〈СОЛОН〉

СПРАВОЧНИК ПО СИСТЕМНЫМ ПРОГРАММАМ

МОСКВА

Данное справочное пособие предназначено для пользователей, уже имеющих опыт работы на компьютерах типа ZX SPECTRUM. Суть составления этой книги - уменьшение количество общих фраз и оставить лишь самую необходимую для работы информацию по языкам БЕЙСИК, АССЕМБЛЕР, ПАСКАЛЬ, а также по наиболее распространенным системным и сервисным программам, таким как программы копирования, графические редакторы и т.д.

При составлении авторы использовали материалы зарубежной печати, самиздатовскую литературу, книги Виккерса, Брайна, Анжела, Джеймса и др.

1402000000
С---------- без объявл.
А50(02)

ISBN 5-7981-0002-2

К Л А В И А Т У Р А

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

Режим отображается курсором, мерцающей буквой, указывающей позицию, в которую будет вводиться очередной символ с клавиатуры. Возможны следующие режимы:

K - (для ключевых слов) KEYWORDS.

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

L - (для букв) LETTER.

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

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

C - (для заглавных букв) CAPITAL.

Режим представляет собой вариант режима L, в котором используются заглавные буквы. CAPS LOCK используется для перехода из режима LBC и обратно.

E - (для расширения) EXTEND.

Используется для получения дальнейших символов, главным образом знаков. Этот режим вводится одновременным нажатием двух переключающих клавиш, с удержанием затем только одной клавиши. В этом режиме клавиша дает один символ или знак (изображенный на зеленом поле клавиши), если не нажата переключающая клавиша, или знак, изображенный на красном поле, если удерживается переключающая клавиша. Цифровые клавиши выдают знак, если нажимаются вместе с SYMBOL SHIFT, в противном случае они выдают последовательность управления цветом.

G - (для графики) GRAPHICS.

Режим вводится после нажатия GRAPHICS (CAPS SHIFT и 9) и сохраняется до следующего нажатия этой клавиши. Цифровые клавиши будут выдавать мозаичные графические символы, сохраняя GRAPHICS, выдавать определенный пользователем графический символ. Если некоторая клавиша будет удерживаться более 2-3х сек., это вызовет повторение производимого ею действия. Ввод с клавиатуры производится в нижнюю часть экрана. Каждый символ (или составной знак) вставляются перед курсором. Курсор может быть переслан влево действием CAPS SHIFT и 5, а вправо CAPS SHIFT и 8. Символ перед курсором можно удалить командой: DELETE(CAPS SHIFT и 0. Целая строка может быть удалена вводом EDIT (CAPS SHIFT и 1) и последующим ENTER. Когда нажимается ENTER, выполняется набранная строка, либо она вводится в программу, либо она используется как входные данные для INPUT-оператора, либо в строке имеются синтаксические ошибки. В этом случае нажатием [?] происходит переход на ошибку.

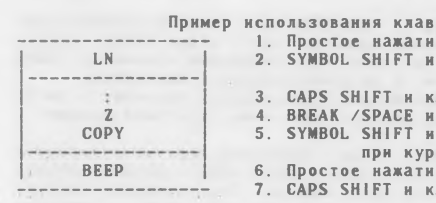
Когда строки программы введены, листинг отображается в верхней части экрана.

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

Если выполняется команда или целая программа, то результаты отображаются в верхней половине экрана и сохраняются до ввода строки программы, ввода пустой строки или нажатия клавиши управления курсором вверх или вниз. В нижней части выдаются сообщения и коды, приведенные в приложении. В сообщении указывается номер ошибочной строки (0 для команды) и позиция оператора в этой строке. Сообщение сохраняется на экране до нажатия любой клавиши (отображается переходом в режим K).

В определенных обстоятельствах CAPS SHIFT и SPACE действуют как BREAK, останавливая компьютер с сообщениями D или L, и при этом до останова: завершается выполнявшийся оператор или завершаются действия, выполняемые компьютером с маг-

нитофоном или принтером.



Э К Р А Н Т Е Л Е В И З О Р А

Экран телевизора содержит 24 строки по 32 позиции в каждой и делится на 2 части. Верхняя часть в 22 строки отображает листинг или вывод из программы. Когда вывод в верхней части достигает низа, необходима свертка на одну строку. При этом может захватываться строка, которую вам хочется сохранить. Компьютер в этом случае останавливается с запросом "SCROLL?". Если теперь нажать клавиши N, SPACE или STOP, то программа остановится с выдачей сообщения: "D BREAK-CONT REPORTS". Нажатие других клавиш разрешает свертку и продолжение выполнения.

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

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

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

Края могут быть установлены в определенный цвет использованием оператора BORDER.

Каждая позиция подразделяется на 8x8 точек, а графика символов обеспечивается индивидуальным определением каждой точки. Атрибуты каждой позиции настраиваются при записи символа или при установке точки (PIXEL). Способ настройки определяется параметрами вывода, имеющими 2 установки (постоянную и временную) в 6 операторах: PAPER, INK, FLASH, BRIGHT, INVERSE и OVER. Постоянные параметры для верхней части экрана устанавливаются в операторах PAPER, INK и т.д. Обычно они имеют черный цвет для закрашенной точки (INK) и белый для фоновой (PAPER), нормальную яркость, немерцающие, неинверсные. Постоянные параметры для нижней части экрана используют цвет рамки (BORDER COLOUR) как цвет фона (незакрашенный), с черным или белым цветом, нормальную яркость, немерцающие.

Временные параметры устанавливаются командами:

PAPER, INK и т.д., вставляемыми в операторы PRINT, LPRINT, INPUT, PLOT, DRAW и CIRCLE, а также PAPER, INK и тому подобными управляющими символами, когда они выводятся на телевизор.

Временные параметры сохраняются до конца действия оператора PRINT.

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

0-черный (BLACK) 1-голубой (BLUE) 2-красный (RED)
3-фиолетовый 4-зеленый (GREEN) 5-синий (CYAN)

6-желтый (YELLOW) 7-белый (WHITE)

Параметр 8 определяет, что цвет должен оставаться при выводе без изменения.

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

Параметры FLASH и BRIGHT могут принимать значения 0, 1 или 8.

Параметр 1 указывает, что включается повышенная яркость и мерцание.

Параметр 0 указывает, что повышенная яркость и мерцание отключаются.

Параметр 8 указывает, что все остается без изменения.

Параметры OVER и INVERSE могут принимать значения 0 или 1.

- OVER 0 – новый символ затирает старый.

OVER 1 – код старого символа и нового символа соединяются операцией 'исключающего или', образуя новый символ (OVERPRINTING)

INVERSE 0 – новый символ печатается в ненверсном (позитивном) виде

INVERSE 1 – новый символ печатается в инверсном (негативном) виде.

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

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

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

ПРИНТЕР

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

а) когда окончен вывод одной строки и вывод переходит к другой строке,

б) при передаче в буфер символа ENTER;

в) при завершении программы, если еще остались другие невыведенные данные;

г) если встретились управляющие символы TAB или запятая, требующие перевода строки.

Управляющие символы TAB и запятая производят вывод пробелов при работе с телевизором. Управляющий символ AT изменяет позицию вывода, используя число, задающее позицию.

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

При вводе BREAK принтер останавливается с выдачей сообщения 'B'. При отсутсвии принтера вывод просто не осуществляется.

ПОРТЫ

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 полуряду:

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 5 клавиш в данном полуряду. D0 -для крайней клавиши, а D4 -для ближней к центру. Состояние одного из этих битов 0 указывает, что соответствующая ему клавиша нажата. D6 принимает свое значение при чтении с разъема 'EAR'.

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

Порт 251 обеспечивает связь с принтером, как чтение, так и запись. Чтение – для проверки готовности принтера к работе. Порты 254, 247, 239 используются для связи с дополнительными устройствами.

ЯЗЫК ПРОГРАММИРОВАНИЯ БЕЙСИК

Все числа в системе могут иметь точность 9 или 10 знаков. Наибольшее число 10**38, а наименьшее положительное число 4*10**(-39). Числа имеют внутреннее представление как числа с плавающей (двоичной) точкой, с выделением одного байта на показатель степени 'e' (экспоненты) в интервале от 1 до 255, и четырех

байтов на мантиссу 'M' в интервале от 0.5 до 1 (M=1). Это представляется числом M*2**e(-128).

Поскольку 1/2<=M<1, старший бит мантиссы всегда 1. Следовательно, мы можем заменить 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).

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

Управляющие переменные для FOR-NEXT циклов имеют имена длиной в одну букву. Числовые массивы имеют имена длиной в одну букву, которая может быть такой же, как имя скалярной переменной. Эти массивы могут иметь произвольное количество измерений и произвольный размер. Начальный индекс всегда 1. Строки символов более гибкие в своей длине. Имя строковой переменной, в отличие от простой переменной, заканчивается символом доллара ($).

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

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

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

а) пустым;

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

*) строковых выражениях (сечениях);

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

В случае *), строка выражения имеет значение S$.

Если сечение массива пусто, то S$ считается подстрокой от самой себя.

Если сечение представлено в форме B 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 (смотри оператор LET). Если часть строки записывается в строковой литерал, она должна удваиваться.

1.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 N 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 A AND B = B, если B=0

	-Строка, тогда------->	A$ AND B = A$, если B<>0 A$ AND B = '' '', 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, заключаемые в скобки	Число, двоичный код которого, представляет собой атрибуты Y-ой позиции X-ой строки экрана. Бит 7 (старший) равен 1 для мерцающего поля, и 0 для немерцающего. Биты с 5 по 3-цвет фона. Биты с 2 по 1 - цвет закрашивания. Выдает сообщение B, если 0<=X<=23 и 0<=X<=31.
BIN		Это необычная 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 рад.	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).
INKEY$	Нет	Чтение с клавиатуры. Возвращает символ введенный с клавиатуры (в режиме [L] или [C], если было действительное нажатие клавиши, иначе - пустую строку.
INT	Число	Округление к ближайшему меньшему целому.
LEN	Строка символ	Длина строки
LN	Число	Натуральный логарифм. Выдает сообщение A, если X <=0
NOT	Число	0, если X<>0, 1, если X=0. Операция имеет приоритет 4
OR	Логическая операция. Оба операнда числа	Операция имеет второй приоритет. A OR B = 1, если B<>0 A OR B = A, если B<>0
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	1, Если точка экрана с координатами (X,Y) закрашена. 0, если эта точка имеет цвет фона. Выдает сообщение

	Y, заключенных в скобки	B, если не выполняются условия 0<=X<=255 и 0<=Y<=175.
RND	Нет	Очередное псевдослучайное число из последовательности, получаемой возведением в 75 степень модуля числа 65537, вычитанием 1 и делением на 65536. Число лежит в интервале 0<=Y<=1.
SCREEN$	Два числовых аргумента X и Y, в скобках.	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 рад.	Синус X
SQR	Число	Корень квадратный. Выдает сообщение A, если X<0
STR$	Число	Строка символов, которая должна быть отображена, если X выводится.
USR	Число	Вызывает подпрограмму в машинных кодах, начальный адрес которой X. При возврате результатом будет содержимое регистровой пары BC.
USR	Строка символов	Адрес группы байтов, задающих определенный пользователем символ для закрепления его за X
VAL	Строка символов	Вычисление X как числового выражения. Выдает сообщение C, если X содержит синтаксические ошибки или дает строковое (нечисловое) значение.
VAL$	Строка символов	Вычисляет X как строковое выражение. Выдает сообщение C, если X содержит синтаксическую ошибку или дает нестроковое (числовое) значение.

2. ОПЕРАЦИИ

Префиксные:

- число стрицательное 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
<= меньше или равно
<> не равно

Оба операнда должны быть одного типа. Результат равен 1, если нет.

Функции и операции имеют следующий приоритет:

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

3. 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 ',' или ';'. Цветовой символ имеет форму операндов: PAPER, INK, FLASH, BRIGHT, INVERSE или OVER.

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

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

Команда или строка программы может содержать несколько операторов, разделенных двоеточием ':'.

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

Все операторы языка сведены в следующую таблицу:

ОПЕРАТОР	ДЕЙСТВИЕ ОПЕРАТОРА
BEEP X,Y	Воспроизводит звук длительностью X сек. и высотой Y полутонов вверх от основного тона до (или вниз, если Y отрицательное).
BORDER M	Устанавливает цвет рамки (бордюра) экрана. Выдает сообщение об ошибке K, если 0> M >Y.
BRIGHT M	Устанавливает яркость выводимого символа: 0-для обычной яркости; 1-для повышенной яркости; 8-сохраняет существующую яркость.
CAT	Без MICRODRIVE не работает.
CIRCLE X,Y,Z	Изображает дугу или окружность с центром в точке с координатами (X,Y) и радиусом Z.
CLEAR	Уничтожает все переменные и очищает занимаемую ими память. Выполняет RESTORE и CLS, устанавливает PLOT позицию в нижнюю левую точку экрана и очищает GO SUB стек.
CLEAR N	Подобно CLEAR, но дополнительно изменяет системную переменную RAMTOP на 'N' и задает новый GO SUB стек.
CLOSE#	Без MICRODRIVE не работает.
CLS	(CLEAR SCREEN) очищает файл экрана.
CONTINUE	Продолжает выполнение программы, начатой ранее и остановленной 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
DATA E1,E2,E3,...	Часть списка данных. Должна располагаться в программе.
DEF FNA(A1,A2,....,AK)	Определяемая пользователем функция. Должна располагаться в программе. Или буквы и $ для строковых аргументов, значений. Используется форма DEF FNA(), если нет аргументов.
DELETE F	Без MICRODRIVE не работает.
DIM A(N1,N2,....,NK)	Уничтожает массив с именем 'A' и устанавливает числовой массив 'A' с 'K' измерениями, присваивает всем его элементам значение 0.
DIM A$(N1,N2,....,NK)	Уничтожает массив или строку с именем A$ и устанавливает символьный массив с 'K' измерениями, присваивает всем его элементам значение "S". Массив может быть представлен как массив строк фиксированной длины NK,с K-1 размерностью. Сообщение 4 выдается,если недостаточно места для размещения массива.Массив не определен до его описания в операторе DIM.
DRAW X,Y	То же самое, что и DRAW X,Y,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.
ERAZE	Без MICRODRIVE не работает.
FLASH N	Определяет:будет ли символ мерцающим или с постоянным свечением. N=0 для постоянного свечения, N=1 - для мерцания,N=8 -для сохранения предыдущего сост-я.
FOR A=X TO Y	FOR A=X TO Y STEP 1
FOR A=X TO Y STEP Z	Уничтожает скалярную переменную 'A' и устанавливает управляющую пепеременную '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 NEXTA или выдача сообщения 1, если нет (см. NEXT). Сообщение 4 выдается, если недостаточно места для размещения управляющей переменной.
FORMAT F	Без MICRODRIVE не работает.
GO SUB N	Проталкивает строку с оператором GO SUB в стек для использования затем как GO TO N. Выдается сообщение

	4. если не все подпрограммы завершились с RETURN.
GO TO N	Продолжает выполнение программы со строки 'N'. Если 'N' опущено, то с первой строки после этой.
IF X THEN S	Если 'X' истинно (не равно 0), то выполняется 'S'. 'S' включает все операторы до конца строки. Форма
	'IF X THEN номер строки' недопустима.
INK N	Устанавливает цвет закрашивания (т.е. цвет, которым будут изображаться символы на цвете фона). 'N' в интервале от 0 до 7 указывает цвет. N = 8 - оставить цвет без изменений, N=9 - увеличение контраста. Выдает сообщение K, если 'N' не лежит от 0 до 9.
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 также, как и в операторе. Все выводятся в нижнюю часть экрана. В случае б) компьютер останавливается и ждет ввода некоторого выражения с клавиатуры, значение которого будет присвоено переменной. Ввод существляется обычным образом, а синтаксические ошибки выдаются мерцающим знаком вопроса [?]. 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 выво дится в обычном виде с прорисовкой цвета 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 текст программы в верхнюю часть экрана, начиная с первой строки меньшей, чем 'N', и делает 'N' текущей строкой.
LLIST	То же,что и LIST 0
LLIST N	Подобно LIST, но вывод осуществляется на принтер.
LOAD F	Загружает программу и переменные.
LOAD F DATA ()	Загружает числовой массив.
LOAD F DAT$()	Загружает строковый массив.
LOAD F CODE M,N	Загружает старшие 'N' байтов, начиная с адреса 'M'.
LOAD F CODE M	Загружает байты, начиная с адреса 'M'.
LOAD F CODE	Загружает байты по тому же адресу, с которого они были разгружены.
LOAD F SCREEN$	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, переменные и используемую память, включая и байт адреса в системной переменной RAMBOT,но сохраняет системные переменные UDG, P RAMT, RASP и PIP.
NEXT A	а) находит управляющую переменную 'A';
	в) если STEP>=0, а значение 'A'стало больше значения 'предел', или STEP<0, а значение 'A' меньше, чем значение 'предел', то происходит переход к оператору цикла
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 регистр A-'N' и выполняет команду ассемблера OUT (C)<A). 0 <= M <= 65535. -255 <= N <= 255, иначе выдается сообщение B
OVER N	Управляющий символ надпечатывания по выведенной строке. Если N=0, то выводимый символ затирает существующий в данной позиции. Если N=1, то новый сим-

	вол соединяется со старым, образуя закрашивающий цвет, при условии, что старый символ имел указание цвета, отличное от старого, или цвет фона, если оба указывают на один и тот же цвет (либо фона, либо закрашивания, сложение по модулю 2).
PAPER N	Подобен INK, но управляет цветом фона.
PAUSE N	Останавливает выполнение программы и задерживает изображение на экране на 'N' кадров (50 кадров в сек. - частота кадровой развертки) или до нажатия любой клавиши. 0 <= N <= 65535, иначе выдается сообщение B. При N=0 время задержки не учитывается и продолжается до первого нажатия клавиши.
PLOT C;M,N	Выводит точку закрашивающего цвета (обработанную OVER и INVERSE) с координатами (ABS(M),ABS(N)) смещает графическую (PLOTPOSITION) позицию. Если цветной символ 'C' не специфицирован иначе, то цвет закрашивания в позиции, где расположена эта точка, изменяется на текущий сплошной закрашивающий цвет, и другие указания (цвет фона, мерцание, яркость) остаются без изменения. 0 <= ABS(M)<= 65535,
	0 <= ABS(N)<= 175, иначе-сообщение B.
POKE M,N	Записывает значение 'N' в байт памяти по адресу 'M'.
	0 <= M <= 65535, -255 <= N <= 255,иначе сообщение B.
PRINT ...	Где '...' последовательность PRINT-символов, разделенных запятыми, точками с запятой или апострофами, которые выводятся в экранный файл для отображения на экране телевизора.Точка с запятой сама действия не вызывает, а используется для разграничения символов. Запятая порождает управляющий символ 'запятая', апостроф порождает символ ENTER.
	В конце оператора PRINT, если он не заканчивается точкой с запятой, запятой или апострофом, автоматически выводится символ ENTER. PRINT-символом может быть: а) пустая строка; б) числовое выражение. Если значение выражения отрицательное, то выводится знак минус. Если X <= 10**-5 или X >= 10**13, вывод осуществляется в показательной форме. Мантисса представляется 8 цифрами (с нормализацией) и десятичной точкой (отсутствует только тогда, когда в мантиссе одна цифра) после первой цифры. Показатель степени записывается после буквы 'e' с последующим знаком и двумя цифрами порядка.Иначе X выводится как обычное десятичное число с 8-ю значащими цифрами.
	в) строковое выражение. В строке возможны пробелы до и после символов.Не отражаемые на экране символы выводятся как '?' г) AT M,N - вывод в строку 'M', позицию 'N' д) TAB N - вывод управляющего символа TAB с последующими 2-мя байтами 'N' (первый байт-старший). Вызывает TAB-останов. е) цветовой символ в форме PAPER, INK,FLASH,BRIGHT,

	INVERSE или OVER оператора.
RANDOMIZE	То же,что и RANDOMIZE 0
RANDOMIZE N	Устанавливает системную переменную SEED, используемую для вычисления очередного значения функции RND. Если N <> 0, то SEED принимает значение 'N'. Если N=0, то SEED принимает значение другой системной переменной FRAMES, подсчитывающей кадры, отображаемые на экране, что обеспечивает вполне случайное число. Оператор запускает сокращение RAND.Сообщение B, если 'N' не лежит интервале от 0 до 65535.
READ V1,V2....,VK	Присваивает переменным одна за другой значения, последовательно представленные в списке DATA.
REM ...	Не выполняется.'...' может быть последовательностью символов (исключая ENTER). Может включать двоеточие для указания отсутствия операторов в строке с REM.
RESTORE	То же самое, что и RESTORE 0
RESTORE N	Перезаписывает указатель данных в первый оператор DATA в строке меньшей, чем 'N'. Следующий оператор READ начнет считывание отсюда.
RETURN	Ссылается на оператор GO SUB в стеке и передает управление на строку после него. Выдает сообщение 7, если нет указываемого оператора в стеке. Характерная ошибка, когда операторы GO SUB не сбалансированы операторами RETURN .
RUN	То же самое, что и RUN 0.
RUN N	CLEAR, а затем GO TO N .
SAVE F	Записывает на ленту программы и переменные.
SAVE F LINE M	Записывает на ленту программу и переменные таким образом, что при загрузке программа автоматически выполняется со строки 'M'.
SAVE F DATA ()	Запись на ленту числового массива.
SAVE F DATA $()	Запись на ленту строкового массива $.
SAVE F CODE M,N	Записывает на ленту 'N' байтов, начиная с адреса M.
SAVE F SCREEN$	Аналогично SAVE F CODE 16384,6912. Выдает сообщение F, если 'F' пустая строка или имеет длину более 10.
STOP	Останавливает выполнение программы с выдачей сообщения 9. CONTINUE (продолжение) будет осуществлятся со следующего оператора.
VERIFY	То же, что и LOAD, за исключением того, что данные загружаются в ОЗУ, но сравниваются с находящимися там. Выдает сообщение B, если обнаружен хотя бы

| один не совпадающий байт. |

4. С о о б щ е н и я

Они появляются в нижней части экрана, если компьютер остановился при выполнении некоторого оператора бейсика и указывает причину, вызвавшую останов. Сообщение содержит кодовый номер или букву. Краткое сообщение помогает найти ошибочную строку и ошибочный оператор в этой строке (команда указывается как строка 0, оператор 1 располагается в строке первым, оператор 2 следует после первого или THEN и т.д.).

От состояния CONTINUE зависит очень многое в сообщениях. Обычно сообщение начинается с оператора, специфицированного в предыдущем сообщении, но имеется исключения - сообщение 0,9,D.

Код	з н а ч е н и е	ситуация
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 адрес) 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 1 или 2 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) 23 строки в нижней половине экрана. Также встречается с PRINT AT 22,...	PRINT, PRINT AT.
6	NUMBER TOO BIG (число больше макс.допуст.) В результате вычислений получилось число больше 10**38.	Арифметич. операции.
7	RETURN WITHOUT GO SUB (RETURN без GO SUB)	RETURN

	Встретилось больше операторов RETURN, чем было операторов GO SUB.	
8	END OF FILE (конец файла)	Операции с внешней памятью.
9	STOP STATEMENT (оператор STOP) После этого сообщения CONTINUE не может повторить STOP, но может передать управление на следующий оператор.	STOP.
A	INVALID ARGUMENT (ошибочный аргумент) Аргумент функции не допустим в данной версии.	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 см. сообщение 3.	RUN, RANDOMIZE, роке, DIM, GO TO GO SUB, LIST, LLIST, PAUSE, PLOT, CHR$, PEEK, USR(с числовым аргументом)
C	NONSENSE IN BASIC (выражение не бейсика) Текст (строка) не распознается бейсиком как допустимое выражение.	VAL, VAL$
D	BREAK-CONT REPEATS Клавиша BREAK нажата во время действия периферийной операции. Действия CONTINUE после этого оператора обычные, те что указаны в операторе. Сравните с сообщением L. заны 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
F	INVALID FILE NAME (неверное имя файла) Оператор SAVE с пустой строкой вместо имени 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 NEXT.	FOR
J	INVALID I/O DEVICE (неверное устройство ввода-вывода.	в операциях с внешними устройст.
K	INVALID COLOUR (неверный цвет) Значение.	INK, PAPER, FLASH, BRIGHT, INVERSE, OVER, а также после одной из передач упр.симв.
L	BREAK INTO PROGRAM (BREAK во время выполнения программы) Нажата клавиша BREAK; это обнаруживается между двумя операторами. Строка и номер оператора в строке указывает на оператор, выполняемый перед нажатием BREAK, но CONTINUE переходит к следующему оператору.	разное
M	RAMTOP NO GOOD (адрес RAMTOP не годен) Число, указанное для RAMTOP, слишком велико Или слишком мало.	CLEAR, возможно RUN.
N	STATEMENT LOST (оператор отсутствует) Переход к оператору, которого уже нет.	RETURN NEXT CONTINUE
O	INVALID STREAM (ошибочный поток данных)	В операц. ввода-выв.
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 может быть считан.	VARIFY, LOAD, MERGE.

Использование машинных кодов

Краткое содержание: USR с числовым аргументом.

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

Программы в машинных кодах пишут обычно на ассемблере с последующей трансляцией. Транслятор с ассемблера встроен в компьютер ZX SPECTRUM.

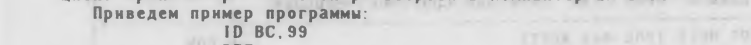
Приведем пример программы:

```
ID BC,99
RET
```

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

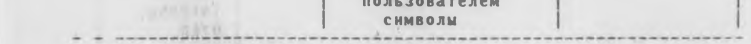
Байты 1, 99, 0 для ID BC,99 и
201 для RET

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:



RAMTOP=32599 UDG=32600 PRAMT=32767

Если вы теперь выполните: CLEAR 32499, то получите дополнительно 100 байтов памяти, начиная с адреса 32500.



32500 UDG=32600 PRAMT=32767
RAMTOP=32499

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'.

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. Книга "Искусство схемотехники" П.Хоровиц, У.Хилл МИР, 1986, том

2, стр. 579-580.

Далее приводятся 78 команд микропроцессора 8085, совместимых с микропроцессором Z80 (158 команд)./U880-ГДР, К1810ВМ-СССР/

мнемоника:	действие	код	циклы
1	2	3	4
пересылка, загрузка, запись			
MOV R,R	переслать регистр в регистр	01RR 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 A косвенно по BC	0000 0010	7
STAX D	запомнить A косвенно по DE	0001 0010	7
LDAX B	загрузить A косвенно по BC	0000 1010	7
LDAX D	загрузить A косвенно по DE	0001 1010	7
STA DD	запомнить A по адресу DD	0011 0010	13
LDA DD	загрузить A 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 A	1000 0RRR:	4[7]
ADC R	прибавить рег к A с переносом	1000 1RRR:	4[7]
SUB R	вычесть регистр из A	1001 0RRR:	4[7]
SBB R	вычесть с заемом	1001 1RRR:	4[7]
ANA R	регистр & A	1010 0RRR:	4[7]
XRA R	искл.или регистра и A	1010 1RRR:	4[7]
ORA R	регистр ! A	1011 0RRR:	4[7]
CMA R	сравнить регистр и A	1011 1RRR:	4[7]
ADI D	прибавить непоср.данные к A	1100 0110:	7
ACI D	прибавить непоср.с переносом	1100 1110:	7
SUI D	вычесть непоср. из A	1101 0110:	7
SBI D	вычесть непоср.с заемом	1101 1110:	7
ANI D	непоср. & A	1110 0110:	7
XRI D	искл.или непоср. и A	1110 1110:	7
ORI D	непоср. ! A	1111 0110:	7
CPI D	сравнить непоср. с A	1111 1110:	7
DAD RP	прибавить пару регистров к HL	00PP 1001:	11
операц.с накопителями и флагами			
RLC	сдвинуть A влево	0000 0111:	4
RRC	сдвинуть A вправо	0000 1111:	4
RAL	сдвинуть A влево чер.разр.пер	0001 0111:	4
RAR	сдвинуть A вправо чр.разр.пер	0001 1111:	4
DAA	десятич.коррекция накопителя	0010 0111:	4
CMA	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 прерываний	1111 1011:	4
DI		запрещение прерываний	1111 0011:	4
NOP		нет операции	0000 0000:	4
HLT		останов	0111 0110:	4
PUSH	RP	занести пару регистров в стек	11PP 0101:	11
POP	RP	взять пару регистров из стека	11PP 0001:	10
XTHL		обменять HL с верхом стека	1110 0011:	19
передачи управления				
JMP	DD	безусловный переход	1100 0011:	10
JCC	DD	перейти по условию CC	11CC C010:	10
CALL	DD	безусловный вызов	1100 1101:	17
CCC	DD	вызов по условию CC	11CC C100:	17(10)
RET		возврат после вызова	1100 1001:	10
RCC		возврат по условию CC	11CC C000:	11(15)
RST	N	возобновление в ячейке 8*N	11NN N111:	11
PCHL		переслать HL в PC	1110 1001:	4

Обозначения:

Поля данных

D - один байт непоср.данных (длина команды 2 байта)
DD - двухбайтовый адрес (длина команды 3 байта). Все остальные команды имеют длину 1 байт.

Циклы

N - число тактов, нужное для выполнения команды.
[N] - число тактов, когда R=M (доступ в память).
(N) - число тактов, если условие не выполнено.

Поля регистров

"R"	RRR
B	000
C	001
D	010
E	011
H	100
L	101
M	110
A	111

["M"=(HL)]

"RP"	PP
BC	00
DE	01
HL	10
PS	11
PSW	11

D) **коды условий**

"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

Полный набор символов

дес. код	Символ	шестн. код	Ассемблер. мнемоника	CBH...	EDH...
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,H	RRC (HL)	
15	не использ.	0F	RRCA	RRC A	
16	INC 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	
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	
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	

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	SRL (HL)	
63	?	3F	CCF	SRL A	
64		40	LD B,B	BIT 0,B	IN B,(C)
65	A	41	LD B,C	BIT 0,C	OUT (C),B
66	B	42	LD B,D	BIT 0,D	SBC HL,BC
67	C	43	LD B,E	BIT 0,E	LD (NN),BC
68	D	44	LD B,H	BIT 0,H	NEG
69	E	45	LD B,L	BIT 0,L	RETN
70	F	46	LD B,(HL)	BIT 0,(HL)	IM 0
71	G	47	LD B,A	BIT 0,A	LD 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	RETI
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 D,(C)
82	R	52	LD D,D	BIT 2,D	SBC HL,DE
83	S	53	LD D,E	BIT 2,E	LD (NN),DE
84	T	54	LD D,H	BIT 2,H	
85	U	55	LD D,L	BIT 2,L	
86	V	56	LD D,(HL)	BIT 2,(HL)	IM 1
87	W	57	LD D,A	BIT 2,A	LD A,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 вврх	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	61	LD H,C	BIT 4,C	OUT (C),H
98	b	62	LD H,D	BIT 4,D	SBC HL,HL
99	c	63	LD H,E	BIT 4,E	LD (NN),HL
100	d	64	LD H,H	BIT 4,H	
101	e	65	LD H,L	BIT 4,L	
102	f	66	LD H,(HL)	BIT 4,(HL)	
103	g	67	LD H,A	BIT 4,A	RRD
104	h	68	LD L,B	BIT 5,B	IN L,(C)
105	i	69	LD L,C	BIT 5,C	OUT (C),L
106	j	6A	LD L,D	BIT 5,D	ADC HL,HL
107	k	6B	LD L,E	BIT 5,E	LD HL,(NN)
108	l	6C	LD L,H	BIT 5,H	
109	m	6D	LD L,L	BIT 5,L	
110	n	6E	LD L,(HL)	BIT 5,(HL)	
111	o	6F	LD L,A	BIT 5,A	RLD
112	p	70	LD (HL),B	BIT 6,B	IN F,(C)
113	q	71	LD (HL),C	BIT 6,C	
114	r	72	LD (HL),D	BIT 6,D	SBC HL,SP
115	s	73	LD (HL),E	BIT 6,E	LD (NN),SP
116	t	74	LD (HL),H	BIT 6,H	
117	u	75	LD (HL),L	BIT 6,L	
118	v	76	HALT	BIT 6,(HL)	

119	w	77	LD (HL),A	BIT 6,A	
120	x	78	LD A,B	BIT 7,B	IN A,(C)
121	y	79	LD A,C	BIT 7,C	OUT (C),A
122	z	7A	LD A,D	BIT 7,D	ABC HL,SP
123	фигур.ск.лев	7B	LD A,E	BIT 7,E	LD SP,(NN)
124	верт.черта	7C	LD A,H	BIT 7,H	
125	фигур.ск.пр.	7D	LD A,L	BIT 7,L	
126	дефис	7E	LD A,(HL)	BIT 7,H	
127	С в окр.	7F	LD A,A	BIT 7,A	
128	o o o o	80	ADD A,B	RES 0,B	
129	o ж o o	81	ADD A,C	RES 0,C	
130	ж o o o	82	ADD A,D	RES 0,D	
131	ж o o o	83	ADD A,E	RES 0,E	
132	o o o ж	84	ADD A,H	RES 0,H	
133	o ж o ж	85	ADD A,L	RES 0,L	
134	ж o o ж	86	ADD A,(HL)	RES 0,(HL)	
135	ж ж o ж	87	ADD A,A	RES 0,A	
136	o o ж o	88	ADC A,B	RES 1,B	
137	o ж ж o	89	ADC A,C	RES 1,C	
138	ж o ж o	8A	ADC A,D	RES 1,D	
139	ж ж ж o	8B	ADC A,E	RES 1,E	
140	ж ж o ж	8C	ADC A,H	RES 1,H	
141	o ж ж ж	8D	ADC A,L	RES 1,L	
142	ж o ж ж	8E	ADC A,(HL)	RES 1,(HL)	
143	ж ж ж ж	8F	ADC A,A	RES 1,A	
144	(A)	90	SUB B	RES 2,B	
145	(B)	91	SUB C	RES 2,C	
146	(C)	92	SUB D	RES 2,D	
147	(D)	93	SUB E	RES 2,E	
148	(E)	94	SUB H	RES 2,H	
149	(F)	95	SUB L	RES 2,L	
150	(G)	96	SUB (HL)	RES 2,(HL)	
151	(H)	97	SUB A	RES 2,A	
152	(I)	98	SBC A,B	RES 3,B	
153	(J)	99	SBC A,C	RES 3,C	
154	(K)	9A	SBC A,D	RES 3,D	
155	(L)	9B	SBC A,E	RES 3,E	
156	(M)	9C	SBC A,H	RES 3,H	
157	(N)	9D	SBC A,L	RES 3,C	
158	(O)	9E	SBC A,(HL)	RES 3,D	
159	(P)	9F	SBC A,A	RES 3,E	
160	(Q)	A0	AND B	RES 4,B	LDI
161	(R)	A1	AND C	RES 4,C	CPI
162	(S)	A2	AND D	RES 4,D	INI
163	(T)	A3	AND E	RES 4,E	OUTI
164	(U)	A4	AND H	RES 4,H	
165	RND	A5	AND L	RES 4,L	

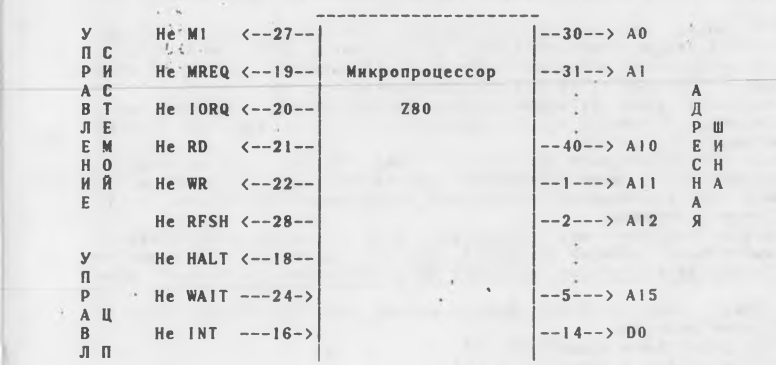
166	INKEY$	A6	AND (HL)	RES 4,(HL)	
167	PI	A7	AND A	RES 4,A	
168	FN	A8	XOR B	RES 5,B	LDD
169	POINT	A9	XOR C	RES 5,C	CPD
170	SCREEN$	AA	XOR D	RES 5,D	IND
171	ATTR	AB	XOR E	RES 5,E	OUTD
172	AT	AC	XOR H	RES 5,H	
173	TAB	AD	XOR L	RES 5,L	
174	VAL$	AE	XOR (HL)	RES 5,(HL)	
175	CODE	AF	XOR A	RES 5,A	
176	VAL	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	
189	ABS	BD	CP L	RES 7,L	
190	PEEK	BE	CP (HL)	RES 7,(HL)	
191	IN	BF	CP A	RES 7,A	
192	USR	C0	RET NZ	SET 0,B	
193	STR$	C1	POP BC	SET 0,C	
194	CHR$	C2	JP NZ,NN	SET 0,D	
195	NOT	C3	JP NN	SET 0,E	
196	BIN	C4	CALL NZ,NN	SET 0,H	
197	OR	C5	PUSH BC	SET 0,L	
198	AND	C6	ADD A,N	SET 0,(HL)	
199	<=	C7	RST 0	SET 0,A	
200	>=	C8	RET Z	SET 1,B	
201	<>	C9	RET	SET 1,C	
202	LINE	CA	JP Z,NN	SET 1,D	
203	THEN	CB		SET 1,E	
204	TO	CC	CALL Z,NN	SET 1,H	
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	
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	VARIFY	D6	SUB N	SET 2,(HL)	
215	BEEP	D7	RST 16	SET 2,A	
216	CIRCLE	D8	RET C	SET 3,B	
217	INK	D9	EXX	SET 3,C	
218	PAPER	DA	JP C,NN	SET 3,D	
219	FLASH	DB	IN A,(N)	SET 3,E	
220	BRIGHT	DC	CALL C,NN	SET 3,H	
221	INVERSE	DD	PREFIXES IN-STRUCTIONS USING IX	SET 3,L	
222	OVER	DE	SBC A,N	SET 3,(HL)	
223	OUT	DF	RST 24	SET 3,A	
225	LLIST	E1	POP HL	SET 4,C	
226	STOP	E2	JP PO,NN	SET 4,D	

227	READ	E3	EX (SP),HL	SET 4,E	
228	DATA	E4	CALL PO,NN	SET 4,H	
229	RESTORE	E5	PUSH HL	SET 4,L	
230	NEW	E6	AND N	SET 4,(HL)	
231	BORDER	E7	RST 32	SET 4,A	
232	CONTINUE	E8	RET PE	SET 5,B	
233	DIM	E9	JP (HL)	SET 5,C	
234	REM	EA	JP PE,NN	SET 5,D	
235	FOR	EB	EX DE,HL	SET 5,E	
236	GO TO	EC	CALL PE,NN	SET 5,H	
237	GO SUB	ED		SET 5,L	
238	INPUT	EE	XOR N	SET 5,(HL)	
239	LOAD	EF	RST 40	SET 5,A	
240	LIST	F0	RET P	SET 6,B	
241	LET	F1	POP AF	SET 6,C	
242	PAUSE	F2	JP P,NN	SET 6,D	
243	NEXT	F3	DI	SET 6,E	
244	POKE	F4	CALL P,NN	SET 6,H	
245	PRINT	F5	PUSH AF	SET 6,L	
246	PLOT	F6	OR N	SET 6,(HL)	
247	RUN	F7	RST 48	SET 6,A	
248	SAVE	F8	RET M	SET 7,B	
249	RUNDOMIZE	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	PREFIXES	SET 7,L	
			INSTRUCTIONS		
			USING IY		
254	RETURN	FE	CP N	SET 7,(HL)	
255	COPY	FF	RST 56	SET 7,A	

А С С Е М Б Л Е Р

Описание центрального процессора

В ЭВМ "СИНКЛЕР" выбран ЦП типа Z80A, представляющий собой более быстродействующий вариант популярного типа Z80. 4 Чипа, Z80, 6502, 6809 и 8088, стали широко распространенными в качестве ЦП для микро-эвм. Z80 - самый популярный из них.



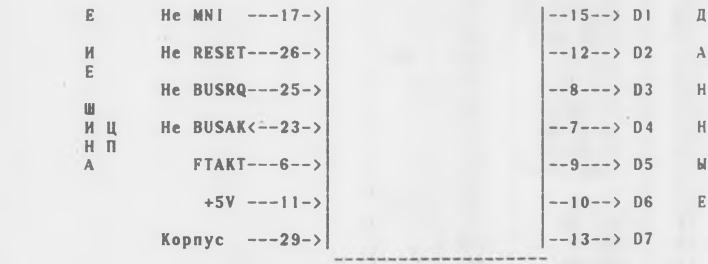


Рис.Назначение выводов микропроцессора Z80,Z80A

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.

Не M - (машинный цикл). Выход. Активный сигнал-низкий. Указывает, что в текущем цикле осуществляется выборка КОП.

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

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

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

Не 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 УВВ.

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

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

Не 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.

Не 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.

Не 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).

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

А) сброс триггера разрешения прерывания IFF.

Б) очистка счетчика команд и регистров I и R.

В) шины адресная и данных в состоянии выс.сопротивления.
Г) для всех управляющих выходных сигналов устанавливается неактивный уровень.

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

Не BUSAK - (подтверждение перевода шин в сост. высокого сопротив.) Выход. Активный уровень-низкий. Сигнал подается на запрашивающее внешнее устройство.

Таблица клавиш "SPECTRUM"

INPUT VALUE IN A FOR OFF H	D4	D3	D2	D1	D0
0FE H	V	C	X	Z	CAP SHIFT
0FD H	G	F	D	S	A
0FB H	7	R	E	W	Q
0F7 H	5	4	3	2	1
0EF H	6	7	8	9	0
0DF H	Y	U	I	O	P
0BF H	H	J	K	L	ENTER
07F H	B	N	M	SYM SHIFT	BREAK SPACE
	X: 16	8	4	2	1

INPUT VALUE IN A OFF - входное значение в А для; CAP - прописные буквы; SHIFT - смена регистра; SYM - сокращение расшифровать не удалось; ENTER - ввод; BREAK - прерывание; SPACE- пробел.

Примечание: чтобы выполнить прерывание по клавише:

1) загрузите в регистр А входное значение из соответствующего ряда.
```
LD  A  07E        : нижний ряд
```
2) примите информацию с входного порта 0FEH.
```
IN  A  (0FEH)
```
3) проверьте, что для нужной клавиши DX имеет низкое значение.
```
AND  1            : прерывание по клавише BREAK/SPACE
```
4) если ноль, то клавиша нажата.
```
JR  Z ,клавиша нажата : в нормальном состоянии значение всегда высокое
```

MEMORI ATTRIBUTE IN HEX	IN HEX	LINE	MEMORI ATTRIBUTE IN HEX	IN HEX
4000	5800	0	401F	581F
4020	5820	1	403F	583F
4040	5840	2	405F	585F
4060	5860	3	407F	587F
4080	5880	4	409F	589F
40a0	58A0	5	40BF	58BF
40c0	58C0	6	40DF	58DF
40e0	58E0	7	40FF	58FF
4800	5900	8	481F	591F
4820	5920	9	483F	593F

4840	5940	10		485F	595F
4860	5960	11		487F	597F
4880	5980	12		489F	599F
48A0	59A0	13		48BF	59BF
48C0	59C0	14		48DF	59DF
48E0	59E0	15		48FF	59FF
5000	5A00	16		501F	5A1F
5020	5A20	17		503F	5A3F
5040	5A40	18		505F	5A5F
5060	5A60	19		507F	5A7F
5080	5A80	20		509F	5A9F
50A0	5AA0	21		50BF	5ABF
50C0	5AC0	22		50DF	5ADF
50E0	5AE0	23		50FF	5AFF

MEMORY IN HEX - память в шестнадцатеричном формате;
ATTRIBUTE IN HEX - атрибут в шестнадцатеричном формате;
LINE - строка

Таблица набора литер "SPECTRUM"

HEX	HOB	0	1	2	3	4	5	6	7
LOB	BITS	000	001	010	011	100	101	110	111
0	0000	NU	INK CTRL	SPACE	0		P		
1	0001	NU	RAPER CTRJ	!	1	A	Q		
2	0010	NU	FLASH CTRI	"	2	B	R		
3	0011	NU	BRIGHT CTRI	#	3	C	S		
4	0100	NU	INVERSE CTRI	S	4	D	T		
5	0101	NU	OVER CTRI	%	5	E	U		
6	0110	PRINT	AT CTRI	$	6	F	V		
7	0111	EDIT	TAB CTRI	"	7	G	W		
8	1000	CURSOR LEFT	NU	(	8	H	X		
9	1001	CURSOR RIGH	NU	)	9	I	Y		
A	1010	CURSOR DOWN	NU	.	:	J	Z		
B	1011	CRSOR UP	NU	+	;	K			
C	1100	DELETE	NU	,	<	L			
D	1101	ENTER	NU	-	=	M			
E	1110	NUMBER	NU	.	>	N			
F	1111	NU	NU	/	?	P	-		
		<---NON PRINTABLE------>		<-------PRINTABLE------->					

NB: NU = NOT USED.

NEX - шестнадцатеричное представление; LOB - младший байт; HOB - старший байт; INK CTRI - управление чернилами; SPACE пробел; PAPER CTRI -управление бумагой; FLASH CTRI - управление режимом мигания; BRIGHT CTRI - управление яркостью; INVERSE CTRI - управление инверсией яркости; OVER CTRI- управление превышением; PRINT - печать; AT CTRI - управление; EDIT - редактирование; TAB CTRI - управление табуляцией; CURSOR LEFT - курсор влево; CURSOR RIGHT - курсор вправо; CURSOR DOWN - курсор вниз; CURSOR UP - курсор вверх; DELETE - удаление; ENTER - ввод; NUMBER - число; NON PRINTABLE - непечатные; PRINTABLE - печатаемые; NB: - замечание; NU - не используется.

Таблицы преобразования десятичных чисел в шестнадцатиричные

NEX	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	00XX	XX00
0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	0	0
1	16	17	18	19	19	20	21	22	23	24	25	27	28	29	30	31	256	4096

2	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	512	8192
3	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	768	12288
4	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	1024	16384
5	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	1280	20480
6	96	97	98	99	100	101	102	103	104	105	1061	07	108	109	110	111	1536	24576
7	112	113	114	115	116	117	118	119	120	121	1221	23	124	125	126	127	1792	28672
8	128	129	130	131	132	133	134	135	136	137	1381	39	140	141	142	143	2048	32768
9	144	145	146	147	148	149	150	151	152	153	1541	55	156	157	158	159	2304	36864
A	160	161	162	163	164	165	166	167	168	169	1701	71	172	173	174	175	2560	40960
B	176	177	178	179	180	181	182	183	184	185	1861	87	188	189	190	191	2816	45056
C	192	193	194	195	196	197	198	199	200	201	2022	03	204	205	206	207	3072	49152
D	208	209	210	211	212	213	214	215	216	217	2182	19	220	221	222	223	3328	53248
E	224	225	226	227	228	229	230	231	232	233	2342	35	236	237	238	239	3564	57344
F	240	241	242	243	244	245	246	247	248	249	2502	51	252	253	254	255	3840	61440

NEX – шестнадцатеричное.

