите на выполнение, а затем нажмите клавишу ENTER пять раз для получения двух хексаграмм. Просмотрите "Китайскую книгу изменений". Текст будет описывать ситуации и последовательность соответствующих этому действий, а Вы должны оценить глубину параллелей между ней и Вашей собственной жизнью. Нажмите клавишу ENTER шестой раз и программа будет обнуляться - это избавит Вас от легкомысленного использования результатов.

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

Следующая программа - игра "Ящеры". Вы задумываете название некоторого животного, а компьютер пытается его отгадать, задавая Вам вопросы, на которые Вы должны отвечать "да" или "нет". Если компьютер не был ранее знаком с таким животным, то он попросит Вас задать ему наводящие вопросы, которые помогут ему найти правильный ответ, или он попросит Вас предложить ему задать название нового животного:

```
  5 REM pangolines
 10 LET nq=100: REM number of questions and animals
 15 DIM q$(nq,50): DIM a(nq,2): DIM r$(1)
 20 LET qf=8
 30 FOR n=1 TO qf/2-1
 40 READ q$(n): READ a(n,1): READ a(n,2)
 50 NEXT n
 60 FOR n=n TO qf-1
 70 READ q$(n): NEXT n

100 REM start playing
110 PRINT "Think of an animal.","Press any key to continue."
120 PAUSE 0
130 LET c=1: REM start with 1st question
140 IF a(c,1)=0 THEN GO TO 300
150 LET p$=q$(c): GO SUB 910
160 PRINT "?": GO SUB 1000
170 LET in=1: IF r$="y" THEN GO TO 210
180 IF r$="y" THEN GO TO 210
190 LET in=2: IF r$="n" THEN GO TO 210
200 IF r$<>"n" THEN GO TO 150
210 LET c=a(c,in): GO TO 140

300 REM animal
310 PRINT "Are you thinking of"
320 LET p$=q$(c): GO SUB 900: PRINT "?"
330 GO SUB 1000
340 IF r$="y" THEN GO TO 400
350 IF r$="Y" THEN GO TO 400
360 IF r$="n" THEN GO TO 500
370 IF r$="n" THEN GO TO 500
380 PRINT "Answer me properly when I'm","talking to you.": GO TO 300

400 REM guessed it
410 PRINT "I thought as much.": GO TO 800

500 REM new animal
510 IF qf>nq-1 THEN PRINT "I am sure your animal is very",
    "interesting, but I don't have","room for it just now.": GO TO 800
520 LET q$(qf)=q$(c): REM move old animal
530 PRINT "What is it, then?": INPUT q$(fq+1)
540 PRINT "Tell me a question which dist-","inguishes
    between"
550 LET p$=q$(qf): GO SUB 900: PRINT "and"
560 LET p$=q$(qf+1): GO SUB 900: PRINT " "
570 INPUT s$: LET b=LEN s$
580 IF s$(b)="?" THEN LET b=b-1
590 LET q$(c)=s$(TO b): REM insert question
600 PRINT "What is the answer for"
610 LET p$=q$(qf+1): GO SUB 900: PRINT "?"
620 GO SUB 1000
630 LET in=1: LET io=2: REM answers for new and old animals"
640 IF r$="y" THEN GO TO 700
650 IF r$="Y" THEN GO TO 700
660 LET in=2: LET io=1
670 IF r$="n" THEN GO TO 700
680 IF r$="n" THEN GO TO 700
690 PRINT "That's no good": GO TO 600
```

```
700 REM update answers
710 LET a(c,in)=qf+1: LET a(c,io)=qf
720 LET qf=qf+2: REM next free animal space
730 PRINT "That fooled me."

800 REM again?
810 PRINT "Do you want another go?": GO SUB 1000
820 IF r$="y" THEN GO TO 100
830 IF r$="Y" THEN GO TO 100
840 STOP

900 REM print whithout trailing spaces
905 PRINT " ";
910 FOR n=50 TO 1 STEP -1
920 IF p$(n)<>" " THEN GO TO 940
930 NEXT N
940 PRINT p$(TO n);: RETURN

1000 REM get rerly
1010 INPUT r$: IF r$="" THEN RETURN
1020 LET r$=r$(1): RETURN

2000 REM initial animals
2010 DATA "Does it live in the sea",4,2
2020 DATA "Is it scaly",3,5
2030 DATA "Does it eat ants",6,7
2040 DATA "a whale","a blancmange","a pangoline","an ant"
```

