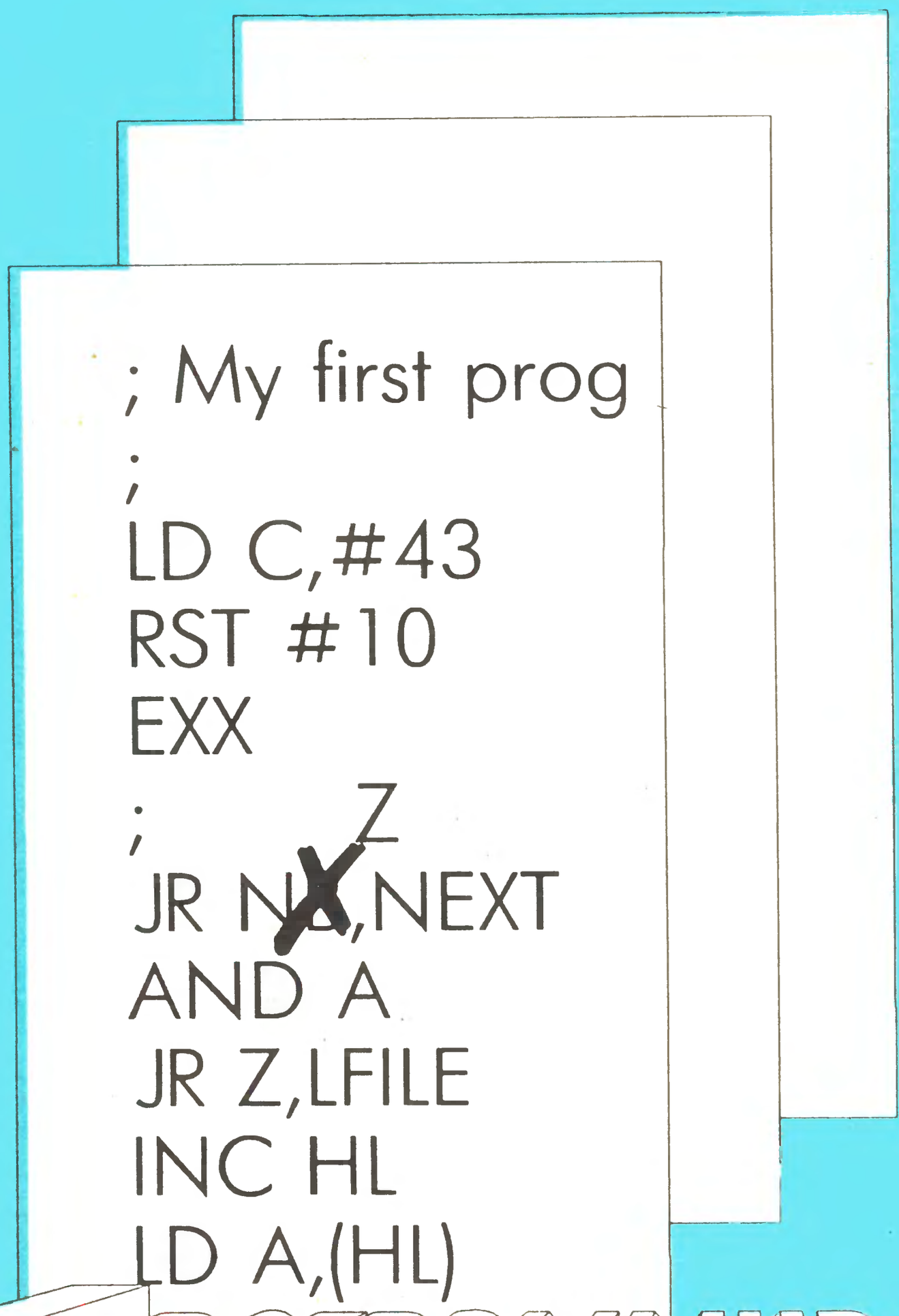


Операционная система IS-DOS для ZX-SPECTRUM

Программирование в среде IS-DOS

Руководство программиста

"ИскраСофт" С-Пб.
"Слот" Москва
1994

П Р О Г Р А М М И Р О В А Н И Е В С Р Е Д Е I S - D O S

РУКОВОДСТВО ПРОГРАММИСТА, РАЗРАБАТЫВАЮЩЕГО ПРОГРАММЫ НА ЯЗЫКЕ "АССЕМБЛЕР" В ОПЕРАЦИОННОЙ СИСТЕМЕ IS DOS.

М О С К В А 1 9 9 4

С О Д Е Р Ж А Н И Е

1. В В Е Д Е Н И Е

Фирма IskraSoft в лице официального дистрибьютера - фирмы Slot - представляет Вашему вниманию описание нового программного продукта - системы программирования на языке ассемблер для SPECTRUM-совместимых компьютеров, работающих под управлением уже получившей широкое признание операционной системы IS-DOS.

IS-DOS ассемблер ориентирован на создание библиотек объектных модулей и структурное программирование. В его задачу входит создание из исходного текста объектного модуля, пригодного для создания прикладных библиотек и программ, которые затем с помощью компоновщика (или редактора связей) LINK позволяют получить исполняемый модуль.

Совместимость на уровне текстов программ с популярной версией ассемблера GENS-4, возможность создания и редактирования текстов программ с помощью полноэкранного редактора IS-DOS, высокая скорость компиляции и компоновки делают IS-ASSEMBLER удобным как для профессионалов, так и для начинающих программистов.

Кроме описания IS-DOS ассемблера в данной книге подробно рассмотрены стандартные соглашения и базовые понятия операционной системы IS-DOS, ядро системы с точками входа, оконная технология, файловая система и другие интересные возможности при программировании в IS-DOS. В качестве справочного пособия приводится полный список команд и инструкций микропроцессора Z-80, а также приведен полный перечень ключей программ AS.COM и LINK.COM системы программирования IS-DOS ассемблер.

2.ЭЛЕМЕНТЫ И ОСНОВНЫЕ КОНСТРУКЦИИ ЯЗЫКА.

2.1.Формат оператора ассемблера.

Исходная программа состоит из строк-операторов или комментариев, набирается в IS-DOS редакторе (одно из достоинств и удобств). Каждая строка содержит не более одного оператора языка ассемблер и имеет длину не более 256 символов (включая управляющие коды). Ограничителем строки служит код 13 (конец строки). Пустые строки допустимы и не влияют на процесс трансляции.

Оператор может содержать до четырех полей. Назначение этих полей определяется порядком их появления и/или разграничительными знаками между ними.Формат оператора имеет вид:

[МЕТКА:][ОПЕРАЦИЯ ОПЕРАНДЫ][;КОММЕНТАРИЙ]

Двоеточие после метки не обязательно, если метка начинает строку (не отделена от начала пробелами или табулятором).

Поле метки и поле комментария не обязаны входить в строку вместе друг с другом , с полями операции и операндов, но наличие и вид поля операндов строго определяется полем операции.

2.1.1.Поле метки.

В поле метки могут находится метки, константы (символы, определенные с помощью директивы EQU) и команды управления листингом трансляции (о них см. ниже).

Метка - это определяемый пользователем символ (имя), которому присваивается текущее значение счетчика адреса. Счетчик адреса является средством, с помощью которого ассемблер определяет адреса памяти для размещения инструкций Z80 и областей данных по мере трансляции операторов исходной программы.

Таким образом, метка является средством символического описания ячейки памяти, на которую можно ссылаться в программе, написанной на ассемблере.

Если метка имеется, она всегда стоит первой в операторе и, в соответствии со стандартом для полей ассемблера, должна начинать строку. Но иногда оказывается полезным выделить в тексте небольшую область программы, сместив начало оператора пробелами.Это разрешается делать, ограничивая метку двоеточием. Это же относится и к константам.

Значение, присвоенное метке, может быть сразу переопределено в этой же строке директивой прямого присваивания EQU (см. п.4.). В этом случае символ будет определен как константа. Но любой символ, уже определенный в одной из строк транслируемого модуля, не может быть переопределен в другой строке в пределах области видимости данного символа (об области видимости символа п.3.2.) Это вызывает появление ошибки *Error*4.

Следующие знаки кодировки ASCII допускаются в символах-метках и символах, определяемых через директиву EQU:

1.Точка, код #2C.
2.Цифры от 1 до 9, коды #31-#39. Но цифрами нельзя начинать метку.
3.Заглавные буквы латинского алфавита от A до Z, коды #41-#5A.
4.Специальные знаки: $ [\ ] ^ _ `, коды #5B-#60.
5.Строчные буквы латинского алфавита от a до z, коды #61-#7A.

Также допускаются знаки русского алфавита альтернативной кодировки ГОСТа (если не действует ключ трансляции /RUS или директива .RUS-)

6.Заглавные буквы русского алфавита от А до Я, коды #80-#8F и #F0.
7.Строчные буквы русского алфавита от а до я, коды #A0-#AF, #E0-#EF и #F1.

При встрече команды управления листингом ассемблер сразу выполняет ее, а, встретив определяемый символ, запоминает его в буфере до анализа поля операции.

2.1.2.Поле операции.

Это поле может содержать мнемоническое изображение инструкции Z80, директиву ассемблера или макрокоманду.

Оно должно всегда отделяться от поля меток хотя бы одним пробелом, табулятором или, в случае присутствия в нем метки или константы, допустимо отделение одним двоеточием. Поле операндов отделяется от поля операции минимум одним пробелом или табулятором, но, в случае отсутствия операндов, для отделения поля операции от поля комментария достаточно ввести знак ";".

Если поле не содержит директиву EQU, ассемблер определяет символ в поле метки как метку, т.е. присваивает ему значение программного счетчика, а в противном случае сразу запускается директива EQU, и символ определяется как константа.

Далее происходит либо запуск на выполнение директивы, либо анализ поля операндов для мнемоники команды Z80. Но в любом случае возможные при дальнейшем анализе ошибки и прекращение трансляции данной строки не приведет к игнорированию определения идентификатора поля метки. Это относится и к связанной с данным полем ошибке *Error*2 - мнемоника не распознана.

2.1.3.Поле операнда.

Для операторов ассемблера - мнемоник инструкций Z80, операндами может быть одна из допустимых комбинаций из круглых скобок, выражений и символов регистров Z80, где круглые скобки используются для задания типа адресации.

Ассемблер поддерживает трансляцию всего стандартного набора мнемоник Z80, но накладывает определенное ограничение на стандарт записи операндов, содержащих индексные регистры IX и IY. Так в аргументах типа (IX+выражение),(IY+выражение) символы регистров обязаны стоять на первом месте, сразу после открывающих круглых скобок.Это пришлось сделать из-за усложнения, по сравнению с GENS4, структуры выражений, допустимых в ассемблере.

После анализа поля операндов ассемблер окончательно идентифицирует команду Z80 с помощью таблицы постоянных символов языка. При неудачной идентификации возможны следующие сообщения об ошибках:

*Error*29 - мнемоника поля операции не допускает операнды.

*Error*30 - операнды не подходят мнемонике поля операции.

Для операторов ассемблера - директив, вид операндов и их число определяется для каждой директивы по-своему (см. п.4.).

Аргументы в поле операнда отделяются друг от друга запятой.

2.1.4.Поле комментария.

Поле комментария не воздействует на процесс трансляции и может начинаться в любом месте строки со знака ";". После этого поля в строке уже не может идти никаких полей. В поле комментария можно использовать любые символьные знаки.Ассемблер лишь проверяет наличие кода 13 для завершения комментария и начала новой строки.

3.ЭЛЕМЕНТЫ ДАННЫХ И ВЫРАЖЕНИЯ.

3.1.Набор знаков языка ассемблера.

Ассемблер работает с перечисленными ниже знаками символьных кодов ASCII и знаками альтернативной кодировки.

Знаки, используемые в символьных именах (символах) языка.

1.Латинские буквы и буквы кирилицы (кирилица допустима если не действует ключ /-rus или директива .NRUS) .
2.Цифры от 0 до 9 (символ не может начинаться с цифры).
3.Знаки:

$ - знак денежной единицы (входит в символ только вместе с другими знаками);
[- правая квадратная скобка;
] - левая квадратная скобка;
\ - обратная дробная черта;
^ - стрелка вверх;
_ - подчеркивание;
  - знак фунта стерлингов.

4."." - точка, зарезервирована для использования в символах системных программ и библиотек.

Специальные знаки:
5.Знаки, используемые в термах выражений ассемблера.

  - Цифры от 0 до 9 и буквы A B C D E F; используются в числах;
$ - знак денежной единицы; представляет собой ссылку на программный счетчик адресов;

" - парные кавычки;индикатор символьного кода для двух или одного символа;

6.Разделительные и ограничительные знаки:
" ",TAB - (коды 32 и 9) пробел и табулятор; необходимы как разделители между полями строки языка ассемблера; в выражениях могут лишь отделять символьные имена друг от друга, а аргументы в поле операндов не отделяют, и служат для выделения фрагментов текста для обеспечения лучшей читаемости программы;

, - запятая; необходима для отделения аргументов команд и директив в поле операндов;

<> - парные угловые скобки; используются в любом месте программы для выделения выражения, которое будет обрабатываться как терм;

: - двоеточие; необходимо для определения метки, если она отделена от начала строки пробелами или табуляторами;

7. Знаки операций.
В языке ассемблера допустимы унарные и бинарные операции. Унарные операции выполняются над одним термом(аргументом или операндом). Унарная операция относится к тому терму, которому она предшествует.
+ - унарный плюс и бинарное сложение;
- - унарный минус и бинарное вычитание;
2^ - двойка и стрелка вверх; унарная операция возведения двойки в степень;
* - звездочка; арифметическое умножение;
/ - дробная черта; арифметическое деление;
& - коммерческое "И"; поразрядное логическое "И";
@ - коммерческое "ЭТ"; поразрядное логическое "ИЛИ";
! - восклицательный знак; поразрядное логическое исключающее "ИЛИ"(XOR);

8. Знаки временного изменения основания системы счисления числа.
% - процент; указатель двоичного числа;
# - решетка; указатель шестнадцатеричного числа;

9. Прочие знаки.

()- левая круглая скобка и правая круглая скобка; используются в аргументах команд Z80 для задания режима адресации и определения конкретной команды Z80 в соответствии со стандартом, принятым для мнемонического обозначения команд Z80 .

3.2.Символы языка ассемблера.

Символы языка строятся из знаков, приведенных в п. 3.1., и об этом уже говорилось частично в п. 2.1.1...
В языке имеются символы двух типов: постоянные символы и символы, определяемые пользователем. Соответственно этому ассемблер имеет таблицы символов трех типов:

PST - таблица постоянных символов,
UST - таблица символов пользователя

PST содержит все постоянные символы языка: мнемоники инструкций и регистров Z80, директив и команд ассемблера. Эти символы не нуждаются в своем определении в программе, т.к. PST входит в сам транслятор.
Таблица UST заполняется при трансляции исходной программы в момент определения соответствующих символов и построена по принципу бинарного дерева без оптимизации его ветвей, что позволяет достичь хорошего быстродействия при трансляции. В связи с этим не рекомендуется располагать определяемые символы на протяжении достаточно больших участков текста программы в строго алфавитном порядке (чтобы избежать вырождения бинарного дерева в линейный список и замедления поиска).
Удобным способом избежать расположения меток в алфавитном порядке явилось бы применение локальных цифровых символов типа 1$, 2$ и т.д., которые планируется реализовать во второй версии ассемблера.
Символы, определяемые пользователем, могут использоваться как метки, либо как константы, определяемые че-

рез директиву прямого присваивания EQU.

Кроме того, возможно их определение как глобальных или локальных символов, различающееся по области видимости.

Обычным образом определенные буквенно-цифровые символы интерпретируются ассемблером как локальные и видны лишь внутри транслируемого модуля.

Если в одном модуле будет определено несколько локальных символов с одинаковыми первыми 6 знаками, то это вызовет ошибку 4. В этом случае верным будет считаться первое определение, однако, второй проход трансляции будет блокирован и объектный код не будет создан.

Удобно бы было использовать локальные цифровые символы, область видимости которых ограничена двумя нецифровыми символами.

Информация о символах, начинающихся знаком "$", записывается в объектный файл и, таким образом, становится доступной компоновщику для использования при связывании объектных модулей в загрузочный.

При встрече двух одинаковых глобальных символов в разных модулях компоновщик также предупредит о многократном определении, но, в отличие от ассемблера, будет работать до конца, создав загрузочную программу со значением этих глобальных символов, равным значению в первом встретившемся из компонуемых модулей.

3.3.Числа.

Все числа в исходной программе интерпретируются в десятичной системе счисления,если нет специальных указаний относительно системы. Переполнение двухбайтового размера числа вызовет округление числа до двух байт и сообщение об ошибке номер 51, если не возникнет более серьезная ошибка 44 (см.ниже).

Два специальных знака % и # меняют систему счисления числа, перед которым они стоят,следующим образом:

% - двоичное число. Переполнение свыше 16 цифровых знаков вызовет ошибку 44. При этом прекращается вычисление выражения, в котором стоит число, и его результат остается непредсказуемым.

- шестнадцатиричное число. Переполнение свыше 16 цифровых знаков вызовет ошибку 44,как и в предидущем случае, а свыше 4 знаков - ошибку 51. В последнем случае результат определится по младшим 4 цифровым знакам, что соответствует двухбайтовому числу.

3.4.Термы.

Терм является компонентой выражения. Он может быть:

1.Числом (со значением в пределах 0-65536).

2.Символом; символы интерпретируются согласно следующим правилам:

- $ означает использование текущего счетчика адресов программы;
- символ, определенный пользователем, помещается в таблицу символов пользователя (UST);используется значение символа из этой таблицы; это может быть метка, либо константа, определенная через директиву EQU;
- неопределенному символу, встретившемуся в поле операнда команды Z80 или директивы размещения блока данных присваивается значение 0 и перемещаемость внешнего символа; если такой символ определен как локальный, то он вызовет сообщение об ошибке 50.
- неопределенный символ, встретившийся в поле операнда директивы EQU, вызовет сообщение об ошибке 13 и прекращение вычислений данного выражения, если на втором проходе потребовалось использование символа, стоящего в поле метки данной директивы EQU; значение такого символа и выражения с ним считается произвольным;

3.Одним либо двумя знаками в символьном коде ASCII или альтернативной кодировке, заключенными в кавычки, например:"A","Яй" или "┼┼";

4.Выражением, заключенным в угловые скобки; любая часть выражения, заключенная в угловые скобки обрабатывается прежде, остальной части; угловые скобки используются для изменения порядка слева направо обработки выражений (для отличия a+b*c от a+<b*c>)или для придания знака унарной операции всему выражению (например, -<a+b+c> или 2^<a+b> ,где 2^-унарный оператор степени);

3.5.Выражения.

Выражение является комбинацией термов, разделенных знаками операций. Принята инфиксная форма записи выражения т.е. когда знаки операции располагаются между своими термами, а для нескольких следующих подряд унарных операторов допускается префиксная запись, когда каждый унарный оператор действует на свой аргумент, стоящий

справа (см. примеры в этом же пункте). При нарушении такой формы записи возникает ошибка 33. Пробелы в выражении служат не обязательными разделителями термов, и их число не ограничено.

Несколько слов скажем об отличиях в действии унарных и бинарных операторах, затем назовем все допустимые операторы выражений ассемблера и, далее, поговорим о перемещаемости выражений.

Выражения обрабатываются слева направо, за исключением унарных операций, которые имеют больший приоритет обработки, например, для унарного оператора 2^:

```
2+2^3*2     дает значение 20
2+8*2       дает значение 20
```

Каждый составной терм выражения обрабатывается целиком, что позволяет менять установленный порядок выполнения операций слева направо с помощью угловых скобок, например:

```
5*< 4+ < 3-<2@1> > > = 20
```

Глубина стека калькулятора ограничена 64 словами, что на практике вполне достаточно.

Запись подряд нескольких унарных операторов интерпретируется ассемблером как префиксная запись, например:

2+2^2^3 эквивалентно 2 + < 2^<2^3> > ,

что дает значение 258; здесь аргументами бинарного оператора "+" являются 2 и 2^2^3, аргументом первого унарного 2^ - 2^3, а аргументом второго унарного оператора 2^ будет число 3.

2+2^3^4 эквивалентно 2 + < 2^<3^4> >,

что вызовет ошибку 27, т.к. оператора 3^ не существует в ассемблере.

-2^10+24 эквивалентно < -<2^10> > + 24 , что дает значение 1000;

Знаки + и - интерпретируются как знаки унарных операций только, если они стоят в начале терма, или выражения, например:

в строке LD HL,-ABC-10 первый минус унарен и его аргументом является символ ABC, а второй минус бинарен и его аргументы - это -ABC и 10;

в выражении 2^-<-ABC> первый минус бинарен, а второй унарен; поскольку перед бинарным минусом стоит оператор 2^, а не аргумент, то этим нарушается инфиксная скобочная форма записи выражений, что приведет к ошибке 33; избежать ошибки можно, сделав первый минус унарным:

```
2^< -<-ABC> >
```

В выражениях можно использовать следующие знаки унарных операций (операторов):

+ унарный плюс, не меняет знака терма (+A есть само A);

- унарный минус, меняет знак терма (-A есть противоположное значение A в дополнительном коде), например: -1 есть 65535 ;

2^ степень двойки, возводит число два в степень, заданную значением терма, например, 2^10 есть 1024.

и следующие бинарные операторы:

+ сложение;
- вычитание;
& поразрядное логическое "и";
@ поразрядное логическое "или";
! поразрядное логическое исключающее "или";
* умножение;

```
/  деление; результат целый;
?  функция MOD: A?B=A-< <A/B>*B >
```

Примеры выражений, при условии, что step1 равна 0, step2 равна 1, а $MET не определена :

выражение	результат
2^step1 @ 2^step2	3
#1234?256+<#1234/256*256>	#1234
-<1+2*3-4>	-5
"Y"-"A" + 1	26
%1000000001 ! 65535	%1111110111111110
"ab"&<2^5*256+2^5!65535>	"AB"
#100-$MET*2	0

В последнем примере результат выражения временно полагается равным 0 до его корректировки компоновщиком.

Все вычисления ведутся в двоичном дополнительном 16 разрядном коде. Поэтому значение выражения не может превосходить 16-разрядной величины.

Предупреждения об ошибках переполнения возникают только, в операции умножения, если переполнение может привести к неверному знаку результата (при превышении результата умножения двух соответствующих положительных величин значения 32767= =#7FFF).(см.пример в описании ошибки 15). Возникновение ошибки переполнения в аналогичной ситуации при сложении и вычитании (изменение знака) не идентифицируется, т.к. дополнительная кодировка гарантирует верность 16 младших разрядов результата этих операций, как в интерпретации чисел без знака , так и в знаковой их интерпретации. Но при комбинировании сложения и вычитания с делением следует соблюдать осторожность, т.к. деление предполагает только знаковую интерпретацию чисел (когда числа > 32767 интерпретируются как отрицательные) . Например:

#1+#FFFF-#FFFE = #10000-#FFFE = 2 ,

в беззнаковой интерпретации,но в знаковой будет то же самое:

1+ <-1>- <-2> = 0- <-2> = 2

В связи с этим бывает полезно иметь возможность определять знак терма, чтобы правильно использовать операции деления и умножения. Такая возможность есть в ассемблере. Побочным эффектом деления на ноль является вычисление некой обратной знаковой функции:

```
A/0= -1  для A >= 0
A/0= +1  для A <  0
```

Воспользовавшись этим, можно, например, корректно работать с неким адресом буфера $BUF при вычислении его старшего байта $HBUF, не зависимо от конкретного значения (знака числа $BUF):

```
$HBUF  EQU  <$BUF-#8000/256+#80>*
*<$BUF/0+1/2> + <$BUF/256*<-$BUF/0+1/2>>
;для $BUFотриц.      для $BUF полож.
```

На самом деле выражение должно быть записано в одну строку.

Здесь терм <$BUF/0+1/2> равен 0 для $BUF<=#7FFF, и он равен 1 при $BUF>#7FFF, а для терма <-$BUF/0+1/2> все будет как раз наоборот. В результате, используя данные термы в качестве сомножителей, можно вычислять $HBUF по разным формулам для положительных и отрицательных значений $BUF.

3.5.1.Перемещаемость при вычислении выражений.

Вычисление выражения включает определение его перемещаемости, т.е. зависимости его значения от изменения адреса компоновки (и загрузки) модуля, в которое входит данное выражение.

Результирующее значение выражения может быть любым из четырех типов, описываемых ниже:абсолютным, относительным, внешним или сложным относительным.

Выражение (и его значение) является абсолютным,если его значение фиксировано, не зависит от адреса компоновки.

Следующие выражения имеют абсолютные значения:

- выражения, термы которых являются числами или знаками символьного кода;
- относительное выражение или терм минус относительный терм, поскольку такое выражение преобразуется ассемблером к концу анализа выражения в единичный терм - абсолютное число;

В листинге трансляции такие выражения не помечаются, т.е. после их значения в поле кода листинга стоит пробел.

Например, в следующем фрагменте, значения выражения MET2-MET1 и значение аргумента команды короткого безусловного перехода будут абсолютными:

```
адр   код     стр  текст программы
 0  18  3       1        JR   MED2
 2               2 MET1
 2  21  3  0     3        LD   HL,MET2-MET1
 5               4 MET2
```

Выражение является относительным, если для коррекции его значения при компоновке достаточно добавить к нему базовый адрес модуля, в котором это выражение встретилось. Такими являются следующие выражения и термы:

- выражения, состоящие из одного простого терма-метки или терма-символа программного счетчика адресов.
- относительное выражение плюс (минус) абсолютный терм.

Значения таких выражений, выводимые в листинге в поле кода,помечаются знаком r.

Информация о них занимает дополнительно к 1 или 2 байтам, индицируемым в поле кода листинга, еще 2 байта.

```
 адр код стр текст программы
 0   0  0  0    1 M1     DEFS 160
A0  21  0  0r   2 M2     LD   HL,M1
A3  11 A6  0r   3        LD   DE,M3+<M2-M1>
A6   0  0  0    4 M3     DEFS M2-M1
```

Выражение является сложным относительным, если выполняется одно из следующих условий:

- величина, получаемая в результате вычисления выражения,требует более, чем одно перемещение; например, если относительные термы MET1 и MET2 образуют выражение вида MET1+MET2, то будут выполнены два перемещения компоновщиком, (т.е. к значению выражения будет добавлен адрес компоновки модуля два раза);
- перемещение должно быть выполнено с обратным знаком, т.е. значение адреса компоновки модуля должно вычитаться, из значения выражения один или несколько раз, например -MET1, где MET1 - относительный терм;

Значения таких выражений, выводимые в листинге в поле кода,помечаются знаком m, и им временно присваивается нулевое значение до окончательной обработки их компоновщиком. Информация о них занимает дополнительно к 1 или 2 байтам, индицируемым в поле кода листинга, еще 6 байт.

Выражение является внешним в следующих случаях:

- оно содержит неопределенный в данном модуле символ (предполагается, что он будет определен как глобальный символ в другом модуле);
- операция, отличная от сложения, вычитания или изменения знака, применяется к относительной величине.

Значения таких выражений, выводимые в листинге в поле кода,помечаются знаком g, и им временно присваивается нулевое значение до окончательной обработки их компоновщиком.

Такие выражения не могут быть вычислены ассемблером и записываются в объектный файл в постфиксной форме для передачи компоновщику, практически целиком. Если же левые термы выражения не являются внешними, то они будут вычислены и выражение запишется в объектный файл в укороченном виде (это не касается выражений с неопределеными символами). Например, следующее выражение будет частично вычислено ассемблером: LENG2-LENG1+BUFF/256 , где LENG1 и LENG2 - относительные термы; а это запишется в объектный файл целиком:

```
BUFF/256+LENG2-LENG1 ,
```

т.к. терм <LENG2-LENG1> не является внешним (он абсолютный) и, поэтому, может быть вычислен, если стоит в выражении слева.

