СООБЩЕНИЯ
ОБЪЕДИНЕННОГО
ИНСТИТУТА
ЯДЕРНЫХ
ИССЛЕДОВАНИЙ
Дубна

11-6005

ЛАБОРАТОРИЯ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ
И АВТОМАТИЗАЦИИ

В.А. Загинайко

СИСТЕМА МАТЕМАТИЧЕСКОГО ОБЕСПЕЧЕНИЯ БЭСМ-4

1971

11-6005

В.А. Загинайко

СИСТЕМА МАТЕМАТИЧЕСКОГО ОБЕСПЕЧЕНИЯ БЭСМ-4

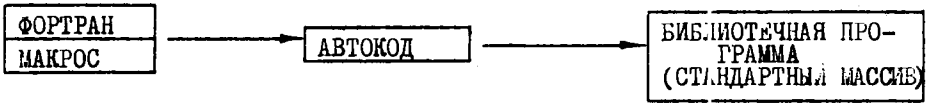
Система математического обеспечения ЭВМ БЭСМ-4, разработанная в ЛВТА ОИЯИ, представляет собой компилирующую систему, которая из библиотечных программ (стандартных массивов) составляет и настраивает рабочую программу, пригодную к счету. /1/

Система состоит из транслятора с автокода в библиотечную программу (СМ) и загрузчика и может иметь в своем составе трансляторы с алгоритмических языков в автокод.

В действующем варианте системы имеется транслятор с инвариантного языка Макрос в автокод (разрабатывается транслятор с Фортрана в автокод).

В постоянную библиотеку математического обеспечения ЭВМ БЭСМ-4 можно автоматически включить любую программу, полученную в результате трансляции (с языка Фортран, с языка Макрос, с автокода) в рамках этой системы.

Общий принцип трансляции в системе:



Задача пользователя транслируется в автокодный текст, затем транслируется в библиотечную программу (СМ).

Задача пользователя может состоять из частей всех трех уровней, т.е. текст на алгоритмическом языке (Фортран и Макрос), текст в автокоде, библиотечные программы (СМ) на перфокартах - результаты предыдущей независимой трансляции.

Система не запрещает использование ранее разработанных систем математического обеспечения типа ИС-2, ИС-22 и т.д.

Система обеспечивает прохождение "больших задач" пользователя (суммарная длина подпрограмм задачи и блоков информации больше ОЗУ) введением на уровне загрузки сегментации (наложения) разделов.

Библиотечная программа (стандартный массив) может работать только в рамках системы.

I. Краткое описание функций основных частей системы

I. Транслятор с Макроса транслирует части задачи пользователя, написанные на языке Макрос в автокод. Части задачи пользователя, написанные в автокоде, переписываются без изменений. Автокодный текст задачи записывается на МБ.

2. Транслятор с автокода транслирует весь автокодный текст задачи; организует каталог и временную библиотеку программ (СМ) на МБ и записывает (режим "запись СМ") программы, полученные в результате трансляции с автокода.

3. Загрузчик вводит в ОЗУ с перфокарт независимо транслированные библиотечные программы (СМ), к которым обращается во время счета программа пользователя, записывает их в каталог и библиотеку программ на МБ (режим "**дозапись** СМ")

затем по информации пользователя (карта "сегментация разделов") начинает настраивать части задачи-разделы - по месту работы в ОЗУ и организует каталог и библиотеку разделов на МБ.

Резидент - часть загрузчика, постоянно находящаяся в ОЗУ ЭВМ при счете задачи пользователя, организует динамическую перепись разделов в ОЗУ из библиотеки разделов на МБ и передает управление с возвратом в резидент на головную программу раздела.

Система при прохождении задачи пользователя выдает (по требованию) следующую информацию:

Печать на АЦПУ задачи пользователя на входном языке.

Печать на АЦПУ автокодного текста задачи.

Печать на АЦПУ таблицы распределения памяти (в относительных адресах).

Печать на АЦПУ информации о загрузке разделов.

Перфорацию и печать библиотечных программ (СМ) - результат трансляции автокодного текста задачи.

Прохождение задачи пользователя организуется с помощью системных (управляющих) карт

*УРР – код пробивки перфокарт – УПП
*TLT – код пробивки перфокарт – телетайп
*MAC – язык текста МАКРОС
*ASS – язык текста АВТОКОД
*END – конец пакета (текста программы)
*FOR – язык текста ФОРТРАН.

2. Описание автокода системы

2.1. Общие сведения о трансляции с автокода

Транслятор с автокода – двухпроходный ассемблер, аналогичный трансляторам[2,3]. Результатом трансляции с автокода является библиотечная программа (стандартный массив).

Стандартный массив состоит из трех основных частей:

Характеристика длин групп кодов СМ.
Рабочая программа.
Информация для загрузчика.

Рабочая программа, это, собственно, та часть стандартного массива, по которой после настройки осуществляется счет; распределение памяти с ячейки 0020 содержит некоторые служебные адреса – ссылки на третью часть стандартного массива. Информация для загрузчика, по которой происходит настройка рабочей программы для счета, состоит из:

Таблицы характеристик строк рабочей программы (команда, константа, *bss*).
Таблицы описаний (сведения о ссылках на внешние объекты, ненастраиваемые адреса и т.д.).
Таблицы внешних идентификаторов.
Указания общей длины внутренних *COMMON* .
Таблицы входов типа *ENTRY* .

Программой в "истинных адресах" стандартный массив становится только на этапе составления и настройки разделов загрузчиком. Тогда же и реализуются требования декларативных операторов автокода:

BLOCK , *COMMON* , *SET* , *DATA* , *ENTRY* , *SUBR* и декларативную часть оператора *CALL* .

Транслятор с автокода при распределении памяти с ячейки 0020 для каждой библиотечной программы (относительные адреса) реализует требования декларативных операторов автокода *BSS* , *EQU* , *REAL* , TEXT и программирует выполняемую часть оператора *CALL* .

Порядок распределения памяти ОЗУ следующий:

1. Для массивов *bss* и команд программы (в порядке их следования).
2. Для неописанных идентификаторов (для тех идентификаторов, которые встречаются только в адресах команд, но не в виде *METOK*) в том порядке, в котором они первый раз встречаются в программе.

Транслятор с автокода реализует декларативные операторы автокода *IDENT* и *END* , транслируя каждую программу, ограниченную этими операторами, в отдельную библиотечную программу.

На АЦПУ печатается для каждой библиотечной программы таблица распределения памяти (с ячейки 0020).

Массивы *bss* , зарезервированные внутри программы, заполняются нулями; состояние ячеек внешних массивов *bss* (массивов вне программы, впереди или после нее) не определено.

При перфорации или печати библиотечной программы внешние массивы *bss* не выдаются.

Библиотечная программа (СМ) не должна превышать 2000_8 кодов; внешние массивы *bss* при этом не учитываются.

2.2. Описание элементов языка и операторов автокода

Автокод системы является расширенным вариантом автокода "Ассемблер"*), который был разработан ранее в ЛВТА ОИЯИ (без арифметического блока) /2/.

Элементы языка автокода:

Символ - латинская буква, цифра, знак раздела или арифметической операции.

*) В.А.Загинайко, И.Н.Силин "Ассемблер" БI-II-4514.

Идентификатор - последовательность букв и цифр, начинающаяся с буквы (не более 6 символов); условный адрес переменной.

Метка - идентификатор, написанный слева от команды (константы, оператора) и отделяющаяся от нее двоеточием (символ :); условный адрес команды, константы, оператора и массива.

Оператор - заранее обусловленная последовательность идентификаторов и знаков раздела; групповая команда или декларация.

Команда - последовательность цифр (код операции), идентификаторов (адреса) текста-комментария (четвертый адрес, не транслируется) и знаков раздела.

Массив - последовательность ячеек памяти, зарезервированная для информации, поступающей в ячейки при счете.

Пример I. Идентификаторы

```
MARS
SUM
SUMMAS
FILTR
F1
```

Операторы

```
CALL , FILTR ;
FILTR : IDENT ;
    A: COMMON, 1 ;
```

Команды

```
  005, a , b , ab ;
  305, a , b , ab , a*b ;
    5, a , b , ab , a*b ;
MULT :5, a , b , ab , a*b;
```

Код операции в команде записывается в том виде, как принято в системе команд машины (05 - умножение).

a, b, ab - идентификаторы

a*b - текст комментарий (не транслируется)

MULT, FILTR, A - метки.

Метка отделяется от команды (оператора, константы) двоеточием. Код операции, адреса-идентификаторы и комментарий разделяются запятыми (символ ,).

Точка с запятой (символ ;) ставится в конце команды, оператора и константы.
Команды и операторы могут быть помечены (иметь метку).
На помеченную команду (оператор) можно ссылаться в других командах (операторах).

Пример 2. 56. 0, *MULT* , 0, ——→ *MULT* ;
56. , *MULT* ;

Если адрес и код операции начинаются с нулей или содержат только нули, выписывать их необязательно; при отсутствии информации в последующих адресах и комментариях все знаки разделов выписывать необязательно.

Пример 3. 52;
72, , *NUM1* ;
, , , *N* ;

Адресом в команде может служить сумма (разность) двух или более идентификаторов или идентификатора и восьмеричного числа (чисел). В таком случае адресом после трансляции будет результат сложения (и, или вычитания) по модулю 2^{12} адресов соответствующих слагаемых.

Пример 4. *SUM* +I
x -2
x - *SUM* -I

В частности, адрес -I равен 7777.
Для удобства программирования введен адрес *f* ; он означает адрес ячейки, в которой окажется транслированная команда, содержащая этот символ. Этот символ очень удобно использовать в "адресной арифметике", т.е. при сложении и вычитании в адресе команды.

Пример 5. Обращение к СП № 5 *sinx* в ИС-2
I6, *f* +I, 750I, 76I0
, *x* , 5 , *y* ;
Передача управления
56, , *f* +3;
Засылка сформированной команды
I3, *f* -7, *N* I, *f* -7;

В приведенных выше примерах адреса-идентификаторы могут быть метками команд (пример 2 и I), **метками** констант и неописанными идентификаторами - адресами переменных.

Распределение памяти происходит автоматически, при трансляции. Сохраняются все возможности системы команд ЭВМ. Программа удобна для чтения.

Описываемые ниже операторы автокода помогают составить программу наиболее удобным образом и "заказать" (т.е. повлиять) распределение памяти (зарезервировать память для массивов, наложить массивы друг на друга, использовать ранее составленную и независимо транслированную программу и т.д.).

