В.Ф.Турчин
/г.Москва/

ТРАНСЛЯТОР С АЛГОЛа, НАПИСАННЫЙ НА ЯЗЫКЕ РЕФАЛ

I.Общие сведения о трансляторе

Язык РЕФАЛ [I], являющийся упрощенным вариантом метаалгоритмического языка [2,3], ориентирован на задачи преобразования цепочек символов, и в частности, на составление трансляторов. В настоящем докладе описывается транслятор, написанный на РЕФАЛе. Входным языком транслятора является сужение АЛГОЛа, получившее название САБСЕТ-АЛГОЛ ("Подмножество АЛГОЛа") и утвержденное Советом Международной Федерации по Обработке Информации (IFIP) в качестве международного языка программирования [4]. Наиболее существенными ограничениями, введенными в САБСЕТ-АЛГОЛ по сравнению с полным АЛГОЛом, являются отсутствие рекурсивных процедур, запрещение побочного эффекта, допущение в качестве фактических параметров, заменяющих вызываемые по наименованию формальные параметры, лишь идентификаторов и строк (но не общих выражений, как в полном АЛГОЛе). Кроме того, по техническим причинам (использование строчных кавычек в РЕФАЛ-интерпретаторе) представляемый в настоящее время вариант транслятора не допускает использования строк во входном языке, а также ограничителей параметра вида) <строка букв> :(. Это ограничение будет устранено в дальнейшем. Выходным языком транслятора является автокод "БЕМИ" для машины БЭСМ-6, созданный группой программистов ИПМ АН СССР во главе с В.С.Штаркманом [5] и снабженный транслятором. Сведения о машине и об автокоде будут проводиться

по мере необходимости.Сейчас укажем только,что БЭСМ-6 - это одноадресная машина с I5-ю индекс-регистрами и оперативной памятью 32 тыс. ячеек,а автокод БЕМШ является автокодом "один к одному", т.е. одной команде автокода соответствует не более одной команды машины (если не считать "пустых" команд,вставляемых для заполнения свободной половины ячейки).

Для практического применения программ,написанных на РЕФАЛе, необходим транслятор с этого языка.Транслятор с РЕФАЛа для БЭСМ-6, работающий в режиме интерпретации,описан в докладе "РЕФАЛ-интерпретатор",представленном на данной конференции (обозначим его [A]). В нем же дается краткое описание языка РЕФАЛ.

Для выполнения трансляции надо в поле памяти РЕФАЛ-интерпретатора ввести описание транслятора на РЕФАЛе,а в поле зрения интерпретатора - выражение **К**'ТРАЛГ' ξ . .Здесь ξ - текст на АЛГОЛе в некотором конкретном представлении, ТРАЛГ - символ процедуры "трансляция с **АЛГОЛа**".В результате работ РЕФАЛ-интерпретатора будет выдан переведенный текст на автокоде БЕМШ. **Заметим,что хотя трансляция и** выполняется в режиме интерпретации,сам транслятор АЛГОЛ-БЕМШ является компилятором,так что в конечном счете выдается эффективная скомпилированная программа.

Используемое нами конкретное представление АЛГОЛа отличается от эталонного языка следующим:

I.Квадратные скобки кодируются как круглые.

Основные символы,изображаемые английскими словами кодируются этими словами,взятыми в строчные кавычки,так что они образуют символы в смысле РЕФАЛа [A] ,например: 'REAL'

3.Коды следующих четырех символов дополнительно содержат скобки:

begin	кодируется как	'BEGIN'(
end	" "	)'END'
if	" "	'IF'(
then	" "	)'THEN'

4. Все буквы - большие (в официальном САБСЕТ-АЛГОЛе они маленькие); разрешается использовать также русские буквы в соответствии с кодом УПП [5] .

Первый параграф описания транслятора имеет вид:

§I. К'ТРАЛГ'⊞ ≡ К'ПОДГ', К'ВАВК' К'КОМП' К'ПРОГ' К'ШПР' ⊞....

