ISSN 2071-2898 (Print)
ISSN 2071-2901 (Online)

Н.Н. Мансуров, Л.К. Эйсымонт

Реализация расширенного языка Рефал на односвязаных списках с кольцевыми цепочками

Рекомендуемая форма библиографической ссылки: Мансуров Н.Н., Эйсымонт Л.К. Реализация расширенного языка Рефал на односвязаных списках с кольцевыми цепочками // Препринты ИПМ им. М.В.Келдыша. 1987. № 20. 35 с.
https://library.keldysh.ru/preprint.asp?id=1987-20

Ордена Ленина
ИНСТИТУТ ПРИКЛАДНОЙ МАТЕМАТИКИ
имени М.В. Келдыша
Академии наук СССР

Н.Н. Мансуров, Л.К. Эйсымонт

РЕАЛИЗАЦИЯ РАСШИРЕННОГО ЯЗЫКА РЕФАЛ НА ОДНОСВЯЗНЫХ СПИСКАХ С КОЛЬЦЕВЫМИ ЦЕПОЧКАМИ

Препринт № 20 за 1987 г.

Москва

Ордена Ленина
Институт прикладной математики
имени М.В.Келдыша
АН СССР

Н.Н.Мансуров, Л.К.Эйсымонт

РЕАЛИЗАЦИЯ РАСШИРЕННОГО ЯЗЫКА РЕФАЛ НА ОДНОСВЯЗНЫХ СПИСКАХ С КОЛЬЦЕВЫМИ ЦЕПОЧКАМИ

Москва
1987

Реализация расширенного языка Рефал на односвязных списках с кольцевыми цепочками. Часть I (списковая память и операторы отождествления). Н.Н.Мансуров, Л.К.Эйсымонт. ИПМ им. М.В.Келдыша АН СССР. Препринт № 20 , М., 1987, стр.32 , библ. 9.

При реализации языка Рефал обрабатываемые символьные выражения обычно отображаются на двусвязную списковую память. Такое представление достаточно удобно и логично с точки зрения языка. Вместе с тем, практика работы на Рефале показывает, что не все его возможности одинаково используются. В частности, чаще выполняется левосторонний, а не правосторонний просмотр выражений. В данной работе сделана попытка использования при реализации Рефала не двусвязной, а односвязной памяти. При этом преследовалась цель более экономно использовать машинную память, а также унифицировать списковую память, применяемую в реализациях языков Лисп, Рефал и Пролог. Унификация списковой памяти позволила бы легко аппаратно реализовать эти языки в одном символьном процессоре. Предлагаемая в работе новая реализация Рефала учитывает также расширение этого языка по типам обрабатываемых данных. При написании работы предполагалось, что читатель знаком с языком Рефал и основными принципами его реализации.

Ключевые слова: обработка символьной информации, списковая память, Лисп, Рефал, алгоритм отождествления.

Содержание

I. Введение.

Начиная с первых реализаций Рефала [I] и до настоящего времени [2,3] обрабатываемая символьная информация представляется в машинной памяти в виде двусвязного списка звеньев, в котором скобки отображаются в отдельные звенья, содержащие ссылки друг на друга (см.рис.I,а).

В реализациях Лиспа символьные выражения обычно представляются в виде односвязного списка с отображением скобок в одно звено и со специальным выделением звеньев скобочных структур (см.рис. I,б).

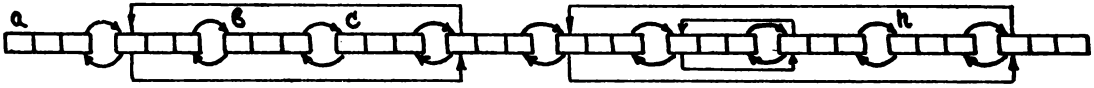
Суть данной работы состоит в том,что предлагается реализовать Рефал на односвязных списках с кольцевыми цепочками (см.рис.I,в). Выбор такой структуры списка объясняется следующими основными причинами.

I. Экономия памяти. Двусвязный список позволяет одинаково эффективно просматривать символьные выражения как слева направо,так и справа налево. Вместе с тем, для него требуется больший объем машинной памяти. Например, для реализаций Рефала и Лиспа на ЕС ЭВМ, структура, изображенная на рис.I,а, требует 30 слов памяти, а структура рис.I,б - лишь I6 слов.

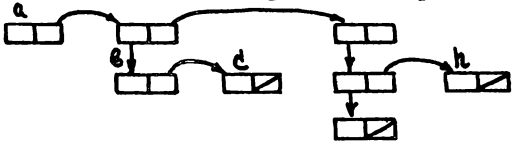
Практика применения Рефала показывает, что обычно обрабатываемые символьные выражения просматриваются в программах слева направо, следовательно, используется лишь одна ссылка звена списка. Например, сбор статистики при работе компилятора Симула-I/АЛМО, написанного на Рефале, показал, что переходов с левого звена на правое - 80%. Оставшиеся 20% переходов - с правого звена на левое, причем значительная их часть приходится на установку правых границ закрытых переменных типа выражение (E-переменных). Для наглядности приведем еще пример - программу, изображенную на рис.2, которая преобразует простые арифметические выражения в последовательности триад. Для эффективного выполнения этой программы двусвязная списковая память не нужна, хотя нужно уметь быстро находить первое справа звено уровня скобочной структуры для эффективного отождествления закрытых E-переменных.

2. Унификация списков для реализаций Лиспа, Рефала, Пролога. Эти языки часто противопоставляют друг другу как взаимоисключающие языки программирования. Это, по-видимому, неправильно. Каждый из них с успехом используется для решения задач различного типа. Более того, опыт их использования говорит о том, что даже для одной и той же задачи на разных этапах ее решения могут

а) двусвязный список (традиционная реализация Рефала)



б) односвязный список (традиционная реализация Лиспа)



в) односвязный список с кольцевыми цепочками (предлагаемая реализация Рефала)

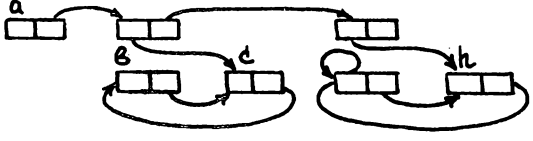


Рис.I. Представления символьного выражения a(bc)(( )h) на списковых памятях с различной организацией.

```
PRIMER   START
         ENTRY  JOB
         EXTRN  CARD,PM
JOB  = K/ТРИАД//1/K/ПОЛИЗ/K/СПАРС/( )K/CARD/....
*  это ввод обрабатываемой строки символов
*- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -*
*_ _ _ лексическая обработка с выделением скобочных структур
СПАРС  (EM) = EM
       (E1)' 'E2 = K/СПАРС/(E1)E2.
       (E1)'('E2 = K/СПАРС/((E1))E2.
       ((E1)E2)')'E3 = K/СПАРС/(E1(E2))E3.
       (E1)SA E2 = K/СПАРС/(E1SA)E2.
*- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -*
* - - - построение дерева разбора в виде обратной польской записи
ПОЛИЗ  E1'+'E2 = K/ПОЛИЗ/E1.K/ПОЛИЗ/E2.'+'
       E1'*'E2 = K/ПОЛИЗ/E1.K/ПОЛИЗ/E2.'*'
        (E1)   = K/ПОЛИЗ/E1.
         E1    = (E1)
*- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
* - - - обход дерева разбора и генерация триад
ТРИАД  SNEX(E1)(E2)SOEY = (SNSO(E1)(E2)) +
                    K/ТРИАД/K/PM/SN.EX(SN)EY.
       SN(SM) = SM
         END
```

Рис.2. Программа преобразования простых арифметических выражений в последовательность триад.