Следующая программа рисует на экране "UNION FLAG":

```
  5 REM union flag
 10 LET r=2: LET w=7: LET b=1
 20 BORDER 0: PAPER 0: INK w: CLS
 30 REM black in bottom of screen
 40 INVERSE 1
 50 FOR n=40 TO 0 STEP -8
 60 PLOT PAPER 0;7,n: DRAW PAPER 0;241,0
 70 NEXT n: INVERSE 0
100 REM draw in white parts
105 REM St. George
110 FOR n=0 TO 7
120 PLOT 104+n,175: DRAW 0,-35
130 PLOT 151-n,175: DRAW 0,-35
140 PLOT 151-n,48: DRAW 0,35
150 PLOT 104+n,48: DRAW 0,35
160 NEXT n
200 FOR n=0 TO 11
210 PLOT 0,139-n: DRAW 111,0
220 PLOT 255,139-n: DRAW -111,0
230 PLOT 255,84+n: DRAW -111,0
240 PLOT 0,84+n: DRAW 111,0
250 NEXT n
300 REM St. Andrew
310 FOR n=0 TO 35
320 PLOT 1+2*n,175-n: DRAW 32,0
330 PLOT 224-2*n,175-n: DRAW 16,0
340 PLOT 254-2*n,48+n: DRAW -32,0
350 PLOT 17+2*n,48+n: DRAW 16,0
360 NEXT n
370 FOR n=0 TO 19
380 PLOT 185+2*n,140+n: DRAW 32,0
390 PLOT 200+2*n,83-n: DRAW 16,0
400 PLOT 39-2*n,83-n: DRAW 32,0
410 PLOT 54-2*n,140+n: DRAW -16,0
420 NEXT n
425 REM fill in extra bits
430 FOR n=0 TO 15
440 PLOT 255,160+n: DRAW 2*n-30,0
```

```
450 PLOT 0,63-n: DRAW 31-2*n,0
460 NEXT n
470 FOR n=0 TO 7
480 PLOT 0,160+n: DRAW 14-2*n,0
485 PLOT 255,63-n: DRAW 2*n-15,0
490 NEXT n
500 REM red stripes
510 INVERSE 1
520 REM St. George
530 FOR n=96 TO 120 STEP 8
540 PLOT PAPER r;7,n: DRAW PAPER r;241,0
550 NEXT n
560 FOR n=112 TO 136 STEP 8
570 PLOT PAPER r;n,168: DRAW PAPER r;0,-113
580 NEXT n
600 REM St. Patrick
610 PLOT PAPER r;170,140: DRAW PAPER r;70,35
620 PLOT PAPER r;179,140: DRAW PAPER r;70,35
630 PLOT PAPER r;199,83: DRAW PAPER r;56,-28
640 PLOT PAPER r;184,83: DRAW PAPER r;70,-35
650 PLOT PAPER r;86,83: DRAW PAPER r;-70,-35
660 PLOT PAPER r;72,83: DRAW PAPER r;-70,-35
670 PLOT PAPER r;56,140: DRAW PAPER r;-56,28
680 PLOT PAPER r;71,140: DRAW PAPER r;-70,35
690 INVERSE 0: PAPER 0: INK 7
```

Если Вы не англичанин, то можете изобразить свой флаг.

Следующая программа - это игра в слова. Первый игрок вводит слово, а второй его отгадывает.

```
  5 REM Hangman
 10 REM set up screen
 20 INK 0: PAPER 7: CLS
 30 LET x=240: GO SUB 1000: REM draw man
 40 PLOT 238,128: DRAW 4,0: REM mouth
100 REM set up word
110 INPUT w$: REM word to guess
120 LET b=LEN w$: LET v$=" "
130 FOR n=2 TO b: LET v$=v$+" "
140 NEXT n: REM v$=word guessed so far
150 LET c=0: LET d=0: REM guess & mistake counts
160 FOR n=0 TO b-1
170 PRINT AT 20,n;"-";
180 NEXT n: REM write-'s instead of letters
200 INPUT "Guess a letter: ";g$
210 IF g$="" THEN GO TO 200
220 LET g$=g$(1): REM 1st letter only
230 PRINT AT 0;c;g$
240 LET c=c+1: LET u$=v$
250 FOR n=1 TO b: REM update guessed word
260 IF w$(n)=g$ THEN LET v$(n)=g$
270 NEXT n
280 PRINT AT 19,0;v$
290 IF v$=w$ THEN GO TO 500: REM word guessed
300 IF v$<>u$ THEN GO TO 200: REM guess was right
400 REM draw next part of gallows
410 IF d=8 THEN GO TO 600: REM hanged
420 LET d=d+1
430 READ x0,y0,x,y
440 PLOT x0,y0: DRAW x,y
450 GO TO 200
500 REM free man
510 OVER 1: REM rub out man
520 LET x=240: GO SUB 1000
530 PLOT 238,128: DRAW 4,0: REM mouth
540 OVER 0: REM redraw man
550 LET x=146: GO SUB 1000
560 PLOT 143,129: DRAW 6,0,PI/2:REM smile
570 GO TO 800
```

```
600 REM hang man
610 OVER 1: REM rub out floor
620 PLOT 255,65: DRAW -48,0
630 DRAW 0,-48: REM open trapdoor
640 PLOT 238,128: DRAW 4,0: REM rubout mouth
650 REM move limbs
655 REM arms
660 PLOT 255,117: DRAW -15,-15: DRAW -15,15
670 OVER 0
680 PLOT 236,81: DRAW 4,21: DRAW 4,-21
690 OVER 1: REM legs
700 PLOT 255,66: DRAW -15,15: DRAW -15,-15
710 OVER 0
720 PLOT 236,60: DRAW 4,21: DRAW 4,-21
730 PLOT 237,127: DRAW 6,0,-PI/2: REM frown
740 PRINT AT 19,0;w$
800 INPUT "again?";a$
810 IF a$="" THEN GO TO 850
820 LET a$=a$(1)
830 IF a$="n" THEN STOP
840 IF a$(1)="N" THEN STOP
850 RESTORE: GO TO 5
1000 REM draw man at column x
1010 REM head
1020 CIRCLE x,132,8
1030 PLOT x+4,134: PLOT x-4,134: PLOT x,131
1040 REM body
1050 PLOT x,123: DRAW 0,-20
1055 PLOT x,101: DRAW 0,-19
1060 REM legs
1070 PLOT x-15,66: DRAW 15,15: DRAW 15,-15
1080 REM arms
1090 PLOT x-15,117: DRAW 15,-15: DRAW 15,15
1100 RETURN
2000 DATA 120,65,135,0,184,65,0,91
2010 DATA 168,65,16,16,184,81,16,-16
2020 DATA 184,156,68,0,184,140,16,16
2030 DATA 204,156,-20,-20,240,156,0,-16
```

ПРИЛОЖЕНИЕ 5

ЗАГРУЗКА ПРОГРАММ С МАГНИТОФОНА.