Список операторов автокода:

BSS - оператор резервирования памяти
EQU - оператор эквивалентности
BLOCK - оператор эквивалентности общих величин нескольких программ
COMMON - описание общей величины
CALL - оператор обращения к внешней программе
SUBR - описание внешней программы
REAL - описание вещественной константы
TEXT - описание текстовой константы
DATA - описание данных, рассылаемых при загрузке на счет
SET - оператор рассылки
ENTRY - описание входа в программу извне
IDENT, *END* - описание библиотечной программы

Рассмотрим более подробно использование этих операторов при составлении программы на соответствующих примерах.

А. Операторы резервирования и распределения памяти

Оператор *bss* используется для резервирования памяти ОЗУ для массивов и для фиксации расположения величин в ОЗУ.

Пример 6:
```
x : bss , 100;
y : bss , 300;
```

Это означает, что массиву с меткой x ("массиву X") нужно отвести 100_8 ячеек; массиву с меткой y ("массиву y ") нужно отвести, вслед за массивом x, 300_8 ячеек.

Пример 7.
```
t : bss , 1 ;
u : bss , 1 ;
v : bss , 1 ;
```

Это означает, что величинам t, u, v будет отведено подряд по одной ячейке памяти.

В автокоде нельзя ставить метку у метки ($x : v : 02, a, b, c$; писать нельзя); можно, используя массив нулевой длины (bss , 0) написать таким образом, что команда или константа, или оператор, или массив будут иметь больше одной метки.

Пример 8.
```
z : bss , 0;
x : bss , 100;
```

Это означает, что массив x, длиной в 100_8 ячеек можно использовать под названием z (с меткой z).

Адрес bss может содержать только ранее описанные идентификаторы (идентификатор, встретившийся в виде метки в выше написанных командах и, в частности, в самой команде bss , которая на него ссылается).

Пример 9.
```
y : bss , 23;
z : bss , z-y ;
```

Это означает, что массив z имеет ту же размерность (23_8 ячейки), что и массив y, вслед за массивом y .

Оператор equ присваивает метке-идентификатору числовое значение или определяет ее через посредство других ранее описанных идентификаторов (меток).

Пример 10. `y : equ , 100;`

Это означает, что при трансляции вместо идентификатора y везде будет поставлен адрес 0100.

Пример 11:
```
x : bss , 400;
y : equ , x+100;
```

Это означает (сравните пример 6), что массиву x будет отведено 400_8 ячеек, но затем второй оператор перераспределит общий массив в 400_8 ячеек, отведя для массива y место спустя 100_8 ячеек от начала массива x .

Нельзя писать x: bss, 400;
y: equ, y-x; (сравните пример 9).

В операторе equ (в отличие от bss) идентификатор, стоящий в левой части (метка), не может появиться в правой части (в адресе); выражение y: equ, y ; бессмысленно.

Операторы block и COMMON используются для резервирования и распределения общей памяти нескольких программ транслированных независимо друг от друга.

Пример I2: Программа I

```
FILTR: IDENT;
    Z: block ;
    A: COMMON,1;
    B: COMMON,3;
    . . . . . .
    . . . . . .
       END,0;
```

Программа 2

```
CORREL: IDENT;
    Z: block;
    E: COMMON,2;
    F: COMMON,2;
    . . . . . .
    . . . . . .
    END, 0;
```

В программах, называемых FILTR и CORREL будут отведены общие ячейки для величин A,B и E,F ; каждой величине отводится указываемое в COMMON число ячеек, а именно:

Z	величина A, величина E
Z+1	величина B, величина E+I
Z+2	величина B+I, величина F
Z+3	величина B+2, величина F+I.

Обе программы могут быть транслированы одновременно или независимо друг от друга. Информация о наличии общего массива Z , и распределение его операторами COMMON поступит в таблицу описаний библиотечной программы (СМ) и будет реализована загрузчиком при компиляции разделов из соответствующих библиотечных программ. Если в программе имеется только один общий блок (в принципе их может быть несколько) оператор BLOCK может быть опущен; транслятор заводит общий блок вида ~~xx~~: block ; Если одна из программ вызывающая (т.е. имеет оператор CALL), а другая вызываемая (к ней обращаются через оператор CALL), то общая длина COMMON в вызываемой программе не должна превышать общей длины COMMON в вызывающей программе, по каждому помеченному BLOCK отдельно.

Б. Операторы описания библиотечной программы

Операторы *IDENT* и *END* описывают программу или подпрограмму, которая в результате трансляции должна стать отдельной библиотечной программой (стандартным массивом).

Пример I3:

```
FILTR: IDENT;
. . . . . .
. . . . . .   тело программы I
. . . . . .
. . . . . .
       END,0;
CORREL: IDENT;
. . . . . .
. . . . . .   тело программы 2
. . . . . .
       END,0
MARK: IDENT;
. . . . . .
. . . . . .   тело программы 3
. . . . . .
       END,0;
```

В данном примере все три программы с названиями *FILTR* , *CORREL* и *MARK* будут транслированы в отдельные библиотечные программы; все три программы в принципе могут быть использованы самостоятельно в других задачах, исходя из реализованного в каждой из них алгоритма; могут быть частью задачи, которая использует их одновременно и т.д.

В. Операторы описания и обращения к внешней программе

Оператор *SUBR* описывает внешнюю программу.

Каждая библиотечная программа по отношению к другой библиотечной программе является внешней. Если одна программа обращается к другой программе по команде I6, *f* +I, *PROGR* +I, *PROGR* ; то одновременно с этой командой должно присутствовать описание *PROGR* : *SUBR* ; здесь *PROGR* название вызываемой программы.

Пример I4: Вызывающая библиотечная программа

```
FILTR : IDENT ;
CORREL: SUBR ;

     ..............
     ..............
16,1+1,CORREL+1,CORREL;
     ..............
     ..............
END,0;
```

Вызываемая библиотечная программа

```
CORREL : IDENT;
     ..............
     ..............
     ..............
     ..............
     ..............
     ..............
     ..............
END,0;
```

Оператор CALL используется для обращения к внешней программе. Этот оператор равносилен обращению к внешней программе примера I4. Транслятор программирует выполняемую часть оператора CALL -командой и декларативную - ссылкой в таблицу внешних объектов.

Пример I5:

```
FILTR : IDENT;
     ..........
     ..........
        CALL,CORREL
     ..........
     ..........
        END,0
```

```
CORREL : IDENT ;
     ..........
     ..........
     ..........
     ..........
     ..........
        END,0
```

Если библиотечные программы FILTR и CORREL (примеры I4 и I5) (по информации к загрузчику "сегментация разделов") будут загружены в один раздел, то обращение к программе CORREL произойдет по команде I6, 1+I, CORREL+I, CORREL ; с возвратом в вызывающую программу FILTR .

Если библиотечные программы FILTR и CORREL будут загружены в разные разделы, то в вызывающей программе FILTR , автоматически (загрузчиком) программируется обращение через резидент, с вызовом в ОЗУ раздела, в котором находится вызываемая программа CORREL ; резидент передает управление с возвратом на программу CORREL и затем в программу FILTR на продолжение работы.
Так же происходит обращение к программе через вход, описанный оператором ENTRY .

Пример I6:

```
FILTR : IDENT;
     ..............
     ..............
          CALL,CORR;
     ..............
     ..............
          END,0;
```

```
CORREL : IDENT;
     ..........
     ..........
CORR : ENTRY;
     ..........
     ..........
     END,0;
```

Оператор *ENTRY* описывает вход в программу (подпрограмму) из другой библиотечной программы. Для некоторых программ по смыслу их использования может понадобиться заведомо больше одного входа извне (один вход - на начало программы: *PROGR : IDENT* ; - имеется всегда); все другие входы описываются оператором *ENTRY* , как в примере I6 в программе *CORREL* . Операторы *ENTRY* обязательно должны быть помечены (в примере метка *CORR*). После трансляции в таблице входов типа *ENTRY* будут отмечены все входы в программу, включая и вход на начало программы.

По этой информации загрузчик (резидент) осуществляет обращение к программе.

Примечание: ячейки, обозначенные операторами *ident* и *entry*, суть ячейки возврата в вызывающую программу.

Г. Операторы описания типа величин

Оператор *REAL* описывает вещественную константу.

Пример I7: *PI: REAL , 3I.4I5926E-I;*
или *PI: REAL , 3.I4I5926;*
или *PI: REAL , 0.03I4I5926E+2;*

Все три записи числа (π = 3,I4I5926) эквивалентны. Целая часть числа отделяется от дробной части десятичной точкой (символ .); буква E указывает, что следующие символы (знак и цифры) являются десятичным порядком. В результате трансляции в ячейке с меткой *PI* будет находиться двоичное число в машинном коде (с машинным порядком и мантиссой).

Оператор TEXT описывает текстовую константу.

Пример I8: *?HALEN: TEXT , 10 , 0123456789 ;*
TEST: TEXT , 18 , ABCDEFGIJKLMNOPQRS;

Текстовая константа может занимать несколько ячеек подряд. После оператора TEXT стоит десятичное число (в примере I0 и I8), которое указывает сколько символов (букв и цифр) в константе. После трансляции символы константы (от запятой до точки с запятой) представлены

во внутреннем шестиразрядном коде, в одной ячейке помещается 6 символов.

Оператор *DATA* описывает данные, которые после рассылки их оператором *SET* во время загрузки раздела исчезают из оперативной памяти.

Пример I9:

```
. . . . . .
    DATA ;
NAME: TEXT, 7, PROGRAM ;
    DATA ,1 ;
CONST: 1; 2 ;
. . . . . .
```

Д. Оператор рассылки

Оператор *SET* рассылает информацию в момент компиляции и загрузки раздела, описанную оператором *DATA* или любую другую информацию (часть программы и т.д.).

Общий вид оператора *SET* :

SET , INDATA , ONDATA , LNDATA , CIKDAT ,

где *INDATA* – метка массива, куда засылаются данные,
ONDATA – метка массива, откуда выбираются данные,
LNDATA – количество данных, участвующих в рассылке,
CIKDAT – число повторений оператора *SET* ("засев" массива повторяющейся информацией).

Если *LNDATA* и *CIKDAT* равны I, то их можно опустить.

