ИНСТРУКЦИЯ

БЭСМ-6

Описание языка

Москва

В.С. Штаркман

Автокод для

АКАДЕМИЯ НАУК СССР

ОРДЕНА ЛЕНИНА ИНСТИТУТ ПРИКЛАДНОЙ МАТЕМАТИКИ

В.С.Штаркман

АВТОКОД ДЛЯ БЭСМ-6

Описание языка

(инструкция)

Май, 1967

ОГЛАВЛЕНИЕ

Стр.

ВВЕДЕНИЕ 3

ОБЩАЯ ХАРАКТЕРИСТИКА АВТОКОДА..... 4

ФОРМАТ ПРЕДЛОЖЕНИЯ НА АВТОКОДЕ..... 7

Бланк..... 7

Элементы предложения на автокоде..... 8

название 10

операция 10

операнд 11

комментарий 12

номер 13

ОБЩИЕ ПРАВИЛА НАПИСАНИЯ ПРЕДЛОЖЕНИЙ АВТОКОДА 14

Набор знаков 14

Кодировка знаков..... 15

Метки 16

определение метки 16

ранее определенные метки..... 18

метки связи программ 18

Счётчик адреса 18

Самоопределенные величины..... 20

Выражения..... 20

выражения простые и составные..... 21

значение выражения..... 23

абсолютные и перемещаемые выражения. 24

ограничения..... 25

НЕКОТОРЫЕ СВЕДЕНИЯ О КОМАНДАХ НА ВНУТРЕННЕМ ЯЗЫКЕ МАШИНЫ..... 26

Формат команд машины БЭСМ-6..... 26

Формирование исполнительн.адреса команды 27

Адресация памяти в коротких командах.. 29

ИЗОБРАЖЕНИЕ МАШИННЫХ КОМАНД на АВТОКОДЕ... 30

Длинные команды..... 30

Короткие команды..... 31

Стр.

Ограничения на перемещаемость программы 33
Ошибки в задании поля операнда..... 39
Машинные команды типа "литерал"..... 40
Расположение машинных команд в памяти.. 35

КОМАНДЫ ТРАНСЛЯТОРА..... 37
Управляющие команды..... 38
КОД - кодировка исходной программы..... 38
СТАРТ - начало программы..... 38
АДРЕС - новое значение счётчика адреса 40
ФИНИШ - конец программы 41
СТРН - начать новую страницу..... 42
СТРОК- пропуск строк на печати..... 42
Определяющие команды 43
ЭКВ - метка..... 43
ПАМ - память..... 44
КОНД - константа длинная 46
целые константы..... 47
число с плавающей запятой (Е)..... 48
константа в форме команды (К)..... 49
адресная константа (А)..... 50
замечание о литеральных константах 51
КОНК - константа короткая..... 51
ТЕКСТ - текстовая константа 51
Команды управляющие базированием..... 53
УПОТР - употреблять регистр для базирования..... 53
ОТМЕН - отменить употребление регистра 54
Команды связи программ 55
ВХОДН - специфицировать входную точку 56
ВНЕШН - установить соответствие с внешней меткой 57
использование свободных переменных.... 58
ПРИЛОЖЕНИЯ 59

ВВЕДЕНИЕ

Описываемый ниже автокод является языком символического программирования для машины БЭСМ-6. Язык дает программисту удобные средства для написания машинных команд распределения памяти и регистров, задания констант и данных. С точки зрения использования всех возможностей машины программирования на автокоде эквивалентно программированию непосредственно в машинных командах. Любая машинная программа может быть легко записана на автокоде и поэтому автокод рассчитан на программистов, хорошо знакомых со структурой и возможностями машины. В некотором смысле автокод является средством "ручного" программирования.

Программу на автокоде будем называть исходной программой. Преобразование исходной программы на внутренний машинный язык выполняется специальной программой - транслятором с автокода. В процессе трансляции транслятор выполняет ряд вспомогательных функций. Одни из них выполняются автоматически, другие - под воздействием специальных команд транслятора, написанных программистом в исходной программе. Строго говоря, транслятор выдает программу, не вполне готовую к исполнению на машине. Окончательная настройка программы производится на стадии загрузки. Загрузка по времени может отстоять как угодно далеко от трансляции и выполняется другой специальной программой - загрузчиком.

ОБЩАЯ ХАРАКТЕРИСТИКА АВТОКОДА

Ниже перечисляются наиболее важные свойства языка и транслятора. Перечисление не включает всех черт и не содержит исчерпывающих пояснений к упомянутым чертам. За всем этим читатель отсылается к последующим разделам.

Мнемонические коды операций: Для кодов операций всех машинных команд имеются мнемонические обозначения. Эти обозначения много легче запомнить и употреблять, чем соответствующие цифровые коды. Например, умножение, имеющее двоичное представление кода операции 001111, на автокоде имеет мнемонику УМН.

Символическая адресация: Различные объекты в программе (команды, константы, массивы данных и т.п.) на автокоде могут иметь символические названия (метки, идентификаторы). Обращения к этим объектам в командах (адресация) может производиться при помощи этих названий, а не действительных машинных адресов.

Широкие возможности задания данных и констант: При задании на автокоде числовых и логических констант человек может выбирать наиболее удобную для каждого конкретного случая форму. В частности, имеется возможность задавать данные в десятичной двоичной, восьмеричной и шестнадцатеричной системе , в виде чисел с плавающей запятой и т.п. Преобразование данных в машинную двоичную форму выполняется транслятором.

Возможность изменения символических названий: Одна метка может быть сопоставлена с другой меткой таким образом, что обе они будут адресовать один и тот же объект в программе (ячейку, индекс-регистр и т.п.). Это позволяет в разных частях программы иметь различные обозначения для одного и того же объекта.

Перемещаемость программ: Выходом транслятора, как правило, является перемещаемая программа, т.е. программа, допускающая на стадии загрузки настройку для работы на любом месте оперативной памяти. Слова „как правило" следует понимать в том смысле, что программисту требуется приложить больше усилий для получения неперемещаемой программы, чем для получения перемещаемой.

Возможности компановки программ: На автокоде предусматривается возможность совместной работы нескольких независимо транслированных программ. Объединение программ и установление соответствия между символическими названиями общих величин (метками связи) выполняется загрузчиком. Настройка передач управления из одной программы в другую является частным случаем обработки меток связи.

Печать результатов трансляции: При каждой трансляции выдается печатный документ, содержащий как исходную программу на автокоде, так и результат трансляции в машинном виде. При желании можно печатать не всю программу, а лишь те места, где обнаружены синтаксические ошибки.

Контроль синтаксических ошибок: При трансляции исходная программа подвергается всесторонней проверке с целью обнаружения ошибок, которые могут возникнуть из-за неправильного использования языка. При любых ошибках трансляция проводится до конца, что позволяет за одну трансляцию получить все ошибки. Сообщение о каждой ошибке появляется в печатном документе.

Отладка программы на символическом уровне: Транслятор помимо машинного вида транслированных команд, констант и других элементов программы выдает некоторую дополнительную информацию о программе, и, в частности, список всех встретившихся в программе меток. Пользуясь этой информацией в дальнейшем, можно будет организовать отладку программы, на символическом уровне, т.е. задавать программе контроля те или иные контрольные операции (напр.,промежуточные выдачи) в терминах меток, а не машинных адресов.

Комментарий: Автокод предоставляет программисту широкие возможности комментировать свою программу. Комментарий не влияет на процесс трансляции, но переносится в печатный документ. Хороший комментарий может в значительной степени выполнить функции описания программы.

ФОРМАТЫ ПРЕДЛОЖЕНИЙ НА АВТОКОДЕ

Исходная программа на автокоде представляет собой последовательность предложений. Каждое предложение задает либо команду машины, либо команду транслятора. По отношению к машинным командам автокод практически является языком „один в один", т.е. трансляция машинной команды на автокоде, порождает одну команду на внутреннем языке машины. О некоторых отклонениях от этого правила будет сказано ниже.

С помощью команд транслятора можно отводить области памяти и давать им символические названия, заводить константы, вводить новые названия, управлять процессом трансляции и выдачей печатного документа о результатах трансляции. Некоторые команды транслятора порождают те или иные объекты в готовой программе (например, заводятся константы), другие - не порождают в готовой программе ничего, а выполняют чисто управляющие функции на стадии трансляции (например, можно дать пробел между строками в печатном документе).

БЛАНК.

Предложения автокода записываются на бланке (см. приложения), а затем пробиваются на перфокарты. В каждой строке бланка может быть записано не более одного предложения автокода. Каждая строка пробивается на отдельной карте. Карты, не содержащие пробивок, игнорируются. Число столбцов на бланке соответствует числу символов, пробиваемых на перфокарте. Столбцы объединены в группы в соответствии с числом символов, располагающихся в одной строке перфокарты. Нумерация групп на бланке соответствует нумерации строк перфокарты. Таким образом, каждая позиция

строки бланка точно соответствуют месту на перфокарте. В верхней части бланка отведено место для названия программы и задания инструкций оператору, перфорирующему карты. Эта информация на карты не попадает.

Примечание I: Как следует из сказанного, исходная программа будет занимать не менее одной перфокарты на каждую команду транслированной программы. Это может показаться расточительством. Тем не менее, удобство работы для человека с исходной программой в такой форме, возможность легко заменять, выбрасывать и добавлять команды с лихвой компенсируют расход карт и затраты времени на их ввод.

Примечание 2: В данном описании изложение ведётся в предположении, что в качестве входного носителя информации используются перфокарты, а в качестве устройства их подготовки - УПП-I. Фактически транслятор будет рассчитан не только на УПП-I, но и на другие устройства и, кроме того, будет допускать использование перфоленты.

ЭЛЕМЕНТЫ ПРЕДЛОЖЕНИЯ НА АВТОКОДЕ

Предложение на автокоде состоит из нескольких элементов (от одного до пяти). Эти элементы располагаются на бланке слева направо и имеют следующие названия: название, операция, операнд, комментарий, номер.

Если перечисленные элементы сопоставить с теми элементами, которые выделяются на бланке для программирования на внутреннем машинном языке, то окажется, что название соответствует адресу ячейки, где располагается команда, операция - коду операции команды, операнд-адресной части самой команды (включая адрес ин-

декс - регистра), и, наконец, комментарий - примечанию. То место, которое занимает элемент на бланке, будем называть полем элемента. На бланке имеется разметка, соответствующая рекомендуемому расположению полей. Поля названия и операции выделены на бланке линиями, которые указывают максимальную длину этих полей. Однако, как будет ясно из дальнейшего, транслятор допускает некоторые отклонения от указанной разметки.