Загрузка и запуск готовых программ выполняются весьма просто: LOAD "" <ENTER>. Помните, что между кавычками не должно быть пробела.

Основную проблему на этом этапе представляют плохая загружаемость программ, о чем свидетельствует сообщение об ошибке TAPE LOADING ERROR, зависание программы после загрузки или самопроизвольный сброс. Причина не обязательно связана с плохой загрузкой, но если лента получена вами из стороннего источника, то это скорее всего так. Приведем некоторые принципы, которыми надо руководствоваться, чтобы улучшить загружаемость программ.

1. По-настоящему надежную загрузку может обеспечить только высококачественная лента, если она записана на том же магнитофоне, на котором и воспроизводится. Поскольку стоимость программ обычно значительно превышает стоимость ленты, то не надо на ней экономить, а понравившиеся вам программы надо откопировать на своем магнитофоне (см. ниже).
2. Для работы с компьютером годится любой бытовой кассетный магнитофон, причем чем он проще, тем легче добиться надежной работы. В то же время желательно, чтобы он имел регуляторы тембров (по-возможности раздельные). Очень хорошо, если есть регулировка скорости движения ленты. Компьютер весьма чувствителен к отклонениям скорости. Монофонические магнитофоны во всех случаях предпочтительнее, чем стерео. Обязательным элементом является счетчик ленты, позволяющий быстро находить нужную программу.
3. Приступая к загрузке программы, установите уровень громкости на магнитофоне примерно на 70%-80% от максимального и, если загрузка не пошла, поэкспериментируйте с ним. Помните, что программы чаще страдают от избыточного уровня, чем от недостаточного. Поиск ведите в сторону уменьшения уровня. Иногда на фирменных компьютерах помогает следующий прием. Загрузочный шнур вставляют не в гнездо EAR компьютера, а в гнездо MIC. На самодельных аппаратах этот метод не сработает, т.к. там эти цепи электрически разорваны.
4. Установка регуляторов тембра также имеет большое значение. Нормально "высокие" надо ставить на 80% от максимума, а "низкие" примерно на 20%. Если программа решительно не загружается, а регулировка громкости ничего не дает, то попробуйте установку 50%-50% и 20%-80%.
5. Загрузка программ сопровождается перемещением цветных полос по бордюрному полю телевизора. По нему можно также многое сказать о настройке магнитофона. Сначала идут в течение 3-х - 5-ти секунд широкие красные и голубые полосы. Это так называемый пилоттон. Он нужен для того, чтобы процессор подготовился к загрузке. Ширина красных и голубых полос должна быть одинаковой. Если голубые полосы шире, то значит уровень слишком большой. Если шире красные полосы - слишком низкий. Эти полосы должны медленно перемещаться по экрану. Слишком быстрое их перемещение говорит о несоответствии скоростей вашего магнитофона и того, на котором была сделана запись. Неравномерное перемещение (с ускорением и рывками) свидетельствует о некачественности лентопротяжной системы магнитофона либо вашего, либо того, на котором делалась запись. Бывают случаи, когда широкие красные и голубые полосы "превращаются" в широкие желтые и синие. Это может указывать на то, что следует поменять местами установку регуляторов тембра.
6. Если у вас стереомагнитофон и программа не загружается, попробуйте загружать ее отдельно по левому и по правому каналам.
7. Если все пути регулировки исчерпаны, но программа не загружается и при этом точно известно, что копия работоспособна, по всей видимости причина в настройке головок у писавшего и воспроизводящего магнитофона. Как правило, недорогие монофонические магнитофоны имеют в крышке отверстие для узкой крестовой отвертки, чтобы можно было при воспроизведении записи на слух поворотом подпружиненного винта регулировать азимутальное положение головки.

При этом следует добиваться наиболее четкого и резкого звучания сигналов. Обычно бывает достаточно поворота в ту или иную сторону на угол до 120 градусов. Если у вас только один магнитофон, и вы используете его также и для записи программ, то регулировать головку нежелательно, либо периодически придется сталкиваться с необходимостью юстировки. Самый лучший вариант - использовать для воспроизведения программ, полученных со стороны, отдельный магнитофон, выбрав его из самых простых и дешевых, а для записи программ - наиболее качественный, по возможности специализированный аппарат.

8. Обычно программы состоят из нескольких блоков, которые загружаются последовательно один за другим. При плохой загрузке бывает очень трудно загрузить программу, разные блоки которой требуют разной установки органов управления магнитофона. Для того чтобы обойти эту проблему, надо проводить загрузку в копирующую программу. Если , предположим, при загрузке второго блока произойдет сбой, то первый блок уже загружен и остается , так что его повторять не надо. К тому же по окончании длительной трудоемкой загрузки, можно сразу же выполнить качественную копию и больше не иметь с этой программой проблем.