Пример 20:

```
. . . . . .
. . . . . .
        DATA,0 ;
NAME: TEXT, 7, PROGRAM;
        SET, M1, NAME ,2 ;
        DATA , 1 ;
CONST: 1 ;
        2 ;
        SET , VES, CONST, 2,3 ;
        DATA , 2 ;
DAT: REAL , 9.0;
        REAL , 9.7 ;
        REAL , 2.3 ;
        SET, P1 , DAT ;
        SET, P4, DAT+1 ;
. . . . . .
```

После трансляции в библиотечной программе все данные находятся пока в массивах *DATA* (с метками *NAME* , *CONST* , *DAT*); каждый оператор *SET* транслируется в особые инструкции и занимает две ячейки; порядок следования информации *DATA* и инструкций *SET* сохраняется; данные еще не рассылались, но операторы TEXT и *REAL* выполнены. При составлении и компиляции раздела загрузчик с помощью инструкций *SET* разошлет информацию *DATA* в массивы (*NAME* → *M1*, *CONST* → *VES* , *DAT* → *P1* , *DAT*+I → P4), а сами массивы *DATA* и инструкции *SET* аннулируются.

3. Описание работы загрузчика

Загрузчик составляет программу в "истинных адресах" из библиотечных программ (СМ), уже записанных во временную библиотеку программ на МБ транслятором с автокода, и тех библиотечных программ (СМ), которые были ранее независимо транслированы, выданы на перфокарты и введены с перфокарт в ОЗУ загрузчиком.

Загрузчик (в режиме "дозапись СМ") продолжает запись библиотечных программ в каталог и временную библиотеку программ на МБ, в порядке их следования в "пакете СМ" пользователя, пока не встретит перфокарту "конец записи СМ".

Затем загрузчик требует ввода перфокарты "сегментация разделов"; по информации в этой перфокарте начинается компиляция (составление и настройка по месту работы в ОЗУ) разделов и запись их в каталог и библиотеку разделов на МБ.

Для простой и короткой программы перфокарта "сегментация разделов" может иметь такой вид:

Пример I: *a* ,

где "*a*" – название программы (метка), на которую должно быть передано управление при выходе на счет задачи пользователя. Все библиотечные программы пользователя в этом случае загрузятся в один раздел, программа "*a*" загрузится головной.

Для сложных программ (суммарная длина программ и блоков информации больше ОЗУ) перфокарта "сегментация разделов" составляется по принципу скобочных ссылок:

Пример 2: $a(b(c,d),e(f,g))$,

где программа "a" обращается (вызывает) к программам "b" и "c" (программы "первого уровня"), программа "b", в свою очередь, обращается (вызывает) к программам "c" и "d"; программа "e" обращается (вызывает) к программам "f" и "g". Программы "c", "d" и "f", "g" - программы "второго уровня".

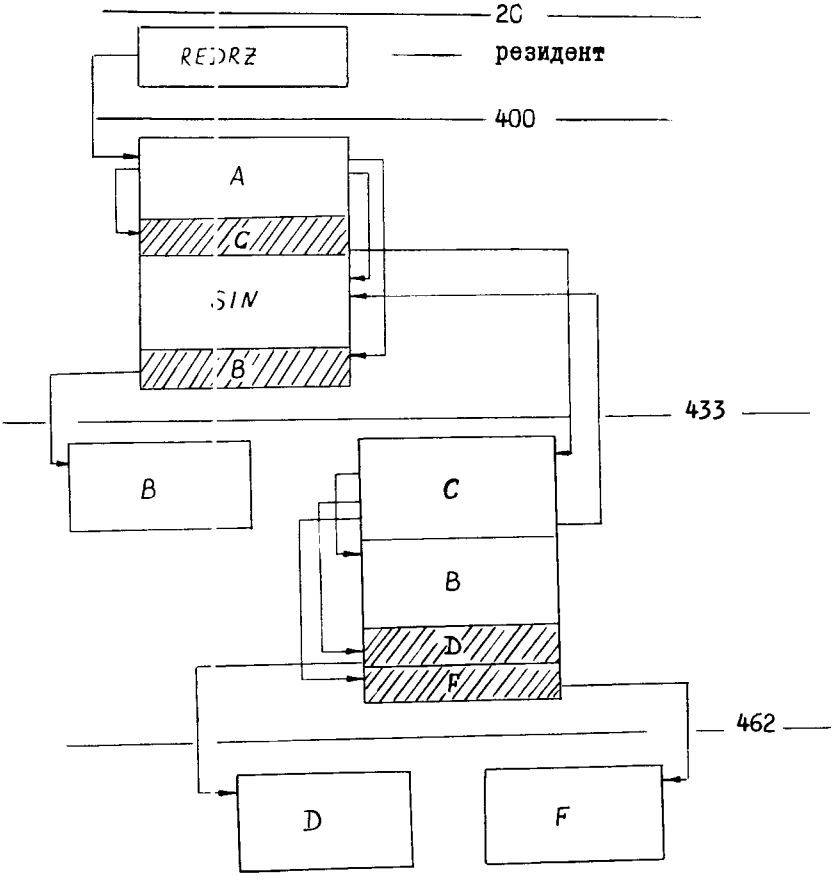
Если в программах "a", "b", "c", "d", "e", "f", "g" (примеры I и 2) есть обращения к другим внешним объектам - библиотечным программам, которые не отражены в скобочных ссылках или вовсе не упомянуты, эта программа (программы) загружается в тот же раздел, в котором находится вызывающая программа. Если программа загрузилась в "верхний" раздел, т.е. этот раздел уже находится в ОЗУ, а работает "нижний" раздел, то к подпрограммэ "верхнего" раздела обращение идет не через резидент, а по командэ передачи управления с возвратом в вызывающую программу "нижнего" раздела. Снизу нельзя вызывать только головную программу "верхнего" раздела, да это и бессмысленно ("a" вызывает "c", "c" вызывает "a"; но "c" может вызывать другие программы раздела, в котором головной программой является "a").

На примере программы с несколькими разделами рассмотрим работу загрузчика по компиляции разделов и составление информационной перфокарты "сегментация разделов".

Пример 3. Пусть в библиотеке программ на МБ имеется несколько программ с названиями "A", "B", "c", "D", "F", "sin", записанных в библиотеку при трансляции с автокода и загрузчиком (в режиме "дозапись СМ"). Кроме этих программ, в библиотеке программ на МБ, могут находиться любые другие программы, записанные при трансляции с автокода, и не используемые при счете задачи.

Предположим, что шесть вышеперечисленных задач имеют общую длину больше ОЗУ, поэтому мы не можем расположить их подряд одну за другой в одном разделе. Исходя из смысла задачи, можно продумать такую сегментацию разделов, чтобы некоторые разделы настраивались для работы на одни и те же участки памяти ОЗУ.

Нарисуем схему задачи, стрелками укажем обращение к программам (от вызывающей к вызываемой) и возврат после их работы.



Программа "A" обращается к программам " *sin* ", "B", "C".

Программа "C" обращается к программам "B", "*D*", "*F*", " *sin* ".

Согласно схеме информация "сегментация разделов": A(B,C(*D*,*F*)).

Начальный адрес настройки разделов (раздела № 0) ячейка 0400.

С ячейки 0020 по ячейку 377 в МОЗУ постоянно находится служебная

программа резидент (во время счета задачи). В информации "сегментация разделов" нет упоминания о библиотечной программе "*sin*", поэтому при составлении разделов она попадает в вызывающий ее раздел (в тот раздел, в котором есть к ней обращение (в раздел *№ 0*). В скобочных ссылках из программы С (*D*, *F*) нет упоминаний о программе В и программе "*sin*", к которым она обращается. Так как программа "*sin*" есть в "верхнем" разделе (разделе *№ 0*), и этот раздел будет находиться в памяти во время работы программы "С", то эта программа не загружается в раздел программы "С". Программа "В"загрузится дважды - во-первых, как раздел, вызываемый программой "А" через резидент, с начальным адресом настройки - адресом α; во-вторых, в раздел программы "С", в которой есть к ней обращение; загрузчик загружает разделы формально, раскрывая поочередно скобки "сегментации разделов" и просматривая обращения к внешним объектам в настраиваемой программе. Программы одного уровня скобок (В,С - первый уровень, *D*, *F* - второй уровень) в "сегментации разделов" настраиваются на работу с одного адреса МОЗУ разделом, в котором поименованная программа является головной (заголовок раздела).

Имея в виду все сказанное выше, можно задать информацию "сегментация разделов" так: А (С(*D* , *F*)) и тогда программа "В" загрузится только один раз в раздел с заголовком А. Но экономии памяти в таком варианте тоже не будет. Только за счет разделов "*D*" и "*F*", работающих с одного адреса МОЗУ (адрес β), экономится память.

Разделы задачи составятся таким образом:

Раздел № 0 - головная программа А, программа *sin* ; адрес настройки раздела в ОЗУ - 0400.

Раздел № I - головная программа В; адрес настройки раздела в ОЗУ - адрес α.

Раздел № 2 - головная программа С, программа В; адрес настройки раздела в ОЗУ - адрес α .

Раздел № 3 - головная программа *D* , адрес настройки раздела в ОЗУ - адрес β.

Раздел № 4 - головная программа *F* ; адрес настройки раздела в ОЗУ - адрес β.

Все обращения к программам других разделов осуществляются через резидент (кроме обращения к неголовной программе "верхнего" раздела; в примере из раздела "С" к программе " *SIN* " раздела "А"). При составлении и настройке раздела загрузчик автоматически программирует "программу - куклу" вызова через резидент раздела, в котором находится вызываемая программа. Резидент в динамике считывает из библиотеки разделов на МБ вызываемый раздел и передает управление на головную программу раздела; вызванный раздел работает, снова обращается в резидент (через программу - куклу) за вызовом нового раздела и т.д. Выход из раздела, после его работы, всегда в резидент (включая и раздел *NO*). Резидент восстанавливает ячейку возврата в вызывающий раздел и передает управление на продолжение работы. Конец счета задачи - останов КРА 0017 77 --- (в резиденте), если не предусмотрен собственный программный останов.

Обращение к программам (подпрограммам) внутри раздела, а также обращение к программам (не головной) "верхнего" раздела осуществляются по команде передачи управления с возвратом в вызывающую программу.

Программа вызова раздела ("программа-кукла"), которую автоматически программирует загрузчик, в автокодной записи выглядит так:

```
NAME : IDENT ;
       16 ;
       CALL , RESIDE ;
INFRA2 : ,, ALFRA2 , NRA2 ;
       56 , , 1-3 ;
```

NAME - заголовок вызываемого раздела (метка головной программы),

ALFRA2 - адрес настройки в ОЗУ вызывающего раздела,

NRA2 - номер вызываемого раздела.