При написании предложений нужно соблюдать следующие общие правила:

I. Единственным обязательным элементом в предложении является операция. Присутствие остальных элементов не обязательно и зависит от операции и желания программиста.

2. Поле названия обязано начинаться в крайней левой позиции бланка. Расположение остальных полей определяется только тем, что они должны быть упорядочены и отделены друг от друга по крайней мере одним пробелом.

3. Поля названия, операции и операнда не должны содержать внутренних пробелов. В поле операнда, как исключение, могут появляться пробелы в качестве символов в текстовых константах.

4. Предложение, начатое в данной строке, не может быть продолжено в следующей.

5. Звёздочка в крайней левой позиции указывает, что вся строка является комментарием.

Из сформулированных правил следует, что в качестве разделителей между полями используются пробелы . Это удобно и не вызывает никаких трудностей при записи предложений автокода на бланке. Для того, чтобы сохранить возможность писать программы без

бланка, допускается другая форма записи, при которой в качестве разделителей между полями используются двоеточие, запятая и точка с запятой.

Детальные сведения о безбланковой форме записи приведены в приложениях.

Название

Поле названия используется для того, чтобы присвоить предложению символическое название - метку. Другие предложения могут ссылаться на данное предложение по этому названию. Если метка присутствует, она должна начинаться с левой позиции на бланке и иметь не более 6 символов. Метка представляет собой последовательность букв и цифр, начинающуюся с буквы. Пробел в крайней левой позиции воспринимается транслятором как отсутствие метки.

Операция

Поле операции служит для того, чтобы задать код операции машинной команды или команды транслятора. Поле операции может содержать любой из допустимых мнемонических кодов операций. Мнемоника кодов операций машинных команд и мнемоника команд транслятора дана в приложениях.

И те и другие команды описываются в соответствующих разделах ниже.

Допустимые мнемонические обозначения никогда не содержат более пяти знаков. Если задана недопустимая операция, транслятор воспримет всё предложение как комментарий и отпечатает сообщение об ошибке.

На рисунке 2 приведен пример мнемонического обозначения команды условного перехода при единичном значении признака резуль-

тата (омеги). Команда имеет метку ПРОБА. Заметим, что мнемонический код операции мог бы располагаться на три позиции левее, поскольку при этом удовлетворялось бы требование о наличии хотя бы одного пробела между полями.

Рис.2. Пример поля метки и операции.

Операнд

Содержимое поля операнда дает транслятору информацию о команде, заданной в поле операции. Если задана машинная команда, операнд определяет то, как будет сформирована адресная часть команды, включая адрес индекс-регистра. Для команд транслятора поле операнда несет ту информацию, которая нужна в данной команде. В зависимости от команды поле операнда может состоять из одного или нескольких подполей, именуемых операндами. Операнды отделяются друг от друга запятыми. В большинстве машинных команд в силу одноадресности машины БЭСМ-6 имеется по одному операнду.

Следует помнить, что пробел сигнализирует об окончании всего поля операнда, и поэтому он не должен попадаться между операндами и запятыми.

В некоторых командах поле операнда может отсутствовать. Отсутствие поля операнда при наличии комментария может быть задано запятой, которую с обоих сторон окружают по крайней мере по одному пробелу (рис.3).

```
        ФИНИШ , ЭТО КОММЕНТАРИЙ
```

Рис.3. Пример отсутствия поля операнда при наличии комментария

Приведем ещё несколько примеров полей операнда. Первая строка на рис.4 содержит ту же команду, что и на рис.2, с адресом перехода равным 69 (десятичная система). Этот же адрес может быть задан в восьмеричной системе и проиндексирован, например, по 5 индекс-регистру (рис.4 - вторая строка). Можно задать адрес перехода, например, равный адресу ячейки, хранящий команду перехода (третья строка), или отличающийся от него на 3 единицы (четвертая строка).

```
ПРОБА   ПЕ   69
ПРОБА   ПЕ   '105'(5)
ПРОБА   ПЕ   ПРОБА
ПРОБА   ПЕ   ПРОБА+3
```

Рис.4. Примеры полей операнда.

Комментарий

Комментарий служит исключительно удобству программиста. Он позволяет в программу вставить строку или раздел описательной информации. Комментарий переносится в печатный документ и не оказывает никакого влияния на транслируемую программу. В комментарии могут встречаться любые допустимые символы, включая пробел.

Поле комментария должно быть правее поля операнда. Всю строку, т.е. все поле предложения, можно использовать для комментария, задав звёздочку в крайней левой позиции.

В случае, когда нужно продолжить строку, являющуюся сплошным комментарием, на новые строки, необходимо ставить звёздочку в левой позиции каждой такой строки.

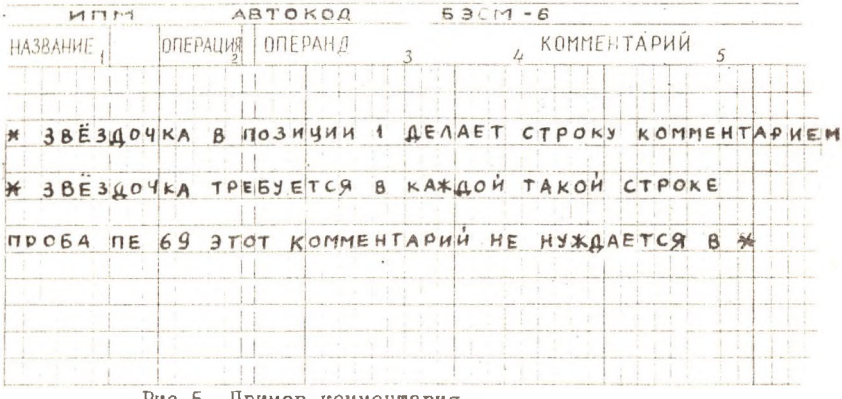


Рис.5. Пример комментария.

Номер

Это поле выделено на бланке для задания информации,идентифицирующей программу, а также для нумерации перфокарт. Для транслятора это поле ничем не отличается от комментария. Оно переносится в печатный документ и не оказывает никакого влияния на транслируемую программу.

ОБЩИЕ ПРАВИЛА НАПИСАНИЯ ПРЕДЛОЖЕНИЙ АВТОКОДА

В этом разделе излагаются общие грамматические правила, которые должны соблюдаться при написании предложений автокода. Вопросы, упомянутые в этом разделе и не нашедшие полного освещения, будут более детально рассмотрены в следующих разделах.

НАБОР ЗНАКОВ

При написании предложений на автокоде можно использовать следующие графические знаки:

I. Буквы русского алфавита заглавные
от А до Я (исключая твёрдый знак).

2. Буквы латинского алфавита заглавные, несовпадающие по начертанию с русскими

D F G I J L N Q R S U V W Y Z

3. Цифры
от 0 до 9

4. Знаки операций и ограничители
* + - / , ( ) ′ . ПРОБЕЛ

Как легко видеть, перечисленные знаки являются подмножеством набора, имеющегося на печатающем устройстве (АЦПУ I28) и устройстве подготовки перфокарт (УПП).

Дополнительные знаки, имеющиеся на УПП и АЦПУ I28, могут использоваться в комментарии и текстовых величинах.

Автокод ориентирован на русский алфавит в том смысле, что в основных символах языка (например, в мнемонических кодах опе-

раций) используются только русские буквы. Латинские буквы могут употребляться в метках, т.е. в тех символических названиях, которые программист дает различным элементам своей программы. Заметим, что с точки зрения пробивки более удобными оказываются метки из русских букв, поскольку все русские буквы и цифры располагаются на одном регистре клавиатуры УПП, тогда как недостающие латинские буквы вынесены на другой регистр.

В трансляторе будет предусмотрена возможность создания чисто латинского „диалекта" автокода, т.е. диалекта, в котором в основные символы языка будут входить только латинские буквы. В этом случае для подготовки карт можно будет использовать зарубежное оборудование. С этой же целью знаки операций и ограничители выбраны в таком составе, какой имеется практически на всех зарубежных клавиатурах и печатающих устройствах.

КОДИРОВКА ЗНАКОВ

При отсутствующей в данный момент унификации способов представления знаков на внешних носителях информации (перфокартах, перфоленте и т.п.),

естественно принять следующее решение.

Предусмотреть возможность воспринимать информацию в различных формах представления и кодировках на внешних носителях, и на входе транслятора перекодировать всю информацию исходной программы в некоторый единый внутренний код. Допускать,чтобы различные части программы имели различное представление на внешнем носителе, и иметь средства с помощью управляющих карт информировать транслятор о форме представления вводимой информации.

Кодировка символов в трансляторе (внутренний код) приведен в приложении . Первый вариант транслятора будет рассчитан на подготовку перфокарт на УПП.

МЕТКИ

Метки [x] создаются программистом и используются им для обозначения областей памяти, команд, устройств ввода - вывода, регистров.

Метка может содержать от одного до шести знаков. В качестве знаков можно использовать буквы и цифры. Начинаться метка должна обязательно с буквы. Нарушение этих правил приведет к тому, что метка воспринята не будет и вызовет печать сообщения об ошибке.

ОПРЕДЕЛЕНИЕ МЕТКИ

Каждая метка в программе должна быть определена. Определяется метка в тот момент, когда она используется в качестве названия какого-либо предложения. В этом случае при трансляции она получает некоторое значение, и это значение используется всякий раз, когда метка встречается в качестве операнда. Если метка нигде не определена, она не может использоваться в качестве операнда. Как правило, значение метки есть ни что иное, как адрес объекта, который описывается предложением, имеющим данную метку в качестве названия.

---

[x] Термин метка в данном описании имеет более широкий смысл, чем в АЛГОЛе. Здесь он является синонимом алгольного термина „идентификатор".

Если предложение задает машинную команду, то это адрес ячейки, где эта команда помещается; если предложение описывает массив памяти, то это адрес первой ячейки массива и т.п. Метка, получившая такое определение, называется перемещаемой или относительной, поскольку её значение должно меняться при перемещении программы в памяти, т.е. при настройке программы для работы на другом месте памяти по сравнению с тем, на которое она рассчитывалась при трансляции.

Заметим, что при трансляции всегда имеется некоторый предположительный начальный адрес программы. Он либо задается программистом, либо транслятор полагает его равным нулю. Счёт ячейкам ведётся от этого предположительного адреса и он определяет значения, которые получают перемещаемые метки. Результат трансляции выдается в такой форме, что для работы программы на предположительных местах памяти ее вообще не нужно настраивать.