3.6.Программный счетчик адресов.

Символ $ зарезервирован для использования в качестве идентификатора текущего счетчика адресов в программе на ассемблере.

В начале каждого прохода его значение полагается равным нулю. После трансляции любого оператора ассемблерной строки он получает значение, равное сумме предыдущего и длины оттранслированного оператора (команды Z80 или блока данных). Кроме того, в директивах DEFB и DEFW, он получает соответствующее приращение при трансляции каждого аргумента этих директив.

Таким образом, при использовании этого символа в поле операнда инструкции Z80 он обозначает адрес первого байта инструкции. При использовании его в поле операнда директивы ассемблера он представляет адрес текущего байта, слова или блока данных (в директиве DEFS).

4.ДИРЕКТИВЫ АССЕМБЛЕРА.

Кроме макрокоманд и команд Z80 в поле операции могут находиться директивы ассемблера, еще называемые псевдокомандами, так как они не вызывают генерацию кодов команд Z80. Директивы выполняют определенные действия по управлению трансляцией и выводом листинга, а также определяют и размещают в объектном файле константы и блоки данных.

Директивы бывают как с аргументами, так и без них. Их вид полностью определен для каждой в таблице постоянных символов, и может быть изменен пользователем только с помощью написания соответствующих макроопределений. При наличии аргументов в директиве, они должны отделяться от мнемоники хотя бы одним пробелом или табулятором.

4.1.Директивы с синтаксисом GENS4.

ENT выражение

- игнорируется.

ORG выражение

- помещает значение выражения в качестве адреса компоновки по умолчанию в описатель объектного файла. Выражение не должно быть внешним и должно иметь абсолютное значение, иначе, при трансляции этой директивы, возникнет ошибка 42.

EQU выражение

- присваивает значение выражения символу, стоящему в поле метки. Выражение может содержать неопределенные ранее символы (метки и "EQU" константы), но все они,так или иначе, должны быть определены в транслируемом модуле (т.е. как метки и "EQU" константы) к концу трансляции .

  В противном случае возникнет ошибка 13 в строке, где используется этот определенный через EQU символ.

4.1.1.Директивы размещения блоков данных.

DEFB выражение[,выражение,...]

- последовательно размещает 8-разрядные значения выражений, начиная с адреса,указанного программным счетчиком адресов($), увеличивая его значение на 1 для каждого нового выражения.

DEFW выражение[,выражение,...]

- последовательно размещает 16-разрядные значения выражений, начиная с адреса,указанного программным счетчиком адресов($), увеличивая его значение на 2 для каждого нового выражения.

DEFM "string"

- последовательно размещает коды символьных знаков, заключенных в парные ограничители, начиная с адреса, указанного программным счетчиком ($).

  Вместо кавычек допускается использование в качестве парных ограничителей любого другого печатного знака, кроме знаков-разделителей.

DEFS выражение

- резервирует заполненную нулями область памяти, с длиной, равной значению выражения.

Выражение не должно быть внешним и должно иметь абсолютное значение, иначе, при трансляции этой директивы, возникнет ошибка 42.

4.1.2.Директивы условной трансляции.

Эти директивы позволят Вам управлять включением в объектный модуль отдельных фрагментов ассемблерной программы или их исключением, в зависимости от некоторых условий, заданных значением выражения.

IF выражение
- выключает трансляцию, если значение выражения равно 0. Во всех остальных случаях на трансляцию не влияет.

ELSE - выключает трансляцию, если она работала и, наоборот, включает, в случае, если она была подавлена.

END - включает трансляцию.

4.2.Новые для GENS4 стандарта директивы.

Все директивы, не вошедшие в GENS4, (а также системные макросы и системные глобальные символы) начинаются с точки. Это сделано для того, чтобы избежать совпадения идентификаторов последующих версий ассемблера и системных библиотек с идентификаторами пользовательских программ и библиотек. В связи с этим не рекомендуется использовать точку в глобальных символах и идентификаторах макросов Ваших программ.

.RUS- запрещает использование русских букв в символах, определяемых пользователем (в метках, константах и макрокомандах) до встречи директивы .RUS+.
Русские буквы будут вызывать появление ошибки *Error*22.

.RUS+ разрешает использование русских букв в символах, определяемых пользователем (в метках, константах и макрокомандах) до встречи директивы .RUS-.

4.2.1.Директивы, управляющие листингом трансляции.

.CODE выражение
определяет поле данных в листинге трансляции равным значению выражения. Выражение должно иметь абсолютное значение (противный случай вызовет ошибку *Error*42). Старший байт значения выражения игнорируется, а младший ,при значении > 52*3, заменяется на это значение.

Следующие две директивы предназначены для включения заголовков или другого текста в листинг трансляции в удобной для чтения форме и для разделения подмодулей в листинге. При этом не происходит вывода этих строк в обычном форматированном по полям виде, что исключает засорение выводимого текста ненужной информацией.
Но если трансляция запрещена, то эти директивы игнорируются, а наличие форматированной строки программы в листинге определяется, как обычно,действием команд *IF+(-) или ключей /+if(/-if) и др.

.EMPTY выражение
выдает в листинг трансляции число пустых строк, равных значению выражения при этом строка с этой директивой исключается из листинга.
Выражение должно иметь абсолютное значение (противный случай вызовет ошибку *Error*42). Старший байт значения выражения игнорируется, а младший, при значении 0 интерпретируется как 256.

.HEAD string
выдает в листинг трансляции строку текста string,причем она будет начинаться прямо в первом поле листинга, и продолжаться в других полях.

5.КОМАНДЫ, УПРАВЛЯЮЩИЕ ЛИСТИНГОМ ТРАНСЛЯЦИИ.

Эти команды находятся в поле метки, не имеют аргументов и всегда начинаются со знака *. Они работают на втором проходе ассемблирования, изменяя формат вывода листинга на экран или принтер, запрещая и снова разрешая вывод, реализуя паузу в нужном программисту месте,все это - прямо по ходу трансляции.

*L+ - включает листинг.

*L- - подавляет листинг.

*IF+ - разрешает выдачу строк блоков условной трансляции даже, если трансляция была выключена одной из директив условной трансляции.

*IF- - исключает из листинга строки блоков условной трансляции, если трансляция была выключена.

*A+ - включает в формат вывода строки листинга адрес соответствующей этой строке команды Z80 или блока данных (текущее значение программного счетчика).

*A- - исключает из формата вывода строки текущее значение программного счетчика.

*C+ - включает в формат вывода строки поле данных (код оттранслированной команды Z80 или блок данных). Длина этого поля может быть изменена директивой управления листингом (.CODE выражение) по ходу трансляции и ключом /codeчисло на все время трансляции.

*C- - исключает из формата вывода строки поле данных.

*N+ - включает в формат вывода строки ее номер во входном текстовом файле.

*N- - исключает номер из формата вывода строки.

*T+ - включает в формат вывода строки текст из входного файла.

*T- - исключает текст из формата вывода строки.

*H+ - приводит к установке 16-ричного формата вывода адреса инструкций или блоков данных в листинге.

*D- - то же самое. Команда введена для совместимости с GENS4.

*D+ - приводит к установке 10-ричного формата вывода адреса инструкции или блока данных.

*O+ - приводит к установке 8-ричного формата вывода адреса инструкции или блока данных.

*S - при встрече этой команды транслятор переходит в режим ожидания нажатия любой клавиши, давая возможность лучше рассмотреть листинг. SS A приводит к прекращению трансляции.
Работает только при включенном листинге. Во избежание лишних остановок при выводе на принтер, для отмены действия этих команд пользуйтесь ключом /-stop.

6.СООБЩЕНИЯ ОБ ОШИБКАХ ТРАНСЛЯЦИИ И РЕАКЦИЯ АССЕМБЛЕРА НА НИХ.

6.1.Ошибки, связанные с невозможностью начать трансляцию.

При возникновении таких ошибок происходит выход в IS'DOS с кодом соответствующей ошибки.

23 - Использован неизвестный ключ в командной строке.
31 - Имя входного файла не введено или имя в командной строке содержит недопустимые символы или символы маски.
51 - Файл ассемблера as.com испорчен.

130 - Не хватает рабочей области памяти для загрузки и работы.
131 - Не инсталлирован драйвер принтера в режиме вывода листинга на принтер.

6.2.Ошибки, возникающие во время трансляции.

Таблица 3 представляет все ошибки трансляции, номера которых выдаются в листинг вместе со всеми его обычными полями. Показано, ошибки с какими номерами возникают на первом, а какие на втором проходе ассемблирования, и в каком поле ассемблерной строки возможно ее появление на данном проходе.

Ошибки с номерами 10,14,15,50,51 принято считать несерьезными. Возникновение таких ошибок на первом проходе трансляции не приводит к запрещению генерации объектного файла, чего нельзя сказать об остальных ошибках трансляции первого прохода.

Таблица 3.

Поле метки		Поле операции		Поле операнда		Краткий вариант расшифровки номеров ошибок
1	2	1	2	1	2	Номера прохода
2		2				Мнемоника не распознана
4						Многократное определение символа
7		7		5		Недопустимый символьный знак
				9	9	Регистр неверно использован
					10	Значение выражения недопустимо
					13	Ссылка из EQU на неопределенный символ
				14	14	Деление на 0
				15	15	Переполнение в умножении
20		20				Метка начинается с цифры.
21						Лишнее двоеточие
22		22		22		Русская буква
23						Такой команды нет.
		24				Недопустимый идентификатор
				25	25	нецифровой знак
				26		Непонятная кавычка
				27	27	Непонятный оператор степени
				28		Непонятная запятая
				29		Нет аргумента
				30		Аргумент не нужен
				31		Неподходящий аргумент
				32		число аргументов >52
				33		Не инфиксная запись или нет выражения
				40	40	Слишком длинное выражение
					41	Слишком длинная цепочка "EQU" ссылок
				42		Значение выражения - не абсолютное число
				43	43	Недопустимая степень
				44	44	Переполнение буфера числа
					45	Запрещенная "EQU"-ссылка.
					50	Символ не определен
				51	51	Переполнение числа

Далее последует подробное описание ошибок трансляции.

6.2.1.Ошибки со смыслом, близким к ошибкам трансляции в GENS4.

Error 2 - мнемоника не распознана.
Error 4 - метка или константа определена больше одного раза.
Error 5 - встретился недопустимый знак ASCII в выражении.
Error 7 - встретился недопустимый знак языка в поле метки или операции.
Error 9 - недопустимое использование регистров Z80.
Error 10 - численное значение выражения выходит за рамки допустимого для данной команды или директивы. Транслятор полагает выражение равным числу 0 и продолжает трансляцию строки.
Error 13 - ссылка из директивы EQU на не определенный вообще, за все время трансляции символ (в том числе и на внешний).

Эта ошибка возникает не в строке, где стоит EQU, а лишь на втором проходе, при трансляции строк команд Z80 или директив DEFW и DEFB, содержащих символы, определенные через такие EQU.
Error 14 - деление на ноль.В этом случае деление действует как обратная знаковая функция и результатом будет +1 для отрицательного делимого и -1 для положительного и нулевого.
Error 15 - переполнение операции умножения.

Транслятор отбрасывает старшую, не поместившуюся в два байта, часть числа и продолжает вычисление. При этом возможна не предусмотренная смена знака, и программист должен сам принимать решение о ее допустимости. Например, после следующего определения символа BUFF1:

BUFF1 EQU #2700*6 недопустимо его использование как некой положительной длины буфера в выражении типа BUFF2 EQU BUFF1/2, т.к.BUFF1 будет интерпретирован транслятором как отрицательная величина,#A200= -#5E00, в результате чего BUFF2 получит значение #D100=-#2F00, что никак не равно #2700*3=#5100.

С другой стороны, выражение BUFF1 может использоваться в смысле положительной величины, если с ним не производится операций деления или умножения,например:

BUFF1-1 ,будет верно как для положительной, так и для отрицательной интерпретации BUFF1 , а для случая: -1*BUFF1 верна лишь отрицательная интерпретация данного значения BUFF1(#2700*6);

6.2.2.Синтаксические ошибки, новые для GENS4 стандарта.

Error 20 - метка начинается с цифры.
Error 21 - неверное использование знака ":".
Error 22 - использование русских букв как допустимых знаков языка запрещено.(Это не касается случаев их использования в аргументе DEFM и в выражениях в конструкции с кавычками).
Error 23 - такая команда управления листингом отсутствует в ассемблере.
Error 24 - идентификатор с такой мнемоникой в поле операции недопустим. (Этот идентификатор зарезервирован под регистр Z80 или команду управления листингом).
Error 25 - встретился нецифровой знак в числе.
Error 26 - ошибка в выражении, в конструкции "*" или "**", где * - любой допустимый знак ASCII или альтерн.кодировки ГОСТа (не допустимая длина такой конструкции).
Error 27 - данный оператор возведения в степень в выражении - недопустим.
Error 28 - непонятная запятая в поле операнда.
Error 29 - отсутствует поле операнда, хотя мнемоника поля операции требует его наличия.
Error 30 - присутствует поле операнда, хотя и мнемоника поля операции не предполагает его наличия.
Error 31 - аргументы в поле операнда, не подходят мнемонике поля операции, а также в случае расположения регистра в аргументе команды Z80 не на первом месте, например:

```
SET   7,(shift+IX)
вместо стандартного
SET   7,(IX+shift)
```

Error 32 - превышение допустимого числа аргументов (52) в директиве.
Error 33 - отсутствует или не правильно построено выражение (не соблюдена инфиксная скобочная форма записи).

6.2.3.Новые ошибки, связанные с вычислением выражений.

В случае ошибок, начинающихся с номера 40, транслятор прекращает вычисление выражения которое теперь остается не определенным, равным произвольному числу, и переходит к новой строке. Результат трансляции в этом случае видно в строке листинга в поле данных(кодов Z80), независимо от текущего режима выдачи листинга.

Error 40 - переполнение буфера калькулятора. Возникает только, если выражение будет содержать > 256/4 операторов.

Error 41 - переполнение буфера EQU ссылок. Возникает только, если число не вычисленных за первый проход констант определенных через EQU в какой-либо цепочке ссылок достигнет числа > 55.

Error 42 - вычисленное значение выражения не является абсолютным числом, что не допустимо для данной директивы.

Error 43 - недопустимый аргумент оператора 2^.

Error 44 - переполнение буфера числа, при считывании его из выражения. Транслятор прекращает вычисление выражения которое теперь остается не определенным, равным произвольному числу.

Error 45 - запрещенная ссылка из директивы EQU на определяемый ей же символ.

Такая ссылка разрешена, если она случается в строке ,где и находится этот самый символ, например:

```
SYMB EQU SYMB+1
```

В этом случае символ SYMB сначала определяется как метка, и затем ее относительный адрес используется для ее же переопределения с помощью директивы EQU (как происходит и в GENS4).

Но в случае:

```
FULL EQU EMPTY+200
      ...
EMPTY EQU FULL-300 ,
```

в строке с командой Z80 или директивами DEFB,DEFW, где используется определение EMPTY или FULL, возникнет сообщение об ошибке 45, что предупредит возникновение неоднозначности определения символов.

В случае ошибок, начинающихся с номера 50, транслятор продолжает вычислять выражение, округляя переполнившееся значение или полагая неопределенное равным 0. В некоторых случаях сообщения об ошибках этого разряда можно рассматривать как "ВНИМАНИЕ!",если требуется выполнить определенные арифметические действия, например, умножение чисел без знака диапазона до 65536, вычислить знак терма с помощью деления на ноль или в случае, когда определенная метка в Вашей программе временно не используется.

Из ошибок в стандарте GENS4 к этому разряду относятся: 10, 14, 15.

Error 50 - неопределенный локальный символ в строке команды Z80 или директивы DEFW или DEFB.

Транслятор действует как и в случае с внешними символами(т.е.заносится информация о них в объектный файл, а значение выражения обнуляется), но дополнительно к этому предупреждает об ошибке.

Error 51 - значение числа, входящего в выражение превышает двухбайтовый размер. Это приводит к отбрасыванию транслятором не поместившейся старшей части числа.

7. ОБЩИЕ ПОНЯТИЯ И ТЕРМИНОЛОГИЯ, ПРИНЯТЫЕ В СИСТЕМЕ IS-DOS

Ядро - набор подпрограмм, резидентных в памяти. Ядро может настраиваться на различную конфигурацию.

Уровень ядра - некоторое подмножество подпрограмм, входящее в ядро. Множество всех уровней является непересекающимся множеством, в совокупности образующим ядро.

Минимальное количество уровней, обязательное для работы операционной системы, равно 3 (0-й, 1-й и 2-й), максимальное количество уровней, которое может быть создано и одновременно доступно разработчику прикладной программы, равно 8 (самый верхний - 7-й).

Обращение к ядру из программ пользователя или из других подпрограмм ядра обычно происходит через инструкцию ассемблера RST 16. При этом в регистре C должен находиться код функции, которую будет выполнять соответствующая подпрограмма соответствующего уровня ядра.

Уровни ядра загружаются транзитной командой SET.com с диска.

Номер Уровня	Диапазон кодов функций	Название	Назначение
0	0..1F	DOS	обеспечение взаимодействия с устройствами
1	20..3F	DUD	файловая служба
2	40..5F	COM	работа с каталогами и транзитными командами
3	60..7F	WIND	функции оконной технологии
4	80..9F	SHELL	создание удобной среды для интерактивной работы с файлами и каталогами дисковой внешней памяти (так называемую "файловую оболочку").

Уровень ядра 0 - набор подпрограмм для обеспечения взаимодействия с устройствами, драйверами и каталогами. Диапазон кодов функций - от 0 до 31.

Уровень ядра 1 - набор подпрограмм, образующих файловую службу. Диапазон кодов функций - от 32 до 63.

Уровень ядра 2 - набор подпрограмм, организующих работу с каталогами и транзитными командами. Диапазон кодов функций - от 64 до 95.

Уровень ядра 3 - набор подпрограмм, реализующих функции оконной технологии. Диапазон кодов функций - от 96 до 127.

Уровень ядра 4 - набор подпрограмм, создающих пользователю удобную среду для интерактивной работы с файлами и каталогами дисковой внешней памяти (так называемую "файловую оболочку"). Диапазон кодов функций - от 128 до 159.

Супервизор ядра - программа, расположенная в нулевом уровне и обрабатывающая инструкции RST 16. Она анализирует содержимое регистра C, в котором находится код выполняемой команды и либо выполняет соответствующую подпрограмму ядра, либо - в случае ошибки - возвращает управление программе, вызвавшей RST 16, помещая код ошибки в регистр A и устанавливая флаг C в регистре состояния процессора.

При выполнении инструкции RST 16 содержимое регистра C интерпретируется следующим образом : старшие 3 бита задают номер уровня ядра (отсюда пошло ограничение на количество уровней), младшие 5 битов задают номер функции в пределах конкретного уровня.

Если в настоящий момент в системе нет требуемого уровня, или в заданном уровне нет требуемой функции, то последствия будут самыми непредсказуемыми, ибо последние версии iS-DOS не следят за этим.

Т.о. из вышесказанного ясно, что в системе IS-DOS в качестве рестартов используется ТОЛЬКО RST 16 *с указанием функции в регистре "C" (сделано по типу INT 21 в IBM/MS-DOS)*

Устройство - аппаратурная компонента ПЭВМ. С точки зрения IS DOS устройства подразделяются на блочные устройства, символьные устройства вывода, символьные устройства ввода. Система IS DOS поддерживает до 8-и устройств каждого типа.

Драйвер - резидентная перемещаемая программа, которая может обслуживать одно или несколько устройств одного типа.

Драйверы загружаются транзитной командой set.com с диска.

Резидентная задача - программа, имеющая такую же структуру, как и драйвер, но не обслуживающая устройства. Может применяться для различных целей.

Резидентная задача запускается на выполнение из монитора или командного файла командой

@имя задачи [параметры]

Резидентные задачи, как и драйверы могут иметь имя, отличное от имени файла.

Фоновые задачи - задачи, выполняющиеся по прерыванию. Они вызываются на выполнение каждые 20 млс.

Цепочка фоновых задач - несколько фоновых задач, последовательно выполняемых по прерыванию. Рекомендуется, чтобы общее время выполнения всех задач из цепочки не превышало 15 млс. В цепочку фоновых задач рекомендуется помещать такие задачи, которые требуется вызывать на выполнение периодически через равные промежутки времени. Цепочка фоновых задач всегда состоит по крайней мере из двух задач: задачи обслуживания 4-х байтного счетчика времени и задачи опроса клавиатуры. Кроме того, периодически при необходимости включается задача мигания курсора.

Транзитные команды - команды операционной системы, обработчики которых находятся на диске (программы пользователя тоже могут рассматриваться как транзитные команды).

Рекурсивная цепочка - цепочка, описывающая вложенность файлов (например, когда программа вызывает на выполнение саму себя).

Диски - с точки зрения пользователя могут быть трех типов :

- архивные диски - предназначены для хранения данных, доступны только для использования в IS DOS и недоступны для TR-DOS (помечены, как занятые) - создаются программой is_tools.com;

- рабочие диски с монитором оболочки - предназначены для работы в IS DOS - создаются из архивных дисков копированием в главный каталог файла extend.txt, добавлением каталогов SHELL, UNIL, RES, EDIT, СЕРВИС и копированием туда необходимых утилит;

- загрузочные диски - предназначены для загрузки IS DOS - создаются из рабочих дисков добавлением файла с копией системы,подключением ее к диску транзитной командой connect.com, фиксирующей необходимые адресные ссылки на диске на файл копии системы для ее загрузки, и запуском программы boot.com на этом диске. Программа boot.com создает в 5-м секторе 0-й дорожки системный загрузчик из среды TR DOS.

Кроме того, можно создать файл autoexec.bat в корневом каталоге - он будет выполняться непосредственно перед выходом в монитор оболочки и осуществлять действия для создания необходимой пользователю комфортной среды для работы;

Канал - именованная непрерывная область памяти, предназначенная для временного хранения или передачи данных от одной программы в другую. Каналы находятся в специально отведенном месте памяти. Доступ к каналам из программ возможен только с помощью специальных обращений к ядру.

Любой канал имеет следующую структуру :

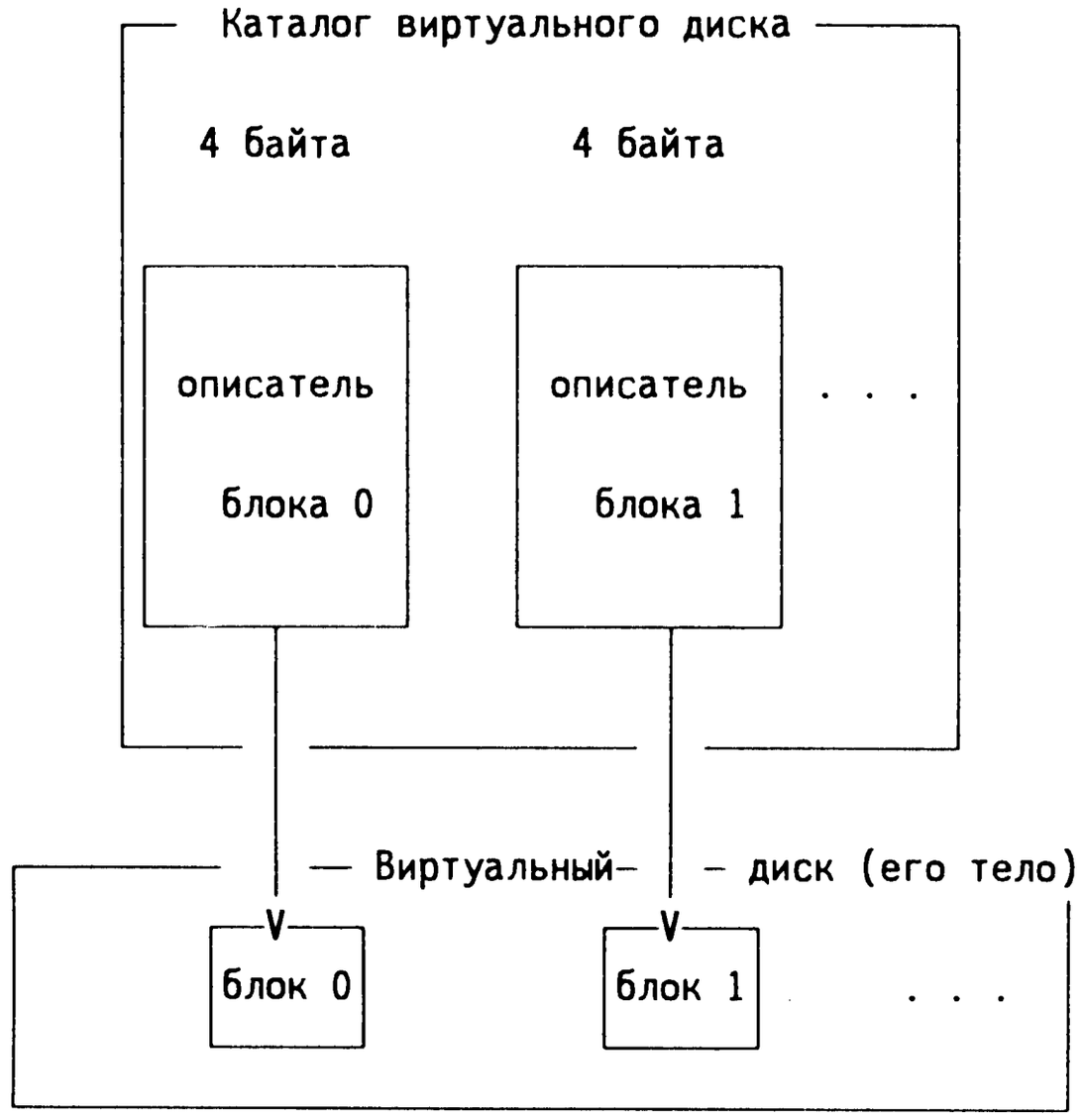
1 байт	2 байта	
логическое имя канала	длина канала	тело канала

Менеджер каналов - программа нулевого уровня, которая по требованию системы или пользователя выделяет новый временный канал, выбирая из каналов, выделенных менеджеру, свободны канал с наименьшим номером.

Каналы с определенными логическими номерами имеют следующее жестко фиксированное предназначение :
#00 ... #07 - для работы блочных устройств;

#08 ... #0F - для работы символьных устройств вывода;
#10 ... #17 - для работы символьных устройств ввода;
#18 ... #D7 - доступны менеджеру каналов, который может вызываться пользователем;
#D8 ... #E7 - зарезервированы для описания резидентных задач;
#E8 ... #EF - зарезервированы для описания драйверов символьных устройств ввода;
#F0 ... #F7 - зарезервированы для описания драйверов символьных устройств вывода;
#F8 ... #FF - зарезервированы для описания драйверов блочных устройств;

Виртуальный диск представляет собой быстродоступную копию некоторых наиболее часто используемых блоков блочных устройств. Он размещается в памяти и имеет следующую структуру:



Размер виртуального диска (количество блоков) в 0-м уровне ядра.

Описатель блока имеет следующую структуру :
1-й байт - регистр состояния блока, совмещенный с номером устройства:

7-й бит : блок модифицирован (1) или нет (0);
6-й бит : блок защищен (1) или нет (0);
5-й бит : блок существует (1) или нет (0);
4-й, 3-й биты : резервные;
2-й ... 0-й биты : номер устройства, которому принадлежит блок.

2-й байт - счетчик обращений к блоку;

3-й и 4-й (мл.,ст.) - номер блока на устройстве.

Минимальный размер виртуального диска равен 6 блокам.

Виртуальный диск создается системой автоматически, т.к. он необходим для работы файловой службы. Пользователь может изменять размер виртуального диска.