Таким образом,сначала выполняется подготовительная процедура'ПОДГ' затем к алгольному тексту применяются последовательно процедуры: 'ШПР' - первый просмотр,'ПРОГ' - программирование на автокоде, 'КОМП' - компоновка программы.После этой третьей процедуры в поле зрения оказывается полный текст программы на автокоде,записанный во внутреннем коде РЕФАЛ-интерпретатора.Процедура 'ВАВК' - выдача текста на автокоде является "машинной командой" [4] ,перфорирующей этот текст в виде,необходимом для трансляции с автокода.

Вследствие ограниченности объема доклада не будем приводить полного описания транслятора,а опишем принципы его работы,приводя лишь некоторые из описаний процедур.В общей сложности транслятор содержит 139 процедур,описываемых с помощью 407 предложений.

Так как автокод БЕМШ не допускает блочной структуры памяти, метки автокода фактически эквивалентны адресам; будем называть их "адрес-метками".Транслятор порождает адрес-метки,имеющие вид одной или нескольких букв,за которыми следует натуральное число (десятичное).Буквы несут информацию о природе адрес-метки,и это

используется при работе транслятора. Адрес-метки, соответствующие описаниям /т.е. являющиеся формальными параметрами/ идентификаторы расшифровываются следующим образом.

Буква L означает метку, буква W - переключатель. Величины, которым может быть приписан тип, обозначаются двумя буквами. Буква, стоящая на первом месте, означает: S - простая переменная, A - массив, P - процедура. На втором месте стоит буква, обозначающая тип: R - real, I - integer B - Boolean. Процедура, не являющаяся процедурой-функцией, имеет обозначение PP. Формальные параметры кодируются следующим образом. Простые переменные, кроме входящих в список значений (см. ниже) имеют буквы F <тип>, где <тип> - одна из букв R, I, B. Метки и переключатели - FL и FW, соответственно. Массивы и процедуры получают третью букву F в дополнение к коду, который они имели, если бы были описаны. Примеры: $SR5$ - адрес-метка действительной величины, $PIF67'$ - формального параметра, имеющего специфику целой процедуры-функции. Полное число символов адрес-метки не должно превосходить шести.

Для того, чтобы иметь счетчики, вводится машинная команда 'СЧ'. Она действует следующим образом. Выполнение конкретизации $\underline{К}$'СЧ'$\uparrow\alpha=\beta_{\perp}$ приводит к тому, что счетчику с идентификатором α присваивается значение β (α и β - цепочки не более чем из шести символов). Выражение $\underline{К}$'СЧ'$\alpha_{\perp}$ (α не должно начинаться со знака $\uparrow$) заменяется на текущее значение счетчика α.

Для распределения памяти при наличии блочной структуры вводится машинная команда 'СТ', работающая следующим образом. Выполнение конкретизации $\underline{К}$'СТ'$\uparrow\alpha=\beta_{\perp}$ приводит к занесению β в

стэк в качестве значения α (учитываются только первые шесть символов цепочки α). При конкретизации К'СТ' > $\perp$ в стэке проводится черта. При конкретизации К'СТ' < $\perp$ стирается последняя черта и все, что было занесено после нее. Выражение К'СТ' $\alpha \perp$ (α не должно начинаться со знаков $\uparrow$ > , <) заменяется на последнее из сохранившихся значений α.

2. Первый просмотр

Процедура 'ШПР' применяется к последовательности операторов, разделенных точкой с запятой. При этом предполагается, что описания, действующие в этом месте программы, уже обработаны и необходимая информация занесена в стэк. Процедура 'ШПР' действует только на одном уровне блочной структуры; встречая блок, она обходит его, помечая символом 'БЛОК', и уже процедура 'ПРОГ' входит в этот блок и обрабатывает описания, после чего рекурсивно включаются 'ШПР' и 'ПРОГ'. Лишних просмотров при этом не делается, ибо блок обходится по адресам скобок, связанных с символами 'BEGIN' и 'END' [А].

