СООБЩЕНИЯ
ОБЪЕДИНЕННОГО
ИНСТИТУТА
ЯДЕРНЫХ
ИССЛЕДОВАНИЙ
Дубна

11 - 5979

В.А. Загинайко

ТРАНСЛЯТОР С ФОРТРАНА НА ЭВМ БЭСМ-4

ЛАБОРАТОРИЯ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ
И АВТОМАТИЗАЦИИ

1971

11 - 5979

В.А. Загинайко

ТРАНСЛЯТОР С ФОРТРАНА НА ЭВМ БЭСМ-4

Загинайко В.А. 11-5979

Транслятор с ФОРТРАНа на ЭВМ БЭСМ-4

Описан транслятор с варианта языка ФОРТРАН, являющегося промежуточным между языками ФОРТРАН-2 и ФОРТРАН-4, включенный в состав математического обеспечения ЭВМ БЭСМ-4.

Сообщения Объединенного института ядерных исследований

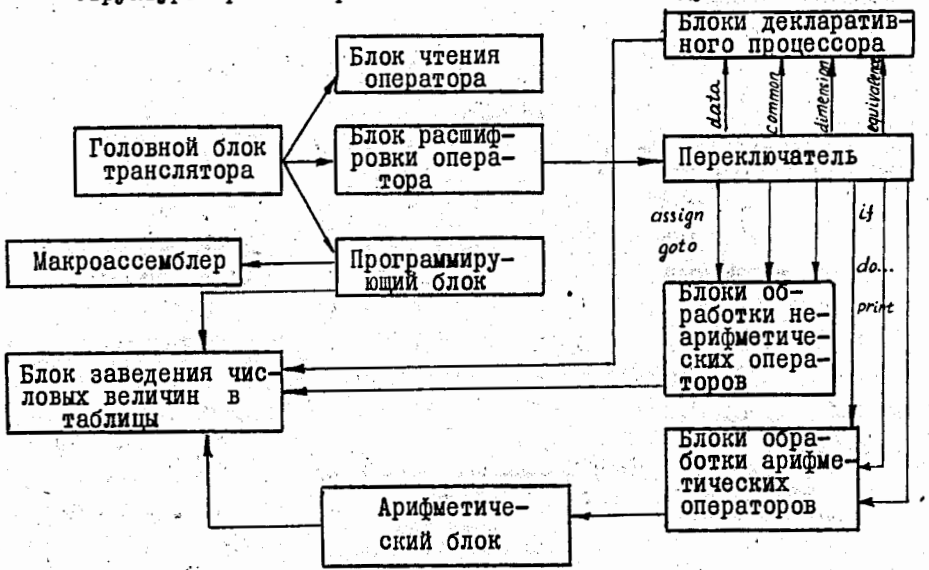
Дубна, 1971

В процессе развития системы программ математического обеспечения ЭВМ БЭСМ-4 автором реализован вариант языка ФОРТРАН, являющийся промежуточным между языками ФОРТРАН-2 и ФОРТРАН-4. Отличия от языка ФОРТРАН-4 состоят в том, что исключено использование логических (булевских) выражений, а также величин комплексных и с двойной точностью. Включение транслятора в состав программ компилирующей системы [1] дает возможность включить в библиотеку системы программы на ФОРТРАНе, отлаженные на других ЭВМ наряду с программами, написанными на языке "Ассемблер"(без использования арифметического блока)[2]. В процессе трансляции осуществляется перевод текста программы с языка ФОРТРАН в текст автокода компилирующей системы, совместимый, в свою очередь, с автокодом "Ассемблер". Пробивку текста на ФОРТРАНе можно производить на телетайпе или на устройстве УПП. Телетайпный текст пробивается тем же способом, что и для автокода "Ассемблер". Текст на УПП, пробитый для данного транслятора БЭСМ-4, может быть введен также и на БЭСМ-6 для трансляции и счета мониторной системой.

Ниже рассматривается структура транслятора, а также методы работы наиболее существенных его частей (арифметического блока и декларативного процессора).

I. Структура и общие характеристики транслятора

Структура транслятора может быть описана следующей схемой:



Стрелки в данной схеме указывают на возможность обращения к подпрограммам соответствующих блоков.

Процесс трансляции осуществляется в два просмотра транслируемой информации. Первый просмотр выполняет головной блок транслятора. Результатом его является программирующая таблица и таблицы числовых величин. Строки программирующей таблицы представляют собой двух- и четырехадресные макрокоманды, являющиеся исходной информацией для второго просмотра, осуществляемого программирующим блоком. При втором просмотре выполняется обращение к блокам макроассемблера.