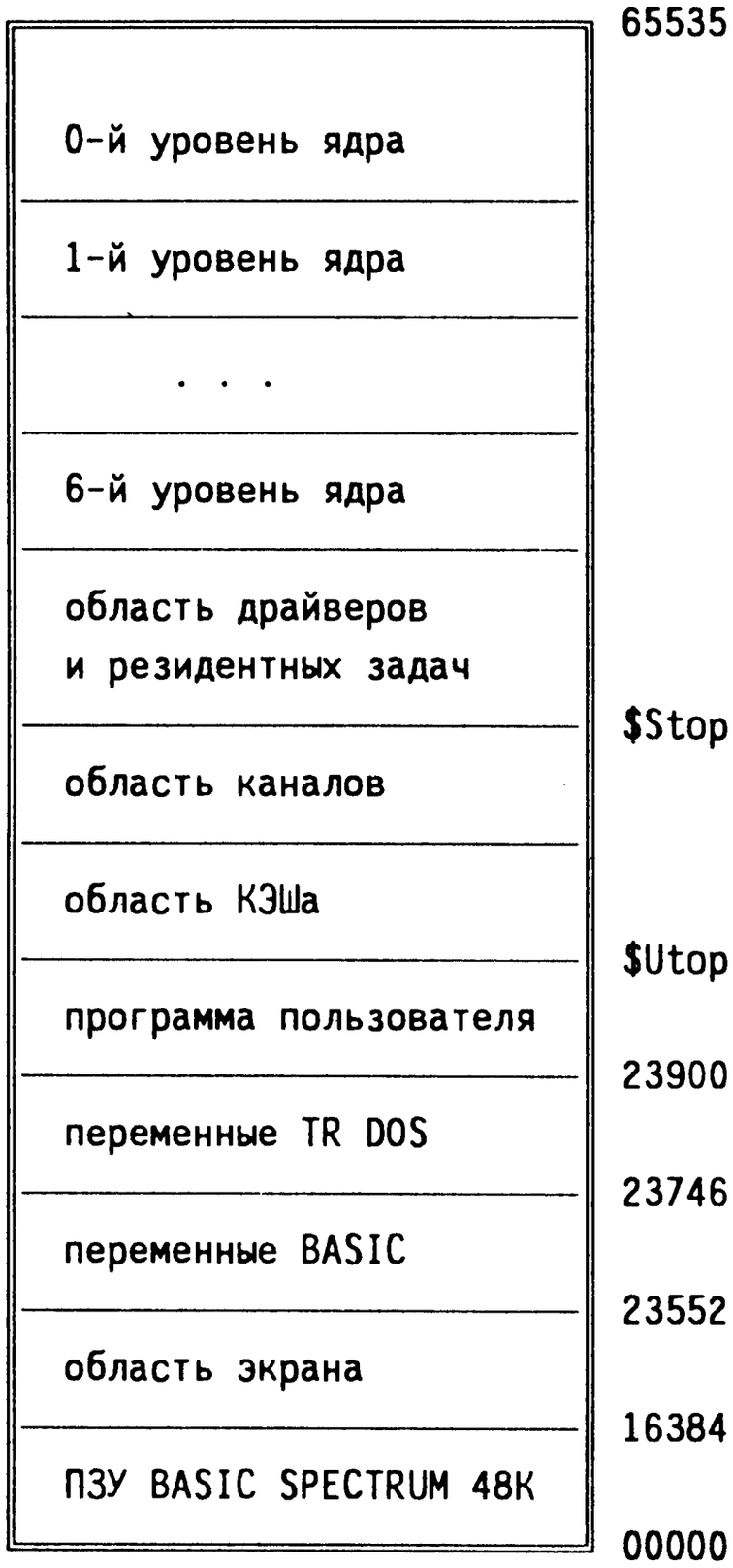
Файлы и каталоги в IS DOS могут быть двух типов - непрерывные и сегментированные. Непрерывным файлом является тот, у которого блоки идут без промежутков один за другим. Сегментированным называется тот, у которого блоки могут располагаться в любой области дискового пространства (по принципу заполнения "дырок"). Непрерывные файлы и каталоги можно только укорачивать, зато доступ к ним - быстрее и они занимают меньше места на диске. Если известны правила использования каталога, его лучше делать непрерывным и нужного размера.

Системное устройство - блочное устройство, на котором находятся основные транзитные команды, необходимые для работы системы.

Быстрое устройство - часть системного устройства, находящаяся на устройстве с быстрым доступом, например, на виртуальном диске. Туда рекомендуется помещать команды и программы наиболее часто используемые системой и пользователем. Поиск файла при запуске осуществляется сначала на быстром устройстве, а затем на системном (на дискете).

Текущее устройство - логическое имя, альтернативное имени системного устройства. Например, на дисковод "A" могут быть назначены логические имена "A" и "C".Может применяться, например, при операциях копирования для назначения устройства, на которое выполняется копирование.

Структура адресного пространства при работе IS DOS.



Драйверы и резидентные задачи являются перемещаемыми в ОП при их удалениях и добавлениях.

В некоторых версиях аппаратуры есть возможность заменить ПЗУ дополнительной областью памяти. В таком случае имеется возможность переместить туда часть ядра системы, драйверов и резидентных задач.

Рекомендации пользователю по написанию прикладных программ.

Если прикладная программа не может целиком разместиться в памяти, то рекомендуется разделить ее на несколько частей и общие функции этих частей вынести в главный модуль, который загружается в память и поручить ему управлять оверлеями. Для связи частей с главным модулем рекомендуется использовать 7-й уровень системы, зарезервированный для пользователя, рестарты с 224 по 255. Для этого главный модуль должен содержать 32-словную адресную таблицу функций, доступных оверлейным модулям. Адрес этой таблицы необходимо сообщить системе при загрузке главного модуля.

При работе с вложенными (см. выше "Рекурсивная цепочка") структурами рекомендуется использовать следующий механизм операционной системы. Все вложенные программы должны иметь доступ к одному общему байту, содержащему 0, если рекурсивная цепочка отсутствует, или номер канала, описывающего самый верхний уровень вложенности. Канал должен иметь следующую структуру:

0-й байт - номер устройства

1-й и 2-й байты - номер каталога
3-й байт - номер файла
4-й байт - номер канала, описывающего предыдущий уровень рекурсии
5-й и остальные байты - используются прикладной программой в интересах пользователя.

При входе в следующий уровень рекурсии менеджер каналов выделяет новый канал, номер которого помещается в общий байт, прежнее содержимое которого сохраняется в 4-м байте выделенного канала. При выходе из очередного уровня рекурсии содержимое 4-ого байта канала, описывающего этот уровень,помещается в общий байт, а канал освобождается. Для удаления всех уровней используется системный рестарт функции второго уровня (см. далее) rsdel - 77.

8. СПИСОК ФУНКЦИЙ

ФУНКЦИИ ЧЕРЕЗ 4-ЫЙ КАНАЛ

Функции управления устройствами:

0(00):create	8(08):kwait	16(10):g_cnfg	24(18):crchn
1(01):clear	9(09):ktest	17(11):g_kedv	25(19):newchn
2(02):flush	10(0A):type	18(12):g_tydv	26(1A):swkey
3(03):modwr	11(0B):tycpl	19(13):g_bldv	27(1B):swtyp
4(04):unird	12(0C):typos	20(14):g_drv	28(1C):swblk
5(05):mltrd	13(0D):dread	21(15):	29(1D):ldnew
6(06):quard	14(0E):write	22(16):stchn	30(1E):l_im2
7(07):key	15(0F):binit	23(17):delchn	31(1F):erdrv

Файловая служба:

32(20):open	40(28):putf	48(30):fcut	56(38):distr
33(21):open1	41(29):rpart	49(31):eadd	57(39):retran
34(22):	42(2A):wpart	50(32):ecut	58(3A):
35(23):crfil	43(2B):rifle	51(33):g_cat	59(3B):crf__
36(24):erfil	44(2C):wifle	52(34):find	60(3C):erf__
37(25):fopen	45(2D):qrvbl	53(35):bkfcb	61(3D):rvblk
38(26):opnum	46(2E):modo	54(36):p_stat	62(3E):wvblk
39(27):gname	47(2F):fadd	55(37):g_stat	63(3F):free

Интерпретатор командной строки:

64(40):oparm	72(48):run	80(50):fncor	
65(41):fmrst	73(49):p_com	81(51):fndev	
66(42):comstr	74(4A):dvtrn	82(52):opres	
67(43):opcat	75(4B):newcom	83(53):exeres	
68(44):exebat	76(4C):newcat	84(54):swrun	
69(45):g_com	77(4D):rcdel		
70(46):trans	78(4E):cp_ir		
71(47):g_way	79(4F):convr		

Консольная служба:

96(60):	104(68):prstr	112(70):scrol	120(78):
97(61):wt	105(69):	113(71):tylin	121(79):
98(62):box	106(6A):panel	114(72):g_scr	122(7A):
99(63):awt	107(6B):wtpos	115(73):cls	123(7B):
100(64):awtc	108(6C):str	116(74):	124(7C):d_a
101(65):lwt	109(6D):lnstr	117(75):	125(7D):a_d
102(66):adrwt	110(6E):smbgt	118(76):y___	126(7E):analys
103(67):lenwt	111(6F):	119(77):n___	127(7F):edstr

Оболочка:

128(80):shel0	136(88):ed_dig	145(91):menu	
129(81):shel1	137(89):mmenu	146(92):funct	
130(82):shel2	138(8A):g_curs	147(93):shexe	
131(83):pnview	139(8B):mwait		
132(84):shout	140(8C):g_dev		
133(85):p^csr	142(8e):shsubr		
134(86):g_csr	143(8F):cpfil		
135(87):g_mpan	144(90):shpanl		

8.1.1 #00. ФУНКЦИИ УПРАВЛЕНИЯ УСТРОЙСТВАМИ

= DOS.SYS =

0(00):create - Инициализация эл.диска с заданием размера (min=6 блоков). A - размер в блоках (260 байт/блок). (Структуру описателя устройства см. в ПРИЛОЖЕНИИ 2.)

1(01):clear - Очистка электронного диска от блоков текущего устройства.

2(02):flush - Принудительное сохранение модифицированных блоков текущего уст-ва. Выход: NC - О.К., C - в A код ошибки.

3(03):modwr - Запись измененного блока на эл.диск DE - номер блока, HL - адрес блока. (Выход см. предыдущий)

4(04):unird - Чтение блока с эл.диска, если его нет, то с устройства. DE - номер блока, HL - адрес блока. Выход -\\-

5(05):mltrd - Чтение нескольких блоков подряд с электронного диска. B - количество блоков, DE - номер блока начала, HL - адрес загрузки. Выход: с обработкой ошибок (см. предыдущий)

6(06):quard - Возвращает адрес блока текущего устройства, предварительно загружая его в электронный диск. DE - номер блока. Выход: HL' - адрес блока. (с обр.ошибок.)

7(07):ttyin - Ввод символа. Выход: A - код нажатой клавиши, флаг C - в расширенном режиме

8(08):kwait - Ожидание состояния клавиатуры с ненажатыми клавишами.

9(09):ktest - Проверка на нажатие какой-либо клавиши. Выход: Z: не нажата, NZ: в A - код клавиши.

10(0A):ttyout - Вывод на экран символа в регистре A и перевод печати на следующую позицию.

11(0B):prcpl - Задание режима печати. A=0: прямой режим, иначе инверсный.

12(0C):prapd - Установка позиции для TTYOUT H = координата Y(сверху), L = координата X(слева).

13(0D):dread - Чтение с устройства минуя электронный диск. B - число бл., DE - номер бл., HL - адрес в памяти.

14(0E):write - Запись на устройство минуя электронный диск. B, DE, HL - аналогично dread.

15(0F):binit - Переход на процедуру инициализации драйвера блочного устройства. Необходима при первом обращении к новому блочному устройству (например к каждой новой дискете). Выход с обр. ош.

16(10):g_cnfg - Возврат вектора конфигурации ядра. HL'- адрес вектора конфигурации ядра, A - номер канала системного устройства (см. ПРИЛОЖЕНИЕ 3.)

17(11):g_kedv - Возврат вектора символьного устройства ввода. HL' - адрес вектора символьного устройства ввода, (HL')=K^CSR - Keyboard Control Status Register (см. ПРИЛОЖЕНИЕ 2.) A - номер канала текущего символьного устройства ввода.

18(12):g_tydv - Возврат вектора символьного устройства вывода. HL' - адрес вектора символьного устройства вывода, (HL')= длина курсора для данного устройства (см. ПРИЛОЖЕНИЕ 5.) A - номер канала текущего символьного устройства вывода

19(13):g_bldv - Возврат вектора блочного устройства HL' - адрес вектора блочного устройства. HL'=SCCSR, A - номер канала блочного устройства (см. ПРИЛОЖЕНИЕ 6.) В тройке "g_..dv"-рестартов в HL' возвращается адрес вектора устройства, DE'=HL'-10, A=(DE').

20(14):g_drv - Возврат адреса тела драйвера в HL' по номеру устройства в A (от 0 до #17). Если О.К., то в A - номер драйвера. (Равносилен :stchn; LD A,(HL)'; stchn; LD HL,(HL'+4).)

21(15):empty - Не используется

22(16):stchn - Проверка состояния канала. A - номер канала. Выход: флаг C: A=код ошибки (нет канала и пр.) NC: A=номер канала, HL'- адрес тела канала, BC'=длина канала, DE'=($PTCHN) (см. ПРИЛОЖЕНИЕ 3).

23(17):delchn - Удаление канала. A = номер канала.

24(18):crchn - Создание канала. A = номер канала, DE = длина.

25(19):newchn - Создание канала пользователя. Вход: DE = длина. Выход: A - номер канала, HL' - адрес тела канала

26(1A):swkey - Переключение символьного устройства ввода. B = номер канала описателя символьного устройства ввода.

27(1B):swtyp - Переключение символьного устройства вывода. B = номер канала описателя символьного устройства вывода.

28(1C):swblk - Переключение блочных устройств. B = номер канала блочного устройства. Выход: с обр. ошибок. В тройке "sw"-рестартов во входном регистре B используются только 3 младших разряда.

29(1D):ldnew - Подключение нового устройства любого типа. A = номер канала устройства, HL - адрес на 9 байт меньше возвращаемого функциями g_kedv(17), g_tydv(18) и g_bldv(19). Каналы устройства и драйвера д.б. уже созданы и заполнены. (см.ПРИЛОЖЕНИЯ 2,5,6,7,8.)

30(1E):l_im2 - Добавление в цепочку (A<>0)/удаление из цепочки (A=0) процедуры, вызываемой по прерываниям от таймера (IM 2). HL= адрес процедуры обработки прерываний. Перед процедурой резервируется 2 байта (#0000) для адреса следующей процедуры.

31(1F):erdrv - Вызывается драйверами в случае ошибки. Передает управление по адресу в векторе g_cnfg (см.ПРИЛОЖЕНИЕ 3.) B=0: Disk Error, D=track, E=sector; B=1: Read Only; B=2: No Disk; B=3: Break.

8.1.2 ОПИСАНИЕ УРОВНЯ DOS.SYS

КАНАЛЫ

Канал - именованная (нумерованная от 0 до 255) непрерывная область памяти, предназначенная для временного хранения или передачи данных от одной программы в другую. Каналы находятся в специально отведенном месте памяти, так называемой "области каналов". Область каналов имеет фиксированную длину (обычно 2000 байт) и располагается между кэшем блочных устройств и резидентами. Адреса области каналов: g_cnfg(#10)-HL'+32..37. При добавлении/удалении резидентных программ из памяти область каналов смещается. Доступ к каналам из программ возможен только с помощью специальных обращений к ядру. Функции работы с каналами: #16..#19.

Структура канала

смещение	имя	длина	комментарий
0		1	логическое имя (номер)
1	CHLEN	2	длина канала
3		CHLEN	тело канала

В области каналов каналы лежат друг за другом по номерам головой к хвосту. Т.о. вся область каналов описывается всего тремя адресами: начала, конца и указателя (см. ПРИЛОЖЕНИЕ 3).

Каналы с определенными номерами имеют следующее жестко фиксированное предназначение:

00..07 - описывают каналы F0..FE блочных устройств
08..0F - E0..EF символьных устройств вывода
10..17 - D0.. символьных устройств ввода
18..D7 - каналы пользователя /newchn(#19)/
D8..E7 - описывают установленные резидентные задачи /fndev(#51)/
E8..EF - описывают драйверы символьных устройств ввода
F0..F7 - символьных устройств вывода
F8..FE - блочных устройств
FF - электронный диск (часть памяти компьютера, как правило дополнительная, используемая как быстрое блочное устройство. Обычно с упаковкой).

Каналы 00..17 называют "каналами, описывающими устройства" или "каналами устройств" или "устройствами". Они ссылаются на каналы, описывающие драйверы (E8..FE), называемые в свою очередь "каналами драйверов". Последние содержат гораздо больше информации чем устройства. Так, на пример, их можно находить по именам /функция fndev(#51)/. Структура (тел) каналов 00..17 приведена в ПРИЛОЖЕНИИ 7, а каналов E8..FE - в ПРИЛОЖЕНИИ 8.

Схема перемещения информации о тек. состоянии при работе с драйверами и устройствами:

Операция	откуда	куда
SET.com	драйвер	устройство
ldnew(#1D)	устройство	вектор
	драйвер	
sw...(1A..1C)	вектор	устройство(стар.)
	устройство(нов.)	вектор

ДРАЙВЕРЫ И РЕЗИДЕНТНЫЕ ПРОГРАММЫ

Резидентная задача - программа, обращение к которой явно состоит из двух (трех) этапов:

1. Установка (загрузка) в память с диска, настройка на адрес и инициализация.
2. Основная работа
3. Удаление задачи из памяти

1-ый и 3-ий этапы осуществляются процедурой SET.com. Она же создает 18-байтовый канал, описывающий резидента (см. ПРИЛОЖЕНИЕ 8).

Структура резидентной задачи:

0 2 Адрес процедуры инициализации. Она вызывается при ее перемещении (на вход задачи подается A=FF) или удалении (A=FE) программой SET.com. Если 0, то не вызывается. Эта программа должна подключать, если это необходимо, резидента к цепочке прерываний (IM 2) или к другой программе, перехватывать рестарты и восстанавливать их.

2 2 Адрес главного входа, т.е. процедуры запуска по имени /exbat(#44),run(#48)/ или по номеру канала /exeres(#53)/ Если 0, то программа запускается с 4-го байта.

4 R Тело задачи (структура произвольная)

R+4 2 #FFFF - разделитель

R+6 2n таблица настраиваемых адресов - 1 (см. примеры программирования). Смещения от начала вычисляются по LSA. Единица вычитается для пущей простоты настройки таких команд как: CALL, JP, LD HL, LD A,(nn), наиболее часто встречающихся в программе.

Д Р А Й В Е Р Ы

Драйвер - это резидентная программа специального назначения. Он обслуживает физическое или логическое устройство одного из трех типов:

канал драйвера	канал устр-ва	тип файла
F8..FF	00..07	blk
F0..F7	08..0F	typ,lpr
E8..EF	10..1F	key

Установленный в систему драйвер имеет только 8-буквенное имя (тип в канале не хранится). typ от lpr можно отличить по 0-му биту в 13-ом байте драйвера (5-ый байт в g_typ(#12)).

В начале каждого драйвера располагается вектор стандартной структуры:

0 2 INST - Программа, вызываемая как и для всех резидентов программой SET.com, а также при каждом переключении устройства, обслуживаемого драйвером. (При этом A=номер устройства).

2 2 ENTRY-1

4 2 ENTRY-2

6 2 ENTRY-3

8 8 STATUS-VECTOR

КЭШ БЛОЧНЫХ УСТРОЙСТВ

Кэш представляет собой быстродоступную копию некоторых наиболее часто используемых блоков блочных устройств. Он размещается в памяти ниже области каналов и имеет следующую структуру:

0	EDSIZ*4	каталог кэша, состоит из 4-байтовых описателей блоков:
0	4	описатель блока
4	4	описатель блока
8	4	описатель блока
. . . . .		
EDSIZ*4	256	блок
	256	блок
	256	блок
. . . . .		

EDSIZ - Размер кэша (количество блоков) задается в 0-м уровне ядра.
Описатель блока имеет следующую структуру:

Смещ.	Кол.байт	
0	1	регистр состояния блока, совмещенный с номером устройства. Биты: 7 - блок модифицирован (1) или нет (0); 6 - блок защищен (1) или нет (0); 5 - блок существует (1) или нет (0); 4 - резервный; 3..0 - номер устройства, которому принадлежит блок.
1	1	счетчик обращений к блоку;
2	2	(мл.,ст.) - номер блока на устройстве.

Минимальный размер Кэша равен 6 блокам. Кэш создается системой автоматически, т.к. он необходим для работы файловой службы.

8.2.1 #20. Ф А Й Л О В А Я С Л У Ж Б А

= DUD.SYS =

32(20):open - Открытие уст-ва и главного каталога. Выход: с обр.ош.

33(21):open1 - Открыть каталог по номеру начального блока каталога в DE.

34(22):empty - Не используется

35(23):crfil - Создание файла по имени. HL = адрес заполненного описателя. (Имя, тип, CSR, длина - обязательно, по желанию: LSA и прочие) (см. ПРИЛОЖЕНИЯ 9,4.). Файл созд-ся непрерывный или односегментный (в зависимости от CSR). Старший байт длины игнорируется. Проверяет наличие файла с таким именем и типом.

36(24):erfil - Удаление файла по имени. HL - адрес описателя (имя, тип), отсутствие файла - не ошибка.

37(25):fopen - Поиск и открытие файла или каталога по имени и типу. HL - адрес описателя (имя, тип). (подкаталог открывается, т.е. его описатель (19 байт) считываются в сис. область описателя каталога. (Структуру описателей см. в ПРИЛОЖЕНИЯХ 9.) Выход: если ошибок нет: и если BIT5=0, то файл открыт, в A - регистр состояния файла (FSTAT), в HL' - адрес 32-байтового описателя файла (FNAME) если BIT5=1, то функция FOPEN открыла внутренний подкаталог

38(26):opnum - Открытие файла по номеру в текущем каталоге. Номер файла - в регистре A. Выход: HL'=FNAME.

39(27):gname - Возвращает описатель файла, не открывая его на входе в рег.E - номер файла. Выход: если О.К., то HL' - адрес описателя файла в электронном диске.

40(28):putf - Закрытие файла (сохраняет описатель текущего файла в каталоге). Необходима пользователю для защиты файлов и инициализации резервных байтов в описателе. Номер в $FNUMB.

41(29):rpart - Чтение части файла. A,HL - указатель, IX - адрес в памяти, DE - длина в байтах. Пользуется эл.диском. (Обращается к $MLTRD и $QRVBL)

42(2A):wpart - Запись части файла. A,HL - указатель, IX - адрес в памяти, DE - длина в байтах,выход: с обр.ош. Пользуется эл.диском (обращается к $MODWR или $WRITE, и $QRVBL,$MODO)

43(2B):rifle - Чтение файла в блоках. B - количество блоков, DE - номер блока начала, HL - куда читать.

44(2C):wifle - Запись файла в блоках. B - количество блоков, DE - номер блока начала, HL - откуда записывать. Эти 2 процедуры эл.диском не пользуются (обращаются к $DREAD и $WRITE)

45(2D):qrvbl - Возврат адреса виртуального блока на электронном диске. Вход: DE - номер блока в файле. Выход: HL' - адрес блока в эл.диске, DE' - адрес заголовка блока в каталоге эл.диска, A = 0, если блока не было на эл.диске (чтение с устройства); возвращается 1, если блок был.

46(2E):modo - Модифицирование (квазизапись) блока, считанного в эл.диск процедурами qrvbl или quard. Вход: DE= адрес заголовка блока в каталоге эл.диска (выдается процедурами quard и qrvbl в DE'). Рекомендуется модифицировать только что считанный блок, т.к. любое обращение к вирт.диску может повлиять на расположение блоков в нем.

47(2F):fadd - Добавление блоков к файлу. A - число блоков, DE - указатель в блоках. Выход: с обр.ош.

48(30):fcut - Удаление блоков из файла. A - число блоков, DE - указатель в блоках. Выход: с обр.ош.

49(31):eadd - Добавление к концу файла DE байт.

50(32):ecut - Удаление от конца файла DE байт.

51(33):g_cat - Возврат номера 0-го блока текущего каталога в HL' и номера устройства в A, E' - количество всех файлов в каталоге (существующие + удаленные), D' - вложенность подкаталога, в BC'- N начального блока старшего каталога.

52(34):find - Поиск и открытие файла или каталога по имени и типу как файла. HL - адрес описателя (имя, тип).HL'=FNAME.

53(35):bkfcb - Возврат адреса описателя файла в HL'(=FNAME) (см. ПРИЛОЖЕНИЕ 9)

54(36):p_stat - Сохраняет в канале номер A(A<>0) текущее состояние системы. A=0: среда сохраняется по адресу IX. Выход с обр. ош. Если О.К., то A(<>0) сохраняется, HL' - адрес канала.

55(37):g_stat - Восстановление среды из канала с номером в регистре A(A=0,A<>0: аналогично p_stat). Выход: с обр.ош. Состояние системы может иметь любую длину. Первые 4 байта инициализируются функцией p_stat: 0(1) номер канала устройства, 1(2) каталог, 3(1) номер файла в данном каталоге.

56(38):distr - Захват блоков на устройстве .A = размер требуемой непрерывной области на устройстве. Выход: BC' = номер 1-го блока области (с обр. ош.)

57(39):retrn - Перенастройка резидентной программы или драйвера. BA = длина,HL = старый адрес ,DE = новый адрес.

58(3A):empty - Не используется

59(3B):crf__ - Создание файла по имени в HL без проверки на существование (с обр. ош)

60(3C):erf__ - Удаление открытого файла (с обр. ош.)
61(3D):rvblk - Чтение блока номер DE (виртуального, т.е. от начала тек. открытого файла) в буфер по адресу HL.
62(3E):wvblk - Запись виртуального блока номер DE из буфера по адресу HL.
63(3F):free - Открепить на текущем блочном устройстве A блоков начиная с DE-го.

8.2.2 ОПИСАНИЕ УРОВНЯ DUD.SYS

СТРУКТУРА БЛОЧНОГО УСТРОЙСТВА

Каталоговое (блочное) устройство в системе iS-DOS может располагаться как на дисках, так и в дополнительной памяти компьютера (т.н. "быстрый диск"). Файловая служба (уровень DUD.SYS) не делает различий между ними (тонкостями занимаются уровень DOS.SYS и драйверы). Т.о. с точки зрения DUD.SYS блочное устройство представляет собой набор блоков (размером 256 байт) с номерами от 0 до некоего максимального. В 0-ом блоке находится заголовок тома (см. ПРИЛОЖЕНИЕ 1). В нем указаны размер устройства, полезная информация для драйвера и ссылка на главный каталог устройства (для 80-дорожечного двухстороннего диска это обычно 3-ий блок). Все каталоги в iS-DOS, включая и главный, представляют собой почти обычные файлы. Файл в iS-DOS - это набор блоков располагающихся на устройстве подряд (непрерывные файлы) или состоящие из нескольких (от 1 до 85) непрерывных участков ("сегментов"). Сегментированные файлы имеют в своем 32-байтовом описателе, лежащем в каталоге, ссылку на специальный блок, описывающий сегментную структуру файла. Нулевой байт этого блока содержит количество сегментов файла. Остальные 255 байт рассматриваются как 85 3-байтовых записей, каждая из которых описывает 1 сегмент файла. Первые 2 байта записи - номер 0-го блока (адрес на устройстве) сегмента, а последний байт записи - размер сегмента в блоках. Файл 0-ой длины имеет заполненный нулями блок описателя сегментов. К сегментированным файлам можно добавить блок или несколько в начало или середину, (а в конец даже с точностью до байта), а также удалить блоки.