Процедура 'ШПР' выполняет следующие действия:

1. Отделяет метки и заносит их в стэк, приписывая им адрес-метки автокода. Это основная задача первого просмотра.

2. Выделяет линейные участки, на которых возможна оптимизация выборки значений переменных с индексами и применяет к ним процедуру 'ОПТИМ' - оптимизация.

3. Осуществляет процедуру 'ПОБВ' - первичная обработка выражений.

Первичная обработка выражений состоит в выделении идентификаторов и чисел, замене идентификаторов на их адрес-метки, заключенные в скобки, и приведении чисел к виду "литеральных констант" автокода, снабженных буквой S спереди и также заключенных в скобки. Аппарат

"литеральных констант", имеющийся в автокоде БЕЛЬ, позволяет использовать в команде в качестве операнда само число (а не его адрес-метку). Соответствующие константы заводятся при трансляции с автокода. Числу N, записанному на АЛГОЛе, отвечает литеральная константа =Е'N' в автокоде БЕЛЬ. Кавычки автокода кодируются чертой | во внутреннем языке РЕФАЛ-интерпретатора и заменяются на кавычки процедурой 'ВАВК' . Пример: если идентификатор SUM имеет адрес-метку $SR121$ в данном блоке, то выражение $(SUM - 1.5_{10} - 6) \uparrow 3$ после первичной обработки приобретает вид:

$$((SR121) - (S=E|1.5_{10}-6|)) \uparrow (S=E|3|)$$

Такая форма записи чисел позволяет при программировании выражений обращаться с константами так же, как с простыми переменными.

Приводим описание процедуры 'ПОБВ' и подчиненных ей процедур.

```
§7.1 К'ПОБВ''TRUE'Е2 ≥ (S=E|0|)  К'ПОБВ'Е2 .
§7.2 К'ПОБВ''FALSE'Е2 ≥ (S=E|1|)  К'ПОБВ'Е2 .
§7.3 К'ПОБВ' S I Е2 ≥ К'ПОБВУ' К'КЛАСС' S I . Е2 .
§7.4 К'ПОБВ' (Е I)Е2 ≥ (К'ПОБВ' Е I .) К'ПОБВ' Е2 .
§7.5 К'ПОБВ' ≥

§8.1 К'ПОБВУ' Б S I Е2 ≥ К'ОТЩИД' ( S I)Е2 .
§8.2 К'ПОБВУ' Ц S I Е2 ≥ К'ОТЩЧ' ( S I)Е2 .
§8.3 К'ПОБВУ' Ч S I S 2 Е3 ≥ К'ОТЩЧ' ( S I S 2)Е3 .
§8.4 К'ПОБВУ' П S IЕ2 ≥ S I К'ПОБВ' Е2 .
```

§I0.I К'ОТЩИД' (ЕI) ≤ А Е2 ≥ К'ОТЩИДУ'(ЕI)К'КЛАСС' ≤ А, Е2 .
§I0.2 К'ОТЩИД' (ЕI)(ЕА)Е2 ≥ (К'СТ'ЕI .)
(К'ПОБВ'ЕА .)К'ПОБВ' Е2 .)
§I0.3 К'ОТЩИД' (ЕI) ≥ (К'СТ' ЕI .)

§II.I К'ОТЩИДУ'(ЕI)Б ≤ А Е2 ≥ К'ОТЩИД'(ЕI ≤ А) Е2 .
§II.2 К'ОТЩИДУ'(ЕI)Ц ≤ А Е2 ≥ К'ОТЩИД'(ЕI ≤ А) Е2 .
§II.3 К'ОТЩИДУ'(ЕI) ≤ К ≤ А Е2 ≥
(К'СТ' ЕI .) ≤ А К'ПОБВ' Е2 .

