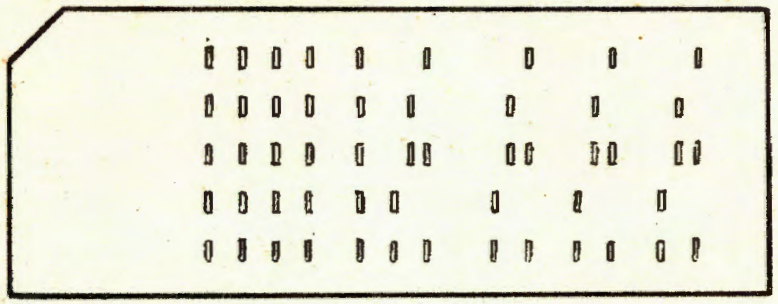
Загинайко В.А., Силин И.Н.

ОБЪЕДИНЕННЫЙ ИНСТИТУТ ЯДЕРНЫХ ИССЛЕДОВАНИЙ

ДЕПОНИРОВАННАЯ ПУБЛИКАЦИЯ

Дубна 19



Объединенный институт ядерных исследований

ЛВТА

В.А.Загинайко, И.Н.Силин

АВТОКОД "АССЕМБЛЕР"

Дубна 1968г.

И Н С Т Р У К Ц И Я

ПО ИСПОЛЬЗОВАНИЮ ПРОГРАММЫ "АССЕМБЛЕР"

Назначение ассемблера

Ассемблер присваивает истинные адреса программе, написанной в коде машины М-20 в условных и истинных (восьмеричных) адресах и отперфорированной на перфокартах с помощью телетайпа с алгольной символикой (пробивка в 5-и разрядном коде, 8 знаков в строке перфокарты). Ассемблер так же может программировать простые арифметические выражения (см. Добавление к инструкции).

Условным адресом может быть идентификатор: последовательность букв и цифр, начинающаяся с буквы. Например,

integer , sq4 , z1m и т.п.

Идентификатор, в принципе, может быть произвольной длины, однако ассемблер воспринимает только первые 6 знаков. Идентификаторы, имеющие большую длину, но совпадающие в первых 6-и знаках, воспринимаются как эквивалентные. Буквы идентификатора могут быть подчеркнутыми. Такие буквы воспринимаются как новый символ и могут быть заменой больших букв. Подчеркивание остальных символов ассемблером игнорируется. Игнорируются также все пробелы, переводы строки и возвраты каретки, которыми можно произвольно пользоваться для повышения наглядности программы. Из соображений краткости и удобства написания следует пользоваться по возможности короткими идентификаторами. Хотя следует стремиться к тому, чтобы они имели смысл (например, были сокращениями от слов и предложений), особенно в логических программах, что увеличивает наглядность программы.

Пример команды в условных адресах:

005, a, b , ab ;

Эта команда означает, что два числа из ячеек памяти, названных a и b , будут перемножены и помещены в ячейку, названную ab . Из примера видно, что код операции и адреса отделяются друг от друга запятой, а в конце команды ставится точка с запятой.

Команды и восьмеричные константы могут снабжаться идентификаторными (и только идентификаторными) метками, чтобы на них можно было ссылаться в других командах. Кроме того, для удобства чтения команды могут снабжаться комментариями.

Пример:

mult: 005, a, b, r, a×b;

Из примера видно, что метка присоединяется к команде слева через двоеточие. Комментарий присоединяется к команде в качестве 4-го адреса. Комментарий может содержать все алгольные символы, кроме точки с запятой, которая служит концом команды, равенства, которое является признаком арифметического выражения, и двоеточия (при наличии в комментарии двоеточия часть комментария воспринимается ассемблером как метка, что приведет к ошибке, если такая метка совпадает с одним из идентификаторов программы).

Метку У метки (т.е. $x: y: \ldots$) писать нельзя.

Адресом команды может быть также неограниченная по числу членов последовательность идентификаторов и восьмеричных чисел, связанных знаками "+" или "-". В этом случае истинным адресом после трансляции будет результат выполнения операций сложения и вычитания по модулю 2^{12} с истинными адресами, соответствующими слагаемым.

Например, пусть истинными адресами, присвоенными при трансляции величинам x и y, будут 0100 и 0200. Тогда:

адрес $x + 1$ окажется равным 0101
адрес $y - 2$ окажется равным 0176
адрес $y - x - 1$ окажется равным 0077 и т.д.

В частности адрес - 1 равен 7777.