оказаться полезными либо присущая Лиспу эффективность, либо удобство и достаточная эффективность правил преобразований, предоставляемых Рефалом, либо возможности логического вывода, обеспечиваемые Прологом. Известны, например, реализация Пролога на Лисп-машине S 3600 [5], языка ТАО (включающего элементы Лиспа и Пролога) на параллельной Лисп-машине ELIS [6]. С другой стороны, известно, что для некоторых задач на Прологе логический вывод даже вредит, так как нужно просто применять правила преобразования "в одну сторону" [7]. Имеется мнение также о необходимости расширения Лиспа правилами подстановок [8] и практическое подтверждение полезности такого расширения [9]. Опыт использования Рефала тоже показывает, что часто разработка машинных операций на Фортране или ПЛ/I делается из-за недостаточной эффективности Рефала, т.е. когда необходимо описание алгоритма на уровне звеньев (возможно операторное), для чего подошел бы язык типа Лиспа.

При "стыковке" реализаций разных языков наибольшие трудности обычно возникают из-за отличий подходов к распределению и управлению памятью. В этом плане, приведение реализации Рефала к списковым структурам, используемым в реализациях Лиспа и Пролога, может оказаться полезным.

Ясно, что переход при реализации Рефала к односвязной памяти может увеличить время выполнения программ, т.к. менее эффективно будет выполняться просмотр выражений справа налево. Очевидно, что усложнятся алгоритмы реализации и увеличатся количества выполняемых команд при работе Рефал-программ. Однако сокращение необходимых для списков объемов памяти может несколько ограничить это изменение времени выполнения программ, т.к. должны снизиться подкачки листов при работе на виртуальной памяти, более эффективно использоваться кеш-память.

Для практической оценки предлагаемого подхода было решено провести экспериментальную реализацию. В этом же эксперименте было решено исследовать еще одну возможность повышения эффективности - расширение типов данных Рефала, предложенное в [4], а также попытаться использовать 32-разрядные адреса связи в звеньях, а не 24-разрядные, как в существующих реализациях.

Описание реализации разделено на две части: часть I - списковая память и операторы отождествления, часть 2 - операторы замены и результаты исследований реализаций. Часть 2 будет опубликована в середине 1987г.

2. Структура списковой памяти.

Для простоты будем считать, что структура списковой памяти разрабатывается для машин серии ЕС, хотя это не принципиально. При этом предположении структура звена может быть такой, как изображено на рис.3.

Звено - это двойное слово памяти, поэтому все ссылки на звенья - с точностью до двойного слова. Поскольку в машинах ЕС ЭВМ адресация с точностью до байта, то правые 3 разряда ссылки на звено всегда нулевые. В предлагаемом варианте эти 3 разряда используются для хранения тега.

Имеются звенья трех типов: С1, С2, и С3. В звеньях всех типов поле СЛЕД (см.рис.3,а) всегда содержит ссылку на следующее звено. Если звено является самым правым на некотором уровне скобочной структуры, то СЛЕД указывает на первое слева звено этого уровня (см.рис.1,в). Такая ссылка и образует кольцевую цепочку.

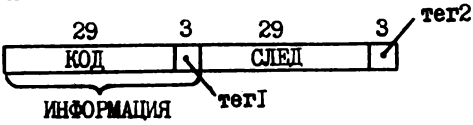
Звено типа С1 в поле ИНФОРМАЦИЯ непосредственно содержит символ (в терминах Лиспа-атом), хранящийся в звене.

Звено типа С2 в поле КОД содержит ссылку на представление символа некоторым односвязным списком. Например, большее $2^{31}-1$ целое число представляется цепочкой звеньев, в каждом из которых располагается меньшее или равное $2^{31}-1$, т.е. число представляется в 2^{31} - системе счисления. Отметим, что такое представление больших чисел имеет преимущество в сравнении с традиционной реализацией Рефала [2], т.к. большое число в данном случае всегда символ. Обычно в прикладных программах требуется возможность быстрого выделения больших чисел в символьных выражениях. В связи с этим, каждое число заключается в скобки, чтобы оно отождествлялось как скобочный терм. В этом случае на каждое большое число тратится два дополнительных звена.

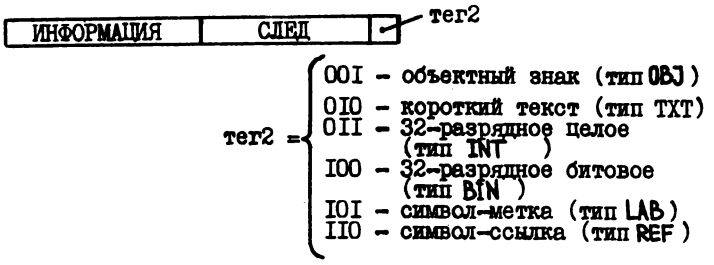
Для краткости изложения, представления символов односвязным списком далее не приводятся.

Звенья типа С3 служат для изображения структурных или функциональных скобок, которые в предлагаемой реализации представляются одинаково. Звено типа С3 является "головой" скобочной структуры и называется головным звеном (см.рис.1,в). Если внутреннее содержимое скобок не пусто, то ссылка в поле КОД звена типа С3 указывает на первое справа звено, которое содержит кольцевую ссылку. Если внутреннее содержимое - пусто, то поле КОД звена содержит ссылку на это же звено.

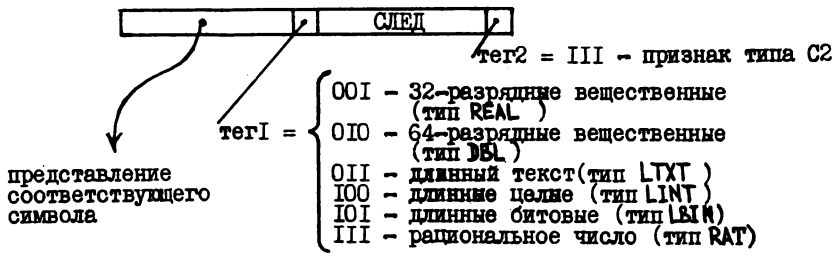
а) Основной формат звена



б) Звенья типа С1



в) Звенья типа С2



с) Звенья типа С3

Рис.3. Формат звена и типы звеньев списковой памяти.

Реализация скобок с использованием звеньев типа СЗ и кольцевых цепочек – одна из основных особенностей реализации. Здесь не использовалось представление скобок, принятое в реализациях языка Лисп, т.к. для эффективной реализации Рефала необходимо иметь доступ сразу к обоим концам содержимого скобочной структуры. Ссылка на последнее звено из звена типа СЗ позволяет, во-первых, быстро находить правую границу закрытой Е-переменной, занимающей крайне правое положение на уровне скобочной структуры в рефал-предложении. Такой тип закрытых Е-переменных типичен, например, в программе на рис.2 закрытые переменные только такие. Во-вторых, кольцевая ссылка позволяет найти первое слева звено уровня скобочной структуры символьного выражения.

Таким образом, предлагаемая реализация скобочных структур использует лишь одно звено (звено типа СЗ), а не два, как в традиционной реализации, однако обеспечивается эффективный доступ к обоим концам внутренности скобок.

Предлагаемая реализация Рефала в остальном, что не связано с отображением обрабатываемой информации в списковую память, повторяет, по-существу, традиционную реализацию [2].

Также, как и в [1], программа на Рефале транслируется в язык сборки, представляющий собой систему команд некоторой абстрактной машины. Команды языка сборки принято называть операторами. Алгоритмы выполнения операторов языка сборки принято описывать на формальном языке более низкого уровня, называемом языком звеньев.

Далее сначала рассмотрим основные операторы языка сборки для отождествления левых частей Рефал-предложений, потом определим язык звеньев, а затем – опишем алгоритмы выполнения основных операторов отождествления.

