ИНСТИТУТ ТОЧНОЙ МЕХАНИКИ И ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ АН СССР

АВТОКОД

БЭСМ-6

Москва – 1969

УДК 681.3.06

Инструкция по "Автокоду БЭСМ-6", транслятор с которого разработан Чайковским М.Г. при участии Назарова Г.В., написана на основе IV главы "Инструкции по программированию на БЭСМ-6". (Автор Чайковский). Исправлены допущенные ранее неточности и неясные места и освещены новые возможности Автокода, возникшие при его усовершенствовании. Увеличилось число примеров.

Автором данной инструкции является к.ф.м.н.
Подшивалов Д.Б.

Корректор: Царицына И.И.
Работа поступила 10/VI-1969 г.
Зак. 580 Т08641

ИТМ и ВТ АН СССР. Москва, В-333, Ленинский проспект, 51

Оглавление

I. Введение

Автокод БЭСМ-6 является одним из средств автоматизации программирования, входящих в состав математического обеспечения машины БЭСМ-6. Он обычно используется в тех случаях, когда необходимо программирование на уровне команд машинного языка. Поэтому его использование предполагает знание программистом системы команд машины и, вообще говоря, системы представления чисел; для этого необходимо знакомство с первыми разделами "Инструкции по программированию на БЭСМ-6". При работе с Автокодом, как правило, программисту не приходится где-либо пользоваться двоичной (или восьмеричной) кодировкой информации: везде, где это возможно, используется метод стандартных перфокарт; исключением является лишь "паспорт задачи", который в силу его специфики не может быть заменен стандартной картой.

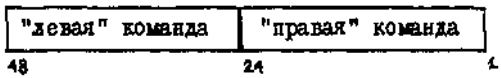
Использование в Автокоде символических обозначений операций и адресов, более естественное представление данных, с которыми работает программа, а также довольно гибкий аппарат отладки, входящий в систему Автокод-БЭСМ-б делает эту систему мощным средством автоматизации программирования практически всех задач, в которых возникает необходимость пользоваться языком машины.

2. Система представления чисел и команд машины БЭСМ-6

Машинное слово БЭСМ-6 содержит 48 двоичных разрядов и может рассматриваться либо как две команды, либо как двоичное число с плавающей запятой, либо просто как набор двоичных разрядов.

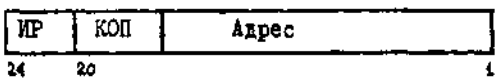
2.1. **Представление команд**

Каждая команда состоит из 24 разрядов. В машинном слове содержится всегда две команды:



Адресуется в машине только слово или левая команда, правая команда не адресуется, и выполнить её, не выполняя её левой команды, невозможно. Поэтому в Автокоде оператор типа "команда" всегда содержит информацию о двух командах, входящих в одно машинное слово.

Каждая команда машины состоит из трех частей:



где:

ИР – номер одного из 16 индексных регистров, используемого, как правило, для формирования исполнительного адреса,

КОП – код выполняемой операции.

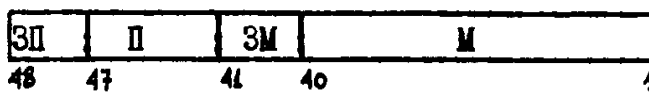
Адрес – часть команды, используемая для указания адреса оперативного запоминающего устройства иди другой информации, требующейся в команде (например, константы, указывающей сдвиг).

Все команды машины разбиваются на две группы в соответствии с размером поля, выделенного под адрес: команды с коротким адресом и команды с полным адресом. В командах с коротким адресом (12 разрядов) можно указать адрес лишь первых или последних 4096 ячеек оперативного запоминающего устройства, в командах с полным адресом можно указать любой адрес ОЗУ.

Такая система адресации требует весьма внимательного подхода к распределению памяти программы и поэтому, в отличие от большинства аналогичных систем программирования, Автокод БЭСМ-6 требует точного определения адресов всех компонент, входящих в программу.

2.2. Представление чисел

Если команда машины воспринимает машинное слово как число с плавающей запятой, то считается, что оно имеет следующую структуру:



где:

М – мантисса числа (40 разрядов),

ЗМ – знак мантиссы (1 разряд),

П – порядок (6 разрядов),

ЗП – знак порядка (1 разряд).

При программировании на Автокоде обычно даже и этих сведений о системе представления чисел бывает вполне достаточно.
Если же задача требует более внимательного подхода к представлению, то их можно получить в "Инструкции по программированию" (стр. 22).

3. Система Автокод БЭСМ-6

3.1. Общие характеристики языка Автокода БЭСМ-6

Язык, используемый в системе Автокод БЭСМ-6 для написания программ, по своей структуре весьма похож на обычный машинный язык, т. е. программа, записанная на этом языке, состоит из последовательности командных слов, констант, рабочих ячеек, рас-

положенных в порядке, нужном программисту. В дальнейшем эти единицы информации будут называться операторами. Как правило, каждому такому оператору будет соответствовать одно слово в программе, полученной после того, как транслятор переведет программу, записанную на языке Автокода, в машинный язык. Такая программа будет называться рабочей. Рабочая программа будет состоять из тех, и только тех команд или ячеек, которые явно заданы в автокод-программе. Такие языки и трансляторы называются "один к одному" и по существу являются аппаратом, позволяющим пользоваться вместо цифровой кодировки команд и чисел символическими обозначениями. Главное достоинство автокода заключается в том, что в качестве ссылки к какой-либо единице информации, занимающей одно слово машины, до можем использовать вместо её цифрового адреса (абсолютного) некий произвольно выбранный программистом символический адрес. Для этого нужно сказать, что такой-то оператор (или ячейку) мы будем называть таким-то символическим именем. Имена всех операторов, используемые в программе, должны быть явно описаны, это означает, что:

1. Данное имя поставлено в соответствие с определенным оператором программы. В этом случае мы говорим, что. оператор имеет метку или что метка является именем данного оператора.
2. Данному имени поставлен в соответствие цифровой (абсолютный) адрес. Это делается с помощью специальных управляющих операторов.

Значение, поставленное в соответствие символическому адресу, можно определить по компонентам, из которых он построен.

Кроме операторов, которые будут переведены транслятором и поставлены в рабочую программу, в Автокод-программе встречаются и операторы, которые управляют работой самого транслятора (управляющие операторы) или всей системы (операторы режима и исправлений).

Операторы каждого из этих типов пишутся по особым правилам и для того, чтобы транслятор мог определить с каким типом оператора он имеет дело в данный момент, используются специальные управляющие (служебные) слова (например, если мы пишем последовательность команд, то перед первой из них должно стоять управляющее слово "К;", после этого все последующие операторы транслятор будет воспринимать как командные до тех пор, пока не найдет другого управляющего слова).

Обычно автокод-программы пишутся на специальных бланках (см. приложение I), и в каждой его строке записывается один оператор. Использование специальных бланков определяется только удобством и наглядностью и совсем необязательно. Кроме разного типа операторов в состав программы могут входить примечания. Они никак не влияют на процесс построения рабочей программы и используются лишь при распечатке программы, т.е. фигурируют в тексте программы, который может отпечатать система Автокода после трансляции.

3.2. Пример программы, написанной на Автокоде

В качестве примера автокод-программы можно привести программу вычисления корней квадратного уравнения $ax^2+bx+c=0$ по формуле:

$$x_{1,2}=\frac{-b\pm\sqrt{b^2-4ac}}{2a}$$

```
В;
Н;        100,          'ОПЕРАТОР_НАЧАЛА_ПРОГРАММЫ_В_ОЗУ'
Ч; ДВА:   -2,           'КОНСТАНТА_МИНУС_ДВА'
   ЧЕТЫРЕ: 0.4_1,       'КОНСТАНТА_ЧЕТЫРЕ'
К;НАЧАЛО:17ПА НАЧ М=, СЧВ=АУ В, 'НАЧАЛО_ВЫЧИСЛЕНИЙ'
         СМ А=АУ С,
         АУ ЧЕТЫРЕ=170В,
         УІ КОМКОР=3050, 'ВЫХОД_К_МЕТКЕ_КОМКОР_ВНЕ_ДАННОГО'
         ЗЧ РЯ=АС В,     'БЛОКА_И_ЭКСТРАКОД_КОРНЯ'
         АЩ А=АД ДВА,
         ЗЧ=СЧ В,
         АВ РЯ=АД А,
         АД ДВА=ЗЧ І,
         3074 =       ,  'ЭКСТРАКОД_КОНЦА_ЗАДАЧИ'
Ч;α1:                 ,  'РЕЗЕРВИРОВАНИЕ_РАБОЧИХ_ЯЧЕЕК'
  α2:                 ,
   А:                 ,
   В:                 ,
   С:                 ,
   РЯ:                ,
Н;НАЧ М: НАЧАЛО,        'ПРОГРАММУ_СЛЕДУЕТ_НАЧАТЬ_ВЫПОЛНЯТЬ'
Е;                      'С_КОМАНДЫ_НАЧАЛО'
```

Мы просим извинения у читателей за нарушение синтаксиса русского языка в примечаниях, но Автокод БЭСМ-6 не допускает употребления в примечаниях большинства знаков препинания. Если вы знакомы с символическими обозначениями команд БЭСМ-6 и их использованием, то, очевидно, командные операторы этой маленькой программы вам уже понятны. Кроме операторов типа команда

(перед ними стоит управляющее слово "К;") в этой программе использованы и некоторые другие типы операторов:

1. Операторы типа числа с плавающей запятой, перед ними стоит управляющее слово "Ч;" Первые два оператора с метками "ДВА" и "ЧЕТЫРЕ" говорят, что в рабочей программе нужно завести две константы, сами операторы представляют собой символическое (общепринятое) представление этих чисел. Остальные операторы этого типа (с метками "А", "В", "С", "x1", " x2", "РЯ") имеют пустое тело оператора и лишь резервируют ячейки программы, необходимые для хранения коэффициентов уравнения, его корней и промежуточных результатов ("РЯ"). Каким образом в эти ячейки могут попасть коэффициенты и как будут выведены результаты, нас пока не интересует.