Загрузка задачи на счет осуществляется резидентом, который считывает в ОЗУ из библиотеки разделов на МБ раздел *NO* и передает управление на головную программу раздела.

При работе загрузчика - составление и настройка очередного раздела - на АЦПУ выдается информация о загрузке, которая в общем случае выглядит так:

```
M  P  α1
M1 E  α2
N  S  α3
Z  C  α4
P  R  α5   N1
Q  R  α6   N2
```

Информация о загрузке в библиотеку разделов на МБ раздела с заголовком M

При этом предполагается, что программа "M" содержит следующие операторы:

```
M: IDENT ;
. . . . . .
M1: ENTRY ;
. . . . . .
    CALL,N;
. . . . . .
Z: block ;
. . . . . .
    CALL,P ;
    CALL,Q ;
. . . . . .
     END,0;
```

Программа "M"

В информации о загрузке метки M, M1, N, Z, P, Q соответствуют названиям загружаемых объектов; символы после меток (P, E, S, C, R)-характеристики объектов; $\alpha 1, \alpha 2, \ldots, \alpha 6$ истинные адреса после настройки раздела; N1, N2 указание номера раздела (для вызываемых разделов). Символы характеристик следующие:

символ P - указание заголовка раздела (название головной программы);

символ E - указание на вход в программу (тип ENTRY) извне;

символ S - указание на внешнюю программу (библиотечную программу), загруженную в этот же раздел;

символ C - указание на блок общих ячеек, описанных оператором block ;

символ R - заголовок раздела, вызываемого в этом разделе, через резидент.

Печать загрузки для задачи примера 3 будет следующей: (автокодный текст ее приведен ниже)

```
A      P    0400          Загрузка раздела № 0
C      R    0415    0002
sin    S    0421
```

```
B   R   0425   0001
z   C   0431
B   P   0433              Загрузка раздела № I
C   P   0435              Загрузка раздела № 2
CI  E   0441
B   S   0445
D   R   0452   0003
F   R   0456   0004
D   P   0462              Загрузка раздела № 3
F   P   0462              Загрузка раздела № 4
```

В каждом разделе первый символ P указывает, что идентификатор слева - название головной программы раздела. В разделе NO есть идентификатор z с указанием характеристики (символом C) на блок общих ячеек. В данном случае блок общих ячеек z (z: block) был описан в программе "C".

Адреса при символах R (415, 425, 452, 456) с номером вызываемого раздела - истинные адреса вызывающих (через резидент) "программ - кукол".

Примечание: Перфокарта "сегментация разделов" пробивается только в пятиразрядном телетайпном коде на устройстве телетайп. При необходимости пробивать сегментацию разделов на устройстве УПП следует воспользоваться перекодировочной программой "УПП —→ ТТ".

Автокодный текст задачи примера 3 см.в Приложении I.

4. Описание инвариантного языка "Макрос"

4.1. Общие сведения о трансляции с языка Макрос

Инвариантный язык Макрос разработан в ЛВТА ОИЯИ для программирования алгоритмов задач на ЭВМ с разными системами команд.

Язык Макрос представляет собой некоторый промежуточный уровень между алгоритмическим языком (Алгол, Фортран...) и автокодом ЭВМ.

Инвариантность реализуется следующим образом. Каждому оператору языка Макрос ставится в соответствие команда (группа команд) автокода конкретной ЭВМ.

Пример 1: Оператор Макроса — Команды автокода (БЭСМ-4)

.IF. X1=X2 .GO TO. X3; < 15, X1, X2 ; 36 , , X3 ; <

"левая часть" — "правая часть"

Металингвистический знак (символ <) разделяет левую и правую части строки. Составленная из подобных строк "таблица соответствий" вводится транслятором с Макроса в ОЗУ и расформировывается в два массива с запоминанием начальных адресов каждого оператора и соответствующей ему группы команд автокода конкретной ЭВМ.

Процесс трансляции программы с языка Макрос состоит в том, что в каждом программном операторе заменяют фактические идентификаторы - формальными X1,X2...X7 в массиве "левых частей" находят оператор, совпадающий с программным; из массива "правых частей" извлекается соответствующая оператору группа команд автокода; в командах автокода формальные идентификаторы X1,X2,...,X7 заменяют фактическими.

Пример 2: Оператор Макроса в программе

.IF. ALFA = BETA .GO TO. SUMMA1 ;

Результат трансляции в автокод БЭСМ-4.

15 , ALFA , BETA ; 36 , , SUMMA1 ;

В примере 2 фактические идентификаторы оператора Макроса *ALFA*, *BETA*, *SUMMA1* будут заменены формальными идентификаторами X1, X2, X3; в массиве "левых частей" найдем оператор Примера I; в командах автокода из массива "правых частей" заменим формальные идентификаторы X1, X2, X3 на фактические идентификаторы *ALFA*, *BETA*, *SUMMA1*;

Имея в виду все вышенаписанное, можно говорить о "выполнении" оператора в том же смысле, как о "выполнении команды" ЭВМ, не входя в подробности вида оператора в записи "истинные адреса", при счете по транслированной программе на ЭВМ.

Программа на языке Макрос представляет собой последовательную запись операторов и констант, определяемую алгоритмом задачи.

Работа программы (счет) состоит в "выполнении" каждого оператора в "естественном" порядке следования; пока не встретится оператор, нарушающий этот порядок (оператор перехода) и задающий последовательное "выполнение" других операторов программы.

Сменой "таблицы соответствий" определяется возможность при трансляции с языка Макрос на ЭВМ БЭСМ-4 получить программы для счета на ЭВМ Минск-22, Минск-2, БЭСМ-6 и др. ЭВМ[4].

В принципе пользователю доступна "таблица соответствий"; ее можно дополнять и изменять, соблюдая некоторые правила; и, таким образом, расширять (сужать) рамки языка Макрос и автокод конкретной ЭВМ. В Приложении П приводится "таблица соответствий"ЭВМ БЭСМ-4.

В программе на языке Макрос можно писать автокодные команды, константы и операторы; транслятор с Макроса транслирует их без изменений, как автокодные константы (в "таблице соответствий" есть в "левой части" соответствующие операторы; одинаковые с "правыми частями" - автокодными командами).

В программе на Макросе нельзя писать автокодные операторы *REAL* (из-за наличия в нем десятичной точки) и TEXT (в константе сохранится только 6 символов текста).

Автокодные операторы *IDENT* и *END* (см. 2.2. Описание элементов языка и операторов автокода) должны обязательно присутствовать в программе на языке Макрос, в начале и в конце программы (описание программы).

Программа на языке Макрос может состоять из нескольких подпрограмм; каждая подпрограмма, описанная операторами *IDENT* и *END*, транслируется (сначала с Макроса, затем с автокода) в отдельную библиотечную программу (стандартный массив).

4.2. Элементы языка Макрос

Символ – латинская буква, цифра, знак раздела или арифметической операции.

Служебный символ – заранее обусловленная последовательность символов, окаймленная левой и правой точкой (символ .).

Идентификатор – последовательность букв и цифр, начинающаяся с буквы (не более 6 символов); условный адрес переменной.

Метка – идентификатор, стоящий слева от оператора и отделяющийся от него двоеточием (символ :); условный адрес оператора, константы, массива.

Оператор – заранее обусловленная последовательность служебных символов, идентификаторов, знаков раздела, знаков арифметических операций и восьмеричных констант.

Константа – целое число или вещественное число.

Шкала – составная величина из целых величин (вырезаются разряды целой величины, несущие информацию и формируются в составную величину).

Знаки арифметических операций:

символ + знак операции сложения
символ – знак операции вычитания
символ * знак операции умножения
символ / знак операции деления
символ ↑ знак операции возведение в степень.

Знаки раздела – разделители

символ . точка; выделение служебного символа слева и справа
символ ; точка с запятой; конец оператора

символ : двоеточие; разделяет метку и оператор
символ , запятая; разделяет идентификаторы
символ > знак засылки; (⟶ в рукописи программы)
символ = знак присваивания
символ < знак раздела левой и правой части "таблицы соответствий"; запрещается использовать в других целях.

Пример 3. Служебные символы

```
.GOTO.   .SUBR.   .IND.
```

Идентификаторы

```
MARS  M1  CORREL  SUM5
```

Операторы

```
.GOTO. MARS ;     SUM : M1 = A + B2 ;
```

Константы целые

```
M1: 172; M2: 7777; M3: 0; M4 : 5;
```

Константы вещественные

```
R1: 101, 4000; K1 : 127, 0240, 1217, 1010;
```

В примере 3 метка оператора - *SUM* ; метки констант: M1,M2,M3,M4, *R1*, K1. Операторы могут быть помечены или не помечены. На помеченный оператор можно ссылаться (например, передавать управление) в других операторах.

4.3. Описание операторов языка Макрос

В описываемых операторах при их использовании в программе меняются только формальные идентификаторы X1,X2...X7, порядок и вид остальных частей оператора не меняется.

Пример 4: Описан оператор *.GOTO. X1* ;
Оператор в программе *.GOTO. MARS*;

Если идентификатор в программе встретился в виде метки (в примере 3: *SUM* , K1, *R1*) он называется описанным, те идентификаторы, которые встретились только в составе оператора, называются неописанными и при распределении памяти (после трансляции) им отводится по одной

ячейки памяти ОЗУ и состояние этих ячеек не определено, пока в них не поступит информация при счете по программе. Таким образом, можно сказать, что неописанные идентификаторы получают значения в результате "выполнения" операторов.

Идентификатор является одновременно названием и адресом величины. Идентификатор можно складывать (вычитать) с другим идентификатором. Идентификатор – сумма имеет смысл суммы двух адресов, которые будут присвоены при распределении памяти обоим идентификаторам-слагаемым. Тот же смысл имеет сумма идентификатора и индекс-регистра ЭВМ, идентификатор-сумма это адрес идентификатора-слагаемого, увеличенный на число единиц индекс-регистра ЭВМ.