Примеры директив загрузки:

LOAD ""	загрузить первую встречную БЕЙСИК-программу
LOAD "DEMO"	загрузить БЕЙСИК-программу "DEMO"
LOAD "" CODE	загрузить первый встречный блок кодов с адреса, указанного в заголовке блока
LOAD "RUNNER" CODE 30000	загрузить блок кодов "RUNNER" с адреса 30000
LOAD "" SCREEN $	загрузить любое изображение в экран
LOAD "FULL" SCREEN $	загрузить картинку "FULL" в экран

9. Сделать процесс настройки магнитофона наглядным и значительно более простым можно с помощью специальных программ, например с помощью программы TAPER.

ПРОГРАММА TAPER

Программа предназначена для настройки кассетного магнитофона. Она анализирует (раскладывает по частотам) сигнал, приходящий на вход компьютера и изображает его на экране в виде спектра. Вертикальная шкала имеет три метки. Против этих меток должны находиться пики сигналов во время загрузки. Самая верхняя метка указывает на пик во время считывания "пилоттрона", а две другие - показывают уровень "нулей" и уровень "единиц". При прослушивании цифровой записи через программу TAPER пики в участках, отмеченных метками, должны быть ярко выражены и никаких других побочных всплесков не должно быть.

КОПИРОВАНИЕ ПРОГРАММ

Копирование программ, особенно компьютерных игр, весьма хитрая процедура из-за того, что ряд программ имеет очень сложную структуру, состоит из большого количества разнотипных файлов, не всегда отвечающих стандартам ОС. Это еще осложняется большими размерами файлов, которые иногда занимают все ОЗУ без остатка, включая экран и область системных переменных ОС, что приводит к ее блокированию.

Кроме того, ряд фирменных программ имеет специальную защиту от копирования и модификации, нередко очень изощренную. Если вы сами написали какую-либо программу на БЕЙСИКе, то выполнить ее копию несложно. Для этого достаточно дать команду SAVE "имя" и нажать ENTER. Если вы хотите, чтобы программа автостартовала после загрузки, начиная со строки М, то команда имеет вид : SAVE "имя" LINE М. Если в программе

есть блоки, записанные в машинном коде, то дается команда SAVE "имя" CODE M,N, где M - адрес, с которого начинается блок кодов, а N - длина этого блока в байтах. При загрузке программ блоки, записанные на БЕЙСИКе индицируются на экране как PROGRAM..., а блоки машинных кодов как BYTES... Если вы выгружаете свою разработку, то вы, конечно, знаете те адреса, в которых располагаются ваши блоки. Однако, если имеете дело с чужой (фирменной) программой, то загрузив ее, вам не так то просто ее выгрузить. Во-первых, она автостартует и остановить ее очень сложно. Во-вторых, вам неизвестны адреса блоков, из которых она состоит.

Упростить процесс копирования и автоматизировать его можно с помощью специальных копирующих программ. Эти программы имеют свою нестандартную загрузочную систему. Таких программ очень много.

Список программ-копировщиков:

COPIER FM-3
COPY-COPY
COPY NEW 1
MR COPY
COPY 86/M
COPY DELUXE
PIRATE 02
PIRACY
ZOTYOCOPY+
MICROCOPY
ZK COPY-87
YF COPY-86
BAUDCOPY
WA COPY
COPY COPY COPY
MONSTER
S-COPY
OUT COPY
SINCLAIR COPY

Мы здесь рассмотрим несколько наиболее широко распространенных.

COPY 86/M

Это, по-видимому, наиболее удобный и наглядный копировщик общего применения. В исходном состоянии он имеет обьем свободного пространства 45 000 байтов, и способен компрессировать данные при загрузке. Компрессирование состоит в том, что например вместо последовательности из N нулевых байтов записывается один "нуль", а следующий байт указывает, сколько их (N). Это позволяет загружать в копировщик значительно более 45 К. Особенно много места экономится при загрузке экранов, т.к. в графических изображениях часто встречаются длинные последовательности нулей.

Результат компрессии данных отправляется на хранение в область экрана, что изображается появлением на экране полос и точек, напоминающих телеграфный код.

Сводка команд программы COPY 86/M.

BREAK - переход в исходное положение.
L - загрузить файлы.
C - выгрузить (скопировать) файлы. Выгрузка производится с паузой 1.5 сек. между записями.
M - то же, но с паузой 3 сек.
V - VERIFY - проверка выгруженных записей.
D - DELETE - удаление ненужных записей.
H - перевод в шестнадцатиричную систему.
X - просмотр записей с отбоем ранее поданных команд.
B - то же, но без отбоя команд.
P - просмотр БЕЙСИК-программы.
A - ALL - после этой команды очередная команда будет одновременно распространяться на все записи.

Все команды подтверждаются нажатием ENTER.

На экране файлы записываются в "окне", имеющем 4 строки. Если записей больше чем 4, то одновременно их вывести на экран нельзя, поэтому и необходимы клавиши B и X для "протягивания" записей через "окно".

При использовании программы следует помнить, что она может не работать с некоторой периферией. Так она работать не будет, если к компьютеру подключен кемпстон-джойстик.

К сожалению, эта программа не работает и со многими самодельными компьютерами. В таком случае, как правило, пользуются другой компрессирующей программой TF COPY.

TF COPY

После загрузки программы на экране появляется исходное меню. Нажмите "0", программа стартует и перед вами появится основное меню программы, которое выглядит так:

LOAD SAVE DELETE VERIFY MODE

Нажав первую букву нужной команды, вы входите в нужный вам режим.