Для удобства программиста, чтобы не придумывать лишних меток в тривиальных случаях, введено дополнительное обозначение. Адрес ↑ означает ячейку, в которой окажется транслированная команда, содержащая этот символ (соответственно, в заготовках для команд, формируемых в других ячейках адрес ↑ ставить нельзя).

Пример: 16 ↑+1, 7501, 7610

, x , 5, y, . sin x;

Это обращение к СП-05, $\sin x$. Из примера видно, в частности, что если истинный адрес или код операции начинаются с нулей, то эти нули писать не обязательно.

Удобно, когда каждая команда пробивается на отдельной перфокарте, хотя это и приводит к большому объему программы. По крайней мере, на перфокарте должно пробиваться целое число команд, иначе невозможно вносить изменения.

Псевдокоманды

Кроме обычных команд М-20, ассемблер допускает две одноадресные псевдокоманды с кодами операции:

bss (block starting with simbol, т.е. блок, начинающийся символом) и

equ (*equivalent* , т.е. эквивалентен).

Псевдокоманда *bss* используется для резервации памяти под массивы и для фиксации порядка расположения величин в памяти.

Пример:
```
x:  bss , 100;
y:  bss , 300;
```
означает, что для массива, обозначенного буквой x , нужно отвести 100_8 ячеек, а под массив y - 300_8 ячеек вслед за массивом x .

Пример:
```
t:  bss , 1;
u:  bss , 1;
w:  bss , 1;
```
означает, что простые переменные t , u, w должны быть расположены в памяти машины подряд. Командой *bss* можно пользоваться и для указания эквивалентности меток.

Пример:
```
Z: bss ,  ;
x: bss , 100;
```
означает, что массив x можно использовать также под названием Z .

Псевдокоманды *bss* , резервирующие значительные массивы, следует помещать перед программой или после программы. В этом случае место, отведенное под массивы, не будет выдаваться при печати или перфорации программы.

Адрес *bss* может содержать и идентификаторы. Однако в отличие от остальных команд (кроме *equ*) идентификаторы, стоящие в адресе *bss* , должны быть описаны раньше. Описанием идентификатора считается написание его в виде метки.

Пример:
```
y: bss , 3 ;
z: bss , z-y ;
```
означает, что размерность массива Z равна размерности массива y , стоящего перед ним.

Элементы массивов используются в программе при помощи переадресации или адресной арифметики (идентификатор массива соответствует первому, точнее нулевому, элементу массива).

Пример:
```
x:  bss ,  100;
y:  bss ,  300;
      10,  x ,  1 + 1;
      00,    ,      , sum ;
      52;
    4 05,  x,  y + 2, 20 , x[i]*y[2];
      01,  sum , 20 , sum ;
    1 12,  y-x -1 , 1 -2,  1;
```

суммирует элементы массива *x* , помноженные на третий по счету элемент массива *y* . Обратим внимание, что точку с запятой можно ставить раньше, чем выписаны все адреса команды, как в приведенном примере у команды I0 и 52. Невыписанные адреса окажутся нулевыми.

Псевдокоманда *equ* позволяет присвоить символическому адресу числовое значение или определить его через посредство других символических адресов, описанных раньше.

Пример:

y: *equ*, I00;

означает, что вместо *y* при трансляции везде будет проставлен адрес I00.

Еще пример:

x: *bss*, 400;

y: *equ*, *x* + I00;

Заметим, что эти две псевдокоманды эквивалентны группе *bss* в вышеприведенном примере с суммой.

В отличие от *bss* идентификатор, стоящий в правой части *equ* , не может быть определен в левой части *equ* , так что нельзя написать *y*: *equ*, *y*-*x*; . Впрочем, это и бессмысленно. Так как псевдокоманды *bss* и *equ* одноадресные, то все, что стоит в них после первого адреса до точки с запятой, считается комментарием. Псевдокоманды *bss* и *equ* обязательно должны быть меченными.

Распределение памяти

Память распределяется, начиная с ячейки 0020. Память, зарезервированная при помощи *bss* и на команды, выделяется в том порядке, в котором они стоят в программе. Под каждую обычную команду выделяется одна ячейка памяти. Под псевдокоманду *equ* память не выделяется.

Под каждый неописанный идентификатор (встречающийся только в адресах команд, но не в виде метки) отводится одна ячейка памяти в конце памяти, распределенной под массивы и программу. Неописанные идентификаторы распределяются в том порядке, в котором они первый раз встречаются в программе.