3. Язык сборки и язык звеньев.

Программа на языке сборки – это последовательность операторов языка сборки, каждый из которых имеет вид:

{метка :} имя-оператора {аргумент 1 {,аргумент 2 {,аргумент 3}}}

где фигурные скобки охватывают конструкции, которые могут отсутствовать.

Аргументами операторов могут быть:

1) C – код символа (байт);

2) S – ссылка на константу из таблицы констант (ТК) рефал-функции

или смещение для относительного перехода на другой оператор языка сборки ;

3) n - номер элемента таблицы элементов (ТЭ) [2] ;

4) t - номер элемента стека трансплантаций (СТ), который используется при выполнении операторов замены.

Список операторов отождествления языка сборки приведен в табл.I. Первая справа колонка этой таблицы содержит номера страниц данной работы, где кратко описаны операторы и приведены алгоритмы их выполнения. В целом, эти операторы похожи на операторы отождествления традиционных реализаций Рефала. Наибольшее отличие имеет место лишь в операторах перехода на отождествление уровня скобочной структуры: NH, NHB, NHR и NHRM .

Рассмотрим теперь язык звеньев. Язык звеньев - это операторный язык, каждый оператор которого может иметь метку-идентификатор. Операторы могут быть следующих типов.

I. Оператор присваивания имеет вид

левая часть = правая часть

где "левая часть" - это индексированная или простая переменная, либо ссылка на поле какой-либо структуры вида имя поля (переменная) и переменная служит указателем на структуру.

Для звеньев используются такие имена полей, какие указаны на рис.3, а имена полей других структур далее вводятся в разделе 4.I.

Правая часть оператора присваивания - это либо переменная, либо ссылка на поле, либо арифметическое выражение с операндами таких типов.

2. Оператор перехода может быть следующих трех типов:

1) перейти метка

2) вызвать метка

3) вернуться

Операторы вызвать и вернуться введены для связи с подпрограммами.

3. Условный оператор имеет вид

если (условие) оператор

где условие - это переменная, ссылка на поле или логическое выражение с операндами этих типов.

Еще имеется два оператора, которые можно определить через введенные выше операторы.

4. Структурный условный оператор имеет вид

если (условие) то
оператор
оператор

Таблица I. Основные операторы языка сборки для реализации Рефала на односвязных вписках с кольцевыми цепочками

№	Оператор	Выполняемое оператором действие	стр.
I	COBJ C	Левостороннее отождествление объектного знака C	2I
2	CTXT S	–"– короткого текста ТК(S)	
3	CINT S	–"– 32-разрядного целого ТК(S)	
4	CBIN S	–"– 32-разрядного битового ТК(S)	
5	CLAB S	–"– символа-метки ТК(S)	
7	CREAL S	–"– 32-разрядного вещественного ТК(S)	
8	CDBL S	–"– 64-разрядного вещественного ТК(S)	
9	CLTXT S	–"– длинного текста ТК(S)	
I0	CLINT S	–"– длинного целого ТК(S)	2I
II	CLBIN S	–"– длинного битового ТК(S)	
I2	CRAT S	–"– рационального числа ТК(S)	
I3 ÷ 23	RCOBJ ÷ RCRAT	Как и операторы COBJ ÷ CRAT , только отождествление правостороннее	2I
24	CHECK	Проверка завершения отождествления дыры	20
25	BTN	Левостороннее отождествление со сменой дыры	2I
26	BTO	Левостороннее отождествление без смены текущей дыры	22
27,28	RBTN, RBTO	Как и BTN , BTO, только правостороннее отождествление	22
29	NHB n	Начало отождествления скобочного уровня	I6
30	NH n_1, n_2	Продолжение отождествления скобочного уровня, у которого отождествлены только левые элементы	I7
3I	NHR n_1, n_2	Продолжение отождествления скобочного уровня, у которого отождествлены только правые элементы	I8
32	NHRM n_1, n_2, n_3	Продолжение отождествления скобочного уровня, у которого отождествлены левые и правые элементы	I9
33	SYM	Левостороннее отождествление главного вхождения S –переменной	23
34	OSYM n	Левостороннее отождествление повторного вхождения S –переменной	23
35,36	RSYM, ROSYM	Как и SYM , OSYM , только отождествление правостороннее	23
37	TERM	Левостороннее отождествление главного вхождения W –переменной	24
38	RTERM	Правостороннее отождествление главного вхождения W –переменной	24
39	EXPR	Отождествление закрытой E-переменной	24
40	OEXPR n	Левостороннее отождествление повторного вхождения E-переменной	25
4I	ROEXPR n	Правостороннее отождествление повторного вхождения E-переменной	
42,43, 44	PL, LEN1, LEN	Левостороннее отождествление открытых E-переменных	26
45,46	RPL, RLEN	Правостороннее отождествление открытых E-переменных	27
47	PBT S	Подготовка перехода на следующее предложение рефал-функции при неотождествлении	20

```
        ⋮
    оператор
     конец
```

5. Оператор цикла имеет вид

```
    пока (условие) то
      оператор
        ⋮
      оператор
       конец
```

Для краткости описания алгоритмов будем также пользоваться операторами READ и WRITE вида

READ имя структуры (переменная I, ..., переменная N)

что эквивалентно

```
переменная I = имя поля I структуры (указатель в структуре)
        ⋮
переменная N = имя поля N структуры (указатель в структуре)
```

WRITE имя структуры (выражение I, ..., выражение)

что эквивалентно

```
имя поля I структуры (указ. в структуре) = выражение I
        ⋮
имя поля N структуры (указ. в структуре) = выражение N
```

Будут также использоваться три "макрокоманды": SHB1 - сдвиг указателя B1 вправо (см.4.I); SHB2 - сдвиг указателя B2 влево; EMBR - распаковка пустой скобочной структуры.

Макрокоманда SHB1

```
если (B1=B2) перейти НЕОТ
B1 = СЛЕД (B1)
```

Макрокоманда EMBR

```
СЛЕД(ФВС) = ФВС
 B1 = ФВС
 B2 = ФВС
 СВТ = ФВС
```

Макрокоманда SHB2

```
P1 = B1
P2 = СЛЕД (P1)
пока (P2≠B2) то
P1 = P2
P2 = СЛЕД(P2)
 конец
B2=P1
```

4. Алгоритмы выполнения основных операторов отождествления языка сборки.

4.1. Основные структуры данных и общие положения.

Как и в традиционной реализации Рефала, будем считать, что операторы языка сборки выполняются некоторым интерпретатором. Для понимания материала, изложенного далее, достаточно знать об интерпретаторе лишь следующее.

1. Каждый оператор языка сборки реализуется программой на языке звеньев, помеченной именем оператора. Номер (адрес) выполняемого оператора всегда содержится в переменной СЧО.

2. Переход на выполнение нового оператора языка сборки в случае его успешного завершения производится по оператору перейти СЛЕД.

3. Выход из программы выполнения оператора языка сборки в случае его неуспешного завершения производится по оператору перейти НЕОТ.

Для запоминания адресов звеньев в процессе отождествления в данной реализации, так же как и в традиционной, используется таблица элементов ТЭ, а указатель на ее первый свободный элемент содержится в переменной НЭЛ.

Границы просматриваемого при отождествлении участка списковой памяти отображаются в переменных В1 и В2:

В1 – содержит адрес звена, предшествующего первому слева звену участка;

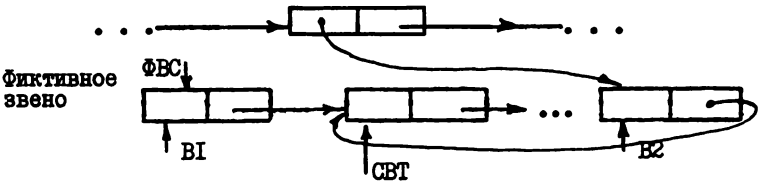
В2 – содержит адрес последнего звена участка.

