**КУРСОВАЯ РАБОТА**

# Разработка элементов учебной системы программирования

по дисциплине «Системы программирования»

Вариант №7

Выполнили студенты гр. 5084/13 «подпись» Благодиров А.В.

«подпись» Черкалова В.В.

«подпись» Афанасьева В.Ю.

Руководитель доцент «подпись» Расторгуев В.Я.

«\_\_\_»\_\_\_\_\_\_\_\_2012

# Разработка элементов учебной системы программирования

по дисциплине «Системы программирования».

# Компилятор с языка высокого уровня.

Вариант №7

Выполнили студенты гр. 5084/13 «подпись» Благодиров А.В.

«подпись» Черкалова В.В.

«подпись» Афанасьева В.Ю.

Руководитель доцент «подпись» Расторгуев В.Я.

«\_\_\_»\_\_\_\_\_\_\_\_2012

# Введение

Данная курсовая работа имеет своей целью получение практических навыков построения компилятора с языка высокого уровня (ЯВУ), являющегося одним из элементов системы программирования, образующих в совокупности следующий технологический конвейер:



При этом предполагается то, что данная система программирования работает на технологической ЭВМ (IBM PC) и является по существу кросс-системой для объектной ЭВМ (ЕС ЭВМ). В этой системе:

* в качестве языка высокого уровня (ЯВУ) выбран язык, образованный из подмножества языковых конструкций ПЛ1, а исходная программа готовится в виде текстового файла технологической ЭВМ с расширением \*.pli;
* язык АССЕМБЛЕРА сформирован из языковых конструкций АССЕМБЛЕРА ЕС ЭВМ, а ассемблеровский эквивалент исходной программы формируется в виде текстового файла технологической ЭВМ с расширением \*.ass;
* объектный эквивалент исходной программы готовится в формате объектных файлов операционной системы ОС ЕС ЭВМ и хранится в виде двоичного файла технологической ЭВМ с расширением \*.tex;
* загрузочный эквивалент исходной программы представляет собой машинный код ЕС ЭВМ, запоминаемый в области ОЗУ технологической ЭВМ, являющейся зоной загрузки для эмулятора объектной ЭВМ.

# Постановка задачи

Необходимо выполнить доработку элементов макета учебной системы программирования до уровня, позволяющего обрабатывать “новые” для макета конструкции языка высокого уровня, примененные в соответствующем варианте:



Где на входе имеется текст программы на ЯВУ PL/I:

EX07: PROC OPTIONS ( MAIN );

DCL A BIT ( 3 ) INIT ( '10'B );

DCL B BIT ( 3 ) INIT ( '101'B );

DCL C BIT ( 16 );

C = SUBSTR (( B !! A ), 2, 3 );

END EX07;

На выходе строится эквивалент программы на ассемблере архитектуры IBM 370:

EX07 START 0

BALR RBASE,0

USING \*,RBASE

L RRAB,@VA

L RSH,@LB

SLR RRAB,0(0,6)

O RRAB,@VB

ST RRAB,@VMID

A RSH,@LA

ST RSH,@LMID

L RRAB,@VMID

L RSH,SL

SLL RRAB,0(0,6)

N RRAB,MSK

ST RRAB,@VC

L RSH,NL

ST RSH,@LC

BCR 15,14

A DC F'3'

@LA DC F'2'

@VA DC BL4'10'

B DC F'3'

@LB DC F'3'

@VB DC BL4'101'

C DC F'16'

@LC DC F'0'

@VC DC BL4''

MID DC F'16'

@LMID DC F'0'

@VMID DC BL4''

SL DC F'1'

NL DC F'3'

MSK DC BL4'111'

RBASE EQU 15

RRAB EQU 5

RSH EQU 6

END EX07

# Анализ поставленной задачи

*Для реализации поставленной задачи необходимо определить структуру, описывающую битовые строки. То есть такую конструкцию на ассемблере, которая бы соответсвовала коду на PLI:*

*DCL A BIT ( 3 ) INIT ( '10'B );*

*Помимо этого не следует забывать про необходимость созхранить уже существующую функциональность.*

*Поэтому нами была выбрана следующая реализация:*

*A DC F'3'*

*@LA DC F'2'*

*@VA DC BL4'10'*

*Где метка А – заданнная в определении максимальная длина строки.*

*метка @LA – длина строки, записанной в значение*

*метка @VA – значение строки*

*Такая структура позволяет нам при минимальных модификациях (добавление типа BR4) реализовать данную задачу, не теряя существующей функциональности, поскольку обработка имеющихся типов остается неизменной.*