2. Управляющие операторы, здесь их несколько, все они выполняют различную роль:

а) Оператор начала программы – В;

б) Оператор конца программы – Е;

(по существу эти операторы состоят только из определяющих их служебных слов);

в) Оператор, указывающий, начиная с какого машинного адреса необходимо формировать рабочую программу. Его служебное слово "Н;", сам оператор "100," указывает, что рабочая программа должна помещаться в памяти машины с адреса "00100";

г) Второй оператор этого же типа (после него нет операторов, которые превратятся в слова рабочей программы), указывает, с какой команды ("НАЧАЛО") следует после трансляции начать выполнение рабочей программы.

3.3. Общая схема работы системы

Вся система Автокод может быть схематически разбита на три основных части:

I. Редактор.
2. Транслятор.
3. Интерпретатор.

Первые две части занимаются построением рабочей программы и работают практически всегда, а интерпретатор работает при выполнении рабочей программы в тех случаях, когда программист отлаживает программу и задает "режим отладки" (прокрутки).

Обычно работа с автокод-программой протекает следующим образом:

I. Автокод-программа попадает к редактору и он определяет, нужно ли считывать основной текст программы с магнитной ленты (постановочной), нужно ли после редактирования выдавать текст на перфокарты или же записывать его на магнитную ленту, нужна ли распечатка программы и т.д. Затем проверяется, нужно ли вносить в программу исправления. После выполнения всех этих действий программа готова к трансляции.

Действия, выполняемые редактирующей частью, управляются специальными операторами режима, исправлений и отладки, расположенными перед основной программой.

2. Затем идет трансляция программы, и на барабане готовится рабочая программа.

3. После окончания работы транслирующей части, если не было обнаружено ошибок в Автокод-программе, рабочая программа либо помещается в ОЗУ и начинает уже работать совершенно автономно от системы, либо, если был задан режим отладки, управление передает

ся интерпретирующей части системы, осуществляющей прокрутку с выдачей очень детальной информации о прохождении рабочей программы. Если в Автокод-программе были обнаружены ошибки, то после окончания работы транслятора производится печать информации об ошибках и задача с решения снимается.

Таким образом, полная Автокод-программа может состоять из следующих групп операторов:

1. Оператора, определяющего режим печати системы Автокод БЭСМ-6.
2. Операторов режима, определяющих характер работы всей системы: работы с лентами, перфокартами и режимы отладки программы.
3. Операторов исправлений, позволяющих внести изменения в следующий за ними текст автокод-программы.
4. Собственно программы, состоящей из операторов, управляющих работой транслятора, и операторов, которые после перевода войдут в рабочую программу.

Любая из этих групп операторов может отсутствовать, но их порядок должен обязательно быть таким, как указано.

4. **Операторы и управляющие операторы.**

Операторы – единицы информации, каждая из которых будет переведена в слово рабочей программы. В соответствии с общепринятой структурой программы их можно делить на операторы, представляющие командам, и операторы, представляющие разнообразные виды констант. Операторы, представляющие команды, определяются служебным словом "К;", операторов же, представляющих константы (или рабочие ячейки), есть несколько типов в соот-

ветствии с разными вариантами представления. Есть операторы типа десятичное число (служебное слово – "Ч;"), восьмеричное число ("С;"), буквенно-цифровая константа ("А;" и "Т;"), сложная (логическая) константа ("Л;")

Каждый из операторов этих типов может быть помечен, т.е. перед ним может стоять метка. После каждого из операторов такого типа может стоять примечание.

Метки используются для того, чтобы дать оператору символическое имя, по которому к нему можно ссылаться из других частей программы. Такое символическое имя состоит из последовательности не более чем шести букв иди цифр (называющейся идентификатором), обязательно начинающейся с буквы. Метка всегда пишется перед оператором и отделяется от него знаком Любая метка (кроме специальных-генеральных) имеет ограниченную область действия, это же относится и к любому идентификатору. Это означает, что данное наименование справедливо лишь в определенной области программы.

Примечания помогают лишь лучшему пониманию программы человеком и транслятором не воспринимаются, однако фигурируют в тексте, выдаваемом после трансляции. Примечание состоит из последовательности любых символов, допустимых в языке (см. Приложение 2), кроме символов " , " ; " : " ! " ' "

Примечания должны начинаться и заканчиваться открывающейся кавычкой (поэтому её и нельзя употреблять в тексте примечаний).

Сам оператор можно было бы формально определить как последовательность символов, заканчивающуюся запятой, т.е. окончив обработку текущего оператора и его примечания, транслятор выбирает информацию до тех пор, пока не встретится запятая. Всю

выбранную информацию он рассматривает как оператор (учитывая, конечно, что перед ним может стоять служебное слово и метка).
От этого принципа есть отклонения, но чаще всего процесс проходит именно так.

Далее будут последовательно описаны все типы представления констант, а затем операторы, представляющие команды.

4.1. **Представление констант**

Любой из этих типов операторов может быть помечен и снабжен примечанием.

4.1.I. Оператор типа десятичное число. Перед ним, или перед группой таких операторов, должно стоять служебное слово "Ч;". Сам оператор представляет собой запись числа в общепринятой форме и заканчивается запятой. При записи числа можно употреблять только десятичную точку, но не запятую.

В общем случае число может состоять из:

а) знака числа,

б) целой части,

в) дробной части (перед ней стоит "."

г) порядка, состоящего из:

. признака порядка – символа " $_{10}$ "

.знака порядка

.порядка (целого числа).

Знак числа и порядка (если это "+") можно опускать. Любая из компонент числа или несколько может отсутствовать. Мантисса числа не должна содержать более, чем 13 цифр, а абсолютное значение десятичного порядка не должно быть больше 18. Максимальное. абсолютное значение допустимых в БЭСМ-6 чисел $-0.985*10^{18}$.

Пример.

Перед всеми операторами стоит служебное слово "Ч;", хотя если бы эти операторы шли в программе подряд, то достаточно было бы лишь первого.

Ч; 12.345 ,

Ч; 12 ,

Ч; 345 ,

Ч; 345_{10}^{-3} ,

Ч; 10^{5} ,

4.1.2. Оператор типа восьмеричное число, Служебное слово "С;". Оператор представляет запись содержимого ячейки рабочей программы в восьмеричном виде. Так как ячейка содержит 48 двоичных разрядов, то оператор С-типа состоит не более, чем из 16 восьмеричных цифр. Если указано меньшее количество цифр, то слева приписываются недостающие до шестнадцати нули.

Примеры (с примечанием):

С; АДРЕС: 0000 0000 0000 7777 ,
7777, "ЭТА КОНСТАНТА ЭКВИВАЛЕНТНА ПРЕДЫДУЩЕЙ'

4.1.3. Операторы типа буквенно-цифровая константа. Операторы этого типа позволяют формировать ячейки рабочей программы, содержащие кодированное представление произвольного текста. В системе Автокод БЭСМ-6 принято два типа кодировки – присущий лишь описываемой системе и стандартный. (Таблица кодировок приведена в приложении 2). В каждом из этих представлений любой символ представляется 8 двоичными разрядами, т.е. в слово помещается 6 символов. Тип кодировки указывается специальным управляющим оператором (его служебное слово – "У;"

и вопросы, связанные с использованием того или иного типа, будут разобраны в разделе 4.4.5. В буквенно-цифровых константах запрещено употреблять символы: [illegible]

4.1.3.1. Оператор со служебным словом "А;" позволяет формировать ячейку программы, содержащую кодированное представление ровно шести произвольных допустимых символов. Эти шесть символов и составляют оператор. Оператор заканчивается запятой. Запятую можно в тексте использовать с тем же условием, о котором говорится в следующем разделе.