Непрерывные файлы и каталоги можно только укорачивать (процедурой ecut(#32)), зато доступ к ним - быстрее. Если известны правила использования каталога, то его лучше делать непрерывным и нужного размера (см. функции).

32-байтовые описатели файлов располагаются в файле каталога подряд по 8 записей в блоке. Каталог - тоже файл, но у него 2 описателя (кроме главного каталога устройства, у которого только 1: внутренний): внешний, в старшем каталоге, и внутренний - 0-ой файл самого каталога. Вся информация о размерах, количестве файлов, уровне вложенности хранится во внутреннем описателе. Именно этот описатель загружается в память при открытии каталога процедурами open1(#21) и fopen(#25). Каталог, как и любой файл, может быть непрерывным или сегментированным. В первом случае при заполнении каталога файлами до предела (8 файлов на блок) Вы получите удовольствие наблюдать сообщение об ошибке с кодом 105. Программа rename.com поможет Вам устранить неудобство переделкой каталога в сегментированный. Сегментированный каталог при его наполнении файлами сам увеличится до нужного размера. Пределом будет лишь ограничение на 128 файлов в каталоге, вызванное несколько небесконечными размерами памяти ZX-Spectrum, что однако, на наш взгляд, полностью компенсируется возможностью создавать подкаталоги до 6-го уровня вложенности.

Переполнение блока описателя сегментов файла (ошибка 94) - еще одна неприятная ситуация, в которой можно оказаться при бесконтрольном добавлении и удалении блоков в файлах большой (более 85 блоков) длины. При добавлении в конец файла с излишней сегментированностью файла легко справиться, добавляя к файлу по несколько блоков сразу, и удаляя все лишнее по окончании рабочего процесса. Так делает, на пример, программа prt.com при печати в файл, добавляя к файлу по 5 блоков. Если же программа добавляет блоки в середину файла, как к примеру edit.com, то выход можно найти лишь в периодическом копировании файлов или разбиении большого файла на несколько поменьше.

Еще одно маленькое неудобство для любителей всего большого: процедуры crfil(#23) и crf__(#3B) обнуляют старший байт длины, после чего получают число блоков файла округляя младшие 2 байта. В результате сходу Вы сможете получить файл длиной от 0 до 65280 байт (255 блоков). Далее Вы можете воспользоваться процедурами добавления fadd(#2F) или eadd(#31). В результате нехитрых комбинаций Вы получите файл до 21675 блоков длиной (5548800 байт), что в 6 с лишним раз превышает вместимость дискеты. Если этот путь Вас по каким-либо причинам не устраивает, то Вы можете напрямую обратиться к бит-карте диска (файл device.sys, блоки 1,2). 1 бит на блок, 0-му блоку диска соответствует 7-ой бит 0-го байта 1-го блока device.sys, 1-ому блоку - 6-ой бит, 8-ому блоку - 7-ой бит 1-го байта и т.д.

ДИСКИ И ФАЙЛЫ

В главном каталоге первый файл, как правило, device.sys. Чтобы он не выбирался по маске, отметке и т.п. в его регистре состояния подняты все биты. Т.е. он невидим, защищен и даже вроде как бы каталог. В нем находятся заголовок тома и бит-карта блоков тома. Т.о пользователь получает к ним доступ через файл. Открыть файл можно процедурой find(#34), а процедурой fopen(#25) и обращающимися к ней процедурами уровня COM в данном случае пользоваться не рекомендуется. Если диск загрузочный, то в главном каталоге обязан присутствовать еще один файл, защищенный подобным образом: boot.sys. Он также создается при инициализации тома программами format.com и create.com. В нем программа BOOT.com размещает загрузчик системы.

На загрузочном диске также должен находится файл ????_dos.sys, представляющий собой выгруженную на диск память компьютера с системой iS-DOS. Сохранение на диске производит программа svsys.com. Пример: svsys smal. Программа SVSYS при подобном обращении создаст файл smal_dos.sys в текущем каталоге. Чтобы система загружалась именно из этого файла, необходимо присоединить этот файл к тому программой CONNECT.com: connect smal.

При загрузке происходит автозапуск командного файла S:autoexec.bat. На системном диске необходимо наличие файла S:extent.txt, определяющего действия системы в зависимости от расширения файла при нажатии клавиши Enter на этом файле. Также обязательно наличие каталога S:SHELL и файлов extkey.txt, extview.txt и extprin.txt, определяющих действия системы (строго говоря, уровня SHELL) при отработке всех функциональных клавиш. Все файлы ищутся на диске Q - "quick", т.н. "быстром" диске. Им разумно назначить электронный диск, если у Вас имеется дополнительная память и соответствующий драйвер. Для работы редактора необходим каталог Q:EDIT со всеми нужными редактору файлами.

8.3 #40. ИНТЕРПРЕТАТОР КОМАНДНОЙ СТРОКИ

= COM.SYS =

64(40):oparm - Открывает файл-параметр. Путь к файлу (полный или от текущего каталога) лежит в буфере командной строки вместе с именем файла или его маской. (см. ПРИМЕЧАНИЯ 1,2) Выход: с обр. ошибок. Флаг NZ: нет заданного пути или ошибка в символьной строке.

65(41):fmrst - Восстанавливает первоначальное состояние среды. (устройство, каталог и файл, запоминаемые процедурами работающими с командной строкой). Выход: с обр. ошибок, иначе сохраняет AF.

66(42):comstr - Разбирается с указанным путем и открывает данное устройство и каталог (файл не открывает!) или снимает ключ. Вход: HL = адрес командной строки. Выход: флаг C: ошибка ввода/вывода (восстанавливается прежняя среда);
NC,Z: A=0: HL'= 11-байтовый описатель/шаблон файла (если имя/маска файла не указано после пути, то шаблон из #FF)
A>0: HL'= адрес символа "/" (обнаружен ключ), A=первый символ после ключа.
NC,NZ: или пути нет, или ключа нет (пустая строка), или синтаксическая ошибка в командной строке.

67(43):opcat - Аналогична процедуре comstr(#42), но работает с внутренним буфером командной строки.

68(44):exebat - Запуск файла по пути и имени файла, находящегося во внутреннем буфере командной строки. Вход: A=0: стандартная обработка типа файла, т.е. обращение к файлу S:extent.txt, A>0: DE = адрес буфера с путем и именем текстового файла. Выход: CARRY SET - серьезная ошибка, иначе: Z - О.К., NZ - какая-нибудь досадная мелочь типа отсутствия каталога или файла. (см. ПРИМЕЧАНИЯ 1,2)

69(45):g_com - Возвращает в HL' адрес внутреннего буфера командной строки. В отрицательную сторону расположен вектор: -10(1) CSR ошибок строковых рестартов: 0:флаг C, FF:-NZ
-9(4) среда последнего exebat(#44)'a (run'a) (устройство(1),каталог(2),файл(1))
-5(2) указатель в командной строке
-3(2) дата
-1(1) номер bat-файла в рекурсивной цепочке.
DE'= 0(4) среда для fmrst(#41), -3(1) номер канала содержащего пути для дополнительного поиска файлов для exebat(#44) и run (#48). (устанавливается утилитой path.com). Если 0, то "тропа" не установлена.

70(46):trans - Преобразует имя файла и тип отделенный точкой (либо маску) в 11-байтовый описатель имени и расширения (или шаблон в случае маски на входе). HL/DE = адреса входного/выходного буферов. Входной буфер заканчивается кодом #0D или пробелом. Выход: флаг C - признак маски-шаблона. Символы "*" и "?" преобразуются в код 255, пробел является разделителем. Используется для поиска по маске. Например: a*.h?p преобразуется в a^^^^^^^h^p, где ^ означает код 255.

71(47):g_way - Формирует в заданном канале или буфере в символьном виде путь текущего подкаталога. (Неиспользованное пространство канала/буфера заполняется пробелами). A>0: A = номер канала A=0: в буфере по адресу в HL, DE=размер буфера. Выход: если все О.К., то восстанавливает текущий каталог (но не файл!). Регистр A не сохраняет.

72(48):run - = p_com(#49) + exebat(#44). (Выход см. exebat). Передаются регистры IX,B.

73(49):p_com - Заполнить внутренний буфер командной строки. HL = адрес строки, оканчивающейся кодом 13. При длине строки более 128 символов возвращает ошибку (флаг C).

74(4A):dvtrn - Преобразует логическое имя устройства в физическое. Вход и выход в рег.A. При отсутствии данного логического устройства возвращает флаг NZ.

75(4B):newcom - Аналогична comstr(#42), но для нового диска.

76(4C):newcat - Аналогична opcat(#40), но для нового диска.
Обе "new" процедуры используют процедуру binit(#0F). В случае ошибки среду не восстанавливают.

77(4D):rcdel - Удаление рекурсивно созданных каналов. В рег.A - номер канала завершающего цепочку. Рекурсивная цепочка состоит из каналов, 4-ый байт которых содержит номер предыдущего канала. Если канал в цепочке первый, то данный байт равен 0.

78(4E):cp_ir - Ищет в таблице по адресу DE слово, совпадающее с началом последовательности символов по адресу HL, оканчивающихся на символ меньше "A"(#41 hex). Если нет, то NZ , иначе Z и в регистре A - номер слова в таблице. Нумерация от 0. Первые 2 байта таблицы - длина таблицы без самой длины. Далее слова длиной от 1 до 7 символов, оканчивающихся "13"(#D). Выход: HL'- продолжение текста после искомой модели, DE'- продолжение таблицы после найденной модели.

79(4F):convr - Преобразует 11-байтовый описатель имени и расширения файла (в HL), в 13-байтовое имя файла с расширением через точку и с "13" в конце (в DE). На выходе DE' - адрес #0D.

80(50):fncor - Проверяет на правильность имя файла или каталога. A: Если 5-ый бит установлен,то проверяется каталог (при этом все маленькие буквы меняются на большие), если не установлен - то файл. Выход: флаг C - если есть запрещенные символы, HL' = адрес 11-байтового описателя имени. В имени файла разрешены: латинские и русские буквы альтернативной кодировки ГОСТа (строчные и прописные), цифры а также 8 символов: #$&+-=_` (коды: 23,24,26,2B,2D,3D,5F,60).

81(51):fndev - По имени в HL (8 букв) ищет установленный драйвер или резидентную задачу. A= номер канала, с которого начинать поиск (#D8 для .res), B - число каналов перебора(#10). Выход: C^ A=37 - не нашла, NC - O.K.: A=номер канала, HL'= тело канала: 0(4) status: dev,cat,file (.res), 4(2) адрес в памяти, 6(2) длина, 10(8) имя. (см. КАНАЛЫ и ПРИЛОЖЕНИЕ 8).

82(52):opres - Аналог opcat для .res, .key, .typ, .lpr, .blk (пробелы вместо типа = .res)(C,NZ -//-), но ключей не снимает (пользуйтесь opcat). По имени и типу (через точку) в буфере командной строки ищет установленную резидентную задачу или драйвер. "@" означает "bat.res". Выход с флагом Z аналогичен fndev(#51).

83(53):exeres - Запуск резидентной задачи по номеру канала в A. (Для запуска по имени используйте run или exebat). Передает резиденту все регистры, кроме AF, чем пользуются ch.res, ren.res и пр.

84(54):swrun - Запуск .com-файла с сохранением тек. среды (IX-начало, DE-длина сохраняемой области, HL-строка для run'a) в файле swap0.swp. По выходе из вызванного т.о. .com-файла восстанавливается сохраненная среда.

ПРИМЕЧАНИЯ к уровню com.

1. Путь может быть задан как в символьном виде, так и в упакованном формате, т.е.: 0(1): код 8, 1(1): номер устройства (диск), 2(2): каталог (см. g_stat(#37)), 4(11): файл (имя и тип либо шаблон (см. trans(#46))).

2. Указатель в командной строке - адрес для обработки путей или ключей внутри буфера командной строки. p_com(#49) устанавливает указатель на начало буфера командной строки. Остальные рестарты ,использующие командную строку, берут свои данные и устанавливают указатель на первый пробел или код 13 после данных. Oparm(#40), opcat(#43), exebat(#44) обрабатывают путь с текущего положения указателя.

3. Уровень поддерживает 3 внутренних команды установки дисков S,T,Q: L_S,L_T,L_Q, команду установки текущего каталога CAT, и команду управления флагами ошибок ERR. Примеры: "L_T /B", "CAT S:SHELL\HELP", "ERR /0": по ошибке типа "нет пути/файла" рестарты opcat(#43), comstr(#42), exebat(#44), run(#48) выйдут с флагом NZ (старый стандарт), "ERR /1": в подобной ситуации выдать флаг C и A=81 (новый стандарт).

8.4 #60. ОКОННАЯ ТЕХНОЛОГИЯ

= WIND.SYS =

97(61):wt - Вывод окна. Подфункция в рег. A: A=0 окно с одинарной рамкой, A=1 без рамки, A=2 с двойной рамкой, A=3..127 без рамки, A=128..255 только подкраска. IX= вектор окна (см. ПРИЛОЖЕНИЕ 10.)

98(62):box - Рисование/стирание рамки окна. Вход: IX = адрес вектора, DE - отступ по Y и X от края окна; Регистр A: бит 7 - рисовать(0)/стирать(1), биты 0..3 - стороны: левая, правая, верх, низ.

99(63):awt - Подкрашивает строку внутри окна, с рамкой. Вход: IX = адрес вектора окна, A = номер строки, которую необходимо подсветить (нумерация от 1), B = цвет.

100(64):awtc - Подкрашивает строку внутри окна, не затрагивая рамку, IX=вектор окна, A=номер строки, B=цвет.

101(65):lwt - Распечатка заданного количества строк в окне:
A - номер строки (начала), IX - вектор окна, HL - адрес строк текста для печати, B - количество строк, которое необходимо распечатать;

102(66):adrwt - Печать текста в окне. IX - вектор окна, после которого лежит текст окна (символ "13" служит в качестве <ВК>, "3" - конец текста);

103(67):lenwt - Аналог adrwt, но в HL - адрес текста окна;

104(68):prstr - Распечатка строки в окне с несколькими подфункциями. Код подфункции в регистре E:

E=0 - prstr - Распечатка строки в окне по логическому имени с центровкой. IX - адрес вектора окна, B -логическое имя строки, HL - адрес массива-анализатора, младшие 6 битов регистра A - - смещение по координате Y в окне, старшие 2 бита - режим центровки. 00 - без центровки, 10 - выравнивание по правому краю окна, 11 - выравнивание по центру;

E=1 - xystr - Распечатка строки в окне заданной длины с центровкой.IX - адрес вектора окна, B - длина строки, HL - адрес строки, A - аналогично prstr

E=2 - ..... - Находит строку с номером в регистре B. На входе: HL-адрес текста, B - номер строки. Выход: HL -возвращает адрес, B - длину

E=3 - tystr - Распечатка строки в окне по номеру в тексте с центровкой. IX - адрес вектора окна, B - номер строки в тексте, HL - адрес текста строк, A - аналогично prstr

105(69):empty - не используется

106(6A):panel - Работа с панелью. Код подфункции в регистре A (см. ПРИЛОЖЕНИЕ 15.)

107(6B):wtpos - Инициализирует системную переменную xypos, вычисляя ее значение по вектору IX и смещению в окне в регистрах: H - по Y (IX+1),L - по X (IX+6).

108(6C):str - Печать строки, кончающейся на "13",в текущей позиции. HL - адрес строки.

109(6D):lnstr - Печать строки. HL - адрес, в B - количество символов. Обращается в цикле к ttyout(#A). Координаты устанавливают следующие функции: typos(#C), wtpos(#6B), ttyout и т.д.;

110(6E):smbgt - Строковый мобильный редактор. Сам печатает содержимое буфера. Вход: HL - адрес буфера, A - ширина окна (X), B (биты 0..4) - высота окна (Y), Старшие биты регистра B:
бит 7: курсор позиционируется на первый пробел с конца(0)/начала(1),
бит 6: на конец(0)/начало(1) буфера,
бит 5: режим "overtype"/"ME"(очистка буфера по нажатии).
D - M^CSR, E - K^CSR. (см.ПРИЛОЖЕНИЕ 2.) Выход: с обр. ош. Флаг C: ошибка ввода/вывода (нет драйвера например). Иначе (NC): флаг Z: Выход по клавише <Enter>, A = длина строки без <ВК>, флаг NZ: Выход по одной из 5 спец. клавиш: CS/9, CS/SS, SS/A, SS/Space, SS/Enter (коды от #0E до #12), A= код клавиши.

111(6F):empty - не используется

112(70):scrol - рег. A=0: настройка процедуры скроллинга на вектор окна. A=1: скроллинг на 1 строку вверх. A=2: вниз в IX адрес расширенного вектора окна во всех трех случаях (-1:1/2/4/8 - дискретность скроллинга, -2:задержка), HL=крайние байты заполнения. Рекомендуем: HL=0 для пробелов, HL=#5A0 для окна с двойной рамкой, HL=#180 для одинарной. Сдвигается область внутри рамки.

113(71):tylin - Распечатка строки по вектору для edstr от XS (позиции курсора) до конца видимой области. IX - адрес вектора ("ошибок быть не может")(см.ПРИЛОЖЕНИЯ 10..13)

114(72):g_scr - Возвращение в регистре HL' адреса вектора экрана и beep'a (см. ПРИЛОЖЕНИЕ 14.)

115(73):cls - A=0: Очистка экрана с помощью переменных PAPER и BORD (см. ПРИЛОЖЕНИЕ 14.), иначе: подкраска экрана цветом A.

116(74):empty - не используется

117(75):empty - не используется

118(76):y___ - Включение мигающего курсора.

119(77):n___ - Выключение мигающего курсора.

120(78):empty - не используется

121(79):empty - не используется

122(7A):empty - не используется

123(7B):empty - не используется

124(7C):d_a - Преобразование целого числа в ASCII-строку. Вход: HL - адрес выходного ASCII-буфера, A = длина буфера, Carry reset: DE - 2-байтовое число, Carry set: DE - адрес 4-байтового числа. B - основание системы счисления Выход с обр. ош. (переполнение буфера)

125(7D):a_d - преобразование строки ASCII-символов в 4-байтовое число. Ведущие пробелы игнорируются. HL - адрес ASCII - буфера, A - длина буфера (если A=0, то до кода 13), B - основание системы счисления по умолчанию (не более 16, B=0 понимается как 10). Система счисления может также определяться по первому символу буфера: #,h,H - 16; .,d - 10; o,O - 8; %,b,B - 2. Выход: 32-разрядное число в DE'HL'. BC' - продолжение ASCII-буфера. Возможные ошибки (флаг C): нецифровой символ (A=0), переполнение (A=1).

126(7E):analys - Передача управления подпрограмме по адресу из таблицы. HL - адрес таблицы, A - код команды. Таблица состоит из записей по 3 байта: 0(1) код команды, 1(2) адрес процедуры. В последней записи должен лежать байт #FF и адрес отработки ситуации "код не найден".

127(7F):edstr - Редактирование строки в тексте. Отрабатывает все печатные коды (символы) + 6 управляющих: клавиши 1,2,3,5,8,0 с Caps Shift'ом. IX - вектор, (на входе необходимо инициировать LNST, XS и COM) Выход: CARRY - системная ошибка. (см. ПРИЛОЖЕНИЯ 10..13).

8.5 #80. Ф А Й Л О В А Я О Б О Л О Ч К А

= SHELL.SYS =

128(80):shel0 - Главный выход в оболочку: создание вирт. диска. (Размер: g_cnfg(#10):HL'-6)

129(81):shel1 - Выход с очисткой экрана, инициализацией ERDEV, QUEST (см. ПРИЛОЖЕНИЕ 3), USPNL, PAPER, BORD (см. ПРИЛОЖЕНИЕ 14), переключением на основное (0-ое) устройство печати /swtyp(#1B)/ в прямом режиме /prcpl(#0B)/. В блочном драйвере устанавливается 7-ой бит в 9-ом байте (см. ПРИЛОЖЕНИЯ 3,6).

130(82):shel2 - Выход с перерисовкой и подкраской верхней строки подсказок и печатью обеих панелей. Все 3 shel-рестарта перехватывают стек и принудительно задают pnCSR=%01010, nLOG=1 (см. ПРИЛОЖЕНИЯ 16,17), причем shel1 является точкой входа в shel0, а shel2 - в shel1.

131(83):pnview - Ожидает отпуска клавиш и возвращает в рег. A код cshel2, если панели одинаковые. Иначе A=cvwpnl. (см.ПРИЛОЖЕНИЕ 16.). Выход с обр.ош. (пользуется ф-цией stchn(22))

132(84):shout - Выход в оболочку с выполнением внутренней команды оболочки (номер в A). Carry set - обработка ошибки (см.ПРИЛОЖЕНИЕ 16.)

133(85):p^csr - Устанавливает маску состояния файла в панели для пометки и проверки файла. Вход: A= маска. По умолчанию маска=1, т.е. уровень SHELL метит файлы поднимая 0-ой бит.

134(86):^_csr - По номеру файла в каталоге (в рег. E) возвращает: A=(HL')= байт состояния отметки файла, (HL'-1)= порядковый номер отметки.

135(87):g_mpan - Запрос о состоянии текущей панели Выход: A - число отмеченных файлов на текущей панели, L' и H' - минимальный и максимальный номера файлов, а именно: L'=0, H'= где-то между общим числом файлов в каталоге, включая удаленные, но без каталогов, и числом неудаленных файлов, но с каталогами. DE' - адрес вектора рабочей панели -48, BC' - адрес вектора оболочки. (см. ПРИЛОЖЕНИЯ 17,18).

136(88):ed_dig - Редактирование десятичного числа в окне (для menu напр.). A - число цифр, DE - координаты для wtpos, HL - начальное значение числа, IX - вектор окна. Выход: Флаг C: ошибка (переполнение числа, ввода/вывода и пр.); NC,Z: HL'= новое значение числа; NC,NZ: выход по одной из 5 спец. клавиш (см. smbgt(#6E).

137(89):mmenu - Перерисовка верхней строки подсказок оболочки с подкраской или без в зависимости от 0-го бита nLOG (см. ПРИЛОЖЕНИЕ 17.)

138(8A):g_curs - Возвращает параметры панельного курсора. Выход: A=E'= номер файла, D'= позиция Y в окне (=H для wtpos), B' - номер текущей панели (0 - левая, 1 - правая), HL' - адрес вектора окна панели. (ПРИЛОЖЕНИЯ 17,18.)

139(8B):mwait - Пишет "Please Wait" на месте курсора

140(8C):g_dev - Переключение устройств. Ввод символа с клавиатуры с миганием курсора в окне. Вход: HL=смещение для wtpos, Carry Set: A=номер устройства, иначе: текущее. Флаг Z: Выход через RST dvtrn(#4A), NZ: через smbgt(#6E).

142(8e):shsubr - Выполнение команд из прикладных задач. A- код команды. (ПРИЛОЖЕНИЕ 16)

143(8F):cpfil - Сравнение 11-байтового шаблона(DE), (например результата RST trans(#46)), с 11-байтовым описателем(HL) файла (имя и расширение). Если равны то флаг Z иначе NZ.

144(90):shpanl - Выполнение команд работы с панелями : Рег. B - регистр состояния pnCSR (ПРИЛОЖЕНИЕ 17). A=0 -аналог shel0, A=1 -аналог shel1, A=2 -аналог shel2, A=3 - движение курсора по панели, A=5 -pnDR - печать обеих панелей,A=6 -TYPNL- печать одной панели.

145(91):menu - Меню. Перепечатывает окно. После стандартного wt-окна необходимо положить буфер CSR строк длиной исходя из расчета по 1 байту на строку. CSR: 0 - проскакивать строку, 1 - останавливаться на ней.

Вход: IX - адрес вектора окна: байты 0..7 - стандартное окно.

В отрицательную сторону:

-12(1) Цвет курсора при отработке Enter

-11(1) Управляющий регистр меню. Биты 0/1:

0 - awt/awtc - Тип курсора

1 - старый/новый стандарт вектора.

(По новому стандарту: 8(2) - Адрес CSR-массива минус 1, 10(2) - Адрес текста.)

7 - Вызывать программу пользователя -6(2) до/после ожидания клавиши.

-10(1) Цвет курсора

-9(1) Позиция курсора (от 1)

-8(2) Адрес программы инициализации символьного массива, вызываемой при входе в меню и при каждом выходе из строковой ф-ции с NZ,A=1.

-6(2) Адрес программы пользователя, выполняющейся при нажатии любой клавиши.

-4(2) Адрес списка дополнительных функций (т.н. "горячих клавиш"). Этот список должен состоять из 3-байтовых значений: 0(1) код клавиши, 1(2) адрес дополнительной процедуры. В этом случае на входе в процедуру B=код клавиши, курсор (IX-9) не перемещается. Если 2(1)=0, то 1(1) означает номер строки и данная клавиша обрабатывается как Enter на строке с указанным номером, (IX-9) принимает соответствующее значение. Если адрес списка доп. ф-ций в векторе = 0, то дополнительно отрабатываемых клавиш нет.

-2(2) Адрес списка функций отработки клавиши Enter. Список должен состоять из 2-байтовых адресов функций, количество которых должно соответствовать количеству строк в меню. Если один из адресов функций или адрес всего списка = 0, то при нажатии ENTER ничего не произойдет.

8(2) Адрес CSR-массива минус 1

10(2) Адрес текста (По новому стандарту)

(По старому стандарту с 8-го байта: CSR-строка, затем - текст окна.) Функции, выполняемые по нажатию клавиши Enter могут управлять дальнейшими событиями флагом Z/NZ и содержимым регистра A: Z: ничего не делать, NZ: A=255 - выйти из меню, A=1 - вызвать процедуру инициализации. Если функция отработки доп. клавиш выходит с флагом Z, то выполнится та строка в меню, номер которой будет на выходе в рег.A (Если 0, то ничего не выполняется). При нажатии SS/A происходит выход из меню, причем происходит установка: рег.A = 225 и флаг NZ.

146(92):funct - Выполнение ф-ций-утилит по текстовому файлу. HL=путь файла, A=код ф-ции. Используется оболочкой для отработки нажатых клавиш (файл S:SHELL\extkey.txt). Среда сохраняется. Для обозначения служебных символов используется десятичное число, начинающееся с одиночной кавычки. Так сама кавычка будет выглядеть как '39. путь и ключи ф-ции отделяются пробелом. (см. extkey.hlp).