§I2.I К'ОТЩЧ' (ЕI) ≤ А Е2 ≥ К'ОТЩЧУ' (ЕI) К'КЛАСС' ≤ А . Е2 .
§I2.2 К'ОТЩЧ' (ЕI) ≥ (S = Е | ЕI |)

§I3.I К'ОТЩЧУ'(ЕI)Ц ≤ А Е2 ≥ К'ОТЩЧ'(ЕI ≤ А) Е2 .
§I3.2 К'ОТЩЧУ'(ЕI)Ч ≤ А ≤ В Е2 ≥ К'ОТЩЧ'(ЕI ≤ А ≤ В) Е2 .
§I3.3 К'ОТЩЧУ'(ЕI) ≤ К ≤ А Е2 ≥ (S = Е | ЕI |) ≤ А К'ПОБВ' Е2 .

Процедура 'КЛАСС' ,описание которой мы опустили,ставит перед символом букву - признак класса этого символа.Буквы предваряются буквой Е ,цифры - Ц ,символы (десятичная точка) и $_{10}$ (указатель десятичного порядка)- буквой Ч ,описатели и спецификаторы - буквой О ,все прочие основные символы АЛГОЛа - буквой П В предложениях §I3.I - I3.3 учитываем,что описатели не могут появиться в аргументе процедуры 'ПОБВ'

Линейный участок состоит из последовательности операторов присваивания,не содержащих указателей функций (кроме стандартных). Оптимизация начинается с составления "списка возможных баз"

- 'СПВБ' .Под "базой" понимаем идентификатор массива и список индексов, определенный с точностью до одного постоянного (целого) аддитивного члена, не превышающего тысячи в последнем индексном выражении. Так, переменные $BAC[i,j-1]$, $BAC[i,j]$, $BAC[i,j+1]$ имеют одинаковую базу. В список возможных баз входят все базы, встречающиеся в линейном участке не менее двух раз. Затем из этого списка исключаются те базы, которые содержат идентификаторы, встречающиеся в левых частях операторов присваивания линейного участка. Оставшимся базам приписываются индекс-регистры, начиная со второго. Последний регистр, отведенный под оптимизацию, - девятый. Если баз больше восьми, то оставшиеся базы игнорируются. Перед началом линейного участка помещаются команды засылки адресов баз в соответствующие индекс-регистры. Эти команды снабжаются признаком 'АВК' ("автокод"); это означает, что при работе процедуры 'ПРОГ' они будут пропущены. Так поступаем всегда, когда 'ПШР' дает готовые команды на автокоде. В линейном участке производится замена переменных с индексами, для которых есть подходящие базы, имеющий вид дополнения до базы, модифицированного соответствующим регистром, причем, этот операнд оформляется так же, как простая переменная, то есть к нему добавляется S и он заключается в скобки. Если, например, базе $K[n \times m]$ соответствует регистр 6, то переменная $K[n \times m + 20]$ заменяется (S20(6)). Если при программировании выражений значение этой переменной надо задать на сумматор, будет записана команда: СЧ,20(6); (признак S отбрасывается). По семантике автокода [5] такая команда означает считывание числа с адресом из регистра 6, увеличенным на 20.

Приведем описание процедуры 'ЗАМЕНА' ,которая,после того как составлен список соответствий между базами и регистрами,записывает команды занесения адресов в регистры и осуществляет соответствующие замены в линейном участке.

```
§42.I К'ЗАМЕНА' ( WI W2 = S 4)E5'НАД' E6 ⊇
                  ('АВК'К'ЗАСАДР' W I W2 . УИИ, S 4 (I);)
                  К'ЗАМЕНА' E5'НАД' К'ЗАМI' ( W I W2 = S 4)
                                                'НАД' E6 . .
§42.2 К'ЗАМЕНА' 'НАД' E6 ⊇ E6

§43.I К'ЗАМI' ( W I(E2)= S 4) 'НАД' EA W I(E2 EB) EC ⊇
          К'ЗАМIУ' ( WI (E2)   S 4 )'НАД' EA .

          К'КОНТР' W I ( E2;EB)'РЕГ'S 4 .
          К'ЗАМI' ( W I (E2) = S 4 )'НАД' EC .
§43.2 К'ЗАМI' EI ⊇ К'ЗАМIУ' EI .

§44.I К'ЗАМIУ' W I 'НАД' E2 (E3) E4 ⊇
          E2 ( К'ЗАМI' W I 'НАД' E3 . )
          К 'ЗАМIУ' W I 'НАД' E 4 .
§44.2 К 'ЗАМIУ' W I 'НАД' E2 ⊇ E2

§45.I К'КОНТР' W I (E2 ; )'РЕГ' S 4 ⊇ (S( S 4))
§45.2 К'КОНТР' W I (E 2;EC (S= E |E3|))'РЕГ' S 4 ⊇
          К'КОНТРУ' E2 ; EC 'ЦНБТ?' К'ЦНБТ?' E3 .( W I S4).
```