Примеры.

```
А; ТЕКСТ:    ЭТОТ Т,
             ЕКСТ Б,
             УДЕТ В,
             ВЕДЕН ,
```

4.1.3.2. Оператор со служебным словом "Т;" позволяет писать уже более связный текст, не похожий на предыдущий пример. При этом нет необходимости следить за распределением текста по ячейкам. Транслятор сам будет записывать по шесть символов в последовательные ячейки, пока не встретится пара символов " :х ", являющихся признаком окончания текста. Эта пара символов обязательно должна заканчивать текст, она будет так же включена и в текст как признак "конец текста" (код 140 или 172). Более подробно см. раздел 4.4.5. Как всегда, оператор должен быть окончен запятой. Оператор Т-типа единственный, который может быть превращен в более чем одну ячейку рабочей программы.

С этим оператором связано и еще одно исключение, в тексте оператора можно использовать запятые, они не будут восприняты транслятором как конец оператора, но редактирующая часть будет

рассматривать такой оператор как несколько операторов, и это необходимо учитывать при внесении исправлений (более подробно см. раздел 6).

Пример.

```
Т; ТЕКСТ : ЭТОТ ТЕКСТ БУДЕТ ЗАПИСАН В 7 ЯЧЕЕК :х
```

Если текст не заполняет полностью последнюю из ячеек, то справа, вслед за кодом, соответствующим " :х ", будут поставлены нули.

4.1.4. Оператор типа сложной ("логической") константы. Управляющее слово "Л;". Операторы такого типа предназначены для более сложного, нестандартного способа формирования констант рабочей программы. В них допускается запись в определенные разряды слова закодированной информации, которую программист может записать в более удобном для себя виде. Например, если нужно записать константу, содержащую лишь I в 41 разряде, то, пользуясь оператором С-типа, пришлось писать бы:

```
С; Р4I : 20 0000 0000 ,
```

а с помощью оператора Л-типа это можно записать как:

```
Л; Р41 : ВI/50, ' 50
```
– восьмеричный номер 40 разряда ‘

Оператор Л-типа может состоять из нескольких элементов, элемент от элемента отделяется символом "=", в конце оператора ставится запятая. Сам элемент состоит из признака элемента, содержимого и, если нужно, номера позиции. Существует пять признаков:

1. В – признак последовательности битов (цифр 0 или 1),
2. С – признак восьмеричной последовательности (цифры от 0 до 7),
3. А – буквенно-цифровые символы (не более 6 в выбранной кодировке),
4. Д – длинный символический адрес (15 разрядов),
5. К – короткий символический адрес (12 разрядов).

Содержимым элемента в соответствии с признаками может быть последовательность двоичных разрядов, восьмеричных цифр, буквенно-цифровых символов (длина последовательности в этом случае указывается первой цифрой, стоящей после А), либо значение символического адреса, используемого где-либо в программе. Этот символический адрес должен быть в программе описан, т.е. с ним должен быть сопоставлен некий адрес оперативной памяти, его мы и называем значением. Если стоит признак "Д", то это значение записывается в указанные 15 разрядов слова, а если стоит признак "К", то выделяются младшие 12 разрядов адреса и записываются на соответствующее место. Кроме этого, вы можете еще сами указывать, сколько младших разрядов адреса нужно выделить для включения в константу, для этого после самого символического адреса записывается знак умножения (" x ") и n – восьмеричное число, определяющее число без единицы младших разрядов, включаемых в константу.

Элемент может заканчиваться знаком " = " или номером позиции его младшего разряда. Номер позиции состоит из символа " / " и восьмеричного числа М, где М+1 – номер младшего разряда элемента.

Все элементы в операторе записываются в порядке убывания номеров их младших разрядов. Поля, занимаемые закодированным содержимым элемента, не должны пересекаться. Исключение делается лишь для элементов типа К и Д. В этом случае вы можете "забить" младшие разряды адреса какой-либо другой информацией, например, написав оператор так:

```
Л; ДА/16 < СО/16,  'ЗАНЕСТИ ПОЛНЫЙ АДРЕС А И ОБНУЛИТЬ
                    3 МЛАДШИХ РАЗРЯДА'
```

Г

Знак " / ", написанный после элемента типа К или Д, указывает, что следующий элемент записывается на поле последнего, заменяя часть его информации. Выходить за указанное поле этот элемент не может.

Примеры:
Л; МЛ : Д15/26 = В010/16 = С777 ,
Л; ТЕКСТ : А4 МАМА 20 ,

Если поля, занимаемые соседними элементами, вплотную примыкают друг к другу, то номер младшего разряда можно указывать лишь у последнего.

4.2. **Операторы, представляющие команды**

Перед таким оператором иди группой операторов должно стоять служебное слово "К;". Каждый такой оператор содержит пару команд, которые будут помещены в одно слово рабочей программы. Перед каждым оператором может стоять метка, а после оператора – примечание. Описывая структуру оператора К-типа мы будем использовать понятие символический адрес. Существует довольно много возможных вариантов его написания. Для понимания данного раздела достаточно лишь сказать, что это символическая ссылка к какой-либо точке программы, либо символическая (или более удобная) запись информации, могущей стоять в позиции адреса команды. Точное описание возможных вариантов символического адреса дано в следующем разделе.

В соответствии со структурой слова, представляющего команды, оператор К-типа состоит из двух символических команд – "левой" и "правой" команды, друг от друга они отделены знаком "=", а после правой команды стоит запятая, заканчивающая оператор.

Каждая из команд состоит из трех частей:

а) номера индексного регистра – восьмеричного номера (одна или две цифры),

б) кода операции, состоящего или из двухбуквенного символического обозначения операции или из восьмеричного кода операции, почти совпадающего с машинным. В последнем случае код операции состоит из буквы "Э", за которой идут две или три восьмеричные цифры. (Символические обозначения операции и их восьмеричные эквиваленты приведены в приложении 3.),

в) символического адреса, представляющего ссылку на какой-либо оператор или ячейку.

Если в команде нет нужды указывать какой-либо индексный регистр или символический адрес, то в этих частях команды можно ничего не писать, и в соответствующих полях сформированной команды будут стоять нули. Однако код операции пропускать никогда нельзя, единственным исключением является случай, когда нужно поставить "пустую" правую команду: в этом случае в ней опускаются все составные части и вслед за знаком "=" сразу же идет запятая.

Примеры.

```
К; НАЧАЛО :  01 ПА 7777 = ВИ 1,
Э000 = 3Ч  А,          'ЛЕВАЯ КОМАНДА ПУСТАЯ'
1 ПВ  НАЧАЛО =  , 'ПРАВАЯ КОМАНДА ПУСТАЯ'
```

4.3. Символические адреса

Как уже говорилось, чаще всего в адресной части команды стоит символический адрес, являющийся ссылкой к какому-либо оператору (ячейке) программы (есть еще и символические адреса

особого вида, но о них позже). Эта ссылка может быть либо явной – указывается фактический адрес ячейки, где будет находиться в рабочей программе нужная информация (это уже абсолютный адрес), либо она "строится" на основе какого-либо идентификатора и восьмеричного числа. Идентификатор, используемый в символическом адресе, должен быть описан, т.е. тем или иным способом с ним должен быть сопоставлен абсолютный адрес. Это может быть сделано либо с помощью управляющих операторов, либо путем употребления этого идентификатора в качестве метки какого-либо оператора.

Символический адрес может быть в одной из следующих форм:

а) абсолютный адрес – восьмеричное целое со знаком, состоящее не более чем из пяти цифр. Знак "+" и старшие нули можно опускать. Если указан отрицательный адрес, то адрес представляется в дополнительном коде,

б) идентификатор, описанный где-либо в программе (переменная),

в) переменная с индексом – адрес, состоящий из идентификатора и "смещения" – восьмеричного целого со знаком, заключенного в круглые скобки. Знак "+" можно опускать так же, как и незначащие нули. Число должно содержать не более 5 цифр. Значение такого символического адреса равно сумме значения идентификатера и смещения.

Примеры.

АЛЬФА(+15)

БЕТА(15)

ГАММА(-1017)

г) отрицательный символический адрес – переменная или переменная с индексом, перед которой стоит знак минус. Значение такого символического адреса – представление в дополнительном коде значения данной переменной,

д) символический адрес может быть алгебраической суммой нескольких символических адресов описанных типов а), б), в) и г). Но в этом случае есть некое существенное ограничение: если раньше мы говорили, что идентификатор, входящий в символический адрес, должен быть описан, но не говорили "где" (т.е. он мог быть описан уже после его употребления в адресе), то в этом случае все идентификаторы, входящие в слагаемые суммы, кроме последнего, должны быть уже описаны до такого их использования. Это означает, что их описания должны находиться в тексте ранее места такого их использования.