Ячейки, зарезервированные с помощью *bss* внутри программы, заполняются нулями. Состояние ячеек, зарегистрированных вне программы (перед ней или после нее) не определено. Программу можно поместить в любое место памяти, используя фиктивные массивы. Например, чтобы поставить программу с 2000, нужно перед ней поставить, например,

z: *bss*, I760; или *z*: *bss*, 2000-*z*;

последнее удобнее, так как в этом случае не мешают стоящие впереди инструкции.

На машинах с несколькими кубами памяти распределение памяти на других кубах может делаться с помощью *bss* (для перехода в новый куб нужно ставить фиктивные массивы типа *ч*: *bss*, I0020 -*ч*;) или *equ*. При этом в программе нужно вовремя ставить команды переключения кубов. Разрывать тело программы в разные кубы при одной трансляции нельзя.

Работа с ассемблером

Для работы ассемблера требуется, чтобы телетайпная информация была записана на МБ. Поэтому сначала в машину вводится программа записи на МБ. При этом, если в I-м адресе ячейки 7774 есть признак 20, то программа считывает информацию с магнитной ленты (номер ленты и начальный номер зоны задаются в младших разрядах первого адреса и втором адресе ячейки 7774). В противном случае программа требует перфокарт с информацией. После записи информации на МБ программа осуществляет начальный ввод (выполняет команду I0,I,I,0; при нулевом КРА), требуя карту ввода следующей программы. Этой программой может быть либо Ассемблер, либо программа внесения изменений (со следующей за ней информацией об изменениях), либо программа переписи информации с МБ на МЛ.

Программа внесения изменений тоже кончает свою работу начальным вводом. После информации об изменениях тоже может стоять либо Ассемблер, либо запись на МЛ, либо еще раз программа изменений для внесения новых изменений.

Программа записи на МЛ кончает работу остановом по команде с КОП = 77, так что если после записи на МЛ нужна еще и трансляция, то Ассемблер нужно вводить нажатием кнопки "ввод".

Содержимое ячейки 7774 задается либо содержимым ДЗУ-4, если включен тумблер ДЗУ, либо при выключенном тумблере, вводится с перфокарт, например, следующей программкой (пробиваемой на обычном перфораторе):

```
56,   2,   0,   0;
I0,   I,   I,     0;
 , 7774,   ,     4000;  К.А.
00, I740,  0,     0;     пример
0;                       К.Σ.
```

Эта программа кончает свою работу начальным вводом и поэтому может быть положена перед картой ввода той из программ, для которой нужно задать содержимое ячейки 7774 (обычно перед картами записи на МБ). Третий адрес 4000 в адресном коде приведенной выше программы означает блокировку контроля правильности К.Σ.

При этом по третьему адресу ячейки 7774 задается начало РП ИС-2. Если АШ=0, то РП=7200. Если АШ больше, чем 7500, то ИС-2 не вызывается.

По первому адресу указывается сумма необходимых программисту признаков:

0400 печать таблицы распределения памяти ТРП

1000 перфорация ТРП для распечатки на телетайпе

0100 печать программы

0200 перфорация программы

0040 останов после трансляции и выдачи вышеуказанной информации

0020 признак того, что телетайпную информацию нужно читать с ленты

0000-0003 номер магнитофона (программный)

По второму адресу указывается номер начальной зоны МЛ.

В начале работы ассемблера два раза печатаются: две строки нулей для очистки буфера и содержимое ячеек 7771-7774 с контрольной суммой. На машине БЭСМ-3М вторая печать будет нулевой, если включен тумблер ДЗУ. Сразу после трансляции печатаются три кода с контрольной суммой, по второму адресу в которых указывается: начало программы, конец программы и адрес первой свободной ячейки МОЗУ, не занятой программой и рабочими ячейками. Эти печати не зависят от признаков.

Печать таблицы распределения памяти производится порциями по 64 строки - по две строки на каждый идентификатор.

В первой строке и коде операции второй строки печатаются символы идентификатора (не больше шести) - по 3 восьмеричных цифры на символ в специальном 7-разрядном коде (см. ниже). По второму адресу второй строки печатается адрес, который присвоен идентификатору. Если идентификатор не описан, то по третьему адресу второй строки печатается адрес команды, в которой данный идентификатор встретился в первый раз. Если идентификатор описан повторно, то по третьему адресу печатается 7777. Неописанные идентификаторы печатаются в конце в порядке возрастания адресов. Первыми печатаются описанные идентификаторы, которые печатаются в том порядке, в котором они были описаны, т.е. также, в основном, в порядке возрастания адресов, кроме, может быть, определенных при помощи псевдокоманды *equ* .