А. Операторы засылки

I.	*X1 > X2;* *X2 = X1;*	Идентификатору X2 присваивается значение идентификатора X1.
2.	*X1 > . IND. ;*	Целая величина X1 посылается в индекс-регистр.
3.	*X1 > X2 + . IND. ;*	Величина X1 посылается в идентификатор-сумму идентификатора X2 индекс-регистра.
4.	*X1 + . IND. > X2;*	Величина из идентификатор-суммы (X1 + . *IND.*) присваивается идентификатору X2.
5.	*X1 > X2 + X3 + . IND. ;*	Величина X1 присваивается идентификатору-сумме (X2+X3+ . *IND.*).
6.	*X1 + X2 + . IND. > X3 ;*	Величина из идентификатора-суммы (X1+X2+ + . *IND* .) присваивается идентификатору X3.
7.	*X1 > X2 + X3 ;*	Величина X1 присваивается идентификатору-сумме (X2+X3).
8.	*(X1 + X2) > X3 ;*	Величина из идентификатора-суммы (X1+X2) присваивается идентификатору X3.
9.	*X1 > X2 - X3 ;*	Величина X1 присваивается идентификатору – разности (X2–X3).
IO.	*(X1 - X2) > X3 ;*	Величина из идентификатора-разности (X1–X2) присваивается идентификатору X3.
II.	*X1 > X2 +, IND. + . CUB. ;*	Величина X1 присваивается идентификатору-сумме (X2 + . *IND.*) второго куба ОЗУ (ОЗУ № 01).

I2. XI+.*IND*.+.*CUB* .>X2; Величина из идентификатора-суммы (X2+.*IND*.) второго куба ОЗУ присваивается идентификатору X2.

Б. Операции над вещественными числами

I3. XI = X2; Оператор присваивания идентификатору XI значения идентификатора X2.

I4. XI = X2 + X3; Операция сложения. Идентификатору XI присваивается значение суммы двух величин X2 и X3.

I5. XI = X2 - X3; Операция вычитания. Идентификатору XI присваивается значение разности двух величин X2 и X3.

I6. XI = X2 x X3; Операция умножения. Идентификатору XI присваивается значение произведения двух величин X2 и X3.

I7. XI = X2 / X3; Операция деления. Идентификатору XI присваивается значение частного двух величин X2 и X3.

I8. XI = X2 ↑ X3; Операция возведения в степень; идентификатору XI присваивается значение результата возведения величины X2 в степень X3.

В. Операции над целыми и вещественными числами

I9. .*INT*.*X1*>.*REAL*.*X2*; Присвоить идентификатору X2 вещественное представление целой величины XI.

20. .*REAL*.*X1*>.*INT*.*X2*; Присвоить идентификатору X2 целое представление вещественной величины XI.

Г. Операции над целыми числами и шкалами

2I. XI + X2 > X3; Сумма целых величин XI и X2 (целое сложение) посылается в идентификатор X3.

22. XI - X2 > X3; Разность целых величин XI и X2 (целое вычитание) посылается в идентификатор X3.

23. XI.*AND*.X2 > X3; Логическое умножение целых величин XI и X2, результат посылается в идентификатор X3.

XI разрядов. Меняя X3 и XI и повторяя оператор расформировываем составную величину-шкалу полностью. "Правое расформирование шкалы".

29. . *LDESH*. XI,X2,X3 > X4; Составная величина - шкала X3 расформировывается слева по X2 разрядам (восьмеричное число) и посылается в целом представлении в X4;
XI - восьмеричное число, номер разряда, ограничивающего слева составную величину X3 (для "полной ячейки" БЭСМ-4 XI равняется 44); оставшаяся часть шкалы сдвигается влево на число освободившихся X2 разрядов. Меняя X4 и X2 и повторяя оператор, расформировываем шкалу полностью. "Левое расформирование шкалы".

D. Операторы перехода (передачи управления)

30. *.GOTO.* XI; Безусловная передача управления на оператор с меткой XI.

Оператор безусловной передачи управления задает последовательное "выполнение" операторов, начиная с помеченного оператора, до тех пор, пока не будет встречен новый оператор перехода (безусловный или условный).

Во всех операторах условной передачи управления переход к "выполнению" помеченного оператора и следующих за ним операторов происходит только при совпадении ("истина" - "истина" или "ложно" - "ложно") высказывания, утверждающего истинность (ложность) выражения и того, что определяется в действительности, при проверке выражения с конкретными XI и X2.

Пример 5: Утверждается, что высказывание "XI меньше X2" "ложно"; и действительно (для XI = 2 X2=2) при проверке конкретных XI и X2 выясняется, что XI не меньше X2; при выполнении оператора

.IF. X1 .L. X2. ELSE X3 ; в таком случае ("ложно"-"ложно") будет передано управление на оператор с меткой X3.

При несовпадении ("истина"-"ложно" или "ложно"-"истина") высказывания и результата проверки конкретных X1 и X2 перехода к помеченному оператору не происходит; порядок "выполнения" операторов остается прежний, т.е. "выполняется" следующий за оператором условной передачи управления оператор, затем следующий и т.д.

31. *.IF. X1 = X2 . GOTO. X3 ;* — Управление передается на оператор с меткой X3, если высказывание <X1 равно X2 есть "истина" > истинно. X1 и X2 целые числа.

32. *.IF. X1 = X2 . ELSE . X3 ;* — Управление передается на оператор с меткой X3, если высказывание <X1 равно X2 есть "ложь"> истинно. X1 и X2 целые числа.

В операторах 33 ÷ 40 X1 и X2 – целые числа.

33. *.IF. X1 . L . X2 . GO TO. X3 ;* — Управление передается на оператор с меткой X3, если высказывание <X1 меньше X2 есть "истина"> истинно.

34. *.IF. X1. G. X2. GOTO. X3 ;* — Управление передается на оператор с меткой X3, если высказывание <X1 больше X2 есть "истина"> истинно.

35. *.IF. X1. LE. X2. GOTO. X3 ;* — Управление передается на оператор с меткой X3, если высказывание <X1 меньше или равно X2 есть "истина" > истинно.

36. *.IF. X1. GE. X2. GOTO. X3 ;* — Управление передается на оператор с меткой X3, если высказывание <X1 больше или равно X2 есть "истина" > истинно.

37. *.IF. X1. L. X2. ELSE. X3 ;* — Управление передается на оператор с меткой X3, если высказывание <X1 меньше X2 есть "ложь"> истинно.

38. *.IF. X1. G. X2. ELSE . X3 ;* — Управление передается на оператор с меткой X3, если высказывание <X1 больше X2 есть "ложь"> истинно.

39. .IF. X1 .LE. X2 .ELSE. X3 ; Управление передается на оператор с меткой X3, если высказывание <XI меньше или равно X2 есть "ложь"> истинно.

40. .IF. X1.GE.X2 ELSE .X3 ; Управление передается на оператор с меткой X3, если высказывание <XI больше или равно X2 есть "ложь"> истинно.

В операторах 41÷50 XI и X2 - вещественные числа

41. .IF.X1.LR.X2. GOTO.X3; Управление передается на оператор с меткой X3, если высказывание <XI меньше X2 есть "истина"> истинно.

42..IF. X1.GR.X2. GOTO.X3 ; Управление передается на оператор с меткой X3, если высказывание <XI больше X2 есть "истина"> истинно.

43. .IF.X1.LER.X2 GOTO. X3; Управление передается на оператор с меткой X3, если высказывание <XI меньше или равно X2 есть "истина"> истинно.

44. .IF.X1.GER. X2.GOTO. X3; Управление передается на оператор с меткой X3, если высказывание <XI больше или равно X2 есть "истина" > истинно.

45. .IF.X1.LR.X2. ELSE. X3 ; Управление передается на оператор с меткой X3, если высказывание <XI меньше X2 есть "ложь" > истинно.

46. .IF. X1.GR . X2 . ELSE. X3 ; Управление передается на оператор с меткой X3, если высказывание <XI больше X2 есть "ложь" > истинно.

47..IF.X1.LER. X2 ELSE. X3 ; Управление передается на оператор с меткой X3, если высказывание <XI меньше или равно X2 есть "ложь"> истинно.

48. . IF.X1.GER.X2. ELSE. X3; Управление передается на оператор с меткой X3, если высказывание < XI больше или равно X2 есть "ложь"> истинно.

49. *.IF. X1=X2, X3. GOTO. X4 ;* Управление передается на оператор с меткой X4, если высказывание < модуль разности /XI–X2/ меньше модуля /X3/ есть "истина" > истинно.

50. *.IF. X1=X2, X3. ELSE. X4 ;* Управление передается на оператор с меткой X4, если высказывание < модуль разности /XI–X2/ меньше модуля /X3/ есть "ложь/ > истинно.

E. Операторы цикла

51. *.BCIKL. ;* … *.ECIKL. X1, X2 ;* Оператор цикла. Группа операторов, начинающаяся оператором *.BCIKL.;* и заканчивающаяся оператором *.ECIKL.* XI, X2; будет повторена XI раз (восьмеричное число); X2 – метка оператора *.BCIKL.;*

Пример 6:

```
DO1 : . BCIKL. ;
      ARK+. IND. > R1 ;
      VES+. IND > R2
      SUM = R1 + R2 ;
      SUM > RES+. IND. ;
      . ECIKL. 100, DO1 ;
```

В примере 6 из массива с меткой *ARK* и из массива с меткой *VES* выбираются последовательно элементы (в данном случае вещественные числа), попарно суммируются и засылаются в массив с меткой *RES*. Длина трех массивов одинакова и равна 100_8 элементов (ячеек). Стрелкой указана передача управления на начало повторяющихся операторов. Оператор *.BCIKL.* по смыслу эквивалентен оператору № 2 XI >*.IND.*, если XI нулевая величина; (или 0 >*.IND.ZK* нулевой идентификатор описывает ячейку 0000, в которой постоянно находится нулевое машинное слово).

52. *.CLEAN. X1, X2 > X3 ;* Оператор циклической засылки; величина X2 засылается в массив с меткой X3, длиной XI (восьмеричное число) элементов. Массив X3 будет "засеен" однородной информацией. Величина X2 может быть целой, вещественной, текстовой.

Ж. Операторы ввода и вывода информации

53. `.DZU. >X1 ;` — С клавишного запоминающего устройства (для ЭВМ БЭСМ-4 с КЗУ-1У) выбирается информация и посылается в X1.

54. `.INPUT CARD. X1. X2 ;` — Ввод информации в ОЗУ, начиная с метки X2 в количестве X1 (восьмеричное число) элементов (строк перфокарт, кодов бумажной ленты и т.д.). (Для ЭВМ БЭСМ-4 это оператор ввода перфокарт с читающего устройства; метка X2 - адрес ввода информации; X1 - безразлично).

55. `.PRINT LINE. X1, X2;` — Вывод информации, начиная с метки X2 в количестве X1 (восьмеричное число) элементов. (Для ЭВМ БЭСМ-4 это оператор печати на АЦПУ массива с меткой X2 в количестве X1 ячеек. Информация в массиве формируется в требуемый формат программой пользователя).