Мы можем показать применение этой таблицы на примере. Давайте найдем шестнадцатеричный эквивалент десятичного числа 6200. Нам нужно определить 16-битовое двоичное число, т.е.,

```
    0001BBBB    BBBBBBBB
      HOB         LOB
```

HOB – старший байт; LOB – младший байт

1) из самой левой колонки таблицы под заголовком XX00 мы находим, что 6200 находится между 4096 и 8192. Так что мы выбираем меньшее значение 4096 и из значения ряда мы берем 4 самых старших бита старшего байта равные 1, т.е. 01.

```
    BBBBBBBB    BBBBBBBB
      HOB         LOB
```

2) второй шаг состоит в том, что мы определяем следующие по старшинству 4 бита старшего байта. Мы находим разность между 6200 и 4096, равную 2104. Поскольку разность все еще превышает 255, мы обращаемся ко второй слева колонке таблицы под заголовком 00XX и выясняем, что 2104 находится между 2048 и 2304. Вновь мы выбираем меньшее значение 2048 и по значению ряда получаем, что следующее по старшинству 4 бита старшего байта равны 8, т.е. 1000.

```
    00011000    BBBBBBBB
      HOB         LOB
```

3) Третий шаг состоит в определении младшего байта числа. Мы обнаруживаем, что разность между 2104 и 2048 равна 56 лежит на пересечении ряда 3 и колонки 8. Так что мы принимаем младший байт равным 38H.

```
    0001100   00111000
      HOB       LOB
```

Итак, шестнадцатеричное значение числа 6200 равно 1838H.

Шестнадцатеричная таблица сложения

+	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10
2	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11
3	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12
4	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13
5	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14
6	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15
7	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16
8	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17
9	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18
A	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19
B	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A
C	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B
D	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C
E	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D
F	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E

Таблица флагов и времени выполнения команд

MNEMONIC – мнемоническое обозначение; BYTES – байты; TIME TAKEN – время выполнения; EFFECT ON FLAGS – воздействие на флаги; REGISTER – регистр; NUMBER – число.

Обозначения флагов:

- # – означает, что флаг изменен операцией;
- 0 – означает, что флаг сбрасывается;
- 1 – означает, что флаг устанавливается;
- – – означает, что флаг не изменился;
- ? – Означает, что флаг не известен.
- C – флаг переноса. C=1, если операция привела к переносу из самого значащего бита операнда или результата.
- Z – флаг нуля. Z=1, если результат операции нулевой.
- S – флаг знака. S=1, если самый значащий бит результата равен единице, т.е. число отрицательное.
- P/V– флаг четности или переполнения. Четность (P) и переполнение (O) относятся к одному и тому же флагу. Для логических операций этот флаг задает четность результата, а для арифметических – переполнение. Если в P/V хранится четность: P/V=1, если результат операции четный, P/V=0, если результат нечетный. Если P/V содержит переполнение: P/V=1, если в результате операции получилось переполнение.
- H – флаг половинного переноса. H=1, если при операции сложения или вычитания произошел перенос или заем в четвертом бите накапливающего регистра.
- N – флаг сложения (вычитания). N=1, если предыдущей операцией было вычитание.

MNEMONIC	BYTES	TIME TAKEN	EFFECT ON FLAGS C	Z	PV	S	N	H
команды операций загрузки для одного регистра								
LD REGISTER,REGISTER	1	4	–	–	–	–	–	–
LD REGISTER,NUMBER	2	7	–	–	–	–	–	–
LD A, (ADDRESS)	3	13	–	–	–	–	–	–
LD (ADDRESS),A	3	13	–	–	–	–	–	–
LD REGISTER,(HL)	1	7	–	–	–	–	–	–
LD A,(BC)	1	7	–	–	–	–	–	–
LD A,(DC)	1	7	–	–	–	–	–	–
LD (HL),REGISTER	1	7	–	–	–	–	–	–
LD (BC),A	1	7	–	–	–	–	–	–
LD (DE),A	1	7	–	–	–	–	–	–
LD REGISTER,(IX+D)	3	19	–	–	–	–	–	–
LD REGISTER,(IY+D)	3	19	–	–	–	–	–	–
LD (IX+D),REGISTER	3	19	–	–	–	–	–	–
LD (IY+D),REGISTER	3	19	–	–	–	–	–	–
LD (HL),NUMBER	2	10	–	–	–	–	–	–
LD (IX+D),NUMBER	4	19	–	–	–	–	–	–
LD (IY+D),NUMBER	4	19	–	–	–	–	–	–
команды арифметических операций для одного регистра								
ADD A,REGISTER	1	4	#	#	#	#	0	#
ADD A,NUMBER	2	7	#	#	#	#	0	#
ADD A,(HL)	1	7	#	#	#	#	0	#

ADD A,(IX+D)	3	19	#	#	#	#	0	#
ADD a,(IY+D)	3	19	#	#	#	#	0	#
ADC A,REGISTR	1	4	#	#	#	#	0	#
ADC A,NUMBER	2	7	#	#	#	#	0	#
ADC A,(HL)	1	7	#	#	#	#	0	#
ADC A,(IX+D)	3	19	#	#	#	#	0	#
ADC A,(IY+D)	3	19	#	#	#	#	0	#
SUB A,REGISTR	1	4	#	#	#	#	1	#
SUB A,NUMBER	2	7	#	#	#	#	1	#
SUB A,(HL)	1	7	#	#	#	#	1	#
SUB A,(IX+D)	3	19	#	#	#	#	1	#
SUB A,(IY+D)	3	19	#	#	#	#	1	#
SBC A,REGISTER	1	4	#	#	#	#	1	#
SBC A,NUMBER	2	7	#	#	#	#	1	#
SBC A,(HL)	1	7	#	#	#	#	1	#
SBC A,(IX+D)	3	19	#	#	#	#	1	#
SBC A,(IY+D)	3	19	#	#	#	#	1	#
CP REGISTER	1	4	#	#	#	#	1	#
CP NUMBER	2	7	#	#	#	#	1	#
CP (HL)	1	7	#	#	#	#	1	#
CP (IX+D)	3	19	#	#	#	#	1	#
CP (IY+D)	3	19	#	#	#	#	1	#
команды логических операций								
AND REGISTER	1	4	0	#	#	#	0	1
AND NUMBER	2	7	0	#	#	#	0	1
AND (HL)	1	7	0	#	#	#	0	1
AND (IX+D)	3	19	0	#	#	#	0	1
AND (IY+D)	3	19	0	#	#	#	0	1
OR REGISTER	1	4	0	#	#	#	0	0
OR NUMBER	2	7	0	#	#	#	0	0
OR (HL)	1	7	0	#	#	#	0	0
OR (IX+D)	3	19	0	#	#	#	0	0
OR (IY+D)	3	19	0	#	#	#	0	0
XOR REGISTER	1	4	0	#	#	#	0	0
XOR NUMBER	2	7	0	#	#	#	0	0
XOR (HL)	1	7	0	#	#	#	0	0
XOR (IX+D)	3	19	0	#	#	#	0	0
XOR (IY+D)	3	19	0	#	#	#	0	0
команды операций загрузки для двух регистров								
LD REG PAIR,NUMBER	3/4	10	-	-	-	-	-	-
LD IX,NUMBER	4	14	-	-	-	-	-	-
LD IY,NUMBER	4	14	-	-	-	-	-	-
LD (ADDRESS,BS OR DE	4	20	-	-	-	-	-	-
LD (ADDRESS),HL	3	16	-	-	-	-	-	-
LD (ADDRESS),IX	4	20	-	-	-	-	-	-
LD (ADDRESS),IY	4	20	-	-	-	-	-	-
LD BC OR DE,(ADDRESS	4	20	-	-	-	-	-	-
LD HL,(ADDRESS)	3	16	-	-	-	-	-	-
LD IX,(ADDRESS)	4	20	-	-	-	-	-	-
LD IY,(ADDRESS)	4	20	-	-	-	-	-	-

команды операций со стеком								
PUSH REG PAIR	1	11	-	-	-	-	-	-
PUSH IX OR IY	2	15	-	-	-	-	-	-
POP REG PATR	1	10	-	-	-	-	-	-
POP IX OR IY	2	14	-	-	-	-	-	-
LD SP, ADDRESS	3	10	-	-	-	-	-	-
LD SP,(ADDRESS)	3	20	-	-	-	-	-	-
LD SP, HL	1	6	-	-	-	-	-	-
LD SP IX OR IY	2	10	-	-	-	-	-	-
арифметические команды для двух регистров								
ADD HL,REG PAIR	1	11	#	-	-	-	0	?
ADD HL,SP	2	11	#	-	-	-	0	?
ADC HL,REG PAIR	2	15	#	#	#	#	0	?
ADC IX,SP	2	15	#	#	#	#	0	?
ADD IX,BC OR DE	2	15	#	-	-	-	0	?
ADD IX,IX	2	15	#	-	-	-	0	?
ADD IX,SP	2	15	#	-	-	-	0	?
ADD IY,BC OR DE	2	15	#	-	-	-	0	?
ADD IY,IY	2	15	#	-	-	-	0	?
ADD IY,SP	2	15	#	-	-	-	0	?
SBC HL,REG PAIR	2	15	#	#	#	#	1	?
SBC HL,SP	2	15	#	#	#	#	1	?
Команды группы вызова и возврата								
CALL ADDRES	3	17	-	-	-	-	-	-
CALL CC,ADDRESS	3	10/17	-	-	-	-	-	-
RET	1	10	-	-	-	-	-	-
RET CC	1	5/11	-	-	-	-	-	-
команды группы сравнения и перемещения блоков								
LDI	2	16	-	-	#	-	0	0
LDD	2	16	-	-	#	-	0	0
LDIR	2	21/16	-	-	0	-	0	0
LDDR	2	21/16	-	-	0	-	0	0
CPI	2	16	-	#	#	#	1	#
CPD	2	16	-	#	#	#	1	#
CPIR	2	21/16	-	#	#	#	1	#
CPDR	2	21/16	-	#	#	#	1	#

Команды ЦП Z80 в порядке кодов операции

HEXADECIMAL	MNEMONIC	HEXADECIMAL	MNEMONIC	HEXADECIMAL	MNEMONIC
00	NOP	EAXXXX	JE PE NN	CBDB	SET 3,E
01XXXX	LD BC,NN	EB	EX DE,HL	CBDC	SET 3,H
02	LD(BC),A	ECXXXX	CALL PE,NN	CBDD	SET 3,L
03	INC BC	EEXX	XOR N	CBDE	SET 3,(HL)
04	INC B	EF	RST 28H	CBDF	SET 3,A

05	DEC B	F0	RET P	CBE0	SET 4,B
06XX	LD B,N	F1	POP AF	CBE1	SET 4,C
07	RLCA	F2XXXX	JR P,NN	CBE2	SET 4,D
08	EX AF,AF"	F3	DI	CBE3	SET 4,E
09	ADD HL,BC	F4XXXX	CALL P,NN	CBE4	SET 4,H
0A	LD A,(BC)	F5	RUSH AF	CBE5	SET 4,L
0B	DEC BC	F620XX	OR N	CBE6	SET 4,(HL)
0C	INC C	F7	RST 30H	CBE7	SET 4,A
0D	DEC C	F8	RET N	CBE8	SET 5,B
0EXX	LD C,N	F9	LD SP,HL	CBE9	SET 5,C
0F	RRCA	FAXXXX	JP N,NN	CBEA	SET 5,D
10XX	DJNZ DIS	FB	EI	CBEB	SET 5,E
11XXXX	LD DE,NN	FCXXXX	CALL M,NN	CBEC	SET 5,H
12	LD (DE),A	FE20XX	CP N	CBED	SET 5,L
13	INC DE	FF	RST 38H	CBEE	SET 5,(HL)
14	INC D	CB00	RLC 8	CBEF	SET 5,A
15	DEC D	CB01	RLC C	CBF0	SET 6,B
16XX	LD D,N	CB02	RLC D	CBF1	SET 6,C
17	RLA	CB03	RLC E	CBF2	SET 6,D
18XX	JR DIS	CB04	RLC H	CBF3	SET 6,E
19	ADD HL,DE	CB05	RLC L	CBF4	SET 6,H
1A	LD A,(DE)	CB06	RLC (HL)	CBF5	SET 6,L
1B	DEC DE	CB07	RLC A	CBF6	SET 6,(HL)
1C	INC E	CB08	RRC B	CBF7	SET 6,A
1D	DEC E	CB09	RRC C	CBF8	SET 7,B
1EXX	LD E,N	CB0A	RRC D	CBF9	SET 7,C
1F	RRA	CB0B	RRC E	CBFA	SET 7,D
20XX	JR NZ,DIS	CB0C	RRC H	CBFB	SET 7,E
21XXXX	LD HL,NN	CB0D	RRC L	CBFC	SET 7,H
22XXXX	LD(NN),HL	CB0E	RRC (HL)	CBFD	SET 7,L
23	INC HL	CB0F	RRC A	CBFE	SET 7,(HL)
24	INC H	CB10	RL B	CBFF	SET 7,A
25	DEC H	CB11	RL C	DD09	ADD IX,BC
26XX	LD H,N	CB12	RL D	DD19	ADD IX,DE
27	DAA	CB13	RL E	DD21XXXX	LD IX,NN
28XX	JR Z,DIS	CB14	RL H	DD22XXXX	LD (NN),IX
29	ADD HL,HL	CB15	RL L	DD23	INC IXC),L
2AXXXX	LD HL(NN)	CB16	RL (HL)	DD29	ADD IX,IX
2B	DEC HL	CB17	RL A	DD2AXXXX	LD IX,(NN)
2C	INC L	CB18	RR B	DD2B	DEC IX
2D	DEC L	CB19	RR C	DD34XX	INC (IX+D)
2EXX	LD L,N	CB1A	RR D	DD35XX	DEC (IX+D)
2F	CPL	CB1B	RR E	DD36XX20	LD (IX+D),N
30XX	JR NC,DIS	CB1C	RR H	DD39	ADD IX,SP
31XXXX	LD SP,NN	CB1D	RR L	DD46XX	LD B,(IX+D)
32XXXX	LD (NN),A	CB1E	RR (HL)	DD4EXX	LD C,(IX+D)
33	INC SP	CB1F	RR A	DD56XX	LD D,(IX+D)
34	INC (HL)	CB20	SLA B	DD5EXX	LD E,(IX+D)
35	DEC (HL)	CB21	SLA C	DD66XX	LD H,(IX+D)
3620XX	LD (HL),N	CB22	SLA D	DD6EXX	LD L,(IX+D)
37	SCF	CB23	SLA E	DD70XX	LD (IX+D),B
38XX	JR C,DIS	CB24	SLA H	DD71XX	LD (IX+D),C
39	ADD HL,SP	CB25	SLA L	DD72XX	LD (IX+D),D
3AXXXX	LD A,(NN)	CB26	SLA (HL)	DD73XX	LD (IX+D),E
3B	DEC SP	CB27	SLA A	DD74XX	LD (IX+D),H
3C	INC A	CB28	SRA B	DD75XX	LD (IX+D),L
3D	DEC A	CB29	SRA C	DD77XX	LD (IX+D),A
3EXXXX	LD A	CB2A	SRA D	DD7EXX	LD A,(IX+D)
3F	CCF	CB2B	SRA E	DD86XX	ADD A,(IX+D)
40	LD B,B	CB2C	SRA H	DD8EXX	ADC A,(IX+D)
41	LD B,C	CB2D	SRA L	DD96XX	SUB (IX+D)
42	LD B,D	CB2E	SRA (HL)	DD9EXX	SBC A,(IX+D)

43	LD B,E	CB2F	SRA A	DDA6XX	AND (IX+D)
44	LD B,H	CB38	SRL B	DDAEXX	XOR (IX+D)
45	LD B,L	CB39	SRL C	DDB6XX	OR (IX+D)
46	LD B,(HL)	CB3A	SRL D	DDBEXX	CP (IX+D)
47	LD B,A	CB3B	SRL E	DDE1	POP IX
48	LD C,B	CB3C	SRL H	DDE3	EX (SP),IX
49	LD C,C	CB3D	SRL L	DDE5	PUSH IX
4A	LD C,D	CB3E	SRL (HL)	DDE9	JP (IX)
4B	LD C,E	CB3F	SRL A	DDF9	LD SP,IX
4C	LD C,H	CB40	BIT 0,B	DDCBXX06	RLC (IX+D)
4D	LD C,L	CB41	BIT 0,C	DDCBXX0E	RRC (IX+D)
4E	LD C,(HL)	CB42	BIT 0,D	DDCBXX16	RL (IX+D)
4F	LD C,A	CB43	BIT 0,E	DDCBXX1E	RR (IX+D)
50	LD D,B	CB44	BIT 0,H	DDCBXX26	SLA (IX+D)
51	LD D,C	CB45	BIT 0,L	DDCBXX2E	SRA (IX+D)
52	LD D,D	CB46	BIT 0,(HL)	DDCBXX3E	SRL (IX+D)
53	LD D,E	CB47	BIT 0,A	DDCBXX46	BIT0,(IX+D)
54	LD D H	CB48	BIT 1,B	DDCBXX4E	BIT1,(IX+D)
55	LD D,L	CB49	BIT 1,C	DDCBXX56	BIT2,(IX+D)
56	LD D,(HL)	CB4A	BIT 1,D	DDCBXX5E	BIT3,(IX+D)
57	LD D,A	CB4B	BIT 1,E	DDCBXX66	BIT4,(IX+D)
58	LD E,B	CB4C	BIT 1,H	DDCBXX6E	BIT5,(IX+D)
59	LD E,C	CB4D	BIT 1,L	DDCBXX76	BIT6,(IX+D)
5A	LD E,D	CB4E	BIT 1,(HL)	DDCBXX7E	BIT7,(IX+D)
5B	LD E,E	CB4F	BIT 1,A	DDCBXX86	RES0,(IX+D)
5C	LD E,H	CB50	BIT 2,B	DDCBXX8E	RES1,(IX+D)
5D	LD E,L	CB51	BIT 2,C	DDCBXX96	RES2,(IX+D)
5E	LD E,(HL)	CB52	BIT 2,D	DDCBXX9E	RES3,(IX+D)
5F	LD E,A	CB53	BIT 2,E	DDCBXXA6	RES4,(IX+D)
60	LD H,B	CB54	BIT 2,H	DDCBXXAE	RES5,(IX+D)
61	LD H,C	CB55	BIT 2,L	DDCBXXB6	RES6,(IX+D)
62	LD H,D	CB56	BIT 2,(HL)	DDCBXXBE	RES7,(IX+D)
63	LD H,E	CB57	BIT 2,A	DDCBXXC6	SET0,(IX+D)
64	LD H,H	CB58	BIT 3,B	DDCBXXCE	SET1,(IX+D)
65	LD H,L	CB59	BIT 3,C	DDCBXXD6	SET2,(IX+D)
66	LD H,(HL)	CB5A	BIT 3,D	DDCBXXDE	SET3,(IX+D)
67	LD H,A	CB5B	BIT 3,E	DDCBXXE6	SET4,(IX+D)
68	LD L,B	CB5C	BIT 3,H	DDCBXXEE	SET5,(IX+D)
69	LD L,C	CB5D	BIT 3,L	DDCBXXF6	SET6,(IX+D)
6A	LD L,D	CB5E	BIT 3,(HL)	DDCBXXFE	SET7,(IX+D)
6B	LD L,E	CB5F	BIT 3,A	ED40	IN B,(C)
6C	LD L,H	CB60	BIT 4,B	ED41	OUT (C),B
6D	LD L,L	CB61	BIT 5,C	ED42	SBC HL,BC
6E	LD L,(HL)	CB62	BIT 4,D	ED43XXXX	LD (NN),IX
6F	LD L,A	CB63	BIT 4,E	ED44	NEG
70	LD (HL),B	CB64	BIT 4,H	ED45	RETN
71	LD (HL),C	CB65	BIT 4,L	ED46	IM 0
72	LD (HL),D	CB66	BIT 4,(HL)	ED47	LD I,A
73	LD (HL),E	CB67	BIT 4,A	ED48	INC C,(C)
74	LD (HL),H	CB68	BIT 5,B	ED49	OUT (C),C
75	LD (HL),L	CB69	BIT 5,C	ED4A	ADC HL,BC
76	HALT	CB6A	BIT 5,D	ED4BXXXX	LD BC,(NN)
77	LD (HL),A	CB6B	BIT 5,E	ED4D	RET I
78	LD A,B	CB6C	BIT 5,H	ED50	IN D,(C)
79	LD A,C	CB6D	BIT 5,L	ED51	OUT (C),D
7A	LD A,D	CB6E	BIT 5,(HL)	ED52	SBC HL,DE
7B	LD A,E	CB6F	BIT 5,A	ED53XXXX	LD (NN),DE
7C	LD A,H	CB70	BIT 6,B	ED56	IN I
7D	LD A,L	CB71	BIT 6,C	ED57	LD A,I
7E	LD A,(HL)	CB72	BIT 6,D	ED58	IN E,(C)
7F	LD A,A	CB73	BIT 6,E	ED59	OUT (C),E
80	ADD A,B	CB74	BIT 6,H	ED5A	ADC HL,DE

81	ADD A,C	CB75	BIT 6,L	ED5BXXXX	LD DE,(NN)
82	ADD A,D	CB76	BIT 6,(HL)	ED5E	IM 2
83	ADD A,E	CB77	BIT 6,A	ED60	IN H,(C)
84	ADD A,H	CB78	BIT 7,B	ED61	OUT (C),H
85	ADD A,L	CB79	BIT 7,C	ED62	SBC HL,HL
86	ADD A,(HL)	CB7A	BIT 7,D	ED67	RRD
87	ADD A,A	CB7B	BIT 7,E	ED68	IN L,(C)
88	ADC A,B	CB7C	BIT 7,H	ED69	OUT (C),L
89	ADC A,C	CB7D	BIT 7,L	ED6A	ADC HL,HL
8A	ADC A,D	CB7E	BIT 7,(HL)	ED6F	RLD
8B	ADC A,E	CB7F	BIT 7,A	ED72	SBC HL,SP
8C	ADC A,H	CB80	RES 0,B	ED73XXXX	LD (NN),SP
8D	ADC A,L	CB81	RES 0,C	ED78	IN A,(C)
8E	ADC A,(HL)	CB82	RES 0,D	ED79	OUT (C),A
8F	ADC A,A	CB83	RES 0,E	ED7A	ADC HL,SP
90	SUB B	CB84	RES 0,H	ED7BXXXX	LD SP,(NN)
91	SUB C	CB85	RES 0,L	EDA0	LDI
92	SUB D	CB86	RES 0,(HL)	EDA1	CPI
93	SUB E	CB87	RES 0,A	EDA2	INI
94	SUB H	CB88	RES 1,B	EDA3	OUTI
95	SUB L	CB89	RES 1,C	EDA8	LDD
96	SUB (HL)	CB8A	RES 1,D	EDA9	CPD
97	SUB A	CB8B	RES 1,E	EDAA	IND
98	SBC A,B	CB8C	RES 1,H	EDAB	OUTD
99	SBC A,C	CB8D	RES 1,L	ED80	LDIR
9A	SBC A,D	CB8E	RES 1,(HL)	ED81	CPIR
9B	SBC A,E	CB8F	RES 1,A	ED82	INIR
9C	SBC A,H	CB90	RES 2,B	ED83	OTIR
9D	SBC A,L	CB91	RES 2,C	ED88	LDDR
9E	SBC A,(HL	CB92	RES 2,D	ED89	CPDR
9F	SBC A,A	CB93	RES 2,E	ED8A	INDR
A0	AND B	CB94	RES 2,H	ED8B	OTDR
A1	AND C	CB95	RES 2,L	ED09	ADD IY,BC
A2	AND D	CB96	RES 2,(HL)	ED19	ADD IY,DC
A3	AND E	CB97	RES 2,A	ED21XXXX	LD IY,NN
A4	AND H	CB98	RES 3,B	FD22XXXX	LD (NN),IY
A5	AND L	CB99	RES 3,C	FD23	INC IY
A6	AND (HL)	CB9A	RES 3,D	FD29	ADD IY,IY
A7	AND A	CB9B	RES 3,E	FD2AXXXX	LD IY,(NN)
A8	XOR B	CB9C	RES 3,H	FD2B	DEC IY
A9	XOR C	CB9D	RES 3,L	FD34XX	INC (IY+D)
AA	XOR D	CB9E	RES 3,(HL)	FD35XX	DEC (IY+D)
AB	XOR E	CB9F	RES 3,A	D36XX20	LD (IY+D),N
AC	XOR H	CBA0	RES 4,B	FD39	ADD IY,CP
AD	XOR L	CBA1	RES 4,C	FD46XX	LD B,(IY+D)
AE	XOR (HL	CBA2	RES 4,D	FD4EXX	LD C,(IY+D)
AF	XOR A	CBA3	RES E,E	FD56XX	LD D,(IY+D)
B0	OR B	CBA4	RES E,H	FD5EXX	LD E,(IY+D)
B1	OR C	CBA5	RES 4,L	FD66XX	LD H,(IY+D)
B2	OR D	CBA6	RES 4,(HL)	FD6EXX	LD L,(IY+D)
B3	OR E	CBA7	RES 4,A	FD70XX	LD (IY+D),B
B4	OR H	CBA8	RES 5,B	FD71XX	LD (IY+D),C
B5	OR L	CBA9	RES 5,C	FD72XX	LD (IY+D),D
B6	OR (HL)	CBAA	RES 5,D	FD73XX	LD (IY+D),E
B7	OR A	CBAB	RES 5,E	FD74XX	LD (IY+D),H
B8	CP B	CBAC	RES 5,H	FD75XX	LD (IY+D),L
B9	CP C	CBAD	RES 5,L	FD77XX	LD (IY+D),A
BA	CP D	CBAE	RES 5,(HL)	FD7EXX	LD A,(IY+D)
BB	CP E	CBAF	RES 5,A	FD86XX	ADD A,(IY+D
BC	CP H	CBB0	RES 6,B	FD8EXX	ADC A,(IY+D
BD	CP L	CBB1	RES 6,C	FD96XX	SUB (IY+D)
BE	CP (HL)	CBB2	RES 6,D	FD9EXX	SBC A,(IY+D

BF	CP A	CBB3	RES 6,E	FDA6XX	AND (IY+D)
C0	RET NZ	CBB4	RES 6,H	FDAEXX	XOR (IY+D)
C1	POP BC	CBB5	RES 6,L	FDB6XX	OR (IY+D)
C2XXXX	JP NZ,NM	CBB6	RES 6,(HL)	FDBEXX	Cp (IY+D)
C3XXXX	JP NM	CBB7	RES 6,A	FDE1	POP IY
C4XXXX	CALL NZ NM	CBB8	RES 7,B	FDE3	EX (SP),IY
C5	PUSH BCHL	CBB9	RES 7,C	FDE5	RUSH IY
C6XX	ADD A,N	CBBA	RES 7,D	FDE9	JP (IY)
C7	RST 0	CBBB	RES 7,E	FDF9	LD SP,IY
C8	RET Z	CBBC	RES 7,H	FDCBXX06	RLC (IY+D)
C9	RET	CBBD	RES 7,L	FDCBXX0E	RRC (IY+D)
CAXXXX	JP Z,NM	CBBE	RES 7,(HL)	FDCBXX16	RL (IY+D)
CCXXXX	CALL Z,NN	CBBF	RES 7,A	FDCBXX1E	RR (IY+D)
CDXXXX	CALL NN	CBC0	SET 0,B	FDCBXX26	SLA (IY+D)
CEXX	ADC A,N	CBC1	SET 0,C	FDCBXX2E	SRA (IY+D)
CF	RST 8	CBC2	SET 0,D	FDCBXX3E	SRL (IY+D)
D0	RET NC	CBC3	SET 0,E	FDCBXX46	BIT 0,(IY+D
D1	POP DE	CBC4	SET 0,H	FDCBXX4E	BIT 1,(IY+D
D2XXXX	JP NC,NN	CBC5	SET 0,L	FDCBXX56	BIT 2,(IY+D
D3XX	OUT (N),A	CBC6	SET 0,(HL)	FDCBXX5E	BIT 3,(IY+D
D4XXXX	CALL NC,NN	CBC7	SET 0,A	FDCBXX66	BIT 4,(IY+D
D5	PUSH DE	CBC8	SET 1,B	FDCBXX6E	BIT 5,(IY+D
D6XX	SUB N	CBC9	SET 1,C	FDCBXX76	BIT 6,(IY+D
D7	RST 10H	CBCA	SET 1,D	FDCBXX7E	BIT 7,(IY+D
D8	RET C	CBCB	SET 1,E	FDCBXX86	RES 0,(IY+D
D9	EXX	CBCC	SET 1,H	FDCBXX8E	RES 1,(IY+D
DAXXXX	JP C,NN	CBCD	SET 1,L	FDCBXX96	RES 2,(IY+D
DBXX	IN A,(N)	CBCE	SET 1,(HL)	FDCBXX9E	RES 3,(IY+D
DCXXXX	CALL C,NN	CBCF	SET 1,A	FDCBXXA6	RES 4,(IY+D
DEXX	SBC A,N	CBD0	SET 2,B	FDCBXXAE	RES 5,(IY+D
DF	RST 18H	CBD1	SET 2,C	FDCBXXB6	RES 6,(IY+D
E0	RET pO	CBD2	SET 2,D	FDCBXXBE	RES 7,(IY+D
E1	POP HL	CBD3	SET 2,E	FDCBXXC6	SET 0,(IY+D
E2XXXX	JP PO,NN	CBD4	SET 2,H	FDCBXXCE	SET 1,(IY+D
E3	EX (SP),HL	CBD5	SET 2,L	FDCBXXD6	SET 2,(IY+D
E4XXXX	CALL PO,NN	CBD6	SET 2,(HL)	FDCBXXDE	SET 3,(IY+D
E5	PUSH HL	CBD7	SET 2,A	FDCBXXE6	SET 4,(IY+D
E6XX	AND N	CBD8	SET 3,B	FDCBXXEE	SET 5,(IY+D
E7	RST 20H	CBD9	SET 3,C	FDCBXXF6	SET 6,(IY+D
E8	RET PE	CBDA	SET 3,D	FDCBXXFE	SET 7,(IY+D
E9	JP (HL)				

Команды ЦП Z80 в порядке возрастания мнемонических обозначений

MNEMONIC	HEXADECIMAL	MNEMONIC	HEXADECIMAL	MNEMONIC	HEXADECIMAL
ADC A,(HL)	8E	LD(ADDR),HL	ED63XXXX	RES6,(IX+D)	DDCBXXB6
ADC A,(IX+D)	DD8EXX	LD(ADDR),HL	22XXXX	RES6,(IY+D)	FDCBXXB6
ADC A,(IY+D)	FD8EXX	LD(ADDR),IX	DD22XXXX	RES 6,A	CBB7
ADC A,A	8F	LD(ADDR),IY	FD22XXXX	RES 6,B	CBB0
ADC A,B	88	LD(ADDR),SP	ED73XXXX	RES 6,C	CBB1
ADC A,C	89	LD (BC),A	02	RES 6,D	CBB2
ADC A,D	8A	LD (DE),A	12	RES 6,E	CBB3
ADC A,N	CEXX	LD (HL),A	77	RES 6,H	CBB4
ADC A,T	8B	LD (HL),B	70	RES 6,L	CBB5
ADC A,H	8C	LD (HL),C	71	RES 7,(HL)	CBBE
ADC A,L	8D	LD (HL),D	72	RES7,(IX+D)	DDCBXXBE
ADC HL,BC	ED4A	LD (HL),N	36XX	RES7,(IY+D)	FDCBXXBE
ADC HL,DE	ED5A	LD (HL),E	73	RES 7,A	CBBF
ADC HL,HL	ED6A	LD (HL),H	74	RES 7,B	CBB8

ADD HL,SP	ED7A	LD (HL),L	75	RES 7,C	CBB9
ADD A,(HL)	86	LD (IX+D),A	DD77XX	RES 7,D	CBBA
ADD A,(IX+D)	DD86XX	LD (IX+D),B	DD70XX	RES 7,E	CBBB
ADD A,(IY+D)	FD86XX	LD (IX+D),C	DD71XX	RES 7,H	CBBC
ADD A,A	87	LD (IX+D),D	DD72XX	RES 7,L	CBBD
ADD A,B	80	LD (IX+D),N	DD36XXXX	RET	C9
ADD A,C	81	LD (IX+D),E	DD73XX	RET C	D8
ADD A,D	82	LD (IX+D),H	DD74XX	RET M	F8
ADD A,N	C6XX	LD (IX+D),L	DD75XX	RET NC	D0
ADD A,E	83	LD (IY+D),A	FD77XX	RET NZ	C0
ADD A,H	84	LD (IY+D),B	FD70XX	RET P	F0
ADD A,L	85	LD (IY+D),C	FD71XX	RET PE	E8
ADD HL,BC	09	LD (IY+D),D	FD72XX	RET PO	E0
ADD HL,DE	19	LD (IY+D),N	FD36XXXX	RET Z	C8
ADD HL,HL	29	LD (IY+D),E	FD73XX	RETI	ED4D
ADD HL,SP	39	LD (IY+D),H	FD74XX	RETN	ED45
ADD IX,BC	DD09	LD (IY+D),L	FD75XX	RL (HL)	CB16
ADD IX,DE	DD19	LD A,(ADDR)	3AXXXX	RL (IX+D)	DDCBXX16
ADD IX,IX	DD29	LD A,(BC)	0A	RL (IY+D)	FDCBXX16
ADD IX,SP	DD39	LD A,(DE)	1A	RL A	CB17
ADD IY,BC	FD09	LD A,(HL)	7E	RL B	CB10
ADD IY,DE	FD19	LD A,(IX+D)	DD7EXX	RL C	CB11
ADD IY,IY	FD29	LD A,(IY+D)	FD7EXX	RL D	CB12
ADD IY,SP	FD39	LD A,A	7F	RL E	CB13
AND (HL)	A6	LD A,B	78	RL H	CB14
AND (IX+D)	DDA6XX	LD A,C	79	RL L	CB15
AND (IY+D)	FDA6XX	LD A,D	7A	RLA	17
AND A	A7	LD A,N	3EXX	RLC (HL)	CB06
AND B	A0	LD A,E	7B	RLC (IX+D)	DDCBXX06
AND C	A1	LD A,H	7C	RLC (IY+D)	FDCBXX06
AND D	A2	LD A,I	ED57	RLC A	CB07
AND N	E6XX	LD A,L	7D	RLC B	CB00
AND E	A3	LD A,R	ED5F	RLC C	CB01
AND H	A4	LD B,(HL)	46	RLC D	CB02
AND L	A5	LD B,(IX+D)	DD46XX	RLC E	CB03
BIT 0,(HL)	CB46	LD B,(IY+D)	FD46XX	RLC H	CB04
BIT0,(IX+D)	DDCBXX46	LD B,A	47	RLC L	CB05
BIT0,(IY+D)	FDCBXX46	LD B,B	40	RLCA	07
BIT 0,A	CB	LD B,C	41	RLD	ED6F
BIT 0,B	CB40	LD B,D	42	RR (HL)	CB1E
BIT 0,C	CB41	LD B,N	06XX	RR (IX+D)	DDCBXX1E
BIT 0,D	CB42	LD B,E	43	RR (IY+D)	FDCBXX1E
BIT 0,E	CB43	LD B,H	44	RR A	CB1F
BIT 0,H	CB44	LD B,L	45	RR B	CB18
BIT 0,L	CB45	LD BC,(ADDR)	ED4BXXXX	RR C	CB19
BIT 1,(HL)	CB4E	LD BC,NN	01XXXX	RR D	CB1A
BIT 1,(IX+D)	DDCBXX4E	LD C,(HL)	4E	RR E	CB1B
BIT 1,(IY+D)	FDCBXX4E	LD C,(IX+D)	DD4EXX	RR H	CB1C
BIT 1,A	CB4F	LD C,(IY+D)	FD4EXX	RR L	CB1D
BIT 1,B	CB48	LD C,A	4F	RRA	1F
BIT 1,C	CB49	LD C,B	48	RRC (HL)	CB0e
BIT 1,D	CB4A	LD C,C	49	RRC (IX+D)	DDCBXX0E
BIT 1,E	CB4B	LD C,D	4A	RRC (IY+D)	FDCBXX0E
BIT 1,H	CB4C	LD C,N	0EXX	RRC A	CB0F
BIT 1,L	CB4D	LD C,E	4B	RRC B	CB08
BIT 2,(HL)	CB56	LD C,H	4C	RRC C	CB09
BIT 2,(IX+D)	DDCBXX56	LD C,L	4D	RRC D	CB0A
BIT 2,(IY+D)	FDCBXX56	LD D,(HL)	56	RRC E	CB0B
BIT 2,A	CB57	LD D,(IX+D)	DD56XX	RRC H	CB0C
BIT 2,B	CB50	LD D,(IY+D)	FD56XX	RRC L	CB0D
BIT 2,C	CB51	LD D,A	57	RRCA	0F
BIT 2,D	CB52	LD D,B	50	RRD	ED67

BIT 2,E	CB53	LD C,C	51	RST 00	C7
BIT 2,H	CB54	LD D,D	52	RST 08	CF
BIT 2,L	CB55	LD D,N	16XX	RST 10	D7
BIT 3,(HL)	CB5e	LD D,E	53	RST 18	DF
BIT 3,(IX+D)	DDCBXX5E	LD D,H	54	RST 20	E7
BIT 3,(IY+D)	FDCBXX5E	LD D,L	55	RST 28	EF
BIT 3,A	CB5F	LD DE,(ADDR)	ED5BXXXX	RST 30	F7
BIT 3,B	CB58	LD DE,NN	11XXXX	RST 38	FF
BIT 3,C	CB59	LD E,(HL)	5E	SBC A,(HL)	9E
BIT 3,D	CB5A	LD E,(IX+D)	DD5EXX	SBC A,(IX+D)	DD9EXX
BIT 3,E	CB5B	LD E,(IY+D)	FD5EXX	SBC A,(IY+D)	FD9EXX
BIT 3,H	CB5C	LD E,A	5F	SBC A,A	9FXX
BIT 3,L	CB5D	LD E,B	58	SBC A,B	98
BIT 4,(HL)	CB66	LD E,C	59	SBC A,C	99
BIT 4,(IX+D)	DDCBXX66	LD E,D	5A	SBC A,D	9A
BIT 4,(IY+D)	FDCBXX66	LD E,N	1EXX	SBC A,N	DEXX
BIT 4,A	CB67	LD E,E	5B	SBC A,E	9B
BIT 4,B	CB60	LD E,H	5C	SBC A,H	9C
BIT 4,C	CB61	LD E,L	5D	SBC A,L	9D
BIT 4,D	CB62	LD H,(HL)	66	SBC HL,BC	ED42
BIT 4,E	CB63	LD H,(IX+D)	DD66XX	SBC HL,DE	ED52
BIT 4,H	CB64	LD H,(IY+D)	FD66XX	SBC HL,HL	ED62
BIT 4,L	CB65	LD H,A	67	SBC HL,SP	ED72
BIT 5,(HL)	CB6E	LD H,B	60	SCF	37
BIT 5,(IX+D)	DDCBXX6E	LD H,C	61	SET 0,(HL)	CBC6
BIT 5,(IY+D)	FDCBXX6E	LD H,D	62	SET0,(IX+D)	DDCBXXC6
BIT 5,A	CB6F	LD H,N	26XX	SET0,(IY+D)	FDCBXXC6
BIT 5,B	CB68	LD H,E	63	SET 0,A	CBC7
BIT 5,C	CB69	LD H,H	64	SET 0,B	CBC0
BIT 5,D	CB6A	LD H,L	65	SET 0,C	CBC1
BIT 5,E	CB6B	LDHL,(ADDR)	ED68XXXX	SET 0,D	CBC2
BIT 5,H	CB6C	LDHL,(ADDR)	2AXXXX	SET 0,E	CBC3
BIT 5,L	CB6D	LD HL,NN	21XXXX	SET 0,H	CBC4
BIT 6,(HL)	CB76	LD I,A	ED47	SET 0,L	CBC5
BIT6,(IX+D)	DDCBXX7	LDIX,(ADDR)	DD2AXXXX	SET 1,(HL)	CBCE
BIT6,(IY+D)	FDCBXX7	LDIX,NN	DD21XXXX	SET1,(IX+D)	DDCBXXCE
BIT 6,A	CB77	LDIY,(ADDR)	FD2AXXXX	SET1,(IY+D)	FDCBXXCE
BIT 6,B	CB70	LD IY,NN	FD21XXXX	SET 1,A	CBCF
BIT 6,C	CB71	LD L,A	6F	SET 1,B	CBC8
BIT 6,D	CB72	LD L,B	68	SET 1,C	CBC9
BIT 6,E	CB73	LD L,C	69	SET 1,D	CBCA
BIT 6,H	CB74	LD L,D	6A	SET 1,E	CBCB
BIT 6,L	CB75	LD L,N	2EXX	SET 1,H	CBCC
BIT 7,(HL)	CB7E	LD L,E	6B	SET 1,L	CBCD
BIT7,(IX+D)	DDCBXX7	LD L,(HL)	6E	SET 2,(HL)	CBD6
BIT7,(IY+D)	FDCBXX7	LDL,(IX+D)	DD6EXX	SET2,(IX+D)	DDCBXXD6
BIT 7,A	CB7F	LDL,(IY+D)	FD6EXX	SET2,(IY+D)	FDCBXXD6
BIT 7,B	CB78	LD L,H	6C	SET 2,A	CBD7
BIT 7,C	CB79	LD L,L	6D	SET 2,B	CBD0
BIT 7,D	CB7A	LD R,A	ED4F	SET 2,C	CBD1
BIT 7,E	CB7B	LDSP,(ADDR)	ED7BXXXX	SET 2,D	CBD2
BIT 7,H	CB7C	LD SP,NN	31XXXX	SET 2,E	CBD3
BIT 7,L	CB7D	LD SP,HL	F9	SET 2,H	CBD4
CALL ADDR	CDXXXX	LD SP,IX	DDF9	SET 2,L	CBD5
CALL C,ADDR	DCXXXX	LD SP IY	FDF9	SET 3,(HL)	CBDE
CALL M,ADDR	FCXXXX	LDD	EDA8	SET3,(IX+D)	DDCBXXCE
CALL NC,ADDR	D4XXXX	LDDR	ED88	SET3,(IY+D)	FDCBXXCE
CALL NZ,ADDR	C4XXXX	LDI	EDA0	SET 3,A	CBDF
CALL P,ADDR	F4XXXX	LDIR	EDB0	SET 3,B	CBD8
CALL PE,ADDR	ECXXXX	NEG	ED44	SET 3,C	CBD9
CALL PO,ADDR	E4XXXX	NOP	00	SET 3,D	CBDA
CALL Z,ADDR	CCXXXX	OR (HL)	B6	SET 3,E	CBDB

CPF	3F	OR (IX+D)	DDB6XX	SET 3,H	CBDC
CP (HL)	BE	OR (IY+D)	FDB6XX	SET 3,L	CBDD
CP (IX+D)	DDBEXX	OR A	B7	SET 4,(HL)	CBE6
CP (IY+D)	FDBEXX	OR B	B0	SET4,(IX+D)	DDCBXXE6
CP A	BF	OR C	B1	SET4,(IY+D)	FDCBXXE6
CP B	B8	OR D	B2	SET 4,A	CBE7
CP C	B9	OR N	F6XX	SET 4,B	CBE0
CP D	BA	OR E	B3	SET 4,C	CBE1
CP N	FEXX	OR H	B4	SET 4,D	CBE2
CP E	BB	OR L	B5	SET 4,E	CBE3
CP H	BC	OTDR	EDBB	SET 4,H	CBE4
CP L	BD	OTIR	EDB3	SET 4,L	CBE5
CPD	EDA9	OUT (C),A	ED79	SET 5,(HL)	CBEE
CPDR	EDB9	OUT (C),B	ED41	SET5,(IX+D)	DDCBXXEE
CPI	EDA1	OUT (C),C	ED49	SET5,(IY+D)	FDCBXXEE
CPIR	EDB1	OUT (C),D	ED51	SET 5,A	CBEF
CPL	2F	OUT (C),E	ED59	SET 5,B	CBE8
DAA	27	OUT (C),H	ED61	SET 5,C	CBE9
DEC (HL)	35	OUT (C),L	ED69	SET 5,D	CBEA
DEC (IX+D)	DD35XX	OUT PORT,A	D3PORT	SET 5,E	CBEB
DEC (IY+D)	FD35XX	OUTD	EDAB	SET 5,H	CBEC
DEC A	3D	OUTI	EDA3	SET 5,L	CBED
DEC B	05	POP AF	F1	SET 6,(HL)	CBF6
DEC BC	0B	POP BC	C1	SET6,(IX+D)	DDCBXXF6
DEC C	0D	POP DE	D1	SET6,(IY+D)	FDCBXXF6
DEC D	15	POP HL	E1	SET 6,A	CBF7
DEC DE	1B	POP IX	DDE1	SET 6,B	CBF0
DEC E	1D	POP IY	FDE1	SET 6,C	CBF1
DEC H	25	PUSN AF	F5	SET 6,D	CBF2
DEC HL	2B	PUSH BC	C5	SET 6,E	CBF3
DEC IX	DD2B	PUSH DE	D5	SET 6,H	CBF4
DEC IY	FD2B	PUSH HL	E5	SET 6,L	CBF5
DEC L	2D	PUSH IX	DDE5	SET 7,(HL)	CBFE
DEC SP	3B	PUSH IY	FDE5	SET7,(IX+D)	DDCBXXFE
DI	F3	RES 0,(HL)	CB86	SET7,(IY+D)	FDCBXXFE
DJNZ,D	10XX	RES0,(IX+D)	DDCBXX86	SET 7,A	CBFF
EI	F8	RES0,(IY+D)	FDCBXX86	SET 7,B	CBF8
EX (SP),HL	E3	RES 0,A	CB87	SET 7,C	CBF9
EX (SP),IX	DDE3	RES 0,B	CB80	SET 7,D	CBFA
EX (SP),IY	FDE3	RES 0,C	CB81	SET 7,E	CBFB
EX AF,AF"	08	RES 0,D	CB82	SET 7,H	CBFC
EX DE,HL	EB	RES 0,E	CB83	SET 7,L	CBFD
EXX	D9	RES 0,H	CB84	SLA (HL)	CB26
HALT	76	RES 0,L	CB85	SLA (IX+D)	DDCBXX26
IM 0	ED46	RES 1,(HL)	CB8E	SLA (IY+D)	FDCBXX26
IM 1	ED56	RES1,(IX+D)	DDCBXX8E	SLA A	CB27
IM 2	ED5E	RES1,(IY+D)	FDCBXX8E	SLA B	CB20
IN A,(C)	ED78	RES 1,A	CB8F	SLA C	CB21
IN A,PORT	D8XX	RES 1,B	CB88	SLA D	CB22
IN B,(C)	ED40	RES 1,C	CB89	SLA E	CB23
IN C,(C)	ED48	RES 1,D	CB8A	SLA H	CB24
IN D,(C)	ED50	RES 1,E	CB8B	SLA L	CB25
IN E,(C)	ED58	RES 1,H	CB8C	SRA (HL)	CB2E
IN H,(C)	ED60	RES 1,L	CB8D	SRA (IX+D)	DDCBXX2E
IN L,(C)	ED68	RES 2,(HL)	CB96	SRA (IY+D)	FDCBXX2E
INC (HL)	34	RES2,(IX+D)	DDCBXX96	SRA A	CB2F
INC (IX+D)	DD34XX	RES2,(IY+D)	FDCBXX96	SRA B	CB28
INC (IY+D)	FD34XX	RES 2,A	CB97	SRA C	CB29
INC A	3C	RES 2,B	CB90	SRA D	CB2A
INC B	04	RES 2,C	CB91	SRA E	CB2B
INC BC	03	RES 2,D	CB92	SRA H	CB2C
INC C	0C	RES 2,E	CB93	SRA L	CB2D