Заметим, что в традиционной схеме реализации В2 – содержит адрес следующего за последним звеном участка звена.

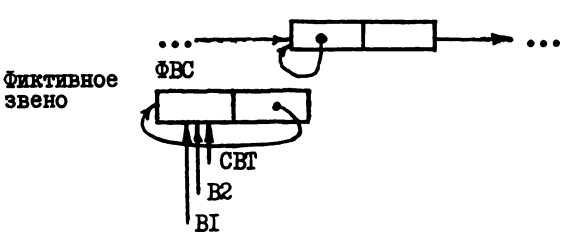
Установка В1 и В2 перед началом отождествления очередного уровня скобочной структуры называется распаковкой скобочной структуры. Положения В1 и В2 изображены на рис.4: случай а – непустая скобочная структура, случай б – пустая скобочная структура. На этих рисунках появились еще две переменные: ФВС – указатель на фиктивное звено, которое в поле СЛЕД содержит ссылку на первое слева звено уровня скобочной структуры; СВТ – указатель, который всегда установлен на звено СЛЕД (ФВС).

Введение фиктивного звена сделано из-за того, что в данной реализации нет звена, соответствующего левой скобке, хотя при отождествлении оно очень полезно и упрощает алгоритмы. Таким образом, в данной реализации пока отождествление скобочной структуры не началось, звена для левой скобки нет, а когда начинается

а) Начальная установка для непустой скобочной структуры



б) Начальная установка для пустой скобочной структуры



в) Структура стека переходов (СП)

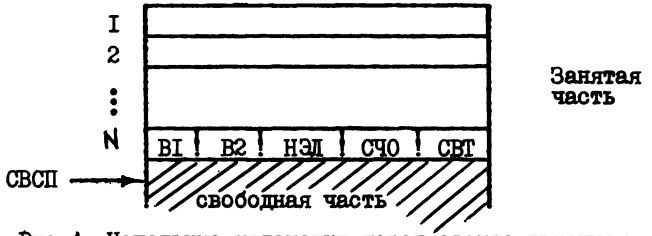
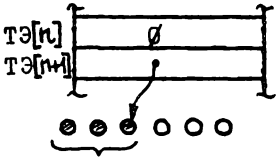


Рис.4. Начальные установки перед отождествлением уровней скобочных структур и структура стека переходов

а) Левостороннее отождествление



б) Правостороннее отождествление

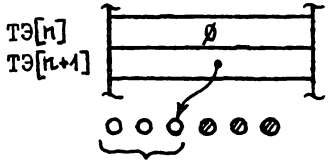


Рис.5. Начальная установка элементов ТЭ для Е-переменной при различных направлениях отождествления.

отождествление – то оно искусственно вводится в виде фиктивного звена. Фиктивное звено для всех скобочных структур одно и то же.

Поскольку функциональные и структурные скобки представляются одинаково, то начальная установка в начале выполнения рефал-функции в точности такая, как на рис.4,а и рис.4,б.

Изменение представления скобочных структур также несколько усложнило переход на отождествление дыр [2] по сравнению с традиционной реализацией. Этот вопрос рассматривается далее в разделах 4.I.I и 4.I.2.

4.I.I. Операторы установки границ отождествляемого участка скобочной структуры.

Каждой дыре поставим в соответствие содержащий ее ближайший уровень скобочной структуры левой части рефал-предложения. Далее будем говорить лишь об отождествлении уровня скобочной структуры, которое начинается после отождествления скобок, а завершается тогда, когда все элементы уровня отождествлены. Отождествление одного уровня скобочной структуры может прерываться отождествлением другого уровня, а потом вновь возобновляться. При переходе на отождествление уровня скобочной структуры возможны четыре случая: I) отождествления еще не было ; 2) отождествлялись лишь крайние слева элементы уровня, т.е. по отношению к начальной установке на скобочную структуру могло переместиться лишь BI ; 3) отождествлялись лишь крайние справа элементы уровня, т.е. могло переместиться лишь B2 ; 4) отождествлялись элементы с обоих концов уровня, т.е. могли переместиться BI и B2.

Для каждого из этих случаев в язык сборки введен соответствующий оператор. Рассмотрим их.

Оператор NHB , где ТЭ[n] – содержит адрес головного звена скобочной структуры.

Приведем пример использования этого оператора. Пусть имеется следующая левая часть рефал-предложения с расставленными проекционными номерами $\overset{1}{A}\ (\overset{2}{B}\ )\ \overset{3}{C}$. На языке сборки она представляется так: COBJ 'A' ; BTO ; COBJ 'C'; CHECK ; NHB 2 ; COBJ 'B'; CHECK .

Алгоритм выполнения оператора NHB является фактически начальной установкой перед отождествлением скобочной структуры (см. рис.4а,б).

```
NHB  BI = ТЭ[n]
     если (КОД(BI)=BI) перейти NHBI
     значит скобочная структура непустая
```

```
        B2 = КОД(BI)
        BI = ФВС
        СЛЕД(ФВС) = СЛЕД(B2)
        СВТ = СЛЕД(B2)
        перейти СЛЕД
_ _     значит скобочная структура пустая
NHB1    EMBR
        перейти СЛЕД
```

Оператор NH n_1, n_2, где ТЭ$[n_1]$ содержит адрес звена скобочной структуры, расположенного слева от ее еще неотождествленного участка, а ТЭ$[n_2]$ – содержит адрес головного звена структуры.

Этот оператор применяется, например, в следующем случае. Пусть левая часть с проекционными номерами имеет вид A (B) C, тогда на языке сборки она представляется так: COBJ 'A'; BTN; COBJ 'B'; CHECK; NH 2,Ø; COBJ 'C'; CHECK . Здесь учитывается, что адрес головного звена функциональной скобки хранится в ТЭ[Ø].

Цель выполнения оператора NH, как и описанного выше оператора NHB, тоже распаковка скобочной структуры, но при условии, что эту скобочную структуру уже ранее отождествляли с левого конца, т.е. указатель BI мог переместиться. Такая неопределенность в отношении указателя BI имеет место из-за переменных E-типа, значениями которых могут быть, в частности, пустые выражения. В случае NH такая неопределенность не вносит осложнений, как и в традиционной схеме реализации, т.к. BI указывает на звено за пределами неотождествленного участка скобочной структуры. Далее при рассмотрении оператора NHR мы увидим, что аналогичный случай неопределенности перемещения указателя B2 необходимо специально учитывать в алгоритме.

Алгоритм выполнения NH таков:

```
NH      BI = ТЭ[n2]
        если (КОД(BI)=BI) перейти NH1
_ _     значит скобочная структура непустая
        B2 = КОД(BI)
        BI = ТЭ[n1]
        СВТ = СЛЕД(B2)
        перейти СЛЕД
_ _     значит скобочная структура пустая
NH1     EMBR
        перейти СЛЕД
```

Прежде чем рассматривать следующий оператор, разберем, что

такое для данной реализации пустое выражение в качестве значения Е-переменной и как оно связано с направлением отождествления. Если ТЭ[n] и ТЭ[n+1] соответствуют Е-переменной, то при пустом значении переменной всегда

ТЭ[n] = 0

ТЭ[n+1] = адрес "левого"звена,

где адрес "левого" звена при левостороннем отождествлении есть адрес последнего уже отождествленного звена перед началом отождествления Е-переменной, а при правостороннем отождествлении - адрес первого справа еще неотождествленного звена перед началом отождествления Е-переменной (см.рис.5 а,в).