*Следует также отметить, что при построении указанной структуры используются спец. символы, которые не используются в языке PLI, что позволяет, сохраняя функциональность добавлять вспомогательные метки. Наличие такого спец. символа обеспечивает отсутсвие совпадений между именами переменных в коде и дополнительными метками при компиляции.*

*Также на ЯВУ нами были добавлены команды SUBSTR и !! (конкатенация).*

*Команда SUBSTR принимает на вход три параметра – строку, номер символа, с которого наинается отсчет, и колличество символов (длину подстроки).*

*Будем воспринимать это следующим образом:*

*SUBSTR(‘0111000’,2,3) = ‘111’*

*Команда конкатенации (!!) принимает два аргумента и “склеивает”их. То есть:*

*‘111’ !! ‘10’ = ‘11110’*

*При реализации ланных операций на ассемблере мы импользовали следующие алгоритмы:*

1. *Подстрока:*

* *Считываем номер элемента N, с которого следует взять подстроку*
* *Сдвигаем строку влево на N-1*
* *При компиляции с ЯВУ создаем маску вида “111000000....”, где колличество 1 – длина подстроки*
* *При помощи операции И “отрезаем” конец старой строки*
* *Записываем результат в поле значения*
* *Записываем новую длину в поле длины*

1. *Конкатенация:*

* *Сдвигаем второй элемент вправо на длину первого*
* *При помощи операции ИЛИ получаем результат*
* *Записываем результат в поле значения*
* *Складываем длину первого и второго аргументов*
* *Записываем новую длину в поле длины*

# Входные ограничения

*При постановке задачи были введены следующие ограничения:*

*1) Максимальная длина битовой строки, обрабатываемая с помощью данного алгоритма – 16.*

*2) Реализованный алгоритм не проверяет длину результата конкатенации двух строк, то есть он не обрабатывает каким либо специфическим образом ситуацию, когда строки А и B заданы таким образом, что их конкатенация превышает максимальную длину битовой строки, определенную в пункте 1.*

*3) В реализованном алгоритме не проверяется попытка взять подстроку выходящую за пределы имебщейся строки (например подстроку длиной 3 из строки длины 1)..*

*4) При задании переменной типа битовая строка без инициализации, компиллятор автоматически записывает в строку 0.*

# Модификация базы данных исходного макета

*Нами были добавлены дополнительные определения операций в синтаксис:*

struct

{

int POSL;

int PRED;

char DER[4];

int ALT;

} SINT [ NSINT ] =

/\* \_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_ \_\_\_\_\_\_\_ \_\_\_\_\_\_

| NN : посл : пред : дер : альт |

|\_\_\_\_\_\_\_\_\_\_:\_\_\_\_\_\_\_\_\_:\_\_\_\_\_\_\_:\_\_\_\_\_\_\_:\_\_\_\_\_\_| \*/