Использование макроассемблера в процессе выдачи автокодного текста дает возможность менять систему команд выходной программы и использовать матобеспечение БЭСМ-4 [1] для получения рабочих программ на ЭВМ с другими системами команд. Это достигается путем смены таблицы соответствий макроассемблера. Подробнее об этом см.в [3].

Предварительное распознавание оператора при первом просмотре исходной информации на ФОРТРАНе выполняется блоком расшифровки оператора. Это достигается путем выделения начальной последовательности символов. Производится сравнение этой последовательности с текстами начал операторов, хранящихся в таблице переключателя. При совпадении управление через переключатель передается на одну из программ обработки конкретных операторов. Каждая из таких программ состоит из опознающей и программирующей части. При обнаружении ошибки при работе распознающей части управление передается в блок расшифровки оператора для продолжения поиска по текстам переключателя. Наименования операторов идут по возрастанию количества символов примерно в такой последовательности:

do, if, end, data . . . subroutine, equivalence.

При отсутствии в переключателе текста, совпадающего с последовательностью начальных символов оператора, а также в случае, если в этой последовательности встретится символ = (равенство), оператор рассматривается как арифметический. При этом ошибка в распознавании указывает на неверно заданное арифметическое выражение.

Программирующая часть обработки оператора работает в том случае, если опознающей частью не было обнаружено синтаксической ошибки. Программы обработки операторов могут быть выделены в следующие группы. В группу, образующую декларативный процессор, входят программы обработки декларативных операторов

dimension, common, equivalence, data.

В группу обработки арифметических операторов входят те из программ, в которых содержатся обращения к подпрограммам арифметического блока. Они выполняют обработку операторов, содержащих индексные функции (операторы обмена (print, real, write, encode, decode), операторы call , if и арифметические операторы). Подпрограммы обработки остальных операторов (format, do, go to, assign . . .), не содержащие обращения к арифметическому блоку, выделяются в третью группу (блок обработки неарифметических операторов).

В заключение настоящей главы следует привести некоторые характеристики транслятора. Длина его составляет примерно 4500 машинных слов. Скорость трансляции составляет примерно I.2 оператора в секунду.

2. Обработка арифметических выражений в трансляторе

В основу программирования арифметических выражений в трансляторе положен метод выделения внутренних подвыражений, на котором следует подробно остановиться. В дальнейшем будут использоваться понятия, определяемые с помощью нормальной формы Дж.Бэкуса следующим образом:

<скобочная структура> :: = <подвыражение>

<подвыражение> := <параметр> | <скобочное подвыражение> |
<открывающая скобка> <список параметров>
<закрывающая скобка>

<скобочное подвыражение> ::= <открывающая скобка> <подвыражение>
<закрывающая скобка>

<список параметров> ::= <параметр> |
<список параметров> , <параметр>

<функциональная открывающая скобка> ::=
<идентификатор функции> (

<арифметическая открывающая скобка> ::= (

<функциональная закрывающая скобка> ::=)

<арифметическая закрывающая скобка> ::=)

Программирование скобочных структур проводится в два этапа. На первом этапе скобочная структура преобразуется в таблицу таким способом, чтобы каждая закрывающая скобка в таблице имела ссылку (адрес) на соответствующую открывающую и следующую закрывающую скобки. Производится программирование скобочного подвыражения, соответствующего левой открывающей скобке в тексте скобочной структуры. При этом осуществляется выдача макрокоманд в программирующую таблицу. Соответствующее скобочное подвыражение уничтожается и заменяется идентификатором величины, которой присваивается результат вычисления запрограммированного подвыражения.

Затем, используя ссылку на следующую закрывающую скобку, производят программирование соответствующего скобочного подвыражения, замену его идентификатором результата и т.д. В этом случае в программируемых подвыражениях отсутствуют открывающие и закрывающие скобки.

Выполнение порядка действий, соответствующего старшинству операции в арифметическом выражении, достигается путем введения дополнительных скобок, делающих порядок действий явным. Для описания этого метода удобно ввести следующие понятия:

<арифметическое выражение> ::= <операнд со знаком> |
<операнд> <арифметическое выражение><операнд со знаком>|<пусто>

<операнд со знаком> ::= <знак операции> <операнд>

<операнд> ::= <простая переменная> | <переменная с индексом> |
(<арифметическое выражение>)

Процедуру введения скобок можно представить в виде одного из следующих преобразований

I) $\langle R1\rangle\langle Z1\rangle\langle O\rangle\langle Z2\rangle\langle R2\rangle \longrightarrow \langle R1\rangle\langle Z1\rangle \underbrace{((..(}_{n\ раз}\langle O\rangle\langle Z2\rangle\langle R2\rangle$

2) $\langle R1\rangle\langle Z1\rangle\langle O\rangle\langle Z2\rangle\langle R2\rangle \longrightarrow \langle R1\rangle\langle Z1\rangle\langle O\rangle \underbrace{))..)}_{n\ раз}\langle Z2\rangle\langle R2\rangle$

3) $\langle R1\rangle\langle Z\rangle\langle O\rangle\langle Z2\rangle\langle R2\rangle \longrightarrow \langle R1\rangle\langle Z1\rangle\langle O\rangle\langle Z2\rangle\langle R2\rangle$

При этом используются следующие обозначения:

<R1> и <R2> - арифметические выражения,

<Z1> и <Z2> - знак операции или пустой символ ("пусто"),

<O> - операнд.