147(93):shexe - Выход в оболочку с запуском RST exebat(#44) с A=0. (см. ПРИЛОЖЕНИЕ 17)

9. ПРИЛОЖЕНИЯ К СИСТЕМНЫМ РЕСТАРТАМ

ПРИЛОЖЕНИЕ 1. Структура описателя устройства:

0-ой БЛОК: (0-ой сектор на 0-ой дорожке)

смещение	длина	комментарии
0	2	Резерв
2	8	Имя устройства
10	3	Признак iS-DOS: "DSK"
13	5	Резерв
18	2	Размер устройства (в блоках по 256 байт)
20	2	Номер 0-го блока главного каталога
22	1	Количество цилиндров на устройстве
23	1	Тип диска. Биты(0/1): 0 - 40/80 дорожек 1 - 1/2 стороны
24	1	Размер сектора: 1/2/4: 256/512/1024 байт
25	1	Количество секторов на дорожке
26	1	Резерв
27	1	Контрольная сумма описателя ????_dos.sys
28	2	Резерв
30	2	Дата
32	32	Описатель ????_dos.sys
64	16	Таблица номеров секторов на дорожке

1-ый БЛОК: Бит-карта устройства.
1 бит/блок: 0-свободен/1-занят

ПРИЛОЖЕНИЕ 2. Структура вектора символьного устройства ввода (СУВВ) / g_kedv(#11) DOS.SYS /

смещение	имя	длина	комментарий
-20	COUNT	4	Счетчик прерываний
-16	R_IM2	2	Адрес процедуры (АП) обработки IM 2
-14	ACOUN	2	Адрес счетчика прерываний COUNT
-12	TIMER	2	АП, вызываемой (если не 0) при (COUNT)=0
-10	CRKEY	1	Номер канала текущего СУВВ
-9	TTYIN	3	Переход на процедуру (ПП) ввода символа с клавиатуры
-6	KWAIT	3	ПП ожидания состояния клавиатуры с ненажатыми клавишами / kwait(8) /
-3	KTEST	3	ПП проверки на нажатие какой-либо клавиши / ktest(9) /
0	K_CSR	1	Байт состояния СУВВ (клавиатуры) биты (0/1): 0 - строчные/ПРОПИСНЫЕ 1 - latin/русские 2 - текст/псевдографика 3 - запрет kwait(1)(сбрасывается сам)
1	M_CSR	1	Маска разрешений смены соответствующих битов K_CSR (1: разрешен)
2	KLAST	1	Код последней нажатой клавиши
3	REPD	1	Интервал между первыми двумя одинаковыми клавишами (значение по умолчанию 29)
4	REPP	1	Интервал между последующими одинаковыми клавишами (значение по умолчанию 1)
5	IKEYB	2	Адрес процедуры опроса клавиатуры

Байты 0(8) лежат в соответствующем канале устройства (см. ПРИЛОЖЕНИЕ 7)

ПРИЛОЖЕНИЕ 3. Структура вектора описания системы
/ g_cnfg(#10) DOS.SYS /

смещение	имя	длина	комментарий
-6		1	Размер эл.диска по умолчанию (обычно 29)
-5	ABODY	2	Адрес тела эл.диска
-3	CRDEV	1	Номер канала устройства T: (текущего)
-2	QUDEV	1	Номер канала устройства Q: (быстрого)
-1	SYDEV	1	Номер канала устройства S: (системного)
0	CNFG	1	Текущее количество компонент ОС
1	Stop	2	Адрес вершины ОС (то же, что BTCHN)
3	Utop	2	Адрес вершины программы пользователя
5	ADCAT	2	Адрес начала электронного диска
7	SZCAT	1	Размер электронного диска в блоках (1 блок - 260 байт)
8	APWR	16	Адреса загрузки 8-ми компонент ОС
24	QNMOD	8	Текущее количество модифицированных блоков на 8-ми блочных устройствах
32	DGCHN	2	Адрес начала области каналов (ОК)
34	BTCHN	2	Адрес конца ОК
36	PTCHN	2	Указатель текущего заполнения ОК
38	ERDEV	2	Адрес программы обработки ошибок блочных устройств
40	QVST	2	Адрес процедуры запроса для блочных устройств (см. ПРИЛОЖЕНИЕ 6.)
55	$ANLZ	-	CALL ANLZ - процедура обработки RST 16
62	EMPTY	-	RET

ПРИЛОЖЕНИЕ 4. Структура тела дискового драйвера
(S:SYSTEM\INI\sys_driv.blk)

смещение	имя	длина	комментарии
0	INSTL	2	Адрес процедуры (АП) инсталляции (если 0, то не нужна)
2	BREAD	2	АП чтения / dread(#D)/
4	BWRIT	2	АП записи / write(#E)/
6	BINIT	2	АП инициализации / binit(#F)/
8	SCCSR	1	Регистр состояния
9	DRCSR	1	Регистр состояния (см.ПРИЛОЖЕНИЕ 6.)
10	OFIM2	2	Смещение от начала драйвера до кода IM2 (если 0, то при выходе IM 2 принудительно не устанавливается
12	TPSYS	1	тип TR-DOS'а (0/1)
13	QRDWR	1	Число попыток чтения/записи
14	TMBUF	2	Адрес буфера драйвера
20		2	Номер тек. сектора в буфере драйвера
30		1	Тип дисковода. Биты(0/1): 0 - 40/80 дорожек 1 - 1/2 стороны
31	TMWT	1	Задержка начала чтения сектора
32	HTIME	1	Время перемещения головок дисковода
33		1	Тип диска. Биты(0/1): 0 - 40/80 дорожек 1 - 1/2 стороны
34		1	Размер сектора: 1/2/4: 256/512/1024 байт
35		1	Количество секторов на дорожке
36		16	Таблица номеров секторов на дорожке

Байты 8..15(8) хранятся в канале и в векторе бл. устройства (см. ПРИЛОЖЕНИЯ 6,7)

Байты 33..51 описывают диск и соответствуют байтам 23..25 и 64..79 в 0-ом блоке описателя устройства. binit(#F) сам настраивает драйвер на iSDOS-диск. Для настройки драйвера на нестандартную дискету (TR-DOS, MS-DOS и т.п.) измените эти байты.

ПРИЛОЖЕНИЕ 5. Структура вектора символьного устройства вывода (СУВыВ) / g_tydv(#12) DOS.SYS /

смещение	имя	длина	комментарий
-10	CRTYP	1	Номер канала текущего СУВыВ
-9	PRCHR	3	Переход на процедуру (ПП) вывода символа на экран (принтер) /ttyoyt(#A)/
-6	PRCPL	3	ПП задания инверсного/нормального режима печати / prcpl(#B) /
-3	PRAD	3	ПП установки координат / pradd(#C) /
0	BYTTST	1	Длина курсора тек. драйвера экрана
1	XYpos	2	Текущие логические координаты печати
3	Xypos	2	Текущие физические координаты печати
5	type	1	Тип устройства:0-экран/1-принтер
6	ertype	2	Адрес обработки ошибки СУВыВ

Байты 0(8) лежат в соответствующем канале устройства (см. ПРИЛОЖЕНИЕ 7)

ПРИЛОЖЕНИЕ 6. Структура вектора блочного устройства (БУ) / g_bldv(#13) DOS.SYS /

смещение	имя	длина	комментарий
-10	CRBLK	1	Номер канала текущего БУ
-9	BREAD	3	Переход на процедуру (ПП) чтения / dread(#D) /
-6	BWRIT	3	ПП записи / write(#E) /
-3	BINIT	3	ПП инициализации / binit(#F) /
0	SCCSR	1	Регистр состояния. Биты 0/1 0..3 - номер устройства для вирт. диска 3 - драйвер без буфера/ с буфером 5 - запрет/разрешение чтения для RST 6 - запрет/разрешение записи для RST 7 - мимо эл.диска/через него
1	DRCSR	1	Регистр состояния. Биты 0/1: 0 - запрос по адресу в QVST при обращении к устройству (ПРИЛОЖЕНИЕ 1 - флаг авто-флаша (сбрасывается пользователем) 2 - TRDOS/нет 7 - не/отрабатывать ошибки (erdrv(#1F))
2		6	(см. ПРИЛОЖЕНИЕ 4.)

ПРИЛОЖЕНИЕ 7. Описание канала устройства: (длина 10 байт)

смещение	имя	длина	комментарий
0	TDISR	1	Номер канала драйвера, обслуживающего устройство
1	PARM	1	Параметры для драйвера /номер устройства
2	SCCSR	8	Байты 0..7 соответствующего вектора (см. ПРИЛОЖЕНИЯ 2,5,6)

ПРИЛОЖЕНИЕ 8. Описание канала драйвера/резидента (длина 18 байт)

смещение	имя	длина	комментарий
0	DEV	1	Номер устройства, с которого он был загружен
1	CAT	2	Каталог, в котором находится драйвер
3	NFILE	1	Номер файла в данном каталоге
4	LSA	2	Адрес загрузки драйвера
6	LINK	2	Длина драйвера
8	DCSR	1	Состояние драйвера
9	QDEV	1	Количество обслуживаемых устройств
10	NAME	8	Имя драйвера/резидентной задачи

ПРИЛОЖЕНИЕ 9. Резидентный вектор файловой службы
/ bkfcb(#35) DUD.SYS /

смещ ение	имя	длина	комментарий
-8	DSIZE	2	Размер открытого устройства в блоках
-6	BMSIZ	2	Размер бит-карты в байтах (см.ПРИЛОЖЕНИ
-4	FCBA	2	Адрес описателя файла пользователя
-2		1	Временный номер файла
-1	FNUMB	1	Номер файла (#FF, если файл не открыт)
0		32	Описатель тек.(открытого) файла:
0	FNAME	8	Имя
8	FTYPE	3	Тип
11	FSTAT	1	Регистр состояния файла. Биты (0/1): 0 - удален/существует 2 - защищен от чтения(1) 3 - защищен от записи(1) 4 - видимый/скрытый файл 5 - файл/каталог (главный) 6 - сегментированный/непрерывный 7 - защищен от удаления(1)
12	FLSA	2	Адрес загрузки по умолчанию
14	FLENG	3	Длина
17	FSDBN	2	Номер блока описателя сегмента (Для непрерывного файла - номер нулевого блока файла)
19	WRKST	1	"Special" (N уровня для *.SYS)
20		6	Резерв
26	FSUM	2	Контрольная сумма файла
28	FTIME	2	Время
30	FDATE	2	Дата
32		23	Описатель тек.(открытого) каталога (внутренний)
32/0	CATLG	8	Имя текущего каталога
40/8		3	пробелы (тип каталога)
43/11		1	CSR каталога
44/12	CDLSA	2	CBBN каталога-прародителя
46/14	*CSIZE	2	Размер каталога в байтах
48/16	*LEVEL	1	Уровень вложенности подкаталога
49/17	CSDBN	2	Номер блока описателя сегмента
51/19	CBBN	2	Номер 0-го блока каталога
53/21	*NFILE	1	Общее число файлов (включая сам каталог и удаленные)
54/22	*NEFIL	1	Число файлов без каталога и удаленных
55		1	

* - неиспользуемые переменные во внешнем описателе каталога.

ПРИЛОЖЕНИЕ 10. Вектор окна для wt(#61)
(Подается в IX на входе большинства рестартов уровня WIND.SYS)

смещ ение	имя	длина	комментарии
0	X_COOR	1	координата X для распечатки окна
1	Y_COOR	1	координата Y окна
2	Y_SIZE	1	высота окна (размер Y)
3	X_SIZE	1	ширина окна (размер X)
4	ATTR	1	атрибуты окна
5	SHADE	1	атрибуты тени. (Если старший бит установлен, тень не выводится)
6	X_coor	1	координата X для распечатки текста. Указывает непосредственно на начальную позицию печати.
7	X_size	1	ширина для распечатки текста с помощью процедур lwt, adrwt, lenwt.

Все расстояния отсчитываются от левого верхнего угла экрана и измеряются в синклер-местах (от 1 до 32) (IX+0..+3) и в знакоместах (от 1 до 42) (IX+6..7)

Вектор часто включается в состав больших векторов (см. menu(#91), scrol(#70), ПРИЛОЖЕНИЕ 18)

ПРИЛОЖЕНИЕ 11. Вектор для строкового редактора
/ edstr(#7F) WIND.SYS /
подается в IX.

сме-щение	имя	длина	комментарии
-23	KDEV	1	Логический номер драйвера клавиатуры (ДК)
-22	KCSR	1	управляющий регистр для ДК (K_CSR)
-21	MCSR	1	..... (M_CSR) (см. ПРИЛОЖЕНИЕ 2)
-20	XS	1	X-координата начала редактируемой области экрана (РОЭ) в знакоместах
-19	YS	1	то же по оси Y.(левый верхний угол (ЛВУ) прямоугольной РОЭ относ-но ЛВУ экрана)
-18	LN	1	длина РОЭ в знакоместах
-17	HT	1	высота РОЭ
-16	SCSR	1	регистр состояния строкового редактора :

Биты 0/1:
- 7 - mod - признак модифицированной строки (1)
- 6 - s_e - smbgt/edstr
- 1 - c_i - change/insert
- 0 - t_d - text/document - отрабатывать(1) или нет(0) логические границы

-15	DBUF	2	адрес ASCII-буфера строки
-13	XC	1	текущая X-координата курсора в буфере
-12	XMAX	1	максимальная длина строки-2
-11	LM	1	левая логичесая граница
-10	RM	1	правая логическая граница
-9	XO	1	число пропускаемых символов
-8	SZ	1	ширина поля требования скроллинга
-7	CODE	1	код, возвращаемый редактором
-6	LNST	1	длина строки с <ENTER>
-5	COM	1	команда, к-рую должен выполнить редактор при входе
-4	TYPE	2	адрес подпрограммы отображения текущего состояния (если 0, то не вызывается)

KDEV: может принимать 2 значения: (0/1). При входе edstr переключается на указанный, на выходе восстанавливает текущий. При загрузки iS-DOS устанавливает 0-ой.

Программа edstr(#7F) является внутренней точкой входа рестарта smbgt(#6E) (см.ПРИЛОЖЕНИЕ 14).

Предполагалось его использование в Базах Данных прямо в файловых панелях, для чего их вектора совместили (см. ПРИЛОЖЕНИЕ 18).

(Коды, с которыми edstr возвращается см. ПРИЛОЖЕНИЕ 13).

ПРИЛОЖЕНИЕ 12

Некоторые коды клавиш формируемые ttyin.key

Код	клавиша	Функция
4	CS/3	Ins/Ovr
5	CS/4	Home
6	CS/2	BIG/smal
7	CS/1	Рус/Lat
8	CS/5	Left
9	CS/8	Right
A	CS/6	Down
B	CS/7	Up
C	CS/0	Delete
D	ENTER	
E	CS/SS	Macro
F	CS/9	End
10	SS/A	PgDn
11	SS/ENTER	Prefix
12	SS/SPACE	Prefix
13	SS/W	Erase
14	SS/E	Insert
15	SS/Q	PgUp
16	CS/SPACE	
17	CS/ENTER	Tabulator

ПРИЛОЖЕНИЕ 13. Коды , формируемые edstr(#7F)
/ WIND.sys /

Xmax = 32 - попытка передвинуть курсор (ППК)
за правую физическую границу (ФГ) XMAX
Xmin = 33 - ППК за левую ФГ (0)
Pdel = 34 - попытка удаления за левой ФГ

Rmrg = 35 - ППК за правую логическую границу (ЛГ) RM
Lmrg = 36 - ППК за левую ЛГ (LM)
Ldel = 37 - попытка удаления за левой ЛГ

Rscr = 38 - требование правого скроллинга
Lscr = 39 - требование левого скроллинга
Sdel = 40 - попытка удаления за окном

Pins = 41 - попытка выдвинуть текст за правую ФГ
Lins = 42 - попытка выдвинуть текст за правую ЛГ
Elns = 43 - длина строки превышает XMAX+2

ПРИЛОЖЕНИЕ 14. Вектор 3-го уровня "Оконная технология"
/ g_scr(#72) WIND.SYS /

смещение	имя		длина	комментарии
-2			2	адрес таблицы ф-ций пользователя для panel(#6A) (см.ПРИЛОЖЕНИЕ 15.)
0	PAPER		1	атрибуты экрана
1	BORD		1	атрибуты бордюра
2	....		9	Не используются
	параметры для smbgt(#6E):			
11	SMDRV		1	Логический номер драйвера клавиатуры(ДК)
12	SMKCS	*	1	управляющий регистр для ДК (K_CSR)
13	SMMCS	*	1	M_CSR (см. ПРИЛОЖЕНИЕ 2.)
14	SMXS	*	1	левая верхняя координаты редактируемой
15	SMYS	*	1	строки на экране в знакоместах
16	SMLN	*	1	длина редактируемой области экрана
17	SMHT	*	1	высота редактируемой области экрана
18	SMCSR		1	регистр состояния строкового редактора. Биты 0/1:
		*	6	- s_e - smbgt/edstr - режим работы (0)
			1	- c_i - change/insert
19	SMDBF	*	2	адрес строки
21	SMXC	*	1	тек X-коор курсора в тексте
22	SMXMX	*	1	максимальная длина строки-2 (LN*HT-1)
23	SMLM	*	1	левая лог граница (0)
24	SMRM		1	правая лог граница
25	SMXO	*	1	число пропускаемых символов (0)
26	SMSZ		1	ширина поля требования скроллинга
27	SMCOD		1	код, возвращаемый редактором
28	SMLNS		1	длина строки с кодом 13
29	SMCOM		1	команда, к-рую должен выполнить редактор при входе
30	SMTYP		2	адрес подпрограммы отображения тек. состояния (если 0, то не вызывается)

Параметры для строкового редактора smbgt полностью дублируют вектор для edstr(#7F), в чем Вы можете убедиться, ознакомившись с ПРИЛОЖЕНИЯМИ 11 и 18.
Значения всех границ могут изменяться в пределах от 0 до 253
* Все параметры отмеченные звездочками smbgt инициализирует сам

ПРИЛОЖЕНИЕ 15. ПОДПРОГРАММЫ ФУНКЦИИ panel(#6A) - РАБОТА С ПАНЕЛЬЮ / WIND.SYS /

Рестарт имеет 21 точку входа, причем первые 12 функций необходимы для его поддержки (он сам обращается к этим функциям), а остальные задают режим работы. Адреса ф-ций 0..11 сведены в таблицу, адрес которой лежит в векторе WIND.SYS со смещением -2 (см. ПРИЛОЖЕНИЕ 14).

Т.о. пользователь может подменить эти 12 ф-ций на свои, что и делают некоторые Базы Данных. Этим же рестартом пользуется оболочка (уровень SHELL.SYS), рисуя на экране свои панели с файлами. Рестарт вынесен в уровень WIND.SYS, что предполагает возможность его использования даже при отсутствии уровня SHELL.SYS.

Нужная функция выбирается с помощью регистра A.

0. fprev - по входному номеру записи в регистре DE найти предыдущую запись.
   Выход : Флаг c -ошибка ввода;
   z -успешно (DE'= номер)
   nz -запись отсутствует.
   (DE'= прежний номер записи)
1. fnext - по входному номеру записи в регистре DE найти следующую запись.
   Выход : Флаг c -ошибка ввода;
   z -успешно (в DE' - номер)
   nz -запись отсутствует.
   (DE'= прежний номер записи)
2. pread - заполнить буфер текущей записью
3. g_rcst - вернуть в регистр A состояние записи
4. p_rcst - установить состояние текущей записи равным рег. B
5. g_no - вернуть в регистр A номер отметки
6. p_no - установить номер отметки текущей записи равным рег. B
7. g_min - вернуть в DE' минимальный номер записи
8. g_max - вернуть в DE' максимальный номер записи
9. crcor - установка курсора
10. tssel - проверка можно ли отмечать данную запись. Z-да, NZ-нет.
11. tsres - проверка на маску.
    Z - подходит, NZ -нет, C -нет.
12. inipnl - самостоятельная инициализация вектора панели. C - системная ошибка.
13. ctman - печать панели с инициализацией начала страницы.
14. ctmin - то же, но конца страницы.
15. tymin - скроллинг вверх с печатью нижней строки.
16. tymax - скроллинг вверх с печатью верхней строки.
17. fcmon - движение по панели.
18. cname - печать текущей записи на месте курсора.
19. oncrs - зажигание курсора.
20. ofcrs - тушение курсора.
21. crset - установка курсора на заданный номер записи и желаемую позицию

ПРИЛОЖЕНИЕ 16. Команды оболочки / SHELL.SYS /

Код	Клавиш	Имя	Примечание
0(00)		fcmon	- выход без каких-л спец. действий
5(37)	"7"	mkdir	- создание подкаталога на текущей панели
6(06)	CS/2	rtdev	- выбор устройства на правой панели
7(07)	CS/1	lfdev	- выбор устройства на левой панели
8(08)	CS/5	plpan	- перемещение курсора на левую панель без открытия устройства
9(09)	CS/8	prpan	- перемещение курсора на правую панель без открытия устройства
13(0D)	ENTER	curun	- запуск файла, на котором стоит курсор, а также вход и выход из каталога
14(0C)	CS/SS	dbpnl	- копирование текущей панели на альтернативную
54(36)	"6"	view	- запуск программы просмотра согласно файлу S:SHELL\extview.txt
56(38)	"8"	gmenu	- вызов главного меню
79(4F)	"O"	nlpan	- перемещение курсора на левую панель с открытием устройства
80(50)	"P"	nrpan	- перемещение курсора на правую панель с открытием устройства
104(68)	"h"	prnt	- запуск программы печати согласно файлу S:SHELL\extprint.txt
111(6F)	"o"	plpan	- то же, что 8 - CS/5
112(70)	"p"	prpan	- то же, что 9 - CS/8

Код	pnCSR	Имя	Примечание
240(F0)	01110	cvwnel	- перепечатка текущей панели (ПТП) с сохранением позиции курсора (СПК)
241(F1)	01010	nvwpnl	- ПТП с установкой курсора на начало панели
242(F2)	01110	cshel2	- аналог shel2(#82) с СПК на обеих панелях (СПК-2П)
243(F3)	01010	nshel2	- полный аналог рестарта shel2
244(F4)	01110	cshel1	- аналог shel1(#81) с СПК-2П
245(F5)	01010	nshel1	- полный аналог рестарта shel1
246(F6)	01110	cshel0	- аналог shel0(#80) с СПК-2П
247(F7)	01010	nshel0	- полный аналог рестарта shel0
248(F8)	00110	_panel	- ПТП с сохранением отметки (СО)
251(FB)	11010	new	- ПТП с открытием устройства

Выполнить все эти команды можно если вернуться в оболочку с флагом Z и NC по RET или по рестарту shout(#84). Чтобы выполнить данные команды и остаться в прикладной задаче пользуйтесь рестартом shsubr(#8E). Эти же команды отрабатывает оболочка при опросе клавиатуры. Все остальные коды клавиш оболочка ищет в файле S:SHELL\extkey.txt. Рестарт shsubr инициализирует панель при ее перепечатке. После чего при повторном его вызове с любым кодом кроме перечисленных (например 255) он перехватит управление и будет сам отрабатывать движение по панели вверх и вниз. По любой клавише кроме A, Q,CS/A,CS/Q он вернет управление.

Запуска файлов рестарт shsubr не производит.

В случае вызова по рестартам флаг Z не существенен.

Биты регистра pnCSR см. ПРИЛОЖЕНИЕ 17.

ПРИЛОЖЕНИЕ 17. Вектор оболочки
/ g_mpan(#87)-BC' SHELL.SYS /

смещение	имя	длина	комментарий
-9	SHLDAT	2	Дата трансляции SHELL.SYS
-7	ERCSR	1	Управляющий байт для shexe(#93) 0-не/1-сообщить об ошибке 81 (нет файла)
-6	FNC	1	Код ошибки или клавиши
-5	nLOG	1	Управление подкраской строки подсказок 1 - красить, 0 - нет / mmenu(#89) /
-4	pnCSR	1	Регистр состояния печати панелей. Биты: 0 - pn - временный номер панели 1 - sw - разрешение (требование) смены панели(1) 2 - crs - сброс(0)/сохранение(1) позиции курсора 3 - in - сброс(1)/сохранение(0) отметки 4 - nw - открытие устройства(1) 5 - err - наличие ошибок (внутр. употребления) / pnCSR подается в регистре B на входе в shpanl(#90) /
-3	DFANL	2	Адрес анализатора 12 ф-ций (см. ПРИЛОЖЕНИЕ 15.)
-1	NPANL	1	номер панели / g_curs(#8A)-B' /
0	CHPAN	2	NN каналов левой и правой панелей / g_mpan(#87)-BC' /
2	PAPER	1	цвет верхней части экрана
3	GRAND	1	цвет бордюра
4	MWCOL	2	цвета надписей "Please Wait" лев. и прав.
6	FCURS	2	
8	Pshl	8	Путь: "S:SHELL\"

ПРИЛОЖЕНИЕ 18. Вектор панели ("плавающее меню")
/ g_curs(#8A) SHELL.sys /

смещение	имя	длина	комментарии
-48	CURDV	1	Номер устройства панели /g_mpan(#87)-DE'
-47	CRCAT	2	Номер каталога панели
-45	WAYS	1	Номер канала хранящего путь панели
-44	PCSR	1	регистр состояния панели. Биты (1): 0 - dubl - обязательная печать обеих панелей 1 - ect - запрет печати каталогов 2 - crt - сохранение позиции курсора при перепечатке панели 3 - dir - текущее направление движения курсора 4 - sort - сортировка разрешена 5 - cat - запрет печати каталогов,если они не подходят под маску 6 - otm - отметка разрешена 7 - emp - панель пустая
-43	RMIN	2	Минимальный подходящий номер записи
-41	RMAX	2	Максимальный подходящий номер записи
-39	CMIN	2	Номер записи открывающий страницу
-37	CMAX	2	Номер записи закрывающий страницу
-35	POS	1	позиция курсора
-34	REC	2	текущий номер записи
-32	DFAL	2	текущее количество подходящих записей
-30	DFSL	2	текущее количество отмеченных записей
-28	MAT	1	цвет признака конца/начала существующих записей
-27	ARAL	1	цвет записи
-26	ACAL	1	цвет курсора на записи
-25	ARSL	1	цвет отмеченной записи
24	ACSL	1	цвет курсора на отмеченной записи
-23	....	21	Вектор edstr(#7F) (см. ПРИЛОЖЕНИЕ 11)
	TYPE	2	Адрес процедуры, вызываемой при движении курсора по панели
-2	Delay	1	Задержка скроллинга (0-255) /scrol(#70)/
-1	Scrol	1	Скорость графического вертикального скроллинга (1/2/4/8)
0	....	8	Стандартный вектор wt (см. ПРИЛОЖЕНИЕ 10)
8		12	Шаблон-маска панели
20	FLCUR	12	Имя, тип и CSR файла под курсором

ПРИЛОЖЕНИЕ 19. КОДЫ ОШИБОК

 0 - нецифровой символ /a_d(#7D)/
 1 - переполнение числа /a_d/ или буфера /d_a(#7C)/

Драйвер дисковода:

 6 - нет диска
 7 - ошибка на диске (сектор не читается/не пишется)
 9 - не iSDOS-диск /binit(#F)/
10 - несоответствие диска и дисковода /binit(#F)/
20 - BREAK (нажат CS/SPACE)

junc.com:

22 - неверная таблица
23 - некорректный ключ

28 - драйвер принтера не установлен (lp_init.com)
31 - запрещенное имя файла /fncor(#50)/
33 - неверное имя устройства (mkdir.com)
34 - неверный размер устройства (device.com)
36 - синтаксическая ошибка (set.com)
37 - нет резидентной задачи /exebat(#44),run(#48),
fndev(#51),opres(#52),exeres(#53)/
39 - фоновая задача отсутствует при попытке удалить ее
/l_im2(#1E)/
50 - неверное имя текстового файла для cnfg.com

Виртуальный диск:

61 - указанный блок не найден
62 - реальное кол-во модифицированных блоков не
соответствует значению ($QNMOD)
/flush(2),modo(#2E)../
63 - сектор защищен
64 - диск защищен
65 - нет места на диске
66 - чтение запрещено
67 - запись запрещена

tv.com, ed.res, tab.com, etc.