{

{/\*. 0 .\*/ -1 , -1 , "\*\*\*" , -1 },

/\* вход с символа - 0 \*/

{/\*. 1 .\*/ 2 , 0 , "0 " , 0 },

{/\*. 2 .\*/ 3 , 1 , "CIF" , 0 },

{/\*. 3 .\*/ 0 , 2 , "\* " , 0 },

/\* вход с символа - 1 \*/

{/\*. 4 .\*/ 5 , 0 , "1 " , 0 },

{/\*. 5 .\*/ 6 , 4 , "CIF" , 7 },

{/\*. 6 .\*/ 0 , 5 , "\* " , 0 },

{/\*. 7 .\*/ 8 , 4 , "MAN" , 0 },

{/\*. 8 .\*/ 0 , 7 , "\* " , 0 },

/\* вход с символа - 2 \*/

{/\*. 9 .\*/ 10 , 0 , "2 " , 0 },

{/\*. 10 .\*/ 11 , 9 , "CIF" , 0 },

{/\*. 11 .\*/ 0 , 10 , "\* " , 0 },

/\* вход с символа - 3 \*/

{/\*. 12 .\*/ 13 , 0 , "3 " , 0 },

{/\*. 13 .\*/ 14 , 12 , "CIF" , 0 },

{/\*. 14 .\*/ 0 , 13 , "\* " , 0 },

/\* вход с символа - 4 \*/

{/\*. 15 .\*/ 16 , 0 , "4 " , 0 },

{/\*. 16 .\*/ 17 , 15 , "CIF" , 0 },

{/\*. 17 .\*/ 0 , 16 , "\* " , 0 },

/\* вход с символа - 5 \*/

{/\*. 18 .\*/ 19 , 0 , "5 " , 0 },

{/\*. 19 .\*/ 20 , 18 , "CIF" , 0 },

{/\*. 20 .\*/ 0 , 19 , "\* " , 0 },

/\* вход с символа - 6 \*/

{/\*. 21 .\*/ 22 , 0 , "6 " , 0 },

{/\*. 22 .\*/ 23 , 21 , "CIF" , 0 },

{/\*. 23 .\*/ 0 , 22 , "\* " , 0 },

/\* вход с символа - 7 \*/

{/\*. 24 .\*/ 25 , 0 , "7 " , 0 },

{/\*. 25 .\*/ 26 , 24 , "CIF" , 0 },

{/\*. 26 .\*/ 0 , 25 , "\* " , 0 },

/\* вход с символа - 8 \*/

{/\*. 27 .\*/ 28 , 0 , "8 " , 0 },

{/\*. 28 .\*/ 29 , 27 , "CIF" , 0 },

{/\*. 29 .\*/ 0 , 28 , "\* " , 0 },

/\* вход с символа - 9 \*/

{/\*. 30 .\*/ 31 , 0 , "9 " , 0 },

{/\*. 31 .\*/ 32 , 30 , "CIF" , 0 },

{/\*. 32 .\*/ 0 , 31 , "\* " , 0 },

/\* вход с символа - A \*/

{/\*. 33 .\*/ 34 , 0 , "A " , 0 },

{/\*. 34 .\*/ 35 , 33 , "BUK" , 0 },

{/\*. 35 .\*/ 0 , 34 , "\* " , 0 },

/\* вход с символа - B \*/

{/\*. 36 .\*/ 37 , 0 , "B " , 0 },

{/\*. 37 .\*/ 38 , 36 , "BUK" , 0 },

{/\*. 38 .\*/ 0 , 37 , "\* " , 0 },

/\* вход с символа - C \*/

{/\*. 39 .\*/ 40 , 0 , "C " , 0 },

{/\*. 40 .\*/ 41 , 39 , "BUK" , 0 },

{/\*. 41 .\*/ 0 , 40 , "\* " , 0 },

/\* вход с символа - D \*/

{/\*. 42 .\*/ 43 , 0 , "D " , 0 },

{/\*. 43 .\*/ 44 , 42 , "BUK" , 45 },

{/\*. 44 .\*/ 0 , 43 , "\* " , 0 },

{/\*. 45 .\*/ 46 , 42 , "C " , 0 },

{/\*. 46 .\*/ 47 , 45 , "L " , 0 },

{/\*. 47 .\*/ 48 , 46 , " " , 0 },

{/\*. 48 .\*/ 49 , 47 , "IDE" , 0 },

{/\*. 49 .\*/ 50 , 48 , " " , 0 },

{/\*. 50 .\*/ 51 , 49 , "B " , 187 },

{/\*. 51 .\*/ 52 , 50 , "I " , 0 },

**{/\*. 52 .\*/ 53 , 51 , "N " , 201 },**

{/\*. 53 .\*/ 54 , 52 , " " , 0 },

{/\*. 54 .\*/ 55 , 53 , "F " , 0 },