INC D	14	RES 2,H	CB94	SRL (HL)	CB3E
INC DE	13	RES 2,L	CB95	SRL (IX+D)	DDCBXX3E
INC E	1C	RES 3,(HL)	CB9E	SRL (IY+D)	FBCBXX3E
INC H	24	RES3,(IX+D)	DDCBXX9E	SRL A	CB3F
INC HL	23	RES3,(IY+D)	FDCBXX9E	SRL B	CB39
INC IX	DD23	RES 3,A	CB9F	SRL C	CB39
INC IY	FD23	RES 3,B	CB98	SRL D	CB3A
INC L	2C	RES 3,C	CB99	SRL E	CB3B
INC SP	33	RES 3,D	CB9A	SRL H	CB3C
IND	EDAA	RES 3,E	CB9B	SRL L	CB3D
INCR	EDBA	RES 3,H	CB9C	SUB (HL)	96
INI	EDA2	RES 3,L	CB9D	SUB (IX+D)	DD96XX
INIR	EDB2	RES 4,(HL)	CBA6	SUB (IY+D)	FD96XX
JP (HL)	E9	RES4,(IX+D)	DDCBXXA6	SUB A	97
JP (IX)	DDE9	RES4,(IY+D)	FDCBXXA6	SUB B	90
JP (IY)	FDE9	RES 4,A	CBA7	SUB C	91
JP ADDR	C3XXXX	RES 4,B	CBA0	SUB D	92
JP C,ADDR	DAXXXX	RES 4,C	CBA1	SUB N	D6XX
JP M,ADDR	FAXXXX	RES 4,D	CBA2	SUB E	93
JP NC,ADDR	D2XXXX	RES 4,E	CBA3	SUB H	94
JP NZ,ADDR	C2XXXX	RES 4,H	CBA4	SUB L	95
JP P,ADDR	F2XXXX	RES 4,L	CBA5	XOR (HL)	AE
JP PE,ADDR	EAXXXX	RES 5,(HL)	CBAE	XOR (IX+D)	DDAEXX
JP PO,ADDR	E2XXXX	RES5,(IY+D)	DDCBXXAE	XOR (IY+D)	FDAEXX
JP Z,ADDR	CAXXXX	RES5,(IY+D)	FDCBXXAE	XOR A	AF
JR C,D	38XX	RES 5,A	CBAF	XOR B	A8
JR D	18XX	RES 5,B	CBA8	XOR C	A9
JR NC,D	30XX	RES 5,C	CBA9	XOR D	AA
JR NZ,D	20XX	RES 5,D	CBAA	XOR N	EEXX
JR Z,D	28XX	RES 5,E	CBAB	XOR E	AB
LD (ADDR),A	32XXXX	RES 5,H	CBAC	XSOR H	AC
LD(ADDR),BC	ED43XXXX	RES 5,L	CBAD	XOR L	AD
LD(ADDR),DE	ED53XXXX	RES 6,(HL)	CBB6		

MNEMONIC - мнемоника. HEAXDECIMAL - шестнадцатиричный код. D - приращение

АЛФАВИТНЫЙ УКАЗАТЕЛЬ МЕТОК

1-CARRY	2AE8	GET-HL*DE	2AF4	OUT-LINE6	18B4	S-CONT-3	2713
1-RESTORE	2AEB	GET-MEM-0	340F	OUT-NUM-1	1A1B	S-DECIMAL	268D
ABS	346A	GET-PARAM	1B55	OUT-NUM-2	1A28	S-FN	25F5
ACS	3843	GO-NC-MLT	30A5	OUT-NUM-3	1A30	S-FN-SBRN	27BD
ADD-BACK	3004	GO-SUB	1EED	OUT-NUM-4	1A42	S-IK$-STK	2660
ADD-CH-1	0F8B	GO-TO	1E67	OUT-SP-1	192B	S-INK$-EN	2665
ADD-CHAR	0F81	GO-TO-2	1E73	OUT-SP-2	1925	S-INKEY$	2634
ADD-REP-6	309F	GRE-8	373D	OUT-SP-NO	192A	S-LETTER	26C9
ADD-RSTOR	3049	GREATER-0	34F9	P-BEEP	1AE3	S-LOOP	2734
ADD-STORE	3040	HL-AGAIN	30BC	P-BORDER	1AF5	S-LOOP-1	24FF
ADDENS-0	2FF9	HL-END	30BE	P-BRIGHT	1AEE	S-LOOPEND	2770
ADDITION	3014	HL-LOOP	30B1	P-CAT	1B14	S-NEGATE	26DF
ADDN-OFLW	303C	HL=HL*DE	30A9	P-CIRCLE	1AE7	S-NEXT	2790
ALL-ADDED	300D	IF	1CF0	P-CLEAR	1AB8	S-NO-TO-$	2707
ALPHA	2C8D	IF-1	1D00	P-CLOSE	1B02	S-NOT-AND	2788
ALPHANUM	2C88	IN	34A5	P-CLS	1ABE	S-NUMERIC	26C3
ARC-END	245F	IN-ASSIGN	21B9	P-CONT	1AB8	S-OPERTR	2723
ARC-LOOP	2425	IN-CHAN-K	21D6	P-COPY	1AD6	S-PI	2627
ARC-START	2437	IN-ITEM-1	20C1	P-DATA	1ACC	S-PI-END	2630
ASN	3833	IN-ITEM-2	20D8	P-DEF-FN	1AF9	S-POINT	267B
ATN	37E2	IN-ITEM-3	20ED	P-DIM	1AA2	S-PUSH-PO	270D
AUTO-L-1	17C5	IN-NEXT-1	21AF	P-DRAW	1AD2	S-Q-AGAIN	25BE

AUTO-L-2	17E1	IN-NEXT-2	21B2	P-ERASE	1B10	S-Q-COPY	25CB
AUTO-L-3	17E4	IN-PK-STK	3480	P-FLASH	1AED	S-Q-PRMS	25D9
AUTO-L-4	17ED	IN-PK-STK	3480	P-FOR	1A90	S-Q-PRMS	25D9
AUTO-LIST	1795	IN-PR-1	211A	P-FORMAT	1B06	S-QUOTE	2583
BC-SPACES	0030	IN-PR-2	211C	P-GO-SUB	1A86	S-QUOTE-S	250F
BE-AGAIN	03F2	IN-PR-3	2129	P-GO-TO	1A7D	S-RND	25F8
BE-END	03F6	IN-PROMPT	20FA	P-IF	1A81	S-RND-END	2625
BE-H&L-LP	03D6	IN-STOP	21D0	P-INC	1AEB	S-RPORT-C	252D
BE-I-OK	0425	IN-VAR-1	213A	P-INPUT	1A9F	S-RPOT-C	2761
BE-IX+0	03D4	IN-VAR-2	2148	P-INT-STO	2D8C	S-RUNTEST	2764
BE-IX+1	03D3	IN-VAR-3	215E	P-INVERSE	1AEF	S-SC-MTCH	255A
BE-IX+2	03D2	IN-VAR-4	2161	P-LET	1A7A	S-SCR-NXT	2573
BE-IX+3	03D1	IN-VARS-5	2174	P-LIST	1AAE	S-SCR-STO	257D
BEEP	03F8	IN-VARS-6	2198	P-LLIST	1ADC	S-SCREEN$	2668
BEEPER	03B5	INC	1988	P-LOAD	1AE0	S-SCRN$-0	2535
BIN-DIGIT	2CA2	INDEXER	16DC	P-LPRINT	1AD9	S-SCRN-LP	254F
BIN-END	2CB3	INDEXER-1	16DB	P-MERGE	1AE2	S-SD-SKIP	26B6
BITS-ZERO	3283	INPUT	2089	P-MOVE	1B05	S-SINTEST	275B
BORDER	2294	INPUT	2096	P-NEW	1AA8	S-STK-DEC	26B5
BORDER-1	22A6	INPUT-2	20AD	P-NEXT	1A98	S-STK-LST	274C
BOTH-NULL	3572	INPUT-AD	15E6	P-OPEN	1AFC	S-STRING	25DB
BREAK-KEY	1F54	INT	35AF	P-OUT	1AF1	S-TIGTHER	2773
BYTE-COMP	3564	INT-CASE	3483	P-OVER	1AF0	S-U-PLUS	25AF
BYTE-ZERO	327E	INT-EXP1	2ACC	P-PAPER	1AEC	SOSC-ROWS	255D
C-ARC-GE1	235A	INT-EXP2	2ACD	P-PAUSE	1AC5	SA-1-SEC	0991
C-ENT	37B7	INT-FETCH	2D7F	P-PLOT	1AC1	SA-8-BITS	0525
C-R-GRE-1	233B	INT-STORE	2D8E	P-POKE	1AB1	SA-ALL	075A
CA=10*A+C	2F8B	INT-TO-FP	2D3B	P-PRINT	1A9C	SA-BIT-1	0514
CALCULATE	335B	IX-END	3290	P-RANDOM	1AB5	SA-BIT-2	0511
CALL-SUB	15F7	JUMP	3686	P-READ	1AC8	SA-BLANK	0629
CASES	37FA	JUMP-C-R	1C16	P-REM	1AA5	SA-BYTES	04C2
CAT-ETS	1793	JUMP-TRUE	368F	P-RESTORE	1ACF	SA-CODE-	0603
CD-PRMS1	247D	JUMP-Z	3687	P-RETURN	1A8D	SA-CODE-1	06E1
CH-ADD+1	0074	K-8-&-9	0382	P-RUN	1AAB	SA-CODE-2	06F0
CHAN-FLAG	1615	K- -CHAR	03B2	P-SAVE	1ADF	SA-CODE-3	06F5
CHAN-K	1634	K-CH-SET	02D1	P-STOP	1A8A	SA-CODE-4	06F9
CHAN-OP-1	1610	K-DECODE	0333	P-VERIFY	1AE1	SA-CONTRL	0970
CHAN-OPEN	1601	K-DIGIT	0367	PASS-BY	1E39	SA-DATA	0652
CHAN-P	164D	K-E-LET	0341	PAUSE	1F3A	SA-DATA-1	0692
CHAN-S	1642	K-END	0308	PAUSE-1	1F3D	SA-DELAY	053C
CHEK-END	1BEE	K-GRA-DGT	0389	PAUSE-2	1F49	SA-FLAG	04D0
CHR$	35C9	K-KLC-DGT	039D	PAUSE-END	1F4F	SA-LEADER	04DB
CIRCLE	2320	K-KLC-LET	034F	PEAR-ZERO	3159	SA-LINE	0716
CL-09-1	1CD6	K-LOOK-UP	034A	PE-OCTAUE	0427	SA-LINE-1	0723
CL-ADDR	0E9B	K-NEW	02F1	PEEK	34AC	SA-LOOP	04FE
CL-ALL	0DAF	K-REPEAT	0310	PERMS	1C96	SA-LOOP-P	0505
CL-ATTR	0E88	K-ST-LOOP	02C6	PF-ALL-9	2EB8	SA-NAME	0648
CL-CHAN	0D94	K-TEST	031E	PF-BITS	2E7D	SA-NULL	0644
CL-CHAN-A	0DA0	K-TOKENS	0364	PF-BYTES	2E8A	SA-OUT	051C
CL-LINE	0E44	KEY-3KEYS	029F	PF-COUNT	2F2D	SA-PARITY	050E
CL-LINE-1	0E4A	KEY-BITS	02A1	PF-DC-OUT	2F5E	SA-SCR$	06A0
CL-LINE-2	0E4D	KEY-CHAN	1113	PF-DEC-oC	2F64	SA-SET	051A
CL-LINE-3	0E80	KEY-CONTR	10FA	PF-DIGITS	2EA1	SA-SPACE	0621
CL-SC-ALL	0DFE	KEY-DATA	1105	PF-E-POS	2F83	SA-START	0507
CL-SCR-1	0E05	KEY-DONE	02AB	PF-E-SBRN	2F4A	SA-SYNC-1	04EA
CL-SCR-2	0E0D	KEY-FLAG	10F4	PF-E-SIGN	2F85	SA-SYNC-2	04F2
CL-SCR-3	0E19	KEY-INPUT	10A8	PF-FR-DGT	2EEC	SA-TYPE-0	073A
CL-SCROLL	0E00	KEY-INT	0048	PF-FR-EXX	2EEF	SA-TYPE-3	0710
CL-SET	0DD9	KEY-LINE	0296	PF-FRACTN	2ECF	SA-V-NEW	0685
CL-SET-1	0DEE	KEY-MOCL	10DB	PF-FRMT	2F6C	SA-V-OLD	0672
CL-SET-2	0DF4	KEY-MODE	10E6	PF-FRN-LP	2EDF	SA-V-TYPE	068F
CLASS-00	1C10	KEY-NEXT	110D	PF-INSERT	2EA9	SA/LD-END	0554

CLASS-01	1C1F	KEY-SCAN	028E	PF-LARGE	2E56	SA/LD-RET	053F
CLASS-02	1C4E	KEYWOARD	02BF	PF-LOOP	2E01	SAUE-ETS	0605
CLASS-03	1C0D	L-ADD$	2BAF	PF-MEDIUM	2E6F	SCAN-ENT	336C
CLASS-04	1C6C	L-CHAR	2B3E	PF-MORE	2ECB	SCAN-LOOP	1B52
CLASS-05	1C11	L-DELETE$	2B72	PF-NEGTVE	2DF2	SCANNING	24FB
CLASS-09	1CBE	L-EACH-CH	2B0B	PF-OUT-DT	2F59	SEC-PLUS	3575
CLASS-0B	1CDB	L-ENTER	2BA6	PF-OUT-LP	2F52	SECND-LOW	3568
CLEAR	1EAC	L-EXISTS	2B66	PF-POSTVE	2DF8	SEPARATOR	1B6F
CLEAR-1	1EB7	L-FIRST	2BEA	PF-R-BACK	2F25	SERIES-06	3449
CLEAR-3	1EDC	L-IN-W/S	2BA3	PF-RND-LP	2F18	SET-DE	1195
CLEAR-PRB	0EDF	L-LENGTH	2B9B	PF-ROUND	2F0C	SET-HL	1190
CLEAR-RUN	1EAF	L-NEWS	2BC0	PF-SAVE	2E1E	SET-MIN	16B0
CLEAR-SP	1097	L-NO-SP	2B0C	PF-SMALL	2E24	SET-STK	16C5
CLOSE	16E5	L-NUMERIC	2B59	PF-TEST-2	2EB3	SET-WORK	16BF
CLOSE-1	16FC	L-SINGLE	2B4F	PIXEL-ADD	22AA	SF-AGRMTS	27D9
CLOSE-2	1701	L-SPACES	2B29	PL-TST-IN	22FD	SF-ARG-LP	2843
CLS	0D6B	L-STRING	2BC6	PLOT	22DC	SF-ARG-VL	2852
CLS-1	0D87	L-TEST-CH	2B1F	PLOT-END	2303	SF-ARGMT1	2802
CLS-2	0D89	LAST	386C	PLOT-LOOP	22F0	SF-BRKT-1	27D0
CLS-3	0D8E	LD	0055	PLOT-SUB	22E5	SF-BRKT-2	27E4
CLS-LOWER	0D6E	LD	05E7	PO-1-OPER	0A7A	SF-CP-DEF	2814
CO-TEMP-1	21E1	LD	30E5	PO-2-OPER	0A75	SF-FLAG-6	27E9
CO-TEMP-2	21E2	LD-8-BITS	05CA	PO-ABLE	0AD9	SF-FND-DF	2808
CO-TEMP-3	21F2	LD-BLOCK	0802	PO-ALL-6	0BD3	SF-NOT-FD	2825
CO-TEMP-4	21FC	LD-BYTES	0556	PO-ANY	0B24	SF-R-BR-2	2885
CO-TEMP-5	2211	LD-CH-PR	07AD	PO-AT-ERR	0AAC	SF-RPRT-C	27E6
CO-TEMP-6	2228	LD-CONT-1	0819	PO-AT-SET	0ABF	SF-RUN	27F7
CO-TEMP-7	2234	LD-CONT-2	0825	PO-ATTR	0BDB	SF-SYS-EN	27F4
CO-TEMP-8	223E	LD-CONTRL	0808	PO-ATTR-1	0BFA	SF-VALUE	288D
CO-TEMP-9	2246	LD-DATA	082E	PO-ATTR-2	0C08	SF-VALUES	2381
CO-TEMP-A	2257	LD-DATA-1	084C	PO-BACK-1	0A23	SFA-CP-VR	296B
CO-TEMP-B	2258	LD-DEC	05C4	PO-BACK-2	0A38	SFA-END	2991
CO-TEMP-C	2273	LD-EDGE-2	05E3	PO-BACK-3	0A3A	SFA-LOOP	295A
CO-TEMP-D	227D	LD-FLAG	05B3	PO-CHANGE	0A80	SFA-MATCH	2981
CO-TEMP-E	2287	LD-LEADER	0530	PO-CHAR	0B65	SGN	3492
CODE	3669	LD-LOOK-H	0767	PO-CHAR-2	0B6A	SHIFT-FP	2FDD
CONTINUE	1E5F	LD-LOOP	05A9	PO-CHARS-3	0B76	SHIFT-LEN	3055
COPY	0EAC	LD-MARKER	05C8	PO-COMMA	0A5F	SHIFT-ONE	316E
COPY-1	0EB2	LD-NAME	07A6	PO-EACH	0C22	SIGN-FLAG	2CF2
COPY-2	0EC9	LD-NEXT	05C2	PO-ENTER	0A4F	SIGN-TO-C	3507
COPY-3	0ED3	LD-PROG	0873	PO-F-PR	0B1D	SIN	3785
COPY-BUFF	0ECD	LD-PROG-1	08AD	PO-FETCH	0B03	SIN	3785
COPY-END	0EDA	LD-SAMFLE	05ED	PO-GR-1	0B38	SING-DONE	2CFE
COPY-L-1	0EFD	LD-START	056C	PO-GR-2	0B3E	SKIP-CONS	33F7
COPY-L-2	0F0C	LD-SYNC	058F	PO-GR-3	0B4C	SKIP-NEXT	33F8
COPY-L-3	0F14	LD-TYPE	078A	PO-MSG	0C0A	SKIP-OVER	007D
COPY-L-4	0F18	LD-VERIFE	05BD	PO-NABLE	0C14	SKIP-ZERO	315E
COPY-L-5	0F1E	LD/BREAK	056B	PO-QUEST	0A69	SKIPS	0089
COPY-LINE	0EFA	LD/WAIT	0574	PO-RIGHT	0A3D	SL-DEFINE	2A94
COS	37AA	LEN	3674	PO-SAVE	0C3B	SL-OVER	2AAB
COUN-ONE	31FA	LESS-0	3506	PO-SCR	0C55	SL-RPT-C	2A7A
CP-LINES	1980	LESS-MASK	328A	PO-SCR-2	0C88	SL-SECOND	2A81
D-L-DIAG	24D4	LET	2AFF	PO-SCR-3	0CD2	SL-STORE	2AAD
D-L-HR-VT	24D8	LINE-AD-1	1974	PO-SCR-4	0D02	SLICING	2A52
D-L-LOOP	24CE	LINE-ADDR	196E	PO-SCR-4A	0D1C	SMALL	37F8
D-L-PLOT	24EC	LINE-DRAW	2477	PO-SCR-4B	0D2D	SQR	384A
D-L-RANGE	24F7	LINE-END	1BB3	PO-SEARCH	0C41	ST-E-PART	2CFF
D-L-STEP	24DF	LINE-NEW	1B9E	PO-SPACE	0AD0	ST-MEM-0	342D
D-LETTER	2C1F	LINE-NO	1695	PO-ST-E	0AF0	STACK-A	2D28
D-RPORT-C	2C05	LINE-NO-A	1961	PO-ST-PR	0AFC	STACK-BC	2D2B
D-RUN	2C15	LINE-RUN	1B8A	PO-STER	0C44	STACK-NUM	33B4
D-SIZE	2C2D	LINE-SCAN	1B17	PO-STORE	0ADC	START	0000

DATA	1E27	LINE-USE	1BBF	PO-T	0B5F	START/NEW	11CB
DATA-2	1E37	LINE-ZERO	168F	PO-T&UDG	0B52	STK-CODE	3671
DE,(DE+1)	2AEE	LIST	17F9	PO-TAB	0AC2	STK-CONST	33C8
DEC-JR-NZ	367A	LIST-1	17F8	PO-TOKENS	0C10	STK-DATA	33C6
DEC-RPT-C	2CCF	LIST-2	1814	PO-TRSP	0C35	STK-DIGIT	2D22
DEC-STO-1	2CD5	LIST-3	181A	PO-TV-1	0A7D	STK-F-ARG	2951
DEC-TO-FP	2C9B	LIST-4	181F	PO-TV-2	0A6D	STK-FETCH	2BF1
DECIMAL	2CCB	LIST-5	1822	POINT-LP	22D4	STK-HALF	32CC
DEF-FN	1F60	LIST-ALL	1835	POINT-SUB	22CB	STK-ONE	32C8
DEF-FN-1	1F6A	LLIST	17F5	POINTERS	1664	STK-PI/2	32CE
DEF-FN-2	1F7D	LN	3713	POKE	1E89	STK-PNTRS	35BF
DEF-FN-3	1F86	LN-FETCH	190F	PR-ALL	0B7F	STK-ST-0	2AB1
DEF-FN-5	1F94	LN-STORE	191C	PR-ALL-1	0B93	STK-STO-$	2AB2
DEF-FN-6	1FA6	LOC-MEM	3406	PR-ALL-2	0BA4	STK-STORE	2AB6
DEF-FN-7	1FBD	LOG(2^A)	2DC1	PR-ALL-3	0BB6	STK-TEN	32D3
DELETE	33A1	LOOK-P-1	1D8B	PR-ALL-4	0BB7	STK-TO-A	2314
DIFFER	19DD	LOOK-P-2	1DA3	PR-ALL-5	0BC1	STK-TO-BC	2307
DIM	2C02	LOOK-PROG	1D86	PR-AT-TB	201E	STK-VAR	2996
DIM-CLEAR	2C7C	LOOK-VARS	2882	PR-END-Z	2045	STK-ZERO	32C5
DIM-SIZES	2C7F	LPRINT	1FC9	PR-ITEM-1	1FFC	STK-ZERO	341B
DIV-34TH	31DB	MAIN-1	12A9	PR-ITEM-2	200E	STK-ZEROS	33F1
DIV-LOOP	31D2	MAIN-2	12AC	PR-ITEM-3	2024	STMT-L-1	1B29
DIV-START	31E2	MAIN-3	122F	PR-POSN-1	204E	STMT-LOOP	1B28
DIVISION	31AF	MAIN-3	1313	PR-POSN-2	2061	STMT-NEXT	1BF4
DIVN-EXPT	313D	MAIN-4	1303	PR-POSN-3	2067	STMT-RET	1B76
DL-LARGER	24CB	MAIN-5	133C	PR-POSN-4	206E	STOP	1CEE
DL-X-GE-Y	24C4	MAIN-6	1373	PR-ST-END	2048	STR$	361F
DOUBLE-A	338C	MAIN-7	1376	PR-STRING	203C	STR-$-NO	352D
DR-3-PRMS	238D	MAIN-8	1384	PRB-BYTES	0EE7	STR-ALTER	2070
DR-PRMS	23C1	MAIN-9	1386	PREP-ADD	2F9B	STR-DATA	171E
DR-SIN-NZ	23A3	MAIN-ADD	155D	PREP-M/D	30C0	STR-DATA1	1727
DRAW	2382	MAIN-ADD1	157D	PRINT	1FCD	STR-TEST	3588
DRAW-LINE	24B7	MAIN-ADD2	15AB	PRINT-1	1FCF	STRINGS	3560
DRW-STEPS	2420	MAIN-EXEC	12A2	PRINT-2	1FDF	STRS-ADD	359C
E-DIVSN	2D6D	MAKE-EXPT	313B	PRINT-3	1FE5	STRT-MLT	3125
E-END	2D7B	MAKE-ROOM	1655	PRINT-4	1FF2	STVT-R-1	1B7D
E-FETCH	2D6E	MASK-INT	0038	PRINT-A-1	0010	SUBN-ONLY	31F2
E-FORMAT	2CEB	ME-CONTRL	08B6	PRINT-A-2	15F2	SUBTRACT	300F
E-FP-JUMP	2D18	ME-ENT-1	093E	PRINT-CR	1FF5	SV-ARRAYS	29AE
E-LINE-NO	19FB	ME-ENT-2	0955	PRINT-FP	2DE3	SV-CH-ADD	29E0
E-LOOP	2D60	ME-ENT-3	0958	PRINTOUT	09F4	SV-CLOSE	29D8
E-SAVE	2D55	ME-ENTER	092C	PTR-DONE	167F	SV-COMMA	29C3
E-TO-STEP	2D4F	ME-NEW-L2	08EB	PTR-NEXT	166B	SV-COUNT	29E7
E-TST-END	2D71	ME-NEW-LP	08D2	PUSH	23C4	SV-DIM	2A48
EACH-S-1	1990	ME-OLD-L1	08DF	R-1-STORE	365F	SV-ELEMS	2A2C
EACH-S-2	1998	ME-OLD-LP	08D7	RAM-CHEK	11DA	SV-LOOP	29EA
EACH-S-3	199A	ME-OLD-V1	0901	RAM-DONE	11EF	SV-MULT	29F8
EACH-S-4	19A5	ME-OLD-V2	0909	RAM-FILL	11DC	SV-NUMBER	2A22
EACH-S-5	19AD	ME-OLD-V3	0912	RAM-READ	11F2	SV-PTR	29C0
EACH-S-6	19B1	ME-OLD-V4	091E	RAM-SET	1219	SV-RPT-C	2A12
EACH-STMT	198B	ME-OLD-VR	08F9	RAND-1	1E5A	SV-SIMPLE$	29A1
ED-AGAIN	0F30	ME-VAR-L1	0921	RANDOMIZE	1E4F	SV-SLICE	2A45
ED-BLANK	1150	ME-VAR-L2	0923	RE-ENTRY	3365	SV-SLICE?	2A49
ED-C-DONE	117C	ME-VAR-LP	08F0	RE-ST-TWO	3293	SWAP-BYTE	343E
ED-C-END	117E	MLT-LOOP	3114	RE-STACK	3297	SYNTAX-Z	2530
ED-CONTR	0F6C	MOVE-FP	33C0	READ	1DED	T-EXPNENT	326C
ED-COPY	111D	MULT-LONG	30F0	READ-1	1E0A	T-FIRST	3233
ED-CUR	1011	MULT-OFLW	30EF	READ-2	1E1E	T-GR-ZERO	3221
ED-DELETE	1015	MULT-RSLT	30EA	READ-3	1DEC	T-NUMERIC	3252
ED-DOWN	0FF3	MULTIPLY	30CA	READ-IN	3645	T-SHIFT	3261
ED-EDGE	1031	N-MOD-M	35A0	REC-EDIT	16D4	T-SMALL	323F
ED-EDGE-1	103E	N-NEGTV	3705	RECLAIM-1	19E5	T-STORE	3267

ED-EDGE-2	1051	NEG-BYTE	2FAF	RECLAIM-2	19E8	T-TEST	325E
ED-EDIT	0FA9	NEG-TEST	3474	REM	1BB2	TAN	37DA
ED-END	1026	NEGATE	346E	REMOVE-FP	11A7	TEMP-PTR1	0077
ED-ENTER	1024	NEW	11B7	REPORT	0552	TEMP-PTR2	0078
ED-ERROR	107F	NEXT	1DAF	REPORT	2244	TEMPS	0D4D
ED-FULL	1167	NEXT-0-1	19C7	REPORT-0	160E	TEMPS-1	0D5B
ED-GRAFH	107C	NEXT-0-3	19D5	REPORT-0	1725	TEMPS-2	0D65
ED-IGNORE	101E	NEXT-0-4	19D6	REPORT-0	1BB0	TEST-CHAR	001C
ED-KEYS	0F92	NEXT-0-5	19DB	REPORT-1	1D84	TEST-NEG	307C
ED-LEFT	1007	NEXT-02	19CE	REPORT-1	1DD8	TEST-NORM	3155
ED-LIST	106E	NEXT-1	1DE2	REPORT-2	0670	TEST-ROOM	1F05
ED-LOOP	0F38	NEXT-2	1DE9	REPORT-2	1C2E	TEST-ZERO	34E9
ED-RIGHT	100C	NEXT-2NUM	1C79	REPORT-3	2A20	TO-POWER	3851
ED-SIMBOL	1076	NEXT-LINE	1BD1	REPORT-4	1F15	TRUNCATE	3214
ED-SPACES	115E	NEXT-LOOP	1DDA	REPORT-5	0C86	TWO-P-1	1EBE
ED-STOP	1001	NEXT-ONE	19B8	REPORT-6	31AD	TWO-PARAM	1E85
ED-UP	1059	NIL-BYTES	3272	REPORT-6	3703	UNSTACK-Z	1FC3
EDITOR	0F2C	NO-&-NO	3524	REPORT-7	1F36	USE-252	2495
END-CALC	3698	NO-ADD	311B	REPORT-8	15E4	USE-ZERO	1CE6
END-COMPL	30A3	NO-L-EQL	3538	REPORT-8	2477	USR-$	34BC
END-TESTS	358C	NO-RESET	0070	REPORT-A	34E7	USR-NO	34B3
ENT-TABLE	338E	NO-RSTORE	31F9	REPORT-A	371A	USR-RANGE	34D3
ERROR-1	0008	NORMALISE	316C	REPORT-B	1E9F	USR-STACK	34E4
ERROR-2	0053	NORML-NOW	3186	REPORT-B	35DC	V-80-BYTE	2932
EX-OR-NOT	3543	NOT	3501	REPORT-C	1C8A	V-CHAR	28D4
EX-OR-STR	354E	NOT-BIN	2CB8	REPORT-C	21CE	V-EACH	2900
EXCHANGE	343C	NUMBER	18B6	REPORT-D	0D00	V-END	294B
EXIT	36C2	NUMERIC	2D1B	REPORT-E	1E08	V-FOUND-1	293E
EXP	36C4	NXT-DGT-1	2CDA	REPORT-F	0642	V-FOUND-2	293F
EXPT-1NUM	1C82	NXT-DGT-2	2D40	REPORT-F	1765	V-GET-PTR	2929
EXPT-2NUM	1C7A	OFLOW-CLR	3195	REPORT-G	1555	V-MATCHES	2912
EXPT-EXP	1C8C	OFLW1-CRL	3146	REPORT-H	21D4	V-NEXT	292A
F-FOUND	1D7C	OFLW2-CLR	3151	REPORT-J	15C4	V-PASS	2943
F-L&S	1D34	ONE	386A	REPORT-L	1B7B	V-RPORT-C	360C
F-LOOP	1D64	ONE-SHIFT	2FE5	REPORT-M	1EDA	V-RUN	28FD
F-REORDER	1D16	ONE-SPACE	1652	REPORT-N	1BEC	V-RUN/SYN	28EF
F-USE-1	1D10	OPEN	1736	REPORT-P	2812	V-SPACES	2913
FETCH-NUM	1CDE	OPEN-2	175D	REPORT-Q	288B	V-STR-VAR	29DE
FETCH-TWO	2FBA	OPEN-3	1767	REPORT-R	0806	V-SYNTAX	2934
FIND-I-1	1E9C	OPEN-END	178B	RESERVE	169E	V-TEST-FN	28E3
FIND-INT1	1E94	OPEN-K	1781	RESET	0066	VAL	35DE
FIND-INT2	1D99	OPEN-P	1789	REST-RUN	1E45	VAL-FET-1	1C56
FIRST-3D	3380	OPEN-S	1785	RESTK-SUB	3296	VAL-FET-2	1C59
FN-SKPOVR	28AB	OR	351B	RESTK-SUB	3296	VALID	371C
FOR	1D03	OTHER-STR	35B7	RESTORE	1E42	VAR-A-1	1C22
FORM-EXP	33DE	OUT	1E7A	RESULT-OK	370C	VAR-A-2	1C30
FP-0/1	350B	OUT-C-1	18F3	RETURN	1F23	VAR-A-3	1C46
FP-A-END	2DE1	OUT-C-2	1909	RO-CONT	0A87	VR-CONT-1	07E9
FP-CALC	0028	OUT-CH-1	195A	RO-SCR-3A	0CF0	VR-CONT-2	07F4
FP-CALC-2	33A2	OUT-CH-2	1968	RS-NRMLSE	32B1	VR-CONT-3	0800
FP-DELETE	2DAD	OUT-CH-3	196D	RS-STORE	32BD	VR-CONTRL	07CB
FP-TO-A	2DD5	OUT-CHAR	1937	RSLT-ZERO	370E	WAIT-KEY	15D4
FP-TO-BC	2DA2	OUT-CODE	15EF	RSTK-LOOP	32B2	WAIT-KEY1	14DE
FREE-MEM	1F1A	OUT-CURS	18E1	RUN	1EA1	X-LARGE	326D
FRST-LESS	3585	OUT-FLASH	18C1	S-2-COORD	2522	X-NEG	36B7
FULL-ADDN	303E	OUT-LINE	1855	S-ALFNUM	2684	XISO	385D
G-LOOP	3453	OUT-LINE	1865	S-ATTR	2672	ZERO-RLST	315D
GEN-ENT-1	335E	OUT-LINE2	187D	S-ATTR-S	2500	ZEROS-4/5	2FFB
GEN-ENT-2	3362	OUT-LINE3	1881	S-BRACKET	25E8	ZPLUS	37A1
GET-ARGT	3783	OUT-LINE4	1894	S-CONT-1	26DD		
GET-CHAR	0018	OUT-LINE5	18A1	S-CONT-2	2712		

К О П И Р О В Щ И К И

COPY - COPY

CAT - клавиша "C" -просмотр содержимого магнитной ленты.
LOAD - клавиша "J" -загрузка файлов в память:
LOAD X - загрузить файл в память на место X-го файла.
LOAD X TO XX - загружаются файлы с номерами от X до XX.
LOAD AT XX - загружать файлы с адреса XX. По умолчанию файл с номером 1 загружается по адресу 23296.
LOAD (XX - считывание первых XX байтов файла.Пример:
LOAD (6912 - считывание только экранной области.
SAVE - клавиша "S" - сохранение загруженных файлов на м/л.
SAVE - сохранение всех загруженных файлов без пауз.
SAVE X - сохранить файлы, начиная с номера X.
SAVE X TO XX - сохранить файлы с номерами от X до XX.
SAVE TO XX - сохранить файлы с номерами от 1 до XX.
SAVE STEP X - сохранение всех загруженных файлов. Между файлами делать паузы X секунд.
SAVE X TO XX STEP XXX - сохранение файлов с номерами от X до XX с паузами между файлами XXX секунд.
VERIFY - клавиша "V" - проверка сохраненных файлов:
VERIFY - аналогично SAVE.
VERIFY X TO XX - аналогично SAVE X TO XX.
VERIFY X - аналогично LOAD X.
LET - клавиша "I" - изменение полей заголовка файла. Например:
LET 3=500,1,5 - файл с номером 3 будет иметь длину 500 байт, стартовый адрес 1, длина программы 5 байт.
LIST - клавиша "K" - распечатка памяти:
LIST [XX] - XX задает адрес памяти. Если адрес не задан, то он равен 0. По этой команде выводится 15 байт памяти.
POKE - клавиша "O" -изменение десятичного значения байта:
POKE X,AA - X - адрес, AA - десятичное значение байта.
USR - клавиша "U" - вызвать программу пользователя:
USR X - вызвать программу пользователя по адресу X.
RETURN - клавиша "Y" -возврат в BASIC.
Copy - клавиша "Z" - копирование больших программ:
"CAPS SHIFT" - выгрузка программы необходимое число раз.
Copy NN - копирование файлов длиной до 49152 байт. Копирование выполняется только один раз.

Условные обозначения типов файлов:

P - программа;
B - машинные коды (BYTES);
A - числовой массив;
$ - символьный массив;
# - числовой массив.

"ENTER" - повторяет предыдущую команду
"CAPS SHIFT + 0" - уничтожает набранную строку.

TF COPY

Программа TF COPY (TAPE FILE COPY) предназначена для копирования информации на магнитную ленту. После окончательной загрузки программы появляется стартовое меню: (звездочкой отмечены мерцающие цифры - режим по умолчанию)

```
      0 - START                    - старт
      1 - CLOCK - SET              - вкл. часов
    * 2 - MODE - SELECT            - выбор режима
      3 - RENAME                   - перенаименование
      4 - 41984 BYTES - 14 LINES .
    * 5 - 44032 BYTES -  6 LINES.
```

6 - 44288 BYTES - ATTR USED

До нажатия на клавишу 0, клавиши 4,5,6 выбирают режим вывода на экран каталога и объем памяти.

После нажатия на клавишу 0:

LOAD (клавиша L) - загрузить файл(ы) в память.

Процесс загрузки прерывается одновременным нажатием клавиш CAPS SHIFT и BREAK SPACE.

SAVE (клавиша S) - выгрузить файл(ы).

DELETE (клавиша D) - удалить файл из каталога.

VERIFY (клавиша V) - сравнить файл, находящийся в памяти и записанный на магнитную ленту.

MODE (клавиша M) - модификация режима вывода программы с очисткой памяти.

RENAME (клавиша R) - перенаименование программ. После нажатия клавиши необходимо ввести номер файла.

CLOCK (клавиша C) - установка параметров ввода информации с магнитофона.

ALL (клавиша A) - выдает команду LOAD, SAVE, DELETE, VERIFY на все файлы, введенные в ОЗУ.

C O P Y / 8 6 M

LOAD (клавиша "L") - загрузка файлов в память с очередным номером исходной свободной памяти - 45 000 байт.

Copy (клавиша "C") - перевод программы в специальный режим копирования.

COPY/PAUSE (клавиша "M") - режим копирования файлов с паузой в 6 секунд между файлами.

VERIFY (клавиша "V") - проверка скопированных файлов.

Клавиша "X" - отмена ранее заданной операции с файлами.

Клавиша "S" - просмотр загруженных файлов.

ALL (клавиша "A") - применяется при операциях со всеми загруженными файлами.

DELETE (клавиша "D") - удаление ранее загруженных файлов.

Адресация (клавиша "H") - триггерная команда. При нажатии вместо десятичного отображения свободной памяти появляется шестнадцатиричное.

PRINT (клавиша "B") - распечатка загруженных файлов (BASIC)

"ENTER" - перевод в рабочий режимы команд копировщика.

"BREAK" -переход из одного режима в другой.

COPIER FM-3

Основные управляющие клавиши

клавиша	команда	производимое действие
A	AUTO	копировать все файлы от указателя
C	COPY	копировать файл под указателем
S	SKIP	передвинуть указатель
R	RESET	стереть все программы
BREAK	BREAK	включить режим копирования
CAPS+1	SOUND	вкл./выкл. Звук
SYMB+4	RADIX	формат чисел 16/10
CAPS+E	EXIT	выход в бейсик

прекратить запись любого файла можно клавишей BREAK.

SINCLAIR copy

После загрузки на экране таблица:

1-я колонка (T) - код обозначающий тип файла:

0 - программа на бейсике;

1 - числовой массив;

2 - текстовый массив;

3 - программа в машинных кодах.

2-я колонка (NEV) - имя файла (не более 10 знаков)

3-я колонка (HOSSZ) - длина в байтах в 16-ричн.форме.

4-я колонка (TCIM) - нач.адрес файла в памяти компьютера.

5-я колонка (BYTE) - факт.длина файла в 16-ричн.Предст-ии.
6-я колонка (MEM) - нач. адрес записи файла в программе.
После считывания нажмите BREAK.

RESTART (клавиша R): - очищается память программы.

END (клавиша E): - очищается память компьютера от программы SINCLAIR COPY. Процедура аналогична нажатию кнопки RESET.

COPY (клавиша C): - приготовьте магнитофон к записи. По нажатию "C" начинается перезапись из памяти на ленту.

SKIP (клавиша S): - выбор файла для перезаписи.

S P E C . C O P Y

Перезапись производится по одному блоку. Сначала чтение, затем запись.

M R . C O P Y

По нажатию кл."0" Появляется главное меню:

1 - загрузка оригинального файла;
2 - запись копии на кассету;
3 - верификация записанной копии;
4 - чтение заголовков файлов;
5 - загрузка и запись копии файлов без заголовков;
6 - сброс программы и выход в бейсик.

При нажатии BREAK в любом режиме MR. COPY: "Нажата клавиша BREAK - нажмите "0". По нажатию "0" программа возвращается к главному меню.

COPY DE LUXE

SPACE - пауза. Исходный режим: ожидание команды.

L - LOAD загрузка в память файлов с заголовком.

D - DATA загрузка блоков без заголовка.

H - HEADER просмотр заголовков файлов. Загрузка самих файлов не происходит.

S - SAVE запись на ленту, начиная с 1-го файла или с того, который выбран в режиме VIEW.

V - VIEW обзор файлов, находящихся в памяти (максимум 8) и выбор начала записи. Из этого режима можно перейти в режим изменения заголовка нажатием N-NAME.

K - CANCEL ликвидация последнего из загруженных файлов.

R - RESET полная очистка памяти копировщик.

Q - QUIT ликвидация копировщика.

M - MAXBYTE максимальное использование ОЗУ (без заголовка).

D - DATA. При нажатии на M выбирается один из трех режимов:

M1 (кл.1) - Многократная загрузка и запись ценой менее полного использования ОЗУ. L-LOAD, S-SAVE.

M2 (2) - однократная загрузка и многократная запись блока известной длины.

M3 (3) - однократная загрузка и многократная запись блока произвольной длины.

OMNI COPY-1

Копировщик программ и данных с магнитофона. Команды:

DELETE - TO RESET SYSTEM (сброс системы)
SPACE - TO STOP (останов инструкции)
Стрелки - выбор файла (курсорные стрелки "вверх", "вниз"
CS - продвижение инструкции
L - загрузка с магнитофона
S - запись на магнитофон
V - сравнение (проверка чтения)
C - TO COUNT
B - TO BREAK (прекращение считывания/записи)
K - TO KEEP
Q - уничтожение файла с наименьшим номером
H - создание заголовка
R - деактивизация строки запуска бейсик-программы
D - двойная скорость (плотность) записи/считывания
P - FOR LEADER PITCH
N - FOR MAXIFILE
M - FOR MEGAFILE

Копировщик защищенных программ LERM-7

Определения:

1.Пилоттон (иногда просто "тон", сокращенно - [T]).

При загрузке программы или при их записи на ленту вы видите в самом начале широкие синие и красные полосы. Это пилоттон [T]. Он обеспечивает надежность ввода кодов в память компьютера.

2.Джеркитон (сокращенно - [J]).

Это прерывистый пилоттон. Он был разработан для защиты от копирования программы. Вместо плавного перемещения красноголубых полос наблюдается их движение рывками, сопровождающееся отравистыми звуковыми сигналами. Этот способ довольно широко применяется в последнее время, например фирмами OCEAN, ULTIMATE и другими.

3.Байты (сокращенно [B]).

После тона идут байты. Они сопровождаются узкими синими и желтыми полосами на экране. Программист может сделать их другого цвета, а может совсем не выводить, используя свой загрузчик.

4.Блок кодов.

Пилоттон вместе с последующими данными называется блоком кодов

5.Скорость загрузки/выгрузки.

Обычно SPECTRUM выполняет LOAD и SAVE со вполне определенной скоростью. Как правило, это 1500 бод (бит/сек). Если вы увеличите темп и снизите время загрузки, вам придется расплачиваться снижением надежности ввода. Современные программы иногда умеют увеличть скорость ввода и LERM-7 может выполнить копирование таких программ.

Основные способы защиты программ:

а) использование очень длинных программ, более 48к;
б) применение различных скоростей ввода ("AUTOMENIA");
в) прерывистый пилоттон - джеркитон ("DECATHLON");
г) тоннинг ("CHUCKIE EGG");
д) короткий пилоттон;
е) комбинация различных методов ("DECATHLON");

Заголовок (HEADER)

Рассмотрим загрузку бейсик-программы. Сначала вы видите широкие красно-голубые полосы пилоттона, слышите звук. Это идет подготовка к загрузке. Затем идет короткая серия узких желто-синих полос. Это первый блок: [T][B]. Здесь [B] это заголовок (хедер). Он короткий, т.к. содержит только 17 байт. Именно поэтому на экране появляется сообщение:

```
PROGRAM: FRED или
  BYTES: FRED
```

После загрузки еще одного блока мы будем иметь:

```
      блок 1                    блок 2
-------------------- пауза --------------------
[T] [H] [T] [B] [H] - это хедер (раньше мы писали [B]).
```

Данные о хедере.

SPECTRUM имеет 17 байтов в хедере. Рассмотрим состав этих данных. Например:

```
      HEAD   : 0-250-10-20
```

Вы видите 4 числа. Их всегда должно быть 4.

Первое число:

- 0 - бейсик
- 1 - числовой массив
- 2 - стринговый массив
- 3 - коды

Отсюда видно, что пример, приведенный выше, записан в бейсике.

Третье число: в бейсике здесь задается номер строки, с которой программа должна автостартовать. Если это число больше, чем 32768, то автостарт не происходит.

В кодах, т.е. если первое число равно 3, здесь указывается на адрес, в который должен быть помещен первый байт.

Четвертое число : задает количество байтов бейсика.

Пример: HEAD: 3-780-36874-780. Задает блок в кодах. Длина его 780 байт. Загрузка начинается с адреса 36874.

Все программы должны начинаться с хедера, за которым идут байты: [T][H].

Инструкция к LERM-7

Цель программы: копирование очень длинных программ, программ с высокоскоростными загрузчиками, программ с джеркованным пилоттоном.