LOAD - загрузка файлов.
SAVE - выгрузка файлов.
DELETE - удаление файлов.
VERIFY - проверка выгруженных файлов.
MODE - переключение режима. В программе имеются 3 режима работы.
Режим 1 - 41984 байта свободной памяти.
Режим 2 - 44032 байта свободной памяти.
Режим 3 - 44288 байта свободной памяти.

Все команды должны завершаться нажатием ENTER.

Обратите внимание, что при переходе на другой режим происходит очистка памяти, т.е. загруженная информация будет уничтожена.

После выбора одного из вышеуказанных режимов программа выдает запрос. Например, нажатие "D" дает режим DELETE и появляется запрос DELETE FROM ... TO ... . Удалить записи с ... по ... .

В ответ надо дать номера тех записей, которые вам нужны. Например, если вы хотите уничтожить записи с третьей по шестую, то соответственно надо ввести 3 и 6.

COPY-COPY (PIRATE 02)

Эта программа находится в эксплуатации довольно давно и представляет классический образец копировщика. Она не выполняет компрессирование и имеет довольно ограниченный обьем свободной памяти (42K), но в ней есть несколько оригинальных вспомогательных режимов, которые в ряде случаев делают ее незаменимой.

Сводка команд.

Все команды являются ключевыми словами "СПЕКТРУМа" и потому не набираются по буквам. Они требуют завершения нажатием клавиши ENTER.

LOAD - выполнить загрузку очередной записи (очередного блока).
LOAD N TO M - выполнить загрузку записей, начиная с номера N до номера M.

При этом ранее существовавшие записи с номером N и выше будут уничтожены, т.е. LOAD 1 уничтожит все ранее существовавшие записи и подготовит компьютер к загрузке новых.

Возможные вариации этой команды:

LOAD N TO
LOAD TO M

SAVE - выполнение загрузки записи (записей) на ленту. Может иметь форму:

SAVE N TO M
SAVE TO M
SAVE N TO
SAVE N TO M STEP K

Параметр K после оператора STEP показывает величину паузы в секундах, которую компьютер выдержит между отдельными блоками.

VERIFY - проверка выгруженных записей. Имеет все те же формы, что и команда LOAD.
LOAD AT ADDR - загрузка блока, начиная с адреса ADDR.
Например: LOAD AT 16384 - загрузка экрана.
LIST ADDR - выдача на экран содержимого памяти, начиная с указанного адреса. LIST = LIST 0.

POKE ADDR,N - поместить по указанному адресу байт N.

Этой командой часто пользуются совместно с командой LIST. Они позволяют вносить изменения в машинный код программы. В принципе с подобными задачами лучше справляются специализированные программы для отладки машинного кода (их часто называют МОНИТОРЫ), но наличие такой функции в копировщике делает его весьма удобным.

COPY 16384 - команда, которая позволяет копировать блоки длиной 49К.

Это режим делает данную программу незаменимой во многих случаях. Дело в том, что если программа имеет один значительный блок длиной порядка 49К и не поддается компрессии, то откопировать его копировщиками типа COPY 86/M и TF COPY не удается, т.к. он не помещается в памяти. (Например, программы FIRELORD и URIDIUM фирмы HUSON CONSULTANTS и другие). В режиме COPY 16384 программа COPY-COPY удаляет сама себя из памяти и оставляет только небольшой блок длиной несколько байтов, предназначенный для выгрузки загруженного блока. Выгрузка выполняется нажатием клавиши CAPS SHIFT. Возможна только однократная выгрузка, после этого копировщик необходимо снова загрузить. Разумеется магнитофон должен быть включен на запись до нажатия CAPS SHIFT.

Дополнительные сведения.

Мы рассмотрели только три наиболее распространенные копирующие программы. На самом деле их очень много, но принципы их работы отличаются не очень значительно. В то же время надо помнить, что с помощью копировщиков можно только копировать программы, в которых не принято специальных мер для защиты от копирования. Многие фирмы такие меры применяют. Это например "спидлок" (ускоренный загрузчик), джеркитон (пилоттон, сопровождающийся рывками), блоки избыточной длины (более 50К), фальшхэдеры (блоки длиной 17 байтов, которые воспринимаются копировщиком как заголовок очередного блока, в то время как они таковыми не являются), укороченный пилоттон, слишком широкий или слишком узкий пилоттон, блоки с замеряемой паузой между ними и многие другие методы. Принцип их действия состоит в том, что первый блок программы является БЕЙСИК-загрузчиком, который подготавливает загрузку второго блока. Второй блок в машинных кодах подготавливает нестандартную загрузку прочих блоков, которые уже не могут быть загружены никуда, где предварительно не отработал второй блок, в том числе и в копировщик. Копирование таких программ довольно трудоемкий процесс, но со многими защитами справляется например специальная программа LERM-7 (TC-7).

Другой подход к таким программам состоит в том, что используются специальные периферийные устройства. Их назначение - остановить ("заморозить") программу в каком-либо месте, а затем сделать полный дамп памяти (полную выгрузку всей памяти). Широко известны устройства типа MULTIFACE, позволяющие кроме этого производить загрузку экрана в любом месте программы, вносить в программу изменения и продолжать исполнение программы с места прерывания. Аналогичными устройствами снабжаются иногда и дисковые системы, например БЕТА-ДИСК ИНТЕРФЕЙС (TP ДОС) фирмы TECHNOLOGY RESEARCH. Он имеет специальную кнопку, называемую MAGIC BUTTON (волшебная кнопка) для прерывания программы и сброса ее на диск. Правда, запустить откопированную таким образом программу через LOAD не удается. Для этого интерфейс реализует нестандартный подход и выполняет это по команде GO TO "имя".