§45.3 К'КОНТР'W I (Е2 ; Е8)'РЕГ'≤ 4 ≠ WI (Е2 Е3)

§46.I К'КОНТРУ' Е2 ; + 'ЦНБТ?' 'ДА' Е3(WI ≤ 4) ≡ (S Е3(≤ 4))
§46.2 К'КОНТРУ' Е2 ; - 'ЦНБТ?' 'ДА' Е3(WI ≤ 4) ≡ (S -Е3(≤ 4))
§46.3 К'КОНТРУ' Е2; 'ЦНБТ?' 'ДА' Е3(W I ≤ 4) ≡ (S Е3(≤ 4))
§46.4 К'КОНТРУ' Е2 ;ЕС 'ЦНБТ?' ≤ 2 Е3 (W I ≤ 4) ≡
W I (Е2 ЕС (S = Е |Е3|))

Из предложения 42.I **видно, какого типа аргумент требует** процедура 'ЗАМЕНА' .Два терма W I W 2 представляют собой базу:ее адрес-метку,заключенную в **скобки** и список индексов в скобках.Знак равенства отделяет базу от соответствующего ей номера регистра - символа ≤ 4 В целом скобка (W I W 2 =≤ 4) есть одно указание к замене;список указаний записывается в виде последовательности **таких скобок** и остальная его часть образует Е5. Выражение Е6 - линейный участок,где надо производить замену - записывается после разделителя 'НАД' .

Процедура 'ЗАСАДР' выполняет засылку в **регистр** I адреса переменной с индексом. УИИ - "установка индекса по **индексу**" - мнемонический код операции машины БЭСМ-6 (в автокоде БЕМШ), выполняющей засылку содержимого одного индекс-регистра (в нашем случае - регистра I) в другой индекс-регистр (в нашем случае в индекс-регистр ≤ 4).

Процедура 'ЦНБТ?' приписывает спереди к цепочке символов 'ДА', если она изображает целое десятичное число,не превышающее тысячи, и 'НЕТ'- в противном случае.

3. Программирование на автокоде

Основная работа по программированию, в том числе и распределение памяти, включается в процедуру 'ПРОГ'. Распределение памяти основано на следующих принципах.

1. Каждому идентификатору (с учетом принадлежности к блоку) кроме идентификаторов меток соответствует ячейка памяти, помеченная адрес-меткой идентификатора, с той только поправкой, что если это адрес-метка простой переменной, то буква S опускается. Кроме того, идентификаторам массивов соответствует набор ячеек для хранения граничных констант; переключатель имеет одну дополнительную ячейку для индексного выражения, процедура - ячейку для хранения адреса начала динамической памяти. Эти ячейки образуют статическую память. Все они расположены подряд в начале программы и при переводе с автокода будут иметь так называемые "короткие" адреса. Число их не должно превышать 4 тыс. Таким образом, экономии статической памяти за счет блочной структуры не делается. Это оправдывается тем, что для большой машины такая экономия бывает на практике несущественной; с другой стороны, при отладке часто полезно иметь информацию о параллельных блоках, которая при экономии памяти портится. Заметим, что в процессе программирования команды, резервирующие ячейки статической памяти, порождаются вразбивку и снабжаются признаком 'SM', а процедура 'КОМП' переставляет их в начало программы.