При вычислении значений символических адресов используется "15-разрядная арифметика". Все значения адресов считаются 15-разрядными двоичными числами без знака. Отрицательные адреса представляются в дополнительном коде. Переносы из старшего разряда "пропадают". Если значение символического адреса должно быть подставлено на место короткого адреса, то это значение не должно выходить за пределы диапазона короткого адреса (первые или последние 4096 ячеек ОЗУ). Если это правило нарушается, то фиксируется ошибка. (Напомним, что на адресацию последних 4096 ячеек указывает один из разрядов кода операции, и если значение символического адреса относится к этому типу, то транслятор сам "наложит" единицу в нужный разряд кода операции.

Примеры.
-АЛЬФА(15) + ВЕТА(5)
714 + ГАММА(9)

Кроме этих пяти типов адресов, допускается еще использование и двух весьма специальных адресов:

а) Константы сдвига – восьмеричного целого со знаком n. (n <=100_8), заключенного между символами "/" и Такой адрес является константой сдвига или константой, аналогичной порядку. Его удобно употреблять в командах СД, КС, КВ. При трансляции он будет заменен на адрес 100 ± n,

б) Списка признаков режимов работы машины, заключенного в круглые скобки. Адрес такого типа употребляют в командах РА, ВР. Список состоит из набора признаков (в любом порядке), объединенных знаком "+". Используются следующие признаки:

О – блокировка округления,

Н – блокировка нормализации,

П – блокировка прерывания по переполнению,

Л – установка признака "логическая группа" ,

У – установка признака "группа умножения" ,

С – установка признака "группа сложения" ,

Примеры.

К; КС/1/ = СД/-17/ , 'КОНСТАНТЫ СДВИГА ЗАДАНЫ СПЕЦИАЛЬНЫМИ'
СД /7/ = РА(Н) , 'АДРЕСАМИ И ПОСТАВЛЕНА БЛОКИРОВКА'
'НОРМАЛИЗАЦИИ'

4.4. Управляющие операторы

Операторы этого типа предназначены для задания транслятору информации, необходимой для правильного перевода автокод- программы в рабочую программу. В основном они связаны с распределением памяти и размещением рабочей программы или

с выбором кодировки буквенно-цифровой информации.

4.4.1. Оператор установки начала рабочей программы. Оператор или группа операторов такого типа определяется служебным словом "Н;". Здесь необходимо сказать несколько слов о самом методе размещения рабочей программы в памяти. В отличие от многих аналогичных систем, выдающих рабочую программу или модуль загрузки, обладающих свойством перемещаемости по памяти, система Автокод БЭСМ-6 жестко настраивает рабочую программу на работу в фиксированной области памяти. Это объясняется тем, что система двух типов адресов не позволяет свободно перемещать программу в памяти, не налагая обременительных ограничений на программирование.

Если нет специальных указаний, то транслятор составляет рабочую программу, начиная с адреса 00010. В трансляторе есть специальный счетчик, указывающий в какое место оперативной памяти следует помещать очередной оператор. В начале трансляции этот счетчик имеет значение 00010. После обработки очередного оператора, попадающего в рабочую программу, счетчик увеличивается на единицу.

Оператор начала программы позволяет программисту управлять размещением своей программы в памяти. Сам оператор содержит только один символический адрес, этот адрес к моменту выполнения оператора должен уже иметь значение. Это значение символического адреса будет присвоено упомянутому счетчику, и идущие следом операторы будут уже располагаться в памяти, начиная с нового значения счетчика. Например, если в программе в начале встретится оператор:

Н; 100,

то рабочая программа будет формироваться, начиная с адреса 00100.

Оператор начала может использоваться для резервирования в программе массивов рабочих ячеек: если программист внимательно следит за абсолютными адресами размещения его программы, то это можно сделать просто передвижением счетчика с помощью оператора начала с абсолютным адресом. Например, так:

```
К;  КОНЕЦ : СА В = ПБ НАЧАЛО ,
Н;  200 ,
К;  НАЧАЛО : СА С = 01 СД/1/ ,
```

Если программист знает, что слово с меткой "КОНЕЦ" будет помещено в ячейку с адресом 00077, то, написав оператор "Н; 0200", он присвоит счетчику значение 00200, т.е. слово с меткой "НАЧАЛО" будет помещено в ячейку о адресом 00200 и в программе останется рабочий массив в 100_8 ячеек. Однако этот способ не всегда удобен хотя бы потому, что необходимо следить за фактическими адресами. Поэтому чаще используется другой способ. Он основывается на том, что оператор начала можно метить. В этом случае идентификатору, выступающему в роли метки, будет присвоено значение, равное содержимому счетчика после обработки предыдущего оператора. В самом операторе начала можно уже поставить символический адрес, построенный на основе этого же идентификатора. Например, ту же часть программы можно записать так:

```
К;  КОНЕЦ : СА В = ПБ НАЧАЛО ,
Н;  МАССИВ : МАССИВ (100) ,
К;  НАЧАЛО : СА С = 01 СД/1/ ,
```

Оператор с меткой "КОНЕЦ” снова будет помещен в ячейку с адресом 00077, а затем счетчик будет передвинут на единицу, и

новое значение (00100) будет присвоено идентификатору "МАССИВ", после этого оператор начала поставит в счетчик значение символического адреса "МАССИВ (100)" т.е. 00200. Достоинством этого метода является то, что нет необходимости следить за абсолютными адресами рабочей программы, кроме того, есть возможность дать резервируемому массиву символическое имя.

В программе может стоять и несколько операторов начала подряд, обычно этим пользуются при резервировании нескольких массивов. Если нам нужно, например, разбить массив с меткой "МАССИВ" на два массива с именами "МАМА" и "ПАПА", то программа может выглядеть так:

```
К;  КОНЕЦ : СА В = ПБ НАЧАЛО ,
Н;  МАМА: МАМА (40) ,
          ПАПА: ПАПА (40) ,
К;  НАЧАЛО : СА С = 01 СД/1/ ,
```

В этом примере использована и еще одна особенность языка Автокод БЭСМ-6: перед оператором может стоять несколько меток. т.е. оператору можно приписать несколько символических имен, и все они будут равноправны.

После оператора начала, как и после других управляющих операторов, может стоять примечание, от других примечаний оно отличается тем, что не войдет в итоговую выдачу транслятора.

Оператор начала выполняет и еще одну функцию в системе: последний встретившийся в программе оператор начала, кроме своих обычных функций, задает еще и адрес, с которого начнется выполнение рабочей программы.

Еще раз подчеркиваем, что все компоненты символического адреса в операторе начала должны быть описаны ранее этого адреса.

4.4.1. **Блоки и глобальные идентификаторы**

Собственно программа, написанная на Автокоде, имеет блочную структуру, аналогичную структуре Алгола. Это означает, что вся программа должна быть блоком. Блоком называется группа операторов, в начале и в конце которой стоят соответственно специальные управляющие операторы, определяемые служебными словами "В;" и "Е;". (Сам блок рассматривается как оператор и может входить в другой блок. Введение блочной структуры позволяет локализовать идентификаторы, описанные внутри блока. Это означает, что если в блоке описан некий идентификатор, т.е. ему придано некое значение, то это значение имеет силу только внутри данного блока. Если мы используем этот идентификатор вне блока, где он описан, то он будет считаться неописанным, и это будет расцениваться как ошибка. Если его описать и во внешнем блоке, то в этих двух блоках один и тот же идентификатор будет идентифицировать разные объекты. В качестве одного из операторов, составляющих блок, может быть использована группа операторов, оформленная как блок, называемый внутренним, а блок в состав которого он входит, называется внешним. Таким образом, во внешнем блоке нельзя использовать идентификаторы, описанные во внутреннем блоке. Описания идентификаторов внешнего блока имеют силу и для внутренних блоков, т.е. там можно использовать идентификаторы, описанные во внешнем блоке.

Если учитывать, что символические имена можно давать и командам, то это означает, что из внешнего блока, например, нельзя перейти в середину внутреннего блока, поскольку метка нужного блока оказывается локализованной, т.е. недействительной во внешнем блоке. Для того чтобы ликвидировать это неудобство,

введено понятие генерального идентификатора, т.е. идентификатора не подлежащего локализации и действительного в любом блоке программы. Идентификатор будет описан как генеральный, если в момент его описания перед идентификатором поставить символ " * " Звездочка ставится только при описании, но не при использовании генерального идентификатора.