Оператор NHR n_1,n_2, где ТЭ[n_1] содержит адрес головного звена скобочной структуры, а в ТЭ[n_2] хранится адрес звена скобочной структуры, находящегося правее еще неотождествленного участка этой структуры, либо ноль, тогда в ТЭ[n_2+1] хранится адрес первого справа звена еще неотождествленного участка. ТЭ[n_2] = Ø, если n_2 - левый проекционный номер Е-переменной и ее значение пусто.

Приведем пример использования оператора. Пусть левая часть рефал-предложения с расставленными проекционными номерами имеет вид $A\ (\ B\ )\ C$, тогда на языке сборки она будет представлена так:
RCOBJ 'C'; RBTN; COBJ 'B'; CHECK; NHR Ø,2; COBJ 'A'; CHECK.

При распаковке, выполняемой оператором NHR, необходимо учитывать, что скобочная структура может быть пустой, а также то, что в зависимости от того, как установлено ТЭ[n_2], надо передвигать В2 влево или нет. Алгоритм выполнения оператора NHR описывается так:

```
NHR  BI = ТЭ[n1]
     если (КОД(BI)=BI) перейти NHR1
__   значит скобочная структура непустая
     СВТ = СЛЕД (КОД(BI))
     СЛЕД(ФВС) = СВТ
     BI = ФВС
     если (ТЭ[n2] ≠ 0) то
__   значит В2 устанавливается на ТЭ[n2] и сдвигается влево
     B2 = ТЭ[n2]
     SHB2
     перейти СЛЕД
     конец
```

```
        _ _  значит В2 устанавливается на ТЭ
        В2 = ТЭ[n2+1]
        перейти СЛЕД
NHR1    EMBR
        перейти СЛЕД
```

Оператор NHRM n_1, n_2, n_3 , где ТЭ$[n_1]$ – содержит адрес звена, предшествующего неотождествленному участку скобочной структуры, ТЭ$[n_2]$ – содержит адрес звена, находящегося справа от неотождествленного участка либо ноль, а ТЭ$[n_3]$ – адрес головного звена скобочной структуры.

Пример использования оператора. Пусть имеем A (B) C, тогда (2 3 4 3 1) представление этой конструкции на языке сборки таково: RCOBJ 'C'; COBJ 'A'; BTN ; COBJ 'B'; CHECK ; NHRM 3,1,Ø ; CHECK .
Алгоритм выполнения оператора NHRM таков.

```
NHRM    СВТ = ТЭ[n3]
        если (КОД(СВТ) = СВТ) перейти NHRM1
_ _     значит скобочная структура непустая
        ВI = ТЭ[n1]
        СВТ = СЛЕД (КОД(СВТ))
        СЛЕД(ФВС) = СВТ
        если (ТЭ[n2] ≠ 0) то
        _ _ значит В2 устанавливается на ТЭ[n2] и сдвигается
                                                влево
            В2 = ТЭ[n2]
            SHB2
            перейти СЛЕД
        _ _ конец
        _ _ значит В2 устанавливается на ТЭ
        В2 = ТЭ[n2+1]
        перейти СЛЕД
_ _     значит скобочная структура пустая
NHRM1   EMBR
        перейти СЛЕД
```

4.I.2. Выбор нового отождествляемого участка скобочной структуры после возникновения ситуации "неотождествление" и установка его границ.

Кроме явной установки по операторам NHB , NH , NHR и NHRM установка границ отождествляемых участков производится через стек переходов (СП) [2] , обращение к которому происходит при выполнении оператора перейти НЕОТ.

Структура одного элемента СП изображена на рис.4,в. Указатель СВСП всегда указывает на первый свободный элемент СП. Поля ВІ,В2,НЭЛ,СЧО – имеют такой же смысл, что и в традиционной реализации Рефала с той лишь разницей, что В2 указывает на первое справа звено неотождествленного участка. Поле СВТ необходимо для того, чтобы восстанавливать поле СЛЕД (ФВС) при установке границ отождествляемого участка скобочной структуры.

Заведение нового элемента СП происходит при подготовке перехода на $i+1$ предложение рефал-функции в процессе отождествления i -го предложения. Подготовка такого перехода осуществляется при выполнении оператора РВТ, он будет рассмотрен ниже. Другой случай заведения нового элемента СП – когда начинается левостороннее или правостороннее отождествление открытой переменной Е-типа. Соответствующие операторы будут разобраны далее в разделе 4.6.2.

Оператор PBT S , где S – смещение первого оператора $i+1$ -го предложения относительно данного оператора РВТ S

```
РВТ  WRITE   СП(ВІ,В2,НЭЛ,СЧО+S ,СВТ)
     СВСП = СВСП + I
     перейти СЛЕД
```

Считывание полей элемента СП для установки границ отождествляемого участка скобочной структуры происходит после возникновения ситуации "неотождествление". Действия, выполняемые в этом случае, таковы:

```
НЕОТ  если (СВСП = I) перейти ОСТАНОВ
_ _ значит в стеке переходов есть элемент, следовательно
_ _ можно выбрать новый участок отождествления
      СВСП = СВСП-I
      READ  СП(ВІ,В2,НЭЛ,СЧО,СВТ)
      СЛЕД(ФВС) = СВТ
      перейти СЛЕД
```

Метка ОСТАНОВ соответствует блоку аварийного завершения работы интерпретатора языка сборки.

В завершение рассмотрим оператор СНЕСК, который проверяет, есть ли еще неотождествленные звенья в участке скобочной структуры. Этот оператор применяется всегда в тех случаях, когда таких звеньев быть не должно (см.примеры в разделе 4.I.I).

```
СНЕСК если (ВІ ≠ В2) перейти НЕОТ
_ _ значит неотождествленных звеньев больше нет
     перейти СЛЕД
```

Далее рассмотрим алгоритмы реализации остальных операторов отождествления. Эти алгоритмы не сильно отличаются от алгоритмов выполнения аналогичных операторов в традиционной схеме реализации Рефала.

4.2. Отождествление символов.

Приведем лишь алгоритмы для COBJ , RCOBJ , CLINT , поскольку алгоритмы для операторов CTXT ÷ CRAT и RCTXT - RCRAT отличаются незначительно.

Оператор COBJ c .

```
COBJ  SHB1
      если ((ТЕГ2(ВI) ≠ "OBJ") ∨ (КОД(ВI) ≠ С)) перейти НЕОТ
      ТЭ [НЭЛ] = ВI
      НЭЛ = НЭЛ+I
      перейти СЛЕД
```

Оператор RCOBJ c .

```
RCOBJ если ((ВI=В2) ∨ (ТЕГ2(В2)≠"OBJ") ∨ (КОД(В2)≠С)) перейти НЕОТ
      ТЭ [НЭЛ] = В2
      НЭЛ = НЭЛ+I
      SHB2
      перейти СЛЕД
```

Оператор CLINT s .

```
CLINT SHB1
      если ((ТЕГ2(ВI)≠"C2") ∨ (ТЕГI(ВI)≠"LINT")) перейти НЕОТ
  _ _ далее сравниваются собственно длинные числа
      вызвать СРАВНЕНИЕ-СИМВОЛ-С-КОНСТАНТОЙ
      ТЭ [НЭЛ] = ВI
      НЭЛ = НЭЛ+I
      перейти СЛЕД
```

4.3. Отождествление скобок.

Для отождествления скобок имеется четыре основных оператора: BTN , BTO , RBTN и RBTO .