2. Ячейки простых переменных содержат значения этих переменных. Ячейки формальных параметров простых переменных или меток содержат

ячеек, соответствующих фактических параметров. **Исключение представляют** формальные параметры простых переменных, вызываемых по значению: они трактуются как локальные простые переменные. Ячейки процедур и переключателей содержат команды передачи управления на соответствующие подпрограммы; информационные ячейки следуют непосредственно за ними. Ячейки формальных параметров процедур и переключателей содержат адреса соответствующих ячеек фактических параметров.

3. Распределение памяти под массивы - полностью динамическое, с учетом блочной структуры. Ячейка идентификатора массива содержит адрес **начала массива**, вырабатываемый при входе в блок. Адрес начала граничных констант хранится в момент компиляции в стэке. Ячейка формального параметра, специфицированного как массив, содержит два адреса: начала массива и начала граничных констант. В любой точке программы должна быть определена ячейка, в которой хранится адрес начала свободного поля для динамической памяти (АНДП). Это достигается введением в процессе компиляции условного "идентификатора" - 0, адрес-метки которого для различных блоков хранятся в стэке на общих правах **с адрес-метками настоящих идентификаторов.** При входе в блок, не содержащий описаний массивов, адрес-метка "идентификатора" 0 не меняется. При входе в блок с новыми массивами заводится новая адрес-метка. В режиме исполнения программы после отведения памяти под все массивы в этой ячейке будет стоять текущее значение АНДП. При выходе из блока с массивами восстанавливается прежнее значение адрес-метки "идентификатора" 0, и, следовательно, значение АНДП предыдущего уровня.

Из процедур, входящих в программирование на автокоде, остановимся лишь на программировании арифметических выражений. Соответствующая процедура имеет детерминатив 'ЗАСУМ' - засылка в сумматор. Конкретизация К'ЗАСУМ' ξ . , где ξ - арифметическое выражение, уже подвергшееся первичной обработке ('ПОБЗ'), есть программа засылки в сумматор этого выражения. На языке РЕФАЛ возможно чрезвычайно краткое описание процедуры 'ЗАСУМ', однако предпочитаем более длинное, но работающее быстрее и приводящее к более эффективной программе описание.

```
§II5.I К'ЗАСУМ''IF'(ЕВ)'THEN' ЕI'ELSE' Е2 = К'АКТ'
        'КАП'('ЕСЛИ-НЕ' ЕВ 'ПЕР' К'М1' .)
        'КАП'('ЗАСУМ' ЕI) ПБ, К'М2' . ;
        К'М1' . : 'КАП'('ЗАСУМ' Е2 )
        К'М2' . : К'+2' М . .
§II5.2 К'ЗАСУМ' + ЕА = К'ЗАСУМ' ЕА .
§II5.3 К'ЗАСУМ' - ЕА = К'ЗАСУМ'(50 ) - ЕА .
§II5.4 К'ЗАСУМ'ЕА = К'ЗАСУМП' ЕА 'ЗАСУМ+' .
```

Процедура 'АКТ' - активизация, - использованная в § II5.I служит для изменения порядка выполнения конкретизаций. Она описывается предложениями:

```
§97.I К'АКТ' ЕI 'КАП' (Е2) = ЕI К К'АКТ' Е2 . . К'АКТ' Е3 .
§97.2 К'АКТ' ЕI = ЕI
```

Здесь 'КАП' "пассивный" символ конкретизации, который "активизируется" процедурой 'АКТ'

Символы 'М1' и 'М2' текущие метки, вырабатываемые самим транслятором. Необходимые правила конкретизации суть

§52 К 'М1' ≥ М К'СЧ'М .

§53 К 'М2' ≥ М К'ВЧСЛ' + К'СЧ'М . , 1 .

§54 К '+2' EI ≥ К'СЧ ↑ EI = К'ВЧСЛ' + К'СЧ' EI . , 2 . .

Последнее предложение определяет процедуру '+2' - увеличение на 2 счетчика номера М текущей метки. Процедура 'ВЧСЛ' - вычисления с целыми десятичными числами. Например К+25,6 . есть 31.