Пример.

```
        Б;
К; ПБ НАЧ = , 'ЭТО ССЫЛКА К ГЕНЕРАЛЬНОЙ МЕТКЕ'
        Б;
К; * НАЧ СА С = 1ПА7, 'ЭТО ОПИСАНИЕ ГЕНЕРАЛЬНОЙ МЕТКИ'
        Е;
        ...
        Е;
```

4.4.3. Операторы описаний

Служебное слово "Б;", кроме того, что оно всегда указывает на начало блока, еще говорит, что следующие операторы рассматриваются как операторы описаний. Мы уже говорили об одном из вариантов описания идентификатора, это случай когда идентификатор стоит в качестве метки и ему приписывается значение счетчика. Таким способом можно описать почти все идентификаторы в программе, но если из программы нужно ссылаться к внешним для нее объектам, например, стандартным подпрограммам, то их символические адреса нельзя описать как меткими для присвоения этим символическим адресам некоторого значения удобно использовать операторы описания. Оператор описания состоит из описываемого идентификатора, за которым следует знак "=", и символический адрес, задающий значение описываемого идентифика-

тора. К моменту описания символический адрес (если это не абсолютный адрес) должен иметь значение. Практически это означает, что все его составляющие должны быть описаны "выше" данного оператора описания.

Пример.

```
B;  A = 177 ,
    C = A(1),
    K = C ,
```

Первый оператор описания в программе в качестве описывающей части должен всегда иметь абсолютный адрес. Показанные операторы означают, что идентификатор А будет иметь значение 00177, С – 00200, К – 00200.

В случае последовательного распределения памяти в некоторых операторах описаний описывающая часть может отсутствовать. Такие операторы могут записываться лишь после полного оператора описания.

Пример.

```
B; A = 177 ,
    C,
    K,
```

В этом случае описываемому идентификатору будет приписано значение на единицу больше, чем значение, приписанное в предыдущем операторе. Т.е. идентификатор А получит значение 00177, С – 00200, К – 00201.

Операторы описаний не могут иметь метки, но за ними разрешается писать примечания, которые, однако, не войдут в распечатку.

В операторах описаний можно описывать и глобальные идентификаторы, в этом случае перед идентификатором ставится " * ". Пример.

```
В; * КОП=0017  , 'ЭТО ГЕНЕРАЛЬНЫЙ ИДЕНТИФКАТОР'
```

Так как служебное слово "В;" является не только указанием операторов описаний, но и признаком начала блока, то все операторы описаний располагаются в начале блока.

Любой идентификатор, используемый в программе, должен быть описан либо в описаниях, либо как метка. Если идентификатор в блоке описан дважды как метка, то это фиксируется как ошибка. Если же однако он описан в описании и в качестве метки, то во всех остальных описаниях блока будет использоваться его значение, полученное в операторе описания, а во всей остальной программе значение, полученное им как меткой.

4.4.4. Оператор конца блока. По существу таким оператором является само служебное слово "Е;". Им заканчивается блок и им же заканчивается вся программа (так как она тоже блок). Каждому служебному слову "В;" обязательно соответствует свое служебное слово "Е;".

4.4.5. Оператор режима кодировки. Этот оператор указывает какую кодировку буквенно-цифровой информации должен использовать транслятор при формировании констант А, Т или Л-типа – собственную автокодовскую или стандартную. Перед такими операторами стоит служебное слово "У;" Оператор содержит число, состоящее из одной или двух цифр и заканчивается запятой. Числа, которые могут быть поставлены в операторе, следующие:

0 - указывает, что требуется автокодовская кодировка,

2 – указывает, что требуется стандартная кодировка "со склеиванием",

4 -указывает, что требуется стандартная кодировка "без склеивания".

Перед этой цифрой может стоять цифра 0 или 1. Нуль указывает, что концевой символ текста (":x") кодируется и заносится в Т- константу как последний символ. Единица указывает, что заносить в текст концевой символ не нужно.

Автокодовская и стандартная кодировка буквенно-цифровой информации дана в приложении 2. Добавления "со склеиванием" и "без склеивания" связаны с тем, что в качестве служебных сигналов, употребляемых при вводе и выводе информации, в системе математического обеспечения БЭСМ-6 используются некоторые комбинации двух символов. Такие пары символов обнаруживаются стандартными программами ввода информации и в машину попадают уже как один 8-разрядный код ("со склейкой"), это и заставляет осуществлять такую склейку и в константах Автокода. Однако часто это не всегда бывает удобно, и поэтому введен режим кодировки "без склеивания", в этом случае все служебные комбинации символов кроме " :x " представляются в константах как два символа. Комбинация " :x " всегда кодируется как один символ.

Для более полного ознакомления с вопросами кодировки и ввода информации следует познакомиться с главой VI "Инструкции по программированию". Здесь следует лишь сказать, что при печати буквенно-цифровой информации можно пользоваться либо автокодовской кодировкой, либо стандартной "со склеиванием". Экстракод печати (экстракод Э064) понимает как ту, так и другую (Т=0 или Т=4).

5. **Операторы режима**

Операторы режима управляют общим процессом прохождения задачи через систему Автокод БЭСМ-6. Мы их будем описывать в том порядке, в каком они могут стоять в начале автокод-программы.

5.1. **Операторы запрета печати и обмена**

Если нет специальных указаний, то после окончания трансляции будет выдан текст автокод программы и рабочей программы (с примечаниями или без них, это зависит от оператора отладки). Если выдача программы не нужна (она занимает много времени), то, поставив оператор запрета печати, можно запретить выдачу исходной программы. Такой оператор представляет собою одну только букву "П". Он может стоять только в начале всего вводимого текста. Если в программе обнаружена ошибка, то печать автокод-программы с места обнаружения ошибки начнется вне зависимости от того, стоит "П" или нет.

Так как ввод с перфокарт при больших задачах требует довольно много времени, а при отладке программ этот процесс повторяется часто, то в системе Автокод БЭСМ-6 предусматривается использование, в качестве основного носителя информации о программе, магнитной ленты. На нее можно записать текст автокод-программы, и в дальнейшем, при работе с этой программой, с перфокарт вводить лишь операторы режима и, если это нужно, операторы исправлений. Текстом программы мы здесь называем текст собственно программы, начинающийся со слова "В;" и заканчивающийся словом

Операторы режима и исправлений на ленту не записываются. Их всегда нужно вводить с перфокарт. Кроме этого часто бывает необходимо "документировать" такой текст, находящийся на магнит-

ной ленте. Самый лучший способ сделать это – отперфорировать текст, хранящийся на ленте. Все эти работы могут быть выполнены с помощью оператора обмена. Он состоит из буквы Л, двух восьмеричных цифр и заканчивается запятой. В качестве первой цифры можно использовать такие:

0 – означает, что работа с магнитной лентой не производится;

1 – означает, что текст программы после внесения в него исправлений и после трансляции нужно записать на магнитную ленту;

2 – означает, что текст программы нужно считать с магнитной ленты, внести в него исправления (хотя бы одно) и передать на трансляцию;

3 – означает, что текст нужно считать с ленты, исправить и после трансляции исправленный текст записать на ленту.

Следует обратить внимание, что текст как записывается, так и считывается целиком, и изменить его можно только операторами исправлений. Причем, если даже ваша программа отлажена и исправлять её не нужно, то все равно должен вводиться хотя бы один оператор "пустого" исправления ("И1-1,0,!"). Вторая цифра в операторе обмена определяет порядок выдачи текста с внесенными исправлениями на перфокарты. Если эта цифра нуль, то выдачи на перфокарты производиться не будет. Если она отлична от нуля, то значение определяет, сколько операторов выводить на одну карту. Вот здесь и существенное замечание по поводу использования запятой в буквенно-цифровых константах: редактирующая часть системы считает оператором любую информацию, заканчивающуюся запятой! Вместе с перфорацией идет печать текста с разбивкой на перфокарты.

Пример.

Л 35 ,

Это означает, что текст нужно считать с ленты, внести в него исправления, записать на ленту на место старого текста и выдать на карты по 5 операторов на карту.

Лента, на которую идет запись, является постановочной лентой программиста. Система обращается к ней как к "нулевой" ленте третьего направления (30) и это следует учитывать при написании паспорта задачи. Запись идет, начиная с 1-ой зоны, в 0-ой зоне хранится лишь служебная информация, в её ячейке 0720 стоит число занятых текстом зон этой ленты. Оставшейся частью ленты (свободной от текста) программист может пользоваться по своему усмотрению. На ленту может быть записан текст только одной автокод-программы.

Внимание! Если в программе запрещена печать (оператор "П" есть), то перфорация производиться не будет. Если в программе обнаружены ошибки, то ни записываться, ни перфорироваться программа не будет.

5.2. **Операторы отладки и печати части текста**

В системе Автокод БЭСМ-6 предусмотрен некий аппарат, способствующий отладке программы. Информация для работающего в этом случае интерпретатора задается оператором отладки. Кроме этого, если вам нужно печатать не весь текст автокод-программы, а лишь его часть, то это можно задать специальным оператором печати части программы. В программе может стоять либо оператор отладки, либо оператор печати части программы.

5.2.1 Оператор отладки

Оператор отладки состоит из буквы "О", за которой следует шкала прокрутки. Шкала состоит из двух восьмеричных цифр,

являющихся отображением 4-разрядной двоичной шкалы. Следом за шкалой через запятую стоят указания об интервалах номеров команд, подлежащих прокрутке.