70 - попытка записи в режиме "view"
71 - ошибочное значение указателя
72 - длина строки превышает 255

79 - неверное имя устройства в командах L_S, L_Q, L_T

Файловая служба:

80 - номер файла слишком велик /opnum,putf,gname/
81 - файл не найден /find(#34),fopen(#25)/
82 - файл уже существует /crfil(#23)/
83 - каталог не пуст (при его удалении) /erfil,erf__/
84 - каталог полон (127 файлов) /crfil,crf__(#3B)/
85 - плохой блок описателя сегмента файла
86 - нет каталога /fopen,open1(#21)/
87 - нет устройства /open(#20)/
88 - файл уже удален /erfil(#24),erf__/
89 - файл защищен от удаления (erfil,erf__)
92 - на диске нет требуемого непрерывного пространства
/distr(#38),crfil,fadd(#2F),eadd(#31)../
93 - попытка освободить блоки за концом диска
/free(#3F),erfil,erf__/
94 - переполнение блока описателя сегментов файла
(85 сегментов) /fadd,fcut(#30),eadd/

98 - нет такой функции в RST panel(#6A)

99 - несоответствие номера блока в заголовке блока при
попытке его модифицирования (вирт. диск)
100 - конец файла по длине в байтах /rpart,wpart/
101 - конец файла в блоках.
/qrvbl,rifle,wifle,rvblk,wvblk/
102 - файл непрерывен /fadd,fcut,eadd/
103 - добавить или удалить нулевую длину
/fadd,fcut,eadd,ecut/
104 - удаление из файла фрагмента длиной больше, чем
файл /fcut(#30),ecut(#32)/

```
105 - дальнейшее увеличение непрерывного каталога
      затруднительно /crfil(#23),crf__(#3B)/
106 - файл не открыт /qrvbl,rpart,wpart,rifle../
107 - Bit Map размера 0 (ошибка открытия устройства)
108 - Файл в начале устройства (номер блока файла или
      его описателя меньше 3 для обычного диска)
      /qrvbl,wpart,rifle../

   swkey(#1A),swtyp(#1B),swblk(#1C),ldnew(#1D):

121 - нет устройства
122 - нет драйвера, обслуживающего данное устройство

    функции работы с каналами:

124 - неверная структура области каналов
125 - нет канала /stchn(#16),delchn(#17)/
126 - канал уже есть /crchn(#18)/
127 - ошибочная длина /crchn(#18)/
128 - нет места для канала /crchn(#18),newchn(#19)/
129 - нет свободного номера /newchn(#19)/

        инсталляция/удаление новых
       уровней/драйверов/устройств. (SET.com)

130 - не хватает памяти /set.com, exebat(#44),run(#48)/
131 - нет драйвера /set.com/
132 - попытка инсталлировать (ПИ) 9-ый драйвер
133 - ПИ 9-ое устройство данного типа
135 - неправильная установка уровня

138 - ошибка в рекурсивной цепочке (неверный номер
      канала) /rcdel(#4D)/
139 - слишком длинная командная строка /p_com(#49)/

        prstr(#68),tystr,xystr:

140 - нет имени в массиве
141 - нет строки в тексте
142 - ошибка в структуре рабочей области

      процедуры работающие с блоками

160 - попытка удалить из файла 0 блоков /B^DEL/
161 - попытка добавить к файлу 0 блоков /B^ADD/
162 - попытка занять на устройстве 0 блоков
      /distr(#38)/
163 - попытка освободить на устройстве 0 блоков
      /free(#3F)/
164 - попытка открепить блок с номером меньше 6
      /free(#3F),erfil(#24)/

      Файловая служба (окончание)

170 - Чтение/запись 0 байт в файл /rpart,wpart/
171 - Чтение файла запрещено /qrvbl,rpart,wpart/
172 - Запись в файл запрещена /wpart(#2A)/

222 - незагрузочный диск
226 - неизвестная версия TRDOS
```

ПРИЛОЖЕНИЕ 20.

Альтернативная кодировка ГОСТа

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
0				0	@	P	'	p	А	Р	а	▒	└	╨	р	Ё	0
1			!	1	A	Q	a	q	Б	С	б	▒	┴	╤	с	ё	1
2			"	2	B	R	b	r	В	Т	в	▓	┬	╥	т		2
3			#	3	C	S	c	s	Г	У	г	│	├	╙	у		3
4			$	4	D	T	d	t	Д	Ф	д	┤	─	╘	ф		4
5			%	5	E	U	e	u	Е	Х	е	╡	┼	╒	х		5
6			&	6	F	V	f	v	Ж	Ц	ж	╢	╞	╓	ц		6
7			'	7	G	W	g	w	З	Ч	з	╖	╟	╫	ч		7
8			(	8	H	X	h	x	И	Ш	и	╕	╚	╪	ш		8
9			)	9	I	Y	i	y	Й	Щ	й	╣	╔	┘	щ		9
A			*	:	J	Z	j	z	К	Ъ	к	║	╩	┌	ъ		A
B			+	;	K	[	k	{	Л	Ы	л	╗	╤	■	ы		B
C			,	<	L	\	l	|	М	Ь	м	╝	╠	▄	ь		C
D			-	=	M	]	m	}	Н	Э	н	╜	═	▌	э		D
E			.	>	N	^	n	~	О	Ю	о	╛	╬	▐	ю		E
F			/	?	O	_	o		П	Я	п	┐	╧	▀	я		F
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	

10. СИСТЕМА КОМАНД ПРОЦЕССОРА Z80

Сводная таблица команд Z-80

Код операции	Функция
ADC A	Сложение и запись в аккумулятор
ADC HL,rp	Сложение содержимого двух регистров и запись результата в пару регистров HL
ADD	Сложение
AND	Логическое И
BIT	Проверка бита
CALL addr	Вызов подпрограммы
CALL cond,addr	Вызов подпрограммы по условию
CCF	Инверсия флажка переноса дополнения
CP	Сравнение
CPD	Сравнение с автоуменьшением адреса
CPDR	Сравнение с автоуменьшением и повторением
CPI	Сравнение с автоувеличением адреса
CPIR	Сравнение с автоувеличением, повторение
CPL	Запись содержимого дополнения в аккумулятор
DAA	Запись в аккумулятор
DEC	Уменьшение счетчика адреса
DI	Запрещение прерываний
DJNZ	Если не нуль, уменьшение счетчика адреса и перескок через 1 команду
EI	Разрешение прерываний
EX	Выполнение обмена
EXX	Обмен содержимого пары регистров и объявление этой пары альтернативной
HALT	Останов
IM	Установка режима работы с прерываниями
IN	Ввод
INC	Увеличение счетчика адреса
IND	Ввод с автоуменьшением адреса
INDR	Ввод с автоуменьшением и повторением
INI	Ввод с автоувеличением и повторением
JP addr	Переход по указанному адресу
JP cond,addr	Переход по условию
JR	Переход по относительному адресу
JR cond,addr	Условный переход по относительному адресу
LD A,(addr)	Загрузка непосредственная в аккумулятор
LD A,(BC) или (DE)	Загрузка в аккумулятор косвенная
LD A,I	Загрузка в аккумулятор из регистра вектора прерываний
LD A,R	Загрузка в аккумулятор из регистра регенерации памяти
LD HL,(addr)	Загрузка пары регистров непосредственная
LD data	Загрузка непосредственная
LD xy,(addr)	Загрузка прямая индексного регистра
LD reg,(HL)	Загрузка регистра
LD reg,(xy+disp)	Загрузка регистра с индексацией
LD rp,(addr)	Загрузка пары регистров непосредственная
LD SP,HL	Запись в указатель стека содержимого регистров HL
LD SP,xy	Запись содержимого индекс-регистра в указатель стека
LD dst,scr	Запись из регистра в регистр
LD (addr),A	Запись в память содержимого аккумулятора непосредственная
LD (BC) или (DE),A	Запись в память содержимого аккумулятора косвенная
LD I,A	Запись содержимого аккумулятора в регистр вектора прерываний
LD R,A	Запись содержимого аккумулятора в регистр регенерации памяти
LD (addr),HL	Запись непосредственная в память содержимого пары HL
LD (HL),data	Запись непосредственная в память
LD (xy+dis),dat	Запись непосредственная в память с индексацией
LD (addr),xy	Запись непосредственная в память содержимого индекс-регистра
LD (HL),reg	Запись в память содержимого регистра
LD (xy+dis),reg	Запись в память содержимого регистра с инде-

	ксацией
LD (addr),rp	Запись непосредственная в память содержимого пары регистров
LDD	Загрузка с автоуменьшением адреса
LDDR	Загрузка с автоуменьшеним и повторением
LDI	Загрузка с автоувеличением адреса
LDIR	Загрузка с автоувеличением адреса и повтором операций
NEG	Формирование дополнительного кода для аккумулятора
NOP	Нет операции
OR	Логическое ИЛИ
OUT	Вывод
OUTD	Вывод с автоуменьшением
OTDR	Вывод с автоуменьшением и повторением
OUTI	Вывод с автоувеличением
OTIR	Вывод с автоувеличением и повторением
POP	Выталкивание из стека
PUSH	Заталкивание в стек
RES	Сброс бита
RET	Возврат из подпрограммы
RET cond	Возврат по условию
RETI	Возврат из обработки прерывания
SETN	Возврат из немаскированного прерывания
RL	Циклический сдвиг влево с переносом
RLA	Циклический сдвиг влево содержимого аккумулятора с переносом
RLC	Циклический сдвиг влево
RLCA	Циклический сдвиг влево содержимого аккумулятора
RLD	Циклический сдвиг влево содержимого аккумулятора и памяти десятичный
RR	Сдвиг вправо с переносом
RRA	Сдвиг содержимого аккумулятора вправо с переносом
RRC	Сдвиг вправо циклический
RRCA	Сдвиг вправо циклический содержимого аккумулятора
RRD	Сдвиг вправо десятичный содержимого аккумулятора и памяти
RST	Рестарт
SET	Установка бита
SBC	Вычитание с переносом (с заимствованием из старших разрядов)
SCF	Установка флажка переноса
SLA	Сдвиг влево арифметический
SRA	Сдвиг вправо арифметический
SRL	Сдвиг вправо логический
SUB	Вычитание
XOR	Логическое исключающее ИЛИ

10.1 МЕТОДЫ АДРЕСАЦИИ

Безоперандные команды. Для их выполнения процессору не требуется дополнительных значений:

HALT

NOP

Регистровая адресация. Операнд содержится в регистре, который непосредственно указан в команде:

INC A

ADD HL,BC

Непосредственная адресация. Операнд расположен в ячейке памяти, непосредственно следующей за кодом команды. Если операнд двухбайтовый, то он занимает две ячейки за кодом команды (расширенная непосредственная адресация):

LD C,%1011

LD HL,#1234

Неявная адресация. Регистр, в котором содержится операнд, однозначно определен данной командой:

RLA

LDIR

Регистровая косвенная адресация. Операнд расположен в ячейке памяти, адрес которой содержится в регистровой паре:

DEC (HL)

LD A,(BC)

Индексная адресация. Адрес размещения в памяти операнда образуется сложением значения индексного регистра и некоторого смещения (-128...+127), явно указанного в команде:

SLA (IX-2)

LD A,(IY-25)

Косвенная адресация. Адрес операнда содержится в ячейках памяти, расположенных непосредственно за кодом команды:

LD A,(1234)

LD (#C000),DE

Побитовая адресация. Командой задается конкретный бит операнда, над которым будет проводиться операция:

SET 4,(HL)

BIT 7,A

Относительная адресация. Адрес задается как однобайтовое смещение (-128...+127) относительно текущего значения регистра счетчика команд (определяется символом счетчика адресов "$"):

JR $+#23

DJNZ $-6

Расширенная адресация. Адрес задается двухбайтовым операндом:

CALL 1234

JP NC,#1601

Различные методы адресации могут быть совмещены в командах, работающих с двумя операндами:

LD (HL),12

10.2 ОПИСАНИЕ КОМАНД

ADC	HL,SS	Сложить с переносом содержимое пары регистров SS с содержимым регистра HL.
ADC	A,S	Сложить с переносом операнд S с содержимым

накапливающего регистра.

ADD	A,N	Сложить величину N с содержимым накапливающего регистра.
ADD	A,R	Сложить содержимое регистра R с содержимым накапливающего регистра.
ADD	A,(HL)	Сложить содержимое ячейки памяти (HL) с содержимым накапливающего регистра.
ADD	A,(IX+d)	Сложить содержимое ячейки памяти (IX+d) с содержимым накапливающего регистра.
ADD	A,(IY+d)	То-же для ячейки (IY+d).
ADD	HL,SS	Сложить содержимое пары регистров SS с содержимым накапливающего регистра HL.
ADD	IX,PP	Сложить содержимое пары регистров PP с содержимым регистра IX.
ADD	IY,PP	ТО-же для регистра IY.
AND	S	Выполнить операцию логического сложения И над операндом S и содежимым накапливающего регистра.
BIT	B,(HL)	Тестировать разряд B из ячейки памяти (HL).
BIT	B,(IX+d)	То-же.
BIT	B,(IY+d)	То-же.
BIT	B,R	Тестировать разряд из регистра R.
CALL	CC,NN	Вызвать подпрограмму, с ячейки NN, если условие CC является истинным.
CALL	NN	Вызвать безусловно подпрограмму с адреса NN.
CCF		Установить обратный код во флажке переноса.
CPS		Сравнить операнд S с содежимым переноса.
CPD		Сравнить содержимое ячейки памяти (HL) с содержимым накапливающего регистра, сообщить отрицательное приращение содержимым регистров HL и BC продолжать процедуру до выполнения условия BC=0.
CPD	R	То-же.
CPI		То-же ,сообщить положительное приращение содержимому регистров HL и отрицательное-содержимому регистра BC.
CPIR		То-же, сообщить положительное приращение содер-

жимому регистра HL и отрицательное-содержимому регистра BC, продолжать процедуру до выполнения условия BC=0.

CPL — Дополнить накапливающий регистр (дополнение до 1).

DAA — Десятичная корректировка содержимого накапливающего регистра.

DEC M — Сообщить отрицательное приращение операнду M.

DEC IX — То-же для содержимого регистра IX.

DEC IY — То-же для регистра IY.

DEC SS — То-же для содержимого пары регистров SS.

DI — Запретить прерывания.

DJNC E — Сообщить отрицательное приращение содержимому регистра B и передать управление на относительный адрес при выполнении условия B<>0.

EI — Разрешить прерывания.

EX (SP),HL — Произвести обмен содержимого между ячейкой памяти (SP) и регистром HL.

EX (SP),IX — То-же и регистром IX.

EX (SP),IY — То-же и регистром IY.

EX AF,AF' — Произвести обмен содержимым между регистром AF и регистром AF'.

EX DE,HL — То-же между ячейкой регистра DE и регистром HL.

EXX — Произвести обмен содержимого между регистрами BC,DE и HL и соответственно регистрами BC',DE', и HL'.

HALT — Приостановить работу (ожидать запроса прерывания или сброса).

IM0 — Установить режим прерывания 0.

IM1 — То-же 1.

IM2 — То-же 2.

IN A,(N) — Загрузить накапливающий регистр входными данными от устройства N.

IN R,(C) — Загрузить регистр R входными данными от устройства C.

INC (HL) — Сообщить положительное приращение содержимому

памяти (HL).

INC	IX	То-же для регистра IX.
INC	(IX+d)	То-же для ячейки памяти (IX+d).
INC	IY	То-же для регистра IY.
INC	(IY+d)	То-же для ячейки памяти (IY+d).
INC	R	То-же для содержимого регистра R.
INC	SS	То-же для пары регистров SS.
IND		Загрузить ячейку памяти (HL) входными данными порта (C), сообщить отрицательное приращение содержимым регистров HL и B.
INI		То-же, сообщить положительное приращение содержимому регистра HL и отрицательное приращение содержимому регистра B.
INDR		То-же, сообщить отрицательное приращение содержимому регистра HL и регистра B, повторять эту процедуру до выполнения условия B=0.
INIR		То-же, сообщить положительное приращение регистру HL и отрицательное регистру B, повторять процедуру до выполнения условия B=0.
JP	(HL)	Безусловная передача управления в (HL).
JP	(IX)	То-же в (IX).
JP	(IY)	То-же в (IY).
JP	CC,NN	Передать управление в ячейку памяти NN, при условии, что значение CC явл. истинным.
JP	NN	Безусловная передача управления.
JP	C,E	Безусловная передача управления в PC+E при условии, что перенос =1.
JP	E	Безусловная передача управления по относительному адресу PC+E.
JP	NC,E	То-же если перенос =0.
JR	NZ,E	То-же, при условии отсутствия нуля (Z=0).
JP	Z,E	То-же, при нулевом условии (Z=1).
LD	A,(BC)	Загрузить накапливающий регистр содержимым ячейки памяти (BC).
LD	A,(DE)	То-же (DE).

LD	A,I	То-же содержимым регистра I.
LD	A,(NN)	То-же содержимым ячейки памяти NN.
LD	A,R	То-же содержимым регистра R.
LD	(BC),A	Загрузить ячейку памяти (BC) содержимым накапливающего регистра.
LD	(HL),N	Загрузить ячейку памяти (HL) величиной N.
LD	DD,NN	Загрузить пару регистров DD величиной NN.
LD	HL,(NN)	Загрузить ячейку памяти HL содержимым ячейки (NN).
LD	(HL),R	То-же содержимым регистра R.
LD	I,A	Загрузить регистр I содержимым накапливающего регистра A.
LF	IX,NN	Загрузить регистр IX вличиной NN.
LD	IX,(NN)	То-же содержимым ячейки памяти (NN).
LD	(IX+d),N	Загрузить в ячейку памяти (IX+d) величину N.
LD	(IY+d),R	То-же содержимое регистра N.
LD	IY,N	Загрузить в регистр IY величину N.
LD	IY,(N)	То-же содержимое ячейки (N).
LD	(IY+d),N	Загрузить в ячейку памяти (IY+d) величину N.
LD	(IY+d),R	То-же содержимое регистра R.
LD	(NN),A	То-же содержимое накапливающего регистра.
LD	(NN),DD	ТО-же содержимое пары регистров DD.
LD	(NN),HL	То-же содержимое регистра HL.
LD	(NN),IX	То-же регистра IX.
LD	(NN),IY	То-же регистра IY.
LD	R,A	Загрузить в регистр R содержимое накапливающего регистра.
LD	R,(HL)	То-же содержимое ячейки памяти (HL).
LD	R,(IX+d)	То-же содержимое ячейки памяти (IX+d).
LD	R,(IY+d)	То-же.
LD	R,N	То-же величину N.
LD	R,R'	То-же содержимое регистра R'.

LD	SP,HL	Загрузить в ячейку памяти содержимое регистра HL.
LD	SP,IX	То-же содержимое регистра IX.
LD	SP,IY	То-же содержимое регистра IY.
LDD		Загрузить ячейку памяти (DE) содержимым ячейки памяти (HL), сообщить отрицательное приращение содержимому регистров DE,HL,BC.
LDDR		То-же повторять процедуру до B=0.
LDI		То-же сообщить положительное приращение содержимому регистров DE,HL и отрицательное содержимому регистра BC.
LDIR		То-же сообщить положительное приращение содержимому регистров DE,HL и отрицательное приращение- содержимому регистра BC; повторять до выполнения условия BC=0.
NEG		Выполнить операцию логического отрицания над содержимым накапливающего регистра (поразрядное дополнение до двух).
NOP		Пустая операция.
OR	S	Логическая операция ИЛИ над операндом S и содержимым накапливающего регистра.
OTDR		Загрузить выходной порт (C) содержимым ячейки памяти (HL), сообщить отрицательное приращение содержимому регистров HL и B; повторять до выполнения условия B=0.
OTIR		То-же, сообщить отрицательное приращение содержимому регистра B, сообщить положительное приращение содержимому регистра HL, повторять до выполнения условия B=0.
OUT	(C),R	Загрузить выходной порт (C) содержимым регистра R.
OUT	(N),A	Загрузить выходной порт (N) содержимым регистра накопления.
OUTD		Загрузить выходной порт (C) содержимым ячейки (HL), сообщить отрицательное приращение содержимому регистров HL и B.
OUTI		То-же, сообщить положительное приращение содержимому регистра HL и отрицательное приращение содержимому регистра B.
POP	IX	Загрузить в регистр IX содержимое ячейки стека (верхней части).

POP	IY	То-же.
POP	QQ	Загрузить в пару регистров QQ содержимое верхней ячейки стека.
PUSH	IX	Протолкнуть содержимое регистра IX в стек.
PUSH	IY	То-же.
PUSH	QQ	То-же пары регистров QQ.
RES	B,M	Сбросить разряд B операнда M.
RET	T	Возврат после завершения стандартной п/п.
RET	CC	То-же, если условие CC истинно.
RETN		То-же после завершения немаскируемого прерывания.
RL	M	Циклический сдвиг влево через разряд переноса операнда M.
RL	A	То-же содержимого накапливающего регистра.
RLC	(HL)	То-же, минуя разряд переноса содержимого ячейки памяти (HL).
RLC	(IX+d)	То-же.
RLC	(IY+d)	То-же, минуя разряд переноса.
RLCA		Циклический сдвиг влево содержимого накапливающего регистра, минуя разряд переноса.
RLD		Циклический сдвиг влево и вправо в контуре, образованном накапливающим регистром и ячейкой (HL).
RRA		Циклический сдвиг вправо содержимого накапливающего регистра через разряд переноса.
RR	M	То-же операнда M через разряд переноса.
RRC	M	То-же содержимого накапливающего регистра через разряд переноса.
RRCA		То-же, минуя разряд переноса.
RRD		Циклический сдвиг разряда вправо и влево в контуре, образованном накапливающим регистром и ячейкой памяти (HL).
RST	P	Рестарт к ячейке памяти P.
SBC	A,S	Вычесть операнд S из содержимого накапливающего регистра с переносом.
SBC	HL,SS	Вычесть содержимое пары регистров SS из

содержимого регистра HL с переносом.

SCF — Установить флажок переноса (C=1).

SET B,(HL) — Установить разряд B ячейки памяти (HL).

SET B,(IX+d) — То-же ячейки памяти (IX+d).

SET B.(IY+d) — То-же.

SET B,R — То-же регистра R.

SLA M — Арифметический сдвиг операнда M влево.

SLA,M — То-же в право.

SRL,M — Логический сдвиг операнда M вправо.

SUB S — Вычесть операнд S из содержимого накапливающего регистра.

XOR S — Выполнить операцию логического объединения по схеме "Исключающее ИЛИ" над операндом S и содержимым накапливающего регистра.

10.3 Мнемоника команд процессора 8080

Код операции	Функция
ADC,ACI	Сложение с переносом
ADD,ADI	Сложение
ANA,ANI	Логическое И
CALL	Вызов подпрограммы
CC	Вызов по условию "перенос состоялся"
CM	Вызов по условию "минус"
CMA	Сформировать дополнительный код содержимого аккумулятора
CMC	Перенос дополнительного кода
CMP,CPI	Сравнение кодов
CNC	Вызов по условию "отсутствие переноса"
CNZ	Вызсв по условию "не нуль"
CP	Вызов по условию "положительный"
CPE	Вызов по условию "четный"
CPO	Вызов по условию "нечетный"
CZ	Вызов по условию "нуль"
DAA	Десятичное округление
DAD	Сложение шестнадцатиричное
DCR	Автоуменьшение адреса
DCX	Автоуменьшение по основанию 16
DI	Запрет прерываний
EI	Разрешение прерываний
HLT	Останов
IN	Ввод
INR	Автоувеличение адреса
INX	Автоувеличение по основанию 16
JC	Переход по условию "перенос состоялся"
JM	Переход по условию "минус"
JMP	Переход безусловный
JNC	Переход по условию "не было переноса"
JNZ	Переход по условию "не нуль"
JP	Переход по условию "плюс"
JPE	Переход по условию "контроль по четности"
JPO	Переход по условию "контроль по нечетности"
JZ	Переход по условию "нуль"
LDA	Загрузка аккумулятора
LDAX	Загрузка аккумулятора косвенная
LHLD	Прямая загрузка пары регистров HL
LXI	Загрузка 16-битовая
MOV	Сдвиг в памяти
MVI	Сдвиг в памяти непосредственный
NOP	Нет операции
ORA,ORI	Логическое ИЛИ
OUT	Вывод
PCHL	Поместить содержимое пары HL в счетчик команд
POP	Взять из стека
PUSH	Положить в стек
RAL	Циклический сдвиг влево с переносом
RAR	Циклический сдвиг вправо с переносом
RC	Возврат по условию "перенос состоялся"
RET	Возврат из подпрограммы
RLC	Циклический сдвиг влево
RM	Возврат по условию "минус"
RNC	Возврат по условию "переноса не было"
RNZ	Возврат по условию "не нуль"
RP	Возврат по условию "положительный"
RPE	Возврат по условию "контроль по чету"
RPO	Возврат по условию "контроль по нечету"
RRC	Циклический сдвиг вправо
RST	Рестарт
RZ	Возврат по условию "нуль"
SBB,SBI	Вычитание с заимствованием единицы из старшего разряда
SHLD	Запись в память непосредственная содержимого пары регистров HL
SPHL	Перезапись содержимого HL в указатель стека
STA	Запись содержимого аккумулятора
STAX	Запись косвенная содержимого аккумулятора
STC	Установка переноса
SUB,SUI	Вычитание
XCHG	Обмен между D и E, H и L
XRA,XRI	Логическое исключающее ИЛИ
XTHL	Обмен между вершиной стека и парой HL

10.4 Z80 Команды и операции в алфавитном порядке

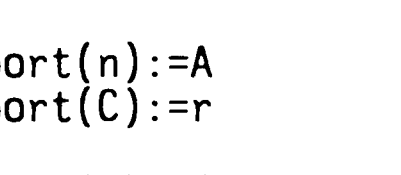
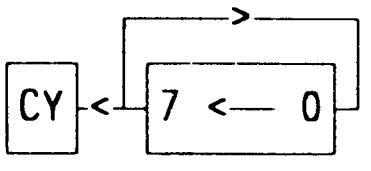
Принятые условные обозначения

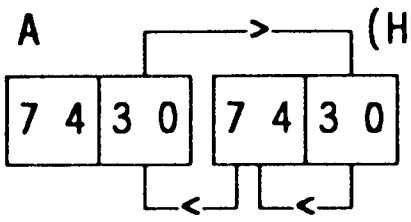
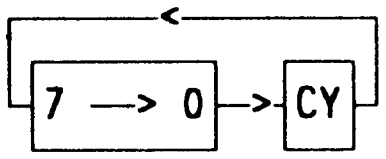
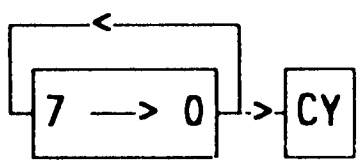
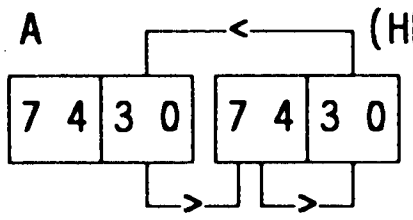
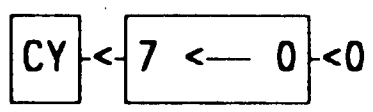
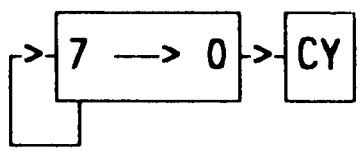
```
r    = A,B,C,D,E,H,L          n   = byte [#00..#FF]
rr   = BC,DE,HL,SP            nn  = word [#0000..#FFFF]
ii   = IX,IY
ry   = BC,DE,SP,IY            b   - bit number [0..7]
rx   = BC,DE,SP,IX            Xb  - bit b of loc. X
qq   = AF,BC,DE,HL

cc   = C,NC,Z,NZ,M,P,PE,PO    e   - relative addr.
cond = C,NC,Z,NZ              adr - byte address (0..00b)
```