Можно предположить и такой способ копирования с помощью двух магнитофонов. Один магнитофон включается на вход компьютера, а другой - на выход. Компьютер считывает информацию бит за битом и тут же транспортирует ее на выход, при этом он выполняет роль компаратора и выравнивает фронты сигналов.

Правда, этот метод не очень надежно работает на фирменных "СПЕКТРУМах", т.к. у них вход и выход имеют электрическую связь, но на компьютерах "ТАЙМЕКС" и на большинстве самодельных дает отличный результат.

ПРИЛОЖЕНИЕ 6

ЧТО НУЖНО ЗНАТЬ ДЛЯ ПОДКЛЮЧЕНИЯ ПРИНТЕРА

По-видимому, принтер - это наиболее желаемое периферийное устройство, перспективу возможности приобретения и подключения которого к СПЕКТРУМу имеет в виду большинство пользователей. К СПЕКТРУМу можно подключить практически любой принтер, но при этом могут возникать те или иные трудности.

Прежде всего надо определиться для чего он вам нужен. Если основное направление вашей работы - программирование, то вам может подойти небольшой узкопечатный ZX-принтер, который подключается к разъему компьютера без каких-либо интерфейсов и не требует никакого программного обеспечения для работы (все уже содержится в ПЗУ компьютера). Такой принтер имеет ширину бумажной ленты 10...13 см и выводит 32 символа в строке. Он также может выполнять графическую копию экрана. Качество печати далеко не идеальное, но такие устройства сравнительно дешевы, просты и удобны в работе и, если их применять в основном для распечатки текстов программ или результатов расчетов, то вполне соответствуют поставленным задачам.

Если же вам необходимо получение качественных документов на листах стандартного формата, необходимо применять полноразмерные принтеры. Полноразмерные принтеры выпускаются двух типов - это точечно-матричные принтеры и принтеры типа "ромашка". Разница в конструкции печатающего элемента. В первом случае это набор иголок, оставляющий через красящую ленту мозаичный отпечаток на бумаге. Из точек складывается изображение символа. В принтерах второго типа печатающие литеры закреплены на вращающемся прорезном диске ("ромашке"). Этот диск может быть сменным для замены набора литер.

Разница в конструкции вызывает и различия в эксплуатации. Матричные принтеры значительно быстрее работают, могут гибко (программно) перестраиваться с одного шрифта на другой, но по качеству печати только самые лучшие экземпляры могут приближаться к принтерам типа "ромашка".

В основном принтеры "ромашка" применяются в тех случаях, когда имеется необходимость в большом количестве деловой переписки, и качество исполненного документа свидетельствует об уровне представительства. Надо также отметить, что выдача графических изображений на принтерах этого типа исключена.

Наиболее гибким и многофункциональным устройством является точечно-матричный принтер. Далее мы остановимся на этих устройствах и отметим, какие вопросы вы должны для себя решить, прежде чем приобретать и подключать такой принтер.

ТИП ИСПОЛЬЗУЕМОГО ИНТЕРФЕЙСА.

Этот пункт является весьма важным, т.к. от него зависит способ подключения к компьютеру. Существуют два основных метода обмена информацией между компьютерами и периферийными устройствами - параллельный и последовательный. При параллельном подключении все биты, составляющие байт посылки, передаются одновременно по параллельным шинам данных. При последовательном байт передается бит за битом по очереди. Исторически за интерфейсами, реализующими параллельный метод передачи, закрепилось название "Центроникс", а за последовательными - RS232.

Большинство современных компьютеров уже имеют встроенный интерфейсный порт того или иного типа, но СПЕКТРУМ-48 его не имеет. Поэтому, приобретая принтер, надо подумать и об интерфейсе. Наибольшее распространение получили следующие модели интерфейсов:

1. ИНТЕРФЕЙС-1. Мы о нем уже упоминали. Он имеет последовательный порт RS232, к которому может подключаться принтер, имеющий такой же интерфейс. Никакой программной поддержки этот интерфейс не требует, она находится в его ПЗУ.
2. ZX-LPRINT III. Это интерфейс может настраиваться и поддерживает аппаратуру как с интерфейсом "Центроникс", так и с RS232.
3. TASPRINT - интерфейс типа "Центроникс". Требует специальной программы (драйвера) для своей работы.

EPSON

4. KEMPSTON-E - интерфейс типа "Центроникс", также требует драйвера.

Компьютеры "СПЕКТРУМ+128" и "СПЕКТРУМ+2" имеют уже встроенный порт RS232. Зато они не могут работать с ZX-принтерами, т.к. область буфера принтера там занята дополнительными системными переменными, хотя существуют программные пути устранения этого недостатка.

Обратим ваше внимание однако на то, что использовать это порт непросто. Дело в том, что он поддерживается только в режиме 128К, а наибольшая часть прикладных программ (текстовые редакторы, графические редакторы и пр.), с которыми хотелось бы использовать принтер, работают в режиме 48К.

НАЛИЧИЕ ДОПОЛНИТЕЛЬНОГО НАБОРА ЗНАКОВ.