Таблица распределения памяти перфорируется без $K.\Sigma$ по одной перфокарте. Эти карты могут быть распечатаны на телетайпе. На каждой карте пробивается несколько Строк ТРП в виде:

Идентификатор: адрес ␣ адрес команды, где первый раз встретился идентификатор, если он не описан или 7777, если он описан повторно. ␣ - пробел. Содержимое новой перфокарты при распечатке отделяется от предыдущей лишней пустой строкой. Если адрес начинается с нулей, то эти нули заменяются пробелами. В конце ТРП пробивается карта с одной строкой латинских регистров.

Печать программы начинается с карты ввода (сумма вместе с командами). Программа печатается поперфокартно - двенадцать кодов и впереди строка с адресом первой команды на перфокарте, по второму адресу. Отдельной строкой печатается контрольная сумма программы.

Перфорируется программа с картой ввода в начале и контрольной суммой программы, пробитой на отдельной карте в конце.

Если задан контроль перфорации программы, то после перфорации будет останов. Нужно поставить отперфорированные карты в ЧУ, отделив предварительно карты ТРП, если была и перфорация ТРП, и нажать "Пуск". Произойдет ввод карт с блокировкой МОЗУ. Если суммы карты ввода и программы совпадут, то перфорация закончена, в противном случае будет останов по команде IO.

"Пуском" перфорация может быть повторена (если сумма не совпадет у вводной карты, то сначала довведется программа по команде 30).

Если признак 0040 не задан, то после выполнения вышеописанных пунктов произойдет передача управления на первую истинную команду программы (с предварительным вызовом ИС-2, если он не заблокирован).

Если признак 0040 задан, то будет останов в ячейке 0015. "Пуском" можно пойти на счет. В ячейке 0016 после ухода на счет оказывается команда передачи управления на начало программы, минуя вызов ИС-2, чем можно пользоваться. В ячейке 0017 формируется команда передачи управления на последнюю команду программы. Ею можно пользоваться для аварийной выдачи. Для этого последней командой программы нужно сделать передачу управления на аварийную выдачу. Тогда в случае авоста можно передавать управление в ячейку 0017.

Поиск формальных ошибок

Данный вариант ассемблера непосредственно не фиксирует почти никаких формальных ошибок. Имеется только один аварийный останов:

Если в процессе трансляции транслированная программа окажется слишком длинной (больше - 7200, не включая внешних *bss*), будут напечатаны две строки (--- 3 пробелы), отпечатаны уже полученные части программы и ТРП и отперфорированна полученная часть ТРП (если такая печать и перфорация была указана в признаках) и будет останов 77 с выдачей на P1 и P2 7777 по второму адресу. Перфорация программы в этом случае блокируется. "Пуск" приводит к тому же останову.

Остальные ошибки должны устанавливаться при рассмотрении выданной транслятором информации. Ошибки в написании и пробивке идентификаторов устанавливаются при помощи ТРП, в которой печатаются все идентификаторы, встретившиеся в адресах команд, и все описанные идентификаторы. Особое внимание, естественно, следует обращать на неописанные идентификаторы, т.к. ошибки в написании и пробивке

идентификаторов, как правило, приводят к появлению новых неописанных идентификаторов.

Дважды или более раз описанные идентификаторы будут столько же раз встречаться в ТРП, однако в программе будет использовано только первое описание идентификатора. Повторные описания идентификаторов снабжаются признаком 7777.

Заблаговременно неописанному идентификатору, встретившемуся в адресе псевдокоманд *bss* или *equ*, будет присвоено значение 0011, которое сработает как описание (если в адресе одной псевдокоманды встретится несколько неописанных идентификаторов, им будут присвоены последовательные адреса 11, 12....). В ТРП они будут оформлены как неописанные, но будут помещены среди описанных идентификаторов.

Если в команде опущена точка с запятой, то следующая за ней команда будет принята за комментарий и в транслированной программе исчезнет.

Если в команде стоит лишняя точка с запятой (например, перед комментарием), то в программе появится лишняя команда. Здесь нужно заметить, что ассемблер не фиксирует бессмысленных кодов операции или адресов. Если код операции оказался символическим, но несовпадающим с *bss* или *equ*, то код операции формируется из трех последних символов (с номерами не более 6-го, считая от начала кода). Причем в качестве соответствующей восьмеричной цифры КОПа берутся последние три двоичных разряда символа в 7-ми разрядном коде (см. ниже). Аналогично ассемблер поступает с истинными адресами, если внутри них оказываются символы (признаком истинного адреса является то, что он начинается с цифры), только истинный адрес формируется из 4-х символов.