3. Операторы описания и обращения к подпрограмме

56. `{ .SUBR. ;` / `{ .EXIT. ;` — Операторы описания подпрограммы; оператор `.SUBR.;` описывает вход в подпрограмму, оператор `.EXIT.;` описывает выход из подпрограммы. Оба оператора должны быть помечены. Метка оператора `.SUBR.` название подпрограммы.

Пример 7:

```
COREL: .SUBR. ;
{ тело
{ подпрограммы
ECOREL: .EXIT. ;
```

или

```
ECOREL: .EXIT. ;
COREL: .SUBR. ;
{ тело
{ подпрограммы
.GO TO. ECOREL ;
```

Оба описания подпрограммы `COREL` эквивалентны.

57. `.CALL. X1, X2;` — Оператор обращения к подпрограмме с названием X1 (метка оператора `.SUBR.`) и выходом X2 (метка оператора `.EXIT.`) Возврат из подпрограммы на оператор, следующий за оператором `CALL` X1,X2; в вызывающей программе.

(Оператор *call* и подпрограмма должны находиться в одной библиотечной подпрограмме).

Пример 8.

```
FILTR : IDENT,0;
EFILTR : .EXIT. ;
FILTR : .SUBR. ;
тело вызывающей подпрограммы:
. . . . . .
    .CALL. COREL , ECOREL ;
. . . . . .
    .GO TO. EFILTR ;

COREL : .SUBR. ;

тело вызываемой подпрограммы:
. . . . . .
. . . . . .
. . . . . .
ECOREL : .EXIT. ;
         END,0 ;
```

В примере 8 присутствуют два автокодных оператора *IDENT* и *END*, которые описывают программу *FILTR* (название головной подпрограммы), как "библиотечную", т.е. в результате трансляции с Макроса, затем с автокода, программа *FILTR* станет отдельной библиотечной программой (стандартным массивом; подпрограмма *FILTR* обращается к подпрограмме *COREL* "выполняя" оператор *.CALL.* ; после работы подпрограммы *COREL* возврат (передача управления) в вызывающую подпрограмму *FILTR* , на следующий за оператором *.CALL.* , оператор.

58. *.CALL. (X1),(X2)* ; Оператор обращения к подпрограмме по информации в ячейках с меткой X1,X2. Отличается от оператора № 57 тем, что название подпрограммы засылается в X1 (адрес оператора *SUBR*), а выход из подпрограммы в X2 (адрес оператора *.EXIT.*) в виде целых констант.

Пример 9.

```
Подпрограмма № 1          Подпрограмма № 2
COREL : .SUBR. ;          MULT : .SUBR. ;
. . . . . .               . . . . . .
. . . . . .               . . . . . .
. . . . . .               . . . . . .
ECOREL : .EXIT. ;         EMULT : .EXIT. ;
```

Если нам известно, что в процессе счета по программе вызывающая подпрограмма должна "вызвать" подпрограмму или *COREL* или *MULT* и какую именно станет известно только из результатов счета, то обращение к подпрограмме можно выполнить через оператор

. CALL. (MARK), (EMARK);

предварительно заслав в *MARK* и *EMARK* название и выход из подпрограммы, выполнив (если нужна, например, подпрограмма *MULT*) такие операторы:

K1 > MARK ;
K2 > EMARK ;

где K1: *MULT ;*
K2: *EMULT ;*
} целые константы Макроса

И. Оператор останова

59. *.STOP. ;* Оператор останова ЭВМ.

К. Оператор описания целой величины.

60. X1 ; Целая величина X1; восьмеричное число ≤ 7777 (для ЭВМ БЭСМ-4 расположено по второму адресу).

Л. Операторы описания автокодных констант

61. X1, X2, X3, X4;
62. , X1, X2, X3;
63. , , X1 , X2;
64. , , , X1;
65. X1, X2 ;
66. X1, X2+X3;
67. X2, X2-X3;

Все эти операторы при трансляции с языка Макрос не меняют вида, т.е. наличие этих операторов в левой части "таблицы соответствий" разрешает включать в текст программы на языке Макрос автокодные команды, операторы и константы. Эти операторы могут быть помечены и не помечены; формальные идентификаторы X1,X2,X3,X4 могут быть фактическими идентификаторами, восьмеричными числами и символами-названиями операторов автокода (*bss , equ , block , COMMON, CALL , subr , DATA , SET , ENTRY , IDENT и END , кроме REAL и TEXT*).

Включение в язык Макрос автокодных команд и операторов разрешает более гибко использовать выгоды написания программы на алгоритмическом языке (проще, нагляднее и т.д.) и выгоды использования автокода (резервирование массивов *bss* , обращение к внешним программам и т.д.).

В автокодных командах и операторах допускается "сложение и вычитание в адресе" в рамках таблицы соответствий
(*VES : EQU , MARS + 300* и т.д.).

Пример IO. Автокодные команды в тексте программы на языке Макрос.

```
Обращение к          I6,  S   , 75JI, 76IO;
СП № 5 sinx ИС-2    S: 0,  ARG ,   5 , SIN ;

Обращение к          16,  t   , 75JI, 76IO;
СП № I2I ИС-22.     t: 52,  a11,  I2I,  bI ;
Решение системы      52,  N   ,   J ,   M ;
уравнений (Гаусс)

Запись на МБ OI с    57,  40IO,   J , RP   ;
OI500 адреса с       50,   I4,  I500, MULT1  ;
контролем массива    70,  MULT,   0 ,   0;
MULT длиной        RP: bss , I;
I00₈ элементов     MULT1: equ , MULT+100;
```

Автокодные константы в тексте программы на языке Макрос

```
вещественные  PI :  I02, 6220, 7732, 5042;   число π
константы     CON1: IOI, 4000;                  I
               M2:  I77, I02I, 4000,  0 ;   I)^I6 · 0,25

              CEND :  777, 7777, 7777, 7777;
              CONA3 :   ,0,0, I5 ;
адресные      CONA2 :   ,0, I5, 0;
константы     CONA1 :   , I5,0, 0;
              IROGR :  0, 0 , COREL , 0;
```

Примечание: При пробивке на Макросе автокодных операторов необходимо, чтобы они имели вид операторов с номерами 61÷67. Для придания им такого вида рекомендуется использовать комментарии, например, оператор

test : ident , 0 ;

имеет вид *X1 , X2 ;* (оператор № 65).

5. Формирование пакета перфокарт задачи пользователя

Система работает с соответственным образом подобранными массивами перфокарт, образующими пакет.

Задача пользователя может одновременно состоять из частей: программы (подпрограммы) на языке Макрос; программы (подпрограммы) в автокоде, библиотечные программы (СМ) на перфокартах.

Библиотечная программа - это результат предыдущей трансляции части задачи пользователя (например, отдельных подпрограмм) или программа, взятая пользователем из библиотеки математического обеспечения системы.

Организация прохождения задачи пользователя на ЭВМ (трансляция и счет) осуществляется с помощью управляющих карт - системных карт.

А. "Пакет текста" задачи пользователя

Пользователь может пробивать текст программ (Макрос и автокод) на устройстве УПП, в семиразрядном коде УПП или на устройстве телетайп, в пятиразрядном телетайпном коде.

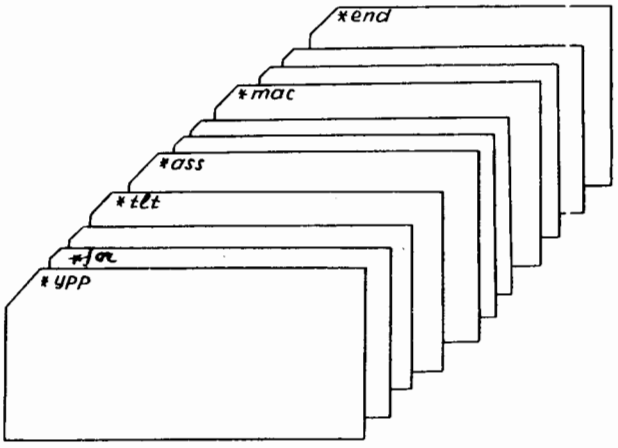
Команда (оператор), пробиваемая в коде УПП, заканчивается кодом 174 ("конец строки"). Команда (оператор), пробиваемая в телетайпном коде, начинается с "латинского (код 37) или цифрового (код 33) регистра" и заканчивается "переводом строки" (код 10) и "возвратом каретки" (код 02). Последняя строка перфокарты добивается латинскими регистрами. Удобно пробивать одну команду (оператор) на одной перфокарте.

Две системные карты ✱ *УРР* и ✱ *TLT* определяют код пробивки перфокарт "пакет текста". Если отсутствуют в пакете обе карты, считается, что код пробивки - УПП.

Две системные карты ✱ *MAC* (Макрос) и ✱ *ASS* (автокод) определяют язык текста программы. Перед перфокартами части программы на языке Макрос ставят карту ✱ *MAC* , перед перфокартами с автокодным текстом карту ✱ *ASS* . Части программы на Макросе и автокоде могут чередоваться в любой последовательности. Если в пакете отсутствуют обе карты, считается, что язык текста пакета - Макрос. За картой ✱ *for* следует текст задачи в языке ФОРТРАН.

Системная карта ✱ *END* - конец текста пакета пользователя (на МАКРОСе и АВТОКОДе). Присутствие этой карты в пакете обязательно в случае использования транслятора с МАКРОСа.

Пример формирования "пакета текста" задачи пользователя.



В примере карта *TLT (код пробивки – телетайп) стоит в середине программы на языке Макрос. Это значит, что все следующие за ней карты пробиты на устройстве телетайп; на устройстве телетайп пробиты также перфокарты следующей программы, написанной в автокоде. Системные карты *UPP, *TLT } пробиваются только в телетайпном коде

Системные карты: *MAC, *ASS, *JOB, *END } могут быть пробиты на телетайпе или УПП

пробиваются , начиная с I позиции, после чего их можно получить в телетайпном виде перекодировкой на ЭВМ.

Б. "Пакет СМ" задачи пользователя

Из библиотечных программ на перфокартах составляется "пакет СМ" задачи пользователя. Порядок следования библиотечных программ (СМ) в пакете любой.

Последняя карта пакета – карта особого вида:

Перфокарта "конец записи СМ": 777 7777 7777 7777 / 777 7777 7777 7777 КΣ

При отсутствии в задаче пользователя библиотечных программ (стандартных массивов) на перфокартах ,"пакет СМ" состоит только из карты "конец записи СМ".