{/\*. 55 .\*/ 56 , 54 , "I " , 0 },

{/\*. 56 .\*/ 57 , 55 , "X " , 0 },

{/\*. 57 .\*/ 58 , 56 , "E " , 0 },

{/\*. 58 .\*/ 59 , 57 , "D " , 0 },

{/\*. 59 .\*/ 60 , 58 , "( " , 0 },

{/\*. 60 .\*/ 61 , 59 , "RZR" , 0 },

{/\*. 61 .\*/ 62 , 60 , ") " , 0 },

{/\*. 62 .\*/ 63 , 61 , "; " , 65 },

{/\*. 63 .\*/ 64 , 62 , "ODC" , 0 },

{/\*. 64 .\*/ 65 , 63 , "\* " , 0 },

{/\*. 65 .\*/ 66 , 61 , "I " , 0 },

{/\*. 66 .\*/ 67 , 65 , "N " , 0 },

{/\*. 67 .\*/ 68 , 66 , "I " , 0 },

{/\*. 68 .\*/ 69 , 67 , "T " , 0 },

{/\*. 69 .\*/ 70 , 68 , "( " , 0 },

{/\*. 70 .\*/ 71 , 69 , "LIT" , 0 },

{/\*. 71 .\*/ 72 , 70 , ") " , 0 },

{/\*. 72 .\*/ 73 , 71 , "; " , 0 },

{/\*. 73 .\*/ 186 , 72 , "ODC" , 0 },

/\* вход с символа - E \*/

{/\*. 74 .\*/ 75 , 0 , "E " , 0 },

{/\*. 75 .\*/ 76 , 74 , "N " , 82 },

{/\*. 76 .\*/ 77 , 75 , "D " , 0 },

{/\*. 77 .\*/ 78 , 76 , " " , 0 },

{/\*. 78 .\*/ 79 , 77 , "IPR" , 0 },

{/\*. 79 .\*/ 80 , 78 , "; " , 0 },

{/\*. 80 .\*/ 81 , 79 , "OEN" , 0 },

{/\*. 81 .\*/ 0 , 80 , "\* " , 0 },

{/\*. 82 .\*/ 83 , 74 , "BUK" , 0 },

{/\*. 83 .\*/ 0 , 82 , "\* " , 0 },

/\* вход с символа - M \*/

{/\*. 84 .\*/ 85 , 0 , "M " , 0 },

{/\*. 85 .\*/ 86 , 84 , "BUK" , 0 },

{/\*. 86 .\*/ 0 , 85 , "\* " , 0 },

/\* вход с символа - P \*/

{/\*. 87 .\*/ 88 , 0 , "P " , 0 },

{/\*. 88 .\*/ 89 , 87 , "BUK" , 0 },

{/\*. 89 .\*/ 0 , 88 , "\* " , 0 },

/\* вход с символа - X \*/

{/\*. 90 .\*/ 91 , 0 , "X " , 0 },

{/\*. 91 .\*/ 92 , 90 , "BUK" , 0 },

{/\*. 92 .\*/ 0 , 91 , "\* " , 0 },

/\* вход с символа - BUK \*/

{/\*. 93 .\*/ 94 , 0 , "BUK" , 0 },

{/\*. 94 .\*/ 95 , 93 , "IDE" , 0 },

{/\* 95 .\*/ 0 , 94 , "\* " , 0 },

/\* вход с символа - IDE \*/

{/\*. 96 .\*/ 97 , 0 , "IDE" , 0 },

{/\*. 97 .\*/ 98 , 96 , "BUK" , 100 },

{/\*. 98 .\*/ 99 , 97 , "IDE" , 0 },

{/\*. 99 .\*/ 0 , 98 , "\* " , 0 },

{/\*. 100 .\*/ 101 , 96 , "CIF" , 103 },

{/\*. 101 .\*/ 102 , 100 , "IDE" , 0 },

{/\*. 102 .\*/ 0 , 101 , "\* " , 0 },

{/\*. 103 .\*/ 104 , 96 , "IPE" , 105 },

{/\*. 104 .\*/ 0 , 103 , "\* " , 0 },

{/\*. 105 .\*/ 106 , 96 , "IPR" , 0 },

{/\*. 106 .\*/ 0 , 105 , "\* " , 0 },

/\* вход с символа - + \*/

{/\*. 107 .\*/ 108 , 0 , "+ " , 0 },

{/\*. 108 .\*/ 109 , 107 , "ZNK" , 0 },

{/\*. 109 .\*/ 0 , 108 , "\* " , 0 },

/\* вход с символа - - \*/

{/\*. 110 .\*/ 111 , 0 , "- " , 0 },

{/\*. 111 .\*/ 112 , 110 , "ZNK" , 0 },

{/\*. 112 .\*/ 0 , 111 , "\* " , 0 },