Наряду с перемещаемыми метками могут появляться метки абсолютные, т.е. такие, значения которых фиксированы и не изменяются при изменении места программы в памяти. Примерами абсолютных меток могут быть метки, обозначающие индекс-регистры, устройства ввода - вывода, абсолютные адреса памяти. Такие метки могут быть определены с помощью команды транслятора ЭКВ. Каждая метка может быть определена в программе только один раз. Если метка появляется в качестве названия предложения более одного раза, то воспринимается только первое вхождение. Все последующие вхождения игнорируются и сопровождаются печатью сообщения об ошибке.

Ранее определенные метки. В некоторых случаях в поле операнда не будут допускаться метки, определенные где угодно в программе, а лишь метки, определенные ранее. Метка определена ранее, если она попадалась в поле названия некоторого предшествующего предложения. Например, в команде ЭКВ в качестве операндов могут появляться только ранее определенные метки.

Метки связи программ. Обычно метки определяются в той же программе, где они используются. Тем не менее, предусмотрена возможность употреблять в программе метки, значения которых определяются значениями меток из других программ. Эта возможность оказывается полезной всякий раз, когда организуется совместное выполнение нескольких независимо транслированных программ.

Метки,значения которых передаются из программы в программу, называются метками связи.

Метки, на которые распространяются значения чужих ("внешних") меток, вводятся в программе специальной командой транслятора ВНЕШН. Эти метки получают конкретные значения лишь на стадии загрузки.

Метки, которые передают свое значение в другие программы, должны быть особо специфицированы с помощью команды транслятора ВХОДН.

Более детальные сведения о метках связи будут даны в разделе "Команды транслятора".

СЧЁТЧИК АДРЕСА

В трансляторе имеется счётчик, который продвигается по мере отведения ячеек памяти элементам транслируемой программы. Он называется счётчиком адреса, и все время указывает на очередную отводимую программе ячейку. Счётчик увеличивается на

единицу после заполнения очередной ячейки парой машинных команд или константой. Если обрабатывается команда транслятора, которая описывает массив памяти или массив констант, то счётчик подвигается на число ячеек в массиве. Обработка некоторых команд транслятора не продвигает счётчика адреса.

Поскольку команда машины БЭСМ-6 занимает половину ячейки, в счётчике адреса предусмотрен один дробный двоичный разряд. Обработка транслятором одной машинной команды, как правило, продвигает счётчик на единицу этого разряда, т.е. на половину.

Программист может воспользоваться текущим значением счетчика адреса в любом месте программы, написав в качестве операнда звёздочку. При этом будет восприниматься только целая часть счётчика адреса. Таким образом, использование звёздочки в машинной команде эквивалентно употреблению адреса ячейки, где эта команда расположена. Звёздочка как операнд является перемещаемой меткой.

Установление начального значения счётчика адреса есть установление предполагаемого начального адреса программы. Оно может быть выполнено специальной командой транслятора СТАРТ. Программист может изменить текущее значение счётчика адреса командой транслятора АДРЕС (см.описание команд транслятора).

Максимальное значение счётчика адреса (с точностью до целой части) равно 32 767. После этого значения идет 0. Переполнение счётчика, т.е. пересчёт через наибольшее значение, фиксируется и вызывает печать сообщения об ошибке, но не прекращает дальнейшей трансляции.

САМООПРЕДЕЛЕННЫЕ ВЕЛИЧИНЫ

Кроме меток и звёздочки в качестве операндов могут использоваться самоопределённые величины. Самоопределенная величина - это стандартный способ задания произвольного значения. Самоопределенной величине может быть дано символическое название (метка) с помощью команды транслятора ЭКВ. Заметим, что самоопределенные величины имеют такой же „адресный характер", как и метки. С их помощью задаются адреса или данные, появляющиеся непосредственно в адресах, например, параметр сдвига в команде сдвига по адресу.

Транслятор различает самоопределенные величины двух типов: десятичные и восьмеричные.

Десятичная величина есть десятичное целое без знака, не превышающее 32 767. Пример:

7 15 007 4095 10351

Восьмеричная величина есть восьмеричное целое без знака, заключенное в одиночные кавычки. Значение восьмеричного целого не должно превосходить восьмеричного числа 77 777. Примеры:

'7' '375' '002' '37763'

ВЫРАЖЕНИЯ

Понятие метки и самоопределенной величины является частным случаем более общего понятия "выражение". Одиночная метка или одиночная самоопределенная величина представляет собой то, что мы будем называть простым выражением. Арифметическая ком-

бинация простых выражений образует составное выражение. Составное выражение может употребляться в поле операнда машинных команд и команд транслятора и, так как входящие в него простые компоненты, носит „адресный характер".

С помощью составных выражений программист может адресовать команды и массивы данных относительно метки или счётчика адреса. Например, если описан массив, которому дано символическое название А, то второй элемент массива можно адресовать как А+I , третий - как А+2 и т.д.; к ячейке, предшествующей массиву А , можно обратиться, задав выражение А-I .

Каждое составное выражение имеет значение, определяемое значениями входящих компонент и структурой выражения.

Если значение выражения не зависит от положения программы в памяти, то выражение естественно называть абсолютным. Из выражений, значения которых зависят от положения программы в памяти, допускаются лишь перемещаемые выражения. Перемещаемым мы называем выражение такой структуры, что его значение меняется ровно на К единиц при сдвиге программы в памяти на К ячеек.

Любое составное выражение не должно содержать более трёх простых компонент.

Перейдем к более систематическому описанию выражений.

Выражения простые и составные.

Простое выражение это или метка, или звёздочка, обозначающая счётчик адреса, или самоопределенная величина. Примеры:

МАССИВ 3 '25'

* X ALPHA

Составное выражение - это комбинация максимум трех простых выражений, связанных друг с другом знаками арифметических операций. Допустимы следующие знаки операций:

+ (плюс), - (минус), * (звёздочка), / (косая черта).

Они обозначают, соответственно, сложение, вычитание, умножение и целое деление. В выражении не должны подряд стоять два простых выражения или две операции. Если выражение начинается со знака операции, то таким знаком может быть только минус. Таким образом, минус обозначает и двуместную и одноместную операцию. Одиночную метку или самоопределенную величину с предшествующим знаком минус мы относим к составным выражениям. Примеры:

МАССИВ+10	КОНТАБ-НАЧТАБ
X+14*100	ДЛИНА/25
*+КОНТАБ-НАЧТАБ	-25
*-200	АЛЬФА*3+К

Звёздочка используется и для обозначения счётчика адреса и для обозначения операции умножения. Тем не менее, это не может вызвать двусмысленностей, поскольку звёздочка как счётчик адреса, является перемещаемой меткой, и поэтому не может быть компонентой операции умножения. Заметим, что ограничение максимального числа компонент и отсутствие скобок не является существенным, поскольку с помощью команды ЭКВ можно все выражение обозначить одной меткой, и эту метку использовать в других выражениях.

Примеры недопустимых выражений:

МАССИВ+-IO	(две операции рядом)
МАССИВ'25'	(два простых выражения рядом)
+A	(выражение начинается со знака плюс)
3**	(счётчик адреса является компонентой умножения)
ДЛН/5+Р+В	(больше трёх компонент).

Значение выражения

Транслятор вычисляет каждое выражение, находящееся в поле операнда. Концом выражения служит запятая, левая или правая скобка или пробел в зависимости от того, что определяется данным выражением. Процесс вычисления состоит в следующем:

I. Каждой простой компоненте дается её численное значение.

2. Выполняются арифметические действия слева направо, причем умножения и деления до сложений и вычитаний. Например, A+B*C будет вычислено как A+(B*C), а не как (A+B)*C.

3. Результат арифметических действий становится значением выражения.

Заметим, что деление определено как деление нацело, и поэтому оно не перестановочно с умножением. Например, выражение 3/7*IO имеет нулевое значение, тогда как 3*IO/7 равно 4. Деление на нуль ненулевого значения недопустимо. Деление нуля на нуль дает нуль.

Абсолютные и перемещаемые выражения

Выражение является абсолютным, если оно (I) состоит только из абсолютных компонент, или же (2), в него входят ровно две перемещаемые компоненты, причем одна со знаком плюс, а другая со знаком минус. Действительно, хотя значение адресов обеих перемещаемых меток меняется при перемещении программы в памяти, их разность остается постоянной, т.е. абсолютной.

Как уже говорилось, перемещаемым называется выражение, значение которого меняется на К единиц при сдвиге программы в памяти на К ячеек. Причем требуется, чтобы это условие выполнялось только за счёт структуры выражения, а не за счёт конкретных значений входящих в него компонент. Отсюда следуют ограничения:

I. Перемещаемое выражение должно содержать или одну или три перемещаемых метки. Если их три, то одной (и только одной) должен предшествовать минус. Если в выражении одна перемещаемая метка, то ей не должен предшествовать минус.

2. Перемещаемая метка не может быть компонентой в операциях умножения и деления.

Ниже даны примеры абсолютных и перемещаемых выражений (П обозначает перемещаемую метку, А - абсолютную).

Абсолютные выражения:

```
П-П+5
A+I4*25
4095
A*A
A/A
```

Перемещаемые выражения:

П+2

П-8*А

П-П+П

*-177

П-А

Следующие перемещаемые выражения недопустимы

П+П П+П-А	(две перемещаемых метки)
П*А	(умножается перемещаемая метка)
П+П+П	(нет знака минус)
А-П -П	(одна перемечаемая метка с минусом)
П-П-П	(два минуса)

Ограничения

Значение выражения должно лежать в диапазоне от -32768 до +32767. Это ограничение является общим для всех выражений. Если оно нарушено, печатается сообщение об ошибке. Некоторые добавочные ограничения накладываются в зависимости от использования выражения и будут появляться по ходу дальнейшего изложения.

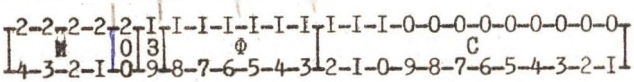
НЕКОТОРЫЕ СВЕДЕНИЯ О КОМАНДАХ НА ВНУТРЕННЕМ ЯЗЫКЕ МАШИНЫ

На автокоде можно изобразить любую машинную команду. Поскольку форма записи команды на автокоде находится в тесной связи с форматом и структурой команды на внутреннем языке машины, полезно напомнить некоторые сведения о машине БЭСМ-6. По ходу изложения вводятся некоторые термины и обозначения, которые используются в дальнейшем.

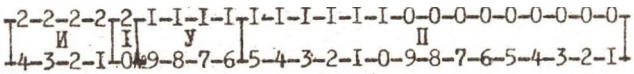
Формат команд машины БЭСМ-6

Команда БЭСМ-6 занимает 24 двоичных разряда. Различаются команды двух структур.

Первая структура:



Вторая структура:



Структура определяется значением 20 разряда:

"0" - первая, "I" - вторая.

На рисунках выделены поля команды, имеющие самостоятельное значение.