Оператор BTN для левостороннего отождествления скобок со сменой отождествляемого уровня скобочной структуры.

```
BTN SHBI
      если (ТЕГ2(ВI)≠"C3") перейти НЕОТ
      ТЭ [НЭЛ] = ВI
      НЭЛ = НЭЛ+I
  _ _ далее следует распаковка скобочной структуры
      если (КОД(ВI)=ВI) перейти BTN1
```

```
   _ _ значит скобочная структура непустая
       СЛЕД(ФВС) = СЛЕД (КОД(В1))
       В2 = КОД(В1)
       В1 = ФВС
       СВТ = СЛЕД(ФВС)
       перейти СЛЕДУЮЩИЙ
   _ _ значит скобочная структура пустая
 ВТN1  EMBR
       перейти СЛЕД
```

Оператор ВТО, для левостороннего отождествления скобок без смены отождествляемого уровня скобочной структуры.

```
 ВТО   SHB1
       если (ТЕГ2(В1)≠"СЗ") перейти НЕОТ
       ТЭ [НЭЛ]  = В1
       НЭЛ = НЭЛ+1
       перейти СЛЕД
```

Оператор RBTN , для правостороннего отождествления скобок со сменой отождествляемого уровня скобочной структуры.

```
 RBTN  если ((В1=В2) ∨ (ТЭГ2(В2)≠"СЗ")) перейти НЕОТ
       ТЭ [НЭЛ] = В2
       НЭЛ = НЭЛ+1
   _ _ далее следует распаковка скобочной структуры
       если (КОД(В2)=В2) перейти RBTN1
   _ _ значит скобочная структура непустая
       СЛЕД(ФВС) = СЛЕД(КОД(В2))
       В2 = КОД(В2)
       В1 = ФВС
       СВТ = СЛЕД(ФВС)
       перейти СЛЕД
   _ _ значит скобочная структура пустая
 RBTN1 EMBR
       перейти СЛЕД
```

Оператор RBTO, для правостороннего отождествления скобок без смены отождествляемого уровня скобочной структуры.

```
 RBTO  если ((В1=В2) ∨ (ТЭГ2(В2)≠"СЗ")) перейти НЕОТ
       ТЭ [НЭЛ] = В2
       НЭЛ = НЭЛ+1
       SHB2; перейти СЛЕД
```

Кроме этих операторов, имеются еще дополнительные операторы, введенные для наиболее часто встречающихся конструкций: для

левостороннего и правостороннего отождествления пустой скобки ( ) – операторы BTE и RBTE; для отождествления E–переменной в скобках вида (Ei) – операторы BEX и RBEX.

4.4. Отождествление S –переменных.

Для отождествления S –переменных имеется четыре оператора: SYM , RSYM и OSYM , ROSYM .

Оператор SYM .

```
SYM  SHB1
     если (ТЭГ2(BI)="СЗ") перейти НЕОТ
     ТЭ [НЭЛ] = BI
     НЭЛ = НЭЛ+I
     перейти СЛЕД
```

Оператор RSYM .

```
RSYM если ((BI=B2) v (ТЕГ2(B2)="СЗ")) перейти НЕОТ
     ТЭ [НЭЛ] = B2
     НЭЛ = НЭЛ+I
     SHB2
     перейти СЛЕД
```

Оператор OSYM n, где ТЭ[n] – адрес звена, являющегося значением главного вхождения этой S –переменной.

```
OSYM SHB1
     PI=BI
     P2=ТЭ[n]
     если ((ТЭГI(PI)≠ТЕГ2(P2)) v (ТЕГ2(PI)≠ТЕГ2(P2)) перейти
                                                      НЕОТ
  _ _ далее непосредственное сравнение символов
     вызвать СРАВНЕНИЕ–СИМВОЛОВ
     ТЭ [НЭЛ] = BI
     НЭЛ = НЭЛ+I
     перейти СЛЕД
```

Оператор ROSYM n , правостороннее проектирование повторного вхождения, ТЭ[n] – адрес главного вхождения.

```
ROSYM если (BI=B2) перейти НЕОТ
      PI=B2
      P2=ТЭ[n]
     если ((ТЕГI(PI)≠ТЕГI(P2))v(ТЕГ2(PI)≠ТЕГ2(P2)) перейти НЕОТ
     вызвать СРАВНЕНИЕ–СИМВОЛОВ
     ТЭ [НЭЛ] = B2
     НЭЛ = НЭЛ+I
     SHB2
     перейти СЛЕД
```

4.5. Отождествление W -переменных.

Отождествление главных вхождений W -переменных производят операторы TERM и RTERM . Отождествление повторного вхождения W -переменной выполняется оператором отождествления повторного вхождения E-переменной OEXPR , который описан в следующем разделе.

Оператор TERM .

```
TERM  SHB1
      ТЭ [НЭЛ] = BI
      ТЭ [НЭЛ+I] = BI
      НЭЛ = НЭЛ+2
      перейти СЛЕД
```

Оператор RTERM .

```
RTERM если (BI=B2) перейти НЕОТ
      ТЭ [НЭЛ] = B2
      ТЭ [НЭЛ+I] = B2
      НЭЛ = НЭЛ+2
      SHB2
      перейти СЛЕД
```

4.6. Отождествление E-переменных.

4.6.I. Закрытые E-переменные и повторные вхождения.

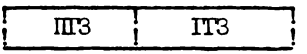
Закрытые E-переменные отождествляются оператором EXPR. Он используется при левостороннем и правостороннем отождествлении.

Оператор EXPR .

```
EXPR  если (BI=B2) перейти EXPR1
      BI = СЛЕД(BI)
      ТЭ [НЭЛ] = BI
      ТЭ [НЭЛ+I] = B2
      НЭЛ = НЭЛ+2
      перейти СЛЕД
EXPR1 ТЭ [НЭЛ]= ∅
      ТЭ [НЭЛ+I] = BI
      НЭЛ = НЭЛ+2
      перейти СЛЕД
```

Повторное вхождение E-переменной отождествляется операторами OEXPR и ROEXPR . Для краткости изложения рассмотрим лишь оператор OEXPR .

При сравнении символьных выражений в данной реализации приходится использовать дополнительный стек СУВ, в нем хранятся адреса головных звеньев скобочных структур, просмотр которых еще не завершен. Элемент стека СУВ имеет вид



где ПТЗ - адрес головного звена в повторном вхождении, ГТЗ - адрес головного звена в главном вхождении. Указатель СВСУВ всегда

указывает на первый свободный элемент стека СУВ.

Оператор OEXPR n , где ТЭ[n] - адрес самого левого звена значения главного вхождения или ноль, если это значение - пустое выражение.

Ниже приведен алгоритм выполнения OEXPR. В этом алгоритме PI продвигается слева направо по значению главного вхождения, а указатель P3 установлен на самое правое звено этого значения.

```
OEXPR  PI = ТЭ[n]
       если (PI=Ø) то
       _ _ значение главного вхождения пустое
           ТЭ[НЭЛ] = 0
           ТЭ[НЭЛ+1] = BI
           НЭЛ = НЭЛ+2
           перейти СЛЕД
           конец
     _ _ значение главного вхождения непустое
           SHB1
           ТЭ[НЭЛ] = BI
           вызвать СРАВНЕНИЕ-ТЕРМОВ
     _ _ пройден первый терм повторного вхождения
           P3 = ТЭ[n+1]
     _ _ проход по Ø-му уровню скоб.стр. главного вхождения
           пока (PI≠P3) то
           SHB1
           PI = СЛЕД(PI)
           вызвать СРАВНЕНИЕ-ТЕРМОВ
           конец
         ТЭ[НЭЛ+1] = BI
     _ _ обход остальных непустых уровней скободных структур
         главного вхождения
         пока (СВСУВ > 1) то
           СВСУВ = СВСУВ-1
           READ СУВ (BI,PI)
           P4 = КОД(BI)
           P3 = КОД(PI)
           BI = СЛЕД(P3)
           PI = СЛЕД(P2)
           вызвать СРАВНЕНИЕ-ТЕРМОВ
           пока (PI≠P3) то
              если (BI=P4) перейти НЕОТ
```

```
                BI = СЛЕД(BI)
                PI = СЛЕД(PI)
                вызвать СРАВНЕНИЕ-ТЕРМОВ
                конец
        _ _  сравнение выражений завершено
             BI = ТЭ[НЭЛ+I]
             НЭЛ = НЭЛ+2
             перейти СЛЕД
```