Mnemonic	8080	Operation	CZPSNH	Ln	Tim	Code	Notes
ADC A,r	ADC r	A:=A+r+CY	xxVx0x	1	4	10001rrr	
ADC A,(HL)	ADC M	A:=A+(HL)+CY		1	7	8E	
ADC A,n	ACI n	A:=A+n+CY		2	7	CE nn	
ADC A,(ii+n)	-	A:=A+(ii+n)+CY		3	19	DD/FD 8E nn	
ADC HL,rr	-	HL:=HL+rr+CY	xxVx0?	2	15	ED 01rr1010	
ADD A,r	ADD r	A:=A+r	xxVx0x	1	4	10000rrr	
ADD A,(HL)	ADD M	A:=A+(HL)		1	7	86	
ADD A,n	ADI n	A:=A+n		2	7	C6 nn	
ADD A,(ii+n)	-	A:=A+(ii+n)		3	19	DD/FD 86 nn	
ADD HL,rr	DAD r*	HL:=HL+rr	x...0?	1	11	00rr1001	
ADD IX,ry	-	IX:=IX+px		2	15	DD 00rr1001	
ADD IY,rx	-	IY:=IY+py		2	15	FD 00rr1001	
AND r	ANA r	A:=A and r	0xPx01	1	4	10100rrr	
AND (HL)	ANA M	A:=A and (HL)		1	7	A6	
AND n	ANI nn	A:=A and n		2	7	E6 nn	
AND (ii+n)	-	A:=A and (ii+n)		3	19	DD/FD A6 nn	
BIT b,r	-	Z:=not r[b]	.x??01	2	8	CB 01bbbrrr	
BIT b,(HL)	-	Z:=not (HL)[b]		2	12	CB 01bbb110	
BIT b,(ii+n)	-	Z:=not (ii+n)[b]		4	20	DD/FD CB nn	01bbb11
CALL nn	CALL nn	PUSH PC; PC:=nn	.....	3	17	CD nn nn	
CALL cc,nn	Ccc nn	If cc then CALL nn		3	17	11cc100 nn	
		else continue			10	nn	
CCF	CMC	CY:=not CY	x...0?	1	4	3F	
CP r	CMP r	A-r	xxVx1x	1	4	10111rrr	
CP (HL)	CMP M	A-(HL)		1	7	BE	
CP n	CMI n	A-n		2	7	FE nn	
CP (ii+n)	-	A-(ii+n)		3	19	DD/FD BE nn	
CPD	-	A-(HL);dec HL;dec BC	.xxx1x	2	16	ED A9	1
CPDR	-	Repeat CPD	.xxx1x	2	21	ED B9	1
		until A=(HL) or BC=0			16		
CPI	-	A-(HL);inc HL;dec BC	.xxx1x	2	16	ED A1	1
CPIR	-	Repeat CPI	.xxx1x	2	21	ED B1	1
		until A=(HL) or BC=0			16		
CPL	CMA	A:=A xor 255	11	1	4	2F	
DAA	DAA	Decimal adjust Acc.	xxPx.x	1	4	27	2

Mnemonic	8080	Operation	CZPSNH	Ln	Tim	Code	Notes
DEC r	DCR r	r:=r-1	.xVx1x	1	4	00rrr101	
DEC (HL)	DCR M	(HL):=(HL)-1		1	11	35	
DEC (ii+n)	-	(ii+n):=(ii+n)-1		3	23	DD/FD 35 nn	
DEC rr	DCX r*	rr:=rr-1	.....	1	6	00rr1011	
DEC ii	-	ii:=ii-1		2	10	DD/FD 2B	
DI	DI	IFF:=0	.....	1	4	F3	
DJNZ e	-	dec B; if B 0 JR e if B=0 continue	.....	2	13 8	10 ee	
EI	EI	IFF:=1	.....	1	4	FB	
EX AF,AF'	-	AF< >AF'	.....	1	4	08	
EX DE,HL	XCHG	DE< >HL		1	4	EB	
EX (SP),HL	XTHL	(SP)< >HL		1	19	E3	
EX (SP),ii	-	(SP)< >ii		2	23	DD/FD E3	
EXX	-	BC< >BC'; DE< >DE' HL< >HL'	.....	1	4	D9	
HALT	HLT	Halt CPU (if IFF=1 => wait interrupt)	.....	1	4	76	
IM 1	-	Interrupt mode 1	.....	2	8	ED 46	?
IM 2	-	Interrupt mode 2		2	8	ED 56	
IM 3	-	Interrupt mode 3		2	8	ED 5E	
IN A,(n)	IN n	A:=port(n)	.....	2	10	DB nn	
IN r,(C)	-	r:=port(C)	.xPx0x	2	11	ED 01rrr000	
-	-	Only set flags as IN r,(C) does	.xPx0x	2	11	ED 70	
INC r	INR r	r:=r+1	.xVx0x	1	4	00rrr100	
INC (HL)	INR M	(HL):=(HL)+1		1	11	34	
INC (ii+n)	-	(ii+n):=(ii+n)+1		3	23	DD/FD 34 nn	
INC rr	INX r*	rr:=rr+1	.....	1	6	00rr0011	
INC ii	-	ii:=ii+1		2	10	DD/FD 23	
IND	-	(HL):=port(C) dec HL; dec B	.x??1?	2	15	ED AA	3
INDR	-	Repeat IND until B=0	.1??1?	2	20 15	ED BA	
INI	-	(HL):=port(C) inc HL; dec B	.x??1?	2	15	ED A2	3
INIR	-	Repeat INI until B=0	.1??1?	2	20 15	ED B2	
JP nn	JMP nn	PC:=nn	.....	3	10	C3 nn nn	
JP cc,nn	Jcc nn	If cc then JP nn	.....	3	10	11ccc010 nn nn	
JP (HL)	PCHL	PC:=HL		1	4	E9	
JP (ii)	-	PC:=ii		2	8	DD/FD E9	
JR e	-	PC:=PC+e	.....	2	12	18 ee	
JR cc,e	-	If cc then JR e		2	12	001cc000 ee	
LD r,r	MOV r,r	r:=r	.....	1	4	01rrrRRR	
LD r,(HL)	MOV r,M	r:=(HL)		1	7	01rrr110	
LD r,n	MVI r,n	r:=n		2	7	00rrr110 nn	
LD r,(ii+n)	-	r:=(ii+n)		3	19	DD/FD 01rrr 10 nn	
LD (HL),r	MOV M,r	(HL):=r		1	7	01110rrr	
LD (ii+n),r	-	(ii+n):=r		3	19	DD/FD 01110rrr nn	
LD (HL),n	MVI M,n	(HL):=n		2	10	36 nn	
LD (ii+d),n	-	(ii+d):=n		4	19	DD/FD 36 dd nn	
LD A,(BC)	LDAX B	A:=(BC)		1	7	0A	
LD A,(DE)	LDAX D	A:=(DE)		1	7	1A	
LD A,(nn)	LDA nn	A:=(nn)		3	13	3A nn nn	
LD (BC),A	STAX B	(BC):=A		1	7	02	
LD (DE),A	STAX D	(DE):=A		1	7	12	
LD (nn),A	STA nn	(nn):=A		3	13	32 nn nn	
LD rr,nn	LXI r*,nn	rr:=nn	.....	3	10	00rr0001 nn nn	
LD ii,nn	-	ii:=nn		4	14	DD/FD 21 nn nn	
LD HL,(nn)	LHLD nn	HL:=(nn)		3	16	2A nn nn	
LD rr,(nn)	-	rr:=(nn)		4	20	ED 01rr1011 nn nn	

Mnemonic	8080	Operation	CZPSNH	Ln	Tim	Code	Notes
LD ii,(nn)	-	ii:=(nn)		4	20	DD/FD 2A nn nn	
LD (nn),HL	SHLD nn	(nn):=HL		3	16	22 nn nn	
LD (nn),rr	-	(nn):=rr		4	20	ED 01rr0011 nn nn	
LD (nn),ii	-	(nn):=ii		4	20	DD/FD 22 nn nn	
LD SP,HL	SPHL	SP:=HL		1	6	F9	
LD SP,ii	-	SP:=ii		1	10	DD/FD F9	
LD I,A	-	I:=A	.....	2	9	ED 47	?
LD R,A	-	R:=A		2	9	ED 4F	
LD A,I	-	A:=I	.xxx00	2	9	ED 57	P/V:=IFF
LD A,R	-	A:=R		2	9	ED 5F	P/V:=IFF
LDD	-	(DE):=(HL) dec DE,HL,BC	..x.00	2	16	ED A8	4
LDDR	-	Repeat LDD until BC=0	..0.00	2	21 16	ED B8	
LDI	-	(DE):=(HL) inc DE,HL; dec BC	..x.00	2	16	ED A0	4
LDIR	-	Repeat LDI until BC=0	..0.00	2	21 16	ED B0	
NEG	-	A:=-A	xxVx1x	2	8	ED 44	
NOP	NOP	No operation	.....	1	4	00	
OR r	ORA r	A:=A or r	0xPx00	1	4	10101rrr	
OR (HL)	ORA M	A:=A or (HL)		1	7	AE	
OR n	ORI n	A:=A or n		2	7	EE nn	
OR (ii+n)	-	A:=A or (ii+n)		3	19	DD/FD AE nn	
OTDR	-	Repeat OUTD until B=0	?1??1?	2	20 15	ED BB	
OTIR	-	Repeat OUTI until B=0	?1??1?	2	20 15	ED B3	
OUT (n),A	OUT n	port(n):=A	.....	2	11	D3 nn	
OUT (C),r	-	port(C):=r		2	12	ED 01rrr001	
OUTD	-	port(C):=(HL) dec HL; dec B	?x??1?	2	15	ED AB	3
OUTI	-	port(C):=(HL) inc HL; dec B	?x??1?	2	15	ED A3	3
POP qq	POP q*	qq:=(SP); SP:=SP+2	.....	1	10	11qq0001	
POP ii	-	ii:=(SP); SP:=SP+2		2	14	DD/FD E1	
PUSH qq	PUSH q*	SP:=SP-2; (SP):=qq	.....	1	11	11qq0101	
PUSH ii	-	SP:=SP-2; (SP):=ii		2	15	DD/FD E3	
RES b,r	-	r[b]:=0	.....	2	8	CB 10bbbrrr	
RES b,(HL)	-	(HL)[b]:=0		2	15	CB 10bbb110	
RES b,(ii+n)	-	(ii+n)[b]:=0		4	23	DD/FD CB nn	10bbb110
RET	RET	POP PC	.....	1	10	C9	
RET cc	Rcc	If cc then RET else continue	.....	1	11 5	11ccc000	
RETI	-	Return from interr.	.....	2	14	ED 4D	?
RETN	-	Return from NMI	.....	2	14	ED 45	
RL r	-		xxPx00	2	8	CB 00010rrr	
RL (HL)	-			2	15	CB 16	
RL (ii+n)	-			4	23	DD/FD CB nn 16	
RLA	RAL		x...00	1	4	17	
RLC r	-		xxPx00	2	8	CB 00000rrr	
RLC (HL)	-			2	15	CB 06	
RLC (ii+n)	-			4	23	DD/FD CB nn 16	

Mnemonic	8080	Operation	CZPSNH	Ln	Tim	Code	Notes
RLCA	RLC		x...00	1	4	07	
RLD		A (HL) 	.xPx00	2	18	ED 6F	
RR r	-		xxPx00	2	8	CB 00001rrr	
RR (HL)	-			2	15	CB 0E	
RR (ii+n)	-			4	23	DD/FD CB nn 0E	
RRA	RAR		x...00	1	4	1F	
RRC r	-		xxPx00	2	8	CB 00011rrr	
RRC (HL)	-			2	15	CB 1E	
RRC (ii+n)	-			4	23	DD/FD CB nn 1E	
RRCA	RRC		x...00	1	4	0F	
RRD	-	A (HL) 	.xPx00	2	18	ED 67	
RST adr	RST x	CALL adr	.....	1	11	11xxx111	5
SBC A,r	SBB r	A:=A-r-CY	xxVx1x	1	4	10011rrr	
SBC A,(HL)	SBB M	A:=A-(HL)-CY		1	7	9E	
SBC A,n	SBI n	A:=A-n-CY		2	7	DE nn	
SBC A,(ii+n)	-	A:=A-(ii+n)-CY		3	19	DD/FD 9E nn	
SBC HL,rr	-	HL:=HL-rr-CY	xxVx1?	2	15	ED 01rr0010	
SCF	STC	CY:=1	1...00	1	4	37	
SET b,r	-	r[b]:=1	.....	2	8	CB 11bbbrrr	
SET b,(HL)	-	(HL)[b]:=1		2	15	CB 11bbb110	
SET b,(ii+n)	-	(ii+n)[b]:=1		4	23	DD/FD CB nn	01bbb110
SLA r	-		xxPx00	2	8	CB 00100rrr	
SLA (HL)	-			2	15	CB 26	
SLA (ii+n)	-			4	23	DD/FD CB nn 26	
SRA r	-		xxPx00	2	8	CB 00101rrr	
SRA (HL)	-			2	15	CB 2E	
SRA (ii+n)	-			4	23	DD/FD CB nn 2E	
SRL r	-		xxPx00	2	8	CB 00111rrr	
SRL (HL)	-			2	15	CB 3E	
SRL (ii+n)	-			4	23	DD/FD CB nn 3E	
SUB r	SUB r	A:=A-r	xxVx0x	1	4	10010rrr	
SUB (HL)	SUB M	A:=A-(HL)		1	7	96	
SUB n	SUI n	A:=A-n		2	7	D6 nn	
SUB (ii+n)	-	A:=A-(ii+n)		3	19	DD/FD 96 nn	
XOR r	XRA r	A:=A xor r	0xPx00	1	4	10110rrr	
XOR (HL)	XRA M	A:=A xor (HL)		1	7	B6	
XOR n	XRI n	A:=A xor n		2	7	F6 nn	
XOR (ii+n)	-	A:=A xor (ii+n)		3	19	DD/FD B6 nn	

Note 1. In CP , CPDR, CPI and CPIR command : if BC=0 then P/V:=0, els P/V:=1;
if A=(HL) then Z:=1, else Z:=0.

Note 2. DAA reduce partial result of addition or substraction to the binarycoded decimal form. This is the only command which used H and N flags.
Note 3. In INI. IND, OUTI and OUTD commands: if B=0 then Z:=1, else Z:=0.
Note 4. In LDD and LDI commands: if BC=0 then P/V:=0, else P/V:=1.
Note 5. Effective address is 00xxx000(binary).

8080 Мнемоника в транскрипции команд Z-80

ACI nn - ADC A,n	IN n - IN A,(n)	RAL - RLA
ADC r - ADC A,r	INR r - INC r	RAR - RRA
ADC M - ADC A,(HL)	INR M - INC (HL)	RC - RET C
ADD r - ADD A,r	INX r* - INC rr	RET - RET
ADD M - ADD A,(HL)	JC nn - JP C,nn	RLC - RLCA
ADI nn - ADD A,n	JM nn - JP M,nn	RM - RET M
ANA r - AND r	JMP nn - JP nn	RNC - RET NC
ANA M - AND (HL)	JNC nn - JP NC,nn	RNZ - RET NZ
ANI nn - AND n	JNZ nn - JP NZ,nn	RP - RET P
CALL nn - CALL nn	JP nn - JP P,nn	RPE - RET PE
CC nn - CALL C,nn	JPE nn - JP PE,nn	RPO - RET PO
CM nn - CALL M,nn	JPO nn - JP PO,nn	RRC - RRCA
CMA - CPL	JZ nn - JP Z,nn	RST x - RST adr
CMC - CCF	LDA nn - LD A,(nn)	RZ - RET Z
CMI n - CP n	LDAX B - LD A,(BC)	SBB r - SBC A,r
CMP r - CP r	LDAX D - LD A,(DE)	SBB M - SBC A,(HL)
CMP M - CP (HL)	LHLD nn - LD HL,(nn)	SBI n - SBC A,n
CNC nn - CALL NC,nn	LXI r*,nn - LD rr,nn	SHLD nn - LD (nn),HL
CNZ nn - CALL NZ,nn	MOV r,r - LD r,r	SPHL - LD SP,HL
CP nn - CALL P,nn	MOV r,M - LD r,(HL)	STA nn - LD (nn),A
CPE nn - CALL PE,nn	MOV M,r - LD (HL),r	STAX B - LD (BC),A
CPO nn - CALL PO,nn	MVI r,n - LD r,n	STAX D - LD (DE),A
CZ nn - CALL Z,nn	MVI M,n - LD (HL),n	STC - SCF
DAA - DAA	NOP - NOP	SUB r - SUB r
DAD r* - ADD HL,rr	ORA r - OR r	SUB M - SUB (HL)
DCR r - DEC r	ORA M - OR (HL)	SUI n - SUB n
DCR M - DEC (HL)	ORI n - OR n	XCHG - EX DE,HL
DCX r* - DEC rr	OUT n - OUT (n),A	XRA r - XOR r
DI - DI	PCHL - JP (HL)	XRA M - XOR (HL)
EI - EI	POP q* - POP qq	XRI n - XOR n
HLT - HALT	PUSH q* - PUSH qq	XTHL - EX (SP),HL

10.5 Система команд процессора Z-80-A (сводная таблица кодов)

Table #01

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00	NOP	LD BC,nn	LD (BC),A	INC BC	INC B	DEC B	LD B,n	RLCA	EX AF,AF'	ADD HL,BC	LD A,(BC)	DEC BC	INC C	DEC C	LD C,n	RRCA
10	DJNZ dis	LD DE,nn	LD (DE),A	INC DE	INC D	DEC D	LD D,n	RLA	JR dis	ADD HL,DE	LD A,(DE)	DEC DE	INC E	DEC E	LD E,n	RRA
20	JR nz,dis	LD HL,nn	LD (nn)HL	INC HL	INC H	DEC H	LD H,n	DAA	JR z,dis	ADD HL,HL	LD HL(nn)	DEC HL	INC L	DEC L	LD L,n	CPL
30	JR nc,dis	LD SP,nn	LD (nn),A	INC SP	INC (HL)	DEC (HL)	LD (HL),n	SCF	JR c,dis	ADD HL,SP	LD A,(nn)	DEC SP	INC A	DEC A	LD A,n	CCF
40	LD B,B	LD B,C	LD B,D	LD B,E	LD B,H	LD B,L	LD B,(HL)	LD B,A	LD C,B	LD C,C	LD C,D	LD C,E	LD C,H	LD C,L	LD C,(HL)	LD C,A
50	LD D,B	LD D,C	LD D,D	LD D,E	LD D,H	LD D,L	LD D,(HL)	LD D,A	LD E,B	LD E,C	LD E,D	LD E,E	LD E,H	LD E,L	LD E,(HL)	LD E,A
60	LD H,B	LD H,C	LD H,D	LD H,E	LD H,H	LD H,L	LD H,(HL)	LD H,A	LD L,B	LD L,C	LD L,D	LD L,E	LD L,H	LD L,L	LD L,(HL)	LD L,A
70	LD (HL),B	LD (HL),C	LD (HL),D	LD (HL),E	LD (HL),H	LD (HL),L	HALT	LD (HL),A	LD A,B	LD A,C	LD A,D	LD A,E	LD A,H	LD A,L	LD A,(HL)	LD A,A
80	ADD A,B	ADD A,C	ADD A,D	ADD A,E	ADD A,H	ADD A,L	ADD A,(HL)	ADD A,A	ADC A,B	ADC A,C	ADC A,D	ADC A,E	ADC A,H	ADC A,L	ADC A,(HL)	ADC A,A
90	SUB B	SUB C	SUB D	SUB E	SUB H	SUB L	SUB (HL)	SUB A	SBC A,B	SBC A,C	SBC A,D	SBC A,E	SBC A,H	SBC A,L	SBC A,(HL)	SBC A,A
A0	AND B	AND C	AND D	AND E	AND H	AND L	AND (HL)	AND A	XOR B	XOR C	XOR D	XOR E	XOR H	XOR L	XOR (HL)	XOR A
B0	OR B	OR C	OR D	OR E	OR H	OR L	OR (HL)	OR A	CP B	CP C	CP D	CP E	CP H	CP L	CP (HL)	CP A
C0	RET nz	POP BC	JP nz,nn	JP nn	CALL nz,nn	PUSH BC	ADD A,n	RST 0	RET z	RET	JP z,nn	****	CALL z,nn	CALL nn	ADC A,n	RST 8
D0	RET nc	POP DE	JP nc,nn	OUT (n),A	CALL nc,nn	PUSH DE	SUB n	RST 10	RET c	EXX	JP c,nn	IN A,(n)	CALL c,nn	****	SBC A,n	RST 18
E0	RET po	POP HL	JP po,nn	EX (SP)HL	CALL po,nn	PUSH HL	AND n	RST 20	RET pe	JP (HL)	JP pe,nn	EX DE,HL	CALL pe,nn	****	XOR n	RST 28
F0	RET p	POP AF	JP p,nn	DI	CALL p,nn	PUSH AF	OR n	RST 30	RET m	LD SP,HL	JP m,nn	EI	CALL m,nn	****	CP n	RST 38

Table #02

CB	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00	RLC B	RLC C	RLC D	RLC E	RLC H	RLC L	RLC (HL)	RLC A	RRC B	RRC C	RRC D	RRC E	RRC H	RRC L	RRC (HL)	RRC A
10	RL B	RL C	RL D	RL E	RL H	RL L	RL (HL)	RL A	RR B	RR C	RR D	RR E	RR H	RR L	RR (HL)	RR A
20	SLA B	SLA C	SLA D	SLA E	SLA H	SLA L	SLA (HL)	SLA A	SRA B	SRA C	SRA D	SRA E	SRA H	SRA L	SRA (HL)	SRA A
30									SRL B	SRL C	SRL D	SRL E	SRL H	SRL L	SRL (HL)	SRL A
40	BIT 0,B	BIT 0,C	BIT 0,D	BIT 0,E	BIT 0,H	BIT 0,L	BIT 0,(HL)	BIT 0,A	BIT 1,B	BIT 1,C	BIT 1,D	BIT 1,E	BIT 1,H	BIT 1,L	BIT 1,(HL)	BIT 1,A
50	BIT 2,B	BIT 2,C	BIT 2,D	BIT 2,E	BIT 2,H	BIT 2,L	BIT 2,(HL)	BIT 2,A	BIT 3,B	BIT 3,C	BIT 3,D	BIT 3,E	BIT 3,H	BIT 3,L	BIT 3,(HL)	BIT 3,A
60	BIT 4,B	BIT 4,C	BIT 4,D	BIT 4,E	BIT 4,H	BIT 4,L	BIT 4,(HL)	BIT 4,A	BIT 5,B	BIT 5,C	BIT 5,D	BIT 5,E	BIT 5,H	BIT 5,L	BIT 5,(HL)	BIT 5,A
70	BIT 6,B	BIT 6,C	BIT 6,D	BIT 6,E	BIT 6,H	BIT 6,L	BIT 6,(HL)	BIT 6,A	BIT 7,B	BIT 7,C	BIT 7,D	BIT 7,E	BIT 7,H	BIT 7,L	BIT 7,(HL)	BIT 7,A
80	SET 0,B	SET 0,C	SET 0,D	SET 0,E	SET 0,H	SET 0,L	SET 0,(HL)	SET 0,A	SET 1,B	SET 1,C	SET 1,D	SET 1,E	SET 1,H	SET 1,L	SET 1,(HL)	SET 1,A
90	SET 2,B	SET 2,C	SET 2,D	SET 2,E	SET 2,H	SET 2,L	SET 2,(HL)	SET 2,A	SET 3,B	SET 3,C	SET 3,D	SET 3,E	SET 3,H	SET 3,L	SET 3,(HL)	SET 3,A
A0	SET 4,B	SET 4,C	SET 4,D	SET 4,E	SET 4,H	SET 4,L	SET 4,(HL)	SET 4,A	SET 5,B	SET 5,C	SET 5,D	SET 5,E	SET 5,H	SET 5,L	SET 5,(HL)	SET 5,A
B0	SET 6,B	SET 6,C	SET 6,D	SET 6,E	SET 6,H	SET 6,L	SET 6,(HL)	SET 6,A	SET 7,B	SET 7,C	SET 7,D	SET 7,E	SET 7,H	SET 7,L	SET 7,(HL)	SET 7,A
C0	RES 0,B	RES 0,C	RES 0,D	RES 0,E	RES 0,H	RES 0,L	RES 0,(HL)	RES 0,A	RES 1,B	RES 1,C	RES 1,D	RES 1,E	RES 1,H	RES 1,L	RES 1,(HL)	RES 1,A
D0	RES 2,B	RES 2,C	RES 2,D	RES 2,E	RES 2,H	RES 2,L	RES 2,(HL)	RES 2,A	RES 3,B	RES 3,C	RES 3,D	RES 3,E	RES 3,H	RES 3,L	RES 3,(HL)	RES 3,A
E0	RES 4,B	RES 4,C	RES 4,D	RES 4,E	RES 4,H	RES 4,L	RES 4,(HL)	RES 4,A	RES 5,B	RES 5,C	RES 5,D	RES 5,E	RES 5,H	RES 5,L	RES 5,(HL)	RES 5,A
F0	RES 6,B	RES 6,C	RES 6,D	RES 6,E	RES 6,H	RES 6,L	RES 6,(HL)	RES 6,A	RES 7,B	RES 7,C	RES 7,D	RES 7,E	RES 7,H	RES 7,L	RES 7,(HL)	RES 7,A

Table #03

ED	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00																
10																
20																
30																
40	IN B,(C)	OUT (C),B	SBC HL,BC	LD (nn)BC	NEG	RETN	IM 0	LD I,A	IN C,(C)	OUT (C),C	ADC HL,BC	LD BC(nn)		RETI		LD R,A
50	IN D,(C)	OUT (C),D	SBC HL,DE	LD (nn)DE			IM 1		IN E,(C)	OUT (C),E	ADC HL,DE	LD DE(nn)			IM 2	
60	IN H,(C)	OUT (C),H	SBC HL,HL					RRD	IN L,(C)	OUT (C),L	ADC HL,HL					RLD
70			SBC HL,SP	LD (nn)SP					IN A,(C)	OUT (C),A	ADC HL,SP	LD SP(nn)				
80																
90																
A0	LDI	CPI	INI	OUTI					LDD	CPD	IND	OUTD				
B0	LDIR	CPIR	INIR	OTIR					LDDR	CPDR	INDR	OTDR				
C0																
D0																
E0																
F0																

IX -> HL при прибавлении DD IY -> HL при прибавлении FD.