И - номер индекс - регистра.

Ф и У - коды операций для команд первой и второй структур, соответственно. Сокращения происходят от слов

Ф - функция,

У - управление,

и их выбор объясняется тем, что во 2-ой структуре собраны в основном управляющие команды.

С - смещение (короткое). Термин "смещение" для этого поля команды, нам кажется, предпочтительнее термина "адрес", поскольку термин "адрес" желательно сохранить за исполнительным адресом.

П - полное смещение.

З - этот разряд команд первой структуры по своим функциям связан с полем смещения С. Если в нем "0",то I2-разрядное поле С расширяется до I5-разрядного приписыванием слева трёх нулей, если же в нем "I", то приписываются три единицы. В дальнейшем этот разряд мы будем называть знаком смещения С.

Несмотря на то, что команды обоих структур имеют одинаковую длину, мы будем для краткости называть команды первой структуры короткими, а команды второй структуры - длинными (по длине смещения).

Формирование исполнительного адреса команды

Везде в дальнейшем термин "адрес" мы закрепим за исполнительным адресом команды, а за полями команды С и П закрепим термин "смещение".

В формировании адреса, как правило, участвуют три компоненты:

I. Смещение

2. Индекс

3. Специальный регистр- модификатор (М).

Для длинных команд смещение - это I5-разрядное двоичное поле П, для коротких - это результат расширения до I5 разрядов I2-разрядного поля С.

Индекс - это I5-разрядный индекс - регистр, номер которого задан в поле И.

Специальный регистр - модификатор М также имеет I5 двоичных разрядов.

Адрес получается сложением значений трех перечисленных компонент. Перенос из старшего двоичного разряда, который может возникнуть в процессе сложения, теряется.

После формирования адреса во всех командах, кроме двух (МОД и МОДА) регистр М сбрасывается в ноль. Команда МОДА посылает в регистр М свой сформированный адрес, а команда МОД посылает в регистр М младшие I5 разрядов операнда, выбранного из памяти. Описанная логика работы регистра М дает возможность использовать команды МОД или МОДА для модификации адреса одной следующей команды.

Единственное исключение из описанного правила формирования адреса составляют семь длинных команд, для которых индекс-регистр, заданный полем И, используется для других целей, и поэтому не входит в число компонент при формировании адреса. Вот эти шесть команд:

УИА	Установка индекса по адресу
СЛИА	Сложение индекса с адресом
ПВ	Переход с возвратом
ПИО	Переход, если индекс равен нулю
ПИНО	Переход, если индекс не равен нулю
ЦИКЛ	Окончание цикла.
УИИ	Установка индекса по индексу.

Адресация памяти в коротких командах

Вполне естественно рассматривать оперативную память "свернутой" в"кольцо", т.е. считать соседними ячейки с наибольшим (32767) и наименьшим (0) адресами. Также вполне естественно использовать в качестве адресов ячеек, наряду с положительными, отрицательные числа, ведя отсчёт в обе стороны от 0. Если стать на такую точку зрения, то окажется, что смещение в коротких командах „перекрывает" диапазон адресов от -4096 до + 4095.

Область памяти с указанными адресами назовем привилегированной, поскольку, пока идет работа в пределах этой области, короткие команды ни в чем не уступают длинным.

Что же касается обращений в коротких командах к остальной памяти, то здесь будут полезными следующие замечания.

В короткой команде можно обратиться к любой ячейке А, если использовать индекс-регистр в качестве базы. Это означает, что в индекс-регистре нужно иметь значение Б, отстоящее от требуемого адреса А на такое расстояние, которое может быть перекрыто смещением (т.е. Б должно лежать в диапазоне от А-4096 до А+4095). Такой метод не вызывает никаких трудностей в тех случаях, когда адрес обращения есть адрес скаляра. Если же обращение производится к элементу массива, и необходимо индексировать адрес номером элемента, то обычно приходится использовать в качестве базы регистр-модификатор М,т.е. затрачивать лишнюю команду МОД или МОДА.

ИЗОБРАЖЕНИЕ МАШИННЫХ КОМАНД НА АВТОКОДЕ

Задать машинную команду - значит задать значения всех её полей. Мнемонический код операции определяет структуру команды (значение 20-го разряда) и в зависимости от структуры задает поля Ф или У . Остальные поля команды определяются информацией, которая размещается на автокоде в поле операнда. Описание этой информации начнем применительно к длинным командам.

Длинные команды.

Поле операнда может быть пустым или содержать максимум два выражения. Значение одного из них заносится в поле П транслированной команды, значение другого - в поле И. Первое выражение назовем адресным, второе - индексным.

Индексное выражение:

(I) Берется в круглые скобки;

(2) Должно быть простым и абсолютным;

(3) Его значение должно быть положительным и не превышать I5.

Примеры:

(3)
(I5)
('I7')
(К)
(ИНДЕКС)

Адресное выражение может быть любым допустимым выражением.

Примеры: X
ИКС+5
*-200/I7
'I777'

Если присутствуют оба выражения, то адресное должно предшествовать индексному.

Если отсутствует одно из выражений (или оба), то соответствующие поля в транслированной команде заполняются нулями. Отрицательное значение адресного выражения кодируется в поле П дополнительным кодом.

Например:

выражение -I эквивалентно 32767 или 77777

выражение -2 эквивалентно 32766 или 77776 и т.д.

Короткие команды

Для коротких команд синтаксис поля операнда выглядит точно так же, как и для длинных. Точно также задается максимум два выражения, из которых одно адресное, а другое - индексное. Основное отличие заключается в том, что для коротких команд не всегда сохраняется полная независимость адресного и индексного выражений. В некоторых случаях задание одного только адресного выражения приведет к заполнению поля И ненулевым значением.

Если адресное выражение задано абсолютным, то все происходит как в длинных командах: адресное выражение определяет значение поля С со знаком, а поле И транслированной команды заполняется значением индексного выражения. Если индексное выражение отсутствует, поле И заполняется нулями. Значение абсолютного адресного выражения должно при этом лежать в диапазоне от -4096 до +4095.

Если задано перемещаемое адресное выражение, то различаются два случая, в зависимости от того, присутствует или отсутствует индексное выражение.

I. Индексное выражение присутствует. В этом случае считается, что адресное выражение должно относиться к привилегированной области памяти, и, следовательно, его значение должно лежать в диапазоне -4096,+4095. Это накладывает определенные ограничения на перемещаемость всей программы, о которых более детально будет сказано ниже.

2. Индексное выражение отсутствует. В этом случае транслятору предоставляется право самому подобрать подходящий индексный регистр, использовать его как базу и вычислить нужное смещение. Для того, чтобы транслятор мог провести такую работу, программист должен предварительно сообщить транслятору содержимое и номера индексных регистров, которые можно использовать под базы. Эта информация сообщается транслятору при помощи команд УПОТР и ОТМЕН, описываемых в разделе "Команды транслятора".

Так или иначе, попытка подобрать базу может оказаться успешной или нет. Если она успешна, то в транслированной команде в поле И вписывается номер подобранного индекс-регистра, а в поле С соответствующее смещение. Если же подходящего индекс-регистра не нашлось, то считается, что адресное выражение может относиться только к привилегированной области памяти и снова возникает ограничение на перемещаемость.

Если при поиске базы окажется, что несколько индекс-регистров подходят для этой цели, то предпочтение будет отдано тому, для которого смещение, во-первых, положительно, во-вторых, минимально по абсолютной величине.

Ограничения на перемещаемость программы

Причиной ограничений на перемещаемость являются короткие команды, в которых

(а) задано перемещаемое адресное выражение, и кроме того,

(в) либо явно присутствует индексное выражение, либо в программе не задано подходящего индекс-регистра для базирования. В таких командах значение адресного выражения не должно выходить за диапазон -4096, +4095. Поскольку значение любого перемещаемого выражения при трансляции определяется предполагаемым начальным адресом, то это вызывает ограничения на задание последнего.

Будем говорить, что программа "некорректна" при данном начальном адресе, если хотя бы в одной команде описанного типа нарушается указанный выше диапазон. Может оказаться, что программа "некорректна" при любом начальном адресе. Может оказаться, что она корректна лишь в определенном диапазоне значений начального адреса. Если трансляция проведена для некоторого значения из этого диапазона, то транслятор выдает вместе с готовой программой информацию обо всем диапазоне возможных значений начального адреса. Этой информацией будет пользоваться загрузчик. В печатном документе о программе отмечаются все команды, порождающие ограничения на перемещаемость.

Ошибки в задании поля операнда

По ходу изложения сообщались различные ограничения, которым должно удовлетворять поле операнда. Повторим основные из них:

I. Индексное выражение должно быть простым абсолютным. Его значение должно лежать в диапазоне от 0 до 15.

2. Значение адресного выражения в длинной команде должно лежать в диапазоне от -32768 до +32767.

3. Значение адресного выражения в короткой команде должно лежать в диапазоне -4096, +4095, если выполнено хотя бы одно из следующих условий:

(I) адресное выражение абсолютное,

(2) задано индексное выражение,

(3) индексное выражение не задано, но и подходящей базы не задано.

При нарушении любого из этих ограничений, а также при синтаксических ошибках в поле операнда печатается сообщение об ошибке.

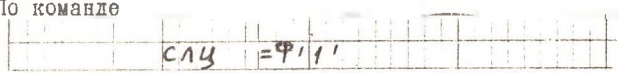
Машинные команды типа „литерал"

В предыдущих разделах мы рассмотрели машинные команды, в которых в поле операнда описывался адрес. На автокоде есть другой сорт машинных команд. Это команды типа литерал. В этих командах в поле операнда вместо адреса описывается константа. Транслятор заведет константу в отдельной ячейке, а в команде сформирует ее адрес. Таким образом, возникает возможность „буквально" в поле операнда задавать информацию. (Литер собственно и означает буквальный)

Перед описанием константы в поле операнда должен стоять знак равенства. Поскольку детальное описание форм задания различных констант будет дано в следующей главе, здесь мы ограничимся несколькими примерами. Любая из команд на автокоде

породит в транслированной программе константу, представляющую собой число с плавающей запятой, равное единице, и команду прибавления этой константы к сумматору.

По команде



будет заведена константа в форме единицы младшего разряда и команда циклического сложения этой константы с сумматором. Все литеральные константы собираются транслятором в один массив в конце программы. Повторяющиеся константы заводятся в единственном экземпляре.

Обращение к заведенной константе в короткой команде может вызвать трудности. Поэтому программист либо должен заботиться о соответствующих индекс-регистрах для базирования, либо учитывать возможные ограничения на перемещаемость программы.

Заметим, что команды типа литерал - это одно из мест, где в автокоде нарушается принцип „один в один" для машинных команд.