Программа СРАВНЕНИЕ-ТЕРМОВ имеет следующий вид.

```
СРАВНЕНИЕ-ТЕРМОВ если ((ТЕГ(PI)≠ТЕГI(BI))v(ТЕГ2(PI)≠ТЕГ2(BI))
                                                  перейти НЕОТ
                  если (ТЕГ2(PI)="СЗ") перейти СКОБКА
        _ _ значите звено является символом
                  P2 = BI
                  вызвать СРАВНЕНИЕ-СИМВОЛОВ
                  вернуться
        _ _ значит звено - голова скобочной структуры
        СКОБКА  если (КОД(PI)≠PI) то
                _ _ значит основное вхождение - непустая скоб.
                                                        стр.
                  если (КОД(BI)=BI) перейти НЕОТ
                  WRITE СУВ(BI,PI)
                  СВСУВ = СВСУВ+I
                  вернуться
        _ _ значит вхождение - пустая скоб.структура
                  если (КОД(BI)≠BI) перейти НЕОТ
                  вернуться
```

4.6.2. Открытые Е-переменные.

Левостороннее отождествление открытых Е-переменных выполняют операторы: PL - заведение элемента стека переходов СП и установка пустого значения Е-переменной; LEN1 - выполнение первого удлинения Е-переменной ; LEN - выполнение 2-го и т.д. удлинений Е-переменной.

Правостороннее отождествление открытых Е-переменных выполняют операторы: RPL - заведение элемента СП и установка пустого значения Е-переменной; RLEN - выполнение последующих удлинений.

Оператор PL .

```
PL    WRITE СП (BI,B2,НЭЛ,СЧО+I,СВТ)
            СВСП = СВСП+I
            ТЭ[НЭЛ] = ∅
```

```
            ТЭ [НЭЛ+I] = BI
            НЭЛ = НЭЛ+2
            СЧО = СЧО+2
            перейти СЛЕД
```

Операторы LEN1, LEN .

```
   LEN1     ТЭ [НЭЛ] = СЛЕД(BI)
            СЧО(СВСП) = СЧО+I
            СЧО = СЧО+I
      _ _   далее перемещение BI и установка правой ссылки
                                                выражения
   LEN      SHB1
            ТЭ [НЭЛ+I] = BI
            BI(СВСП) = BI
            СВСП = СВСП+I
            НЭЛ = НЭЛ+2
            перейти СЛЕД
```

Оператор RPL .

```
   RPL      WRITE СП(BI,B2,НЭЛ,СЧО+I, СВТ)
            СВСП = СВСП + I
            ТЭ [НЭЛ] = Ø
            ТЭ [НЭЛ+I] = B2
            НЭЛ = НЭЛ+2
            СЧО = СЧО+I
            перейти СЛЕД
```

Оператор RLEN .

```
   RLEN     если (BI=B2) перейти НЕСТ
            ТЭ [НЭЛ] = B2
            SHB2
            ТЭ [НЭЛ+I] = B2
            B2(СВСП) = B2
            СВСП = СВСП+I
            перейти СЛЕД
```

5. Оценки эффективности реализации операторов отождествления на машинах серии ЕС.

Рассмотренные алгоритмы отождествления реализованы на машине ЕСІ045. В настоящее время ведется отладка. В табл.2 приводятся сравнительные оценки эффективности реализации описанных выше операторов отождествления и аналогичных им операторов, имеющихся в традиционной реализации Рефал/2 на ЕС І045.

Таблица 2. Оценки эффективности реализации операторов отождествления на машине ЕС-1045.

Обозначения:

$N^1_{ср}$ - среднее число команд по различным ветвям в предложенной реализации

$N^2_{ср}$ - среднее число команд по различным ветвям в традиционной реализации

$T^I_{ср}$ - среднее время выполнения оператора в предложенной реализации (мкс)

$T^2_{ср}$ - среднее время выполнения оператора в традиционной реализации (мкс)

L - число звеньев между BI и B2.

Предложенная реализация $\frac{N^1_{ср}}{T^1_{ср}}$			Традиционная реализация $\frac{N^2_{ср}}{T^2_{ср}}$			$\frac{T^1_{ср}-T^2_{ср}}{T^2_{ср}}\cdot 100\%$	
оператор	неотождест.	отожд.	оператор	неотождест.	отожд.	неотождест.	отожд.
I	2	3	4	5	6	7	8
COBJ	$\frac{6.25}{3.81}$	$\frac{14}{8.92}$	LSCO	$\frac{5}{2.52}$	$\frac{11}{6.48}$	+52%	+38%
CINT	$\frac{7.75}{4.26}$	$\frac{19}{10.73}$	LSC	$\frac{4}{2.63}$	$\frac{9}{6.12}$	+62%	+75%
RCOBJ	$\frac{5}{2.8}$	$\frac{18+6L}{10.76+3.05}$	RSCO	$\frac{5}{2.52}$	$\frac{11}{6.48}$	+11%	66+47L%
RCINT	$\frac{6.25}{3.35}$	$\frac{23+6L}{12.57+3.05}$	RSC	$\frac{4}{2.63}$	$\frac{9}{6.12}$	+27%	$\frac{105+}{50L}$ %
BTN	$\frac{4}{2.2}$	$\frac{17.5}{10.67}$	LB	$\frac{4}{1.95}$	$\frac{12}{6.73}$	+13%	+59%
BTO	$\frac{4}{2.2}$	$\frac{10}{5.62}$	LBY	$\frac{4}{1.95}$	$\frac{12}{6.73}$	+13%	-16%
RBTN	$\frac{4}{2.2}$	$\frac{17.5}{10.67}$	RB	$\frac{4}{1.95}$	$\frac{12}{6.73}$	+13%	+59%
RBTO	$\frac{4}{2.2}$	$\frac{14+6L}{7.46+3.05}$	RBY	$\frac{4}{1.95}$	$\frac{12}{6.73}$	+13%	$\frac{11+45}{\%}$
BTE	$\frac{5.33}{2.84}$	$\frac{12}{6.47}$	LBNIL	$\frac{6}{2.75}$	$\frac{15}{7.95}$	+3%	-19%
RBTE	$\frac{5.33}{2.84}$	$\frac{16+6L}{8.31+3.05}$	RBNIL	$\frac{6}{2.75}$	$\frac{15}{7.95}$	+3%	5+38%L
BEX	$\frac{4}{2.2}$	$\frac{16}{10.38}$	LBCE	$\frac{4}{1.95}$	$\frac{17}{10.32}$	+13%	1%