Не налезает ли программа или рабочие ячейки на рабочее поле ИС-2, можно установить по печати первой свободной ячейки МОЗУ (см. выше).

Настоящий вариант ассемблера допускает не более 1279_{10} строк ТРП, т.е. не более 1279 различных идентификаторов (если нет дважды описанных). Контроля переполнения ТРП не делается.

Псевдокоманды *bss* и *equ*, написанные без метки, воспринимаются ассемблером как обычные исполняемые команды с кодами операции соответственно 233 и 515. Метки у меток ассемблером игнорируются.

Промежуточный семиразрядный код

В этом коде хранятся идентификаторы в ТРП. В этом же коде идентификаторы ТРП выдаются на быстродействующую печать.

a	001	б	033	)	064	A	114	Б	146
b	002	в	034	!	065	B	115	В	147
c	003	г	035	/	066	C	116	Г	150
d	004	д	036	*	067	D	117	Д	151
e	005	ж	037	;	070	E	120	Ж	152
f	006	з	040	(	071	F	121	З	153
g	007	и	041	<	072	G	122	И	154
h	010	й	042	=	073	H	123	Й	155
i	011	к	043	>	074	I	124	К	156
j	012	л	044	,	075	J	125	Л	157
k	013	м	045	:	076	K	126	М	160
l	014	н	046	.	077	L	127	Н	161
m	015	п	047	0	100	M	130	П	162
n	016	т	050	1	101	N	131	Т	163
o	017	ф	051	2	102	O	132	Ф	164
p	020	ц	052	3	103	P	133	Ц	165
q	021	ч	053	4	104	Q	134	Ч	166
r	022	ш	054	5	105	R	135	Ш	167
s	023	щ	055	6	106	S	136	Щ	170
t	024	ъ	056	7	107	T	137	Ъ	171
u	025	ы	057	8	110	U	140	Ы	172
v	026	ь	060	9	111	V	141	Ь	173
w	027	э	061	+	112	W	142	Э	174
x	030	ю	062	-	113	X	143	Ю	175
y	031	я	063			Y	144	Я	176
z	032					Z	145		

Русские буквы введены, т.к. на имеющихся в Дубне трехрегистровых телетайпах на русский регистр внесены русские буквы, не пересекающиеся с латинскими (однако по техническим причинам пока ими пользоваться нельзя).

Трансляция программы по частям

Имеется принципиальная возможность транслировать большую программу по частям. Если программа разбивается на куски, оформленные в системе ИС-2, то все ясно. Каждую СП можно транслировать и отлаживать независимо.

Однако можно разбить программу на части и вне системы ИС-2. Допустим, частей две. Тогда нужно все общие величины и массивы зарезервировать в начале каждой из программ. Кроме того, в одной из программ нужно зарезервировать место для другой программы с учетом ее внутренних величин и рабочих ячеек (лучше с запасом), а также предусмотреть в одной из программ ввод другой программы.

Например, есть две программы *pr1* и *pr2*, у которых есть общие величины: массивы *x* и *y* по 10_8 элементов в каждом, простая переменная *z*. Эти величины одни и те же в обеих программах и поэтому должны в обеих программах попасть в общие ячейки памяти. Кроме того, в обеих программах есть группа рабочих ячеек *r*. В одной 200_8 ячеек, в другой 400_8, которые можно без вреда наложить из соображений экономии памяти. *pr1* обращается к *pr2*. Длина *pr1* вместе с рабочими ячейками (кроме *r*) меньше чем 300_8. Тогда можно написать программы следующим образом:

```
x : bss , 10;
y : bss , 10;
z : bss , 1;
r : bss , 400;
```

```
x : bss , 10;
y : bss , 10;
z : bss , 1;
r : bss , 400;
```

```
pr1: 10, pr2, 1+1;
       :   :   :   :
     16, 1+1, pr2+1, pr2;
pr2: equ, pr1+300;
```

```
pr1: bss, 300;
pr2: 16; ← ячейка возврата
     .   .   .   .
     16, pr2; ← конец счета pr2
```

В этом примере *pr2* должна быть транслирована (а желательно, и отлажена) в начале и выведена на перфокарты. *pr1* вводит *pr2* с перфокарт и в процессе счета обращается к ней. Так как обе программы транслируются независимо, то внутренние переменные и метки могут у них совпадать по написанию. И, наоборот, если в *pr2* заменить обозначения общих величин *x, y, z, r* на *u, v, w, t*, результат не изменится (главное, чтобы суммарная длина памяти, зарезервированной в качестве общей, была одинаковой).