Далеко не все импортные принтеры имеют встроенный русский набор знаков, что необходимо учитывать. Наилучшие модели имеют возможность загрузки шрифта от компьютера.

ПРИМЕНЯЕМАЯ БУМАГА.

Некоторые принтеры используют термобумагу, что создает весьма ощутимые неудобства в связи с отсутствием ее в широкой продаже. Кроме того, качество исполнения документа в этом случае ниже.

КРАСЯЩАЯ ЛЕНТА.

Многие принтеры работают с лентой, находящейся в специальных неразборных кассетах. Это дает удобство при смене ленты, но может поставить вас в тупик, если доступа к таким кассетам вы не имеете и применение обычной ленты на катушках конструктивно невозможно.

СКОРОСТЬ ПЕЧАТИ.

Если вы предполагаете исполнение многочисленных копий документов, этот параметр может быть важным. Обычно точечно-матричный принтер имеет скорость печати 80...100 зн./сек. Даже если этот параметр вам безразличен, имейте в виду, что чем он выше, тем совершеннее принтер. Наилучшие модели имеют скорость выше 180 зн./сек.

КОЛИЧЕСТВО ИГОЛОК В ПЕЧАТАЮЩЕЙ ГОЛОВКЕ.

Как правило, принтер среднего качества имеет 9 иголок в печатающей головке. Если у него только 7 иголок, то некоторые буквы, например y, j, p могут лишиться своих нижних элементов. Наиболее совершенные модели имеют до 24 иголок.

НАЛИЧИЕ РЕЖИМА NLQ.

Режим NLQ - NORMAL LETTER QUALITY обеспечивает печать символами, приближающимися по качеству к стандартной пишущей машинке. Такой режим имеется далеко не у всех принтеров, и печать в этом режиме выполняется значительно медленнее, зато он позволяет получать представительные документы.

КОДЫ УПРАВЛЕНИЯ ПРИНТЕРОМ.

Принтер может не только распечатывать текст, который поступает в него из компьютера, но выполнять дополнительно и целый ряд специфических действий (подчеркивать текст, переключаться с одного набора знаков на другой, изменять размеры шрифта, выполнять печать двойным ударом и еще многие другие действия). Эти переключения делаются с помощью управляющих кодов, которые в нужных местах вставлены в текст. Как правило, управляющие коды состоят из последовательности кодов, начинающейся с символа 27 (ESCAPE), поэтому часто коды управления принтером называют ESCAPE-кодами. В зависимости от того, какому коду какое действие принтера соответствует, различают несколько стандартов этих кодов. Наиболее широко распространена система кодов фирмы EPSON, которая применяется на принтерах этой фирмы. В зависимости от того, поддерживает ли принтер этот стандарт кодов, он может быть или не быть EPSON-совместимым. Желательно, чтобы такая совместимость была, тогда вы сможете распечатывать подготовленные документы не только

на своем принтере, а также упростить задачу настройки фирменных программ на работу с принтером, т.к. они, как правило, готовятся в расчете на EPSON-совместимые принтеры.

DIP- ПЕРЕКЛЮЧАТЕЛИ.

Эти переключатели служат для задания определенных режимов работы принтера. Многие их функции дублируются и могут быть изменены кодами управления принтером, но те условия, которые заданы этими переключателями и являются исходными в момент включения принтера. Чем их больше, тем больше возможностей имеет принтер. Нормально их должно быть порядка 24 шт.

БУФЕР ПРИНТЕРА.