В. Перфокарта "сегментация разделов"

Информация к загрузчику пробивается только на устройстве телетайп, в телетайпном пятиразрядном коде, эти карты могут быть получены на ЭВМ путем перекодировки с УПП.

Для небольшой задачи (в ОЗУ помещаются все программы и блоки информации, необходимые при счете задачи) перфокарта "сегментация раздела" имеет вид:

PROGR ,

В этом случае *PROGR* - идентификаторная метка головной программы, на которую передается управление при выходе на счет задачи в системной программе резидент.

Для программ сложной конфигурации с большими массивами обрабатываемой информации перфокарта сегментации имеет более сложный вид (скобочные ссылки) и составление такой перфокарты подробно описано в главе 3.

Г. Массив числовой информации к задаче

Массив перфокарт обрабатываемой информации к задаче - "массив ЧМ" задачи - составляется пользователем, исходя из требований задачи, и в том порядке, которого потребует при вводе информации в ОЗУ транслированная программа при счете по ней.

Д. "Инструкция к задаче" пользователя

К задаче прилагается инструкция, в которой пользователь отмечает требования к оператору:

1. Печать на АЦПУ текста задачи на входном языке.
2. Печать на АЦПУ автокодного текста задачи (результат трансляции с языка Макрос).
3. Печать библиотечной программы (стандартного массива) (результат трансляции с автокода).
4. Перфорация библиотечной программы (стандартного массива) с контролем или без контроля.
5. Выход на счет по транслированной программе.

Таким образом, для прохождения задачи пользователя на ЭВМ (трансляция и счет) должны быть представлены дежурному оператору:

1. "Пакет текста" задачи.
2. "Пакет СМ" задачи.
3. Перфокарта "сегментация разделов".
4. "Массив ЧМ" задачи.
5. Инструкция к задаче.

Примечание: В программе (программах) пользователя должны присутствовать два автокодных оператора *IDENT* и *END* , описывающих программу. Подробнее об этом написано в главе 2 "Описание автокода системы"и в главе 4 "Описание инвариантного языка Макрос".

3. Инструкция для оператора

Задача пользователя состоит из:

"Пакет текста"
"Пакет СМ"
Перфокарта "сегментация разделов"
"Массив ЧМ"
Инструкция к задаче.

Системные программы записаны на МЛ *NO* и считываются с помощью карт:

"Вызов задачи ввода"
"Вызов Макроса"
"Вызов Автокода"
"Вызов загрузчика"
"Вызов распечатки с МБ" .

Имеются три системных массива перфокарт

"Таблица соответствий"
"Резидент (СМ)"
"Служебные карты распечатки с МБ".

Порядок действий при работе с системой.

I. Задача ввода.

Поставить в ЧУ перфокарту "Вызов задачи ввода" и "Пакет текста" пользователя, поставить системную ленту. Нажать "Ввод". После считывания с МЛ и ввода перфокарт - запись пакета текста задачи в телетайпном коде на МБ 0. Команда останова КРА 0707 77 --- Конец работы "Задачи ввода".

2. Коммутация МБ 0 ⟹ МБ I.

3. Трансляция с Макроса.

Поставить в ЧУ перфокарту "Вызов Макроса" и системный массив "Таблица соответствий". Нажать "Ввод". После считывания с МЛ и ввода перфокарт - трансляция с Макроса; автокодный текст переписывается без изменений (по системной карте *ASS); запись порциями по 177_8 кодов с контролем всего автокодного текста задачи на МБ *NO* (физический № I). Команда останова КРА 7704 77 --- ("не проталкиваемый" останов) - конец трансляции.

Примечание: Выход при трансляции на останов КРА 0017 77 --- служит указанием на синтаксическую ошибку в тексте на Макросе. Ошибочный оператор сопровождается в автокодном тексте словом " *error* ". Продолжение трансляции - нажать "Пуск".

4. Печать на АЦПУ автокодного текста задачи.

(По требованию пользователя или при наличии синтаксических ошибок в тексте Макроса).

Поставить в ЧУ перфокарту "Вызов распечатки с МБ" и системный массив "Служебные карты распечатки с МБ". Нажать "Ввод". После считывания с МЛ и ввода перфокарт начинается печать текста на АЦПУ. Выход программы на команду ввода IO 0001 0001 0000 - конец работы программы. Нажать "О МОЗУ".

5. Печать на АЦПУ текста задачи на входном языке (по требованию пользователя).

Коммутация МБI ⟷ МБО.
Выполнить действия пункта 4.
Коммутация МБО ⟷ МБI.

Примечание к п.4 и п.5: Текст задачи на входном языке записан на МБО (физический) в задаче ввода. Автокодный текст задачи записан на МБI (физический) транслятором с Макроса.

6. Трансляция с автокода

Набрать на ДЗУ IУ логическую сумму кодов (по требованию пользователя):

код - - 0100 - признак печати стандартного массива
код - - 0200 - признак перфорации стандартного массива
код - - 2000 - признак контроля перфорации (нажимать клавишу только совместно с признаком перфорации СМ)
код - - 0400 - печать ТРП.

Поставить в ЧУ перфокарту "Вызов автокода".
Нажать "Ввод". После считывания с МЛ начинается трансляция автокодных программ с формированием стандартных массивов. По требованию пользователя (информация на ДЗУ IУ) производится печать и перфорация стандартного массива; если предусмотрен контроль перфорации, то программа выходит на останов КРА 7762 77 - - - , поставить в ЧУ контролируемый массив и нажать "Пуск"; при несовпадении контрольной суммы массив перфорируется заново и т.д.; при совпадении контрольной суммы контролируемого массива на АЦПУ печатается таблица распределения памяти (в "относительных адреса" с ячейки 0020) для сформированного стандартного массива, запись его в библиотеку программ на МБ и затем трансляция всех остальных автокодных программ по этому порядку.

Останов КРА 6317 77 - - - (не "проталкиваемый" останов) - конец трансляции с автокода.

7. Загрузка задачи на счет

Если в процессе трансляции с Макроса были обнаружены синтаксические ошибки, задача снимается со счета после выполнения п.4.

7.1. Поставить в ЧУ перфокарту "Вызов загрузчика", системный массив "Резидент (СМ)"и "Пакет (СМ)" задачи пользователя.
Нажать "Ввод". После считывания с МЛ и ввода перфокарт происходит запись библиотечных программ (СМ) (из "Пакета СМ") в библиотеку программ на МБ и программа загрузчик выходит на останов.

7.2. Поставить в ЧУ перфокарту "Сегментация разделов" с $K\Sigma$.
Нажать "Пуск".
После ввода перфокарт, происходит компиляция разделов, запись их в библиотеку разделов на МБ; печать на АЦПУ информации о загружаемом разделе, пока не будет исчерпана вся информация "Сегментация разделов". Останов КРА 0014 77 - - - конец работы загрузчика.
Режим дозаписи СМ - ДЗУ-4≠0.

7.3. Счет по транслированной программе.

Поставить в ЧУ "Массив ЧМ" задачи пользователя.

Нажать "Пуск".

Происходит счет по программе с вызовом разделов через резидент и, если не предусмотрен останов в программе пользователя, выход на системный останов КРА 0017 77 - - - конец счета.

Таблица системных остановов

Адрес КРА останова	Программа системы	Пояснения
0707	Задача ввода	Конец работы "Задачи ввода".
7704	Транслятор с Макроса	Конец трансляции с Макроса.
0017	- " -	Диагностический останов: указание на синтаксическую ошибку. Нажать "Пуск".
6317	Транслятор с автокода	Конец трансляции с автокода.
1547	Загрузчик	Конец записи в библиотеку программ на МБ.
1546	- " -	Диагностический останов: разбаланс скобок в "сегментации разделов". Задача снимается со счета.
4670	- " -	Диагностический останов: неправильна задана информация "сегментация разделов". Задача снимается со счета.
4726	- " -	Диагностический останов: внутренний *COMMON* больше внешнего. Задача снимается со счета.
4734	- " -	Диагностический останов: неправильная структура каталога программ на МБ. Повторить задачу сначала.
4437	Загрузчик	Диагностический останов. Нет программы в каталоге программ. Сбой машины или неправильно задана информация "Сегментация разделов". Задача снимается со счета.
0014	Загрузчик	Конец загрузки разделов в библиотеку разделов на МБ.
0017	Резидент	Конец счета задачи пользователя.

Примечание: Конец счета задачи пользователя (в Резиденте Останов КРА 0017 77 - - -) только в том случае системный, если головная программа задачи пользователя не имеет предусмотренного останова в программе.

7.3. Счет по транслированной программе.

Поставить в ЧУ "Массив ЧМ" задачи пользователя.

Нажать "Пуск".

Происходит счет по программе с вызовом разделов через резидент и, если не предусмотрен останов в программе пользователя, выход на системный останов КРА 0017 77 - - - конец счета.

Таблица системных остановов

Адрес КРА останова	Программа системы	Пояснения
0707	Задача ввода	Конец работы "Задачи ввода".
7704	Транслятор с Макроса	Конец трансляции с Макроса.
0017	- " -	Диагностический останов: указание на синтаксическую ошибку. Нажать "Пуск".
6317	Транслятор с автокода	Конец трансляции с автокода.
1547	Загрузчик	Конец записи в библиотеку программ на МБ.
1546	- " -	Диагностический останов: разбаланс скобок в "сегментации разделов". Задача снимается со счета.
4670	- " -	Диагностический останов: неправильна задана информация "сегментация разделов". Задача снимается со счета.
4726	- " -	Диагностический останов: внутренний *common* больше внешнего. Задача снимается со счета.
4734	- " -	Диагностический останов: неправильная структура каталога программ на МБ. Повторить задачу сначала.
4437	Загрузчик	Диагностический останов. Нет программы в каталоге программ. Сбой машины или неправильно задана информация "Сегментация разделов". Задача снимается со счета.
0014	Загрузчик	Конец загрузки разделов в библиотеку разделов на МБ.
0017	Резидент	Конец счета задачи пользователя.

Примечание: Конец счета задачи пользователя (в Резиденте Останов КРА 0017 77 - - -) только в том случае системный, если головная программа задачи пользователя не имеет предусмотренного останова в программе.

Приложение I.