В общем случае оператор отладки имеет такой вид:

O<шкала>, N (A1:A2, B1:B2 .....),
B (A1:A2, B1:B2, ...),
A (A1:A2, B1:B2 .....),
E (A1:A2, B1:B2 .....)!

Буквами "N", "B", "A", "E" указываются типы интервалов прокрутки.

Двоичные разряды шкалы имеют следующие значения (разряды нумеруются справа налево):

I-й разряд, если он равен 1, означает печать информации при выполнении команд передач управления.

II-й разряд, если он равен 1, означает печать информации при выполнении команд, меняющих содержимое каких-либо регистров.

III-й разряд, если он равен 1, означает печать информации при выполнении команд, меняющих содержимое сумматора.

Внимание! Печать, определяемая всеми этими условиями, выполняется, только в том случае, если её "разрешает" IV-й разряд шкалы.

Если он равен 1, то печать информации будет выполняться, если равен нулю, то печати не будет. В начале работы этому разряду (с помощью 0-оператора) может быть присвоено нулевое значение, а в процессе самой прокрутки, при проходе через команды, попадающие в интервал типа "B", ему можно единожды дать значение 1, включив тем самым на все оставшееся время прокрутки печать, заданную шкалой. Кроме этой особенности, печать по шкале

никак не связана с печатью, задаваемой интервалами.

Задавать интервалы команд для отладки можно только указывая их символические адреса, но так как обычные символические адреса не всегда точно определяют место команды (например, в разных блоках есть несколько меток "НАЧАЛО"), то здесь используются полные символические адреса, состоящие из номера соответствующего блока и символического адреса. Номер блока фактически является номером последнего служебного слова "B;", стоящего выше нужной команды. Символический адрес – адрес, который вы использовали бы, например, в программе для команды передачи управления на требуемую команду: это либо метка, либо переменная с индексом (но, очень важно, положительным индексом!), т.е. адресация всегда основывается на вышестоящей метке. Таким образом полный символический адрес имеет вид:

<№ блока>-<символический адрес>

Для разделения используется знак (минус). С помощью этих адресов можно задавать интервалы команд: для этого записывается адрес начала интервала, затем символ и адрес конца интервала.

Интерпретирующая система предусматривает несколько типов работ, выполняющихся при попадании во время интерпретации номеров команд в заданные интервалы:

работа типа N – печать полной информации о выполнении команды,
работа типа B – печать полной информации и установка 1 в IV разряд шкалы, т.е. включение печати по шкале,
работа типа A – интервалы задают исполнительные адреса, если исполнительный адрес команд попадает в такой интервал, то производится

печать информации,

работа типа Е- при попадании команды в этот интервал прокрутка и решение задачи прекращается, и она снимается с решения.

Все интервалы одинакового типа объединяется в одну группу (между собою они отделяется знаком и заключается в круглые скобки. Перед скобками становится буква ("В", "А", или "Е"), определяющая тип работы. Таким образом, в операторе отладки может стоять до 4 групп интервалов, между собою они отделяются знаком ",". Заканчивается оператор знаком "!".

Любая из групп интервалов или все четыре группы могут отсутствовать, но две цифры, задающие шкалу, должны быть.

Примеры:

О 01, N(1 – НАЧАЛО : 1 – М), В(1 – М : 1 – КОНЕЦ)!
О 17!

Последний оператор указывает, что нужно выводить все, что возможно. Если в программе стоит оператор отладки, то распечатка программы будет содержать лишь текст автокод-программы без примечаний и без рабочей программы.

5.2.2. **Оператор печати части программы.**

Если поставить этот оператор (вместо оператора отладки), то будут распечатаны только те операторы программы, которые попадают в указанные в операторе печати интервалы. Эти интервалы задаются точно так же, как в операторе отладки. Печать будет производиться только если она разрешена (нет оператора "П"). Оператор имеет такой вид:

Т, (А1:А2, В1:В2, _____)!
Пример:
Т, (I-M:I-M(I00))!

В этом случае будет распечатано 100_8 операторов программы, начиная с оператора с меткой "М", расположенного после первого служебного слова "В;".

5.3. **Операторы исправлений**

Если в процессе отладки программы возникает необходимость изменить часть текста программы, то это можно сделать либо заменой соответствующих перфокарт, либо воспользоваться операторами исправления. Такими операторами можно исправить только собственно программу, но не операторы режима.

Место внесения исправления указывается полным символическим адресом (таким же, каким пользуются при указании интервалов). Кроме этого в операторе указывается, сколько единиц информации, заканчивающихся запятой, следует "стереть" и какой текст нужно вставить в указанное место. Текст не должен содержать символа "!". Если мы хотим только вставить что-то, не стирая ничего, то нужно указать, что стирается 0 элементов.

Оператор исправления имеет такой вид:

И ПСА, М, ТЕКСТ ВСТАВКИ!

где И – буква "И", начинающая оператор исправления,

ПСА – полный символический адрес,

М – восьмеричное число, указывающее сколько элементов стереть,

! – символ "!", заканчивающий оператор исправления.

Пример.

И1 - М,0. СЧ А = ЗЧ В, !

После метки "М" в первом блоке будет вставлен оператор "СЧ А = ЗЧ В,". После внесения исправления окажется, что меткой "М" помечен вставленный оператор.

```
ИI - М(I), I, СЧ А = ЗЧ В, !
```

Будет стёрт оператор, следующий за оператором с меткой "М", и на его место будет подставлен новый оператор.

Внимание! Если полный символический адрес есть просто идентификатор (метка), то стирание начнется после метки, и сама метка не будет уничтожена. Стираться будут единицы информации заканчивающиеся запятой. Это важно помнить, если вы употребляете запятую в буквенно-цифровых константах.

Естественно, что таким образом нельзя внести исправления в операторы описаний Сони не могут быть помечены). В этом случае пользуются другими модификациями оператора исправления.

Поясним их на примерах:

```
ИI,  В; А, 2, В=А,!
```

Это означает, что после первого служебного слова "В;" нужно найти оператор описания, начинающийся с идентификатора А, стереть два следующих за ним описания и вставить оператор "В=А". Если стирать ничего не нужно, то в соответствующей позиции нужно ставить 0, т.е. оператор будет таким:

```
ИI, В; А, 0, В=А,!
```

Если исправление нужно внести сразу же за служебным словом "В;", то идентификатор ставить не нужно:

```
ИI, В; 0, В=А,!
```

Этот оператор означает, что нужно вставить после первого "В;", ничего не стирая, оператор "В=А",

Если перед блоком автокод-программы стоит несколько операторов исправлений, то первым будет выполнен оператор, ближайший к блоку, вторым – стоящий перед ним и т.д.

6. Ошибки

Транслятор обнаруживает большинство ошибок, связанных с нарушением правил построения операторов и неправильными описаниями. В этих случаях транслятор начинает выдавать программу,

начиная с места, где была зафиксирована первая ошибка. В местах обнаружения ошибок печатается слово "ОШИБКА" и номер этой ошибки. Список номеров и самих ошибок приведен в Приложении 4.

Система построена так, что она пытается обнаружить максимальное число ошибок. Любая из ошибок означает, что после трансляции задача не пойдет на решение и будет снята.

Следует учитывать, что есть такие ошибки, которые фиксируется, но распечатки программы при этом не производится. Например, это происходит, при неверном заголовке исправления.

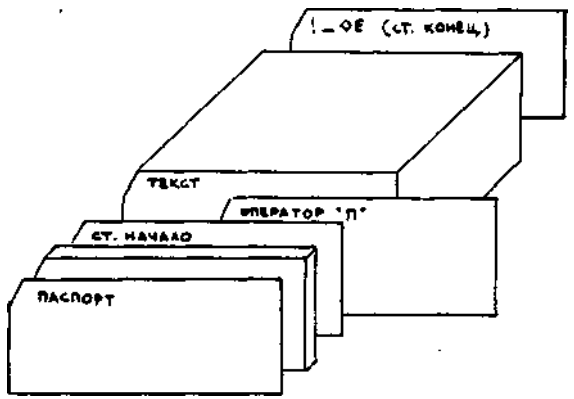
Для того, чтобы получить в этом случае текст программы, следует воспользоваться программой "распечатки перфокарт".

Внимание! Транслятор не может обнаружить все ошибки и поэтому следует очень внимательно следовать данному руководству. Бели вы заметили где-либо здесь места, которые можно толковать по-разному, то сообщите об этом нам, а не пытайтесь "экспериментировать" сами. Это может привести к тому, что через некоторое время в казалось бы полностью отлаженной программе могут обнаружиться ошибки, так как сама система "живет".

7. Ввод программ, написанных на Автокоде

Написанная и отперфорированная на перфокартах программа может быть введена в машину, где она будет обработана системой Автокод БЭСМ-6. Для этого перед и после колоды карт, содержащей операторы режима и собственно рабочие операторы, должны быть поставлены стандартные карты начала (вызова транслятора) и конца, а в самом начале колоды должны стоять карты паспорта. Стандартные карты начала и конца приведены в Приложении 5.