После загрузки экран темнеет и появляется меню:

L – загрузка нормальных хедеров и байтов

H – загрузка программ без хедеров

S – запись того, что загружено в режимах L и H с подсчетом байтов (счетчик)

Q – прекращение работы (Q=NEW) в вход в режим "B" для загрузки программ избыточной длины и программ со скоростными загрузчиками

J – вход в режим "J" для копирования программ с джерковаными пилоттонами

Режимы "H" и "L" копируют только части программ, записанные с нормальной скоростью, длиной до 38590 байтов, здесь копируется только одна часть [T][H] [T][B] или один блок без заголовка [T][B]. Этим она отличается от LERM-6. Наиболее целесообразно для нормальных блоков применять LERM-6, а для более трудных блоков оставлять LERM-7, работая с ними в режимах "B" и "J".

Копирование программ

А) загрузив ТС-7, установить программу, которую вы хотите скопировать.

Б) нажмите "L" на спектруме и "PLAY" на магнитофоне. Убедившись в том, что уровень сигнала установлен в "обычном" для загрузки программы положении.

В) после загрузки первой части [T][H] [T][B], остановите магнитфон. Вставьте чистую ленту, включите запись, нажмите "S" на компьютере. Чтобы выполнить вторую копию на другой ленте, нажмите "S" еще раз.

Г) дальнейшие части программы могут быть "нормальными", с хедерами и байтами, а могут быть без хедеров. Для копирования первых используйте "L", для вторых – "H". Если вы не уверены, пользуйтесь "H". Загрузив очередную часть, повторите пункт В.

Д) повторяйте пункт "Г" до тех пор, пока вам не встретится:

- часть с джерковаными пилоттонами;
- блок с увеличенной скоростью загрузки;
- очень длинный блок;
- блок с очень коротким пилоттоном.

Для работы с такими блоками вам надо пользоваться режимами "B" или "J". Для первого – "J", а для второго – "B".

Эта операция позволяет посчитать количество байтов. Она работает и на скоростных загрузчиках, но не с джерками.

Работа счетчика:

1) установить начало блока, подлежащего измерению;

2) нажать "PLAY" на магнитофоне и "C" на компьютере. Цвет бордюра изменится. После прогона блока на экране будет изображена его длина.

Внимание! Если после этого вы перейдете в рездим "B", то замеренная длина будет передана в этот режим, где есть свой собственный счетчик. Это сэкономит время, т.к. измеренне длины придется выполнять только один раз.

Если программа имеет блок данных и вы не уверены, сможет ли режим "H" загрузить ее, то после измерения "C" назначьте "H" , если блок менее 38к. Если блок больше 38к, переходите в режим "B", при этом измерять длину блока уже не нужно.

Кроме того, имейте в виду, что весьма важно знать длину блока до перехода в режим "B", т.к. выбор клавиши для его копирования зависит от длины блока.

Прекращение работы осуществляется клавишей "Q".

Фальшхедеры. Режим "L" не может загружать фальшхедеры, но это можно преодолеть, если копировать каждый блок по отдельности через "H".

[T][F] [T][B] здесь [F] – это фальшхедеры.

Вход в режимы "B" и "J".

Для этого нажмите "B" или "J". Бордюр станет красным, а на экране появится сообщение: "Нажать любую клавишу".

Замечание: вы обнаружите, что режимы "J" и "B" обычно смещают катинку на экранее на один квадрат. Это не ошибка. Во всем остальном картина должна быть нормальной. Если нет, то проверьте уровень громкости.

Когда бордюр станет красным, нажмите пробел. Экран станет черным, бордюр желтым. Если это не так, попробуйте еще раз.

Замечание: в режимах "B" и "J" клавиша "H" предназначена для измерения длины, что она и делает в специально выделенной для этого области памяти. Поэтому если вы намерены делать "SAVE" еще раз, ни для чего другого эту клавишу использовать нельзя, т.к в программе будут испорчены байты. Заметьте также, что если программа очень длинная, то счетчик "B" может быть "затерт" загружаемой программой. В этом случае клавиша "B" станет бесполезной.

Режим "J".

Этот режим работает с джеркитонами. Предупреждаем сразу, что это не просто, см. Раздел "Полезные советы". Эти загрузчики обычно являются также скоростными, но они запрограммированы, как правило, на одну и ту же скорость.

Меню "J"-режима следующее (на экране не показывается):

B - измеритель джерков;
L - первая загрузочная программа;
M - первая загрузочная программа;
S - клавиша SAVE;
S/O - "SAVE" с нормальными пилоттоном и скоростью
A - прерывание. Очищается экран, бордюр становится желтым
Q - сброс (аналогично "NEW").

Начало работы

Прежде всего вам надо померить "джерки" клавишей "B". Для этого перемотайте ленту и установите ее в начало джеркитона. Нажмите PLAY на магнитофоне и держите "B" на компьютере до тех пор, пока пройдет вся джеркованная часть программы. Как только пойдут байты, нажмите SPACE и выключите магнитофон. Если вы будете ждать слишком долго, программа может сбиться.

Фактически нет необходимости в перегоне всего джеркитона, можно начать и с середины. Важно, чтобы измеритель захватил и несколько последующих байтов. Замер можно повторить. Обычно программы имеют один джеркитон на каждый блок.

Загрузка частей с джерками.

Для этой цели служат две клавиши - "L" и "M". Более ранние программы обычно требуют "L", а более поздние - "M". Чтобы выбрать, какая именно клавиша вам нужна, надо:

А) установить программу в том же месте, где начинается загрузка экрана.

Б) нажать "PLAY" на магнитофоне и "L" на компьютере. Если цветовые атрибуты идут правильно, значит клавиша "L" выбрана верно.

В) в противном случае использовать "M".

Запись.

Установите чистую кассету, включите "RECORD" и нажмите клавишу "S" для записи с джеркитонами или для записи без них.

Клавиша "A".

Нажатие этой клавиши приводит к очистке экрана, но она не влияет на состояние измерителя джерков. Нажатием клавиши SPACE во время загрузки или выгрузки вы можете прервать работу программы и возвратиться к главному меню.

Режим "B".

В этом режиме есть возможность копирования программ со скоростными загрузчиками, в том числе и тех, которые при загрузке могут выполнять проверку экрана.

Меню "B"-режима:

L - загрузка блоков длиной до 48300 байтов
M - загрузка блоков от 48300 до 50к
N - загрузка блоков свыше 50к
J - загрузка блоков (только для нормальной скорости)
3 - загр.длинных блоков, начинающихся на экранной области
S - запись блоков
P - спец. клавиша для записи (используется после "C")
$ (доллар) - запись с нормальной скоростью и нормальном пилоттоном
C - подсчет числа байтов для работы с режимами "J" и "L"
B - измеритель числа скорости ввода
Q - сброс
A - прерывание и очистка экрана

Замечание: после использования клавиш J или 3 для загрузки и последующей выгрузки, программа автоматически выполняет NEW.

Изменение скорости ввода.

При первом входе в LERM в компьютере задана нормальная скорость ввода/вывода. Если блок имеет нестандартную скорость, то прежде чем что-либо делать, вам необходимо замерить скорость, для этого:

А) установите ленту в начале этого блока, который вы собираетесь копировать. Нажмите "PLAY" и пропустите пилоттон.

Б) когда пилоттон пройдет и пройдут байты, нажмите "B", постарайтесь найти достаточно "шумную" зону (при изображении экрана, например, встречаются подряд сотни одинаковых байтов, что воспринимается как один чистый звуковой тон).

Примечание: некоторые программы имеют различные скорости для различных блоков, поэтому измеритель надо применять всякий раз снова. Можно проверить, правильно ли производится замер скорости. Если в программе есть SCREEN$, попробуйте загрузить экран операцией "L". Если экран и атрибуты правильные, но сдвинуты на один элементарный квадрат, то значит замеренное значение - правильное.

Операции загрузки.

Самая большая проблема состоит в том, что кроме повышенной скорости блоки могут иметь и избыточную длину. Чтобы с ними справиться, существуют операции, которые загружают программу за счет порчи картины экрана. В то же время, есть программы, которые выполняют проверку экрана, бывают также программы более 48к.

А) клавиша "L".

Загружает блоки объемом до 48300 байтов. Картина экрана сдвигается на один элементарный квадрат. После копирования накакой порчи экрана нет. Это основной режим загрузки.

Б) клавиша "M".

Загружает блоки от 48300 до 50к. Картинка портится. Копия также будет иметь испорченную картинку. Тем не менее, если программа не проверяет состояние экрана, то она будет работать нормально. Этот режим - ваша первая линия атаки на блоки, превышающие 48.3К.

Все остальные режимы предполагают предварительное использование счетчика "C". Помните, что счетчик может быть задействован как до, так и после перехода в режим "B".

В) клавиша "N".

Загружает программы более длинные, чем "M", но искажение картинки здесь присутствует также.

Г) клавиша "J".

Применяется для блоков, которые загружаются с нормальной скоростью и имеют длину более 48.3К и не начинается с загрузки экрана. В этом случае картинка не портится.

Д) клавиша "3".

Применяется, как и клавиша "J", но в тех случаях, когда блок начинает загружаться с экрана.

Резюме. Итак, вам надо определить длину своего блока, используя "C" и выбрать загрузочную клавишу в зависимости от длины блока. Скорости загрузки и наличия проверки искажений экрана.

Что же делать, если программа имеет и скоростной загрузчик, и проводит проверку экрана одновременно? Для этого есть специальный обходной путь, см. Режим "P".

Загрузка.

Установить ленту на начало блока, который вы собираетесь копировать. Нажмите "PLAY" и соответствующую загрузочную клавишу. После загрузки остановите ленту.

Запись.

1) Клавиша "S" (нормальная запись).

Вставьте чистую ленту в магнитофон. Нажмите "RECORD" и затем "S". Программа будет записана с той же скоростью, с какой загружалась. Повторение нажатия "S" даст повторение выше.

2) Клавиша 0.

Если у вас программа со скоростным загрузчиком, а вы хотите конвертировать ее в нормальную, нажмите 0 до нажатия "S".

3) Клавиша "P" (используется с "C" и "L").

Если у вас есть скоростной блок, слишком длинный для "L", но в то же время использовать "M" нельзя, т.к. в программе есть проверка области экрана, то вы можете попробовать следующее (хотя все это метод проб и ошибок):

- замерить длину "C";

- загрузить через "L";
- записать через "P".

Заметьте, что счетчик "C" может быть запущен либо после главного меню (см. П.2), либо из режима "B" (см. п. 19 ниже).

Клавиша "C" для подсчета байтов.

В этом режиме клавиша "C" работает точно также, как и в п.2. После нажатия "PLAY" на магнитофоне просто держите клавишу "C". Хотя при этом отсчет байтов на экране не изображается.

Клавиша "Q".

Клавиша "Q" сбрасывает программу. Ее действие аналогично команде "NEW".

Клавиша "A".

Она очищает экран и устанавливает все в исходное положение положение.

Полезные советы.

Выполнение копий с джеркованных программ и программ со скоростными загрузчиками - это довольно сложная задача, потому что все прочие факторы становятся гораздо более критичными. Вам придется экспериментировать со своим аппаратным обеспечением. Обычно вопрос правильности ввода программы упирается в правильный подбор громкости сигнала.

1) Желательно иметь магнитофон попроще, по возмржности без атоматической регулировки уровня сигнала при записи. Головка должна быть чистой и точно установленной. Лента должна быть высококачественной, а оригинал - в достаточно пригодном состоянии. Если программа загужается только один раз из четырех, то вам придется потратить больше времени на ее ввод.

2) Загружайте программу в компьютер с уровнем громкости, который вы обычно используете для ввода программ.

3) Вполне возможно, что копия, которую вы выполните, будет иметь более высокий уровень, чем оригинал. Поскольку джеркованные программы и программы с ускоренными загрузчиками крайне чувствительны к уровню записи,вам придется поэкспериментировать с установкой уровня записи таким образом, чтобы копия имела надежную загрузку. Обычно мы рекомендуем:

А) для программ со скоростными загрузчиками проводить загрузку с пониженным уровнем (примерно на 20%) по сравнению с загрузкой нормальных программ. После того, как вы подберете нормальный уровень для ввода, отметьте его на ленте, чтобы потом долго не подбирать его.

Б) для программ с джеркованными пилоттнами - так же, как и для скоростных загрузчиков, или ниже (на 20-50%). На практике устанавливают уровень лишь чуть выше, чем это необходимо для приема джеркитона, т.е. так низко, как это возможно.

В) если экран испорчен (а не просто сдвинут), то нет смысла продолжать дальнейшую загрузку. Остановите ленту и измените уровень.

Г) поскольку уровень копии выше, чем уровень оригинала, возможно, что при работе с копией вам придется работать в меньшем диапазоне уровней громкости. Мы обнаружили, что для многих копий, в частности, с джерками, можно повысить надежность ввода, если соединить гнездо "EAR" магнитофона при загрузке с гнездом "MIC" компьютера вместо "EAR". Это возможно, т.к. и "EAR", и "MIC" могут использоваться для загрузки, но "MIC" требует более высокого уровня сигнала. Такой подход может позволить использовать более широкий диапазон установки уровня на магнитофоне при загрузке.

К сожалению, вам придется экспериментировать. Многие отмечают, что защищенные программы очень плохо вводятся. Количество программ с рекламациями очень высоко. Они вводятся с одного магнитофона, но не вводятся с другого.

4) Мы считаем, что как правило критичным является только один блок в программе, поэтому переделать бывает нужно только его.

5) Следует применять чистые (новые) ленты для записи программ. Если на ленте уже была какая-то запись, то ее магнитофон должен стереть, а стирающая головка не у всех магнитофонов достаточно эффективна, поэтому остаются фоновые шумы.

6) Будьте терпеливы. Осилить защищенные программы очень не просто. К тому же имейте в виду, что LERM не все программы в состоянии копировать.

P A S C A L

Компилятор занимает около 12к, в то время как при запуске программы используется дополнительно 4к, редактор занимает 2к. Таким образом, общий размер пакета около 19к, оставшаяся часть памяти использована под PASCAL-программы.

PASCAL использует различные управляющие коды вводимые с клавиатуры, в большинстве случаев из редактора . Конечно различные системы могут иметь разные клавиатуры и методы генерерирования управляющих кодов. Используемые в данном ENTER, CC, CH, CI, CP, CS и CX.

Когда нр ждет ввода, управляющие коды используются следующим образом:

ENTER	используется для ввода линии;
CC	возврат в редактор;
CH	удаление последнего напечатанного символа;
CI	табуляция;
CP	прямой выход на принтер (если подключен) или возврат на экран;
CX	удаление строки.

HISOFT PASCAL загружается в компьютер "LOAD", после успешной загрузки автоматически появляется сообщение:

TOP OF RAM?

Вам следует ответить либо вводом положительного десятичного числа до 65536, (с последующим нажатием ENTER), либо просто ENTER.

Если вы ввели номер, то он будет использоваться как наивысшая позиция ОЗУ +, 1, иначе эта величина будет вычислена автоматически. Стек компилятора будет устатановлен равным этой величине и, таким образом, вы можете зарезервировать на вершине памяти область ввводом величины меньше, чем действительная вершина ОЗУ.

ZX-SPECTRUM версии HISOFT PASCAL 'правильная' вершина ОЗУ берется равной началу области, отводимой под пользовательскую графику (UDG-по руководству к SINCLAIR). Сейчас на экране появилось сообщение:

TOP OF RAM FOR 'T'

Здесь вы можете ввести десятичное число. То, что вы введете, будет использовано как стек, когда результирующий об'ектный код будет получен после использования команды 'T' редактора (см.Рзд.4). Вам необходимо назначить стек, используемый работающей программой, отличным от вершины ОЗУ, если, например, у вас написано расширение для выполняемой программы и вы хотите сохранить его в верхней области ОЗУ.

В конце появится сообщение:

TABLE SIZE?

То, что введете сейчас, определит построение памяти под таблицу (с последующим нажатием ENTER) или просто ENTER. Если вы ввели ENTER, то вычисленная величина (свододная ОЗУ деленная на 16) будет установлена в качестве размера таблицы. Во многих случаях вычисленная величина резервирует более чем достаточно пространства под символы.

С этого момента компилятор и редактор будут перемещены в конец таблицы и управление будет передано редактору.

Компилирование и запуск.

Когда компилятор требует помощи, он генерирует сообщение в следующей форме

ПППП текст строки:

Где: # # # # адрес начала расположения кода, генерируемого этой линией. ПППП-номер мтроки содержащей ошибку.

Если строка содержит более 80 символов, компилятор образует новую строку, так как строка не может содержать более 80 символов.

Если потребуется, этот листинг может быть выведен на принтер использованием команды "P" (Рзд.3).

Вы можете остановить листинг нажатием CS; для возврата в редактор используйте CC; продолжить листинг можно нажатием любой клавиши.

Если во время компиляции будет обнаружена ошибка, то появится сообщение *ERROR*, знак '/\', указывающий на позицию после символа вызвавшего ошибку и номер ошибки (пр.1). Листинг остановится; нажмите 'E'для возврата в редактор, для того, чтобы отредактировать высвечиваемую линию или 'P' для редактирования предыдущей линии (если она есть) или любую другую клавишу для продолжения компиляции.

Если программа заканчивается некорректно (т.е. без END), появится сообщение 'NOMORE TEXT' (текста больше нет) и управление будет передано редактору.

Если редактор выйдет за границы таблицы , то появится сообщение'NO TABLE SPACE'(нет места в таблице) и управление будет передано редактору. В этом случае программист должен сохранить программу на ленте, перегрузить компилятор и задать большие размеры таблицы 'TABLE SIZE' (Рзд.0.0).

Если компиляция закончилась корректно, но содержит ошибки- появятся сообщения об ошибках и об'ектный код будет уничтожен . Если компиляция прошла успешно, то появится 'RUN?'. Если вы желаете запустить программу немедленно нажмите 'Y',в противном случае управление будет передано редактору.

В процессе работы программы могут появиться ошибки исполнения (см. Приложение 1). Вы можете приостановить выполнение, используя CS; прервать, воспользовавшись CC и продолжить, нажав любую другую клавишу.

Р е д а к т о р

Введение в редактор.

Примечание: в этом разделе клавиша 'DELETE' используется вместо управляющего кода CH.

Редактор вводится автоматически при загрузке PASCAL с ленты, и ожидает ввода команды высветив символ '>'.

Командная строка имеет следующий формат:

C N1,N2,S1,S2

после чего необходимо нажать ENTER, где:

C	выполняемая команда (раздел 4.2);
N1	число в диапазоне от 1 до 32767 включительно;
N2	число в диапазоне от 1 до 32767 включительно;
S1	строка символов максимальной длинны 20;
S2	строка символов максимальной длинны 20.

Запятая используется для разделения различных аргументов (хотя ее можно заменить – смотри команду 'S'), пробелы игнорируются, за исключением пробелов внутри строк символов ни один из аргументов не является обязательным, хотя некоторые команды (например'D'ELETE) не выполняются без указания N1, N2. Редактор помнит предыдущие числа и стоки,которые вы ввели и использует эти первые значения где возможно, если не задан конкретный аргумент внутри строки команды. Значение N1,N2 обычно устанавливает равными 10 и строки вначале пустые. Если вы ввели неправильную команду, например:

F-1,100,HELLO

команда будет игнорирована и появится сообщение

'PARDON?'

И вам необходимо ввести правильную строку

F1,100, HELLO

Это же сообщение появится если длина S2 превышает 20; если S1 длиннее 20,то любой избыточный символ игнорируется, команды могут вводится в верхнем или нижнем регистре.

При введении строки команды, все соответствующие управляющие коды, описанные в разделе 0.0, могут вводиться.

Следующий подраздел детализирует команды имеющиеся в редакторе. Он должен обязательно присутствовать для выполнения команды.

Команды редактора

Ввод текста

Текст может быть вставлен в текстовый файл, или печатая номер строки,пробел, а затем требуемый текст, или используя команду I.

Команда : I N,M

Использование этой команды обеспечивает ввод в автоматическом режиме: на экране автоматически высвечивается номера вводимых строк. Можно использовать все управляющие коды, ввод строки завершается нажатием 'ENTER' . Для выхода из режима используйте управляющий код 'CC'.

Если вы вводите строку с номером, который уже существует в тексте, существующая строка будет стерта и заменена на новую после нажатия клавиши 'ENTER'. Если автоматическое наращивание номера строки дает номер больший чем 32767, то редактор автоматически выходит из режима ввода.

Если при вводе текста вы достигли конца экрана, но при этом строка содержит

менее 128 символов (размер буфера), строки на экране поднимутся и вы можете печатать следующую строку.

4.2.2 Распечатка текста

Текст может быть распечатан, используя команду 'L'; количество строк изображаемых одновременно на экране в течение процедуры команды изначально фиксировано, но может быть изменено с помощью команды 'K'.

Команда : L N,M

Выводит текущий текст на экран со строки N до строки M включительно. По умолчанию длина для N всегда 1, а для M 32767, т.е. эти значения не берутся из предварительно введенных аргументов. Чтобы распечатать весь файл текста, используйте просто 'L' без всяких аргументов. Строки ограничены слева меткой ; справа границей экрана. Количеством строк экрана можно управлять с помощью команды 'K'. По выводу определенного количества строк текст остановится, введите управляющий код CC, для возврата в основной цикл редактора, или любую клавишу, для продолжения листинга.

Команда : K N

Команда устанавливает количество строк экрана, которое должно быть выведено перед тем, как вывод приостановится. Величина (N 256) вычисляется и хранится в памяти. Например, используйте ее, если хотите, чтобы команда L за один раз листала 5 строк.

Редактирование текста

Когда текст введен, возникает необходимость отредактировать некоторые строки. Для исправления, стирания, сдвига и перенумерации существуют следующие команды.

Команда : D<N,M>

все строки от N до M включительно стираются из текстового файла. Если M<N или описано менее двух аргументов команда выполняться не будет.

Команда : M N,M

в этом случае текст со строки N вводится на строку M; содержимое последней при этом стирается. Строка N при этом сохраняется.

Команда : N <N,M>

используется для перенумерации строк начиная с N шагом M. Оба параметра обязательны.

Команда : F N,M,F,S

в тексте заключенном между N<X<M ищется выражение F. Если такое найдено, то высвечивается эта строка и вводится режим редактирования.Вы можете использовать команды режима редактирования для поиска выражения F или замены на выражение S текущего значения F,а затем для поиска следующего положения выражения F.

Команда : E N

дает возможность редактировать строку N. Если номер не задан, то ничего не происходит, в противном случае строка заносится в буфер и высвечивается на экране (с номером) также номер располагается под строкой и вводится режим редактирования. Редактирование производится в буфере, а не в памяти поэтому в любой момент может быть вызвана первоначальная строка.

В этом режиме курсор как бы движется вдоль строки начиная с первого символа и различные подкоманды позволяют проводить редактирование. Эти подкоманды следующие :

(SPACE)	передвижение курсора вдоль строки
DELETE	возвращение курсора назад вдоль строки (нельзя продвичуть левее,
BACKSPACE)	если курсор стоит на первом символе)
Ci(упр.код)	передвигает курсор на очередную метку табуляции, но не до конца строки.
<ENTER>	закончить редактирование сохранив изменения
Q	отмена редактирования данной строки (оставляет строку в состоянии до режима редактирования)
L	просмотреть оставшуюся часть редактируемой строки; например справа от курсора.Остается режим редактирования с курсором вначале строки
K	стирает символ на котором находится курсор.
Z	стирает правую часть строки, включая символ на котором находится курсор
F	найти следующее положение выражения предварительно определенное

аналогичной командой. Эта команда автоматически переведет редактор на следующую строку, содержащую искомое выражение (сохранив все введенные изменения) если не будет найдено такого же выражения в текущей строке. Заметьте, что курсор расположен вначале найденного выражения в случае удачного поиска.

S заменить на предварительно замененные выражения, выражения найденные по команде F, а затем выполнить команду F, т.е.осуществить поиск следующей строки содержащей искомое выражение. Вместе с командой F используется для пошагового прохождения всего файла и замены выражений. (См. раздел 4.3)

I ввод символа в текущую позицию курсора; находитесь в этом подрежиме до нажатия клавиши ENTER и возврата в основной режим редактирования с курсором в позиции после последнего введенного символа. Используя DELETE(или BACKSPACE) можно стереть символ по левую сторону от курсора, а используя управляющий код CI - переместить курсор в следующую позицию табуляции.

X продвигает курсор к концу строки и автоматически вводит вставки подрежимов описанных выше.

C меняет подрежим. Позволяет переписать знак в позиции курсора и, затем, передвигает курсор на одну позицию. Работа в подрежиме продолжается до нажатия клавиши ENTER для возврату к основному режиму; при этом курсор располагается после последнего исправленного символа. В этом подрежиме DELETE(или BACKSPACE) просто сдвигает курсор влево, в то время как CI не действует.

Команды работы с магнитофоном

Текст может быть записан или загружен с помощью команд 'P' и 'G'.

Команда P N,M,S

Строки, номера которых находятся в диапазоне N<X<M, будут сохранены на ленте в формате HISOFT PASCAL под именем пределенным строкой S.

Команда G ,, S

В записи находится файл записанный в HISOFT PASCAL формате с именем S. Пока идет поиск на экране высвечивается : 'SEARCHING...'. Если в записи встретится файл в данном формате, но с другим именем, на экране появится надпись'FOUND' за которой следует имя встретившегося файла и поиск будет продолжен. По нахождению нужного файла он будет загружен в память. Если при загрузке обнаружится ошибка, то появится сообщение об этом и загрузка будет прервана.

В этом случае вам необходимо вернуть ленту назад и снова запустить G.

Если имя файла не задано, то загрузится первый найденый файл в PASCAL формате.

Пока продолжается поиск вы можете прерывать его с помощью CC и возвратиться в основной режим.

Если в памяти находится другой файл, то загружаемый с ленты файл будет присоединен к нему и образованный файл перенумерован начиная с номером 1 с шагом 1.

Компиляция и запуск программ из редактора

Команда: C N

Текст, начиная со строки N будет откомпилирован. Если номер не указан, то текст будет откомпилирован начиная с первой существующей строки (см.Рзд 0.2)

Команда: R

Предварительно откомпилированный объектный код будет исполнен, только если он не был утрачен (см.Рзд 0.2)

Команда: T N

Эта команда транслирования. Текущий источник компилируется со строки N (или сначала, если N отсутствует), если компиляция прошла успешно, на экране появится 'OK?'.

Другие команды

Команда: B

Управление возвращается к операционной системе

Команда: O N,M

Если вы имеете текст (возможно от другого редактора) который не закодирован, то используя команду 'O' вы можете его закодировать для дальнейшего использования с компилятором.

Команда: S.,D

Эта команда позволяет заменить разделитель, который используется для разделения аргументов в командной строке. При входе в редактор в качестве разделителя используется ',' ; он может быть изменен командой S на первый символ строки D.

Команда: V

Не имеет аргумента и высвечивает текущие назначеные величины диапазона строк и два выражения.

Команда: X

Высвечивается адрес компилятора в шеснадцатиричном коде.

Ошибки

Перечень ошибок генерируемых компилятором

.NUMBER TOO LARGE
.SEMI-COLON OR 'END' EXPECTED.
.UNDECLARED IDENTIFIER. (Необъявленный идентификатор)
.IDENTIFIER EXPECTED.
.USE'='NOT':='IN A CONCTANT DECLAGATION.
.'='EXPECTED.
.THIS IDENTIFIER CANOOT BEGIN A STATAMENT
.':='EXPECTED.
.')'EXPECTED.
0.WRONG TYPE.
1'.'EXPECTED.
2.FACTOR EXPECTED.
3.CONSTANT EXPECTED.
4.THIS IDDENTIFIER IS NOT A CONSTANT.
5.'THEN' EXPECTED.
6.'DO' EXPECTED.
7.'TO' OR 'DOWNTO' EXPECTED.
8.')' EXPECTED.
9.CANNOT WRITE THIS TYPE OF EXPRESSION.
20.'OF' EXPECTED.
21.',' EXPECTED.
22.':' EXPECTED.
23.'PROGRAMM' EXPECTED.
25.'BEGIN' EXPECTED.
26.VARIABLE EXPECTED IN CALL TO READ.
27.CANNOT COMPARE EXPRESSIONS OF THIS TYPE.
28.SHOULD BE EITHER INTEGER OR TYPE REAL.
29.CANNOT READ THIS TYPE OF VARIABLE.
30.THIS IDENTIFIER IS NOT A TYPE.
31.EXPONENT EXPECTED IN REAL NUMBER.
32.SCALAR EXPRESSION (NOT NUMERIC) EXPECTED.
33.NULL STRINGS NOT ALLOWED (USE CHR (0)).
34.'(' EXPECTED.
35.')' EXPECTED.
36.ARRAY INDECS MUST BE SCALAR.
37.'..' EXPECTED.
38.')'OR',' EXPECTED IN ARRAY DECLARATION.
39.LOWERBOUND GREATER THEN UPPERBOUND.
40.SET TOO LARGE (MORE THEN 256 POSSIBLE ELEMENTS).
41.FUNCTION RESULT MUST BE A TYPE IDENTIFIER.
42.',' OR ')' EXPECTED IN SET.
43.'..' OR ',' OR ')' EXPECTED IN SET.
44.TYPE OF PARAMETER MUST BE A TYPE IDENTIFIER.
45.NULL SET CANNO BE THE FIRST FACTOR IN A NON-ASSIGNMENT STATEMENT.
46.SCALAR (INCLUDING REAL) EXPECTED.
47.SCALAR (NOT INCLUDING REAL) EXPECTED.
48.SETS INCOMPATIBLE.
49.'<' AND '>' CANNOT BE USED TO COMPARE SETS.
50.'FORWARD','LABEL','CONST','VAR','TYRE' OR 'BEGIN' EXPECTED.

51.HAXADEZIMAL DIGIT EXPECTED.
52.CANNOT POKE SETS.
53.ARRAY TOO LARGE (>64к).
54.'END' OR ';' EXPECTED IN RECORD DEFINITION.
55.FIELD IDENTIFIER EXPECTED.
56.VRIABLE EXPECTED AFTER 'WITH'.
57.VARIABLE IN 'WITH' MUST BE OF 'RECORD' TYPE.
58.FIELD IDENTIFIER HAS NOT HAT ASOCIATED WITE STATEMENT.
59.UNSIGNED INTEGER EXPECTED AFTER 'LABEL'.
60.UNSIGNED INTEGER EXPECTED AFTER 'GOTU'.
61.THIS LEBEL IS ON THE WRONG LEVEL.
62.UNDECLARED LABEL.
63.THE PARAMETER OF SIZE SHOULD BE A VARIABLE.
64.CAN ONLY USE EQUALITY TESTS FOR POINTERS.
65.THE ONLY WRITE PARAMETER FOR INTEGERS WITH TWO ':' S IS E:M:H.
66.STRINGS MAY NOT CONTAIN END OF LINE CHARACTERS.
67.THE PARAMETER OF NEW, MARK OR RELEASE SHOULD BE A VARIABLE OF POINTER TYPE.
68.THE PARAMETER OF ADDR, SHOULD BE A VARIABLE.

Ошибки исполнения.

Когда при исполнении программы обнаруживается ошибка появляется одно из следующих сообщений, перед которым печатается 'AT PC=####', где #### - адрес памяти, где ошибка возникла. Также будет указан источник ошибки, если нет, то необходимо просмотреть листинг компиляции на предмет обнаружения места возникновения ошибки.

1.HALT (останов)
2.OVERFLOW (переполнение)
3.OUT OF RAM (мала озу)
4./BY ZERO (деление на ноль)
5.INDEX TOO LOW (индекс слишком мал)
6.INDEX TOO HIGH (индекс слишком велик)
7.MATHS CALL ERROR
8.NUMBER TOO LARGE (номер слишком велик)
9.NUMBER EXPECTED
10.LINE TOO LONG (длинная строка)
11.EXPONENT EXPECTED

Обнаружение ошибки исполнения прерывает выполнение программы.

Зарезервированные слова и предопределенные идентификаторы.

Зарезвированные слова

AND	ARRAY	BEGIN	CASE	CONST	DIV	DO	DOWNTO	ELSE	END
FORWARD	FUNCTION	GOTO	IF	IN	LABEL	MOD	NIL	NOT	OF
OR	PACKED	PROCEDURE		PROGRAMM	RECORD	REPET	SET	THEN	TO
TYPE	UNTIL	VAR	WHILE	WITH					

Специальные символы

.	+	-	*	/	=	< >	<	<=	>=	>	(	)
	(* | *)	/\	:=	.	,	;	:	^	..		

Предопределенные идентификаторы.

```
CONST   MAXINT=32767
TYPE    BOOLEAN=(FALSE,TRUE);
  "     CHAR (устанавливает ASCII символ);
  "     INTEGER = - MAXINT..MAXINT;
  "     REAL (предустанавливает дейтвительные числа)
тогда:
PROCEDURE   WRITE; WRITELN; READ; READLN; PAGE; HALT; USER; POKE; INLINE;
            OUT; NEW; MARK; RELEASE; TIN; TOUT;
FUNCTION    ABS; SQR; ODD; RANDOM; ORD; SUCC; PRED; INCH; EQLN; PEEK; CHR;
            SQRT; ENTIER; ROUND; TRUNC; FRAC; SIN; COS; TAN; ARCTAN; EXP;
            LN; ADDR; SIZE; INP.
```

Представление данных

Информация о методе хранения может понадобиться в большинстве случаев (может быть использована функция SIZE см. Рзд.2,3,6,7). Остальные детали необходимы тем, кто хочет использовать PASCAL-программу с программами в объектном коде

Целые числа.

При хранении целые числа занимают 2 байта , в двоичной комплементарной форме. Например:

```
1        =   #  0001
265      =   #  0100
-256     =   #  FF00
```

стандартный регистр Z-80 используемый компилятором для хранения целых HL.

Символы, логические и другие скаляры.

Занимают 1 байт в виде беззнаковых двоичных чисел.

Символы 8 бит в ASCII коде.

```
'E'     =     #45
```

Логические:

ORD(TRUE)=1 итак, TRUE представлена 1

ORD(FALSE)=0 FALSE представлена 0

стандартный регистр Z-80 используемый компилятором для хранения - A.

Действительные числа.

Используется стандартная интерпретация экспоненциальной формы только с основанием 2 вместо 10. Например:

2=2*10 или 1.0*2

Для хранения используется 4 байта:

```
-----------------------------------------------------------
| знак | нормализованная мантисса | экспонента | данные |
-----------------------------------------------------------
```

Пример:

```
2=     0 1000000 00000000 00000000 00000001
-12.5=1 1100100 00000000 00000000 00000011
```

Итак, регистровые пары HL DE используются для хранения действительных чисел. В памяти хранятся в последовательности ED LH.

Записи и массивы

Записи хранятся в таком же виде, как и их компоненты.

Массивы: если N = количество элементов в массиве и

S = размер каждого элемента, тогда

количество байт отведенное под массив N*S. Например:

ARRAY(1..10)OF INTEGER занимает 10*2=20 байт

ARRAY(2..12,1..10)OF CHARS имеет 11*10=110 элементов и требует 110 байт.

П.3.1.5 Множества.

Множества хранятся как битовые строки и, если базовый тип содержит N элементов, то количество байт используемых под это равно: (N-1) DIV 8+1. Например:

SET OF CHAR требует (256-1)DIV 8+1 =32 байт.

SET OF(BLUE,GREEN,ELLOW) требует (3-1) DIV 8+1=1 байт

Указатели.

Указатели занимают два байта, которые содержат адрес (в формате INTEL,т.е. младший байт 1) переменной на которую он указывает.

Хранение переменных во время выполнения программы

Есть три случая, в которых пользователю необходимо знать как хранятся переменные во время выполнения программы:

а.Глобальные переменные - объявляются в основном блоке программы;

б.Локальные переменные - объявляются во внутреннем блоке программы;

в.Параметры и возвращаемые величины-передаются в и из процедур и функций.

Эти индивидуальные функции описаны ниже и даны примеры использования этой информации в пр.4.

Глобальные переменные.

Глобальные переменные располагаются от вершины исполнительного стека вниз, т.е. если стек располагается с #B000 и основные переменные программы:

```
VAR     I :INTEGER;
        CH:CHAR;
        X :REAL;
```

тогда:

I (которая занимает 2 байта-см.Предыдущий рзд.) будет располагаться в позиции #B000-2 и #B000-1, т.е. #AFF9 и #AFFF.CH (1 байт) будет расположена #AFFE-1, т.е. #AFFD X (4байт) будет расположена #AFF9, #AFFA, #AFFB, #AFFC.

Локальные переменные.

Локальные переменные не могут быть так легко переданы через стек, поэтому вместо этого IX регистр указывает на начало каждого внутреннего блока так, что (IX-4) указывает на начало локальных переменных блока,т.е.

```
PROCEDURE          TEST;
VAR                I,J:INTEGER;
```

Тогда: I (целые 2 байта) будут расположены IX-4-2 и IX-4-1, т.е IX6 и IX-5.J будут расположены IX-8 и IX-7.

Параметры и возвращаемые величины.

Параметры - значения расматриваются как локальные переменные и также объявляются, как старший адрес их расположения в памяти. Однако, в отличие от переменных, наименьший (не старший) адрес фиксируется в (IX+2), например PROCEDURE TEST (I:REAL;J:INTEGER); тогда:

I по IX+4,IX+5,IX+6 и IX+7.

Параметры - переменные расматриваются также, как и параметры-значения за исключением того, что они всегда занимают 2 байта и эти 2 байта содержат адрес переменнрй; например :

PROCEDURE TEST (I:INTEGER;VARX:REAL);

Ссылка на X расположена по IX+2 и IX+3; эти позицини содержат адрес, где X хранится. Величина I находится по IX+4 и IX+5.

Значения, возвращаемые функциями располагаются над первым параметром памяти, т.е. FUNCTION TEST(I:INTEGER):REAL; тогда I находится по IX+2 и IX+3, а также резервируется пространство для возвращаемых величин по IX+4,IX+5,IX+6,IX+7.

Особенности применения HISOFT PASCAL на ZX-SPECTRUM.

Загрузка HP4TM с ленты.

PASCAL загружается LOAD"" и после удачной загрузки запускается автоматически.

Применение на SPECTRUM

ZX-SPECTRUM необычный компьютер и, естественно, это отразилось на использовании HISOFT RASCAL. Различные управляющие коды, описаные ранее, в ZX-SPECTRUM имеют следующие значения:

```
RETURN          ENTER
CC              CAPS SHIFT + 1
CH              DELETE или CAPS SHIFT + 0
CI              CAPS SHIFT + 8
CP              CAPS SHIFT + 3          - текст на принтер
CX              CAPS SHIFT + 5
CS              CAPS SHIFT + SPACE
```

Схема ввода с клавиатуры ZX-SPECTRUM необычная,т.к. содержит только алфавитно - цифровые коды. Используя SYMBOL SHIFT+ любую клавишу (кроме I) можно получить другие символы; SYMBOL SHIFT+T даст '>' , а SYMBOL SHIFT+G даст '<'.

Внимание!!!! Вы не должны использовать единые символы <=,<>,>=.Вместо них необходимо применять комбинации символов <,>,=.

Вы можете управлять некоторыми атрибутами различных позиций на экрана, используя стандартные управляющие коды (так WRITE (CHR(17), CHR(4))) сделает лист (PAPER) экрана зеленым. Но вы не можете изменить атрибуты окраски. Если во время использования управляющих кодов обнаружится недопустимый, будет высвечено сообщение 'SYSTEM CALL ERROR' и выполнение прекратится. Помните, что любой управляющий код интерпретируется PASCAL (так CHR(8) воспринимается как DELETE) и не могут быть прямо посланы в SPECTRUM. Используйте процедуру SPOUT, если вы хотите напечатать CHR код без интерпретации PASCAL.

Дополнительные команды добавлены в редактор PASCAL из ZX-SPECTRUM; это команды 'W' которая работает как 'P'-команда только она записывает блок текста на ленту в формате удобном для дальнейшего использования (через команду компилятора 'SF'). Помните, что вы не можете 'включить' текст, если он был записан на ленту используя 'P' команду. Если вы хотите включить текст используйте команду 'W'.

Оптимизирующий компилятор языка бейсик - BLAST

Начало работы

Бласт загружается так:

LOAD "BLAST"

Бласт самостартует и на экране появляется надпись:

BLAST OCSS 1985 xxxxx BYTES FREE.

(здесь xxxxx - об'ем свободной памяти)

В этом месте Бласт делает проверку на то, имеете ли вы право пользоваться этой программой.

ENTER THE COLOR IN SQUARE x-xx (W,Y,G,R)?

Для те.., кто пользуется оригинальной версией программы, с неудаленной защитой, имеется таблица:

Таблица кодов для входа в Бласт

	ABC	DEF	GHI	JK	LMN	OPQ	RST	UVW	XYZ		ABC	DEF	GHI	JK	LMN	OPQ	RST	UVW	XYZ
40	RYG	RYG	GRY	GW	GGY	RRY	GYG	YGW	YGG	20	RGY	YGY	RYG	RY	RYW	YRR	GYG	RYG	WYR
39	YGR	YWG	YGG	RG	YRG	YGG	RGR	GYG	RGR	19	YGR	GGW	YGY	GR	GWG	WGG	WGW	GGY	RRG
38	GYG	GYR	RGY	GY	RGY	WYR	YRG	RGR	GYG	18	RYG	RGY	RGR	RY	WYG	YRG	RYG	YGR	GRY
37	RGY	GRY	GGR	YW	YRG	YGW	GYG	YGY	GGR	17	YWG	YRG	RGY	GG	YGW	GYR	YGY	GYG	GYR
36	YRG	GYG	RYG	GG	RGY	GRG	RGW	GYG	GGY	16	GGG	RGR	YRG	RG	RGY	GGG	GGG	RGR	YRG
35	GGY	YGW	YEY	GR	YRG	RGY	GYR	YGW	YRY	15	YRG	YRY	WYG	RR	YRR	YRG	RYR	YRG	WYG
34	RYG	RRY	GYR	YG	YYG	YYG	YRG	GRY	GYG	14	GYR	RGG	YGG	RG	WGG	RGY	GWG	YGY	GGY
33	YRG	YGR	YGG	RG	GGR	GRY	GYG	YGW	YRG	13	GGY	GYG	YWY	GY	GWY	GRG	YGY	GYG	YRG
32	RGY	GYG	RYR	YW	GGY	RGW	YGY	RGY	RYG	12	RYR	GGR	GGR	YR	GWG	WYG	RGG	YRG	WGY
31	GYW	GRY	GRG	GR	YGR	YRG	YGR	YRG	GRG	11	YRY	WGY	GRY	WY	GGY	GRY	GYG	RYR	YGR
30	YRG	YGR	YRY	RG	GRY	GYR	YRY	WYG	YGW	10	RGG	TTG	RGR	YG	RYW	YGR	GGW	GWG	RGY
29	RYR	GWG	GGY	RY	WYG	RGG	RYR	GGW	GRY	9	GYG	RGR	YGG	GY	RGY	GGY	RYG	WYG	YYG
28	GRG	YGG	YRG	YG	YGY	GRR	YGY	RRY	RGG	8	RGW	YRY	GRY	RR	GRG	RYG	YGG	YGR	GRW
27	RYG	RYR	RYR	GG	GWG	RYG	RWR	GYR	GRY	7	YRG	GYW	GYG	GY	GYR	YYW	GYR	GRY	RYG
26	YRG	GYG	YRG	YR	YGR	YRG	RGG	YWY	GYR	6	GYR	YRG	GGR	GG	RRG	RGG	YRY	WYG	YRG
25	GYR	YRW	GYR	GG	RYR	GYR	GYR	GYG	RGY	5	YWG	GGY	YWG	YW	GGR	GYR	GYG	YGY	GYR
24	RGG	RGY	GRY	WG	RGG	RGW	YGG	RGY	GYR	4	GGY	RGR	GGY	RR	YRG	YWY	GRY	RGW	YRG
23	GGG	YRG	RYG	GY	YRY	GRG	GWY	GRR	YGY	3	RYR	GYG	RYR	YG	RGY	GGR	RGR	GYG	RGY
22	YWY	RYG	YGG	YG	RYR	GYR	YGR	RGY	WYG	2	YGG	GRG	YGG	RY	GRG	RYR	YRY	RGY	RYG
21	GGR	GRY	GYW	GR	YGY	GGY	RRY	GRR	GRY	1	WYR	YGY	WYG	YG	RYR	GWY	GYG	GRY	GGW

W - белый, Y - желтый, G - зеленый, R - красный. Введите соответствующую букву и нажмите ENTER. Чтобы выполнить проверку, надо 4 раза правильно ответить на запрос. Команды Бласта начинаются со звездочки (*), чтобы отличать их от команд бейсика.

Теперь загрузите бейсик-программу, не превышающую 3к и наберите *C, чтобы выполнить компиляцию.

Чтобы запустить откомпилированную программу, наберите: *R.

Запись откомпилированных программ: *S.

Компилирование больших программ.

Компилирование RAM-RAM возможно только для коротких программ. Чтобы обойти это неудобство, Бласт имеет возможность читать исходный файол с ленты или микродрайва и записывать об'ектный код на любое из этих устройств.

Выбор входного и выходного устройств.

Нажмите *I и ответьте на появившийся вопрос клавишей:

R - память RAM.

E - магнитофон (лента).

M - микродрайв.

Чтобы задать выходное устойство, нажмите *O.

Работа с лентой.

Вследствие ограниченной возможности ввода/вывода при работе с лентой, програм-

ма, подлежащая компиляции, должна быть сначала переписана на ленту в специальном формате. Средства для этого находятся в прилагаемой к бласту программе TOOLKIT.

После того, как исходная программа будет записана на ленту в нужном формате, Бласт может быть перезагружен и программа откомпилирована.

Компилирование с ленты на ленту.

В этом режиме используется две ленты. Одна с исходным файлом в специальном формате (см. Выше).

Когда вы нажмете *C для компиляции, Бласт попросит вставить кассету с исходным файлом и включить магнитофон. Через некоторое время прозвучит сигнал, и вам будет предложено поменять ленту. Необходимо остановить ленту с исходным файлом в течение 5-ти секунд после звукового сигнала. Если вы этого не сделаете, то есть вероятность, что какие-либо данные будут потеряны.

Пи-код и машинный код.

Тип кода, который должен быть сформирован, задается с помощью директив:

REM! P-CODE - генерирует пи-коды (а также по умолчанию).

REM! MASHINE-CODE - генерирует машинные коды.

Обработка машинного кода.

1. Откомпилированная программа может резервировать пространство для блока в машинных кодах путем понижения RAMTOP обычным путем.

2. Переменные бейсика храняться в Бласте таким же образом, что и в бейсике, поэтому блок в машинных кодах, имеющий к ним обращение, будет работать и в Бласте.

3. Процедуры в машинных кодах, расширяющие бейсик перехватом процедур обработки ошибок (или какими-либо другими методами) по- прежнему будут работать.

Выполняется это следующим образом. Если во время компиляции "Бласт" встречает выражение, которое выглядит синтаксически неверным, компилятор компилирует его в об'ектный файл и приписывает ему выходной код. Когда во время работы программы процедуры RTS встречают этот выходной код, они вызывают бейсик-интерпритатор для его обработки. Если выражение содержит ошибку, интерпритатор сообщает об этом и выходит из нее обычным путем. Если же это не ошибка, а предусмотренное расширение бейсика, интерпритатор ведет себя так, будто текст и не компилирован.