Тип преобразования определяется в зависимости от знака числа z, вычисляемого по формуле:

$$z = fl(\langle Z1\rangle) - fl(\langle Z2\rangle)$$

введенных

количество введенных скобок определяется модулем числа z ($|z|$). Если $z < 0$, то выполняется преобразование I, если $z > 0$, то преобразование 2, если $z = 0$, то преобразование 3. Функция $fl(\langle Z\rangle)$ для каждого знака операции вычисляет старшинство этой операции. В данном трансляторе функция принимает следующие значения:

$$fl(\langle(\rangle) := fl(\langle)\rangle) := fl(\langle+\rangle) := fl(\langle-\rangle) := fl(\langle)\rangle) := 0$$

$$fl(\langle*\rangle) := fl(\langle/\rangle) := 1 \quad \text{(умножение и деление)}$$

$$fl(\langle**\rangle) := 2 \quad \text{(возведение в степень)}.$$

Подобного рода функции использовались Флойдом [4] для введения скобок в алгоритмах синтаксического анализа.

Описанные алгоритмы используются в блоке обработки арифметических выражений рассматриваемого транслятора в следующей последовательности. Вначале производится выделение скобочных подвыражений, соответствующих функциям. При этом открывающая и закрывающая скобки понимаются следующим образом:

< открывающая скобка > ::= < функциональная открывающая скобка >

< закрывающая скобка > ::= < функциональная закрывающая скобка >

Задача сводится к программированию скобочных подвыражений, в которых отсутствуют функциональные скобки. При этом выполняется введение скобок по алгоритму, описанному выше. После вставления скобок производится выделение скобочных подвыражений, причем скобки понимаются следующим образом:

< открывающая скобка > ::= < арифметическая открывающая скобка >

< закрывающая скобка > ::= < арифметическа закрывающая скобка >

Таким образом, задача программирования арифметического выражения сводится к задаче программирования бесскобочных подвыражений с одинаковым старшинством операций. В результате такого программирования подвыражение преобразуется в серию макрокоманд программирующей таблицы.

Обработка декларативных операторов в трансляторе

Трансляция операторов dimension, common, equivalence проводится в два этапа. На первом этапе эти операторы преобразовываются в таблицу блоков и таблицу эквивалентностей. На втором этапе (в случае окончания обработки декларативных операторов) эти две таблицы преобразуются в последовательность макрокоманд программирующей таблицы. При этом каждая строка таблицы блоков соответствует одному оператору описания массива (bss или common) автокодного текста транслированной программы; строка таблицы эквивалентностей соответствует оператору equ автокодного текста. Трансляция операторов data в программирующую таблицу проводится в один просмотр программы на языке ФОРТРАН.

При трансляции оператора equivalence происходит переупорядочение строк таблиц блоков и эквивалентностей. В частности, если между элементами двух блоков (массивов) имеется соотношение

эквивалентности equivalence $(A(i), B(j))$, один из них исчезает в таблице блоков, в таблице эквивалентностей возникает дополнительная строка эквивалентности начала одного блока элементу другого. Идентификатор результирующего блока получается по формуле:

$$\text{ident} := \underline{if}\ i \geqslant j\ \underline{then}\ A\ \underline{else}\ B.$$

Длина блока вычисляется по формуле:

$$l\,(\text{ident}) = \max(i,j) + \max(l(A)-i,\ l(B)-j).$$

Эквивалентность, возникающая при этом, имеет вид:

$$\text{ident}\ 1 :\ \text{equ}\ ,\ \text{ident} + K\ ,$$

где $K := |i-j|$, а $\text{ident}\ 1 := \underline{if}\ (\text{ident}) = A)\ \underline{then}\ B\ \underline{else}\ A.$

Здесь $l(A)$, $l(B)$ – длины блоков, $\max(\alpha, \beta)$ – максимум α и β.

ЛИТЕРАТУРА

1. В.А.Загинайко. Компилирующая система. II-5923, I97I.

2. В.А.Загинайко, И.Н.Силин. Автокод "Ассемблер". БI-II-45I4, I969.

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

4. Floyd R.W. On Syntactic Analysis and Operator Precedence. Woburn, Mass. 1962.

Рукопись поступила в издательский отдел
4 августа I97I года.