11. ПРИМЕРЫ ПРОГРАММ НА АССЕМБЛЕРЕ ДЛЯ ОС IS-DOS

1. Драйвер печати EPROF.LPR

Модуль компиляции драйвера EPROF.BAT

```
;-----------------------------------
     as eprof.as
     link eprof.obj /a25000 /out eprof.lpr
;-----------------------------------
```

Модуль программы драйвера EPROF.AS(M-для старых версий AS.COM)

```
;
; ┌──────────────────────────┐
; │ eprof.lpr ( Драйвер печати )
; │       02.02.94 23:56
; │     Набран в IS-EDITe
; │ Адрес загрузки:  50000/#c350
; └──────────────────────────┘
;
;
        ORG     50000
;
;
; ┌──────────────────────────────┐
; │ Вектор символьного устройства вывода
; └──────────────────────────────┘
;
;
MEM00   EQU     $-1     ;
        DEFW    LC3CE   ;
MEM01   EQU     $-1     ;
        DEFW    LC37D   ;
MEM02   EQU     $-1     ;
        DEFW    LC3BF   ;
MEM03   EQU     $-1     ;
        DEFW    LC3BF   ;
;
        DEFB    #03     ;
        DEFW    #0000   ;
        DEFW    #0000   ;
        DEFB    #01     ;
        DEFW    #0000   ;
;
;
LC360   PUSH    BC
        PUSH    DE
        PUSH    HL
        LD      BC,#0006
        ADD     HL,BC
MEM04   EQU     $
        CALL    LC372
        POP     HL
        POP     DE
        POP     BC
        DI
        RET     NC
        POP     HL
        JR      LC3B8
;
LC372   EI
        PUSH    AF
        LD      C,#08 ;Была ли нажата клавиша ?
        RST     #10
```

```
┌──────────────────────────────────────┐
│      Адрес ПП инициализации          │
│      Адрес ПП вывода символа         │
│  Адрес ПП инверсия режима вывода     │
│  (для принтера не используется)      │
│    Адрес ПП установки координат      │
│  (для принтера не используется)      │
│     Длина курсора тек.драйвера       │
│   Текущие лог. координаты печати     │
│   Текущие физ. координаты печати     │
│ Тип устройства (0-экран/1-принтер)   │
│  Адрес программы обработки ошибок    │
└──────────────────────────────────────┘
```

```
        POP     AF
        LD      E,(HL)
        INC     HL
        LD      D,(HL)
        EX      DE,HL
        JP      (HL)
;
;
; ПП вывода символа ( A="код символа" )
;
;
LC37D   PUSH    AF
        LD      C,#12 ; Возврат вектора символьного устройства
        RST     #10
        EXX
        POP     AF
        CP      #0D
        JR      NZ,LC396  ;Переход, если не конец строки (код 13)
        BIT     0,(HL)
        JR      Z,LC3C0
MEM05   EQU     $
        CALL    LC396
        RET     C
MEM06   EQU     $
        CALL    LC3BF     ;Ret
        LD      A,#0A     ;После кода 13 вывести код перевода строки
        JR      LC396
;
LC396   LD      E,A
        LD      A,#0F
        OUT     (#3F),A
        DI
LC39C   CALL    #1F54     ;Проверка на "BREAK"
        JR      NC,MEM07  ;Если нажата
        IN      A,(#5F)
        RLA
        JR      C,LC39C   ;Ожидание сигнала готовности принтера
        LD      A,E
        OUT     (#3F),A   ;Вывести символ в порт данных
        LD      A,#0E
        OUT     (#5F),A   ;Выставить строб
LC3AD   CALL    #1F54
        JR      NC,MEM08
        IN      A,(#5F)
        RLA
        JR      NC,LC3AD  ;Ожидание готовности принтера
LC3B7   XOR     A
LC3B8   LD      E,#0F
        LD      C,#5F
        OUT     (C),E     ;Вывести строб
        EI
LC3BF   RET
;
LC3C0   LD      A,#96
        JR      LC360
;
MEM07   CALL    LC3C0
        JR      LC39C
```

```
;
MEM08   CALL    LC3C0
        JR      LC3AD
;
;
;  ПП инициализации принтера
;
;
;
LC3CE   INC     A
        RET     Z
        LD      A,#88
        OUT     (#7F),A
        JR      LC3B7
;
;Область адресов настройки
;
        DEFW    #FFFF ;Маркер начала области
;
        DEFW    MEM00,MEM01,MEM02,MEM03
        DEFW    MEM04,MEM05,MEM06,MEM07
        DEFW    MEM08
```

2. Программа изменения размера кэша

Модуль компиляции программы CACHE.BAT

```
;-------------------------------------
     as cache.as
     link cache.obj /a24000
;-------------------------------------
```

Модуль программы CACHE.AS(M-для старых версий AS.COM)

```
;      МОДУЛЬ CACHE.AS, набран в IS-EDITe
;      cache.com - программа,  позволяющая  изменять
;размер кэш диска в IS'DOS.
;    Текст программы cache.as также служит  примером
;работы с командной строкой.
;    Компонуется программа из  cache.obj  и  таблицы
;системных рестартов rst.gtb.
;    Возмжен запуск  компоновки  по  нажатию  ENTER,
;выбрав  курсором cache.obj.  В этом случае в файле
;extent.txt должна быть строка:
;         as :as /old/sym S:AS\rst /auto  ,
;
;  предполагается, что as.com находится в каталоге,
;определенном для поиска по умолчанию,  а rst.gtb в
;каталоге S:AS\
;      Она разбирается с командной строкой,  снимает
;ключи и , встретив ключ с, считывает значение чис-
;ла после него в регистр  A  и  вызывает  системный
;рестарт  creat  для  изменения  размера кэш памяти
;системы.
;      Программа достаточно "умная", поэтому не поз-
;волит задать размер кэш  диска  меньше  6  блоков.
;Программа также служит примером работы с командной
;строкой
;    Формат запуска, например:
;
;                    cache /c10
;
;После запуска  можно  посмотреть  размер доступной
;пользователю памяти с  помощью  show.com,  которая
;выдает адрес начала кэш памяти. Это и есть вершина
;для размещения программ пользователя.
;    Системная переменная  utop  ограничивает макси-
;мальный размер задания кэш памяти.  Если он превы-
;шает допустимый,  то ключ будет считаться недопус-
;тимым и произойдет выход с ошибкой 23.
;
         ORG      23904
         CALL     KLINI
         RET      C
         XOR      A
         LD       A,#F4
         RET
;       keyan.as
KLINI
         LD       C,$opcat
         RST      16
         RET      C
```

```
        RET     NZ
        EXX
        CALL    KEYAN
        RET     C
        JR      KLINI
KEYAN   LD      A,"/"
        CP      (HL)
        JR      NZ,NAME
WKO     INC     HL
        LD      B,(HL)
        PUSH    BC
        INC     HL
        PUSH    HL
        LD      B,0
WK1     LD      A," "
        CP      (HL)
        JR      Z,WK2
        LD      A,"/"
        CP      (HL)
        JR      Z,WK2
        LD      A,13
        CP      (HL)
        JR      Z,WK2
        INC     HL
        INC     B
        JR      WK1
WK2     POP     DE          ;-pointer to begining of number ASCII  buffer.
        PUSH    HL          ;pointer of  the new element -1.
        INC     B
        DEC     B
        JR      Z,WK3       ;no numeric symbols.
        EX      DE,HL       ;HL-begining of ASCII buffer.
        LD      A,B         ;number of elements in buffer.
        LD      B,10        ;base of  the number by ignor.
        LD      C,$a_d
        RST     16
        JR      C,ERR
        EXX
        EX      DE,HL
WK3     POP     HL
        POP     AF
        PUSH    HL
        LD      HL,KLANL
        RES     5,A
        LD      C,$analy
        RST     16
        LD      A,23
        POP     HL
        RET     C
        LD      A,"/"
        CP      (HL)
        JR      Z,WKO       ;key "/" followed without the " " before it.
        XOR     A
        RET     Z
ERR
        POP     HL
        POP     HL
        RET
```

```
**************************
KLANL
            DEFB    "C"
            DEFW    CACHE
            DEFB    255
            DEFW    EMPTY
**************************
CACHE
            LD      A,E
            CP      6
            JR      NC,CC1
            LD      A,6
CC1
            LD      C,$creat
            RST     16
            RET     C
            XOR     A
            RET
EMPTY
            SCF
            RET
**************************
```

ФАЙЛ, СОДЕРЖАЩИЙ СПИСОК СИСТЕМНЫХ РЕСТАРТОВ

```
;     МОДУЛЬ RST.AS, набран в IS-EDITe
; Если метка или имя процедуры начинается с символа '$', она
; считается глобальной и видна во всех вызывающих программах.
; (имеет смысл, когда программа компонуется из разных модулей .OBJ)
; Локальные метки видны в каждом .OBJ свои, даже если в 2-х
; отдельных они будут одинаковы.
;
$creat      EQU  0          ; Управление устройствами
$clear      EQU  1
$flush      EQU  2
$ttyin      EQU  7

$kwait      EQU  8
$ktest      EQU  9
$ttyou      EQU 10
$prcpl      EQU 11
$prapd      EQU 12
$binit      EQU 15

$g_cnf      EQU 16
$g_tyd      EQU 18
$stchn      EQU 22
$delch      EQU 23

$newch      EQU 25
$swtyp      EQU 27
$ldnew      EQU 29
$l_im2      EQU 30
$erdrv      EQU 31

;-----------------------

$open       EQU 32          ; Файловая служба
$open1      EQU 33
$mkdir      EQU 34
$crfil      EQU 35
$erfil      EQU 36
$fopen      EQU 37
$opnum      EQU 38
$gname      EQU 39

$putf       EQU 40
$rpart      EQU 41
$wpart      EQU 42
$rifle      EQU 43
$wifle      EQU 44
$fadd       EQU 47

$fcut       EQU 48
$eadd       EQU 49
$ecut       EQU 50
$g_cat      EQU 51
$find       EQU 52
$bkfcb      EQU 53
$p_sta      EQU 54
$g_sta      EQU 55
```

```
$distr      EQU 56
$retrn      EQU 57
$paste      EQU 58
$crf__      EQU 59
$erf__      EQU 60
$rvblk      EQU 61
$wvblk      EQU 62
$free       EQU 63

;-------------------------

$oparm      EQU 64          ; Работа с командной строкой
$fmrst      EQU 65
$comst      EQU 66
$opcat      EQU 67
$exeba      EQU 68
$g_com      EQU 69
$trans      EQU 70
$g_way      EQU 71

$run        EQU 72
$p_com      EQU 73
$dvtrn      EQU 74
$newco      EQU 75
$newca      EQU 76
$rcdel      EQU 77
$cp_ir      EQU 78
$convr      EQU 79

$fncor      EQU 80
$fndev      EQU 81
$opres      EQU 82
$exere      EQU 83
$swrun      EQU 84

;-------------------------

;$empt0      EQU  96        ; Оконная служба
$wt         EQU  97
$box        EQU  98
$awt        EQU  99
$awtc       EQU 100
$lwt        EQU 101
$adrwt      EQU 102
$lenwt      EQU 103

$prstr      EQU 104
$iniwt      EQU 105
$panel      EQU 106
$wtpos      EQU 107
$str        EQU 108
$lnstr      EQU 109
$smbgt      EQU 110
$beep       EQU 111

$scrol      EQU 112
$tylin      EQU 113
```

```
$g_scr          EQU 114
$cls            EQU 115
;$empt1          EQU 116
;$empt2          EQU 117
$y___           EQU 118
$n___           EQU 119

$cr^on          EQU 120
$cr^of          EQU 121
;$empt3          EQU 122
;$empt3          EQU 123
$d_a            EQU 124
$a_d            EQU 125
$analy          EQU 126
$edstr          EQU 127

;-------------------------------

$shel0          EQU 128         ; Оболочка (SHELL)
$shel1          EQU 129
$shel2          EQU 130
$pnvie          EQU 131
$shout          EQU 132
$p^csr          EQU 133
$g_csr          EQU 134
$g_mpa          EQU 135

$ed_di          EQU 136
$mmenu          EQU 137
$g_cur          EQU 138
$mwait          EQU 139
$g_dev          EQU 140
;$em0            EQU 141
$shsub          EQU 142
$cpfil          EQU 143
$shpan          EQU 144

$menu           EQU 145
$funct          EQU 146
;-------------------------------
```

12. ПЕРЕЧЕНЬ КЛЮЧЕЙ AS.COM и LINK.COM Version 1.1

12.1 КЛЮЧИ АССЕМБЛЕРА AS.COM

Ключи ассемблера предназначены для задания режимов трансляции и режимов вывода листинга на все время трансляции файла, и применяются, когда требуется изменить режим трансляции по умолчанию или отменить действие определенных команд управления листингом и директив ассемблера.

Они пишутся либо перед, либо после имени транслируемого файла, при этом большие и маленькие буквы не различаются.

Для лучшей читаемости *.bat файлов все ключи допускают расширение имени, один из возможных вариантов которого приведен в квадратных скобках. В расширение может входить любой символ, кроме "/",13," " и цифр (цифры нельзя включать в расширение, если ключу требуется числовой аргумент). Проверяются только первые 3 знака ключа.

Числа, используемые в качестве аргументов ключей, по умолчанию задаются в десятичной системе счисления, но можно и определить число как шестнадцатиричное или двоичное, начав его со знаков # или %. Числа пишутся сразу после ключа без пробелов.

/obj[ect]число
- задает размер объектного файла в блоках равным введенному сразу после /obj числу.

/obj[ect] [путь][имя_файла[.расш-ие]]
- задает путь и имя объектного файла. Пробел после символа ключа обязателен.

/sym[bols]число
- задает размер файла-таблицы локальных символов (локальной части UST) в блоках равным введенному сразу после /sym числу.

/sym[bols] [путь][имя_файла[.расш-ие]]
- задает путь и имя файлатаблицы лок.символов пользователя. Бывает полезен для размещения ее на быстром устройстве (например, на электронном диске).

/.ke[y]число
- помещает в таблицу локальных символов пользователя символ .KEY и присваивает ему значение числа, введенного сразу после имени ключа. Число может быть от 0 до 65535.

  Этот ключ позволяет управлять генерацией кода трансляции прямо из командной строки или из командных файлов, если в текст программы включать символ .KEY в аргументы директив .IF.

  Возможно независимое управление до 16-ю блоками условной трансляции или генерация одного из 65536 вариантов программы (т.е. управление 65536-ю взаимоисключающими блоками).

  Также возможна комбинация независимого и взаимоисключающего управления.

  Например, /.key%101 включит трансляцию для 1-ого и 3-его блоков условной трансляции в следующем фрагменте программы:

```
IF .KEY&%1
  блок 1
   END

IF .KEY&%10
  блок 2
   END

IF .KEY&%100
  блок 3
   END
```

/-rus
- запрещает использование русских букв в символах, определяемых пользователем (в метках, константах и макрокомандах). При этом встреча русской буквы в этих символах будет вызывать появление

ошибки *Error*22.

/+rus

- разрешает использование русских букв в символах, определяемых пользователем (в метках, константах и макрокомандах).

/-ob[j]

- приводит к трансляции без генерации объектного модуля *.obj.

/-ob[j]число

- действует аналогично предыдущему ключу, но одновременно определяет адрес трансляции модуля равным числу (0-65535), стоящему сразу после ключа.

Последние два ключа применяются для отладочной трансляции и для генерации настроенного на заданный адрес листинга трансляции модуля.

Далее идет описание ключей, управляющих выводом листинга трансляции:

/lp[(на_принтер)]

- переключает вывод листинга трансляции на принтер.

/-st[op]

- выключает отработку команд останова *S(пауз) во время выдачи листинга.

/-er[ror]

- выключает отработку остановки при возникновении ошибки трансляции.

/scr[oll]

- устанавливает поэкранный режим вывода листинга на экран (когда после каждого нового заполнения окна листинга появляется вопрос SCROLL? и ожидается нажатие любой клавиши для продолжения скроллинга).

/cut

- устанавливает режим, когда ассемблер обрезает строку листинга, если она не помещается по длине окна.

/+l[ist]

- заставляет выводить листинг, игнорируя команды *L- .

/-l[ist]

- подавляет вывод листинга; при этом команды *L+ игнорируются.

/+if

- разрешает выдачу строк блоков условной трансляции, даже если трансляция была выключена одной из директив условной трансляции.

/-if

- исключает из листинга строки блоков условной трансляции, если трансляция была выключена.

/+a[dress]

- включает в формат вывода строки листинга адрес соответствующей этой строке команды Z80 или блока данных (текущее значение программного счетчика).

/-a[dress]

- исключает из формата вывода строки текущее значение программного счетчика.

/+c[ode_list]

- включает в формат вывода строки поле данных (код оттранслированной команды Z80 или блок данных). Длина этого поля может быть изменена директивой управления листингом .CODE выражение по ходу трансляции, и ключом /codчисло на все время трансляции.

/-c[ode_list]
- исключает из формата вывода строки поле данных.

/+n[umber]
- включает в формат вывода строки ее номер во входном текстовом файле.

/-n[umber]
- исключает номер из формата вывода строки.

/+t[ext]
- включает в формат вывода строки текст из входного файла.

/-t[ext]
- исключает текст из формата вывода строки.

/cod[e]число
- устанавливает длину поля данных (кода) строки листинга на все время трансляции равным значению "число".

/+h[ex]
- приводит к установке 16-ричного формата вывода адреса инструкций или блоков данных.

/+d[ec]
- приводит к установке 10-ричного формата вывода адреса инструкции или блока данных.

/+o[ct]
- приводит к установке 8-ричного формата вывода адреса инструкции или блока данных.

Режимы "по умолчанию".

Если начать трансляцию без указания ключей и имени выходного файла, то она будет проходить в режиме "по умолчанию", описанном в таблице.

Таблица режима трансляции "по умолчанию"

Путь к выходному файлу	Текущий каталог
Имя выходного файла	Имя входного файла *.as
Расширение имени выходного файла	.obj
Генерация объектного файла	включена
Длина выходного файла при его создании	Половина от длины входного (но не меньше 1б.)
Путь к файлу-таблице	Текущий каталог
Имя файла-таблицы	Имя входного файла *.as
Расширение имени файла-таблицы	loc
Длина файла-таблицы при его создании	Четверть от длины входного (но не меньше 1б.)
Адрес трансляции	Равен 0
Выдача листинга	выключена
Вывод листинга	На экран
Относит. адрес лист.	Шестнадцатиричный
Наличие полей листинга	все
Длина поля кода в листинге	12 знаков
Остановка по каждой ошибке	Нет
Остановка по заполнению экрана сообщениями об ошибках	Да
Остановка по встрече *S	Да

12.2 КЛЮЧИ КОМПОНОВЩИКА LINK.COM

Ключи компоновщика (может называться линковщиком) предназначены для изменения режимов компоновки "по умолчанию". Ключи пишутся как перед, так и после имен компонуемых модулей, при этом большие и маленькие буквы не различаются.

Для лучшей читаемости *.bat файлов все ключи допускают расширение имени, один из возможных вариантов которого приведен в квадратных скобках. В расширение может входить любой символ, кроме "/",13," " и цифр (цифры нельзя включать в расширение, если ключу требуется числовой аргумент). Проверяются только первые 3 знака ключа.

Числа, используемые в качестве аргументов ключей, по умолчанию задаются в десятичной системе счисления, но можно и определить число как шестнадцатиричное или двоичное, начав его со знаков # или %. Числа пишутся сразу после ключа без пробелов.

/ачисло

- задает адрес компоновки для собираемой из модулей .obj программы (по умолчанию адрес берется из описателя первого в командной строке входного файла *.obj).

  Этот ключ позволяет указать адрес компоновки для произвольного модуля задачи или для ее вершины (адрес первого свободного байта после задачи).

  Для этого требуется поместить ключ /ачисло перед именем модуля, адрес которого нужно указать. Если ключ стоит после имен всех входных файлов, то он задает адрес вершины задачи.

/a [путь]имя_файла.расширение

- этот ключ действует также, как и ключ /ачисло, но значение адреса берет из описателя файла, имя и путь к которому указаны после ключа /a и пробела. Он позволяет автоматически в *.bat файлах генерировать загрузочные файлы для задачи, состоящей из разных модулей. Например, он использовался при генерации уровней IS'DOS (dos.sys, dud.sys ...), которые хоть и состоят из различных файлов, но адреса их загрузки зависят от длин друг друга. Имя файла и его расширение здесь указывать нужно обязательно. В противном случае вместо компоновки Вы выйдете в систему с ошибкой 31, или не будет найден файл с расширением из 3-х пробелов.

/symчисло

- задает размер файла-таблицы глобальных символов в блоках, для поддержания непрерывного типа файла в процессе трансляции (размер по умолчанию - 5 блоков).

/sym[bols] [путь][имя_файла[.расш-ие]]

- задает путь и имя файла-таблицы глобальных символов компонуемой задачи, которая строится и используется компоновщиком. Бывает полезен для размещения ее на быстром устройстве (например на электронном диске), в случае дальнейшего использования этой таблицы, а также, когда используется "старая" таблица.

/out [путь][имя_файла[.расширение]]

- задает путь и имя с расширением компонуемой загрузочной программы. Пробел после символа ключа обязателен.

/-de[l]

- запрещает стирать таблицу глобальных символов компонуемой задачи. Имя ее файла и путь можно задать ключом /out.

/old

- заставляет компоновщик использовать "старую" таблицу глобальных символов, полученную при компоновке одной из ранних задач. Путь к ее файлу и имя задаются ключом /sym .

/res[число]

- приводит к генерации выходного файла типа *.res и позволяет задавать его размер в блоках для поддержания непрерывного типа выходного файла *.res во время трансляции. (размер по умолчанию равен длине соответствующего *.com файла + 4 блока).

Ключи управления листингом ошибок.

/-st[op]

- запрещает вход в режим ожидания после заполнения экрана сообщениями об ошибках. Используется, если планируются только сообщения о допустимых ошибках для компонуемой задачи

/lp

- определяет вывод листинга ошибок на принтер.

/+o

- приводит к выводу относительного адреса настраиваемой ячейки в модуле, при возникновении ошибки, в восьмеричной системе исчисления.

/+d - ---"---десятичной---"---.

Режимы "по умолчанию".

Если начать компоновку без указания ключей и имени выходного файла, то она будет проходить в режиме "по умолчанию", описанном в таблице.

Таблица режима компоновки "по умолчанию"

Путь к выходному файлу	Текущий каталог
Имя выходного файла	Имя первого входного файла
Таблица символов	Новая
Путь к файлу-таблице	Текущий каталог
Имя файла-таблицы	Имя первого входного файла *.obj
Стирать файл-таблицу	Да
Адрес компоновки	Из описателя первого входного об. файла (адр.загр.)
Тип задачи	*.com
Вывод листинга	На экран
Относит. адрес лист.	Шестнадцатиричный
Остановка по ошибке	Да

Составитель: Криштопа Сергей Владимирович
Корректор: Самыловский Сергей Владимирович

Подготовлено к печати: ИИКЦ "Эльф-М"
Ул. Маркса-Энгельса, 8

Мытищинская межрайонная типография

Зак. 4043 Тир. 5000

Фирма *"СЛОТ"* (Москва) предлагает операционную систему IS-DOS, общесистемные и прикладные программы к ней на отдельных дискетах и в комплекте:

---

1 (БАЗОВЫЙ КОМПЛЕКТ) - Содержит ПОЛНЫЙ набор сервисных программ, утилит, драйверов и дополнительных средств:
- Инсталлятор системы;
- Утилиты в полном составе;
- Работа с TR-DOS дисками (запись, чтение файлов);
- Работа с MS-DOS дисками (форматирование на 720, чтение, запись файлов, создание каталогов, удаление файлов, переименование);
- HELPER - автоматизированная система обучения пользователя;
- Упаковщики файлов;
- Турбо-редактор текстов, печать текстов и полный набор драйверов принтеров;
- Шифровщик текстовых файлов SCRYPT-кодом;
- Перекодировщик файлов TRAN с полным набором таблиц перекодировки;
- Драйверы клавиатуры-48 клавиш (ЯВЕРТ, ЙЦУКЕН, ЦУКЕН), RAM-дисков на 128, 256, 512, 1024 кБ (3 последних для PROFI), 256 для SCORPIO;
- Расширенную систему USER MENU;
- Пара игр, адаптированных под iS-DOS;
- Архиваторы текстовых файлов.

23 (LUX-PRINT&e.t.c.) - Содержит НЕПОЛНЫЙ набор IS-DOS, а также пакеты:
- LUX-PRINT - программа печати экранных TR-DOS файлов. Масштабирование, режим лупы, инверсия, оттенки, размещение на листе;
- RECLAMSHOW - программа создания рекламных роликов (в редакторе) и их проигрывания;
- RESIDENT - резидентное музыкальное сопровождение на базе AY-процессора. MUSIC

4 (ЗАПИСНАЯ КНИЖКА) - Содержит НЕПОЛНЫЙ набор IS-DOS, а также пакет "ЗАПИСНАЯ КНИЖКА" :
- Назначение: Ведение по алфавиту ФИО адресов,телефонов, ...; Ведение календаря; Вывод на принтер содержимого ЗК.

5 (ДЕЛОВОЙ КАЛЕНДАРЬ)- Содержит НЕПОЛНЫЙ набор IS-DOS, а также пакет "ДЕЛОВОЙ КАЛЕНДАРЬ" :
- Назначение: Ведение записей по числам, составление долгосрочных планов, распечатывание на принтере.

6 (АРМ "ФИНАНСЫ") - Содержит НЕПОЛНЫЙ набор IS-DOS, пакет "ФИНАНСЫ":
- Назначение:Ведение бухгалтерии (счета, проводки, перерасчет, календарь, операции, отчеты, оборотный баланс, анализ;
- Вых.Докум.:Дерево счетов, дерево проводок, отчет по операциям, налоги, баланс предпр-я.

7 (АРМ "СКЛАД") - Содержит НЕПОЛНЫЙ набор IS-DOS, пакет "СКЛАД" :
- Назначение: Приход, расход товаров;
- Вых.Докум.: Общая справка, составление счет-фактуры на выдачу товара.

8 (АРМ "МАТЦЕННОСТИ")- Содержит НЕПОЛНЫЙ набор IS-DOS, пакет "МАТЦЕННОСТИ":
- Назначение: Учет, списание МЦ. Ориентировано на учет выдачи МЦ во ВРЕМЕННОЕ пользование. Для работы с внешними орг. см."МАТРЕСУРСЫ";
- Вых.Докум.: Акт взятия на учет, Акт выдачи МЦ.

9 (АРМ "МАГАЗИН") - Содержит НЕПОЛНЫЙ набор IS-DOS, пакет "МАГАЗИН":
- Назначение: Приход, расход, списание, касса, продажа за НАЛ., БНАЛ., перевод, возврат;
- Вых.Докум.: В виде таблиц-рабочих тетрадей.

10 (КАРТОТЕКА) - Содержит НЕПОЛНЫЙ набор IS-DOS, универсальную базу данных "КАРТОТЕКА":
- Назначение : Картотечный учет;
- Вых.Докум. : Картотека в виде таблицы.
- Ограничения: Не более 200 картотек и 200 записей в каждой.

11 ("FORTH") - Система программирования FORTH-93 для IS-DOS.

12 ("ПЛАТ.ПОРУЧЕНИЯ")- Содержит НЕПОЛНЫЙ набор IS-DOS, пакет "ПЛАТЕЖНЫЕ ПОРУЧЕНИЯ":
- Назначение: Учет и составление плат. поручений;
- Вых.Докум.: Заполненные плат. поручения.

13 (ZXMODEM) - Модемная связь для 2 типов ZX-модемов (CDOS и TYPE-in/out) на 600, 900 и 1200 бод.

14 (ЭЛЕКТРОННЫЕ ТАБ.)- Содержит НЕПОЛНЫЙ набор IS-DOS, пакет "ЭЛЕКТРОННЫЕ ТАБЛИЦЫ":
Практически полностью эмулирует SUPERCALC. Предназначен для создания и работы со своими БД.
- Ограничения: Не более 200 записей в одной таблице.

15 (АРМ "МАТРЕСУРСЫ")- Содержит НЕПОЛНЫЙ набор IS-DOS, пакет "МАТРЕСУРСЫ":
- Назначение: Сочетает в себе "СКЛАД" и "МАТЦЕННОСТИ". Предназначен для учета, списания МР из одной организации в другую;
- Вых.Докум.: Автоматическое подведение итогов, составление счет-фактур, накладных, актов о приеме и передаче МР.

16 (ГРАФИЧЕСКИЕ ШРИФТЫ "GFX") - Набор графических шрифтов и редактор шрифтов. Для печати текстов в графическом режиме. Можно использовать в качестве русификатора для принтеров без русской прошивки.

17 (IS-ASSEMBLER) - Система программирования "IS-DOS АССЕМБЛЕР" с полным описанием системы прерываний IS-DOS, примерами программ и т.д.

18 (A/P СЛОВАРЬ) - Англо-русский словарь на 25000 слов. 2 дискеты, но можно работать и на 1-м дисководе. Быстрый поиск, возможность удаления, добавления, изменения и выгрузки/загрузки из текстовых файлов

****

ПРИМЕЧАНИЕ: понятие НЕПОЛНЫЙ набор IS-DOS включает в себя:
- Загружаемую систему;
- Утилиты, необходимые для работы системы;
- Текстовый редактор;
- Программа печати с набором драйверов;
- Драйвер RAM-диска на 128 кБ.

ПО ВОПРОСАМ ПРИОБРЕТЕНИЯ КНИГ И ДИСКЕТ ОБРАЩАТЬСЯ В ФИРМУ "СЛОТ" -
официальный дистрибьютер фирмы "ИСКРАСОФТ" (С-Пб.)
Телефон фирмы "СЛОТ": (095) 143-11-91