Расширение бейсика, предусмотренное Бластом, также как и директивы компилятора, вводятся в форме специальных инструкций REM, которые распознаются Бластом во время компиляции.

Если Бласт встречает выражение REM, начинающееся с символа "%", он вырабатывает код, который вызывает передачу выражений REM с опущенным знаком "%" в интерпритатор во время рабочего запуска. Бласт сообщает об этом :

COMMENT TRANSFERID AT LINE xxxx

Использование целых переменных

Об'явление типа переменной выполняется следующей директивой, например:

REM ! INT I, J, K, A(10,5)

здесь переменные I,J,K, и весь массив A, т.е. DIM A(10,5) уже давать не надо. Об'явление типа должно быть до употребления переменной.

Защита откомпилированных программ

1. Автозапуск.

Если в начале бейсик-программы включить строку

REM ! AUTORUN, Бласт вызовет автоматическое выполнение файла после загрузки.

2. Защищенный пи-код.

Пи-код, который генерируется Бластом, это недокументированный язык и поэтому предоставляет более высокий уровень его защищенности, чем машинный код. Поэтому рекомендуем защищающие процедуры писать на бейсике, и компилировать их в пи-код.

Копирование программ, прошедших Бласт

Откомпилированные программы не могут быть выгружены напрямую через SAVE. Команда S доступна для программ, которые были откомпилированны из RAM в RAM, но S не выгрузит откомпилированную программу, если она загружалась с ленты или с

микродрайва. Если вы хотите выгрузить программу на одно из этих устройств, продолжайте так:

Запись на ленту:

1.Загрузите откомпилированную программу.

2.Наставьте следующие строки:

```
15 LOAD "<PROG>"
20 RANDOMISE USR PEEK 23635+ 256*PEEK 23636 + 150
```

3.Наберите:

```
SAVE "<PROG>" LINE 15
```

Запись на микродрайв

Процедура записи на микродрайв точно такая же, за исключением того, что должны присутствовать обычные параметры микродрайва. Например:

```
15 LOAD* "M";1;"<PROG>"
20 RTANDOMIZE USR PEEK 23635+256*PEEK 23636 + 150
```

и наберите:

```
SAVE*"M";1;"<PROG>"LINE15
```

Директивы компиллятора.

Бласт предоставляет определенные возможности выбора при компиляции. Они реализуются посредством директив компиллятора. Выполняется это специальными выражениями REM в форме:

REM! <Директива>

Существуют также две дополнительные формы инструкций REM.

REM% обеспечивает передачу текста или комментария интерпритатору во время работы программ.

REM& применяется для открывания дополнительных инструкций бейсика, предусмотренных Бластом.

1. REM! PCODE - по этой директиве бласт компиллирует программу в пи-коды.
2. REM! MACHINE CODE - генерация в машинные коды.
3. REM! INT I,J,K - об'являет переменные I,J,K как целые.
4. REM! AUTORUN - обеспечивает автоматический запуск откомпиллированной программы после загрузки.

Список дополнений к бейсику:

1. Включение и выключение клавиши BREAK:

```
REM& BREAK ON
REM& BREAK OFF  ( по умолчанию кл. выключена)
```

2. WHILE...WEND:

форма:

```
REM& WHILE <условие>
REM& WEND
```

По этим командам блок операторов, замыкаемый REM& WEND выполняется многократно, до тех пор, пока <условие> истинно (не нуль). Если <услозие> ложно, операторы пропускаются.

3. REPEAT...UNTIL

форма:

```
REM& REPEAT
REM& UNTIL <условие>
```

Блок операторов между REM& REPEAT и REM& UNTIL выполняется многократно, пока <условие> имеет ненулевое значение.

4. DOKE

форма:

```
REM& DOKE <NE>,<NE>
```

Здесь "NE"-числовое выражение. Все это 16-битная инструкция POKE. Результат второго выражения помещается в адрес, на который указывает первое выражение.

5. DEEK - 16-битная инструкция PEEK. Форма записи аналогична предыдущей.

6. CALL

Форма:

```
REM& CALL <NE>\<список параметров>\
```

Эта команда вызывает процедуру в машинных кодах, содержащуюся по адресу, задаваемому выражением "NE". Параметры разделяются запятыми; они могут быть либо числовыми переменными в диапазоне от 0 до 65535 или адресами числовых переменных, тогда они записываются так:

& <имя переменной>

Эти параметры хранятся в таком порядке, что первый из них находится по адресу, на который указывает регистр IX.

7. ELSE форма: REM& ELSE <список инструкций>.
Это дополнительное разрешение расширение к конструкции IF. ..ELSE.

Программа TOOLKIT

К Бласту прилагается довольно обширный набор средств для облегчения разработки программ (TOOLKIT).

TOOLKIT автостартует после загрузки и выдает на экран сообщение:

BLAST TOOLKIT OCSS 1985

Как и компилятор Бласт, TOOLKIT загружается в верхнюю часть RAM и устанавливает RAMTOP ниже себя. При этом об'ем доступной памяти снижается примерно на 2к.

TOOLKIT не может находиться в памяти компьютера одновременно с Бластом.

Ниже приведен список команд TOOLKITA. Они начинаются со знака (*), обозначаются одной буквой, за ней идут параметры.

N1 - N2 - обозначает: от строки N1 до строки N2.
N1 - - обозначает: от строки N1 до конца программы.
- N2 - обозначает: от начала программы до строки N2.

Если номера строк не даны, предполагается, что речь идет о всей программе.

Команды обработки строк

1. Редактировать строку N1. - *EN1.
2. Копировать строку N1 в N2 - *C N1,N2.
3. Убрать строку N1: - *D N1.
4. Перенос строки N1 в N2 (строка N1 уничтожается)-*M N1,N2.

Команды обработки блоков

1. Копирование: *C<диапазон>,N. По этой команде копируются все строки из заданного диапазона в строки, начиная с N.
2. Стирание: *D<диапазон>
3. Перенос строк: *M<диапазон>,N.
4. Перенумерация: *R<диапазон>,N1,N2 строки из заданного интервала перенумеровываются, им присваиваются номера от N1 с шагом через N2 . По умолчанию N2 равно 10.

Команды обработки стрингов

1. Поиск: *F<диапазон>,N$ по этой команде в заданном диапазоне строк разыскивается заданный стринг. По умолчанию программа будет искать последний введенный стринг.
2. Поиск с заменой: *S <диапазон> N1$, N2$ отыскивается первый стринг и заменяется на второй.

Прочие команды

1. TRACE: *TN по этой команде на экране печатается номер выполняемой в данный момент инструкции, начиная со строки N. Замедлить вычисления можно клавишей пробел, а остановить клавишей ENTER.
2. Устранение всех строк REM, кроме тех, которые начинаются с "!", "%", "&".
   *K
3. Запись на магнитофон блока программы.
   *W <диапазон>,<имя файла>
4. Запись на ленту программы в формате, удобном для компиляции Бластом с ленты. Программа будет выгружена блоками, совместно с информацией, необходимой для компиляции.
   *B <имя файла>
5. Конец работы : *Q

Карта памяти Бласта

Карта памяти во время рабочего прогона программы

```
RAMTOP ---------------------------------
       стеки GO SUB, машинный стек
       свободная область RAM
STKEND ---------------------------------
       стек калькулятора
STKBOT ---------------------------------
       рабочее пространство
WORKSP ---------------------------------
       область редактирования
ELINE  ---------------------------------
       переменные бейсика
VARS   ---------------------------------
       программа на бейсике
PROG   ---------------------------------
       информация о каналах
CHANS  ---------------------------------
       карты микродрайва
       системные переменные интерфейса-1
       системные переменные
       буфер принтера
       атрибуты
       экранная память
       ROM
```

```
       стеки GO SUB, машинный стек
       свободная область RAM
STKEND ---------------------------------
STKBOT ---------------------------------
       рабочее пространство
WORKSP ---------------------------------
       область редактирования
ELINE  ---------------------------------
       переменные
VARS   ---------------------------------
       откомпилированная программа
PROG   ---------------------------------
       информация о каналах
CHANS  ---------------------------------
       карты микродрайва
       рабочие процедуры
       системные переменные интерфейса-1
       системные переменные
       буфер принтера
       атрибуты
       экранная память
       ROM
```

Р е д а к т о р " T H E L A S T W O R D 2 "

О б щ е е о п и с а н и е

TLW – это тектовый редактор для компьютеров "спектрум" и "спектрум плюс".

Экран содержит 256 точек в ширину и 192 в высоту. Генератор символов позволяет печатать символы шириной 6, 5, 4, или 3 точки, что дает 42, 51, 64 или 85 символов в строке.

Операционная система редактора выделяет верхние три строки экрана для заголовка ("хэдера"), где отображается 18 единиц информации о текущем состоянии программы, текстового файла и положении курсора.

Р а з б и в к а т е к с т а

- нажмите обе клавиши SHIFT, в хэдере изменится режим WRITE на режим COMMAND;
- нажав CAPS SHIFT, нажмите "V" и в хэдере появится слово "VIDEO";
- теперь вы можете набрать 48, 60 или 80 для установки количества символов в строке вместо первоначальных 40;
- в конце нажмите ENTER и ваш текст будет переписан в соответствии с выбранным размером строки.

Клавиша ENTER – конец абзаца. Внутри файла–переход на следующую строку.

К о м а н д ы	перемещения курсора
– Вправо на один символ	– CAPS SHIFT "8"
– Влево на один символ	– CAPS SHIFT "5"
– Вверх на одну строку	– CAPS SHIFT "7"
– Вниз на одну стоку	– CAPS SHIFT "6"
– Табуляция вправо	– CAPS SHIFT "I" курсор смещается на величину TAB или до левого поля, если оно ближе.
– Вправо на одно слово	– CAPS SHIFT "4"
– Влево на одно слово	– CAPS SHIFT "3"
– Вправо на одно предложение	– CAPS SHIFT "W". Курсор сдвигается до ближайшей точки, а затем устанавливается около следующей позиции.
– Вправо до конца строки	– режим E CAPS SHIFT "8"
– Влево к началу строки	– режим E CAPS SHIFT "5"
– К началу первой строки	– режим E CAPS SHIFT "7"
– К началу последней строки	– режим E CAPS SHIFT "6"
– Вниз к началу первой строки следующей страницы	– режим E CAPS SHIFT "D".

Экран перепечатывается так, что последняя строка данной страницы устанавли-

вается первой строкой следующей.
- Вверх к началу предыдущей страницы - режим E "U"
- К началу текстового файла - режим E "A"
- К концу текстового файла - режим E "Z"
- К заданной строке - режим E "N". Запрашивается и вводится номер требуемой строки.
- Экран вверх на одну строку - режим E CAPS SHIFT "3"
- Экран вниз на одну строку - режим E CAPS SHIFT "4"

Команды обработки текста

- Стирание символа - CAPS SHIFT "0".
- Стереть текст до конца строки - CAPS SHIFT "9".
- Стереть текст до конца файла - режим E CAPS SHIFT "9".
- Стереть весь текст - режим E CAPS SHIFT "Z".
- Переключатель INSERT/OWERWRITE - (вставить/переписать) - режим E "I".

В режиме OWERWRITE, когда вводится новый текст, курсор находится на верхней границе файла и файл постепенно расширяется. Если курсор находится внутри файла, идет его "переписывание".
- Сдвинуть весь текст влево - CAPS SHIFT "Q".
- Сдвинуть текст вправо - CAPS SHIFT "E".
- Централизовать текст в строке - режим E "H"
- Выровнять (JUSTIFY) строку - режим E "J".
- Ограничить пробелы в строке - режим E CAPS SHIFT "L".
- Переоформить (REFORM) текст от курсора до "конца абзаца"
- режим E CAPS SHIFT "R".
- Вставить оператор управления печатью - CAPS SHIFT "I".

1. Число от 1 до 24, связанное с соответствующим кодом принтера, предназначенным для управления принтером во время печати.

2. Число от 90 до 99, указывающее на строку в бейсике, к которой надо сделать переход (строка определяется как 100, умноженное на это число, т. е. от 9000 до 9900).

3. Буква от A до Z со знаком $, когда требуется напечатать избранный бейсиковский стринг.
- Ввод символа "копирайт" - режим E CAPS SHIFT "K".Синклеровский знак "копирайт", CHR$127, вводится по этой команде в файл.
- Повтор куска текста - режим E "R".
- Удаление куска текста - режим E "k".
- Поиск или изменение некоторого блока текста - режим E "X".

Структурные команды и утилиты

- Переключатель "JUSTIFY/RAGGED" - режим E "J".
- Переключатель W.WRAP/W.SPLIT - (округление слов/перенос слов) - режим E "W".
- Установка регистра букв на одну строку - режим E "2".
- Включение в текст бейсиковского стринга - режим E CAPS SHIFT "4".
- Включение/выключение контурной линии - режим E "o".
- Включение/выключение видеомаркеров - режим E "V".
- Переключатель регистров вр/нр - CAPS SHIFT "2".
- Установка величины правого поля - режим E CAPS SHIFT "M".
- Установка величины разбивки экрана - режим E CAPS SHIFT "V".
- Установка величины табуляции - режим E CAPS SHIFT "T".
- Изменение цвета бордюра - режим E CAPS SHIFT "B".
- Изменение цвета хэдера - режим E CAPS SHIFT "H".
- Счетчик слов - режим E CAPS SHIFT "W".
- Работа калькулятора - режим E CAPS SHIFT "C".

При написании математических выражений клавиатура модифицируется т.о.:

Функция	клавиша	SHIFT	Функция	клавиша	SHIFT	Функция	клавиша	SHIFT
+	K	SYMB	ACS	C	CAPS	PI	M	CAPS
-	J	SYMB	ATN	T	CAPS	<	R	SYMB
*	B	SYMB	exp	X	CAPS	=	L	SYMB
/	V	SYMB	SQR	H	CAPS	<>	W	SYMB
^	H	SYMB	INT	R	CAPS	>	T	SYMB

SIN	Q	CAPS		LN	Z	CAPS		> =	E	SYMB
COS	W	CAPS		VAL	J	CAPS		OR	U	CAPS
TAN	E	CAPS		SGN	F	CAPS		AND	Y	CAPS
ASN	S	CAPS		ABS	G	CAPS		NOT	S	CAPS

- Установка времени - режим E CAPS SHIFT "X".
- Отбой введенной команды - CAPS SHIFT "Q".
- Возврат к бейсику - режим E CAPS SHIFT "B".

К о м а н д ы п е ч а т и

- Управление печатью - режим e "G".
- Печать текста - режим E "P". Загорается меню, на котором изображены:
  - номера первой и последней строк файла;
  - установленное количество строк файла;
  - состояние операторов управления печатью.
- Оператор BREAK переводит в бейсик.
- Файл PCT (файл операторов управления принтером) - режим E "T".

К о м а н д ы х р а н е н и я и в ы з о в а ф а й л о в

- Записать текст на внешний носитель - режим E "S".
- Загрузить текст - режим E "L".

После команды LOAD вводится номер носителя 0 - для ленты и от 1 до 8 для микродрайва и имя, которое нужно отыскать.

- Раскаталогизировать картридж - режим E "C".
- Стереть файл на картридже - режим E "E".
- Выполнить форматирование картриджа - режим E "F".

О п е р а т о р ы н е п о с р е д с т в е н н о г о у п р а в л е н и я п р и н т е р о м

Используя команду управления печатью, можно ввести стандартный префиксный байт, который будет выдываться перед группой байтов PCT и выполнять роль инициатора для интерфейса или принтера.

N	метка	упр. коды
1	FRM.F	1 12 - -
2	ELIT+	2 27 77 -
3	PICA+	2 27 80 -
4	ENLG+	3 27 87 49
5	ENLG-	3 27 87 48
6	COND+	1 15 - -
7	COND-	1 18 - -
8	UNDL+	3 27 45 49

N	метка	упр. коды
9	UNDL-	3 27 45 48
10	емрн+	2 27 69 -
11	емрн-	2 27 70 -
12	D.ST.+	2 27 71 -
13	D.ST.-	2 27 72 -
14	SUB+	2 27 83 48
15	SUB-	3 27 83 49
16	SP/в-	2 27 84 -

N	метка	упр. коды
17	ITAL+	2 27 52 -
18	ITAL-	2 27 53 -
19	BELL!	1 7 - -
20	PPR-	2 27 56 -
21	PPR+	2 27 57 -
22	INITL	2 27 64 -
23	a4/70	3 27 67 70
24	RRF	3 27 78 9

стандартный префикс - 1

* Интерфейс ZX LPRINT III. Дополнительных кодов не надо.
* Интерфейс типа TASPRINT. Необходим собственный кодовый блок. Кодовый блок "TASBUFF" загружается через бейсик до редактора. Адрес вызова 23296.
* Интерфейс типа KEMPSTON. После загрузки программы возврат в бейсик и два ввода: POKE 65176,245: POKE 65177,1
* Интерфейс типа MOREX. Вам надо написать простую бейсик-программу и запустить ее после загрузки редактора. Например, такую:

```
10 DATA 245,219,251,230,1,32,250,241
20 DATA 211,251,62,1,211,127,175,211,127,201
30 FOR N=0 TO 17: READ A: POKE 65176+N,A
40 NEXT N: STOP
```

С в о д к а с и с т е м н ы х п е р е м е н н ы х

Блок в машинных кодах	от 50000 до 65535
Адрес вызова	52410
Адрес возврата при печати	65154
Режим работы процессора	с прерываниями 2-го типа
Размер текстового файла	от 0 до 24к, начиная от RAMTOP+2
Файл операторов PCT	от 52193 до 52408

формат: 24 оператора, каждый из них включает в себя:
байты от 1 до 5 – коды ASCII
байты от 6 до 9 – коды от 0 до 254

Стандартный префикс – от 0 до 254	52409, 1 байт
Операторы управления печатью 1...24	значения от 128 до 151
Операторы управления печатью 90 – 99	значения от 152 до 161
Операторы управления печатью A – Z	значения от 230 до 255
Блок кодов интерфейса принтера (под интерфейс кемпстон – E)	от 64954 до 64977
Память калькулятора	10 переменных от м0 до м9
Переменные калькулятора	как в бейсике
Длина включаемых стрингов	до 4к
Имена включаемых стрингов	от A$ до Z$

Некоторые системные переменные

Адрес очистки текстового файла	64978
Установка таймера	65189
Вершина пространства текстового файла	65193, 2 байта
Начало текстового файла	65195, 2 байта
Список транслируемых символов экрана	от 65157 до 65165
Список транслируемых символов принтера	от 65173 до 65181
Код "ВК"	65241
Код "ПС"	65242
Строка бейсика для выхода	65244, 2 байта
Величина разбивки экрана	65254
Адрес вызова принтера	65257
Последний байт текстового файла	65268
Адрес курсора в файле	65270
Колонка курсора в тексте	65272
Строка курсора в тексте	65272, 2 байта
Левое поле	65281
Правое поле	65282
Интервал между строк	65285
Количество печатных копий	65286
Установка величины табуляции	65289

Введение русского шрифта в редактор

Если вы хотите, чтобы русский шрифт начинался с адреса 30000, то вам надо заслать числа:

POKE 63661, 48: POKE 63662,117

Обратите внимание, что 117*256+48 = 30000.

При возврате же на латинский шрифт нада естественно опять восстановить исходные значения.

Соответствие между латинскими и русскими символами

код	ASCII (КОИ-7/Н0)	кои-7/Н1	код	ASCII (КОИ-7/Н0)	кои-7/Н1	код	ASCII (КОИ-7/Н0)	кои-7/Н1
32		пробел	64		Ю	96	фунт стерл	ю
33	!	!	65	A	А	97	a	а
34	"	"	66	B	Б	98	b	б
35	#	#	67	C	Ц	99	c	ц
36	$	$	68	D	Д	100	d	д
37	%	%	69	E	Е	101	e	е
38	&	&	70	F	Ф	102	f	ф
39			71	G	Г	103	g	г
40	(	(	72	H	Х	104	h	х
41	)	)	73	I	И	105	i	и
42	*	*	74	J	Й	106	j	й
43	+	+	75	K	К	107	k	к
44	,	,	76	L	Л	108	l	л

45	-	-	77	M	М	109	m	м
46	.	.	78	N	Н	110	n	н
47	/	/	79	O	О	111	o	о
48	0	0	80	P	П	112	p	п
49	1	1	81	Q	Я	113	q	я
50	2	2	82	R	Р	114	r	р
51	3	3	83	S	С	115	s	с
52	4	4	84	T	Т	116	t	т
53	5	5	85	U	У	117	u	у
54	6	6	86	V	Ж	118	v	ж
55	7	7	87	W	В	119	w	в
56	8	8	88	X	Ь	120	x	ь
57	9	9	89	Y	Ы	121	y	м
58	:	:	90	Z	З	122	z	з
59	;	;	91	[	Ш	123	{	ш
60	<	<	92	\	Э	124	|	э
61	=	=	93	]	Щ	125	}	щ
62	>	>	94	^	Ч	126	~	ч
63	?	?	95	_	'	127		

В соответствии с этой таблицей ваша задача при создании русского шрифта - обеспечить, чтобы, например, 81-м шаблоном вместо шаблона буквы "Q" стоял шаблон буквы "Я", а вместо "Q" - "Я" и т.д.

К о н с т р у к ц и я б у к в р у с с к о г о а л ф а в и т а

Запустите (RUN) нижеприведенную программу. Когда она отработает, выгрузите сформированный шрифт на ленту. Этот блок кодов расположен, начиная с адреса 30000 (запомните этот адрес) и имеет длину 768 байт (96 символов по 8 байт).

```
10  FOR I = 1 TO 768
20  READ A
30  POKE (29999+I),A
40  NEXT I
50  SAVE "RUS" CODE 30000,768
100 DATA 0,0,0,0,0,0,0,0
101 DATA 0,16,16,16,16,0,16,0
102 DATA 0,36,36,0,0,0,0,0
103 DATA 0,36,126,36,36,126,36,0
104 DATA 0,8,62,40,62,10,62,8
105 DATA 0,98,100,8,16,38,70,0
106 DATA 0,16,40,16,42,68,58,0
107 DATA 0,8,16,0,0,0,0,0
108 DATA 0,4,8,8,8,8,4,0
109 DATA 0,32,16,16,16,16,32,0
110 DATA 0,0,20,8,62,8,20,0
111 DATA 0,0,8,8,62,8,8,0
112 DATA 0,0,0,0,0,8,8,16
113 DATA 0,0,0,0,62,0,0,0
114 DATA 0,0,0,0,0,24,24,0
115 DATA 0,0,2,4,8,16,32,0
116 DATA 0,60,70,74,82,98,60,0
117 DATA 0,24,40,8,8,8,62,0
118 DATA 0,60,66,2,60,64,126,0
119 DATA 0,60,66,12,2,66,60,0
120 DATA 0,8,24,40,72,126,8,0
121 DATA 0,126,64,124,2,66,60,0
122 DATA 0,60,64,124,66,66,60,0
123 DATA 0,126,2,4,8,16,16,0
124 DATA 0,60,66,60,66,66,60,0
125 DATA 0,60,66,66,62,2,60,0
126 DATA 0,0,0,16,0,0,16,0
127 DATA 0,0,16,0,0,16,16,32
128 DATA 0,0,4,8,16,8,4,0
146 DATA 0,0,72,72,120,72,72,0
147 DATA 0,0,48,72,72,72,48,0
148 DATA 0,0,120,72,72,72,72,0
149 DATA 0,0,56,72,72,56,72,0
150 DATA 0,0,112,72,72,112,64,0
151 DATA 0,0,48,72,64,72,48,0
152 DATA 0,0,124,16,16,16,16,0
153 DATA 0,0,72,72,120,8,120,0
154 DATA 0,0,84,56,16,56,84,0
155 DATA 0,0,120,72,120,72,120,0
156 DATA 0,0,64,112,72,72,112,0
157 DATA 0,0,68,116,76,76,116,0
158 DATA 0,0,120,8,56,8,120,0
159 DATA 0,0,84,84,84,84,124,0
160 DATA 0,0,48,72,24,72,48,0
161 DATA 0,0,84,84,84,84,124,4
162 DATA 0,0,72,72,56,8,8,0
163 DATA 0,0,96,56,40,40,56,0
164 DATA 0,92,84,116,84,84,92,0
165 DATA 0,56,72,72,120,72,72,0
166 DATA 0,120,64,120,72,72,120,0
167 DATA 0,72,72,72,72,72,120,8
168 DATA 0,56,72,72,72,72,120,72
169 DATA 0,120,64,120,64,64,120,0
170 DATA 16,124,84,84,84,124,16,0
171 DATA 0,120,64,64,64,64,64,0
172 DATA 0,72,48,48,48,48,72,0
173 DATA 0,72,72,88,104,72,72,0
174 DATA 16,72,72,88,104,72,72,0
175 DATA 0,36,40,48,48,40,36,0
176 DATA 0,56,72,72,72,72,72,0
177 DATA 0,68,108,84,68,68,68,0
178 DATA 0,72,72,120,72,72,72,0
179 DATA 0,48,72,72,72,72,48,0
```

```
129 DATA 0,0,0,62,0,62,0,0
130 DATA 0,0,16,8,4,8,16,0
131 DATA 0,60,66,4,8,0,8,0
132 DATA 0,0,72,84,116,84,72,0
133 DATA 0,0,56,72,72,120,72,0
134 DATA 0,48,64,112,72,72,120,0
135 DATA 0,0,72,72,72,72,120,8
136 DATA 0,48,8,120,72,72,120,0
137 DATA 0,0,48,72,120,64,56,0
138 DATA 0,0,56,84,84,56,16,0
139 DATA 0,0,120,64,64,64,64,0
140 DATA 0,0,72,48,48,48,72,0
141 DATA 0,0,72,88,104,72,72,0
142 DATA 0,16,72,88,104,72,72,0
143 DATA 0,0,72,80,96,80,72,0
144 DATA 0,0,56,72,72,72,72,0
145 DATA 0,0,68,108,84,84,68,0
```

```
180 DATA 0,120,72,72,72,72,72,0
181 DATA 0,56,72,72,120,40,72,0
182 DATA 0,120,72,72,120,64,64,0
183 DATA 0,120,64,64,64,64,120,0
184 DATA 0,124,16,16,16,16,16,0
185 DATA 0,72,72,72,120,8,56,0
186 DATA 0,84,84,84,56,84,84,0
187 DATA 0,120,72,120,72,72,120,0
188 DATA 0,64,64,120,72,72,120,0
189 DATA 0,68,68,116,76,76,116,0
190 DATA 0,120,8,8,56,8,120,0
191 DATA 0,84,84,84,84,84,124,0
192 DATA 0,48,72,24,8,72,48,0
193 DATA 0,84,84,84,84,84,124,4
194 DATA 0,72,72,72,120,8,8,0
195 DATA 60,66,153,161,161,153,66,60
```

Порядок переделки программы на русский шрифт

1. Загрузите фирменную программу TLW2.
2. На вопрос о том, подключен ли принтер, ответьте "N".
3. Перейдите в режим 40 знаков в строке (режим E +"V"). На экране появится запрос VIDEO: дайте в ответ число 40. Выйдите в бейсик (режим E + "B"). Теперь можно вносить изменения в программу.
4. Чтобы обеспечить место для изменений дайте прямую команду CLEAR 31000. Теперь в строке 30 поднимите адрес RAMTOP, задаваемый оператором CLEAR до 31000.

Далее в строке 30 поместите команду на загрузку русского шрифта, сформированного ранее.

Строка 30 примет вид (изменения подчеркнуты):

```
30 CLEAR VAL "31000": GO SUB VAL 100: PRINT '': LOAD "TLW2" CODE
                -----
   LOAD "RUS"CODE 30000,768: GO TO VAL 1000
   ------------------------
```

5. Строки 40,70,80, предназначенные для работы с микродрайвом, можете удалить, если у вас его нет.
6. Строка 60 предназначена для выгрузки настроенной программы на ленту. Внесите в нее изменения с тем, чтобы выгружался и встроенный русский шрифт.

```
60 SAVE "TLW2"CODE VAL "50000,VAL"15535": SAVE "RUS" CODE 30000,768:
   GO TO VAL"90"
```

7. В строке 3000, которая инициализирует программу, введите два новых параметра L и R. Это номера строк, в которых начинается переделка шрифта:

```
L = 7000 - русский шрифт.
R = 8000 - латинский шрифт.
3000 LET R=7000:LET L=8000:RANDOMIZE USR VAL "52410"
     ---------------------
```

8. Введите новые семитысячные строки, обеспечивающие печать русскими буквами:

```
7000 REM                                  первый вектор
7010 REM *** RUSSIAN LETTERS ***          7500 DATA 131,7,7,7,7,7
7020 REM                                  7510 DATA 131,7,7,7,7,7,7
ввод 1-го вектора для вырезания           7520 DATA 131,7,7,7,7,7,7
столбцов.                                 7530 DATA 131,7,131,7,7,131,7
7030 RESTORE 7500                         7540 DATA 131,7,131,7,7,131
7040 FOR I=63133 to 63195                 7550 DATA 7,7,7,7,7,131
7050 READ A                               7560 DATA 7,7,7,7,131,7,131
7060 POKE I,A                             7570 DATA 7,7,7,131,7,7,131
7070 NEXT I                               7580 DATA 7,131,7,7,131,7,131
ввод 2-го вектора для вырезания строк     7590 DATA 7,131,7
7080 RESTORE 7600                         второй вектор
7090 FOR I=63231 TO 63293                 7600 DATA 171,167,167,167,167,167,171
7100 READ A                               7610 DATA 167,167,167,167,167,167,171
```

```
7110 POKE I,A
7120 NEXT I
подключение русского шрифта
7130 POKE 63661,48:POKE 63662,117
отключение процедур изображения
символов "A","Y", "D"
7140 POKE 63669,250:POKE 63670,250:
     POKE 63683,250
```

```
7620 DATA 167,167,167,167,167,167,199
7630 DATA 167,171,167,167,179,167,171
7640 DATA 167,171,167,151,155,167,167
7650 DATA 167,167,167,171,167,167,151
7660 DATA 151,211,167,171,167,167,167
7670 DATA 167,167,167,199,167,171,167
7680 DATA 167,169,167,171,167,171,167
7999 GO TO 3000
```

9. Введите новые восьмитысячные строки, обеспечивающие печать латинскими буквами.

```
8000 REM
8010 REM *** ENGLISH LETTERS ***
8020 REM
ввод 1-го вектора
8030 RESTORE 8500
8040 FOR I=63133 TO 63195
8050 READ A
8060 POKE I,A
8070 NEXT I
ввод 2-го вектора
8080 RESTORE 8600
8090 FOR I=63231 TO 63293
8100 READ A
8110 POKE I,A
8120 NEXT I
поключение латинского шрифта
8130 POKE 63661,0:POKE 63662,61
подключение процедур печати
символов "A","D","Y"
8140 POKE 63669,97:POKE 63676,68:
     POKE 63683,89
```

```
первый вектор
8500 DATA 137,13,13,13,25,13,13
8510 DATA 21,13,82,13,49,7,137
8520 DATA 13,13,13,37,25,13,35
8530 DATA 13,137,137,137,137,67,112
8540 DATA 131,7,131,7,73,25,69
8550 DATA 67,11,11,35,11,11,67
8560 DATA 67,67,35,131,11,11,11
8570 DATA 25,67,11,67,11,131,131
8580 DATA 131,11,67,82,49,70
второй вектор
8600 DATA 185,185,185,185,185,185,185
8610 DATA 185,185,229,185,179,185,185
8620 DATA 157,185,185,173,185,185,199
8630 DATA 185,181,181,181,173,211,241
8640 DATA 199,143,199,248,211,157,217
8650 DATA 211,179,179,227,179,179,199
8660 DATA 211,203,227,227,179,179,179
8670 DATA 179,203,179,199,179,171,171
8680 DATA 179,179,199,229,227,203,227
```

10. Загрузите заранее подготовленный 768-байтный блок кодов, содержащий русский шрифт.

```
LOAD ""CODE 30000,768
```

11. Теперь вы можете выгрузить созданную копию редактора на ленту командой GO TO 50.

12. Для проверки войдите в редактор командой GO TO 3000.

13. Далее выполняйте выход из редактора в бейсик через (режим E +"B"), а вход:

GO TO R - в русский шрифт;
GO TO L - в латинский шрифт;

M A S T E R F I L E - 09

меню , запросы , режимы

Мастерфайл почти во всех случаях управляется от меню или по запросу. После загрузки высвечивается главное меню MM (MAIN MENU). Кроме того изображается имя файла (до 10 символов) и номер версии программы. После выбора того или иного пункта меню нажатием соответствующей клавиши, появляется новое меню или запрос. Запрос - это инструкция, которая высвечивается в нижней части экрана и имеет преимущество перед другими видами меню. Обычно меню или запросы требуют ответа в виде нажатия одной клавиши, но когда необходим текстовый ответ, то загорается мигающий курсор L.

В режиме "дисплей" меню может забить информацию на экране, поэтому обычно оно не изображается, а накладывается после нажатия " ".

Командный режим (COMMAND MODE)

За редким исключением мастерфайл работает в машинном коде, но вы можете перейти в бейсик-область через командный режим. Это возможно при курсоре L, т.е.

нажмите в главном меню -L. Если теперь нажать кл. "6", то вы увидите резолюцию H STOP IN INPUT- это командный режим, теперь вы можете работать в бейсике.
Для возврата в Мастерфайл - GO TO 1; так же выход из ошибок, допущенных в командном режиме.

Главное меню (MAIN MENU)

A - добавление новой записи к концу файла.
C - печатать список меток сообщений вместе с заголовками. Переход в режим "дисплей"- нажатием избранной метки. Возврат в MM - ENTER.
D - переключение в режим "дисплей" для просмотра записей. Формат сообщения при этом используется либо самый первый, либо последний применявшийся. Для изменения формата надо перейти в режим редактирования (EDIT). Переход по опции D возможен и из других меню тоже.
E - переключение в режим редактирования (EDIT) для изменения форматов.
L - загрузить файл (с ленты).
N - просмотр или изменение имен.
S - переключение в режим " поиск " (SEARCH). Предназначается для отыскания записей по их содержимому.
I - инверсия. Изменение статуса выборки записей. Выбранные записи (SELECT) станут не выбранными и наоборот.
R - RESET. Переустанавливает статус SELECT всех записей. Все записи становятся не выбранными, что индицируется SEL=00 00.Чтобы сделать все записи выбранными, сделайте R,а затем I.
P - стирание всех выбранных записей.
T - подсчет полного и среднего объема по выбранным записям. В "T" можно войти и из режима " дисплей ".
V - запись на ленту программы и файла или только файла.
U - произвести расчет выбранных записей в бейсике.

Создание пустого файла

а) стереть все записи: MM R I P Y
б) стереть все форматы: MM E R I X Y R 2 X Y и т. д.
в) стирание всех имен данных: MM N E Y E Y...

Теперь запишите программу на ленту через MM V P EMPTY. Вы можете также списать пустой файл сам по себе: MM V F EMPTY.

Имена данных

Желательно, хотя и необязательно, дать всем меткам записей имена. Они могут быть до 128 символов и хранятся как части файла. Для их просмотра и замены - > MM N. Появится новое меню:

A-> добавить новое имя.
N-> перебор имен.
R-> замена имени.
E-> стирание.
M или D -> возврат в MM.

Режим редактирования (EDIT MODE)

Предназначается для создания и изменения форматов сообщений. Всего может быть задано до 36 форматов.

Вход: через MM E. Далее:
A-> создание нового формата;
R-> просмотр / изменение;
M-> возврат к мм.

После A или R дайте по запросу метку сообщения 0-1; A->Z. При этом справа вверху появится сообщение REF N.
Когда формат создается впервые, ему присваивают следующие основные параметры:

а) цвет бумаги (фона) -> из 7
б) цвет бордюра (полей) -> из 7
в) без сортировки
г) интервал: одна запись на две строки.

Эти параметры можно изменить клавишей R,а затем по запросам от а) до г).
В ответ на а) или б) дайте цифру от 0 до 7.

В ответ на в) дайте метку записи, за которой должно идти ваше сообщение.
В ответ на г) дайте от 1 до 22.
Одна запись на экране -> 22.
Одна запись на строку -> 1.

Вы можете также добавить некоторые элементы, такие как линии, квадраты, и т.п. Нет ограничений на количество вспомогательных элементов. Для введения нового элемента -> режим E, опция A. Появится меню с запросом какой элемент нужен.

L - литеральный (буквенный), т.е. заголовок, рубрика, имя колонки и т.п.
B - квадрат или прямоугольник.
H - горизонтальная линия.
V - вертикальная линия.

Для L выдайте по запросу следующие данные:
- номер строки;
- микропечать *;
- колонка;
- цвет фона;
- яркость;
- инверсия;
- мигание;
- сам литеральный текст;

Если текст перейдет за правую границу, то он автоматически будет продолжен на второй строке. Цвет текста задавать не надо. Мастерфайл всегда выбирает контраст.

Для "B" задайте по запросу координаты левого верхнего угла прямоугольника, высоту в пикселях минус 1 и ширину в пикселях минус 1.

Для "H" или "V" задайте по запросу координаты левого пикселя и длину в пикселях минус 1.

Это то, что касается статичных элементов, которые выбираются через опцию A меню режима E.

Теперь рассмотрим опцию D режима E. Она определяет какие элементы, входящие в запись, должны изображаться, где и с какими атрибутами.

Нажав "A" в режиме E, нажмите "D" и ответьте на довольно большой список запросов:

A) меню данных;
B) строка , на которой должна изображаться первая избранная запись.
C микропечать *
D) колонка
E) ширина -количество колонок в ширину.
F) высота -количество строчек в высоту.
G) атрибуты -цвет фона, яркость, инверсия, мигание.
H) заполнение прямоугольника выбранным цветом фона.
I) нулевой текст. Если данные отсутствуют, то вы можете обеспечить изображение, например, прочерка и т.п.

Перебор элементов сообщения осуществляется клавишей N (NEXT), каждый из элементов может быть заменен клавишей R, далее - по запросам. Из этого режима можно перейти в режим "дисплей" через "D".

Формат сообщения может быть стерт целиком - "X" или скопирован -"C". Это удобно, если вы хотите построить формат, похожий на уже имеющийся. После "C" или "X" вы получаете исходное меню режима редактирования (EDIT).

Микропечать

Генератор символов позволяет получить разбивку экрана на 42 колонки или 51 кроме первоначальной разбивки 32.

В режиме EDIT имеется запрос: MICROPRINT Y/N.
Ответ "N"-> 32 символа в строке.
Ответ "Y"-> запрос: 42 Y/N.
Ответ "Y"-> 42 символа.
Ответ "N"-> 51 символ.

Обработка текста

В режиме "дисплей" Мастерфайл выполняет элементарные операции по обработке текста. Например, минимизируется количество переносов слов.

При этом может применяться спектрумовский символ вертикальной линии (клавиша S в режиме "E"). Этот символ формирует окончание строки и начало новой. Вы можете вводить текст в одну строку, но там, где нужен переход к другой строке, ставить этот символ .

Введение дополнительных данных (ADD A RECORD MODE)

Обычно новая запись вводится через главное меню (MM), затем -> "A". Меню предлагает вам добавить новую информацию -> "A", работать по самозапросу - "P" или выйти в режим " дисплей""D" или в главное меню -> "M". Чтобы добавить новую запись , наберите "A", затем метку, а затем текст. Текст максимум из 128 символов. Токены, графика, коды управления цветом не допускаются.

Другой способ ввода -"P" - это ввод по автозапросу. Автозапрос исключает необходимость помнить метки и исключает возможность пропуска входных данных. Вы просто вводите поля своей записи по запросу с последующим "ENTER". Если у вас нет данных по данному полю, то просто нажмите "ENTER". "ENTER" с быстрым последующим нажатием "SPACE" прекратит запросы. Добавив одно или несколько полей, вы можете просмотреть их -"N", заменить -"R", стереть "E". Вместо перебора полей задержкой "N" можно выйти на нужное поле записи через "G", а затем метку. Но если вы хотите перейти на несуществующее поле через "G",то мастерфайл встанет в ожидании другой метки или возврата через ENTER.

Если вы начали замену -"R", но хотите вернуться назад, просто сотрите свой ответ (DELETE), а затем нажмите ENTER.

Прежде , чем добавить очередную запись , вы должны вернуться в MM через "M".

Режим "дисплей" (DISPLAY MODE)

Этот режим служит для просмотра избранных записей в выбранном формате. Один из способов перехода в него - это MM ->"D". Содержимое строк 0-21 зависит от вашего формата.Строка 22 показывает следующее:

REPORT N Q=MENU NO MORE
(красн.) (Желт.) (Голуб.)

Здесь N - метка сообщения. Голубая запись показывает есть ли еще сообщения.

Q - применяется для обращения к меню.

N - если в строке 22 стоит "MORE", то N позволяет просмотреть следующую страницу записей.

1-9- переход вперед на 1-9 записей (обратного хода к сожалению нет).

B - возобновить с первой записи.

P - выдать строки 1-21 на ZX PRINTER или аналог. Используйте S для печати только этой страницы или A для печати всех страниц. Для останова печати нажмите H или BREAK с последующим GO TO 1.

U - переход в режим переработки (UPDATE) для верхней изображенной записи.

E - стирание верхней записи.

O - исключение верхней записи.

C - копирование верхней записи. Идентичная запись помещается вслед за ней. Это также является средством для вставки записей внутри файла.

S - переход в режим поиска (SEARCH).

T - рассчитать и изобразить результат.

R - переключение на другой формат.

M - возврат к главному меню (MM).

Q - изображение меню.

Режим переработки (UPDATE MODE)

Запись, подлежащая переработке - это всегда верхняя запись из режима "дисплей" и войти в режим "UPDATE" можно только из режима "дисплей", если в нем есть хотя бы одна запись.

Используя N и клавиши 1-9 установите вверху экрана нужную запись и нажмите "U". Теперь перед вами примерно те же возможности , что и в режиме "добавление записи". Разница в том, что вы можете использовать "D" для перехода в режим "дисплей".

Режим поиск (SEARCH MODE)

Первое меню предлагает:

A -поиск по всем записям;

L -поиск по выбранным записям;
M -возврат к главному меню.

После нажатия A или L, по запросу вводится метка поля данных, которое подлежит сравнению. Выход через ENTER.

Следующее меню запрашивает, является ли об'ект поиска символом (S) или числом (N). Числовые данные нормируются ONNNNNNNNNNN.NN, т.е. 23.198 воспринимается как 00000000023.19 , что нужно в целях сравнения.

Следующее меню запрашивает тип сравнения.

Ответ: G; L; U; E; S.

Заметьте, что S имеет место для поиска символьных данных (в предыдущем меню -E). И, наконец, запрос аргумента, т.е. значения с которыми д.б. выполнено сравнение.

G -больше
L -меньше
U -неравенство
E -равенство

Если ваш аргумент числовой, а при поиске встретится нечисловое поле, поиск прекратится с сообщением:

NON-NUMBERIC DATA:SKIP(OR)UPDATE

Если вы далее наберете U (переработать), то поиск не может быть возобновлен непосредственно. Если выберете S (пропустить), то поиск перейдет к следующей записи.

Символьный поиск несколько более сложен.

1. Верхний и нижний регистры букв считаются равными.
2. Если аргумент короче поля, но равен ему в своих пределах, то они считаются равными. Так, для аргумента фред равными будут: фредерик, фреда, фред и др.
3. Если аргумент длиннее поля, то - неравенство. Аргумент фред не равен полю фре.

Сканирование (S) осуществляет поиск внутри пунктов записей. Так аргумент фред совпадает со стрингами: "альфред великий" "манфред манн", "тетя фреда" и т. д.

Подсчет итога (TOTAL/AVERAGE)

Как и при поиске, данные нормируются. Все нечисловые пункты останавливают подсчет, как и в режиме "поиск" с изображением меню S/U. Если выбранная запись не имеет соответствующего поля, ему присваивается нулевое значение. Отрицательные числа не принимаются.

Подсчет итога может быть выполнен из MM или из режима " дисплей". Результат изображается в строках 19-21.

Возможно подвести итог только по одному полю данных. Для более сложных арифметических операций небходимо записатьпрограмму в бейсик-области.

Загрузка и запись на ленту

Для SAVE --> MM V.

Далее запрос F/P: F - только файл.
P - программа с файлом.
ENTER - выход.

После выбора P -запрос имени (до 10 символов). Просто ENTER означает: "то же имя, что и раньше".

Вы можете вставить автоматический VERIFY в строку 4020 перед GO TO USR R: VERIFY " " DATA F$():

(см. Командный режим, как войти в бейсик, чтобы сделать такие изменения).

При этом предолагается, что VERIFY применяется только к записи одного файла F, а не к программе.

После записи файла перемотайте ленту и проиграйте. Если все в порядке --> MM. Иначе GO TO 1 и повторите SAVE.

Для загрузки файла, который был выгружен самостоятельно через MM V F, используйте MM V L.

Вы должны дать имя файла точно!

Если вы его забыли, дайте имя xxx и проиграйте ленту. Спектрум сообщит: CHARCTER ARRAY: FILENAME, затем:

BREAK - GO TO 1 --> MM и все сначала.

Счетчики

В строке 23 почти постоянно изображается состояние файловых счетчиков.
REC S = NNNNN SEL = NNNNN SPA = NNNNN, где:
RECS - количество записей в файле,
SEL - данные, считающиеся выбранными для режима "дисплей" и т.п. Заметьте, что при создании нового файла все данные считаются выбранными.
SPA - приблизительный размер свободного пространства файла в байтах.

Бейсик

Хотя мастерфайл и предназначен для хранения файлов, возможна и обработка их через бейсик-область. Например, у вас есть файл результатов экзаменов учеников с группой пунктов экзаменационных оценок в каждой записи. Вы можете рассчитать средний балл каждого ученика и хранить его в качестве дополнительного поля в записи.

Вызов бейсика - MM и Y. Только выбранные записи поступают на обработку. В исходном состоянии бейсика в программе нет, кроме простого возвращения.

Когда вы вызываете бейсик через MM и Y, управление передается специальным строкам:

Строка 4900. Управление сюда передается только однажды, перед тем, как какие-либо записи начнут обрабатываться. В этой точке можно установить параметры, которые будут в дальнейшем использоваться в других точках. Выход через GO TO USR R .

Строка 5000. Управление передается сюда в начале каждой записи из выбранных, но до того, как пройдут какие-либо данные из записи. Используйте эту точку для инициализации любой записи. Выход через GO TO USR R.

Строка 6000. Сюда управление передается один раз в начале каждого поля выбранной записи. Поле хранится в C$ длиной 130 . C$(1) - это метка данных, а C$(2TO) - это сами данные. Вы можете переприсваивать их другой переменной если вы хотите использовать эти данные позже. Выход- GO to USR R.

Строка 7000. Сюда передается управление один раз для каждой выбранной записи после того, как все поля просмотрены в строке 6000. Здесь можно выполнить действия с данными "схваченными " в строке 6000, а затем установить C$ одним из следующих способов:

A) LET C$="" -для указания на то, что никаких изменений с записью делать не надо.