После паспорта задачи и после карты стандартного начала прокладывается по несколько пустых перфокарт.



Более подробно с информацией, входящей в паспорт задачи, можно познакомиться в главе 5 (ст. 261) "Инструкции по программированию". Однако, так как в паспорт задачи при работе с Автокодом входят как информация, характеризующая саму задачу (например, указание о постановочных лентах), так и систему Автокод, то следует дать некоторые добавочные пояснения.

а) Требуемое число листов ОЗУ определяется как максимум из числа листов, требующихся транслятору (14_8), требующихся самой рабочей программе (это вы должны знать) или требующихся для размещения текста автокод-программы (эту величину вы должны оценить, исходя из того, что в ячейке помещается шесть

символов и, кроме того, требуется еще три листа дополнительно!

б) Если ваша программа требует выдачи на ПК, ПЛ, ТТ или если вы указали в операторе обмена такую работу, то это должно быть указано в паспорте.

в) Если вы задали в операторе обмена работу с МЛ, то эта лента должна быть включена в общее число лент, требующихся в программе и, кроме того, должна быть указана как постановочная (номер "30").

г) Число заказываемых трактов МБ определяется как максимум из удвоенного числа трактов (листов), требующихся для размещения программы, числа трактов, заказываемых собственно вашей программой и числа трактов (плюс 1), необходимых для размещения текста. Первые два вы знаете точно, а последние нужно оценить так же, как это делалось при определении требуемого числа листов.

д) В качестве адреса входа в программу записывается адрес обращения к транслятору – 04240. Этот адрес связан со стандартной картой начала и, вообще говоря, может быть изменен, если изменить эту карту.

е) При работе с автокодом всегда заказывают листы, начиная с нулевого.

ж) Шифр задачи должен быть четным и отличным от нуля. Общий вид паспорта задачи при работе с Автокодом приведен в Приложении 6.

7. **Стандартная программа стыковки**

В некоторых случаях появляется необходимость объединить программу, написанную на машинном языке, с программой, написан-

ной на языке Автокода. Это можно сделать с помощью специальной стандартной программы "стыковка". Её номер в библиотеке – 0003. Для работы с этой программой требуется знание общих принципов работы библиотечных программ и умение работать на машинном языке. Это предусматривает довольно подробное знакомство с "Инструкцией по программированию". Вообще говоря, программа стыковки обеспечит вам только размещение в ОЗУ как машинной программы, так и рабочей программы, полученной после трансляции; установление взаимных ссылок является заботой программиста. Управление стыковкой идет из машинной программы, т.е. она уже работает и обращается к стандартной программе.

Процесс стыковки идет следующим образом:

1) Вводится машинная программа и сразу начинает выполняться. В ней есть часть, управляющая стыковкой.

2) Командами, находящимися в этой части, обеспечивается считывание в ОЗУ текста автокод-программы с ленты или перфокарт. (В последнем случае, перед текстом не должно быть паспорта и карты стандартного начала, а должен стоять признак АО).
(Машинную программу и текст автокод-программы можно ввести в память одновременно, тогда этот пункт не выполняется)

3) Затем через экстракод Э066 происходит обращение к программе стыковки.

4) После работы этой программы в памяти машины будет находиться машинная и полученная рабочая программа и управление будет передано на команды, следующие за командами обращения к СП.

При работе с программой стыковки следует учитывать, что текст автокод-программы должен быть записан с начала листа ОЗУ.
Программе стыковки задаются таким параметры:

1) в информационном слове экстракода в 7-12 разрядах указывается номер начального листа машинной программы, а в 1-6 рр. номер послед него листа.

2) в ячейке 0002 таким же образом задается интервал листов, где находится текст автокод-программы. (Если текст автокод-программы введен с устройств ввода, то ячейка 0002 сформирована без вашего участия).

3) в 13-ом разряде первого из указанных слов задается признак выполнения стыковки – 1.

4) Сама программа "стыковки" должна помещаться вне указанных интервалов листов. Программа стыковки может быть включена в работу и другими способами (например, обращение к ней можно вводить как автономную часть), однако, все их перечислить не представляется возможным, и, если вы знакомы с принципами прохождения и ввода "машинных" программ, то вы их "изобретете" легко сами.

Приложение I.

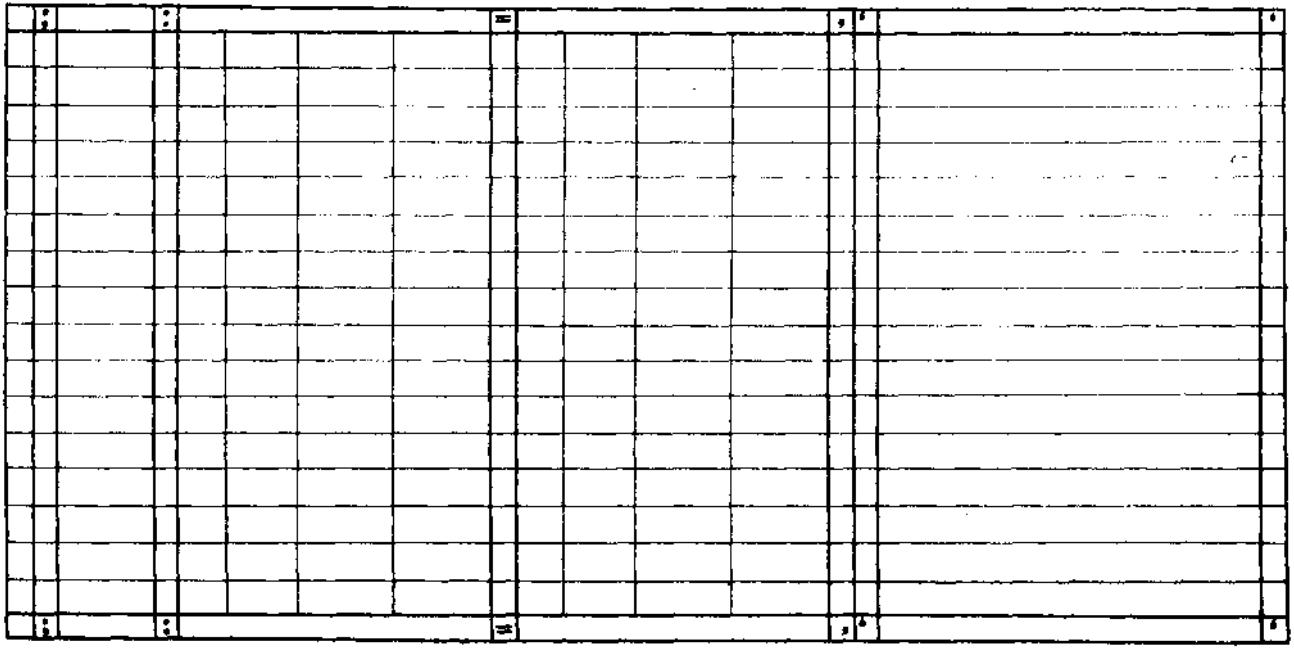
На рисунке приведен вид бланка для написания Автокод-программы. На нем заранее напечатаны некоторые из символов. Если в соответствующих полях бланка нет никаких надписей, то эти символы не перфорируются.

Исполнитель _____

Организация _____

Название работы

Лист _____



буквы – А,Б,В,Г,Д,Е,Ж,З,И,Й,К,Л,М,Н,О,П
Р,С,Т,У,Ф,Х,Ц,Ч,Ш,Щ,Ы,Ь,Э,Ю,Я,
D,F,G,J,I,L,N,Q,R,S,U,V,W,Z

знаки ∨ ∧ ⊃ ¬ + ≡ % ◊ ↓ ¯ _ ! : ≤ ≥
< > + − / , · ∽ ₁₀ ↑ ( ) × = ; [
] ≠ ' ` ≠ — ":x" ":=" ":>" ":<"

Таблица символов и их код.