При разбиении программы на большее число частей принцип остается тем же. Организовав в начале программ какого-либо вида переключатели, можно обращаться из них друг в друга

через большое число входов.

Заметим, что, слегка изловчившись, можно после трансляции *pz2* записать ее сразу на МЗУ (соответствующие команды должны стоять, например, в общей памяти, но на МЗУ не записываться или же должны быть введены как отдельная программа с использованием начала *pz2* на вводной карте), а *pzI* тогда может вызывать *pz2* с МЗУ, и т.п.

Части программы могут использовать одно и то же место в памяти (если это логически допустимо), поочередно забивая друг друга.

Есть более простой, но менее удобный способ увязывания частей программы. Внешним переменным в транслируемой части можно при помощи *equ* присваивать истинные адреса из уже транслированных частей.

Принцип работы ассемблера

Чтение информации производится порциями по 177_8 кодов по мере надобности.

Трансляция производится в два просмотра телетайпной информации. При первом просмотре составляется таблица всех меток в 7-разрядном коде и подсчитываются истинные адреса, которые им должны соответствовать (они тоже сводятся в таблицу). При втором просмотре генерируется программа в истинных адресах, причем встречающимся неописанным идентификаторам выделяются ячейки вслед за ячейками, распределенными при первом просмотре. Данные о неописанных идентификаторах включаются в ту же таблицу, что и об описаниях, но при этом еще дается указание о том, в команде с каким адресом идентификатор первый раз встретился.

Транслированная программа по мере накопления порциями по 100_8 кодов сбрасывается на нулевой барабан. Первый код транслированной программы, независимо от его адреса, поступает в 0-ю ячейку барабана. *КΣ* записывается вслед за последним кодом программы. Внешние *bss* на барабан не записываются, однако последняя сотня ячеек барабана, в которую еще попали команды программы и следующая за ней ячейка портятся.

Программа записи информации на МБ

Программа вводит информацию с перфокарт или с МЛ (если есть признак 20 в AI ячейки 7774) и записывает ее на МБ порциями по 177_8 кодов с 200-й контрольной суммой, начиная с 0-й ячейки первого барабана. Информация может вводиться массивами, каждый из которых не более 3776_8 кодов (на МЛ зоны по 3777_8 кодов + *КΣ*, однако последний код служебный).

Признаком того, что вводимый массив не является последним, служит код 777, 0000, 0000, 0000; в конце массива (этот код пробивается на обычном перфораторе). Если объем вводимой части больше 3776_8- печатаются 2 строчки со всеми семерками и программа оста-

навливается по КОП 77.

После записи информации на барабан печатается команда 50 записи на МБ в которой номер МБ и А2 указывают начало свободного места на барабанах. Суммарная информация может занимать МБ1, МБ2 и (если не нужна ИС-2) МБ-3.

Программа кончает свою работу начальным вводом (выполняет команду 10, 1, 1, 0; при нулевом КРА). Запись на МБ делается с полным контролем МОЗУ (в информации не должно быть адресных кодов). В случае сбоя машины могут быть остановы по КОП 35, в том числе непроталкиваемые.

Программа сама заводит на барабане признак конца информации - все семерки, которым пользуются другие программы, в том числе Ассемблер.

В случае несовпадения $K.\Sigma.$ в каком-либо из вводимых массивов информации происходит останов по КОП 10. При пуске программа повторно требует карты того же массива.

Аварийный режим: если программист все же хочет ввести массив с неверной суммой, то нужно его вводить не менее двух раз и последний раз перед пуском сбросить РК машины (регистр команд). При этом, если суммы двух последних вводов совпадали, то запись пройдет, если не совпали, то будет авост по КОП 35. Причем после этого всю запись нужно начинать сначала.

Если последние массивы информации плохо вводятся, а программист хочет записать предыдущие (чтобы напр. переписать их на МД), то для окончания записи нужно ввести фиктивный массив: все семерки и контрольная сумма со всеми семерками.