B) LET C$="X" - указывает на то, что поля, имя которых "X", должны быть стерты. "X" должен быть прописной буквой от A до Z.

C) LET C$=X и C$(2TO)=DATA - указывает на то, что поле с именем "X" должно быть вставлено или заменено, если оно уже есть. "X" должен быть прописной буквой от A до Z. Передача управления в мастерфайл - через GO TO USR R.

Заметьте, что только одно поле данных может быть задействовано в строке за один проход бейсика.

Несколько пунктов можно обрабатывать за ряд проходов, но для этого надо мудро использовать строку 4900.

Строка 9000. Управление передается сюда только один раз после того, как обработаны все выбранные записи. Эту строку можно использовать, чтобы распечатывать итог, переустанавливать переключатели, выдавать звукосигналы и т. д. Возврат - GO TO USR R.

Микродрайв

Загрузите "мф" с ленты, как обычно. Включите необходимые бейсик-строки. Очистите файл. Запишите машинный код через MM L и к.ш.6. Входя в бейсик без номера строки:

```
SAVE * "M";1;"MFMCODE"CODE N+1,65535-N
```

(здесь N -число в операторе CLEAR при загрузке программы.) Далее измените строки, содержащие SAVE и LOAD.

```
4020 SAVE *"M" ;VAL"1"; C$(1OVAL"10") DATA F$(): GO TO USR R
4030 SAVE *"M" ;VAL"1"; C$(1OVAL"10") LINE VAL"4035": GO TO USR R
4035 LOAD *"M" ;VAL"1"; '"MF MCODE"CODE! GO TO VAL"1"
4050 LOAD *"M" ;VAL"1"; C$(TO VAL 4"10") DATA F$():  GO TO USR R
```

Теперь измените строку 1:

```
1 PRINT;: PAPER VAL"7": GO to USR( )
```

Если вы не знали, то знайте, что VAL "NNN" экономит 3 байта RAM по сравнению с просто-NNN.

Далее GO TO 1,затем: SAVE через MM V P.

Теперь вы можете перенести свои файлы на картридж:

```
10  LOAD  " "DATA F$(): INPUT N$: SAVE*"M" ;1; N$ DATA F$() модифицированный
```

"мф" будет запускаться с микродрайва 1. Мы рекомендуем записывать файлы, а не программу с файлом. Заметьте, что на картридже нельзя хранить файлы с одинаковым именем, поэтому рекомендуем кроме имени фиксировать и дату, например:

```
ACCT AUG 24
```

Это поможет вам заодно при чтении картриджей решить, какой файл старше, а какой моложе. Вам надо серьезно подумать над тем, чтобы завести запасную кассету или картридж с файлами.

```
10 INPUT N$: LOAD *"M" ;1; N$ DATA F$(): SAVE N$ DATA F$()
```

Монитор 16/48

Монитор-программа предназначена для программирования в машинных кодах. Она включает в себя средства ввода и отладки программ в машинных кодах и дизассемблер.

Монитор занимает чуть больше 4к памяти у ее вершины и не может перемещаться.

Загрузка монитора стандартная : LOAD " "

Для доступа к монитору адресуйтесь к нему в зависимости от его версии:

```
RANDOMIZE USR 30479      ( для 16к )
RANDOMIZE USR 63247      ( для 48к )
```

Таким образом, вы можете в любой момент обратиться к монитору из бейсика. При этом в нижней части экрана, в дополнение к прежде написанному появится сообщение:

```
PRESS BREAK FOR MONITOR
```

После нажатия клавиши "BREAK" экран очищается и курсор помещается в нижнюю часть экрана. Монитор использует свой собственный стек, делая его недоступным для ваших программ.

Приглашение (>) указывает, что монитор готов к приему команды.

Команды монитора

```
M- MEMORY   - вывод и изменение содержимого ячеек памяти
X- ESCAPE   - выход из командного режима к началу монитора
A- AREA     - перемещает область озу в новое место.
F- FILL     - заполняет обозначенную область озу обозначенным значением байта.
I- INSERT   - вставляет до 255 байт в программу
D- DELETE   - стирает до 255 байт из программы
B-BREAKPOINT-устанавливает адрес останова в программе.
K- CODES    - восстанавливает коды после прохождения останова.
R- REGISTERS-высвечивает содержимое регистров процессора.
C- CONTINUE- продолжает выполнение программы после останова
Y- RETURN   - возвращение к бейсику.
P- PRINTER  - печать на принтере в HEX-виде.
$- DOLLAR   - ввод текста.
Z- Z80      - дизассемблирует любую область памяти
N- NUMBER   - превращение чисел  HEX-DEC  и  DEC-HEX.
```

Внимание !: Все обращения к адресам и их содержимому в этой инструкции будут в HEX-виде и приводятся в виде дву- или четырехзначных чисел, без суффикса "H".

Далее каждая команда будет разобрана подробно, с примерами, для разъяснения ее действия.

M - после ввода команды вводится адрес ячейки памяти после ввода четвертой цифры адреса справа от него появится значение ячейки:

```
>M6000 00_
```

можно изменить содержимое ячейки,например на FF:

```
>M6000 00 FF_
```

как только введена вторая HEX-цифра, содержимое ячейки изменилось.

```
M6000 00 FF
```

```
6001 00_
```
монитор выводит адрес следующей ячейки и ее со держимое. Выход из режима "M" клавиша "X".

X - команда позволяет выйти из командного режима и возвращает вас к подпрограмме монитора. Этой командой прерывается любая команда кроме "R" и "K".

I - команда позволяет вставить до 255 байтов в любое место вашей программы, сдвигая конечную часть программы на соответствующее число байтов.
Формат команды:
I AAAA BBBB NN
где I - команда INSERT;
AAAA - начальный адрес вставки;
BBBB - исходный адрес конца смещаемого блока;
NN - количество вставляемых байтов.

```
Пример:
6002 03
6003 04  / место в которое надо
6004 05    вставить 5 байт /
6005 06
6007 08
6008 09
600в 0C  - конец программы
```

вводим команду : >I 6004 6008 05

D - команда имеет действие противоположное "I".
Формат: D AAAA BBBB NN
параметры команды аналогичны команде INSERT.

A - команда перемещает указанную область озу и имеет формат:
A AAAA BBBB CCCC
где: AAAA - начальный исходный адрес смещаемой области
BBBB - конечный исходный адрес смещаемой области
CCCC - новый начальный адрес.
Эта команда смещает заданную область памяти в любом направлении, даже если новая область накладывается на исходную. Исходная область памяти сохраняется, если она не переписана в результате перемещения.

F - эта команда позволяет заполнить произвольную область памяти произвольным значением. Формат команды:
F AAAA BBBB XX
где: AAAA - начальный адрес указанной области;
BBBB - конечный адрес указанной области;
XX - вводимое значение.

Y - над клавишей "Y" напечатана команда "RETURN", и после нажатия этой клавиши и клавиши "ENTER", когда на нижней строке экрана появится приглашение и курсор, происходит возвращение в бейсик. Поскольку этот монитор не имеет собственных команд "SAVE" и "LOAD", вы можете воспользоваться командой "Y" для использования "SAVE" и "LOAD" из бейсика. Если, вернувшись в бейсик, вы снова захотите обратиться к монитору, используются следующие адреса с функцией USR, в зависимости от версии монитора:
версия 16к : RANDOMIZE USR 30479
версия 48к : RANDOMIZE USR 63247

B - позволяет временно прервать выполнение программы машинных кодах в любой точке и вернуть управление монитору, формат:
B AAAA
где: AAAA - адрес останова (AAAA должен быть первым байтом команды многобайтной команды). Коды трех адресов аааа, аааа+1, аааа+2 автоматически сохраняются в байтах данных внутри монитора, а эти ячейки загружаются командами:
CD OF F1 (для версии 48к)
CD OF 77 (для версии 16к),
что является обращением к монитору.

J - команда позволяет запустить любую вашу программу, формат :
J AAAA,
где: AAAA - начальный адрес вашей программы. Команда начинает свое действие с очистки экрана.

K - команда "K" восстанавливает только последний BREAKPOINT.

R - выводит значения всех регистров Z80 на экран.

C - позволяет продолжить работу после BREAKPOINT и выполняется нажатием "C" и "ENTER". Вы можете выйти в монитор, напечатав "X" и "ENTER". Запущенная команда будет продолжаться как будто BREAKPOINT не было. Экран очищается, программный стек восстанавливается, регистры процессора из своих блоков данных до введения адреса возврата в программный счетчик и исполнение возобновляется.

P - команда вывода содержимого памяти на SINCLIAR PRINTER. Формат команды:
P AAAA BBBB,
где: AAAA - начальный адрес для вывода;
BBBB - конечный адрес для вывода.

$ - действует таким же образом, как и "M", и позволяет вводить текст прямо с клавиатуры. Формат команды:
$ AAAA,
где : AAAA - начальный адрес текстового блока. С помощью " CAPS SHIFT " и " SIMBOL SHIFT " можно вводить любые символы, исключая "$". Графика, UDG и команды, вводимые в режиме "EXETEND MODE" не могут вводиться непосредственно. Инвертированные символы образуют доступ к областям памяти, содержащим атрибуты.

Z - эта команда дизассемблирует любую часть ОЗУ или ПЗУ, с выводом только на экран, или на экран и принтер. Распечатка включает HEX-адрес первого байта команды, HEX-значения байтов, относящихся к этой команде и мнемонику Z80 относящуюся для этой команды. Формат команды:
Z AAAA BBBB,
где: AAAA, BBBB - начальный и конечный адреса части памяти, которую вы хотите дизассемблировать.

N - эта команда превращает HEX-DEC или DEC-HEX. После ввода команды на экране появится: NUMBER N/D ? Необходимо ответить, какое число превращаем. Hex-числа должны быть четырехзначными.

Регистры Z80

регистр	16к	48к
R	7F3D	FF3D
I	7F3E	FF3E
F'	7F3F	FF3F
A'	7F40	FF40
C'	7F41	FF41
B'	7F42	FF42
E'	7F43	FF43
D'	7F44	FF44
L'	7F45	FF45

регистр	16к	48к
H'	7F46	FF46
F	7F47	FF47
A	7F48	FF48
C	7F49	FF49
B	7F4A	FF4A
E	7F4B	FF4B
D	7F4C	FF4C
L	7F4D	FF4D
H	7F4E	FF4E

регистр	16к	48к
IX/мл./	7F4F	FF4F
IX/ст./	7F50	FF50
IY/мл./	7F51	FF51
IY/ст./	7F52	FF52
SP/мл./	7F53	FF53
SP/ст./	7F54	FF54
PC/мл./	7F55	FF55
PC/ст./	7F56	FF56

M O N S 4

Раздел 1. З а п у с к .

MONS4 загружаете с желаемого адреса, и, затем, вызываете с этого же адреса. Если вы хотите запустить MONS4 вновь (вернувшись из MONS4 в BASIC), то вы должны ввести адрес, с которого первоначально загрузили отладчик.

Будучи однажды помещенным в память, MONS4 занимает приблизительно 6к в длину но вы должны предоставить непрерывную область 7к на загрузку MONS4, т.к. после основных кодов надо поместить таблицу перемещаемых адресов. MONS4 содержит собственный внешний стек, так что он является самообслуживающейся программой.

MONS4 по умолчанию загружается с адреса 55000.

П о л у ч е н и е к о п и и .

Однажды загрузив MONS4 в память вашего SPECTRUM"а, вы можете сделать копию, проделав следующие операции:

SAVE "MONS4"CODE XXXXX,6656 <ENTER> - на кассету
SAVE *"M",1,"MONS4"CODE XXXXX,6656 <ENTER> - на дискету
SAVE *"M",1,"MONS4"CODE XXXXX,6780 <ENTER> - на диск OPUS где XXXXX - адрес, с которого вы загрузили MONS4.

На входе MONS все адреса, отображаемые внутри ф.п., представлены в шестнадцатиричном формате. Вы можете изменить это с помощью команды:

<SYMBOL SHIFT> 3 и адреса будут представлены в десятичном формате.

Некоторые команды, эффект от которых может быть гибельным в случае их ошибочного использования, требуют нажатия клавиши <SYMBOL SHIFT>. Это ручное нажатие клавиши <SIMBOL SHIFT> будем представлять знаком " ", т.е. запись " Z" означает нажатие <SIMBOL SHIFT> и "Z" вместе.

Команды выполняются немедленно и не требуют после себя ввода <ENTER>. Неверные команды просто игнорируются.

Многие команды требуют ввода 16-ричного числа. При вводе такого числа вы можете ввести любое количество 16-ричных цифр (0...9 И A...F) и закончить ввод любым отличным от 16-ричного числа символом. Если ограничителем является значащая команда, то эта команда выполняется после того, как будет выполнена предыдущая. Если ограничителем является знак "-", то введенное отрицательное число будет возвращено в двойном дополнительном коде: "1800-" дает "E800". Если при вводе 16-ричного числа вы ввели более 4-х цифр, то только 4 последние введенные цифры сохраняются и отображаются на экране.

Для возврата из MONS4 в BASIC просто нажмите <STOP> (т.е. " A").

Раздел 2. К о м а н д ы M O N S 4 .

1. <SYMBOL SHIFT> 3 или #

Переключает систему счисления, в которой отображаются адреса, с 16-ричной на десятичную и обратно.

2. <SYMBOL SHIFT> 4 или $

Отображает дизассемблированную страницу, начиная с адреса, указанного MP.

3. <ENTER>

Наращивает счетчик адресов на 1, так что 24-х байтовая область памяти сдвигается на 1 байт вперед.

4. <Стрелка вверх>

Уменьшает MP (указатель памяти) на 1.

5. <Стрелка влево>

Уменьшает значение MP на 8. Используется для быстрого просмотра.

6. <Стрелка вправо>

Увеличивает значение MP на 8. Используется для быстрого просмотра.

7. <,> (Запятая)

Заменяет MP адресом, находящимся в стеке (SP). Это полезно, когда вы хотите посмотреть адрес возврата вызванной подпрограммы.

8. <G>

Поиск по образцу введенной строки. Например, скажем, вы хотите исследовать память, начиная с адреса #80000 на появление образа #3E #FF (2 байта). Последовательность действий должна быть следующей:

```
M:80000 <ENTER>      - устанавливает MP
G:3E    <ENTER>      - определяет 1-й байт строки
FF      <ENTER>      - определяет 2-й байт строки
<ENTER>              - ограничивает строку
```

9. <H>

Вам выводится подсказка ":" на ввод десятичного числа, ограниченного не цифрой (любым символом, кроме 0...9).

10. <I>

Используется для копирования блока памяти из одного места в другое, причем блок памяти может быть скопирован в ячейки с перекрытием своего первоначального местоположения. <I> выдает подсказки на ввод стартового и конечного адресов блока, который должен быть скопирован (<FIRST:>, <LAST:>) и затем для адресов,

по котрым он должен быть помещен (<TO:>).

11. <J>

Выполняет задачу с определенного адреса. Эта команда выдает подсказку ":" для 16-ричного числа.

12. <SYMBOL SHIFT> K

Продолжает выполнение с адреса, находящегося в счетчике инструкций

13. <L>

Распечатывает блок памяти, начиная с адреса, находящегося в данный момент в MP.

14. <M>

Заносит в MP определенный адрес. Выдается подсказка ":" на ввод 16-ричного адреса. Mp обновляется введенным адресом и соответственно на экране меняется отображение области памяти

15. <N>

Находит следующее появление строки, определенной в прошлый раз командой <G> <G> позволяет вам определить строку и затем осуществить поиск ее первого появления.

16. <O>

Команда берет байт, адресуемый в данный момент MP, и представляет его как относительное смещение, соответственно обновляя отображение.

17. <P>

Заполняет память между определенными границами нужным байтом. <P> выдает подсказки <FIRST:>, <LAST:>, <WITH:>. Введите 16-ричные числа в соответствии с этими подсказками (соответственно: начальный и конечный адреса (включительно) блока, который вы желаете заполнить, и байт заполнитель).

18. <Q>

Переключение наборов регистров. На входе ф.п. отображается стандартный набор регистров (AF, HL, DE, BC). При использовании <Q> будет отображаться альтернативный набор регистров (AF", HL", DE", BC").

19. <SYMBOL SHIFT> T

Устанавливает точку останова после текущей инструкции и продолжает выполнение.

20. <T>

Дизассемблирование. Дизассемблирует блок кодов на принтер и/или дискету (по ключу).

21. <U>

Используется совместно с командой <O>. Помните, что она обновляет отображение памяти в соответсвии с относительным смещением.

22. <V>

Используется совместно с командой <X> и подобна команде <U> по эффекту, с той разницей, что она обновляет отображение памяти на картинку, которая была перед последней командой <X>.

23. <W>

Установка точки останова. Точка останова для MONS4 это просто инструкция вызова внутри MONS4 подпрограммы, которая отображает ф.п. так, что программисту дается возможность остановить выполнение программы и проверить регистры Z80, флаги и любые значащие ячейки памяти.

24. <X>

Используется для обновления MP по назначению абсолютного вызова подпрограммы или по инструкции JP.

25. <Y>

Ввод ASCII. <Y> дает вам новую строку, которую вы можете ввести символами ASCII прямо с клавиатуры.

26. <SYMBOL SHIFT> Z

Перед использованием " Z" (или " T") PC должен быть установлен на адрес инструкции, которую вы хотите выполнить. "Z" просто выполняет текущую инструкцию и обновляет ф.п., чтобы отобразить изменения, вызванные выполнением данной инструкции.

27. <SYMBOL SHIFT> P

Эта команда аналогична LIST, но вывод вместо экрана идет на принтер. Помните, что в конце страницы вы нажимаете <EDIT> для возврата к ф.п. или любую другую клавишу для получения другой страницы.

28. Модификация памяти

Значения ячеек по адресам, которые дает MP, могут быть модифицированы вводом 16-ричного числа, за которым следует ограничитель (см. Раздел 1). Две последние 16-ричные цифры (если одна, то она дополняется нулем) вводятся в ячейку, на которую указывает MP, и затем команда (если она есть), определенная ограничителем, выполняется. Если ограничивается незначащая команда, то она игнорируется.

G E N S 4

Р а з д е л 1. З а п у с к.

1.1. Введение и инструкции загрузки

GENS4 занимает приблизительно 10к в памяти и использует свой внешний стек, так, что это - замкнутая часть программного обеспечения. Она содержит собственный строчный редактор, который помещает текстовой файл сразу за кодами GENS4, в то время, как таблица символов ассемблера создается за текстовым файлом, поэтому вы должны предоставить достаточное пространство, чтобы поместить сам ассемблер, таблицу символов максимального размера и текстовый файл, который вы, вероятно, будете использовать в текущем сеансе. Поэтому загружать GENS4 часто удобнее в нижние адреса памяти.

Существует 2 версии ассемблера на кассете, обе на стороне 1. Сначала помещена версия с 51 символом в строке, за ней следует рассматриваемая версия с 32 символами. Их имена на ленте являются соответственно GENS4-51 и GENS4 и вы должны использовать ту версию, которая больше вам подходит. Версия GENS4-51 на 400 байт длиннее, чем GENS4.

Для загрузки GENS4 поместите магнитную ленту с ним в кассетный магнитофон и введите:

LOAD ""CODE XXXXX [ENTER] и нажмите клавишу "воспроизведение" магнитофона.

или:

LOAD "GENS4" CODE XXXXX [ENTER], или:

LOAD "GENS4-51" CODE XXXXX [ENTER], где: XXXXX - десятичный адрес, с которого вы хотите загрузить GENS4.

Только после того, как вы загрузите коды GENS4 в компьютер, вы можете войти в ассемблер, набрав:

RANDOMIZE USR XXXXX, где

XXXXX - адрес, с которого загружены коды ассемблера.

Если в любой последующий момент вы вновь пожелаете войти в ассемблер, то вы должны просто адресоваться к ячейке XXXXX, которая хранит предварительно созданный файл информации.

1.2. Получение копии системы

После загрузки GENS4 в память вашего SPECTRUM"а, вы можете получить копию ассемблера следующим образом:

```
SAVE "GENS4-51" CODE XXXXX,11392 [ENTER]          или
SAVE "GENS4"2 CODE XXXXX,11010 [ENTER]            для кассеты
SAVE * "M";1;"GENS4-51" CODE XXXXX,11392 [ENTER]  или
SAVE * "M";1;"GENS4" CODE XXXXX,11392 [ENTER]     для микродрайвера,
```

где: XXXXX - адрес, с которого вы загрузили GENS4.

Р а з д е л 2. П о д р о б н о с т и G E N S 4

2.0. Принцип работы GENS4, ключи ассемблера, формат листинга

GENS4 является быстрым двухпроходным ассемблером для Z80, который ассемблирует всю стандартную мнемонику для Z80 и содержит следующие особенности: макросы, условное ассемблирование, множество команд ассемблера и таблицу символов в виде двоичных троек.

Когда вы вызываете ассемблирование, то вы используете команду "A":

A4,2000,1:TEST [ENTER]

Первый код (4 - см.выше) после A определяет условия трансляции, которые вы хотите иметь в данный момент. Эти условия трансляции и их ключи будут перечислены ниже. Если вы не хотите использовать ключи, то номер набирать не надо, поставьте только запятую.

Второй код (2000 - см.выше) - это десятичный размер символов в байтах. По умолчанию (при простом использовании запятой без кодов), GENS4 будет выбирать

размер таблицы символов, подходящий к размеру исходного файла - обычно это бывает вполне приемлемо. Однако, при использовании ключа "включить" вы можете задать размер таблицы символов больше обычного, ассемблер в этом случае не будет определять размер таблицы символов, которая будет построена.

После указания размера таблицы символов вы можете ввести имя файла, например, 1:TEST, как указано выше. Если вы это сделаете, то результирующий объектный код, полученный в результате ассемблирования, будет сохранен автоматически, независимо от размера генерируемого кода. Если вы не хотите использовать такую возможность, то имя файла не вводите. Не вводите вторую запятую, иначе GENS4 будет считать, что вы вводите пустое имя файла! Более подробно с командой "A" можно познакомиться в приложении 3.

Ключи ассемблирования:

1 - выдает листинг таблицы символов в конце второго прохода ассемблирования;
2 - запрет на генерацию объектного кода;
4 - производит листинг ассемблирования (отметим, что в ранних версиях ассемблера такой возможности не было);
8 - направить листинг ассемблирования на печать;
16- просто помещает объектный код, если он, конечно, получен,после таблицы символов. Счетчик адресов обновляется таким образом, что объектный код может быть помещен в одну область памяти, а работать - в какой - либо другой;
32- исключение проверки того, куда помещается объектный код (это бывает полезно для ускорения ассемблироания);

Для того, чтобы скомбинировать ключи, просто складывайте их один с другим. Так, код "A33" производит быстрое ассемблирование - не выдается листинг, не производится проверка, чтобы увидеть, куда в настоящий момент помещают объектный код, при этом в конце выдается листинг таблицы символов.

Отметим, что при использовании ключа "A16" директива "ENT" не будет иметь эффекта. Если использован ключ "A16", то вы можете определить, где находится объектный код, используя команду редактора "Y". По этой команде можно определить конец текста (его называют второй номер) и затем добавить к нему назначенный размер таблицы +2.

Ассемблирование происходит за два прохода: в течение 1-го прохода GENS4 ищет ошибки и собирает таблицу символов, второй проход генерирует объектный код (если не указан ключ 2). В течение первого прохода на экране и принтере ничего не отображается, пока не встретится ошибка. В случае ошибки будет выдано сообщение об ошибке с номером ошибки за ним (см. Приложение 1). Ассемблирование приостанавливается - нажмите клавишу "E" для возврата в редактор или любую другую клавишу для продолжения ассемблирования следующей строки.

В конце первого прохода будет выдано сообщение:

```
PASS 1 ERRORS: NN
```

Если хоть одна ошибка будет обнаружена, то ассемблирование затем будет остановлено и перехода ко второму просмотру не произойдет. Если какие-нибудь метки были обнаружены в поле операнда, но не объявлены в поле метки, то последует сообщение:

```
*WARNING* LABEL ABSENT
```

В течение второго прохода генерируется объектный код (если генерация не запрещена ключом 2 (см.выше)).

Во время второго прохода листинг ассемблера не производится, кроме случая, когда указан ключ 4 или команда ассемблера L+.

В 32-х символьной версии листинг ассемблера состоит обычно из двух строк в следующей форме:

```
C000   210100   25   LABEL
                     LD HL,1
1      6        11   21...26
```

Тогда как в 51-символьной версии листинг печатается в одну строку. Если листинг не помещается в одну строку, то он завершается на следующей.

На первом месте в строке стоит значение счетчика адресов, который соответствует данной строке. Если в строке встречаются инструкции ORG, EQU или ENT (см.2.7), то на первом месте будет стоять значение поля операнда инструкции. Эта запись обычно отображается в шестнадцатиричном виде, хотя используя команду "*D+" ее можно отобразить и в неудобном десятичном виде (см.2.9).

Следующая запись с позиции 6 занимает до 8 символов в строке (предстаВляет до 4-х байтов) и является объектным кодом текущей инструкции (см.ниже команду *C).

Затем следует номер строки - целое число в диапазоне 1... 32767.

Позиции 21...26 Первой строки содержат 6-символьную метку, определенную в этой строке.

После метки следует мнемоника инструкции, которая занимает поз. 21 . 24.(В 32-символьной версии это будет новая строка, если только не использовать команду *C).

Затем, с позиции 2, следует поле операндов; и завершают строку комментарии, которые при необходимости можно продолжить на другой строке.

Команда *C ассемблера может быть использована для получения короткой строки листинга ассемблирования - ее вводят, чтобы не включать 9 символов, представляющих объектный код инструкции. Это дает возможность отобразить большинство строк ассемблера в виде, удобном для экрана со строкой в 32 позиции (см.ниже разд.2.8.).

Изменение формата листинга. (Только для 32-символьной версии)

С помощью команды "POKE" вы можете модифицировать форму, по которой каждая строка листинга разбивается внутри 32-символьной версии GENS4 на 3 части. Указания, как это сделать, приведены ниже. Мы отличаем понятие "строка ассемблера", которая является текущей строкой листинга ассемблера и хранится во внешнем буфере, от понятия "экранная строка", которая является строкой, действительно возникающей на дисплее. Строка ассемблера обычно порождает более одной экранной строки.

1. 51-Й (#33) байт от начала GENS4 содержит значение, определяющее позицию, до которой будет отображаться строка листинга на экране. Циклически меняя этот байт от 0 до любого другого значения (256) можно заканчивать первую экранную строку в необходимом месте (это особенно полезно при использовании микроформатного принтера).

2. 52-Й (#34) байт от начала GENS4 дает номер позиции, с которой должна начинаться каждая последующая экранная строка.

3. 53-Й (#35) байт от начала GENS4 указывает количество символов остатка строки ассемблера, который будет выведен во второй экранной строке после первой.

Например, вы хотите, чтобы каждая первая экранная строка содержала 20 символов (то есть не включала бы поле метки) и, затем, каждая последующая экранная строка начиналась с позиции 1 и занимала бы всю строку. Также предположим, что вы загрузили GENS4 с десятичного адреса 26000. Чтобы получить вышеуказанные изменения, надо из бейсика ввести следующие инструкции:

```
POKE 26051,20
POKE 26052,1
POKE 26053,31
```

Вышеприведенные модификации возможны только в том случае, когда не была использована команда *C.

Листинг может быть приостановлен в конце строки нажатием [CAPS SHIFT] и [SPACE] вместе. В последующем нажимайте клавишу "E" для возврата в редактор или любую другую клавишу для продолжения листинга.

Единственными ошибками, которые могут появиться во втором проходе, являются ошибки типа "*ERROR*10" (см.Прил.1) И "BAD ORG!" (Которая появляется, когда объектный файл накладывается на GENS4, исходный файл или на таблицу символов - это может быть предотвращено использованием другого формата). *ERROR* 10 - это не фатальная ошибка, поэтому вы можете продолжать ассемблирование, как и на первом проходе, тогда как ошибка "BAD ORG!" Является фатальной и сразу же возвращает управление редактору.

В конце второго прохода на дисплее будет отображено сообщение об отсутствующих метках:

PASS 2 ERRORS:NN

Затем появится следующее сообщение:

TABLE USED: XXXXX FROM YYYYY, это сообщение информирует о том, какая часть таблицы символов была заполнена. В этом случае, если была правильно использована директива "ENT", выдается сообщение:

EXECUTES: NNNNN IS DISPLAYED

Это показывает стартовый адрес задачи. Вы можете выполнить задачу, используя

команду редактора "R". Будьте осторожны при использовании команды "R" в том случае, когда вы не завершили успешно ассеблирование и не увидели сообщения "EXESUTES:NNNNN".

Итак, если указан ключ "1", то используется список меток в алфавитном порядке и будут напечатаны соответствующие им значения. Количество информации, отражаемой в одной строке, может быть изменено директивой "POKE" с адреса "начало GENS4+50" помещением в эту ячейку уместного значения, по умолчанию заносится 2.

Контролируйте возврат в редактор.

2.1. Формат оператора ассемблера.

Каждая строка текста, которая будет обрабатываться GENS4, должна иметь следующий формат, где определение поля необязательно:

```
метка   код команды   операнды   комментарий
START        LD       HL,LABEL ; получение метки
```

Пробелы и табуляция обычно игнорируется. Строка обрабатывается следующим образом:

Первый символ строки проверяют и последующие действия зависят от его сущности:

- `;` – целую строку трактуют как комментарий, т.е. просто игнорируют;
- `*` – предполагается, что следующий символ (символы) образуют команду ассемблера (см. пункт 2.8). Все символы после команды трактуются как комментарии;
- `<CR>` – символ конца строки. Линия просто игнорируется;
- `-` – пробел или табуляция. Если первым символом строки является пробел или табуляция, то GENS4 предполагает, что следующий отличный от пробела или табуляции символ будет началом кода команы Z80.

Если первый символ строки отличается от вышеприведенных, то в строке должна быть метка – см.п.2.2.

После обработки присутствующей метки (или если первый символ в строке – пробел или табуляция) ассемблер исследует следующий значащий символ и предполагает, что он будет или символом конца строки, или началом команды Z80 (см.Прил. 2), состоящий из 4-х символов и ограниченный пробелом (табуляцией или <CR>).

Иногда присутствует код команды и требуется один или более операндов, тогда поле операндов следует после набора пробелов (табуляции).

Оператор может состоять только из одной команды. Это бывает полезно для улучшения читаемости листинга.

Комментарии могут следовать в любом месте после поля операндов или поля кода команды (если у нее нет аргументов).

2.2. Метки

Метка – это символьное имя, представляющее 16 бит информации.

В метках можно использовать символы 0...9,A...Z и $. Можно использовать как большие, так и малые буквы, символы [,\,], ,#,и -. Метка должна начинаться с буквы.

2.3. Таблица символов

Когда метка встречается в первый раз, ее помещают в таблицу вместе с двумя признаками, которые позже показывают, как эту метку соотносят по алфавиту с другими, находящимися внутри таблицы. Если в первый раз метка появляется в поле меток, то ее значение (данное счетчиком адресов или значение выражения "EQU") заносят в таблицу символов. В другом случае значения заносятся, когда бы ни встретилось имя метки в последующем в поле метки.

Запись в таблице занимает от 8 до 13 бит, в зависимости от длины имени.

Если за время первого прохода имя встречается более одного раза, то будет выдано сообщение об ошибке (*ERROR *4).

Если значение для имени не найдено, то в конце ассемблирования будет выдано сообщение:

WARNING SYMBOL ABSENT

2.4.Выражения

Выражения -это запись операндов, образованная или простым термом или комбинацией термов, разделенных операндом. Ниже определены понятия терма и операнда:

Терм:

десятичная константа	например:	1024
шестнадцатиричная константа	например:	#4A5
двоичная константа	например:	%100101
символьная константа	например:	"A"
метка	например:	L1029

Может также использоваться для обозначения текущего значения счетчика адресов.

Оператор:

+ - сложение
- - вычитание
$ - логическое "и" (AND)
& - логическое "или" (OR)
! - логическое "XOR"
* - алгебраическое умножение
/ - алгебраическое деление
? - MOD-функция (A?B=A-(A/B)*B

Замечание:

\# - используют для того, чтобы отметить начало 16-ричного числа.
% - используют для того, чтобы отметить начало двоичного числа.
" - символьная константа.

2.5.Макроопределения

Макроопределение состоит из набора инструкций ассемблера вместе с именем макроопределения. Когда это имя в последующем используется в поле кода операции, то оно будет заменено на все инструкции ассемблера, составляющее это макроопределение.

Так, макроопределение NSUB может быть определено следующим образом:

```
NSUB   MAC
       OR   A
       SBC  HL,DE
       ADD  HL,DE
       ENDM
```

Когда бы мы в дальнейшем ни применили NSUB как код операции, оно будет генерировать три инструкции ассемблера: OR A;SBC HL,DE и ADD HL,DE.

Макроопределения не могут быть вложенными, так что вы не сможете ни определить макроопределение, ни вызвать макроопределение из макроопределения.

2.6. Директивы ассемблера

Возможны директивы:

ORG <выражение> - устанавливает счетчик адресов равным значению выражения.

EQE <выражение> - этой директиве должна предшествовать метка. Значению метки присваивается значение выражения. Выражение не должно содержать имен, которым еще не присвоено значение. В противном случае ассемблер выдает вам сообщение об ошибке - *ERROR*13.

DEFB <выражение>,<выражение>... - Каждое выражение должно оценивать 8 бит; байт, адрес которого в настоящий момент содержится в счетчике адресов, принимает значение выражения и счетчик адресов увеличивается на 1. Все вышеуказанное повторяется для каждого выражения.

DEFW <выражение>,<выражение>... - Дополняет слово (2 байта), адрес которого указан счетчиком адресов, значением выражения и увеличивает счетчик адресов на 2. Младший байт помещается сначала, за ним следует старший байт. Повторяется для каждого выражения.

DEFS <выражение> - увеличивает счетчик адресов на значение выражения. Эквивалент резервирования области памяти размером, равным значению выражения.

DEFM "S" - определяет, что последовательность N байтов памяти будет

равна представлению строки "S" в коде ASCII, где N - длина строки.

ENT <выражение> - используется для определения адреса, на который осуществляется переход по команде редактора "R".

2.7. Команды условной трансляции

Команды условной трансляции обеспечивают программисту возможность включения или не включения определенных секций исходного текста в процесс ассемблирования.

IF <выражение> - эта команда оценивает выражение. Если результат равен нулю, то прекращается ассемблирование следующих линий до тех пор, пока не встретится ELSE или END. Если значение выражений отлично от нуля, то ассемблирование происходит нормально.

ELSE - эта команда просто переключает ассемблирование. Если перед появлением ELSE было ассемблирование, то в последующем оно выключается и наоборот.

END - просто включает ассемблирование. Отметим, что команды условной трансляции не могут быть вложенными.

2.8 Команды ассемблера

В распоряжении программиста имеются следующие команды:

*E - выдает 3 пустые строки на экран или принтер.

*HS - выдает строку "S" - заголовок, который будет печататься после каждого вывода "*E". *H автоматически выполняет *E.

*S - указывает, что с данной строки листинг должен быть остановлен.

*L+ - прекращает листинг и печать, начиная с данной строки.

*L- - начинает листинг и печать, начиная с данной строки.

*D+ - указывает, что значение счетчика адресов в начале каждой строки должно выдаваться в десятичном виде вместо обычного 16-ичного.

*D- - возвращает к использованию шестнадцатиричного вида значения счетчика адресов в начале каждой строки.

*C- - укорачивает листинг ассемблера, начиная со следующей строки.

*C+ - возвращает к полному листингу.

*M+ - включает печать макроопределения.

*M- - выключает печать макроопределения.

*F <имя файла> - позволяет ассемблировать текст с ленты или дисковода.

*F 2:TEST - для файла на микроприводе N2;

*F TEST - для файла на ленте.

Если не указано имя файла, то с ленты считывается первый найденный файл, но при считывании с дисковода вышеуказанное нельзя принимать в расчет. Если вы работаете с дисководов, то файл должен быть предварительно сохранен с помощью команды редактора "P" обычным способом.

Раздел 3. Строчный редактор

3.1 Вставка текста

Текст может быть введен в текстовый файл или указанием номера строки, пробела и требуемого текста, или посредством команды "I".

Клавиша <DELETE> будет уничтожать предыдущий символ (но не далее начала строки текста).

Текст вводится во внешний буфер внутри GENS4 и вы должны оградить его от переполнения использованием клавиш <DELETE> или "<--" для освобождения свободного пространства.

Команда " I N,M "

Использование этой команды переводит ввод в автоматический режим: вам подсказываются номера строк, начиная с N, с приращением M на каждом шагу. После высвечивающегося номера вы вводите нужный текст, по желанию используя нужные клавиши, и завершаете строку текста вводом <ENTER>. Для выхода из этого режима используется <EDIT>.

Если вы вводите строку с уже существующим номером, то строка текста с этим номером удаляется и заменяется на вновь введенную после нажатия <ENTER>.

Если автоматическое увеличение номера строки дает значение, большее 32767, то происходит автоматический выход из режима ввода.

Если при вводе текста вы добрались до конца строки на экране, но еще не ввели 64 символа (размер буфера), то экран сдвинется вверх на одну строку и вы можете продолжать ввод со следующей строки - номер строки автоматически будет соответствовать введенному тексту.

3.2 Распечатка текста

Команда "L N,M"

Эта команда выводит листинг со строки N до строки M включительно на терминал. По умолчанию N присваивается значение 1, M - 32767, т.е. значениями по умолчанию не являются ранее введенные аргументы.

Для листинга целого файла просто используйте команду "L" без аргументов.

Команда " K N "

"K" устанавливает количество экранных линий, которые должны отбражаться на терминале перед паузой, как это было описано выше (см.команду "L"). Значение N (не более 256) хранится в памяти. Например, если вы хотите при последующем использовании "L" выдавать на экран по 5 строк, то введите команду "K5".

3.3 Редактирование текста

Команда " D [N,M] "

Все линии от N до M включительно удаляются из текстового файла. Если M<N или определено менее двух аргументов, то команда игнорируется.

Команда " M N,M "

Эта команда помещает текст строки с номером N в строку M, уничтожая текст, уже существующий там.

Команда " N [N,M] "

Команда "N" перенумеровывает M строк текстового файла, начиная со строки N. N и M должны быть реальными и если номер линии превышает 32767, то остается первоначальная нумерация.

Команда "F N,M,F,S "

Текст со строки N до строки M исследуется на появление строки F. Если такая строка найдена, то она отображается на терминале и включается режим "EDIT" (см. ниже).

Вы можете использовать команды внутри режима EDIT для нахождения последующих появлений строки F в пределах определенной области строк или для замещения строкой S текущего появления строки F и затем продолжить поиск строки F. Отметим, что диапазоны строк F и S могут быть установлены ранее другой командой , так что возможно вводить только F для инициализации поиска.

Команда " E N "

Редактирует строку с номером N. Если строки N не существует, то команда игнорируется.

В вашем распоряжении имеются следующие подкоманды:

<SPACE> - перемещает курсор на одну позицию к следующему символу в строке.
<DELETE> - возвращает курсор на предыдущий символ в строке.
< --> > - Перемещает курсор на следующую позицию табуляции в каждой экранной строке.
<ENTER> - конец редактирования данной строки. Сохраняет все сделанные изменения.
<Q> - выход из режима редактирования данной строки.
<R> - перезагружает буфер редактирования текстом.
<L> - распечатывает остаток строки, который должен быть отредактирован.
<K> - уничтожает символ в указываемой курсором позиции.
<Z> - уничтожает все символы, начиная с указанного курсором и до конца строки.
<F> - ищет следующее появление строки "F",ранее определенной в командной строке.
<S> - замещает ранее определенной строкой "S" текущее появление цепи символов "F" и затем выполняет подкоманду "F".
<I> - вводит символ в указанную курсором позицию. Вы будете оставаться в этом режиме до тех пор, пока не нажмете <ENTER>.
<X> - перемещает курсор в конец строки и включает описанный подрежим.
<C> - изменяет подрежим. Это позволяет перезаписать символ в текущей позиции, затем передвигать курсор через одну позицию.

3.4 Команды дисковода и магнитной ленты

Команда " P N,M,S "

Строки с номерами от N до M сохраняются на ленте/диске в файле с именем, заданным строкой S. Текст будет записан на диск, если перед именем файла через двоеточие стоит номер дисковода.

Команда " G,,S "

На ленте или дискете производится поиск файла с именем "S". Когда файл найден, он загружается в конец текущего текста. Если строка "S" пустая, то загружается первый файл с кассеты. Для дисковода обязательно надо указать имя файла и номер дисковода.

При работе с кассетой, после того как вы ввели команду "G", появится сообщение: START TAPE.

Вы должны нажать клавишу "воспроизведение" своего магнитофона. Ведется поиск файла с указанным именем или первого файла по умолчанию. Когда нужный файл найден, появится сообщение: USING FILENAME, в противном случае высветится сообщение: FOUND FILENAME и поиск на ленте продолжается. При использовании дисковода и в том случае, если не найден искомый файл, то появится сообщение "ABSENT".

Команда T N,M,C

Выводит блок текста между строками N и M включительно на ленту в формате, подходящем для дальнейшей работы под управлением директивы ассемблера *F. Обрабатывается файл с именем "S". Вывод начинается сразу после нажатия клавиши <ENTER>.

Команда " O,,S "

Выводит ваш объектный код на кассету или дисковод. Имя файла может иметь длину более 8 символов и должно начинаться с номера привода (1...8) и двоеточия, если вы хотите сохранить объектный код на дискете.

3.5 Ассемблирование и запуск из редактора

Команда " A O,S,F "

Эта команда ассемблирует текст с первой строки текстового файла. Команда "O" позволяет вам задать ключи, которые должны использоваться во время этой трансляции. Обычно достаточно использовать значения ключей по умолчанию, просто употребляя запятые.

S - дает возможность изменить размер таблицы символов для этой трансляции.

F - имя файла, имеющегося на дискете. Имя файла должно начинаться с <D:>, где D - номер дисковода, на котором размещен файл. Имя файла не должно содержать более 10 символов.

Команда "R" - для выполнения объектной программы.

3.6 Другие команды

Команда "B"

Просто возвращает управление операционной системе.

Команда "C"

Позволяет вам изменить размер входного буфера и буфера макроопределений (только для версий DEVPAC на магнитной ленте).

Команда " S,,D "

Эта команда позволяет вам изменить символ-разделитель аргументов в командной строке. На входе редактора таким разделителем является запятая, это может быть изменено с помощью команды "S" на первый символ определяющей строки "D".

Команда " U N "

Позволяет установить верхнюю границу памяти равной N. Если N не указать (т.е. ввести только U и <ENTER>), то отображается текущая верхняя граница памяти.

По умолчанию верхняя граница памяти принимается равной вершине стека памяти системы SPECTRUM.

Команда " V "

Выдает на дисплей полезную информацию: значение параметров команды N1 и N2 по умолчанию; символ-разделитель команды по умолчанию; десятичное значение начала и конца текста и значение первой командной строки S1.

Команда " W N,M "

Выводит строки текста с N по M включительно на принтер. Если ни N, ни M не указаны, то будет напечатан весь файл. Печать будет приостановлена после вывода

некоторого числа линий, определенного командой "к" – нажмите любую клавишу для продолжения печати.

Команда " X N "

Выдает каталог диска. В версии с 51-м символом перед выдачей каталога происходит очистка экрана.

Команда " Z "

Эффективно уничтожает весь ваш текст.

Команда " H "

Выдает подсказку на экран: список возможных команд в 2 столбца с заглавной буквой, обозначающий команду и отображающий текущее значение определенных важных параметров.

П р и л о ж е н и е 1 .

Коды ошибок и их значение.

*ERROR*1 – ошибка в константе этой строки;
*ERROR*2 – мнемоника не распознана;
*ERROR*3 – утверждение плохо сформировано;
*ERROR*4 – символ определен более 1 раза;
*ERROR*5 – строка содержит неверный символ (т.е. ничего не значащий в данном случае);
*ERROR*6 – один из операндов в строке – незаконный;
*ERROR*7 – символ в строке является резервной инструкцией;
*ERROR*8 – ошибочная пара регистров;
*ERROR*9 – слишком много регистров в строке;
*ERROR*10 – выражение, которое должно занимать менее 8 бит, занимает больше;
*ERROR*11 – неверные инструкции JP(IX+N) и JP(IY+N);
*ERROR*12 – ошибка в директиве ассемблера;
*ERROR*13 – незаконная ссылка вперед, т.е. EQU ссылается на символ,который еще не был определен;
*ERROR*14 – деление на ноль;
*ERROR*15 – переполнение в операции умножения;
*ERROR*16 – вложенное макроопределение;
*ERROR*17 – этот идентификатор – не макро;
*ERROR*18 – вложенный макровызов;
*ERROR*19 – вложенный условный оператор;

BAD ORG! – Директива ORG работает с адресом, который может испортить GENS4, текстовый файл или таблицу символов. Управление возвращается в редактор.

OUT OF TABLE SPACE – появляется во время первого прохода, если на таблицу символов выделено недостаточно памяти. Управление возвращается редактору.

BAD MEMORY! – Нет места для ввода текста, т.е. конец текста близок к вершине ОЗУ памяти. Вы должны спасти текущий текстовый файл или его часть.

П р и л о ж е н и е 2 .

Резервирование слов, мнемоника и т.д.

Список резервных инструкций внутри GENS4. Эти символы не могут использоваться как метки, хотя они могут быть частью меток. Все резервные инструкции состоят из заглавных букв:

A	B	C	D	E	H	L	I	R	S
AF		AF"		BC		DE		HL	IX
IY		SP		NC		Z		NZ	M
P		PE		PO					

Ниже приводится список мнемоники Z80, директив ассемблера и его команд. Они также должны состоять из заглавных букв.

ADC	ADD	AND	BIT	CALL	CCF	CP	CPD	CPDR	SPI
CPIR	CPL	DAA	DEC	DI	DJNZ	EI	EX	EXX	HALT
IM	IN	INC	IND	INDR	INI	INIR	JP	JR	LD
.....	LDD	LDDR	LDI	LDIR	NEG	NOP	OR	OUTDR	OTIR
OUT									
OUTD	OUTI	POP	PUSH	RES	RET	RETI	RETN	PL	RRA
RLC	RLCA	RLD	RR	RLA	RRC	RRCA	RRD	RST	SBC
SCF	SET	SLA	SRA	SRL	DEFB	DEFM	DEFS	DEFW	ELSE
END	ENT	EQU	IF	ORG	MAC	ENDM			
*D	*E	*H	*L	*S	*C	*F	*M		

Z E U S

Система ZEUS обеспечивает оптимальный способ написания машинного кода Z80.

О б щ и й ф о р м а т и с х о д н о г о т е к с т а

Текст должен состоять из одного или нескольких операторов, разделяемых двоеточиями. Каждый оператор состоит из:

1) необязательной метки;
2) команды;
3) необязательного комментария.