Процедура 'ЕСЛИ-НЕ' служит для программирования логических выражений. Она дает программу, которая передает управление на следующую ячейку, если условие выполнено, и на метку, следующую за символом 'ПЕР' (переход), если условие не выполнено.

Символ 'ЗАСУМП' расшифровывается как "засылка в сумматор первичного арифметического выражения". Эта процедура рассчитана на то, что последний символ в ее аргументе является детерминативом процедуры, которую надо применять, если аргумент (без последнего символа, разумеется) не оказался первичным выражением, (§116.9).

§116.1 К'ЗАСУМП' (S EI) S P ≥ СЧ, EI;

§116.2 К'ЗАСУМП' (F EI) S P ≥ МОД, F EI; СЧ, 0;

§116.3 К'ЗАСУМП' (A EI) W 2 S P ≥ К'ЗАСАДР'(A EI) W 2 .
СЧ, (1);

§II6.4 К'ЗАСУМП' (Q Е1)(Е2) $ Р ⊇ К'ЗАСУМ' Е2 . КQЕ1 .

§II6.5 К'ЗАСУМП' (Р Е1) W 2 $ Р ⊇ К'ПРОЦ' (Р Е1) W 2 .

§II6.6 К'ЗАСУМП' (Р Е1) $ Р ⊇ К'ПРОЦ' (Р Е1) .

§II6.7 К'ЗАСУМП' (Е1) $ Р ⊇ К'ЗАСУМ' Е1 .

§II6.8 К'ЗАСУМП' $ Р ⊇ СЧ,0 ;

§II6.9 К'ЗАСУМП'Е1 $ Р ⊇ К $ Р Е1 .

Здесь первый параграф срабатывает в случае простой переменной, второй - формального параметра простой переменной,третий - переменной с индексом,четвертый - стандартной функции, пятый и шестой - процедуры с формальными параметрами и без них .Седьмой параграф осуществляет вхождение в скобки,восьмой появляется в связи с тем, что в процессе трансляции удобно узаконить пустое арифметическое выражение (не разрешаемое синтаксисом АЛГОЛа) со значением 0. Команда МОД,А; приводит к тому,что в следующей команде к исполнительному адресу прибавляется адрес,хранящийся в ячейке А .

§II7.1 К'ЗАСУМ+' ($Е1)+Е2 ⊇ К'ЗАСУМ+' Е2 . СЛ, Е1 ;

§II7.2 К'ЗАСУМ+' Е1 + ($Е2) ⊇ К'ЗАСУМ+' Е1 . СЛ, Е2 ;

§II7.3 К'ЗАСУМ+' Е1 + Е2 ⊇ К'ЗАСУМП'Е1'ЗАСУМ-'. ЗП,0(I5);
К'УВСТ'.
К'ЗАСУМ+' Е2 . СЛ, 0(I5) ; К'УМСТ' .

§II7.4 К'ЗАСУМ+' Е1 ⊇ К'ЗАСУМП' Е1 'ЗАСУМ-' .

§II8.1 К'ЗАСУМ-' Е1 - ($Е2) ⊇ К'ЗАСУМ-'Е1 . ВЧ, Е2 ;

§II8.2 К'ЗАСУМ-' ($Е1)- Е2 ⊇
К'ЗАСУМ+' К'ЗАМ-+' Е2 . . ВЧОБ, Е1 ;

```
§II8.3 R К 'ЗАСУМ-' Е1 - Е2 ≥
        К'ЗАСУМ-' Е1 . ЗП,0(I5) ; К'УВСТ' .
        К'ЗАСУМП' Е2 'ЗАСУМ×' . ВЧОБ,0(I5); К'УМСТ' .
§II8.4 К'ЗАСУМ-' Е1 ≥ К'ЗАСУМП' Е1 'ЗАСУМ×' .

§II9.I К'ЗАМ-+' Е1 - Е2 ≥ Е1 + К'ЗАМ-+' Е2 .
§II9.2 К'ЗАМ-+' Е1 ≥ Е1
```