/\* вход с символа - IPR \*/

{/\*. 113 .\*/ 114 , 0 , "IPR" , 0 },

{/\*. 114 .\*/ 115 , 113 , ": " , 0 },

{/\*. 115 .\*/ 116 , 114 , "P " , 0 },

{/\*. 116 .\*/ 117 , 115 , "R " , 0 },

{/\*. 117 .\*/ 118 , 116 , "O " , 0 },

{/\*. 118 .\*/ 119 , 117 , "C " , 0 },

{/\*. 119 .\*/ 120 , 118 , " " , 0 },

{/\*. 120 .\*/ 121 , 119 , "O " , 0 },

{/\*. 121 .\*/ 122 , 120 , "P " , 0 },

{/\*. 122 .\*/ 123 , 121 , "T " , 0 },

{/\*. 123 .\*/ 124 , 122 , "I " , 0 },

{/\*. 124 .\*/ 125 , 123 , "O " , 0 },

{/\*. 125 .\*/ 126 , 124 , "N " , 0 },

{/\*. 126 .\*/ 127 , 125 , "S " , 0 },

{/\*. 127 .\*/ 128 , 126 , "( " , 0 },

{/\*. 128 .\*/ 129 , 127 , "M " , 0 },

{/\*. 129 .\*/ 130 , 128 , "A " , 0 },

{/\*. 130 .\*/ 131 , 129 , "I " , 0 },

{/\*. 131 .\*/ 132 , 130 , "N " , 0 },

{/\*. 132 .\*/ 133 , 131 , ") " , 0 },

{/\*. 133 .\*/ 134 , 132 , "; " , 0 },

{/\*. 134 .\*/ 135 , 133 , "OPR" , 0 },

{/\*. 135 .\*/ 0 , 134 , "\* " , 0 },

/\* вход с символа - CIF \*/

{/\*. 136 .\*/ 137 , 0 , "CIF" , 0 },

{/\*. 137 .\*/ 138 , 136 , "RZR" , 0 },

{/\*. 138 .\*/ 0 , 0 , "\* " , 0 },

/\* вход с символа - RZR \*/

{/\*. 139 .\*/ 140 , 0 , "RZR" , 0 },

{/\*. 140 .\*/ 141 , 139 , "CIF" , 0 },

{/\*. 141 .\*/ 142 , 140 , "RZR" , 0 },

{/\*. 142 .\*/ 0 , 141 , "\* " , 0 },

/\* вход с символа - MAN \*/

{/\*. 143 .\*/ 144 , 0 , "MAN" , 0 },

{/\*. 144 .\*/ 145 , 143 , "B " , 147 },

{/\*. 145 .\*/ 146 , 144 , "LIT" , 0 },

{/\*. 146 .\*/ 0 , 145 , "\* " , 0 },

{/\*. 147 .\*/ 148 , 143 , "0 " , 150 },

{/\*. 148 .\*/ 149 , 147 , "MAN" , 0 },

{/\*. 149 .\*/ 0 , 148 , "\* " , 0 },

{/\*. 150 .\*/ 151 , 143 , "1 " , 0 },

{/\*. 151 .\*/ 152 , 150 , "MAN" , 0 },

{/\*. 152 .\*/ 0 , 151 , "\* " , 0 },

/\* вход с символа - IPE \*/

{/\*. 153 .\*/ 154 , 0 , "IPE" , 0 },

{/\*. 154 .\*/ 155 , 153 , "= " , 159 },

**{/\*. 155 .\*/ 156 , 154 , "AVI" , 220 },**

{/\*. 156 .\*/ 157 , 155 , "; " , 0 },

{/\*. 157 .\*/ 158 , 156 , "OPA" , 0 },

{/\*. 158 .\*/ 0 , 157 , "\* " , 0 },

{/\*. 159 .\*/ 160 , 153 , "AVI" , 0 },

{/\*. 160 .\*/ 0 , 159 , "\* " , 0 },

/\* вход с символа - LIT \*/

{/\*. 161 .\*/ 162 , 0 , "LIT" , 0 },

{/\*. 162 .\*/ 163 , 161 , "AVI" , 0 },

{/\*. 163 .\*/ 0 , 162 , "\* " , 0 },

/\*. вход с символа - AVI \*/

{/\*. 164 .\*/ 165 , 0 , "AVI" , 0 },

{/\*. 165 .\*/ 166 , 164 , "ZNK" , 0 },

{/\*. 166 .\*/ 167 , 165 , "LIT" , 168 },

{/\*. 167 .\*/ 197 , 166 , "AVI" , 0 },