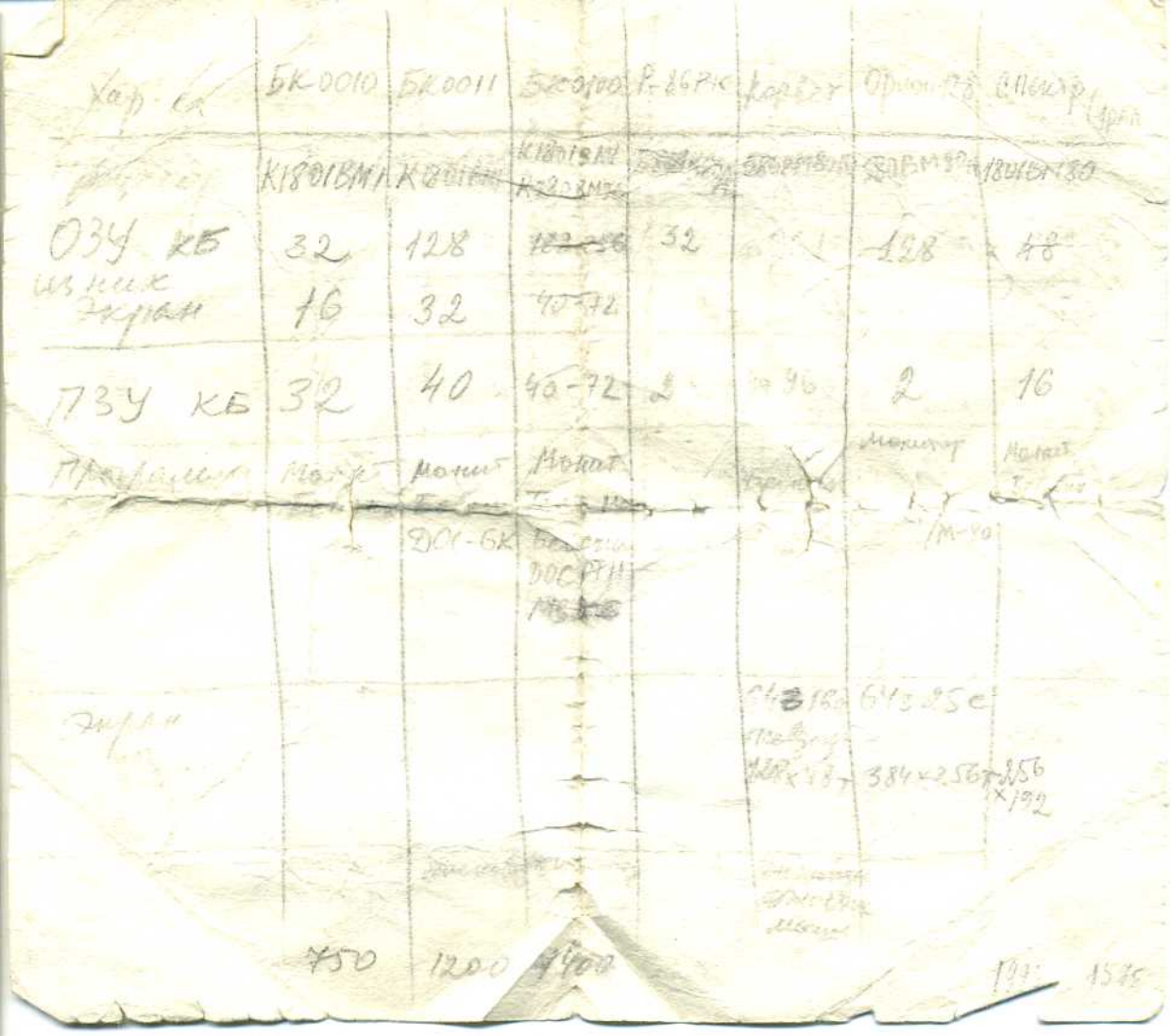
Принтер должен иметь определенный обьем свободной оперативной памяти, где он размещает пакет, поступающий от компьютера, и выполняет с ним преобразования (перед выводом на печать). Именно наличие буфера позволяет принтеру, например, печатать и при обратном проходе головки (справо налево), что вдвое повышает его быстродействие. При наличии достаточно емкого буфера компьютер очень быстро может перебросить в него выводимый текст, и вы сможете, например, готовить к печати очередной блок, пока принтер ведет печать. Если же емкость буфера мала, компьютеру приходится все время подзагружать буфер, и вы можете использовать его для других целей. В принципе, для большинства практических задач емкость порядка 0.5К достаточна, но лучше, если она больше 2К, а совершенные модели имеют буфер больше 4К.

ВЫБОР ТЕКСТОВОГО РЕДАКТОРА.

Текстовые редакторы представляют наиболее удобные условия для работы с принтером, поэтому, предполагая подключение принтера, надо иметь в виду и необходимость наличия редактора. Наиболее широко распространены различные версии текстового редактора TASWORD. Эта программа отличается простотой в работе, предоставляет возможность изображения на экране как 32 символов в строке, так и 64-х. Имеет несколько кодов управления принтером, которые вставляются в текст в виде символов блочной графики. Достаточно просто переделывается на работу с русским или другим национальным шрифтом.

Наиболее совершенным и гибким в работе является редактор THE LAST WORD 2. Среди программ для бытовых компьютеров любых моделей трудно найти сравнимый по возможностям. Он может воспроизводить текст на экране в режимах 40, 48, 60 и 80 знаков в строке. Имеет очень удобное меню настройки условий печати, обширный набор операторов управления принтером, который к тому же может гибко перестраиваться без выхода из программы. Имеет встроенный таймер и калькулятор, средства поиска информации и мощные средства для оформления текста и составления комплексных документов.

Строго говоря, это не просто программа, а пакет программ, т.к. к нему прилагаются дополнительные файлы, позволяющие перенастроить редактор на работу в дисковых системах "TR DOS", "OPUS", "GORDON" и др., а также дополнительные программные средства для обеспечения возможности работы с текстами, выполненными в других редакторах и базах данных.



1. Предлагаю рабочие платы:
   а) Ленинград-1
   б) Пентагон-1
2. Настраиваю:
   а) Ленинград-1
   б) Пентагон-1

Тел. 458-52-25. 20.00
Спросить Павла 24.00

Тел. 458-52-25. 20.00
Спросить Павла 24.00

Корпуса

233 × 144 × 38 ZX-Sp

310 × 260

400 × 300 Пиок

отв. для плавная

340 + 45 × 95

52609230 доб 28 08

Богданов Пл. др. Влад

Фрезин Мир БД

Бульв. кл 97

5462 ~~Тул~~ Басманн

25774 Пушк Ул. Д.К.

стр 14

Адреса с 23296 по 23551 область буфера принтера

стр 15

Адреса с 23552 по 23733 (182 байта)

область системных переменных.

стр 16

23633 адрес данных для ввода-вывода

стр 17

23677 X } координаты точки графопостроителя

23678 Y }

23679 позиционирование принтера

23680. Мл. байт адр. по з LPRINT для печати

стр 18.

Область информац. каналов

адреса 23631 и 23632. - CHANS.

канал „P" обеспеч. вывод на принтер.

Обл. 23734 по 23754.

инф

каналы

стр 67

порт 251 обеспечивает связь с принтером как чтение, так и запись.

стр 111.