Процедура 'ЗАСУМ+' - "засылка в сумматор суммы" - открывает последовательность процедур 'ЗАСУМ+' , 'ЗАСУМ-' , 'ЗАСУМ×' , 'ЗАСУМ /', 'ЗАСУМ↑', расположенных по старшинству соответствующих операций. Операция 'ЗАСУМ-' применяется только тогда, когда в аргументе на основном уровне скобочной структуры заведомо нет знаков + , процедура 'ЗАСУМ×' - когда нет знаков + и - и т. д. Порядок старшинства внутри пар (+,-) и (×, /) определяется тем, что при вынесении обратной операции (т.е. - и /) за скобки надо было бы обращать все оставшиеся в скобках прямые операции (**делаем это только** в §II8.2). Описания остальных процедур этой последовательности совершенно аналогичны приведенным. Процедура 'ЗАСУМ×' включает процедуру **'ЗАСУМ/'** , а последняя - процедуру 'ЗАСУМ↑' .

Описание системы команд машины БЭСМ-6 гласит, что операнд 0(I5) приводит к выполнению операции в стэковом режиме. Адрес первой свободной ячейки стэка хранится в индекс-регистре I5. Процедуры 'УВСТ' и 'УМСТ' ("увеличить стэк" и "уменьшить стэк") описываются предложениями:

```
§71 К'УВСТ' ≥ К'СЧ'↑ТС = К'ВЧСЛ' + I, К'СЧ'ТС . . .
                К'УВСТУ' К'ВЧСЛ' - К'СЧ'МС . , К'СЧ'ТС . . .

§72 К'УВСТУ' - EI ≥ К'СЧ'↑МС = К'СЧ'ТС . .
§72.2 К'УВСТУ' EI ≥

§73 К'УМСТ' ≥ К'СЧ'↑ТС = К'ВЧСЛ' - К'СЧ'ТС . , I . .
```

Они нужны для определения того, сколько ячеек **отводится под** стэк. Это число дается счетчиком МС ("максимальный стэк") после конца программирования.

Ограниченный объем доклада не дает возможности описать транслятор более подробно.

Автор выражает глубокую благодарность Л.Б.Морозовой за частые и продолжительные беседы, которые помогли ему в написании данной статьи.

Литература

1. В.Ф.Турчин, С.Н.Флоренцев, В.А.Фисун, Язык РЕФАЛ и его использование для автоматизации программирования, доклад на межвузовской научной конференции "Алгоритмизация и программирование экономических расчетов", М., Май, 1967.

2. В.Ф.Турчин, Метаязык для формального описания алгоритмических языков, в сб. "Цифровая вычислительная техника и программирование", изд-во "Советское радио", М., 1966.

3. В.Ф.Турчин, Метаалгоритмический язык, журн."Кибернетика", 1968 /в печати/.

4. Сообщение о Подмножестве АЛГОЛ-60, в сб. "Современное программирование", изд-во "Советское радио", М., 1966.

5. В.С.Штаркман, Автокод для БЭСМ-6, ИПМ АН СССР, М., 1967.

6. Л.Б.Морозова и др., Транслятор ТАВ, ИПМ АН СССР, М., 1968.

Межведомственная комиссия по математическому обеспечению

Государственного комитета Совета Министров СССР по науке и технике

ИНСТИТУТ КИБЕРНЕТИКИ АН УССР

Научный совет по комплексной проблеме "Кибернетика" АН СССР

Научный совет по кибернетике АН УССР

Киевский дом научно-технической пропаганды

ПЕРВАЯ ВСЕСОЮЗНАЯ КОНФЕРЕНЦИЯ ПО ПРОГРАММИРОВАНИЮ

В. Процессоры с известных языков

К И Е В -1 9 6 8

БФ 01988.Подписано к печати 8.X 1968 г.Изд.№1-53.
Зак.653.Объем 6,4 п.л.Тираж 1050 экз.Цена 45 коп.

Лаборатория офсетной печати ИК АН УССР.
Киев-28,Б.Китаевская,109.