Расположение машинных команд в памяти

Как уже говорилось, команда на внутреннем языке машины занимает половину ячейки, и в одной ячейке располагаются две

команды. В то же время адресоваться в БЭСМ-6 можно только к ячейке целиком, и поэтому, например, нельзя передать управление на правую команду в ячейке. Эти особенности машины получили в трансляторе следующее отражение:

I. Транслятор, отводя место в памяти последовательным транслированным командам умеет считать с точностью до половины ячейки. Для этого в счётчике адреса отведен один дробный разряд.

2. Всякую отмеченную (т.е. имеющую название) команду транслятор располагает в левой половине новой ячейки, даже если для этого придется пропустить правую половину предыдущей ячейки.

3. Может случиться, что команды в программе перемежаются числами или константами, которые занимают ячейку целиком. В этом случае также могут возникнуть пропуски правых половин ячеек.

4. Пропущенные правые половины транслятор автоматически заполняет командами, которые "ничего не делают", но и ничего не портят (МОДА с нулями в полях И и П).

Продвижение счётчика адреса, естественно, согласовано с описанными правилами. Каждый раз, когда требуется разместить отмеченную команду или элемент программы, занимающий ячейку целиком, предварительно делается проверка, не настроен ли счётчик адреса на правую половину ячейки. Если он настроен на правую половину ячейки, то эта половина заполняется пустой командой МОДА и счётчик продвигается к началу следующей ячейки. Пропуск правых половин ячеек - это второе и последнее место, где нарушается принцип "один в один" для машинных команд.

КОМАНДЫ ТРАНСЛЯТОРА

В то время как машинные команды используются для задания последовательности действий машины, команды транслятора используются для задания определённых действий, выполняемых транслятором. Некоторые из команд транслятора уже упоминались в предыдущих разделах. В таблице ниже дан список всех команд транслятора с их названиями и мнемоникой. Далее следует их полное описание. Справочные сведения по всем командам транслятора содержатся в приложении.

УПРАВЛЯЮЩИЕ КОМАНДЫ

КСД	Кодировка программы
СТАРТ	Начало программы
ФИНИШ	Конец программы
АДРЕС	Новое значение счётчика адреса
ЛИТЕР	Начало литерных констант
СТРН	Начать новую страницу
СТРОК	Пропуск строк

ОПРЕДЕЛЯЮЩИЕ КОМАНДЫ

ЭКВ	Эквивалентность
ПАМ	Память
КОНД	Константа длинная (в полной ячейке)
ТЕКСТ	Текстовая константа
КОНК	Константа короткая (в половине ячейки)

КОМАНДЫ УПРАВЛЕНИЯ БАЗИРОВАНИЕМ

УПОТР	Употреблять регистр для базирования
ОТМЕН	Отменить употребление регистра

КОМАНДЫ СВЯЗИ ПРОГРАММ

ВХОДН	Специфицировать входную точку
ВНЕШН	Установить соответствие с внешней меткой

Команды транслятора в отличие от машинных команд не всегда вызывают появление команд в транслированной программе. Одни из них (например, ПАМ, КОНД) не порождают команд, а вызывают выделение отдельных областей памяти для данных или констант. Другие (например, ЭКВ, СТРОК) имеют силу только во время трансляции. Они не порождают команд в окончательной программе и не влияют на счётчик адреса.

УПРАВЛЯЮЩИЕ КОМАНДЫ

С помощью управляющих команд указывается форма задания и кодировка исходной программы, задается начало и конец трансляции, устанавливается значение на счётчике адреса, производится управление печатью и т.п. Ни одна из этих команд не порождает команд в готовой программе.

КОД - кодировка исходной программы.

Команда КОД сообщает транслятору, на каком из устройств готовилась исходная программа, и на каком диалекте она написана.Эти сведения располагаются в поле операнда. Точный способ их задания будет уточняться в будущем. Если команда КОД отсутствует, то считается, что программа готовилась на УПП и в коде УПП. Допускается трансляция программы, отдельные части которой имеют различную форму кодировки. Перед каждой частью с новой формой кодировки должна идти команда КОД. Поле названия команды КОД игнорируется.

СТАРТ - Начало программы

Команда СТАРТ используется для указания начала трансляции, для задания названия программы и установки на счётчик адреса предполагаемого начального адреса программы.

Формат команды:

Название	Операция	Операнд
Метка	СТАРТ	Самоопределенная величина

Метка в поле названия становится названием программы. Ей присваивается значение самоопределенной величины, заданной в поле операнда. Название программы можно использовать в других программах как внешнюю метку, специфицировав её в этих других программах командой ВНЕШН. Появление метки в качестве названия программы автоматически специфицирует её как входную точку. Специфицировать её командой ВХОДН не требуется. Если название отсутствует, то программа получит название из шести пробелов. Значение величины, заданной в поле операнда, заносится на счётчик адреса в качестве предполагаемого начального адреса. Если поле операнда пусто, начальный адрес берется равным нулю.

Команда СТАРТ может отсутствовать в программе, тогда в качестве названия будут взяты пробелы, а значение начального адреса будет нулевым.

Команда СТАРТ воспринимается только в том случае, когда она либо первая в программе, либо вторая. Причём, если - вторая, то ей должна предшествовать команда КОД. В любом другом случае команда СТАРТ игнорируется.

Обе приводимые ниже команды устанавливают на счётчик адреса 2040 и присваивают программе название ПРОГ2

```
ПРОГ2    СТАРТ    2040
ПРОГ2    СТАРТ    '3770'
```

АДРЕС – Новое значение счётчика адреса

Команда АДРЕС устанавливает на счётчик адреса значение перемещаемого выражения, заданного в поле операнда; поле названия игнорируется. Таким образом, команда имеет следующий формат:

Название	Операция	Операнд
Игнорируется	АДРЕС	Перемещаемое выражение

Команда может употребляться в программе в любом месте и как угодно часто. Команды АДРЕС, которые заносят на счётчик адреса величину меньшую начального значения, заданного в команде СТАРТ, сопровождаются печатью сообщения об ошибке. Пустое поле операнда в команде АДРЕС эквивалентно отсутствию команды.

Все метки в выражении должны быть определены ранее, в противном случае команда не исполняется и печатается сообщение об ошибке.

Команда

```
        АДРЕС     *+500
```

увеличит текущее значение счётчика адреса на 500. Никакой трансляции для этих ячеек не проводится, т.е. эти ячейки не очищаются транслятором.

Команда АДРЕС представляет одну из возможностей отведения областей памяти, хотя обычно более естественной возможностью считается команда ПАМ (память). Однако, иногда удобно воспользоваться и командой АДРЕС. Например, с помощью следующей программы можно отвести в памяти две одинаковые по размерам области.

```
ТАБЛ1       ПАМ       50
            ПАМ       30
ТАБЛ2       ЭКВ       *
            АДРЕС     * +ТАБЛ2-ТАБЛ1
```

Заметим, что использование метки ТАБЛ2 в команде АДРЕС стало возможным только после её определения в команде ЭКВ.

При описании расположения машинных команд в памяти отмечалось, что транслятор умеет считать „по половинкам ячеек". Это значит, что в каждый конкретный момент трансляции он бывает "настроен" поместить очередной объект транслированной программы либо с начала, либо с середины ячейки.

При программировании следует иметь в виду, что команда АДРЕС не только засылает в счётчик адреса новое значение, но и "настраивает" транслятор на начало ячейки.

Если транслятор до этого был "настроен" на правую половину, то эта правая половина заполняется командой МОДА с нулями в полях И и П.

ФИНИШ - Конец программы.

Командой ФИНИШ останавливается процесс трансляции программы. В поле операнда в ней можно задать точку в программе, в которую нужно передать управление для исполнения программы после загрузки.

Команда имеет следующий формат:

Название	Операция	Операнд
игнорируется	ФИНИШ	Перемещаемое выражение

Обычно точкой, в которую передается управление, является первая машинная команда в программе, как например, в этой группе команд:

```
                СТАРТ    2000
МАССИВ          ПАМ      (100)
ВХОД            УИА      МАССИВ (13)
                 .
                 .
                 .
                ФИНИШ    ВХОД
```

СТРН - начать новую страницу

Команда СТРН заставляет транслятор поместить очередную строку печатного документа с начала новой страницы. Поля названия и операнда игнорируются.[x] Сама команда СТРН печатается до перехода к новой странице. С помощью этой команды, например, можно выделить в печатном документе подпрограммы данной программы.

СТРОК - Пропуск строк на печати.

Команда СТРОК используется для вставления одной или нескольких пустых строк в печатный документ. Команда имеет следующий формат:

Название	Операция	Операнд
игнорируется	СТРОК	десятичное число

Пропускается число строк, заданное в поле операнда, но не более, чем до конца страницы. При пустом поле операнда пропускается одна строка.

---

[x]) Тот факт, что поле игнорируется, т.е. не влияет на ход трансляции, не мешает ему быть напечатанным. Это относится к игнорируемым полям во всех командах.

ОПРЕДЕЛЯЮЩИЕ КОМАНДЫ

Определяющие команды используются, чтобы определить и запасти в программе константы, или отвести области в оперативной памяти. К полям памяти, порожденным этими командами, можно обращаться по символическим названиям. Команда ЭКВ включается в число определяющих команд, поскольку она определяет метку.

ЭКВ - Метка

Команда ЭКВ имеет формат:

Название	Операция	Операнд
Метка	ЭКВ	Выражение

Команда определяет метку, заданную в поле названия, присваивая ей значение выражения, заданного в поле операнда. Выражение может быть перемещаемым или абсолютным, и метка при этом получает соответствующее определение. Все метки в выражении должны быть определены заранее.

Если выражение в поле операнда или метка в поле названия (или оба) отсутствуют или заданы неправильно, команда не воспринимается и печатается сообщение об ошибке.

С помощью команды ЭКВ можно сопоставить метки с номерами индексных регистров, номерами устройств ввода - вывода, с фактическими адресами и другими произвольными величинами.

Например:

```
ВОЗВР     ЭКВ     I3
МАГ       ЭКВ     I5
ТАБЛА     ЭКВ     2000
РАЗМЕР    ЭКВ     I50
```

Для сокращения записи программист может сопоставить метку с некоторым часто встречающимся составным выражением и использовать её как операнд вместо выражения. Например, команда

ПОЛЕ ЭКВ АЛЬФА-БЕТА+ГАММА

определяет метку ПОЛЕ, и её можно использовать вместо выражения АЛЬФА-БЕТА+ГАММА. Подчеркнем, что метки АЛЬФА, БЕТА и ГАММА должны быть определены заранее.