{/\*. 168 .\*/ 169 , 165 , "IPE" , 0 },

{/\*. 169 .\*/ 170 , 168 , "AVI" , 0 },

{/\*. 170 .\*/ 0 , 169 , "\* " , 0 },

/\* вход с символа - OPR \*/

{/\*. 171 .\*/ 172 , 0 , "OPR" , 0 },

{/\*. 172 .\*/ 173 , 171 , "TEL" , 0 },

{/\*. 173 .\*/ 174 , 172 , "OEN" , 0 },

{/\*. 174 .\*/ 175 , 173 , "PRO" , 0 },

{/\*. 175 .\*/ 0 , 174 , "\* " , 0 },

/\*. вход с символа - ODC \*/

{/\*. 176 .\*/ 177 , 0 , "ODC" , 0 },

{/\*. 177 .\*/ 178 , 176 , "TEL" , 0 },

{/\*. 178 .\*/ 0 , 177 , "\* " , 0 },

/\*. вход с символа - TEL \*/

{/\*. 179 .\*/ 180 , 0 , "TEL" , 0 },

{/\*. 180 .\*/ 181 , 179 , "ODC" , 183 },

{/\*. 181 .\*/ 182 , 180 , "TEL" , 0 },

{/\*. 182 .\*/ 0 , 181 , "\* " , 0 },

{/\*. 183 .\*/ 184 , 179 , "OPA" , 0 },

{/\*. 184 .\*/ 185 , 183 , "TEL" , 0 },

{/\*. 185 .\*/ 0 , 184 , "\* " , 0 },

{/\*. 186 .\*/ 0 , 73 , "\* " , 0 },

{/\*. 187 .\*/ 188 , 49 , "C " , 0 },

{/\*. 188 .\*/ 189 , 187 , "H " , 0 },

{/\*. 189 .\*/ 190 , 188 , "A " , 0 },

{/\*. 190 .\*/ 191 , 189 , "R " , 0 },

{/\*. 191 .\*/ 192 , 190 , "( " , 0 },

{/\*. 192 .\*/ 193 , 191 , "RZR" , 0 },

{/\*. 193 .\*/ 194 , 192 , ") " , 0 },

{/\*. 194 .\*/ 195 , 193 , "; " , 0 },

{/\*. 195 .\*/ 196 , 194 , "ODC" , 0 },

{/\*. 196 .\*/ 0 , 195 , "\* " , 0 },

{/\*. 197 .\*/ 0 , 166 , "\* " , 0 },

{/\*. 198 .\*/ 199 , 0 , "\* " , 0 },

{/\*. 199 .\*/ 200 , 198 , "ZNK" , 0 },

{/\*. 200 .\*/ 0 , 199 , "\* " , 0 },

**{/\*. 201 .\*/ 202 , 51 , "T " , 0 },**

**{/\*. 202 .\*/ 203 , 201 , "( " , 0 },**

**{/\*. 203 .\*/ 204 , 202 , "RZR" , 0 },**

**{/\*. 204 .\*/ 205 , 203 , ") " , 0 },**

**{/\*. 205 .\*/ 206 , 204 , "; " , 208 },**

**{/\*. 206 .\*/ 207 , 205 , "ODC" , 0 },**

**{/\*. 207 .\*/ 208, 206 , "\* " , 0 },**

**{/\*. 208 .\*/ 209 , 207 , "I " , 0 },**

**{/\*. 209 .\*/ 210 , 208 , "N " , 0 },**

**{/\*. 210 .\*/ 211 , 209 , "I " , 0 },**

**{/\*. 211 .\*/ 212 , 210 , "T " , 0 },**

**{/\*. 212 .\*/ 213 , 211 , "( " , 0 },**

**{/\*. 213 .\*/ 214 , 212 , "LIT" , 0 },**

**{/\*. 214 .\*/ 215 , 213 , ") " , 0 },**

**{/\*. 215 .\*/ 216 , 214 , "; " , 0 },**

**{/\*. 216 .\*/ 186 , 215 , "ODC" , 0 },**

**/\* вход с символа - ! \*/**

**{/\*. 217 .\*/ 218 , 0 , "! " , 0 },**

**{/\*. 218 .\*/ 219 , 217 , "ZNK" , 0 },**

**{/\*. 219 .\*/ 0 , 218 , "\* " , 0 },**

**{/\*. 220 .\*/ 221 , 154 , "S " , 0 },**

**{/\*. 221 .\*/ 222 , 220 , "U " , 0 },**

**{/\*. 222 .\*/ 223 , 221 , "B " , 0 },**

**{/\*. 223 .