Автокодный текст задачи примера 3.

```
TEST:   IDENT;
Z:      BLOCK;
A;      COMMON,1;
B;      COMMON,2;
SUBP:   SUBR;
SUBP1:  SUBR;
MASS1:  BSS,10;
MASS2:  BSS,20;
        50,2500,A,R,COMMENT;
        16,*+1,SUBP,SUBP1+1;;
        77;
LABEL:  BSS,0;
        ,-1,-1;
LABEL1:BSS,1;
        ,,,R1-R
END:    BSS,40;
        END;
TEST1:  IDENT;
X:      COMMON,6;
X1:     COMMON,  ;
A:      EQU,15;
B:      EQU,16;
C:      EQU,A+B-10;
        CALL,PROG;
TEST2:  ENTRY;
        ,D+E+1,D+E+1;
D:      EQU,TEST2+1;
E:      EQU,PROG+TEST  ;
        DATA;
PI:     REAL,31.41592653E-1;
PI1:    REAL,-1.0;
        SET,X,PI,3,2;
TEXT:   7,PROGRAM;
        SET,X1,TEST2,1;
        77;
        END;
A:      IDENT;
C:      SUBR;
Z:      BLOCK;
Z1:     COMMON,2;
R:      52,*,Z1,R1;
        16,*+1,A2,EA2;
        CALL,SIN;
        CALL,B;
        16,*+1,C+1,C;
        77; 56,,R;
A2:     72,,R1;
        150,2500,,100; 470;
EA2:    16;
        END;

C:      IDENT;
Z:      BLOCK;
Z2:     COMMON,2;
R:      56,,*+2;;
        52,Z2,R;
        150,2500,,R1-R; 470;
        CALL,B;
C1:     ENTRY;
        CALL,D;
        CALL,F;
        CALL,SIN;
R1:     56,,R;
        END;
SIN:    IDENT;;
        50,2500,,*+2;
        70,*-2; 56,,*-3;
        END;
B:      IDENT;;
        50,2500,,C;
        70,*-2; 56,,*-3;
C:      1;
        END;
D:      IDENT;;
        50,2500,,C;
        70,*-2; 56,,*-3;
C:      2;
        END;
F:      IDENT;
C1:     SUBR;
R:      BSS,1;
        52,C1,R;
        150,2500,,R1-R,PRINT;
        470;
R1:     56,,R;
        END;
```

Распределение памяти примера 3.

TEST

A	0005	7000
B	0005	7001
SUBP	0005	7002
SUBP1	0005	7003
MASS1	0001	0020
MASS2	0001	0030
TEST	0001	0050
LABEL	0001	0054
LABEL1	0001	0055
END	0001	0057
R	0001	0117
R1	0001	0120

TEST;

X	0005	7000
X1	0005	7001
A	0000	0015
B	0000	0016
C	0000	0023
TEST1	0001	0020
PROG	0005	7002
TEST2	0000	7002
D	0000	7003
TEST	0005	7002
E	0005	7003
R1	0001	0022
R11	0001	0023
TEXT	0001	0022
PROGRAM	0001	0023

A

C	0005	7000
Z1	0005	7001
R	0001	0020
A	0001	0020
SIN	0005	7002
B	0005	7003
A2	0001	0030
EA2	0001	0033
R1	0001	0034

C

Z2	0005	7000
R	0001	0020
C	0001	0020
B	0005	7001
D	0005	7002
F	0005	7003
SIN	0005	7004
R1	0001	0031

SIN

SIN	0001	0020

B

B	0001	0020
C	0001	0024

D

D	0001	0020
C	0001	0024

F

C1	0005	7000
R	0001	0020
F	0001	0021
R1	0001	0024

Печать загрузки
примера 3.
Информация о разделах
имеет вид $a(b,c(d,f))$

```
A       P   0400
C       R   0415    0002
SIN     S   0421
B       R   0425    0001
Z       C   0431

E       P   0433

C       P   0433
C1      E   0441
E       S   0445
D       R   0452    0003
F       R   0456    0004

D       P   0462

F       P   0462

REDRZ   P   0020
```

Печать загрузки
примера 3.
Информация о разделах
имеет вид $a,$

```
A       P   0400
C       S   0415
C1      E   0423
SIN     S   0427
B       S   0433
Z       C   0440
D       S   0442
F       S   0447

REDRZ   P   0020
```

Приложение П.

Таблица соответствий БЭСМ-4.

```
X1>X2;<  ,X1,,X2;<
.IF.X1=X2.GO TO.X3;<15,X1,X2;36,,X3;<
.IF.X1=X2.ELSE.X3;<  15,X1,X2;76,,X3;<
.CALL.X1,X2;<16,**1,X1,X2;<
.GO TO.X1;<56,,X1;<
.SUBR.;<HSS,0;<
.EXIT.;<16;<
X1>.IND.;<72,,X1;<
X1>X2+.IND.;<100,X1,,X2;<
X1+.IND.>X2;<400,X1,,X2;<
X1+X2>X3;<13,X1,X2,X3;<
X1-X2>X3;<33,X1,X2,X3;<
.FORM.X1,X2>X3;<  54,100+X1,X5,X3;54,64,X2,1;
75,1,X3,X3;<
.DESF.X1,X2>X3;<  55,CM  ,X2,X3;54,114,X3,X3;
54,100-X1,X2,X2;<
X1,AND.X2>X3;<  55,X1,X2,X3;<
X1. OR.X2>X3;<  75,X1,X2,X3;<
X1,X2;<X1,X2;<
X1;<,,X1;<
X1=X2+X3;<  1,X2,X3,X1;<
X1=X2-X3;<  2,X2,X3,X1;<
X1=X2*X3;<  5,X2,X3,X1;<
X1=X2/X3;<  4,X2,X3,X1;<
.INT.X1>.REAL.X2;<  75,X1,**2,1;56,,**2;130;21,1,,X2;<
.REAL.X1>.INT.X2;<  61,X1,**2,X2;56,,**2;130;
                    ,,-1;55,X2,*-1,X2;<
,,,X1;<,,,X1;<
.LDESH.X1,X2,X3>X4;<  54,100-X1+X2,X3,1;55,CX2,1,1;
                      54,114,1,X4;54,100+X2,X3,X3;<
.LFORM.X1,X2,X3>X4;<  54,100-X2,X4,X4;
                       54,64+X1-X2,X3,1;75,1,X4,X4;<
.OZU.>X1;<20,4,,X1;<
.STOP.;<77;<
.CLEAN.X1,X2>X3;<52;100,X2,,X3;112,X1,*-1,1;<
.INPUT CARD.X1,X2;<10,X2,*;<
X1=X2*X3;<  ,X2,,BASIC',X3,,GRADE;
            16,**1,STEPN,ESTEPN;,RESULT,,X3;<
.IF.X1.L.X2.GO TO.X3;<  33,X1,X2;36,,X3;<
.IF.X1.L.X2.ELSE.X3;<  33,X1,X2;76,,X3;<
.IF.X1.G.X2.GO TO.X3;<  33,X2,X1;36,,X3;<
.IF.X1.G.X2.ELSE.X3;<  33,X2,X1;76,,X3;<
.IF.X1.LE.X2.GO TO.X3;<  33,X2,X1;76,,X3;<
.IF.X1.LE.X2.ELSE.X3;<  33,X2,X1;36,,X3;<
.IF.X1.GE.X2.GO TO.X3;<  33,X1,X2;76,,X3;<
.IF.X1.GE.X2.ELSE.X3;<  33,X1,X2;36,,X3;<
X1>X2+X3+.IND.;<  100,X1,,X2+X3;<
```

Приложение П (продолжение).

```
X1+X2+.IND.>X3;< 400,X1+X2,,X3;<
.BCIKL.;<52;<
.ECIKL.X1,X2;<112,X1-1,X2+1,1;<
.LFORM.X1,X2,X3,X4>X5'<  ,,64+X1-X2;33,1-1,X3,1;
                         54,13,,1,1;74,1,X4,2;
                         75,X5,2,X5;13,X2,X3,X5;<
.PRINT LINE.)1,X2;< 72,,X1;150,2140,,X2;70,X2;<
.CALL.(X1),()2);< 72,,X2;116,1+3,1+1;72,,X1;250;<
.IF.X1=X2,X3.GO TO.X4'<2,X1,X2,1;3,1,X3;36,,X4;<
X1=X2;<00,X2,,X1;<
.IF.X1=X2,X3.ELSE.X4;<2,X1,X2,1;3,1,X3;76,,X4;<
X1>X2+.IND.+)3;<100,X1,,X2+X3;<
X1+.IND.+X2>>3;<400,X1+X2,,X3;<
.IF.X1.LR.X2.GO TO.X3'<2,X1,X2;36,,X3;<
.IF.X1.LR.X2.ELSE. X3'<2,X1,X2;76,,X3;<
.IF.X1.GR.X2.GO TO.X3'<2,X2,X1;36,,X3;<
.IF.X1.GR.X2.ELSE. X3'<2,X2,X1;76,,X3;<
.IF.X1.LER.X2.GO TO.X3;<2,X2,X1;76,,X3;<
.IF.X1.LER.X2.ELSE. X3;<2,X2,X1;36,,X3;<
.IF.X .GER.X2.GO TO.X3;<2,X1,X2;76,,X3;<
.IF.X1.GER.X2.ELSE. X3;<2,X1,X2;36,,X3;<
X1,X2,X3,X4;<X1,X2,X3,X4;<
,X1,X2,X3;<,X1,X2,X3;<
X1,X2+X3;<X1,X2+X3;<
X1,X2-X3;<X1,X2-X3;<
X1>X2+X3;<,X1,,X2+X3;<
X1>X2-X3;<,X1,,X2-X3;<
X1+X2>X3;<,X1+X2,,X3;<
X1-X2>X3;<,X1-X2,,X3;<
X1>X2+.IND.+.CUB.;<57,500,1,1+2;100,X1,,X2;;<
X1+.IND.+.CUB.>X2;<57,500,100,1+2;400,X1,,X2;;<
,,X1,X2;<,,X1,X2;<
,X1,X2,X3,X4;<,X1,X2,X3,X4;<
```

ЛИТЕРАТУРА

I. В.А.Загинайко.
Компилирующая система II-5923, I97I г.

2. В.А.Загинайко, И.Н.Силин.
"Ассемблер" Б-II-45I4, I968 г.

3. А.М.Волков.
Автокод *Madlen* . Описание транслятора.
Дубна, I970 г. (ОИЯИ, ЛВТА II-5427).

4. В.А.Загинайко.
Инвариантное программирование на ЭВМ М-20, Минск-22, БЭСМ-6.
Препринт ОИЯИ Р-II-3993, I968 г.

Рукопись поступила в издательский отдел
I6 августа I97I года.