С помощью команды ЭКВ можно метку сопоставить некоторой другой метке с тем, чтобы присвоить одно и тоже значение меткам, использующимся в разных частях программы.

ПАМ - Память

Команда ПАМ используется для того,чтобы зарезервировать некоторую область оперативной памяти и дать этой области название. Команда имеет следующий формат:

Название	Операция	Операнд выражение
Метка	ПАМ	Взятые в скобки самоопределенная величина или ранее определенная абсолютная метка.

Метка в поле названия задает название резервируемой области, самоопределенная величина или метка в поле операнда - число ячеек в этой области. Подчеркнем, что метка, задаваемая в поле операнда, должна быть определена заранее.

Команда ПАМ резервирует область памяти, так сказать, "вместо себя", т.е. в том месте программы, где написана сама команда ПАМ.

Обработка команды ПАМ для транслятора состоит в следующем:

I) Проверяется, не "настроен" ли счётчик адреса на правую половину ячейки. Если да, то эта правая половина заполняется командой МОДА с пустыми полями И и П, а счётчик продвигается к началу следующей ячейки.

2) Значение счётчика адреса берется в качестве значения метки, заданной в поле названия.

3) К счётчику адреса прибавляется значение выражения, заданного в поле операнда.

Командой ПАМ, очевидно, можно воспользоваться для настройки счётчика адреса на ближайшее начало ячейки. Для этого достаточно задать в поле операнда (0).

Заметим, что команда ПАМ резервирует область памяти, но не очищает её. Поэтому программист не должен предполагать, что при загрузке программы в этой области будут нули.

Все остальные определяющие команды транслятора описывают в той или иной форме константы программы. В качестве констант могут задаваться целые числа с запятой фиксированной справа; для таких чисел допускается десятичная, двоичная, восьмеричная и шестнадцатеричная форма представления.

Могут задаваться машинные числа с плавающей запятой в десятичной исходной форме с десятичным порядком или без него.

Константа может быть текстовой, т.е. задаваться в виде последовательности алфавитно-цифровых знаков, и адресной, т.е. задаваться в виде выражения.

Для задания текстовых констант служит команда ТЕКСТ. Число с плавающей запятой может быть задано только командой КОНД; константы остальных типов могут задаваться как командой

КОНД, так и КОНК. Основное отличие между двумя последними командами состоит в том, что первая задает длинную константу (в целой ячейке), а вторая - короткую (в половине ячейки).

КОНД - Константа длинная

Команда имеет следующий формат:

Название	Операция	Операнд
Метка	КОНД	Один операнд, задающий константу в форме, описываемой ниже.

Операнд определяет, во-первых, тип константы, во-вторых, саму константу. В нем может быть указано, сколько раз повторить константу в памяти. Сама константа занимает ровно одну ячейку.

Формат операнда зависит от типа константы. Имеется два основных формата:

(р)Т'с' и Т(с) ,

где:

(р)- коэффициент кратности, представляющий собой самоопределенную величину или абсолютную ранее определенную метку, заключенную в скобки. Если коэффициент опущен, будет изготовлена одна константа.

Т - Код типа константы. Для кодировки типа константы используются следующие буквы:

КОД	ТИП КОНСТАНТЫ
Ф, F	целая десятичная
Ю, U	целая двоичная
В	целая восьмеричная
Х	целая шестнадцатеричная

Е	число с плавающей запятой
А	адресная
К	команда

Для некоторых типов можно с равным успехом использовать любую из двух букв (русскую или латинскую).

'с' - сама константа, заключенная в одиночные кавычки.

Подчеркнем, что для константы типа А выражение, определяющее константу, заключается в скобки.

Последующий текст относится к описанию констант различных типов.

Целые константы.

Целые константы (Ф , F , Ю , U ; В , Х) отличаются лишь системой счисления, в которой задается исходное целое число со знаком. Для любой из констант результат трансляции есть двоичное представление исходного числа, расположенное в 48 разрядах полного машинного слова, которое в этом случае мыслится как целое число. Если задано отрицательное исходное число, то константа формируется в дополнительном коде. Для положительных исходных чисел знак плюс можно опускать.

Для любой из констант потеря значащих разрядов слева вызывает печать сообщения об ошибке.

В шестнадцатеричной константе (Х) цифры больше 9 кодируются первыми буквами латинского алфавита, причем, для тех из них, которые по начертанию не совпадают с русскими, имеются русские эквиваленты.

Цифра	Кодировка
I0	A
II	B
I2	C
I3	D, Д
I4	E
I5	F, Ф

Например, константа, имеющая нули во всех разрядах, кроме восьмого, может быть задана любой из следующих команд:

```
КОНД    Ю'I0000000'
КОНД    В'200'
КОНД    Ф'I28'
КОНД    Х'80'
```

Если же требуется получить все единицы, кроме восьмого разряда, то подходит, например, такая команда:

```
КОНД    В'-20I'
```

ЧИСЛО С ПЛАВАЮЩЕЙ ЗАПЯТОЙ (Е)

Константа, описывающая число с плавающей запятой, задается в виде десятичной мантиссы и десятичного порядка. Порядок может отсутствовать. Константа порождает 48-разрядное нормализованное машинное число с плавающей запятой. Ниже описываются форматы частей константы.

Мантисса: Мантисса представляет собой десятичное число со знаком, в котором может присутствовать, а может отсутствовать десятичная точка. Десятичная точка может располагаться в начале, середине или конце числа. Точка в конце числа может

опускаться. Если опущен знак, то мантисса считается положительной. Абсолютная величина мантиссы, отвлекаясь от положения запятой, не должна превосходить +I 099 5II 627 775.

Порядок: Порядок может отсутствовать, если десятичная точка в мантиссе стоит в нужном месте. Если порядок задан, то он располагается правее мантиссы и состоит из буквы E, сопровождаемой десятичным целым число со знаком. Число задает степень десятки, на которую должна быть домножена мантисса. Порядок считается положительным, если знак опущен.

Например, для задания числа 3.I4I5 в форме с плавающей запятой подходит любая из следующих команд:

```
КОНД    E'3.I4I5'
КОНД    E'3I4I5E-4'
КОНД    E'+3I4I5.E-4'
КОНД    E'.3I4I5EI'
КОНД    E'.03I4I5E+2'
```

Если мантисса лежит вне указанного выше диапазона или все число выходит за диапазон представимых в машине нормализованных чисел, то печатается сообщение об ошибке.

Константа в форме команды (К)

Константа этого типа представляет собой запись команды в виде последовательности

одной двоичной цифры
одной восьмеричной цифры
одной четверичной цифры
шести восьмеричных цифр девяти ?

Эта форма используется в ряде документов о машине БЭСМ-6. Например, команда из всех единиц может быть записана в виде

```
КОНД      К'I73777777'
```

Команда располагается в правой половине ячейки; левая заполняется нулями.

Адресная константа (А)

Адресная константа задается в виде перемещаемого или абсолютного выражения, заключенного в скобки, а не в одиночные кавычки. Значение выражения порождает константу. Это значение, будучи целым числом, располагается в ячейке как целое число, т.е. разрядом единиц является крайний правый разряд. Если выражение имеет отрицательное значение, то константа формируется в дополнительном коде. Поскольку чаще всего выражение является адресом памяти, константы этого типа названы адресным, хотя, может быть, более точным было бы название „константы-выражения". Заметим, что в тех случаях, когда константа порождается перемещаемым выражением, её значение меняется при настройке программы загрузчиком. В константе типа А в отличие от предыдущих констант нельзя задавать коэффициент повторения.

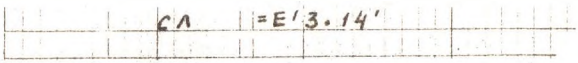
В следующей программе, приводимой в качестве примера, в первой ячейке будет сформирован начальный адрес программы

```
ПРОГI     СТАРТ
          КОНД     А(*)
            .
            .
            .
          КОНД  А (ВХОД+1(5))
```

можно задавать кроме адресного, еще и индексное выражение

Замечание о литеральных константах

В машинных командах типа литерал используется абсолютно та же форма задания типа константы и самой константы, что и в команде КОНД. Константа любого типа, допустимого в команде КОНД, может использоваться в качестве литеральной. В команде типа литерал нельзя задавать коэффициент кратности и описанию константы должен предшествовать знак „равно", например:



КОНК – константа короткая

Команда транслятора КОНК отличается от команды КОНД следующим:

I. Сформированная константа помещается в очередную половину ячейки, а не в целую ячейку.

2. Нельзя задавать коэффициента кратности.

3. Нельзя задавать константу типа E.

В остальном эти команды совпадают. Формы заданий самих констант абсолютно идентичны. Окончательное 24-х разрядное значение константы получается откидыванием левой половины константы, формируемой точно так же, как в команде КОНД. Если при отбрасывании левой половины теряются значащие разряды, то печатается сообщение об ошибке.

ТЕКСТ – Текстовая константа

Команда ТЕКСТ, с помощью которой задаются последовательности буквенно-цифровых символов, имеет следующий формат:

Название	Операция	Операнд
Метка	ТЕКСТ	Операнд, описывающий константу

Операнд задается в следующей форме:

Т'с'

где

Т - буква Т - код типа константы

'с' - сам текст, т.е. взятая в одиночные кавычки последовательность допустимых символов.

Допустимыми являются различные наборы символов в зависимости от устройства, на котором готовится (перфорируется) программа. При вводе все символы перекодируются во внутренний код транслятора. Наборы символов различных устройств и их соответствие внутреннему коду даются в Приложениях.

Во внутреннем коде на каждый знак (символ) отводится по 8 двоичных разрядов. Символы располагаются по 6 штук в ячейке слева направо. Под текстовую константу всегда отводится целое число ячеек. Если число символов в тексте не кратно 6, то свободное место в последней ячейке заполняется пробелами. Максимальная длина последовательности символов ограничивается только размерами строки бланка.

Подчеркнем, что пробел является допустимым символом. Например, следующая команда

ТЕКСТ Т'ВЫДАЧА НА АВТОКОДЕ'

породит цепочку из 18 символов, занимающих ровно три ячейки.

Одиночная кавычка, используемая как буквенно-цифровой символ, в тексте может быть задана с помощью двух одиночных кавычек. Например, в команде

ТЕКСТ Т'ОБ''ЕКТ'

задана последовательность из шести символов, которая уложится в одну ячейку.

КОМАНДЫ, УПРАВЛЯЮЩИЕ БАЗИРОВАНИЕМ

Выше при описании машинных команд короткой структуры говорилось о том, что транслятор в некоторых случаях может сам выбрать подходящий индекс-регистр в качестве базы и вычислить нужное смещение. Для того, чтобы воспользоваться этими возможностями, программист должен сообщать транслятору определенную информацию с помощью команд УПОТР и ОТМЕН.