Программа внесения изменений

Программа вводит с перфокарт массив изменений и вносит эти изменения на МБ (предполагается, что телетайпная информация уже записана на МБ). Информация об изменениях прокладывается картами со служебными словами EXPEL (исключить), BELOW (ниже), FIX (вставить), пробитыми специальным образом. Программа ищет массив (перфокарту или группу перфокарт), стоящий после слова EXPEL или BELOW, на барабане по точному соответствию всех кодов. Массив, стоящий после EXPEL, исключается. Если в информации для изменений после этого есть слово FIX, то следующий за ним массив вставляется на место исключенного (м.б. с расширением или сокращением объема). Массив, стоящий после слова BELOW, задает только координату места, где нужно внести изменения. Если после этого массива стоит слово FIX, то следующий за ним массив вставляется непосредственно ниже массива, указанного после BELOW. Если же после массива BELOW идет EXPEL или BELOW, то поиск нужного нового массива идет ниже предыдущего.

Это можно использовать, например, для указания того, какую из нескольких одинаковых карт информации нужно исключить, в противном случае исключена будет первая из таких карт.

В конце работы программа печатает команду 50-чтения с барабана, второй адрес и номер барабана, в которой указывают начало свободного места на барабанах, после чего делает начальный ввод.

Диагностика: В случае ошибки в задании информации об изменениях печатается вопросительный знак (нулями на фоне пробелов) и после него две строки с КΣ . Во втором адресе первой строки указывается восьмеричный номер контролируемого массива (EXPEL или BELOW), в котором найдена ошибка, а во втором адресе второй строки номер ошибки: 0, если не находится контрольный массив или информация об изменениях не начинается словом EXPEL или BELOW ; 1, если массив, стоящий после FIX , кончается словом FIX ; 2, если длина информации для изменений превышает 3776_8 кодов; 3, если после FIX нет соответствующего массива.

Внесение изменений делается с полным контролем МОЗУ (К.Σ. предсказывается и сравнивается с полученной). В случае сбоя могут быть остановы по КОП 35, в том числе не проталкиваемые. В информации не должно быть адресных кодов.

При несовпадении суммы в массиве изменений происходит останов по КОП 10. При пуске программа требует снова карты информации об изменениях.

Замечания к программе внесения изменений

Поиск контролируемого массива (стоящего после EXPEL или BELOW) делается не по совпадению смысла, а по совпадению пробивок. Поэтому контролируемые массивы должны быть дубликатами или оригиналами перфокарт изменяемой телетайпной информации.

Поиск контролируемого массива идет только до первого совпадения. Если перед этим массивом не было BELOW с соответствующим массивом (см.выше), то поиск идет с начала информации.

Следует иметь в виду, что при поиске перфокарты, в принципе она может совпасть с концом более длинной перфокарты, совпадающим с ней по написанию. Например, на одной карте пробиты две команды 01, *a* , *b* , *c* ; 52;

на другой одна 52;

и мы хотим исключить 2-ю карту. Без принятия специальных мер (использования BELOW) может вместо нее исключиться вторая команда первой карты (если произойдет совпадение по пробивке).

Еще больше возможностей для этого в арифметических выражениях.

Указанное совпадение может произойти только с концом карты, т.к. в конце карты есть характерный признак (точнее должен быть)-возврат каретки и перевод строки (если по каким-либо соображениям есть возврат каретки и перевод строки внутри карты, то в конце карты желательно пробивать возврат каретки и два перевода строки).

Недоразумения невозможны, если все карты в комментарии содержат номер перфокарты, однако в настоящем варианте ассемблера это невозможно в арифметических выражениях.

Недоразумений также не должно быть, если контролируемый массив содержит метку.

Программа переписи информации с магнитных барабанов на магнитную ленту

Программа считывает информацию с МБ и записывает ее на МЛ во столько зон, во сколько поместится. Длина каждой зоны 3777_8 + КΣ . Номер ленты задается в двух младших разрядах I-го адреса, а номер начальной зоны - во втором адресе ячейки 7774. Признак 20 в AI яч. 7774 на работу данной программы не влияет.

В конце информации на каждой не последней зоне МЛ программа сама проставляет признак продолжения (код 777, 0000, 0000, 0000). В конце информации в последней зоне записывается код 777, 7777, 7777, 7777. Оба признака нужны для считывания информации с МЛ программой записи информации на МБ.

Запись на МЛ идет с полным контролем МОЗУ. При сбое машины могут быть остановы по КОП 35. Запись на ленту не следует делать, если при записи информации на МБ или при внесении изменений вводимая информация проталкивалась с неверной КΣ , так как после неверной записи на ленту, записанную информацию будет практически невозможно исправить с помощью программы внесения изменений.

МЛ должна быть размечена на зоны по 4000_8 кодов.

ДОБАВЛЕНИЕ К ИНСТРУКЦИИ ПО ПОЛЬЗОВАНИЮ АССЕМБЛЕРОМ

В настоящее время имеется новый вариант ассемблера, в котором допускаются арифметические выражения.