Н е о б я з а т е л ь н а я м е т к а

-метка может содержать буквы верхнего и нижнего регистра и цифры;
-метка должна начинаться с буквы;
-длина метки может составлять 14 знаков;
-метка не должна быть идентичной резервируемому слову;
-метка должна быть отделена от команды пробелом.

К о м а н д а

Команда может быть любой из стандартных команд ЦП ZILOG Z80 или может быть директивой ассемблера(смотрите ниже).

Н е о б я з а т е л ь н а й к о м м е н т а р и й

Комментарий может быть добавлен в конце любой команды. Он должен быть отделен от команды точкой с запятой .

К о н с т а н т ы

Константы могут быть представлены либо в десятичном, либо в шестнадцатиричном виде следующим образом:

десятичный	1	99	234	4096
шестнадцатиричный	#A	#FE	#6843	#5000

литералы могут быть представлены следующим образом: "a "7 "? "$ "=

О п е р а т о р ы

В ZEUS допускается использование логических операторов:

+	сложение
-	вычитание
&	логическое "и" (AND)
!	логическое "или"("OR")

Никакой приоритет операторов не соблюдается; выражения вычисляются строго слева направо.

В ы р а ж е н и я

Где бы ни требовалось использование в команде константы, на ее месте может быть употреблено выражение . Выражения строятся из меток и/или констант, разделяемых операторами .

Д и р е к т и в ы а с с е м б л е р а.

ORG NNNN	укороченное имя директивы ORIGIN (начало). Данная директива инструктирует ассемблер ZEUS ассемблировать блок машинного кода, согласно утверждению ORG, начиная с адреса NNNN (при условии, что текущее значение в директиве DISP равно 0).
DISP NNN	укороченное имя директивы DISPLACEMENT (смещение). Директива DISP инструктирует изменить адрес, с которого генерируется последующий код, ассемблируется для выполнения с адреса, специфицированного посредством текущего параметра директивы ORG.
ENT	устанавливает точку входа. Команда 'X' выполняет ассемблированный код, начиная с последней директивы ENT (вход) в исходном файле.
EQU	укороченное имя директивы EQUATE (приравнять) или EQUALS (равно). Метка может иметь значения, назначенные ей посредством утверждения вида: метка EQU значение
DEFB NN,NN	эта директива вставляет байты NN по текущему адресу ассемблирования
DEFW NNNN,....	Эта директива вставляет слова (адреса) NNN по текущему адресу ассемблирования
DEFM/строка/	текст,заключенный ограничителями '/,", будет вставлен по текущему адресу ассемблирования

Наряду со всеми командами языка ассемблера директивы могут иметь в качестве

префикса метку, например:

```
   DATAI DEFW/4000,3456,6879
   OFFSETS DEFB 3,2,14,20,9,2
   USRADDR ENT
```

Команды ZEUS

A X ассемблирует исходный файл и выводит на дисплей сообщение об ошибках.

D X Y удаляет все строки, расположенные между строками X и Y включительно.

F "цепочка знаков" XYZ эта команда выполняет поиск исходного файла для всех случаев присутствия заданной цепочки. Поиск начинается со строки с номером X и завершается на строке с номером Y. Любые строки, содержащие эту цепочку знаков, выводятся на дисплей и после вывода Z строк инициируется пауза.
По умолчанию: X = наименьший номер строки
Y = наибольший номер строки
Z = 14
"цепочка знаков" = ""

I X Y после ввода этой команды ассемблер ZEUS будет автоматически генерировать номера строк, начиная с номера X. По умолчанию: X = 10, Y = 10. Выход – удаление номера текущей строки.

L X Y H выводит листинг исходного файла, начиная со строки с номером X до строки с номером Y. Параметр H предоставляет число строк, подлежащих выдаче для листинга до наступления паузы. По умолчанию: X = начало, Y = конец, H = 14.

M выполняет вход в программу монитора.

N X создает новый незаполненный файл, начиная с адреса X.
По умолчанию: X = 32 768 (#8000).

O X эта команда приводит к тому, что "старый" исходный файл, расположенный с адреса X, становится текущим исходным файлом.
По умолчанию: X = 32 768 (#8000)

P X работа устройства печати. Параметр X=1 включает устройство печати. Вся выводимая на дисплей информация выводится также на устройство печати ZX PRINTER. По умолчанию X=0.

Q укороченное имя команды QUIT (выход). Выполняется выход в систему BASIC. Для того, чтобы вернуться в ассемблер ZEUS, введите
PRINT USR 57344

R X Y Z перенумеровывается исходный файл. Параметрами являются:
X=первый номер строки
Y=промежуток между последовательными новыми номерами строк;
Z=номер строки, с которой начинается перенумерование.
По умолчанию: X = 10, Y = 10, Z = наименьший текущий номер строки

S x распечатывает таблицу символов. Параметр X предоставляет число символов/меток и связанные с ними шестнадцатиричные значения, выдаваемые в листинге до наступления паузы. По умолчанию: x=15.

T выводит на дисплей начальный адрес и длину, выражения в байтах текущего файла с текстом. Файл может быть сохранен при возврате в систему BASIC (с помощью команды Q) и при использовании команды:

```
 SAVE  "имя файла" CODE начало, длина
```

Для того, чтобы перезагрузить старый исходный файл по тому же самому адресу, с которого он был сохранен используйте команду:

```
 LOAD "имя файла" CODE,
```

по другому адресу, используйте команду:

```
 LOAD "имя файла", начальный адрес.
```

После загрузки старого исходного файла выполните повторный вход в ассемблер ZEUS (команда PRINT USR 57344) и введите команду O.

X выполняет машинный код, полученный при последнем ассемблировании.

Команды монитора

A X эта команда выводит на дисплей как десятичного так и шестнадцатиричного представления величины X, которая может быть представлена в любом виде, например: A 59 напечатает HEX=#3B DESIMAL=59 (шестнадцатиричное представление – 3B, десятичное представление – 59).

Y Z эта команда копирует блок памяти. В команде пользуются 3 параметра:
X - начальный адрес блока, который будет копироваться.
Y - начальный адрес блока,в который будет выполнено копирование.
Z - число копируемых байтов.
эта команда выполняет машинный код,начиная с адреса X.
эта команда распечатывает значение X в порте ввода-вывода.

Y эта команда позволяет изменить цвета границы, бумаги и печатной краски. Параметр X представляет байт, используемый для заполнения таблицы атрибутов: он состоит из четырех полей.
Биты 0-2 управляют цветом краски;
биты 3-5 управляют цветом бумаги;
бит 6 устанавливается для "яркого" цвета;
бит 7 устанавливается для "сверкающего" цвета.
Параметр Y представляет требуемый цвет границы,например:
K 5 1 - белая краска на голубом фоне с голубой границей.
K 38 7- черная краска на белом фоне с белой границей.
выполняется вход в реж.модификации, начиная с адр. X. Эта команда модификации представляет крайне гибкое и мощное средство работы с памятью, например: M 6000 напечатает
6000 XX (отмечена позиция маркера)

параметр XX представляет настоящее содержимое ячейки с адресом 6000. Вы можете теперь продолжить работу четырьмя способами:

1. Изменить содержимое по этому адресу. Напечатайте новое значение над старым и нажмите клавишу ENTER. Команда модификации напечатает следующий адрес-6001 вместе с настоящим содержимым.
2. Изменить содержимое по этому адресу и последующим одновременно. Напечатайте новое значение для настоящего адреса и вслед за ним значения, подлежащие записи. При нажатии клавиши ENTER команда модификации напечатает адрес и значение байта, расположенного вслед за последним модифицированным байтом.
3. Изменить адрес модификации. Напечатайте '/' и вслед за этим новый адрес. При нажатии клавиши ENTER действие команды модификации переместится к новому адресу.
4. Выйти из режима модификации. Напечатайте полностью команду останова и нажмите клавишу ENTER.

Y выполняет вывод значения параметра Y в порт X ввода-вывода.
устанавливает точки останова меток табуляции TAB. Любой отличный от пробела знак в той же строке, что и знак 'S', устанавливает точку остановки метки TAB (при условии, что строка введена). Клавиша CAPS SHIFTED '2' перемещает маркер между точками остановки, например:
S I I

Y устанавливает точки остановки метки TAB в колонках, занимаемых знаком 'I'.

Y табулирует память, начиная с адреса X. Восемь байтов памяти выводится на дисплей за один раз в шестнадцатиричном виде, каждая строка имеет в качестве префикса адрес первого байта в этой строке. Табуляция прекращается после того, как Y строк были выведены на дисплей. Для продолжения нажмите клавишу ENTER, нажатие любой другой клавиши возвратит вас в обычный режим (режим команды).
Отметим, что модифицировать память можно следующим способом: выполнить табуляцию требуемой области описанным выше способом и после возвращения в режим команд ввести режим модификации (команда 'M'). Используя клавиши управления маркером, переместить маркер наверх в листинг, полученный после выполнения табуляции. После изменения всех требуемых значений в данной строке нажать клавишу ENTER для модификации действительных ячеек памяти. Выход из режима модификации выполняется также, как и перед этим с помощью печати полной команды останова и нажатия клавиши "ENTER".
возвращает управление ассемблеру.

летим, что все, вводимые в программу монитор константы должны быть гавлены в шестнадцатиричном виде (за исключением констант вводимых в ко-

манде 'A'.

Дизассемблер ZEUS

После загрузки программы в память следует ввести: PRINT USR 29369(версия 16к) или PRINT USR 62137(версия 48к).

Список команд:

символ	команда	параметры	действие
A	-ASSING	регистр=значение	Изменение содержимого регистров
B	-BREAK	адрес	Устанавливает точку останова
C	-COPY	начальный адрес 1, конечный адрес 1, начальный адрес 2.	Перенос группы байтов из 1 ячейки памяти в другую
D	-DASASSEMBLE	адрес	Обратное ассемблирование
E	-EDIT	адрес	Редактирование. На экране 20 адресов и их содержимое.
F	-FIND	начальный адрес, Конечный адрес, Искомая цепочка символов	Поиск цепочки символов в интервале
G	-GOTO	адрес	переход по адресу
J	-JUMP RELATIVE	адрес с, Адрес в.	Вычисление относительного смещения
K	-CONVERT	число	Преобразование 16-ричного в 10-ричное и наоборот
M	-MESSAGE	"сообщение" адрес	Для вставки цепочки символов по адресу
P	-PRINT	адрес	Печать содержимого 64 адресов памяти в символьном виде
R	-REGISTERS	без параметров	Вывод содержимого регистров
S	-SUBSTITUTE	"имя" начальный адрес, конечный адрес, байт 1, байт 2.	Байт 1 заменяется на байт 2 в заданном интервале
T	-TABULATE	адрес	печать содержимого 64 адресов памяти в HEX виде
V	-VERIFY	начальный адрес 1, конечный адрес 1, начальный адрес 2.	Сравнение 2-х областей памяти
X	-EXIT	без параметров	Возврат в бейсик
Z	-ZERO	начальный адрес, конечный адрес	Обнуление

Редактор экрана

'CAPS SHIFTED'/1 очищает строку, содержащую маркер ("редактирование")
'CAPS SHIFTET'/2 перемещает маркер в следующую точку ("блокировка прописных букв")
'CAPS SHIFTED'/3 удаляет знак, расположенный под маркером. ("Прямое изображение")
'CAPS SHIFTED'/4 вставляет пробел в позицию маркера. ("Обратн. изображение")
'CAPS SHIFTED'/5,6,7,8 перемещение маркера.
'CAPSHIFTED'/9 очищает экран и возвращает маркер "домой".

Коды сообщений об ошибках

0 Недопустимый знак или неразрешенное утверждение.
1 Слишком длинная метка.
2 Ожидаемый символ ')'.
3 Ошибка усечения или выход за пределы диапазона.
4 Ожидаемая ','.
5 Ошибка в контексте.
6 Повторно определенная метка.
7 Ожидаемый символ '('.
8 Недопустимое мнемоническое обозначение.
9 Неопределенная метка.

Резервируемые слова

Следующий список представляет собой список резервируемых слов, т.е. тех слов, которые не должны использоваться в качестве меток, хотя любая метка может их содержать.

A	ADC	ADD	AF'	AF	AND	B	BC	BIT	C	CALL
CCF	CP	CPO	CPDR	CPI	CPIR	CPL	I	DAA	DE	DEC
DEFB	DEFM	DEFS	DEFW	DL	DISP	DJNZ	E	EI	ENT	EQU
EX	EXX	H	HALT	HL	I	IM	IN	INC	IND	INDR
INI	INIR	IX	IY	JP	JR	L	LD	LDD	LDDR	LDI
LDIR	M	NC	NEG	NOP	NV	NZ	OR	ORG	OTDR	OTIR
OUT	OUTD	OUTI	P	PE	PO	POP	PUSH	RES	RET	RETI
RETN	RL	RLA	RLC	RLCA	RLD	RR	RRA	RRC	RRCA	RRD
RST	SBC	SCF	SET	SLA	SP	SRA	SRL	SUB	V	XOR

План памяти системы ZEUS

Системой ZEUS используются следующие области памяти

A. Машинный код ZEUS и машинная память — 57 344-65279 (#E000- #FF00)

B. Исходный файл — первоначально начинается с адреса 32 768(#8000), но может быть изменен командой 'N'.

C. Ассемблированный код (объектный код) — определяется пользователем через директивы ORG и DISP.

D. Таблица символов — это таблица, которая содержит алфавитно-цифровые символы и метки, определенные в исходном файле, вместе со связанными с ними значениями. Таблица начинается с адреса 57 343 и увеличивается в размере вниз со скоростью 16 байтов на одну используемую метку.

Стандартные программы

Имя прог-мы	Адрес	Функция
INPUTCHAR	#F652	Ожидает нажатия клавиши; результат помещается в резервируемое слово A
PRINTCHAR	#F503	Печатает знак, находящийся в A. Может также являться управляющим знаком (см. ниже). Если флаг устройства печати не равен нулю, то выполняется также вывод на устройство печати.
PRINTDECIMAL	#F5A3	Записывает резервируемое слово NL в десятичном виде и за ним пробел.
PRINTNEXBYTE	#F2DF	Записывает A в шестнадцатиричном виде и вслед за ним пробел.
PRINTSTPING	#E4e3	Записывает цепочку знаков, следующую завызовом подпрограммы; цепочка должна завершаться нулевым байтом.
INPUTLINE	#F6E2	Выполняет вход в редактор экрана; помещаетстроку, имеющую маркер, в буфер.
BUFFER	#FE00	Буфер из 32 знаков, оканчивающихся нулевым байтом.
PRINTERFLAG	#F4CB	Не ноль, для того, чтобы допустить работу устройства печати.
PRINTDECNZ	#E59e	Печатает десятичное число в NL без ведущих нулей.

Список управляющих знаков редактора:

#04 удааляет знак, находящийся под маркером.
#05 вставляет пробел в позицию маркера.
#06 перемещает маркер в следующую позицию табуляции.
#07 очищает строку, содержащую маркер.
#08 перемещает маркер влево.
#09 перемещает маркер вправо.
#0A перемещает маркер вниз.
#0B перемещает маркер вверх.
#0C перемещает маркер влево и знак пробела под маркер.
#0D перемещает маркер в начало следующей строки."Прокручивает" строки, если маркер на строке, расположенной снизу экрана.
#0F очищает экран и возвращает маркер на свое место.

Программа ЛАЗЕР - БЕЙСИК

Разработка фирмы OASIS SOFTWARE, автор - Кевин Хэмбитом

Лазер-бейсик разработан для обеспечения легкости и и простоты построения сложных движущихся графических обьектов.

Загрузить лазер-бейсик можно по команде:

LOAD * "M";1;"LAZER"

Если происходит ошибка, то нажмите "N" и "ENTER", или RANDOMIZE USR 58830 и ENTER или RANDOMIZE USR 58820

Терминология:

1. Спрайты-управляемый графический обьект.
2. Окна-часть спрайта, заданная 4-мя переменными: .COL, .ROW, .HGT, .LEN.
3. Окна спрайтов- часть спрайта, заданная перемекными: .SCL,.SFN,.LEN,.SPN.
4. Область спрайтов - область памяти,где хранятся все заданные спрайты. Ее верхняя граница-56575, а нижняя: PRINT PEEK (62464)+ 256*PEEK(62465), или LET X=PEEK (62464):PRINT X

Граница RAMTOP: PRINT PEEK (23730)+256*PEEK (23731) или LET X=PEEK (23730): PRINT X. При каждом задании спрайта расходуется 9*HGT*LEN+5 байт.

5. Пиксель -элементарный блок в один символ 8x8 точек.
6. Атрибуты -цвета INK и PAPER, параметры BRIGHT и FLASH.
7. Операции с экраном - все команды имеют в конце букву "V". Например: .SRIV,.INVV,.RV и пр.
8. Операции между экраном и спрайтом: команды этой группы начинаются с "PT" или "GT" - .GTBL,.PTHR.
9. Операции со спрайтами-имеют окончание "M".
10. Операции спрайт - спрайт: только 2 команды: .SFNM и . DSFM.
12. Пустой спрайт: не содержит данных для экрана.
13. Сервисные команды:

Перенумерация строк: .RNUM X,Y,Z ,где
X - начальный адрес строки (по умолчанию =0)
Y - новый адрес (=старому)
Z - шаг (10)
.REMK - удаление всех REM-строк.
.TRON - начало трейсинга (пошаговое выполнение)
.TROF - конец трейсинга.

14. Графические переменные:

-ROW - вертикальная координата (от 0 до 23). Верхний ряд имеет ROW=0.
-COL - горизонтальная координата (от 0 до 31).Левый столбец имеет COL=0.
-HGT - высота текущего окна (1...24).
-LEN - длина текущего окна (1...32).
-SRW - вертикальная координата внутри спрайта (от 0 до HGT-1).
-SCL - горизонтальная координата внутри спрайта (от 0 до LEN-1).
-NPX - выражает величину и направление вертикального скроллинга. Положительное значение-вверх, отрицательное-вниз. Единица измерения-пиксели (H-символы). Диапазон-(+127...-128).
-SPN - номер спрайта (от 1 до 255).Применяется для тех команд, которые работают с одним спрайтом.
-SP1 - операции между спрайтом и окном другого спрайта. Здесь содержится номер первого спрайта, имеющего окно.

Пример:

```
10 .SET=0: .SPN=4: .ROW=0: .COL=-4
20 BORDER 1: BRIGHT 1: INKG: PAPER 1: CLS: .ATOF
30 FOR I=-4 to 32
40 .PTXR: .COL=I+1: .PTXR
50 PAUSE 4
60 NEXT I
70 STOP
```

Строка 10 - выбирается графический набор 0. Спрайт номер 4: ROW устанавливается в 0, а COL в - 4.

Строка 20 - устанавливает атрибуты экрана, экран очищается. Флаг атрибутов отключается.

Строка 30 - организует цикл для движения спрайта 4 (утка) по горизонтали от координаты (-4) до (+32).
Строка 40 - удаляет старый спрайт, затем помещает новый.
Строка 50 - пауза для обеспечения необходимой плавности перемещения.

15. Изменение значений переменных.

Для изменения значений переменных существуют 11 функций:

?COL	?ROW	?LEN	?HGT	?SPI	?SP
?SPN	?NPX	?SCL	?SRW	?SET	

Например:

```
LET X=?COL*3+ROW/8  -  нельзя.
```

Вместо этого:

```
LET COL=?COL" LET Y=?ROW: LET X=COL*3+4/8
```

Подробное описание команд ЛАЗЕР-БЕЙСИКА.

1. Процедуры обработки спрайтов.

.SPRT - для создания новых спрайтов. Начало области остается неизменным, а верхняя граница поднимается на величину: 9* HGT*LEN+5 байтов

.ISPR все то же самое, что и .SPRT, но здесь области спрайтов развиваются вниз.

.WSPR служит для стирания соответствующего спрайта, верхняя граница области понижается.

.DSPN аналогична предыдущей. При стирании спрайтов нижняя граница области повышается.

Параметры : SPN,LEN,HGT

2. Горизонтальный "скроллинг экрана

Горизонтальный скроллинг выполняется на 1, 4 или 8 пикселей влево или вправо, с возвратом или без него.

WL1V,WL4V,WL8V - влево на 1,4,8 пикселей с возвратом
WR1V,WR4V,WR8V - вправо на 1,4,8 пиксель с возвратом
SL1V,SL4V,SL8V - влево ..... Без возврата
SR1V,SR4V,SR8V - вправо..... Без возврата

Параметры команд: COL,ROW,LEN,HGT

3. Вертикальный "скроллинг" экрана

Как и горизонтальный скроллинг, к параметрам окна COL, ROW, HGT, LEN добавляется переменная NPX, которая задает размер и направление скроллинга в пикселях

.NPX=-1 - на один пиксель вниз
.NPX=1 - на один пиксель вверх

Команды	действия
.WCRV	вертикальный скроллинг с возвратом
.SCRV	вертикальный скроллинг без возврата

Проверка достаточности емкости буфера:

```
LET X=?NPX: LET Y=?LEN: IF ABS (X) *Y <256 THEN WCRV
```

4. Скроллинг аттрибутов экрана.

Скроллинг атрибутов экрана похож на скроллинг пикселей, но выполняется на ширину символа и всегда с возвратом.

Команда	действие
.ATLV	скроллинг аттрибутов влево
.ATRV	скроллинг аттрибутов вправо
.ATUV	скроллинг аттрибутов вверх
.ATDV	скроллинг аттрибутов вниз

Параметры: COL,HGT,ROW,LEN

5. Горизонтальный скроллинг спрайтов.

Выполняется как и скроллинг экрана. Все команды отличаются последней буквой (вместо "V"-"M": .WL1M,.WR1M)

Параметры SPN - намер спрайта.

6. Вертикальный скроллинг спрайтов.

Работа аналогична вертикальному скроллингу экрана.

.WCRM вертикальный скроллинг с возвратом
.SCRM вертикальный скроллинг без возврата

Параметры: SPN,NPX

7. Скроллинг аттрибутов спрайта.

.ATLM влево
.ATRM вправо
.ATUM вверх
.ATDM вниз

Параметр SPN - номер спрайта (1-255)

8. Операторы GET и PUT,группа 1.

Операторы PUT помещают спрайт на экран или в другой спрайт, GET - выполняет противоположное действие - берут данные с экрана или из спрайта и помещают их в спрайт.

Команда	действие
.GTEL	поместить блок из экрана в спрайт
.GTOR	наложить по OR (или) блок экрана на спрайт
.GTXR	наложить по XOR (искл.или) блок экрана в спрайт
.GTND	наложить по AND (и) блок экрана в спрайт
.PTBL	поместить спрайт в окно экрана
.PTOR	наложить по OR спрайт на окно экрана
.PTXR	то же по принцепу XOR
.PTND	то же по принцепу AND

Параметры: COL,ROW,SPN

Например: .SPN=39: .ROW=1: .COL=1 .PTBL - помещает спрайт номер 39 на экран в позицию (1,1).

9. Операторы GET и PUT , группа 2.

Эти операторы выполняют операции между окнами экрана и окнами спрайтов.

.GWEL - перенос блока с экрана в окно спрайта
.GWOR - наложение по OR
.GWXR - наложение по XOR
.GWND - наложение по AND
.PWEL - перенос блока из окна спрайта на экран
.PWOR - наложение по OR
.PWXR - наложение по XOR
.PWND - наложение по AND
.GWAT - перенос атрибутов блока с экрана в окно спрайта
.PWAT - перенос атрибутов блока из окна спрайта на экран

Параметры: COL,ROW,SCL,SRW,HGT,LEN,SPN

10. Операторы GET и PUT ,группа 3.

Эта группа, по-видимому, наиболее широко применяемая. Содержит команды для операций между спрайтом и окном в другом спрайте. Эти команды начинаются с "PM" и "GM".

.GMEL - перенос спрайта в окно другого спрайта
.GMOR - наложение по принцепу OR
.GMXR - то же по принцепу XOR
.PMND - то же по принцепу AND
.PMBL - перенос окна спрайта в другой спрайт
.PMOR - наложение по принцепу OR
.PMXR - то же по XOR
.PMND - то же по AND
.SMAT - перенос атрибутики спрайта в окно другого спрайта
.Pмат - перенос атрибутики окна спрайта в другой спрайт.

Параметры: SCL,SRW,
SP1-номер первого спрайта (1...255)
SP2-номер второго спрайта (1...255)

11. Оператор .MOVE

Используется для эффекта мультиплицированного перемещения спрайтов.

Параметры: COL,ROW,HGT,LEN,SP1,SP2

По команде MOVE берется через PUT спрайт SP1 с позиции экрана, заданной ROW и COL и помещается спрайт SP2 в позицию COL+HGT+ROW .

12. Операторы .ATON и .ATOF .

Эти две команды определяют, надо ли переносить атрибуты спрайтов при выполнении команд PUT и GET .

.ATON - включение переноса аттрибутов
.ATOF - выключение

Изменение изображения.

.INVV - инвертирование окна экрана. Параметры для окна: COL,ROW,HGT,LEN .
.INVM - аналогичная над спрайтом с номером SPN
.MIRV - содержимое окна зеркально отображается относительно вертикальной оси, проходящей через его центр. Параметры: COL, ROW,HGT,LEN
.MIRM - аналогично для спрайта с номером SPN
.MARV - то же для спрайта SPN
.SFNM - поворот спрайта SP2 на 90 градусов по часовой стрелке. После поворота присваивается номер SP1. Параметры : SP1,SP2 .
.DSPM - увеличение размеров спрайта SP2 в 2 раза.

П р о ч и е к о м а н д ы :

.SETV - установка атрибутов INK и PAPER в окне.Параметры : COL,ROW,HGT,LEN.
.SETM - то же,но для спрайта,заданного в SPN. Параметр : SPN .
.CLSV - очистка заданного окна экрана.
.CLSM - то же,но для спрайта. Параметр : SPN .
.ADJM - для настройки переменных COL, ROW, HGT, SCL, SRW,SPN т.о., чтобы отдельный спрайт мог быть взят командами PUT или GET 2-ой группы.
Параметры : SPN, COL, ROW, SPN, SRW.
.ADJV - то же,что и в предыдущей команде для окна. Параметры: COL, ROW, HGT, LEN.

П р и с в а и в а н и е .

Эти команды присваивают графическим переменным значения бейсиковских выражений.

.COL= .ROW= .HGT= .LEN= .SP1= .SP2=
.SPN= .SRW= .SCL= .NPX= .SET=

Д о п о л н и т е л ь н ы е ф у н к и и .

?COL ?ROW ?HGT ?LEN ?SPN ?SP1 ?SP2 ?SET
?SCL ?SRW ?NPX ?KEF ?SCV ?SCM ?TST ?PEK

Их форма:
LET VAR=?FUN , где:
- VAR - обычная бейсиковская переменная;
- ?FUN - одна из 16-ти функций.

Эти функции не должны быть частью бейсиковских выражений.
- ?COL - выдает значение переменной COL .

А н а л о г и ч н о : ?ROW. ?HGT. ?LEN. ?SPN.
?SP1. ?SP2. ?SCL. ?SRW.
?SET. ?NPX.

?KEF - для определения нажатия клавиш. Переменные ROW и COL для задания полуряда и столбца на клавиатуре.

Р я д	к л а в и ш и
1	от CAPS SHIFT до V
2	от A до G
3	от Q до T
4	от 1 до 5
5	от 0 до 6
6	от P до V
7	от ENTER до H
8	от SPACE до B

?SCV - элементарный блок экрана (8x8), заданный COL и ROW проверяется на предмет включения пикселей.
?SCM - проверяются пиксели спрайта,номер которого содержится в SPN.
?TST - отыскивается спрайт,номер которого хранится в SPN.
П р и м е ч а н и е: спрайты хранятся в памяти в следующем формате:
Байт1 - номер спрайта.
Байт2 - младший байт адреса следующего спрайта.
Байт3 - старший байт адреса следующего спрайта.
Байт4 - длина спрайта.
Байт5 - высота спрайта.
8*HGT*LEN = данные о состоянии пикселей.
HGT*LEN = аттрибуты.
Итого: 9*HGT*LEN+5 байтов.
?PEK (адр.) - Это 16-битная версия оператора PEEK.
Например: LET X=?PEK (64264) эквивалентно :

```
LET X=PEEK*(64264)+256*PEEK (64265).
```

Дополнительные команды

POKE X,V - это 16-битная версия обычного бейсиковского POKE. Она помещает младший байт числа V в X, а старший байт V в X+1.
Диапазон V - (0...65535). Диапазон X - (0...65534)

Процедуры:

При задании процедуры ее описание содержит список параметров, например:

```
10 DEF FN A*(X, Y, Z, A&,B&)
```

здесь параметры X, Y, Z, A&, B& - локальные переменные.

Вызов процедур.

Для выполнения приведенной выше процедуры используется оператор:

```
.PROC FN A*(3,2,K,"HELLO",E&)
```

тогда локальные переменные примут значения :

```
X=3        Y=2        Z=K
A&="HELLO"            B&=E&
```

Процедура должна всегда заканчиваться командой .RETN.
Разрешается применять до 52 процедур (26 букв латинского алфавита +26 букв со знаком &).

Движение спрайтов.

Выполним скроллинг окна на один пиксель, используя функцию ?KBF.

```
10 INK 6; PARER 0; BRIGHT 1; BORDER 0; CLS
20 .COL=14; .ROW=10; .SPN=2; .PTEL
30 .SET=5; .COL=5; .ROW=4
40 .SET=6; .COL=3; .ROW=5
50 .SET=7; .COL=0; .ROW=5
60 .SET=5; LET KB=?KBF; IF KB <>0 THEN .SET=7; .WL1V
70 .SET=6; LET KB=?KBF; IF KB <>0 THEN .SET=7; .WL1V
80 GO TO 60
```

Использование PUT

Возьмем спрайт 30 размером 3x3. Сначала очистим данные в спрайте:

```
.SPN=30; .CLSM
```

Теперь поместим в его центр спрайт 51 (размер 1x1): .SP1=5 1; .SP2=30; .SCL=1; .SPW=1; .GMBL
Программа для его перемещения с помощью курсорных клавиш (нажатие их проверяется ?KBF) будет выглядеть так:

```
10 .ATOF;INK 5;BRIGHT 1;PAPER 0;BORDER 0;CLS
20 .SET=1;ROW=4; .COL=5
30 .SET=2;ROW=5; .COL=5
40 .SET=3;ROW=5; .COL=4
50 .SET=4;ROW=5; .COL=3
60 .SET=7;ROW=10; .COL=10;SPN=30
70 .LET x=13; LET V=10
80 .SET=1;LET KB=?KBF; IF KB <> 0 THEN LET X=X-1
90 .SET=2;LET KB=?KBF; IF KB <> 0 THEN LET Y=Y+1
100 .SET=3;LET KB=?KBF; IF KB <> 0 THEN LET Y=Y-1
110 .SET=4;LET KB=?KBF; IF KB <> 0 THEN LET X=X+1
120 IF X>30 THEN LET X=30
130 IF X>-1 THEN LET X=-1
140 IF Y>22 THEN LET Y=22
150 IF Y>-1 THEN LET Y=-1
160 .SET=7; .COL=X; .ROW=V;PTBL
170 GO TO 80
```

Использование команды .MOVE

После того, как заданы значения COL и ROW и задано их приращение на каждом шаге HGT и LEN. Каждое испол нение команды .MOVE приводит к увеличению COL и ROW .

```
10 INK 0;PAPER 6;BRIGHT 1;BORDER 7;CLS; .ATOF
20 FOR N=1 TO 100
30 LET X=INT(FND*32);LET V=INT(FND22)
40 PRINT AT X,V;"*"
50 NEXT N
60 LET OR=1;HGT=OR;LET DC=1; .LEN=DC
```

```
 70 .SP1=50;SP2=50;LET R=10; .ROW=R;LET C=13;COL=C;.SPN=50: .PTXR
 80 .MOVE
 90 LET C=?COL;LET R=?ROW
100 IF R=19 OR R=0 THEN LET DR=DR-1; .HGT=DR
110 BEEP 0.01,5
120 GO TO 80
```

Помещение спрайта на экран с повышенной точностью.

Если вам надо переместить спрайт слева направо с шагом по 2 пикселя, а скроллинг экрана при этом делать нельзя, то нужно сначала с помощью программы "генератор спрайтов" создать 4 спрайта, каждый из которых сдвинут относительно предыдущего вправо на 2 пикселя.

Если их пронумеровать от 1 до 4, то программа будет выглядеть так:

```
 10 FOR C=0 TO 31
 20 .COL=C; .SPN=1; .PTBL; .SPN=2; .PTBL
 30 .SPN=3; .PTBL; .SPN=4; .PTBL
 40 NEXT C
```

Определение столкновений спрайтов.

Для этого служат команды: ?SCV и ?SCM.

?SCV - служит для проверки позиции символа на экране.

?SCM - команда проверяет спрайт с номером SPN на наличие включенных пикселей.

Наборы переменных.

Если вы хотите перемещать 4 окна на экране, вы можете задать параметры для каждого окна в виде набора переменных, например:

```
.SET=1;  .HGT=5;  .LEN=5;  .ROW=0;  .COL=0
.SET=2;  .HGT=4;  .LEN=6;  .ROW=0;  .COL=5
.SET=3;  .HGT=6;  .LEN=4;  .ROW=0;  .COL=12
.SET=4;  .HGT=7;  .LEN=3;  .ROW=0;  .COL=18
```

Теперь для того, чтобы выполнить скроллинг для каждого окна достаточно задать:

```
.SET=1; .WR1V; .SET=2; .WL2V; .SET=3; .WL8V; .SET=4; .WR1V
```

Загрузка и запись программ в ЛАЗЕР-БЕЙСИКЕ.

Загрузка и запись прямой командой.

Чтобы записать неавтостартующую программу напрямую используйте:

- для ленты: SAVE "FILENAME"
- для микродрайва: SAVE * "M";N; "FILENAME"

Запись автостартующей программы прямой командой

- SAVE "FILENAME" LINE N
- SAVE * "M"; N; "FILENAME" LINE N

Программа "LASER BASIC COMPILER"

(компилятор Лазер-бейсика)

Разработка фирмы OASIS SOFTWARE, автор - Крис Меллинг

Введение

Компилятор языка Лазер-бейсик предназначен для создания автономной программы независимо от интерпретатора Лазер-бейсика.

Любая программа, созданная в расширенном интерпретаторе Лазер-бейсика поместится в компиляторе.

Расположение файлов на ленте.

```
A.COMPCODE      LOAD ""CODE
  LOADER        LOAD ""
  RT CODE       LOAD "RT CODE" CODE
B.DEMO          LOAD ""
```

Компиляция

Программу следует записать на ленту, сбросить компьютер и загрузить в компилятор только свою программу без интерпретатора.

Перед загрузкой выполнить: CLEAR 59799, загружается компилятор: LOAD ""CODE или LOAD "COMPCODE" CODE

Начало компиляции - RANDOMIZE USR 59800

При компиляции экран изображает случайную информацию. Это не сбой, все в по-

рядке. После окончании компиляции экран очищается. Появляется сообщение "O.K.", а также номер последней откомпилированной строки.

Скомпилированный код вытирает исходный текст. Тем не менее компьютер воспринимает его как обычную бейсик-программу.

Выполнение откомпилированной программы

Для выполнения откомпилированной программы необходимо снабдить ее дополнительно пакетом рабочих процедур. RAMTOP следует установить 59799 (не забывайте об этом при загрузке откомпилированных модулей). Пакет рабочих процедур загружается командой:

```
LOAD "RT CODE" CODE
```

Если программа имеет файл SPRITES, его следует загрузить точно также, как и в программе, выполненной на языке Лазер-бейсик.

Инициализация выполнения программы:

```
RANDOMIZE USR 59800  или  GO TO 1
```

Компоновка автономной программы.

Программу, написанную на Лазер-бейсике, необходимо соответственно настроить, чтобы для работы было достаточно нажать "LOAD".

Для этого программу необходимо скомпоновать и снабдить загрузчиком. Он входит в набор, имеющийся на исходной ленте. Вот его текст :

```
10 CLEAR 59799
20 LET SPST=0
30 LOAD "RTCODE" CODE
40 LOAD "SPRITES" CODE SPST
50 LET T=INT(SPST/256):POKE 62464,SPST-256*T:POKE 62456,T
60 POKE 62466,255: POKE 62467,200
70 POKE 56575,0
80 LOAD " "
```

Здесь:

строка 10-выделение пространства для пакета рабочих процедур.

Строка 20-установка начала области,размещения файла SPRITES (смотри описание генератора спрайтов)

строка 30-загрузка пакета рабочих процедур

строка 40-загрузка файла спрайтов

строка 50-внесенне в программу адреса начала области размещения спрайтов.

Строка 60-внесенне в программу адреса конца размещения области спрайтов.

Строка 70-внесения конца области размещения знакогенератора спрайтов

строка 80-загрузка текста скомпилированной программы.

Примечание: спрайты в программе не используются, строки 20-70 можно исключить.

Вам остается внести требуемое значение SPST и подготовить набор файлов на чистой ленте. Помните, что делая SAVE для загрузчика и для откомпилированной программы необходимо указывать строку автостарта.

```
SAVE "LOADER" LINE 10
SAVE "COMPPROG NAME" LINE 1
```

Ограниченне для стандартного бейсика. Нижеперечисленные ограниченния были заложены в компилятор на этапе его проектирования.

1. Нельзя использовать вычисляемые выражения в операторах GO TO, GO SUB, RUN, RESTORE

2. В операторе DIM допустимы вычисляемые границы, но только такие, которые в процессе компиляции будут установлены в 0.

3. Использование оператора CLEAR не допускается.

4. Команда RUN выполняется также, как и в бейсике, но сброса переменных не производится.

5. Недопустимы операторы: MERGE,CONTINUE,LIST,LLIST.

6. Командой CONTINUE нельзя запускать сбившуюся программу или программу с оператором STOP.

7. Команды OPEN# и CLOSE# работают только с потоками S,K и P.

8. Команды LOAD, SAVE и VERIFY являются допустимыми если загружается бейсик-программа.

Сообщения об ошибках:

ILLIGAL STATEMENT FOUND - встретился один из недопустимых операторов: CLEAR, MERGE,LIST,LLIST,CONTINUE.

PROCEDURE DEFINITION NESTING ERROR - найдено два описания процедуры без проме-

жуточного оператора .RETN. В описаниях процедур вложенность не допускается.

RETN WITHOUT DEF FN# ERROR – найден оператор .RETN без соответствующего описания процедуры.

PROCEDURE NOT FOUND – обращение к неописанной процедуре. Описание процедуры не обязательно должно предшествовать ее исполнению.

PROGRAM NOT COPILED – это сообщение пакета рабочих процедур. При старте программы ее начало не было идентифицировано как начало откомпилированной программы.

Карта памяти.

Компилятор и пакет управляющих процедур загружается начиная с адреса 59800. Пакет рабочих процедур занимает всю верхнюю часть памяти. Конец компилятора по адресу 62586, т.е. его длина 3057 байтов. Следовательно перед загрузкой этих модулей необходимо выполнить:

CLEAR 59799

Откомпилированная программа размещается в памяти следующим образом:

программа:	переменные:	спрайты:
PROG	VARS	ELINE
стек	микропроцессора	RT CODE
65675	59800	65535

Программа SPRITE GENERATOR

(генератор спрайтов)

Разработка фирмы OASIS SOFTWARE, автор – Пол Ньюхем

Вступление

Генератор спрайтов предназначен для создания и редактирования спрайтов.

Загрузка

Программа загружается LOAD "SPTGEN" или LOAD " ". Если вы хотите записать ее на ленту, сделайте BREAK, затем GO TO 999. Обе части программы будут выгружены на ленту. Правда теперь чтобы запустить программу, ее надо перезагрузить.

Чтобы записать программу на микродрайвер:
вместо LOAD " " CODE поставьте
LOAD * "M";1;"G" CODE,
теперь дайте GO TO 9998

Начало работы:

После загрузки программы вы получите запрос: Какой запуск? Холодный (COLD) или теплый (WARM). В 1-ый раз всегда выбирается "C"(COLD), затем дается Y(YES), появляется рабочий экран.

Терминология:

" Холодный запуск "

Все спрайты имеющиеся в программе вычищаются, системные переменные устанавливаются в исходное положение.

" Теплый запуск "

При таком входе в программу все спрайты сохраняются и системные переменные не изменяются.

CHR& SQR

Эта аббревиатура используется для обозначения элементарного квадрата и относится к сетке 8x8, расположенной в левой части экрана. Здесь создаются и редактируются спрайты.

Экран спрайта.

Это область экрана размером 15x15 символов.

Курсор CHR& SQR

Это мигающий курсор, который используется для конструирования и редактирования изображения находящегося в CHR& SQR.

Курсоры экрана спрайта.

Это два мигающих курсора. Они расположены ниже и правее экрана спрайтов. Применяются для задания позиции левого верхнего угла окна с которым идет работа.

Окно экрана спрайтов.

Часть экрана, с которым непосредственно идет работа – это окно. Его координаты задаются XPOS и YPOS.

Спрайты.

Для записи спрайтов на ленту вы имеете две различных операции:
Операция 1: этот режим предполагает дальнейшее редактирование спрайтов.
Операция 2: этот режим выгружает спрайты готовые к использованию в лазер-бейсике.

Для тех, кто не чувствует в себе художественных способностей, мы изготовили два набора спрайтов на прилагаемой кассете.
SPRITE 1A - набор из 50 спрайтов различных аркодных персонажей.
SPRITE 2A - это файл спрайтов входящих в демонстрационную программу.
В информационном прямоугольнике изображается следующая информация:

```
MEMORY LEFT 7488      XPOS 1        YPOS 1
SPRITE 60218          SPRITE        HEIGHT-2
SPST   60218          SPRITE        LENGHT-2
SPND   65279          SPRITE        NUMEER-1
          < текстовая строка >
```

MEMORY LEFT - обьем памяти оставшийся для размещения спрайтов.
XPOS YPOS - текущее положение курсоров экрана X и Y.
SPRITE - адрес памяти, где заданный вами спрайт.
SPST - начало области спрайтов в памяти компьютера.
SPND - указывает на конец области спрайтов.
SPRITE HEIGHT - обозначает высоту заданного спрайта.
SPRITE LENGHT - обозначает длину заданного спрайта.
SPRITE NUMBER - номер текущего спрайта
<текстовая строка> - выполняемая в данный момент команда.
A - переключение атрибутов (аналог .ATOF и .ATON) 1-вкл. 0-выкл.
B - задает переменную BRIGHT. 1-Вкл. 0-Выкл.
C - задает переменную PAPER. Клавиши от 0 до 7 - цвет.
SIMBOL SHIFT+C - создание больших спрайтов.
D - прямой ввод данных. Принимает 8 байтов данных по одному байту за один раз с последующим ENTER.
E - задействует функции экрана, имеет 3 операции:

1. Инверсия (аналогична INVV). По этой операции все выключенные пиксели включаются, а включенные выключаются. Действует на все окно, размеры которого заданы "SPRITE HEIGHT" и "SPRITE LENGHT".

2. Зеркальное отображение (аналогична.MIRV). Эта операция "переворачивает" относительно вертикальной оси окно, заданное теми же переменными, что и в операции 1.

3. Зеркальное отображение атрибутов (аналогична .MARV). Эта операция "переворачивает" относительно вертикальной оси атрибуты в окне, заданном теми же переменными, что и в операции 1.

F - задействует режим FLASH (мигание) окна. Обеспечивает мигание окна экрана, заданного переменными (SPRITE HEIGHT) и (SPRITE LENGHT).
SYMBOL SHIFT+1 - отформатировать картридж микродрайвера. Картридж подготавливается к записи спрайтов. Устанавливается 5 пустых спрайтов от 1 до 5.
G - занесение спрайта с экрана в память. По этой команде берется спрайт, размеры которого хранятся в "SPRITE NUMVER", снимается с окна, заданного позициями курсоров экрана спрайтов и отправляется в память компьютера.
H - ввод высоты спрайта "SPRITE HEIGHT" (аналогична .HGT) допускается высота от 1 до 15 символов.
SYMBOL SHIFT+T - скроллинг окна влево на 1 пиксель.
U - берет параметры атрибутов символа из экрана спрайтов с позиции, отмеченной курсорами и помещает их в 4 переменных атрибутов.
SYMVOL SHIFT+U -скроллинг окна вверх.
V - включение переменной FLASH 1-вкл. 0-выкл.
W - включение функции стирания спрайтов "WIPE SPRITE" (аналогично .DSPR).
X - задает переменную INK. Клавиши 0-7 цвет.
Y - дает положительный ответ "да" при запросе "Y/N".
SYMBOL SHIFT+Y - скроллинг окна вниз на 1 пиксель.
SPACE - дает возможность поместить спрайт небольших размеров внутрь второго, большего спрайта в позицию, заданную переменными ROW и COL второго спрайта.

Здесь существует четыре операции:

1. GETBLS - меньший спрайт помещается в большой.

2. GETORS - накладывается по "OR" меньший спрайт на больший.
3. GETXRS - то же, но по XOR.
4. GETND - то же, но по AND.
5,6,7,8 - перемещение курсора с CHR$ SQR в соответствующих направлениях.
9. - включает CHR$ SQR в выбранной позиции.
0. - выключает CHR$ SQR в выбранной позиции.
SYMVOL SHIFT+5,6,7,8 - управление перемещениями курсоров на экране спрайтов.

Р е з ю м е :

Итак,порядок создания спрайтов в генераторе спрайтов такой:
1. Загрузить генератор спрайтов и выполнить холодный запуск.
2. Установить клавишей "A" флаг атрибутов в 1.
3. Установить цвета INK и PAPER.
4. Создать спрайт, используя CHR$ SQR или прямой ввод данных.
5. Увеличить размеры окна до размеров своего спрайта.
6. Установить курсоры экрана спрайтов в левый верхний угол вашего спрата.
7. Высветить окно клавишей "F".
8. Загрузите спрайт в память клавишей "G".
9. Проверте, правильно ли прошла операция следующим образом: переведите курсор в свободную часть экрана спрайтов и поместите туда спрайт клавишей "P".
10. Выполните другие необходимые операции или создайте другие спрайты.
11. Выгрузите спрайты на ленту в формате операции 1 с тем, чтобы они могли быть загружены впоследствии в генератор спрайтов для доработки или редактирования.
12. Выгрузите спрайты на ленту в форме операции 2 для дальнейшего их использования в программе Лазер-бейсик.
13. Очистите память компьютера через RANDOMIZE USR 0.
14. Загрузите Лазер-бейсик.
15. Загрузите спрайты, используя операцию 2 главного меню. Загрузчик запросит от вас "SPRITE START ADRESS" (адрес начала области спрайтов), который вы запомнили, когда выполняли пункт 12.

B E T A - B A S I C

В в е д е н и е

BETA-BASIC версии 1.8 добавляет 30 новых команд и 21 новую функцию к уже имеющимся в ZX-SPECTRUM.

Г л а в а 1
Описание команд BETA-BASIC

ALTER <список атрибутов> TO список атрибутов [A]. Быстрая смена атрибутов экрана (INK,FLASH,BRIGHT, PAPER) без необходимости очистки экрана.