УПОТР - Употреблять регистр для базирования

В команде УПОТР указывается индекс-регистр, который можно использовать для базирования, и сообщается транслятору значение того базового адреса, который будет находиться в регистре в момент исполнения программы. Подчеркнем, что команда УПОТР не загружает регистр. Загрузка регистра возлагается на программиста. Команда имеет следующий формат:

Название	Операция	Операнд
Игнорируется	УПОТР	Перемещаемое выражение и заключенное в скобки простое абсолютное выражение

Перемещаемое выражение задает значение, которое транслятор может использовать в качестве базового адреса. Абсолютное выражение индекс-регистр, где по предположению будет находиться значение, заданное перемещаемым выражением. Например, в следующей программе:

задаёт

```
ПРОГ1   СТАРТ   2000
        УИА     *(5)
        УПОТР   ПРОГ1(5)
          .
          .
          .
```

Команда УПОТР сообщает транслятору, что в момент исполнения последующих команд в индекс-регистре 5 будет находиться начальный адрес программы ПРОГ1. Фактическая установка на индекс-регистр значения ПРОГ1 выполнена командой УИА.

Если программист меняет значение базового регистра, то новое значение должно быть сообщено транслятору с помощью новой команды УПОТР. В следующих командах АЛЬФА является перемещаемой меткой:

```
УПОТР     АЛЬФА(7)
  .
  .
  .
УПОТР     АЛЬФА+4096(7)
```

Сначала транслятор будет считать, что в регистре 7 находится значение АЛЬФА. Вторая команда заставит транслятор предположить, что в регистре 7 стоит АЛЬФА+4096.

ОТМЕН - Отменить употребление регистра

Если регистр более не должен использоваться для базирования, то он должен быть указан в поле операнда команды ОТМЕН, имеющий следующий формат:

Название	Операция	Операнд
Игнорируется	ОТМЕН	Простое абсолютное выражение, взятое в скобки

Команда ОТМЕН игнорируется, если в ней указан регистр, который никогда не появлялся в команде УПОТР. Если значение выражения превышает 15, то команда не воспринимается и печатается сообщение об ошибке.

Регистр, ставший недоступным для базирования в результате команды ОТМЕН, может быть снова причислен к доступным с помощью команды УПОТР.

КОМАНДЫ СВЯЗИ ПРОГРАММ

На автокоде предусмотрены специальные средства связи программ. С помощью этих средств оказывается возможным в одних программах адресовать объекты, определенные в других программах. В терминах символических обозначений это означает, что в программах допускается употребление меток (как абсолютных,так и перемещаемых), значения которых определяются значениями меток из других программ.

Средства связи позволяют иметь для „общего" объекта свои (внутренние) обозначения в каждой из программ, т.е. обеспечивается полная независимость выбора обозначений при составлении отдельных программ, а процедура установления соответствия обозначений становится отдельным и самостоятельным процессом.

Для программиста средства связи представлены двумя трансляторными командами ВНЕШН и ВХОДН. Но прежде чем перейти к их описанию, введем некоторые понятия и термины, причем для наглядности воспользуемся примером. Пусть в программе с названием ПРОГ1 определена метка МАССИВ, соответствующая некоторой области памяти. Пусть помимо программы ПРОГ1 с этой же областью памяти должна работать программа ПРОГ2, в которой введено свое внутреннее обозначение для этой области ВЕКТОР.

Метка МАССИВ в программе ПРОГ1, т.е. в программе, где она определена, называется входной точкой. Эта же самая метка МАССИВ в программе ПРОГ2 называется внешней меткой. Метка ВЕКТОР программы ПРОГ2, являющаяся внутренним эквивалентом внешней метки МАССИВ, называется свободной переменной. В качестве объединяющего термина для всех упомянутых меток будем употреблять термин <u>метка связи.</u>

Для того, чтобы транслятор заготовил всю необходимую информацию, с помощью которой загрузчик смог бы присвоить правильные значения всем меткам связи, программы ПРОГ1 и ПРОГ2 могли бы, например, выглядеть так:

```
ПРОГ1   СТАРТ                 ПРОГ2   СТАРТ
          .                             .
          .                             .
        ВХОДН  МАССИВ         ВЕКТОР  ВНЕШН  ПРОГ1.МАССИВ
          .                             .
          .                             .
МАССИВ  ПЛМ    (100)                    СЛ     ВЕКТОР
          .                             .
          .                             .
```

Команда ВХОДН в ПРОГ1 специфицирует входную точку МАССИВ, а команда ВНЕШН в ПРОГ2 устанавливает соответствие между свободной переменной ВЕКТОР и внешней меткой МАССИВ из программы ПРОГ1.

Обратим внимание на то, что в программе ПРОГ2 может быть определена и использоваться своя внутренняя метка МАССИВ, а в программе ПРОГ1 никто не запрещает иметь свою внутреннюю метку ВЕКТОР.

Теперь, после того как практическая сторона дела ясна из примера, дадим более полное описание команд ВХОДН и ВНЕШН.

ВХОДН – Специфицировать входную тему

Команда ВХОДН специфицирует входную точку программы. Каждой входной точке должна соответствовать отдельная команда ВХОДН следующего формата:

Название	Операция	Операнд
Игнорируется	ВХОДН	Метка

Метка в поле операнда – это метка, которая определена в данной программе и значение которой может присваиваться свободным пе-

ременным других программ. Если метка нигде в программе не определена, печатается сообщение об ошибке.

Команда ВХОДН может располагаться как выше, так и ниже команды, определяющей метку.

ВНЕШН – Установить соответствие с внешней меткой

Команда ВНЕШН несет две функции: она вводит (и специфицирует) свободную переменную программы и устанавливает эквивалентность этой свободной переменной с внешней меткой.

Команда имеет следующий формат:

Название	Операция	Операнд
Метка	ВНЕШН	Метка.метка

(две метки, разделенные точкой)

Метка в поле названия задает свободную переменную, т.е. внутреннюю метку данной программы, эквивалентную внешней метке. Пара меток в поле операнда задают название „чужой" программы и внешнюю метку, которая в этой чужой программе специфицирована как входная точка.

Команда ВНЕШН – это единственное место в программе, где может быть задана внешняя метка. Во всех остальных командах в теле программы нужно употреблять соответствующую свободную переменную.

Следует иметь в виду важное ограничение. Команда ВНЕШН, описывающая свободную переменную, должна предшествовать в программе всем командам, использующим эту переменную в качестве операнда.

Существуют упрощенные формы команды ВНЕШН, при которых

одна из двух меток в поле операнда опускается. Если опущена внешняя метка (операнд имеет вид: ПРОГ1.), то свободная переменная получит значение начального адреса указанной чужой программы. Если же опущено название программы (операнд имеет вид: .МАССИВ), то заданная внешняя метка будет разыскиваться во всех программах в порядке их загрузки, а свободная переменная получит значение первой найденной метки.

Использование свободных переменных

На использование свободных переменных в программе накладываются следующие ограничения:

1. Свободная переменная может появляться только в машинных командах и адресных константах.

2. В машинных командах она может появляться только в адресном (но не индексном) выражении.

3. Свободная переменная может входить в выражение только как положительное слагаемое.

4. Свободная переменная не может быть входной точкой.

Нарушение этих ограничений фиксируется и приводит к печати сообщения об ошибке.

Обратим внимание на то, что принципиально можно употреблять свободные переменные как в длинных, так и в коротких машинных командах. Однако, следует иметь в виду, что если при загрузке значение свободной переменной "не поместится" в короткий адрес, то программа окажется неработоспособной. При трансляции все короткие команды, употребляющие свободные переменные, особо отмечаются в печатном документе.

ПРИЛОЖЕНИЯ

БЛАНК ДЛЯ ПРОГРАММИРОВАНИЯ НА АВТОКОДЕ

МАШИННЫЕ КОМАНДЫ

Мнемоника в алфавитном порядке.
Буквой п отмечены привилегированные команды.

МНЕМОНИКА	КОП	НАЗВАНИЕ
ВЧ	005	вычитание
ВЧАБ	007	вычитание абсолютных величин
ВЧОБ	006	вычитание обратное
ВЧП	025	вычитание порядков
ВЧПА	035	вычитание из порядка адреса
ВЫПР п	32	выход из прерывания
ДЕЛ	016	деление
ЗНАК	014	изменение знака числа
ЗП	000	запись
ЗПМ	001	запись в магазинном режиме
И	011	"И" логическое поразрядное
ИЛИ	015	"ИЛИ" логическое поразрядное
МОД	22	засылка в регистр-модификатор
МОДА	23	засылка адреса в регистр-модификатор
НЕД	023	номер единицы
НТЖ	012	нетождественность логическая поразрядная /сложение по модулю 2/
ПБ	30	переход безусловный
ПВ	31	переход с возвратом
ПИО	34	переход, если индекс ноль
ПИНО	35	переход, если индекс не ноль
ПО	26	переход, если признак результата ноль
ПЕ	27	переход, если признак результата единица
РЕГ п	002	работа со специальными регистрами
РЖ	027	установка режима
РЖА	037	установка режима по адресу
РЗБ	021	разборка
СБР	020	сборка
СД	026	сдвиг
СДА	036	сдвиг по адресу
СЛ	004	сложение
СЛИ	045	сложение индексов
СЛИА	25	сложение индекса с адресом
СЛП	024	сложение порядков
СЛПА	034	сложение порядка с адресом
СЛЦ	013	сложение циклическое
СТОП п	33	останов машины
СЧ	010	считывание
СЧИ	042	считывание индекса
СЧИМ	043	считывание индекса в магазинном режиме
СЧМ	003	считывание в магазинном режиме
СЧМР	031	считывание младших разрядов
СЧРЖ	030	считывание режима
УВВ п	032	управление вводом-выводом
УИ	040	установка индекса
УИА	24	установка индекса по адресу
УИИ	044	установка индекса по индексу
УИМ	041	установка индекса в магазинном режиме
УМН	017	умножение
ЧЕД	022	число единиц
ЦИКЛ	37	окончание цикла

МАШИННЫЕ КОМАНДЫ С КОРОТКИМ АДРЕСОМ

(в порядке номеров)