Символ	код ГОСТ	код АК	Символ	код ГОСТ	код АК
0	000	000	Q	I06	235
I	000I	00I	R	I07	2I2
2	002	002	S	II0	224
3	003	003	U	III	234
4	004	004	V	II2	2I7
5	005	005	W	II3	23I
6	006	006	Z	II4	22I
7	007	007	∨	I20	060
8	0I0	0I0	∧	I2I	07I
9	0II	0II	⊃	I22	055
А	040	230	¬	I23	I45
Б	04I	328	+	I24	065
В	042	228	≡	I25	075
Г	043	3I8	%	I26	062
Д	044	322	◊	I27	042

Символ	код ГОСТ	код АК	Символ	код ГОСТ	код АК	
Е	045	220	¦	130	044	ВЕРТ. ЧЕРТА
Ж	046	317	—	131	150	ТИРЕ
З	047	321	_	132	043	ПОДЧ.
И	050	314	!	133	063	
Й	051	332	¯	115	050	НАДЧ. ^
К	052	236	≤	116	074	<
Л	053	311	≥	117	054	>
М	054	207	:×	172	140	КОНЕЦ ТЕКСТА
Н	055	205	::	173	142	НОМЕР ПОЗИЦИИ
О	056	203	:>	174	173	ЧИСЛО ПОВТОРЕНИЙ
П	057	315	:<	175	073	ПЕРЕВОД СТРОКИ
Р	060	215	:=	177	040	
С	061	216	+	012	061	
Т	062	201	-	013	070	
У	063	225	/	014	067	
Ф	064	326	,	015	046	
Х	065	227	.	016	047	
Ц	066	316	⌴	017	017	
Ч	067	331	ю	020	176	
Ш	070	324	↑	021	143	
Щ	071	301	(	022	076	
Ы	072	325	)	023	051	
Ь	073	327	×	024	072	
Э	074	320	=	025	057	
Ю	075	334	;	026	045	
Я	076	335	[	027	053	
			"	134		
			ъ	135		
			?	136		

Символ	код ГОСТ	код АК	Символ	код ГОСТ	код АК
D	077	222	]	030	066
F	100	226	*	031	170
G	101	213	'	032	064
I	102	214	\	033	041
J	103	232	≠	034	152
L	104	211	<	035	146
M	105	206	>	036	147
			:	037	056

Кодировка по ГОСТ"у указана без восьмого (контрольного) разряда. Автокод-программа перфорируется на устройстве УПП-2, где есть полный набор всех приведенных символов.

Внимание! В автокод-программе нет строчных букв!

Приложение 3.

Список команд машины БЭСМ-6 с символическими и восьмеричными кодами операции, используемыми в Автокоде

Команды с коротким адресом

Операция				Номер параграфа и стр.в "Инструкции по программированию"	
Название	код	группа	Название		
АС	3004	с	арифметическое сложение	2.30	51
АВ	3005	с	арифметическое вычитание	2.31	52
ОВ	3006	с	обратное вычитание	2.32	53
МВ	3007	с	вычитание модулей	2.33	54
ИЗ	3014	с	изменение знака	2.34	55
АУ	3017	у	арифметическое умножение	2.35	56
АД	3016	у	арифметическое деление	2.36	57
СП	3024	у	сложение порядков	2.37	58
ВП	3025	у	вычитание порядков	2.38	59
КС	3034	у	коррекция порядков сложением	2.39	60
КВ	3035	у	коррекция порядка вычитанием	2.40	61
МР	3031	-	выдача младших разрядов	2.41	62
РК	3027	СУЛ	установка режима по коду	2.42	63
РА	3037	СУЛ	установка режима по адресу	2.43	65
ВР	3060	-	выдача режима	2.44	68
ЗЧ	3000	-	запись числа	2.45	69
СЧ	3010	л	считывание числа	2.46	70
СМ	3005	л	считывание магазинное	2.47	70

ЗМ	3001	Л	запись магазинная	2.48	71
ЛУ	3011	Л	логическое умножение	2.49	72
ЛС	3015	Л	логическое сложение	2.50	72
СР	3012	Л	сравнение	2.51	73
ЦС	3013	У	циклическое сложение	2.52	74
СК	3026	Л	сдвиг по коду	2.53	75
СД	3036	Л	сдвиг по адресу	2.54	76
СБ	3020	Л	сборка	2.55	78
РБ	3021	Л	разборка	2.56	79
ВЧ	3022	Л	выдача числа	2.57	80
ВН	3023	Л	выдача № старшей ед.	2.58	81
УИ	3040	-	установка ИР	2.59	83
УМ	3041	Л	установка магазинная	2.60	84
ВИ	3042	Л	выдача ИР	2.61	86
ВМ	3043	Л	выдача магазинная	2.62	87
ПИ	3044	-	передача ИР	2.63	88
СИ	3045	-	сложение ИР	2.64	89

Команды с полным (длинным) адресом.

ПА	324 –	передача адреса в ИР	2.65	90
СА	325 –	сложение адреса с ИР	2.66	91
ИА	322 –	изменение по адресу	2.67	92
ИК	323 –	изменение по коду	2.68	93
УО	326 –	условный переход по "0"	2.69	95
УI	327 –	условный переход по "I"	2.70	96
ПБ	330 –	безусловный переход	2.7I	98
ПВ	33I –	переход с возвратом	2.72	98
ИО	334 –	условный переход по ИР=0	2.73	99
ИI	335 –	условный переход по ИР≠0	2.74	I00
КЦ	337 –	конец цикла	2.75	I0I

Приложение 4

Список ошибок, выявляемых транслятором.

Ниже приводится список ошибок, которые может обнаружить система Автокод БЭСМ-6, и их номера, которые будут выданы при распечатке.

00 – "запрещенная" команда, фиксируемая при прокрутке

1- в адресе использован запрещенный символ

2- в команде использовано неверное название операции

3- число цифр в цифровой константе больше 16_{10}

4- в словаре использован запрещенный знак

5- цифра 8 или 9, или буква стоит в зепрещенном месте

6- в программе не описан данный идентификатор

7- неверный индекс в символическом адресе

10- этот длинный адрес используется в команде 1-ой структуры

11- после левой команды нет знака "="

12- в символическом адресе нет закрывающей скобки

13- неверный индекс в символическом адресе

14- в символическом адресе нет закрывающей скобки

15 – символический адрес, используемый в управляющем операторе, еще не описан.

16 – неопределенность из-за отсутствия служебного слова

17 – элемент описания не содержит идентификатора

20 – ошибка в логической константе

21 – после описаний нет служебного слова

22- использовано не существующее служебное слово

23- после алфавитно-цифровой константы нет знака

24 – в командном слове нет знака ","

25- в данном блоке использованы две одинаковые метки

26- номер индекс-регистра больше $I7_8$

27- неверное название операции

30- неверное название операции

31- неверное название операции

32- длинный адрес в команде 1-ой структуры

33- неверное название операции

34- запрещенный символ в цифровой константе

33- порядок числа больше 18

36 – число "B;" больше числа "E;"

37 – в описании употреблен запрещенный знак

40- в истинном адресе больше 5 восьмеричных цифр

41- ошибка в исправлении

43- ошибка в символическом адресе заголовка исправления

44- ошибка в символическом адресе заголовка исправления

45- ошибка в символическом адресе заголовка исправления

46- ошибка в адресе исправления описания

47- исправление требует стереть слишком много операторов

30- ошибка в названии метки заголовка исправления

31- неверный индекс в символическом адресе заголовка исправления

52- идентификатор начинается с цифры

53- следствие некоторых ошибок, зафиксированных при трансляции

54- помечена меткой правая команда

55- в самом конце автокод-программы нет знака "!"

60 – в операторе прокрутки нет шкалы

71- неверное информационное слово у экстракода

72- неверное задание символических адресов в операторе прокрутки

74 – неверное задание символических адресов в операторе прокрутки

Приложение 5.

Стандартные карты Автокода.

```
Стандартная карта начала  (Вызов транслятора и ввод текста
                           программы)

  B 00 000 4240           адрес входа в задачу трансляции
    00 000 0000

  K I5 240 0000           установка нуля в ИР15
    00 0I0 0003           считывание адреса конца текста в
                          сумматор
  K 00 066 0000           обращение к транслятору (СП 0002)
    00 000 0002
  B 00 000 7000           адрес начала ввода текста
    00 000 0000
 A0                       признак начала текста

      Стандартная карта конца (символьная)

          ! _0E

где !  - признак конца автокод-программы,

   _0  - признак конца текста,

     E - признак конца задачи
```

Внимание! Если вы работаете с текстом, записанным на МП, и во вводимой колоде находятся лишь исправления, то в карте конца символ "!" должен отсутствовать.

Приложение 6.

Паспорт автокод программы:

Восьмеричные цифры, которые вы должны подставить, обозначены буквами "X", "У", "Z".

```
B  00 000 754I
   00 000 0000
C  00 00  0000
   00 00 00XX      число заказываемых листов ОЗУ ≥ I4(8)
C  X0 00 yyz2      X -признак вывода на ПК. УУ-число
                     "постановочных" лент
                   ZZ- общее число лент
   00 00 XXXX        xxxx - число трактов МБ
C 00 00 00XX         xxxx - шифр задачи
  XX 00 4240
C 00 0I 0203       список номеров листов ОЗУ, если
  04 05 0607       вам нужно больше,чем I4(8) листов,
                   то список продолжается еще на
C I0 II I2 I3      две ячейки, иначе они "нулевые"
  00 00 00 00
Ч0
Ч0
C  XX YY ZZ00      список номеров направлений и лент,
   00 00 0000      используемых для обращения к
                     постановочным лентам
E
```

Дальнейшую, более подробную информацию, о паспорте задачи и, в частности, о процессе "сбойного пуска" можно найти на стр. 261 "Инструкции по программированию".