Например: $y = (a+b) + (c+d) \times (b+c) \uparrow (a-d);$

Арифметические выражения могут снабжаться метками, например:

$test: \quad y = x \uparrow 3 - x - 1;$

Для операции приняты следующие обозначения:

сложение: +

вычитание: -

деление: / (косая черта)

возведение в степень: ↑

оператор присваивания: =

В качестве переменных могут быть простые переменные, обозначающие номер ячейки, а также переменные с индексами и переменные с точкой. Переменные с индексами понимаются в смысле адресного сложения.

Например, $a(I)$ - номер ячейки, следующей за a;

$a(c+d-2)$ это ячейка с номером $a+c+d-2$.

Вложение индексов, например: $a(c(d(-2)))$, не допускается.

В обычных командах также можно писать вместо $a+b$

$a(b)$

Переменная с точкой обозначает, что при вычислениях к номеру соответствующей ячейки добавляется текущее значение регистра адреса.

Например, выражение

$$y. = a. \times b \ ;$$

ассемблер запрограммирует в таком виде:

505, a , b , y ;

Это позволяет записать программу суммирования массива чисел размерности n примерно в таком виде:

```
summa=0;   52;
cikl : summa = summa + a. ;
112 , n-1 , cikl , 1      ;
```

Числовые выражения обозначают номера ячеек. Например, выражение

$$y = x + 2 \ ;$$

запрограммируется в виде

0I, x , 0002, y ;

Возведение в степень ассемблер программирует в три команды как обращение к специальной стандартной программе возведения в степень СП-57. Например, $y. = x \uparrow a$ ассемблер запрограммирует так:

```
I6,    ↑ +I,    750I,    76I0;
00,      0,     0057,       0;
06,      x,       a ,       y;
```

СП-57 работает таким образом, что три случая

$$y = x \uparrow 1; \quad y = x \uparrow 2; \quad y = x \uparrow 3;$$

всегда воспринимаются как

$$y = x; \quad y = x \times x; \quad y = x \times x \times x;$$

СП - имеет рабочие ячейки 0001 + 0003.

После работы СП там находятся коды: 101, 4000;
102, 4000;
102, 6000.

Для сложных арифметических выражений ассемблер заводит рабочие ячейки с идентификаторами

$$af0; \quad af1; \quad af2; \quad \ldots af№; -$$

одни и те же для всех арифметических выражений. Поэтому не рекомендуется использовать ячейки с такими идентификаторами для хранения промежуточной информации, т.к. она может быть испорчена при счете арифметики.

Если число вложений скобок друг в друга превышает 31_{10} (включая и те скобки, которые нужно было бы расставить, если бы не было известно старшинство операций), то начинает портиться начало Ассемблера, что до поры до времени еще не влияет на его работу. Старший номер в $af№$, которые выдумывает Ассемблер, соответствует скобочному уровню (в восьмеричной системе).

ИНФОРМАЦИЯ ОБ ОШИБКАХ

1. Если в формуле не хватает открывающих скобок, то печатается код

00, 0001, *nistr*, 0;

где *nistr* - место в программе, где примерно надо искать ошибку, и трансляция продолжается.

2. Если при работе транслированной программы отрицательное число возводится в дробную степень, происходит печать $|x|, y$ и ячейки 7610 (информация о месте ошибки в программе) с $K\Sigma$ в восьмеричной системе, в качестве результата Z засылается $|x| \uparrow y \times (-1) \uparrow n$, где n ближайшее целое y, и программа продолжает вычисления.

П Р И М Е Ч А Н И Е :

Транслированная программа использует МБ-3 (ИС-2) и МБ-2 (ячейки 5602 - 6003) (туда записывается при вводе ассемблера СП-57), если они нужны.

Если необходимо повторно считать уже транслированные программы, в которых используется степень, необходимо перед транслированной программой подложить колоду с СП-57.

ЗАМЕЧАНИЯ О КОММЕНТАРИЯХ

В связи с подключением арифметического блока к ассемблеру комментарии, стоящие в командах, не должны содержать символа присваивания = . Комментарии в арифметических выражениях не допускаются.

В.А.Загинайко, И.Н.Силин.

Настоящий вариант ассемблера обладает значительно более широкими возможностями по сравнению с предыдущим вариантом.

Дальнейшим совершенствованиям будет подвергаться только этот вариант.

Издательский отдел ОИЯИ, заказ № 6897