КОП		МНЕМОНИКА	НАЗВАНИЕ
000		ЗП	запись
001		ЗПМ	запись в магазинном режиме
002	п	РЕГ	работа со специальными регистрами
003		СЧМ	считывание в магазинном режиме
004		СЛ	сложение
005		ВЧ	вычитание
006		ВЧОБ	вычитание обратное
007		ВЧАБ	вычитание абсолютных величин
010		СЧ	считывание
011		И	"И" логическое поразрядное
012		НТЖ	нетождественность логическая поразрядная
013		СЛЦ	сложение циклическое
014		ЗНАК	изменение знака числа
015		ИЛИ	"ИЛИ" логическое поразрядное
016		ДЕЛ	деление
017		УМН	умножение
020		СБР	сборка
021		РЗБ	разборка
022		ЧЕД	число единиц
023		НЕД	номер единицы
024		СЛП	сложение порядков
025		ВЧП	вычитание порядков
026		СД	сдвиг
027		РЖ	установка режима
030		СЧРЖ	считывание режима
031		СЧМР	считывание младших разрядов
032	п	УВВ	управление вводом-выводом
033	п	УВВ	управление вводом-выводом
034		СЛПА	сложение порядка с адресом
035		ВЧПА	вычитание из порядка адреса
036		СДА	сдвиг по адресу
037		РЖА	установка режима по адресу
040		УИ	установка индекса
041		УИМ	установка индекса в магазинном режиме
042		СЧИ	считывание индекса
043		СЧИМ	считывание индекса в магазинном режиме
044		УИИ	установка индекса по индексу
045		СЛИ	сложение индексов
046		Э46	экстракоды
...		...	
077		Э77	

МАШИННЫЕ КОМАНДЫ С ДЛИННЫМ АДРЕСОМ

(в порядке номеров)

КОП		МНЕМОНИКА	НАЗВАНИЕ
20		Э20	экстракод
2I		Э2I	экстракод
22		МОД	засылка в регистр-модификатор
23		МОДА	засылка адреса в регистр-модификатор
24		УИА	установка индекса по адресу
25		СЛИА	сложение индекса с адресом
26		ПО	переход, если признак ноль
27		ПЕ	переход, если признак единица
30		ПБ	переход безусловный
3I		ПВ,	переход с возвратом
32	п	ВЫПР	выход из прерывания
33	п	СТОП	останов машины
34		ПИО	переход, если индекс ноль
35		ПИНО	переход, если индекс не ноль
36	п		свободный код операции
37		ЦИКЛ	окончание цикла

МНЕМОНИКА КОМАНД ТРАНСЛЯТОРА

УПРАВЛЯЮЩИЕ КОМАНДЫ

КОД	Кодировка программы
СТАРТ	Начало программы
ФИНИШ	Конец программы
АДРЕС	Новое значение счетчика адреса
ЛИТЕР	Начало литеральных констант
СТРН	Начать новую страницу
СТРОК	Пропуск строк

ОПРЕДЕЛЯЮЩИЕ КОМАНДЫ

ЭКВ	Эквивалентность
ПАМ	Память
КОНД	Константа длинная (в полной ячейке)
ТЕКСТ	Текстовая константа

КОМАНДЫ УПРАВЛЕНИЯ БАЗИРОВАНИЕМ

УПОТР	Употреблять регистр для базирования
ОТМЕН	Отменить употребление регистра

КОМАНДЫ СВЯЗИ ПРОГРАММ

ВХОДН	Специфицировать входную точку
ВНЕШН	Установить соотьетствие с внешней меткой

ФОРМАТЫ КОМАНД ТРАНСЛЯТОРА

В косых скобках дается семантика. В командах ЭКВ, ПАМ и КОНД компоненты выражения должны быть определены ранее

Название	Операция	Операнд
игнорируется	КОД	/форма кодировки программы/
метка /назв.прогр./	СТАРТ	самоопред.величина /предполагаемый начальный адрес/
игнорируется	АДРЕС	перемещаемое выражение /новое значение счётчика адреса/
игнорируется	ФИНИШ	перемещаемое выражение /адрес входа в программу/
игнорируется	СТРН	игнорируется
игнорируется	СТРОК	десятичное число /число пропускаемых строк/
метка /определяемая метка/	ЭКВ	выражение /определяющее выражение/
метка /название массива/	ПАМ	(простое абсолютное выражение) /число ячеек массива/
метка /назв.масс.конст./	КОНД	(прост.абс.выражение) константа /коэфф.кратности/ /см.сл.стр./
метка /название конст./	КОНК	константа /см.след.стр./
метка /название конст./	ТЕКСТ	текстовая константа /см.след.стр./
игнорируется	УПОТР	перемещ.выраж. (прост.абс.выраж.) /значение базо- /адрес индекс- вого адреса/ регистра/
игнорируется	ОТМЕН	(прост.абс.выражение) /адрес индекс-регистра/
игнорируется	ВХОДН	метка /входная точка/
метка /своб.переменная/	ВНЕШН	метка.метка /наз- /внешняя вание метка/ чужой програм- мы/

ОСНОВНЫЕ ПРАВИЛА ЗАДАНИЯ КОНСТАНТ

Типы констант

F, Ф – целая десятичная

U, Ю – целая двоичная

B – целая восьмеричная

X – целая шестнадцатеричная

E – число с плавающей запятой

A – адресная

K – команда

T – текстовая

"Тело" константы берется в одиночные кавычки для всех типов, кроме типа A. Выражение, задающее константу типа A, берется в скобки.

В команде КОНД допускается задание коэффициента кратности для всех констант, кроме типа A.

Константа типа E может задаваться только в команде КОНД.

ВНУТРЕННИЙ КОД ТРАНСЛЯТОРА

Внутренний код транслятора представляет собой 8-элементное расширение 7-элементного стандартного кода для передачи данных (СКПД), разработанного на основе рекомендаций международной организации по стандартизации (ИСО).

В кодовой таблице указаны только графические символы СКПД. Номер столбца определяет 4 старших разряда кода, номер строки - 4 младших разряда.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0					ПРО-БЕЛ	0			ю	п		Р	Ю	П		р
1					!	1			а	я	А	Q	А	Я	а	q
2					"	2			б	р	В	R	Б	Р	ь	r
3					#	3			ц	с	С	S	Ц	С	с	s
4					$	4			д	т	D	Т	Д	Т	d	t
5					%	5			е	у	Е	U	Е	У	е	u
6					&	6			ф	ж	F	V	Ф	Ж	f	v
7					'	7			г	в	G	W	Г	В	g	w
8					(	8			х	ь	Н	Х	Х	Ь	h	x
9					)	9			и	ы	I	Y	И	Ы	i	y
10					*	:			й	з	J	Z	Й	З	j	z
11					+	;			к	ш	К	[	К	Ш	k	
12					,	<			л	э	L		Л	Э	l	
13					-	=			м	щ	М	]	М	Щ	m	
14					.	>			н	ч	N	^	Н	Ч	n	~
15					/	?			о		О	_	О		о	ЗБ

Графические символы УПП и АЦПУ и их кодировка
во внутреннем коде транслятора.

▪ - Символы УПП и АЦПУ, которых нет в коде ИСО.

■ - Символы ИСО и УПП, которых нет в АЦПУ.

• - Символы УПП, которых нет ни в ИСО,ни в АЦПУ

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0						0	◇	␣					Ю	П		
1					!	1						Q	А	Я		
2					"	2						R	Б	Р		
3						3						S	Ц	С		
4						4					D	T	Д	Т		
5					%	5						U	Е	У		
6					&	6	∧	∨			F	V	Ф	Ж		
7					'	7	‘	|			G	W	Г	В		
8					(	8						X	Х	Ь		
9					)	9					I	Y	И	Ы		
10					*	:					J	Z	Й	З		
11					+	;						[	К	Ш		
12					,	<	⩽	¬			L		Л	Э		
13					-	=	≠	≡				]	М	Щ		—
14					.	>	⩾	⊃			N		Н	Ч		
15					/		×	÷				_	О			

КОД УСТРОЙСТВА ПОДГОТОВКИ ПЕРФОКАРТ УПП-1

	0	1	2	3	4	5	6	7
0	ПРО-БЕЛ	10	А	Р	F	∨	0	
1	1	↑	Б	С	G	∧		
2	2	(	В	Т	I	⊃		
3	3	)	Г	У	J	¬		
4	4	×	Д	Ф	L	÷		
5	5	=	Е	Х	N	≡		
6	6	;	Ж	Ц	Q	%		
7	7	[	З	Ч	R	◊		
8	8	]	И	Ш	S	|		
9	9	✲	Й	Щ	U	—		
10	+	‘	К	Ы	V	_		
11	−	’	Л	Ь	W	!		
12	/	≠	М	Э	Z			
13	,	<	Н	Ю	—			
14	.	>	О	Я	≤			
15	␣	:	П	D	≥			

КОД ШИРОКОЙ ПЕЧАТИ АЦПУ-128-2

	0	1	2	3	4	5	6	7
0	0	10	А	Р	D			
1	1	↑	Б	С	F			
2	2	(	В	Т	G			
3	3	)	Г	У	I			
4	4	×	Д	Ф	J			
5	5	=	Е	Х	L			
6	6	;	Ж	Ц	N			
7	7	[	З	Ч	Q			
8	8	]	И	Ш	R			
9	9	*	Й	Щ	S			
10	+	‘	К	Ы	U			
11	–	’	Л	Ь	V			
12	/	≠	М	Э	W			
13	,	<	Н	Ю	Z			
14	.	>	О	Я				
15	␣	:	П	_				

О безбланковой форме записи предложений автокода

В безбланковой форме в качестве разделителей между полями используются следующие символы:

: (двоеточие) - между меткой и операцией

, (запятая) - между операцией и операндом

; (точка с запятой) - между операндом и комментарием.

При написании предложений в безбланковой форме нужно соблюдать следующие правила:

I. Нельзя в одном предложении смешивать формы записи. Нельзя, например, название от операции отделить двоеточием, а операцию от операнда - пробелом.

2. Название не должно содержать пробелов и не должно быть пробелов между названием и следующим за ним двоеточием. Двоеточие нельзя опускать, даже если название отсутствует. В этом случае двоеточие должно занимать крайнюю левую позицию на перфокарте.

3. Внутри полей операции и операнда могут попадаться пробелы. Они не влияют на ход трансляции, за исключением случая, когда эти пробелы являются символами в текстовой константе.

4. Общее число символов в предложении (включая пробелы) не должно превышать 5?, т.е. того количества, которое можно написать на бланке.

Для того, чтобы облегчить запоминание этих правил, скажем несколько слов о логике транслятора, имеющей к ним непосредственное отношение.

Транслятор отличает ту или иную форму записи предложения по тому, какой из двух символов - пробел или двоеточие - попадается первым при просмотре предложения слева направо.

Если попалось двоеточие, то транслятор настраивается на форму с разделителями и в процессе дальнейшего просмотра предложения все пробелы выбрасывает, а двоеточие, точку с запятой и самую левую запятую заменяет пробелами. Полученное таким образом предложение транслируется по обычным правилам.