I	2	3	4	5	6	7	8
PBEX	4 / 2.2	20+6 / I2.22+3.05	PBCE	4 / I.95	I7 / I0.32	+I3%	I8+30 %
CHECK	2 / I.I	4 / I.82	NIL	2 / I.22	4 / I.94	-I0%	-6%
NH		I4.5 / 9.53	SB		I0 / 6.69		42%
NHB		I3.5 / 8.94	SB		I0 / 6.69		34%
NHR		25+6 / I4.64+3.05	SB		I0 / 6.69		II9+46Ł %
NHR*		I7 / I0.68	SB		I0 / 6.69		60%
NHRM		28+6Ł / I7.08+3.05	SB		I0 / 6.69		I55+46Ł %
NHRM*		I8.5 / II.9	SB		I0 / 6.69		78%
SYM	4 / 2.2	I0 / 5.62	LS	4 / I.95	9 / 4.76	+I3%	+I8%
RSYM	4 / 2.2	I4+6Ł / 7.46+3.05	RS	4 / I.95	9 / 4.76	+I3%	57+64Ł %
TERM	2 / I.I	9 / 5.9	LW	3 / I.46	II / 6.56	-25%	-I0%
RTERM	2 / I.I	I3+6 / 7.74+3.05	RW	3 / I.46	II / 6.56	-25%	I8+46Ł %
EXPR		8.5 / 5.9I	CE		7 / 4.67		27%
PL		II / II.07	PLE		9 / 8.96		24%
I) LEN4, LEN	9 / 5.I8	I5 / I0.34	LE	4 / I.82	I2 / 6.03	I85%	+7I%
LEN	2 / I.I	I0 / 6.26	LE	4 / I.82	I2 / 6.03	-40%	+4%
RPL		II / II.07	PRE		9 / 8,96		+24%
RLEN	2 / I.I	I0+6Ł / 5.77+3.03	RE	4 / I.82	I2 / 6.03	-40%	-4+5IŁ%
PBT		I4 / I0.74	SJUMP		6 / I3.7		-22%
2) HEOT	2 / I.22	9 / 7.I2	FAIL	2 / I.22	5 / 4.39	Ø	+62%

*) особые ситуации, не требующие сдвига B2
I) первое удлинение
2) обработка ситуации неотождествления

Видно, что предложенная реализация Рефала должна замедлить выполнение Рефал-программ, даже тех, в которых нет правостороннего отождествления. Это связано с тем, что для увеличения количества информации, хранимого в одном звене, и одновременно уменьшения самого звена, увеличения до 32 разрядов адреса связи, пришлось повысить сложность кодировки и при этом несколько потерять в скорости выполнения.

Мы предполагаем, что использование односвязной списковой памяти позволит сократить объемы требуемой для символьных выражений машинной памяти в I.5÷2 раза, а введение дополнительных типов данных - еще в 4 раза, обоснование этой оценки дается в [4]. Таким образом, ставится цель повышения эффективности использования памяти в 7-8 раз.

По скорости выполнения рефал-программ предложенная реализация медленнее, мы надеемся, что проигрыш от введения односвязной памяти будет не более, чем в 2 раза. Например, если пользуясь табл.2 подсчитать время выполнения отождествления для функции СПАРС (см.рис.2), обрабатывающей строку (A+B)*(C+D), то для традиционной реализации оно будет 2308 мксек, а для предлагаемой 2356 мксек. В данном случае сказывается то, что частота выполнения различных операторов - неодинакова, для этого примера: PBT - 26.6%; BTO - 27.6%; COBJ ⟶ HEOT - I4.4%; EXPR - I2.8%, NHB - 6.9%; CHECK ⟶ HEOT - 5.9%; SYM - 3.7%; COBJ - 2.I%. Представляет интерес также то, что из приведенных времен 730 мксек в каждой реализации тратится только на программную дешифрацию кода оператора языка сборки.

Работа по усовершенствованию алгоритмов предложенной реализации и их кодировке продолжается. Например, будет рассматриваться вариант отказа от использования операторов NH , NHB и т.д., за счет введения таблицы отождествляемых уровней скобочных структур. Будет рассмотрен вариант использования 24-разрядного адреса (однако вариант с 32-х разрядным адресом связи мы по-прежнему считаем основным) и ряд других усовершенствований. После проведения этой работы и введения расширений Рефала, описанных в [4], мы надеемся достичь большей скорости выполнения рефал-программ, чем в традиционной реализации.

6. Заключение

Видно, что описанные выше алгоритмы отождествления оказались не на много сложнее соответствующих алгоритмов традиционной схемы реализации и достаточно эффективно реализуются. В следующей работе, которая будет содержать описание операторов замены, мы покажем, что для них это тоже справедливо. В этой же работе будут даны результаты экспериментального сравнения с традиционной реализацией на различных Рефал-программах.

Однако решающим для данной реализации остается пока проверка на практике предположения о редком использовании ссылки на предыдущее звено или, по крайней мере, предположения о том, что такое ограничение доставит незначительные неудобства пользователям Рефала.

Литература

I Флоренцев С.Н. и др. Эффективный интепретатор языка Рефал. Препринт ИПМ им. М.В.Келдыша АН СССР, 1969, № 29, 103 стр.

2 Романенко С.А. Машинно-независимый компилятор с языка рекурсивных функций. Дисс. на соискание ученой степени канд. физ.-мат. наук. М., ИПМ им. М.В.Келдыша АН СССР, 1979, 211 стр.

3 Myamlin A.N., Smirnov V.K., Golovkov S.L. A specialized symbol processor. "Proc. of the IFIP TC 10 Working Conference on Fifth generation computer architectures. Manchester, U.K., 15-18 July, 1985". North-Holland, 1986, 301-318.

4 Эйсымонт Л.К. Выбор и оценка элементов базового языка символьного процессора. Дисс. на соискание ученой степени канд. физ.-мат. наук. М., ИПМ им. М.В.Келдыша АН СССР, 1983, 293 стр.

5 Moon D.A. Architecture of the Symbolics 3600. "SIGART Newsletter", 1985, Vol. 13, Issue 3, 76-83.

6 Okuno H.G. et al. TAO: a fast interpreter-centered system on lisp machine ELIS. "ACM Symp. LISP and Funct. Programm. Austin, Tex., Aug. 6-8, 1984. Conf. Rec.", New York, 1984, 140-149.

7 Warren D.H. et al. Prolog - the language and it's implementation compared with Lisp. "SIGART Newsletter", 1977, №64, 109-115.

8 McCarthy J. LISP - Notes on it's past and future. "Conf. Rec. of the 1980 LISP Conference. Stanford, Aug. 25-27, 1980", LISP Company, Stanford, 1980, V-VIII.

9 Griss M.L., Cowan R.M. Hashing - the key to rapid pattern matching. "Lect. Notes Comput. Sci.", 1979, №72, 266-278.

Ссылки на издание рекомендуется делать по следующей форме: и.о., фамилия, название, препринт Ин. прикл. матем. им. М.В. Келдыша АН СССР, год, №.

Распространение: препринты института продаются в магазинах Академкниги г. Москвы, а также распространяются через Библиотеку АН СССР в порядке обмена.

Адрес: СССР, 125047, Москва-47, Миусская пл. 4, Институт прикладной математики им. М.В. Келдыша АН СССР, ОНТИ.

The references should be typed by the following form: initials, name, title, preprint, Inst.Appl.Mathem., the USSR Academy of Sciences, year, N(number).

Distribution. The preprints of the Keldysh Institute of Applied Mathematics, the USSR Academy of Sciences are sold in the bookstores "Academkniga", Moscow and are distributed by the USSR Academy of Sciences Library as an exchange.

Adress: USSR, I25047, Moscow A-47, Miusskaya Sq.4, the Keldysh Institute of Applied Mathematics, Ac.of Sc., the USSR, Information Bureau.

Цена 12 коп.

Мансуров Николай Николаевич, Эйсымонт Леонид Константинович.
" Реализация расширенного языка Рефал на односвязных списках
с кольцевыми цепочками."
Редактор А.П. Фролов. Корректор В.Г. Перегудов.

Подписано к печати 08.01.87г. № Т- 04745. Заказ № 48.
Формат бумаги 60X90 1/16. Тираж 200 экз.
Объём 1,3 уч.-изд.л. Цена 12 коп.

055 (02)2

Отпечатано на ротапринтах в Институте прикладной математики АН СССР
Москва, Миусская пл. 4.