Пример: ALTER TO PAPER 1, INK 6 или
ALTER INK 7 TO PAPER 2,INK 0

AUTO <номер строки><,шаг> [6]
Включение автоматической нумерации строк. По умолчанию принимается шаг=10.

BREAK [SHIFT SPACE]
Останов выполнения программы.

CLOCK строка [C]
Задание параметров в часах-будильнике

примеры:

CLOCK "09:29:05"-установка точного времени в часах,
CLOCK "A06:20" -установка времени срабатывания будильника.
CLOCK аргумент [C]
Управление часами с помощью заданного аргумента.

	Переход на строку	звуковой сигнал	световой сигнал
CLOCK 0	откл	откл	откл
CLOCK 1	откл	откл	вкл
CLOCK 2	откл	вкл	откл
CLOCK 3	откл	вкл	вкл

```
        CLOCK 4          вкл          отк          откл
        CLOCK 5          вкл          от           вкл
        CLOCK 6          вкл          вк           откл
        CLOCK 7          вкл          вкл          вкл
```

CHR$ цифра

Знаки вставляемые в строку печатаемую оператором PRINT для изменения позиции печати.

CHR$ 8 - курсор влево;
CHR$ 9 - курсор вправо;
CHR$ 10 - курсор вниз;
CHR$ 11 - курсор вверх.

DEF KEY имя клавиши: строка SHIFT!

DEF KEY имя клавиши: инструкция: инструкция:...

После подачи этой команды нажатие указанной клавиши будет вызывать появление заданной строки или цепочки инструкций.

DEF PROG имя процедуры [1] ключевое слово обозначающее начало проц-ры.

DELETE <номер строки> TO <номер строки> [7]

Стирание строк в заданном диапазоне.

Примеры:

```
DELETE TO 50
DELETE 10 TO 50
```

DO [D]

DO WHILE условие входа [D]

DO UNTIL условие входа [D]

Ключевое слово обозначающее начало цикла.
Цикл DO начинается всегда и без условий.
Цикл DO WHILE начинается, если заданное условие истинно.
Цикл DO UNTIL начинается, если заданное условие ложно.
Цикл должен закончиться командой LOOP.

DPOKE адрес,число [P]

Загрузка двухбайтового числа в диапазоне 0...65535

EDIT <номер строки> [0]

Позволяет нормальное редактирование строки. Появляется при нажатии кл.[0], если до этого была нажата клавиша ENTER.

ELSE инструкция [E]

Третья (альтернативная) часть структуры IF-THEN.
Пример: IF условие THEN инструкция ELSE инструкция

END PROG [3]

Слово, обозначающее конец процедуры вызываемой по имени.

EXIT IF условие [1]

Выход по заданному условию из цикла до-LOOP.

FILL X,Y [F]

FILL <INK цвет>;X,Y [F]

FILL <PAPER цвет>;X,Y [F]

Закрашивание знака данным цветом (если нажато FILL или FILL +NK) фона знака (если нажато FILL PAPER). Допускается вызов сложных команд. Например:

```
FILL INK 2; PAPER 1; FLASH 1;X
```

GET числовая переменная или переменная-строка [G]

Присвоение переменной номера клавиши (счет клавиш идет с 11 для A, 12 для B, 13 для C и т.д.) Или самой буквы на нажатой клавише.

JOIN <номер строки> [SHIFT 6]

Объединение строки с заданным номером (или текущей строки, если номер не указан) со строкой, находящейся в нижней части экрана. Новый номер равен указанному номеру или номеру текущей строки.

KEYIN строка [SHIFT 4]

Ввод в программу поданной строки.

KEYWORDS 0 [8]

KEYWORDS 1 [9]

Переключатель ключевых слов BETA-BASIC на знаки псевдографики, вызываемые в режиме GRAFICS.

LIST номер строки TO номер строки

LLIST номер строки TO номер строки

Печать строк программы в заданном диапазоне.
LOOP [L]
LOOP UNTIL условие [L]
LOOP WHILE условие [L]
Ключевое слово, обозначающее конец цикла.
При LOOP цикл кончается всегда и без условий.
При LOOP UNTIL цикл кончается, если заданное условие истинно.
При LOOP WHILE цикл кончается, если заданное условие ложно.
GO TO ON переменная; номер строки, номер строки,...[O]
GO SUB ON переменная; номер строки, номер строки,...[O]
Переход на строку в зависимости от значения переменной.
ON ERROR номер строки [N]
Включение части программы обслуживания ошибок. При ошибке управление передается на заданную строку. Переменная ERROR получает значение кода ошибки.
RLOT X,Y; строка
Вывод строки знаков в любое место экрана. Координаты задают позицию левого верхнего угла знака строки.
Poke адрес, строка
Вводит в память знаки заданной строки, начиная с данного адреса.
Pop <числовая переменная> [Q]
Возвращает адрес, откуда был сделан вызов GO SUB, DO-LOOP, PROC.
PROC имя [2]
Выполнение процедуры с данным именем.
RENUM <начало TO конец><LINE новое начало>
<STEP шаг> [9]
Перенумерация строк из заданного диапазона на строки, начинающиеся с заданной строки с данным шагом. По умолчанию первая строка = 10, шаг = 10.
ROLL направление <,сдвиг><;X,Y;ширина,высота> [R]
"Перемотка" заданного окна экрана. Изображение, исчезающее с одной стороны окна, появляется с противоположной.

Код	направление	обьект передвижения
1	влево	атрибуты
2	вниз	атрибуты
3	вверх	атрибуты
4	вправо	атрибуты
5	влево	изображение
6	вниз	изображение
7	вверх	изображение
8	вправо	изображение
9	влево	изображение и атрибуты
10	вниз	изображение и атрибуты
11	вверх	изображение и атрибуты
12	вправо	изображение и атрибуты

SCROLL <направление><,сдвиг><;X,Y;ширина,высота>[S]
Сдвиг содержимого заданного окна экрана на одну линию. Синтаксис такой же, как в команде ROLL.
SORT таблица или строка [M]
SORT INVERSE таблица или строка [M]
Упорядочение таблицы знаков или строки.
SRLIT
В месте нахождения курсора строка разделяется на 2 части.
TRACE <номер строки> [T]
Пуск пошагового выполнения программы, начиная с заданной строки. Выключение режима командами RUN,CLEAR и TRACE 0.
USING строка-образец;число [P]
Использование в команде PRINT USING позволяет задавать формат распечатки чисел. Знаки "+" в строке-абзаце означают пробелы перед числами.

Глава 2
Описание функций BETA-BASIC

Ниже приводится краткое описание функций, добавляемых системой BETA-BASIC с описаниями аргументов. Способы ввода функций в программу указаны в квадратных

скобках.

AND (число,число) [FN A( ] - двоичная операция "логическое и".
BIN$ (число) [FN B&( ] - двоичная распечатка данного десятичного числа.
CHAR$ (число) [FN C$( ] - преобразование целого числа без знака в диапазоне 0-65535 в эквивалентную строку из двух знаков.
COSE (число) [FN C( ] - косинус числа, вычисляемый с 4 значащими цифрами.
DEC (строка) [FN D( ] - преобразование строка с записью шестнадцатиричного числа в десятичное число.
DPEEK (адрес) [FN P( ] - возвращает двухбайтовое число, находящееся в памяти по данному адресу.
FILLED () [FN F( ] - количество элементов изображения, заполненных последней командой FILL.
HEX$ (число) [FN H( ] - преобразование десятичного числа в строку с записью шестнадцатиричного числа.
INSTRING (начало,строка 1,строка 2) [FN I( ] - возвращает позицию первого знака строки 2, находящейся внутри строки 1 при просмотре строки 1, начиная с заданной стартовой позиции. Если внутри строки 1 нет строки 2, то возвращается 0.
MEM () [FN M( ] - возвращает количество свободных байтов памяти.
MEMORY$ () [FN M$( ] - возвращает значение всей памяти от адреса 0 до 65535 интерпретированное как одна строка.
MOD (число 1,число 2) [FN V( ] остаток деления числа 1 на число 2 (деление по модулю).
NUMBER (строка) [FN N( ] - преобразование строки из двух знаков в двухбайтовое число, где каждый байт содержит число, отвечающее заданному коду ASCII.
OR (число 1,число 2) [FN O( ]- двоичная операция "логическое или", выполненная над заданными числами.
RNDM (число) [FN R( ] - возвращает псевдослучайное число из диапазона от 0 до заданного числа.
SCRN$ (ряд,столбец) [FN K$( ] возвращает знак находящийся на экране в заданной позиции.
SINE (число) [FN S( ] - синус числа, вычисляемый с 4 значащими цифрами. Более быстрое вычисление, чем в оригинальной системе SPECTRUM BASIC.
STRING$ (количество,строка) [FN S$( ] - повторение строки заданное количество раз.
TIME$ () [FN T( ] - текущее время, измеренное по часам CLOCK.
USING$ (строка-образец,число) [FN U$( ] - знаковая запись числа в заданном формате (как USING).
XOR (число 1,число 2) [FN X( ] - двоичная операция "логическое исключающее или", выполненная над заданными числами.

Глава 3
Специальные переменные

В системе BETA-BASIC имеются встроенные переменные доступные по имени.

XOS, YOS - относительное начало координат, первоначально установленное на 0,0. Может быть изменено в диапазонах 0-255 для XOS и 0-175 для YOS. Обнуляются по командам CLEAR и RUN.

XRG, YRG - максимальные границы изображения, первоначально установленные на 256 для XRG и на 176 для YRG.

Во время выполнения команд ON ERROR и TRACE обновляются следующие специальные переменные:

ERROR - код последней обнаруженной ошибки.

LINE - номер последней выполненной строки программы (при TRACE), номер строки с обнаруженной ошибкой (при ON ERROR).

STAT - номер последнего выполненного оператора программы (при TRACE), номер оператора с обнаруженной ошибкой (при ON ERROR).
Имена специальных переменных можно вводить заглавными и строчными буквами.

M E G A - B A S I C

I. К л а в и а т у р а.

Как только программа будет загружена, вы увидите короткое сообщение и инверсный квадрат в левом нижнем углу экрана. Это новый курсор который будет показывать где будет появляться входная информация. При использовании "MEGA BASIC" весь экран используется для входной информации в отличии от обычного бейсика, где весь экран разбивается на 2 части.

При записи операторов необходимо набирать все символы, например, если в бейсике оператор PRINT мог быть записан за одно нажатие клавиши "P", то теперь необходимо будет писать "P" "R" "I" "N" "T". Однако "Мега-бейсик" позволяет некоторые операторы записывать в виде аббревиатуры. Список таких операторов приведен ниже. При этом аббревиатура должна заканчиваться точкой.

A.TTR	ER.ASE	ME.RGE	RES.TORE
BE.Ep	E.Xp	M.OVE	RET.URN
B.IN	FL.ASH	NE.XT	R.ND
BO.RDER	F.ORMAT	N.OT	SA.VE
BR.IGHT	GOS.UB	OP.EN#	S.CREEN$
CH.R$	G.O TO	OV.ER	ST.R$
CI.RCLE	I.NKEY$	PA.PER	T.Ab
CLE.AR	INP.UT	PAU.SE	TH.EN
CL.OSE#	INV.ERSE	PE.EK	U.SR
C.ODE	L.EN	PL.OT	V.AL$
CON.TINUE	LI.NE	P.OINT	VE.RIFY
DA.TA	LL.IST	PR.INT	
D.EFFN	LP.RINT	RA.NDOMIZE	
DR.AW	LO.AD	RE.AD	

Нижняя строка на экране используется для индикации режима курсора, ниже в таблице показаны обычные режимы SPECTRUMA, а рядом сообщение на экране которое вы получаете в нижней строке.

Таблица 1.

режим курсора	нижняя строка экрана
L	CAPS OFF
G	CAPS OFF GRAPHICS
C	CAPS ON
E	CAPS OFF EXTENDED

Режим курсора изменяется обычным путем - нажатием клавиш CAPS LOCK, GRAPHICS, CAPS SHIFT, SIMBOL SHIFT.

2. Р е д а к т о р.

Возможности у "MEGA-BASIC" намного больше чем у обычного бейсика (по редактированию строк).

Таблица 2.

команда	с о д е р ж а н и е
EDIT	копировка очередной программной строки во входную строку, готовую для редакции.

INVERSE VIDEO	удаление символа правее курсора
CURSOR LEFT	перемещение курсора влево на один символ
CURSOR RIGHT	перемещение курсора вправо на один символ
CURSOR DOWN	перемещение курсора вниз на одну строку
CURSOR UP	перемещение курсора вверх на одну строку
DELETE	удаление символа слева от курсора.
<=	перемещение курсора на начало входной строки
<>	удаление всех символов от начала курсора до конца входной строки
>=	перемещение курсора в конец входной строки.
SCREEN$	автоматический листинг. (выход на окно 1) верхней строкой листинга является текущая строка
OR	перемещение очередной строки вниз на одну и листинг
AND	перемещение очередной строки ВВЕРХ на одну и листинг
STOP	перемещение COPY-курсора влево на один символ
STEP	перемещение COPY-курсора вверх на одну строку.
NOT	перемещение COPY-курсора вниз на одну строку
TO	перемещение COPY-курсора вправо на один символ
AT	копировка символа от COPY курсора к входному курсору. Этот оператор используется для копировки символов, имеющих стандартный размер и не может применяться для символов при 64 рабочих столбцах экрана,символов с двойной шириной и высотой
OVER	перемещение COPY-курсора на следующее окно
INVERSE	удаление COPY-курсора в верхний угол очередного окна.

Заметим, что экран разделен на 4 различные секции или окна и каждое используется для специфических задач.

Таблица 3.

секция	с о д е р ж а н и е
ОКНО 0	для индикации поступающей от пользователя информации и сообщений об ошибках
ОКНО 1	индикация листинга программы начиная с редактируемой строки
ОКНО 2	выходная информация по командам бейсика
ОКНО 3	используется как фронт-панель

При размещении на экране окна перекрывают друг друга.

Заметим также, что 2-ой курсор может быть использован как COPY. Этот курсор появляется как мигающий квадрат на экране и может перемещаться внутри очередно-

го окна с помощью запрограммированных управляющих клавиш. COPY-курсор функционирует только в окнах 0, 1, 2.

Также, как использование CAPS SHIFT 1 для редакции очередной строки, возможна редакция любой строки в программе, используя команду EDIT. За этой командой следует числовое выражение, указывающее какая строка будет редактироваться. Если требуемая строка не существует, то происходит переход к следующей строке для ее редакции. Если же нет и следующей строки, то на экране появляется сообщение:

L I N E N O T F O U N D
(с т р о к а н е н а й д е н а)

3. Клавиши определяемые пользователем (U D K)

Имеется возможность запрограммировать верхний ряд клавиш и создать тем самым более 255 символов. Для программирования используйте команду KEY, за которой должно следовать числовое и символьное выражение, разделенные запятой. Числовое выражение определяет, какая из клавиш программируется, символьное выражение - содержание этой клавиши. Записывая ENTER символ CHR$13 в конце определяемого вами символьного выражения, вы получите автоматическое его выполнение после нажатия UDK. "MEGA-BASIC" уже включает в себя следующие UDK.

Таблица 4

U D K	С о д е р ж а н и е
VERIFY	В символьном выражении записано RUN+CHR$13, при нажатии на эту клавишу выполняется очередная программа в памяти.
VAL$	В символьном выражении записано LOAD "", RUN+CHR$13 при нажатии этой клавиши загружается, а затем выполняется очередная программа на кассете.

Во время выполнения программы SPACE-клавиша может быть использована как SHIFT-клавиша. Поэтому, чтобы получить пропуски между символами, обе эти клавиши должны быть нажаты одновременно. SPACE-клавиша теперь, в сочетании с другими клавишами, будет представлять собой управляющую клавишу. Ниже приведены возможные клавиши управления во время выполнения программы (RUN-TIME).

Таблица 5.

Клавиша	с о д е р ж а н и е
CONTROL F	Вызов фронт-панели
CONTROL E	Останов выполнения программы печать сообщения ESCAPE и переход к строке редактора.
CONTROL E	Останов выполнения программы установка начальных значений PAPER, BORDER и INK. Переход к строке редактора. Эта функция не уничтожает программы на бейсике в памяти компьютера

4. Команды экрана.

Размер и форма символов, а также способ по которому они могут быть выведены - факторы облегчающие работу с "MEGA-BASIC". Площадь экрана для выходной информации может быть любого размера. Отдельные участки площади экрана называются "окнами". Их всего 10. Каждое окно имеет свой номер от 0 до 9. Все они могут быть использованы в вашей программе.

CURRENT - N - переход на другое окно. N - номер окна (0 ... 9).
WINDOW - Y, X, D, W - определение размера и положения окна на экране.
Y - строковая позиция верхнего левого угла (может изменяться от 0 до 23).
X - столбцовая позиция верхнего левого угла (может изменяться от 0 до 63).

D – глубина окна (в строках).
W – ширина окна (в столбцах).

"MEGA-BASIC" разбивает экран на 24 строки и 64 столбца. Функции ATTR и SCREEN$ используют старую координатную систему, а оператор PRINT AT новую систему. Координаты для PRINT AT задаются относительно верхнего левого угла очередного экрана, в то время как ATTR SCREEN$ используют абсолютные координаты экрана. Если Y+D больше 24 или X+W больше 64 , то вы получите сообщение об ошибке WINDOW TOO LARGE (окно слишком большое). Если же значение меньше 0, то вы увидите WINDOW TOO SMALL (окно слишком маленькое). После выполнения команды WINDOW позиция устанавливается (для печати) в верхнем левом углу окна.

CLS – очистка всего экрана.

CLW-N,(A) – очистка одного окно или его части.
N – номер окна
A – номер очищаемой записи

Существует 4 различных типа команды CLW

Таблица 6.

CLW	содержание
CLW-N,0	Затушувмвается окно, используя цвет PAPER
CLW-N,1	Затушевывается окно, используя цвет INK
CLW-N,2	Инвертируется содержимое экрана. Цвет PAPER меняется на цвет INK
CLW-N,3	Очистка только от атрибутов. Эта опция позволяет пользователю изменить цвет окна, не разрушая его содержимое

Во всех случаях позиция для печати устанавливается в верхнем левом углу окна. Команда CLW всегда использует текущие атрибуты. Так, например, если текущее окно 3, то после записи команды CLW 0,0 и ее выполнения окно 0 будет затушевываться, используя атрибуты окна 3.

PAN-B,Z – перемещение окна влево, вправо.

SCROOL-B,Z – перемещение окна вверх, вниз.
B – цвет кромки экрана.
Z – количество пикселей, направление переноса (положительное число – вправо, вверх; отрицательное число – влево, вниз).

PANW-Z – поворот окна влево, вправо.

SCROOLW-Z – поворот окна вниз, вверх.

Все эти команды действуют только на дисплейный файл.

FX-M,N – выбор окна. M – число, указывающее назначение окна,
N – номер окна

Таблица 7.

FX	содержание
FX-0,N	Окно N назначается для входной информации и сообщений об ошибках
FX-1,N	Окно N назначается для автоматического листинга.
FX-2,N	Окно N назначается для выходной информации
FX-3,N	Окно N назначается для фронт-панели.

MODE-N,(D) – выбор размера символов, выводимых на экран. D – размер символа.

Таблица 8.

MODE	содержание

MODE-N,1	Используется 64 столбца и каждый символ имеет размер 4*6 пикселей
MODE-N,2	Обычный размер 8*8 пикселей
MODE-N,3	Символы двойной высоты 8*16
MODE-N,4	Символы размером 16*16

Для каждого окна могут быть заданы свои размеры символов. При использовании команды MODE-N,4 возможно получение символов с оттенками, поскольку в этом случае каждый пиксель занимает 4 пикселя экрана. т.е., установив один образ для 4-х точек, можно получать теневые эффекты.

STRIPPLE-W – запись образа (задает интенсивность тени)
W – номер образа (0 ... 15)
FONT-L – выбор набора символов.
L – номер набора символов

Таблица 9.

FONT	с о д е р ж а н и е
FONT-0	Обычные символы SPECTRUMA
FONT-1	Символы подобные применяемым в BC MIKRO AKORN
FONT-2	Символы подобные применяемым в AMSTRAD CPC 464

Новые символьные наборы хранятся в ОЗУ и могут быть использованы с помощью описанных команд. Каждый символ занимает 8 байт и имеет определенную форму. В псевдо BC MIKRO символы находятся в памяти, начиная с адреса 48000, а в псевдо AMSTRAD – с адреса 45000.
Если A – определенный символ, то адрес, по которому он находится определяется по формуле: S + 8 * (CODE A – 32), где S – начальный адрес.

SHR$ символьный набор SPECTRUMa содержит печатаемые символы и управляющие символы. Несколько новых управляющих символов, входит в "MEGA-BASIC".

Таблица 10.

SHR$	с о д е р ж а н и е
SHR$1-4	Выбор режима очередного окна
SHR$7	Перестановка символа к курсору
SHR 24-31	Выбор окна для выходной информации. SHR$ 24 соответствует окну 0, SHR$ 31 окну 7

Команда VDU эквивалентна PRINT CHR$. К примеру: VDU-7 обеспечивает одинаковый размер всех последующих символов, а VDU 65,66 печатает "AB".

DOWN-Y,X,A$ – вывод строк символов внизу экрана.
Y,X – соответственно начальная строка и столбец на которые выводится информация.
A$ – выводимая информация. Если она занимает больше 1 строки, то продолжение появится уже наверху.

SPRINT-X,Y,A,B,A$– вывод на экран символов любого размера.
X,Y – позиция первого курсора на экране.
A,B – коэффициент увеличения символа в направлении соответственно.
A$ – строка выводимых символов. Если символы достигли правой кромки экрана, то запись появляется с левой стороны экрана.

PRINTER

Обеспечивает вывод информации на периферийные устройства такие, например,

как принтер. Вслед за командой следует числовое выражение. Если результат его равен 0, то информация будет выводится только на экран. Если результат отличен от 0, то при каждом появлении символа на экране, будет вызываться подпрограмма вывода в машинных кодах. Адрес ее хранения находится в ячейках 59934 и 59935. Примечание : команды CLEAR#, OPEN#2, CLOSE#2 не выполнимы в "MEGA-BASIC".

5. Графика.

CHANGE и SWAP предназначенны для манипулирования файлом атрибутов.
CHANGE-M,A изменение определенной части каждого атрибута байта.
M - маска, указывает, какой из битов каждого атрибутного байта должен быть изменен.
A - данные, показывают в чем должно состоять это изменение.

Таблица 11.

CHANGE	содержание
CHANGE-1	Логическое NOT, все 1 превращаются в 0, а все 0 в 1.
CHANGE-2	Логическое AND для байта файла атрибутов и отрицательной маски.
CHANGE-#	Каждый атрибутный байт через логическое OR соединяется с данным байтом.

```
SWAP-W,S          - замена одних атрибутов на другие.
                  W - новые атрибуты, S - старые атрибуты.
FADE              - производит некоторые специальные эффекты.
```

Введите следующую короткую программу:

```
10 FOR A=0  TO 703
20 POKE 22528+A, PEEK A
30 NEXT A
40 FADE-0
```

Первые 3 строки заполняют файл атрибутов случайными символами. По команде FADE файл атрибутов исследуется. Каждый байт, который равен численному выражению, после команды FADE остается неизменным, другие уменьшаются. Этот процесс продолжается до тех пор, пока каждый байт в файле атрибутов не будет равен числовому выражению, стоящему после команды FADE.

```
INVERT            - Инвертирование всего экрана. Цвет INK меняется на цвет
                    PAPER и наоборот.
DEFG-&,,,,,,      - Создание новых графических элементов.
                &   - Любой символ от A до U, определяет местонахождение нового
                      графического элемента
                    - 8 чисел, разделенных запятыми, определяют форму графического элемента (первое число определяет верхний ряд, последнее - нижний)
GET-O,A,Y,X,D,W   - Перенос содержимого экрана в память.
                A   - Адрес начиная с которого хранится информация экрана.
              Y,X   - Соответственно номер строки и столбца верхнего угла записанной площади экрана.
              D,W   - Соответственно глубина и ширина этой площади.
PUT-F,A,Y,X,W,D   - Перенос содержимого памяти на экран.
                F   - Способ востановления экрана из памяти (F=0.....6).
      A,Y,X,W,D   - Аналогично GET.
```

PUT и GET используют ту же координатную систему, что и ATTR и SCREEN$. Их координаты абсолютны и не зависят от положения верхнего левого угла окна на экране.

Команда GET обеспечивает сохранение дисплейного файла в памяти с информацией и об атрибутах. Использование команды CLEAR позволяет резервировать участок памяти для сохранения информации экрана. Можно определить число байтов, зани-

маемых информацией экрана, с помощью уравнения Y, W, D.

SPUT-A,X,Y,B,C,W,D - перенос содержимого памяти на экран, но с увеличением выводимого изображения.

B,C - коэффициенты увеличения.

6. Управление выполнением программ.

"MEGA-BASIC" обеспечивает работу с процедурами, которые могут быть вызваны по их имени на выполнение, как будто это новые команды. И точно также , как и после команды, эти выражения могут быть назначены как переменные, готовые для манипуляции внутри программы. Недостатком процедур "MEGA-BASIC" является то, что они не могут манипулировать с локальными переменными, а их переменные являются общими для всей программы. Используя процедуры, можно разделить программу на ряд задач. Для каждой частной задачи составляется своя процедура. Каждая процедура может быть проверена отдельно и затем они все объединяются вместе для окончательной программы. Выбирайте внимательно имена для ваших процедур и тогда будет намного легче находить их в программном потоке.

Начало процедуры определяется символом : следом - имя процедуры. Оператор, определяющий процедуру должен быть первым в строке. После имени процедуры должны следовать ее параметры. Конец процедуры определяется оператором ENDPROG. Как только встретится этот оператор, компьютер перейдет к оператору, следующему за оператором вызова процедуры. В конце оператора можно записать имя процедуры.

Пример:

```
9000 DISPLAY  A,A$
9010 PAPER A: INK 9
9020 MODE -4 : STRIPPLE-6
9030 PRINT A$
9040 ENDPROG-DISPLAY
```

Строка 9000 определяет процедуру DISPLAY A,A$, показывает, что для него необходимы числовое выражение (A) и символьное выражение (A$). Строка 9010 устанавливает цвет, а строка 9020-правильный размер символов. Строка 9030 выводит на печать символьное выражение A$ и, наконец, строка 9040 определяет конец процедуры. Для активизации этой подпрограммы вы будете использовать в своих программах строку подобную этой: DISPLAY-2, "MEGA SPECTRUM".

REPEAT-UNTIL

Так же хорошо, как и с процедурами, "MEGA-BASIC" обеспечивает работу с циклами типа REPEAT-UNTIL. REPEAT команда начинает цикл, после команды UNTIL следует числовое выражение. Если этому числовому выражению присваивается нулевое значение, то выполнение программы возвращается назад, к оператору записанного после последнего REPEAT. Если выражение является положительным числом, то выполняется следующий оператор программы. Допускается до 10 циклов вложения типа REPEAT-UNTIL.

Для сохранения номеров строки операторов процедур и циклов REPEAT-UNTIL используется стек. Когда вызывается процедура, номер строки, следующий за оператором вызова, размещается в стеке - это позволяет программе знать, куда возвращаться после выполнения оператора процедуры. Когда будет выполнена команда REPEAT, номер строки с оператором после команды REPEAT заталкивается в стек. Это необходимо чтобы после выполнения команды UNTIL было известно где последняя команда REPEAT. После выпонения команды ENDPROG, и перехода к строке номер которой хранится в стеке, последний убирается оттуда. Также и после выполнения команды UNTIL верхнее значение выталкивается из стека. Если стек пустой и будет попытка вытолкнуть из него, то на экране появится сообщение об ошибке:

" PROG STACK UNDERFLOW"

Любая попытка записать в стек более 10 значений заканчивается сообщением об ошибке, и на экране появится: " PROG STACK OVERFLOW ".

POP - извлечение значения из стека.

PUSH M,S - запись значения в стек.

M - номер оператора

S - номер строки

PCLEAR - очистка стека.

BRANCH - переход к подпрограмме после окончания каждой программной строки.

- Числовое выражение, указывающее на начало подпрограммы. При =0 "MEGA BASIC" действует как обычно в конце строки.

Конец подпрограммы определяется оператором ENDPROC.

MTASK R - выполнение программы с двух различных мест.

R - числовое выражение, указывающее на начало второй части программы. Считается, что программа разделена как бы на две части. ПервыЯ начинается после оператора MTASK, а вторая со строки указанной в R. "MEGA BASIC" выполняет поочередно строки из каждой программы. При R=0 такой многопрограмный режим невозможен.

7. Отладка и редактирование программ.

Команды BRANCH, TRON, TROFF, SPEED.

BRANCH - Описана выше. С ее помощью можно вызывать подпрограмму печати после любой программной строки для распечатки значений интересующих вас переменных.

TRON S - Печать на экране номера очередной выполняемой строки.

S - Номер строки, расположен в нижнем углу экрана.

TROFF - Отмена режима, задаваемого командой TRON.

SPEED V - Изменение скорости выполнения программы.

V - Число, определяющее скорость выполнения программы. При V=0 обычная скорость выполнения программы. При V=255 - самая медленная. При этом компьютер останавливается после выполнения каждого оператора и ждет, пока пользователь не нажмет клавишу.

AUTO S,P - Ввод номера очередной строки при нажатии клавиши ENTER. Используется при вводе больших программ.

S - номер первой строки программы.

P - шаг нарастания номеров.

Остановить режим автоматической нумерации строк можно, используя EXTEND SYMBOL SHIFT 'L'.

DELETE S1,SN - Удаление большого блока программных строк

S1 - Номер первой строки удаляемого блока.

SN - Номер последней строки удаляемого блока.

BROFF - Программное аннулирование клавиши BREAK. Используется для защиты своих программ от несанкционированного доступа.

BRON - Возвращение клавише BREAK ее возможностей.

RESTART S - Обход строки с ошибкой при выполнении программы.

S - Номер строки, к которой осуществляется переход при возникновении ошибки.

RESTART OFF - Отмена вышеприведенного режима.

Команда RESTART не действует с INTERFASE 1 и по отношению к ошибкам "MEGA-BASIC". После обработки ошибки с помощью команды RESTART, некоторая полезная информация сохраняется в ячейках памяти:

Таблица 12.

Адрес	с о д е р ж а н и е
69873,69874	Номер строки в которой обнаружена ошибка.
69875	Оператор внутри строки в котором обнаружена ошибка
59862	Код обнаруженной ошибки

8. Звук.

"MEGA-BASIC" открывает для вас 2 новых пути создания звука на вашем SPECTRUMe. Это применение оператора PLAY и использование генератора звуковых прерываний (ISG).

PLAY - N,L,S,D,F - создание звука.

N=0 - звук.

N=1 - шум,
L - длительность шага,
D - число шагов,
F - изменение частоты после каждого шага.

I S G

Команда PLAY является расширением команды BEEP, но при использовании "MEGA-BASIC" имеется другая возможность производить звук даже при выполнении другой программы. Эта возможность имеется благодаря использованию ISG. При выполнении обычных программ SPECTRUM через каждую 1/50 секунды останавливает выполнение программ и вызывается подпрограмма в машинных кодах, по которой происходит сканирование клавиатуры. При использовании "MEGA-BASIC" также производится сканирование клавиатуры, но при этом если нужно можно производить звук имеются следующие команды, управляющие ISG:

Таблица 13.

команда	содержание
SOFF	Отключение ISG
SON	Включение ISG
SREP N	Если N=0 то данные хранимые в звуковом буфере, будут воспроизводиться только 1 раз. Если N=1, то каждый раз.
SOUND	Это новая команда управления звуковым буфером. Формат команды SOUND приведен ниже.

SOUND - N,A,B,C,D
N=0 - очистка звукового буфера,
N=1 - добавление в звуковой буфер новых звуков,
A=0 - звук,
A=1 - шум,
B - шаг звуковой частоты,
C - число шагов,
D - число повторений одной последовательности.

Более сложные звуки можно получать, замедлив выполнение программы. Вот короткий пример, показывающий работу ISG:

```
10  SOUND-0,0,1,20,255
20  SREP-1
30  SOH
40  MODE-4: STIPPLE-6: FONT-2
50  VDU-/128+RND*15/
60  PAPER RND 7: INK 9
70  GO TO 50
```

9. Машинные коды и "MEGA BASIC"

DOKE и CALL

В "MEGA-BASIC возможно выполнение подпрограмм в машинных кодах. Но, во-первых все подпрограммы в машинных кодах должны размещаться в участке памяти до адреса 45000. Начиная с этого адреса и выше начинается сама программа "MEGA-BASIC ". Во-вторых, "MEGA-BESIC" позволяет сразу же записывать в память двухбайтовое число с помощью команды DOKE. После этой команды следует 2 числа. Первое указывает адрес, а второе - записываемую в память информацию.

Вызов подпрограммы в машинных кодах осуществляется с помощью команды CALL, за которой следует число, указывающее адрес этой подпрограммы. За адресом может следовать число, являющееся параметром для вызываемой подпрограммы, которое записывается в стек.

10. Фронт- панель.

Фронт-панель обеспечивает возможность выполнять изменения в памяти и в ре-

гистрах Z80. Фронт-панель активизируется либо по команде M0N, либо по команде CONTROLF при выполнении программы. Фронт-панель использует окно 3, а это всегда почти по крайней мере 40 столбцов и 20 строк.

Как только фронт-панель активизируется, вы сразу же увидите столбцы 16-ричных чисел. Слева - список регистров с аккумулятором вверху. Звездочка указывает текущий регистр. Справа - текущий участок памяти, на инверсной полосе которого находится выполняемая строка. Манипуляции с данными фронт-панели можно выполнять с помощью ряда однобуквенных команд, после которых может идти 3 16-ричных числа. Числа должны быть записаны полностью, т. е. даже если отдельные разряды равны 0, то это надо указать. Отметим, что N представляет собой 8-ми битовое число, а NN 16 битовое число.

Таблица 14.

команда	с о д е р ж а н и е
K NN	Запись в текущий регистр 16-ричного числа.
P	Смещение указателя регистра
L NN,NN,NN	Перемещенну блока памяти. 1-е число указывает адрес откуда второе число - адрес куда. 3-е - длительность блока
M NN	Задание адреса текущей ячейки памяти.
K	Продолжение выполнения программы после указателя
I NN,NN,NN	Заполнение блока памяти. 1-е число начальный адрес блока. 2-е длительность блока. 3-е значение числа записываемого в блок
J NN	Вызов подпрограммы в машинных кодах записанной по адресу NN
ENTER	Смещение указателя текущей ячейки
"="	Возврат к предыдущей ячейке

A R T S T U D I O

Графический редактор O.C.P. OXFORD 86

В о з м о ж н о с т и р е д а к т о р а

1. Система работы с перекрывающимся меню (аналогично "мышинтошу").
2. Использование следующих инструментов :
   - карандаша с произвольно назначаемым сечением трафета;
   - кисти с произвольными размерами и формой;
   - краскораспылителя с регулируемым факелом краски;
   - валика для заливки с различной текстурой (кирпич, рельеф и т.д.);
   - 2-Х, 4-х, 8-крат. лупы для рассматривания мелких деталей.
3. Работа с окнами: размножение, переносы, повороты и т.д.
4. Работа с изображением : растяжение, сплющивание, наложение и т.д.
5. Работа с примитивами: точки, линии, окружности, треугольники, дуги и т.д.
6. Режим резиновых нитей (все примитивы - резиновые)
7. Печать надписей любым шрифтом в любом направлении
8. Операции с файлами на кассетах (дискетах).
9. Работа с цветом и атрибутами.
10. Отмена любого неправильного действия.

Основное меню

печать	файлы	атрибуты кисти		экран	отмена	указатель роллинга
PRINT	FILE	ATTRS	PAINT	MISC	UNDO	/!
WINDOWS	FILL	MAGNIFY	Text	SHAPES		V
окна	закраска валиком	лупа	текст	примитивы		роллинг экрана

Это меню всегда (кроме особых случаев) находится вверху экрана. Под ним две строки "невидимы" и при попадании курсора в них его не видно. Если это произошло (курсор "пропал") продолжайте перемещать курсор вверх (вниз) и он появится .

Все управление курсором

Русск.	Англ.	KEYBOARD	SINCLAIR
Вверх	UP	Q	
Вниз	DOWN	A	SINCLAIR
Влево	LEFT	O	JOYSTICK
Вправо	RIGHT	P	N 1
Действие	FIRE	Любая клав. нижн. ряда	

Режим ТЕКСТ
Клавиша "ENTER" завершает печать строки

SHAPES
POINT
LINES
CONT. LINE
RESTANGLES

Примитивы
Точка * 1
Линия 1 *----------* 2
Ломаная линия 1 *---* 2 3 *---* / 4
Прямоугольник 1 *--------* 2 3 4

TRIANGLES	Тре- * 2 уголь- / ! ник I *---* 3
CIRCLES	Окружность I *) 2
RAYS	3 * * 4 Лучи ! / I *----------* 2
ELASTIK V- вкл. X- выкл.	Режим резиновых нитей
SHAP HRZ. V- вкл. X- выкл.	Дискретность координат по "x"
SHAP VRT. V- вкл. X- выкл.	Дискретность координат по "y"
UNDO	Отмена

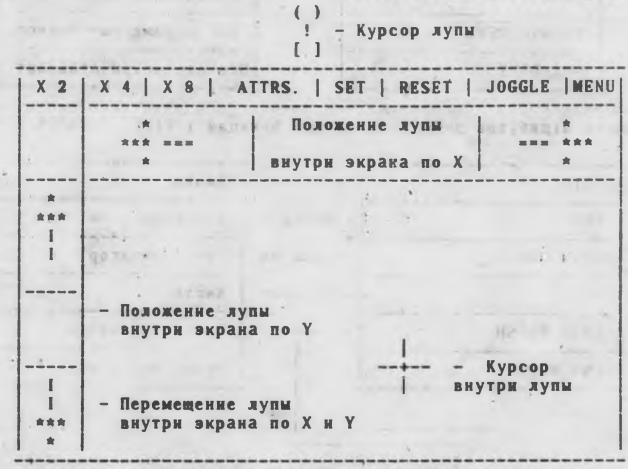
* – Курсор примитивов (здесь изображен звездочкой). Цифры рядом с ним указывают последовательность ввода точек. "UNDO" отменяет любое последнее (!) действие выполненное над экраном (в том числе операции в окнах). Для отмены подведите курсор меню к надписи "UNDO" и нажмите кнопку "FIRE" (взять).

WINDOWS	Окна
DEFINE WINDOW	Определить I *----* 2 Окно !---* 3
LAST WINDOW	Последнее I *----* 2 окно !---* 3
WHOLE SCREEN	Весь экран – окно
CLEAR WINDOW	Очистка окна
CUT $ PASTE	Перенести и размножить
CUT, CLEAR $ PASTE	Перенести и стереть старое
INWERT WINDOW	Инвертировать окно
RE-SCALE WINDOW	Перенести и масштабировать
CLEAR $ RE-SCALE	Масштабировать и стереть старое
FLIP HORIZONTAL	Горизонтальный перево-рот (зеркало)
ROTATE 1/4	Поворот на 90 град.
ROTATE 1/2	Поворот на 180 град.
ROTATE 1/4	Поворот на -90 град.

MERGE x	Режим наложения
MULTIPLE x	Режим многократных действий

Окна имеют прямоугольную форму. При масштабировании с уменьшением тонкие линии и мелкие детали исчезают, а при увеличении, линии увеличивают свою толщину. Многократные действия выполняются от последнего окна.

MAGNIFY	Лупа
MAG. X2	Увеличение x2
MAG. X4	Увеличение x4
MAG. X8	Увеличение x8
GRID x	Режим сетки



В режиме триггера первое нажатие "F" зажигает точку, второе - гасит (или наоборот, если точка уже была зажжена).

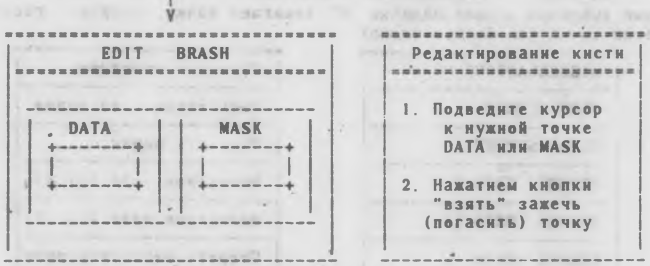
MISCELLANEOUS	Сервис - операции
VIEW SCREEN	Весь экран - на экран
CLEAR SCREEN	Очистить экран
BRIGHT GRID 1	Шахматное поле 1x1 AT
BRIGHT GRID 2	Шахматное поле 2x2 AT
REMOVE GRID	Стереть шахматное поле
CHANGLE COLOUR	Назначить цвет
VERSION NUMBER	Номер версии редактора

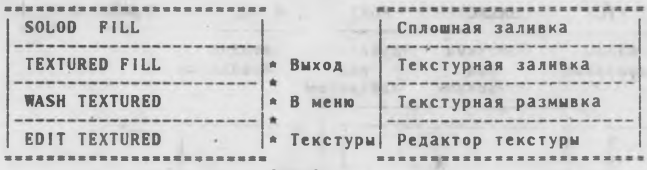
Шахматное поле получается битами яркости (BRIGHT) в атрибутах и используется для совмещения точек на экране.

ATTRIBUTES	Атрибуты
SET INC	Цвет карандаша
SET PAPER	Цвет бумаги
SET BORDER	Цвет бордюра
BRIGHT	Яркость MAX/MIN
FLASH	Мигалка вкл/выкл
OVER x	Режим надпечатки
INVERSE x	Режим инверсии
TRANSPARENT	Все параметры- трансп.
STANDART	Все параметры-стандарт.

Установка атрибутов действует на все функции (FILL , SHAPES и т.д.).

PAINT		Кисти
PEN	* Выход в *	Карандаш
SPAY CAN	*режим ин- *	Пульвелизатор
BRASH	*струмент.	Кисть
EDIT BRASH	------	Редактор кисти
INVERSE		Режим инверсии





[XXX]--
----|
| - курсор заливки (валик)
[]
[]

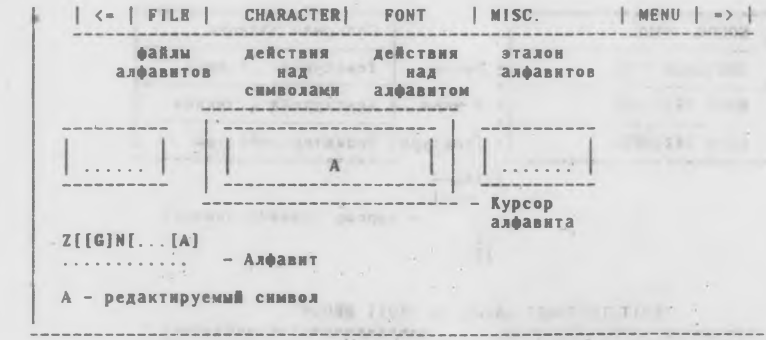
"EDIT TEXTURE" похож на "EDIT BRUSH".

Text		Текст
LEFT TO RIGHT		Печать слева направо
DOWN WARDS		Печать сверху вниз
NORMAL HEIGT		Нормальная выс. симв.
BOUBLE HEIGT	V	Двойная высота символа
TREBLE HEIGT		Тройная высота символа
NORMAL WIDTH		Нормальная шир. симв.
BOUBLE WIDTH	V	Двойная ширина символа
TREBLE WIDTH		Тройная ширина символа
SID WAYS	x	Буквы повернуть на 90
BOLD	V	Жирная печать
CAPS LOCK	x	Только заглавные буквы
SHAP HRZ.	X	Дискретность по гориз.
SHAP VRT.	X	Дискретность по верт.
FOND EDITOR		Редактор символов

][- Курсор начала строки -- - курсор символов

Курсор начала строки перемещается как любой другой курсор, но после нажатия клавиши "взять" исчезает, а на его месте появляется курсор ввода символов, после чего можно набивать любой текст с клавиатуры компьютера. После нажатия клавиши "ENTER" ввод символов заканчивается и на экране появляется курсор начала строки. "SHAP" совмещает символы с атрибутами.

возврат
в меню



CHARACTER	Действия над символ.
CLEAR INWERT	Стирание символа Инверсия символа
FLIP HRZ.	Горизонт. зеркальный поворот
FLIP VRT.	Вертикал. зеркальный поворот
ROTATE 1/4	Поворот на 90
SCROLL RIGHT	Роллинг вправо
SCROLL DOWN	Роллинг вниз

Такое же меню для действий над всем алфавитом. Алфавит можно загружать с магнитной ленты и выгружать на ленту.

CASSETTE	Кассета
SAVE FILE ...	Записать файл ...
LOAD FILE ...	Загрузить файл ...
LOAD NEXT FILE	Загрузить следующ.файл
VERYFY FILE ...	Проверить файл ...
VERYFY NEXT FILE	Проверить следующ.файл
MERGE FILE ...	Наложить файл ...
MERGE NEXT FILE	Наложить следующ. файл

- Для выбора режима нажмите "FIRE" ("взять"), указав курсором меню на необходимый режим.
- На запрос: "FILE NAME ?" - введите с клавиатуры имя файла и нажмите "ENTER".

Содержание

1.	Клавиатура	1
2.	Экран телевизора	2
3.	Принтер	3
4.	Порты	3
5.	Язык программирования БЕЙСИК	3
5.1	Функции	4
5.2	Операции	6
5.3	Операторы	7
5.4	Сообщения	13
5.5	Использование машинных кодов	16
5.6	Полный набор символов	19
6.	Ассемблер	23
6.1	Описание центрального процессора	24
6.2	Таблица клавиш "Спектрум"	25
6.3	Таблица набора литер "Спектрум"	26
6.4	Таблицы преобразования 10-ичных чисел в 16-ичные	26
6.5	16-ичная таблица сложения	27
6.6	Таблица флагов и времени выполнения команд	28
6.7	Команды ЦП в порядке кодов операции	30
6.8	Команды ЦП в порядке возрастания мнемонических обозначений	34
7.	Алфавитный указатель меток подпрограмм ПЗУ	38
8.	Копировщики	43
8.1	COPY-COPY	43
8.2	TFCOPY	43
8.3	COPY/86M	44
8.4	COPIER FM-3	44
8.5	SINCLAIR COPY	44
8.6	SPEC.COPY	45
8.7	MR.COPY	45
8.8	COPY DE LUXE	45
8.9	OMNI COPY-1	45
8.10	Копировщик защищенных программ LERM-7	46
9.	PASCAL	50
10.	Оптимизирующий компилятор языка БЕЙСИК - BLAST	58
11.	Редактор TLW 2	63
12.	MASTERFILE 09	69
13.	Монитор 16/48. Команды монитора	75
14.	MONS 4	77
15.	GENS 4	80
16.	ZEUS	89
17.	LAZER-BASIC	94
18.	BETA-BASIC	103
19.	MEGA-BASIC	107
20.	ARTSTUDIO	116