\*/ 224 , 222 , "S " , 0 },**

**{/\*. 224 .\*/ 225 , 223 , "T " , 0 },**

**{/\*. 225 .\*/ 226 , 224 , "R " , 0 },**

**{/\*. 226 .\*/ 227 , 225 , "( " , 0 },**

**{/\*. 227 .\*/ 228 , 226 , "( " , 0 },**

**{/\*. 228 .\*/ 229 , 227 , "AVI" , 0 },**

**{/\*. 229 .\*/ 230 , 228 , ") " , 0 },**

**{/\*. 230 .\*/ 231 , 229 , ", " , 0 },**

**{/\*. 231 .\*/ 232 , 230 , "RZR" , 0 },**

**{/\*. 232 .\*/ 233 , 231 , ", " , 0 },**

**{/\*. 233 .\*/ 234 , 232 , "RZR" , 0 },**

**{/\*. 234 .\*/ 235 , 233 , ") " , 0 },**

**{/\*. 235 .\*/ 236 , 234 , "; " , 0 },**

**{/\*. 236 .\*/ 237 , 235 , "OPA" , 0 },**

**{/\*. 237 .\*/ 0 , 236 , "\* " , 0 }**

};

struct

{

char SYM [4];

int VX;

char TYP;

} VXOD [ NVXOD ] =

/\* \_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_ \_\_\_\_\_\_

| NN | символ | вход| тип |

|\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_|\_\_\_\_\_\_| \*/

{

{/\*. 1 .\*/ "AVI" , 164 , 'N' },

{/\*. 2 .\*/ "BUK" , 93 , 'N' },

{/\*. 3 .\*/ "CIF" , 136 , 'N' },

{/\*. 4 .\*/ "IDE" , 96 , 'N' },

{/\*. 5 .\*/ "IPE" , 153 , 'N' },

{/\*. 6 .\*/ "IPR" , 113 , 'N' },

{/\*. 7 .\*/ "LIT" , 161 , 'N' },

{/\*. 8 .\*/ "MAN" , 143 , 'N' },

{/\*. 9 .\*/ "ODC" , 176 , 'N' },

{/\*. 10 .\*/ "OEN" , 0 , 'N' },

{/\*. 11 .\*/ "OPA" , 0 , 'N' },

{/\*. 12 .\*/ "OPR" , 171 , 'N' },

{/\*. 13 .\*/ "PRO" , 0 , 'N' },

{/\*. 14 .\*/ "RZR" , 139 , 'N' },

{/\*. 15 .\*/ "TEL" , 179 , 'N' },

{/\*. 16 .\*/ "ZNK" , 0 , 'N' },

{/\*. 17 .\*/ "A " , 33 , 'T' },

{/\*. 18 .\*/ "B " , 36 , 'T' },

{/\*. 19 .\*/ "C " , 39 , 'T' },

{/\*. 20 .\*/ "D " , 42 , 'T' },

{/\*. 21 .\*/ "E " , 74 , 'T' },

{/\*. 22 .\*/ "M " , 84 , 'T' },

{/\*. 23 .\*/ "P " , 87 , 'T' },

{/\*. 24 .\*/ "X " , 90 , 'T' },

{/\*. 25 .\*/ "0 " , 1 , 'T' },

{/\*. 26 .\*/ "1 " , 4 , 'T' },

{/\*. 27 .\*/ "2 " , 9 , 'T' },

{/\*. 28 .\*/ "3 " , 12 , 'T' },

{/\*. 29 .\*/ "4 " , 15 , 'T' },

{/\*. 30 .\*/ "5 " , 18 , 'T' },

{/\*. 31 .\*/ "6 " , 21 , 'T' },

{/\*. 32 .\*/ "7 " , 24 , 'T' },

{/\*. 33 .\*/ "8 " , 27 , 'T' },

{/\*. 34 .\*/ "9 " , 30 , 'T' },

{/\*. 35 .\*/ "+ " , 107 , 'T' },

{/\*. 36 .\*/ "- " , 110 , 'T' },

{/\*. 37 .\*/ ": " , 0 , 'T' },

{/\*. 38 .\*/ "I " , 0 , 'T' },

{/\*. 39 .\*/ "R " , 0 , 'T' },

{/\*. 40 .\*/ "N " , 0 , 'T' },

{/\*. 41 .\*/ "O " , 0 , 'T' },

{/\*. 42 .\*/ "T " , 0 , 'T' },

{/\*. 43 .\*/ "S " , 0 , 'T' },

{/\*. 44 .\*/ "( " , 0 , 'T' },

{/\*. 45 .\*/ ") " , 0 , 'T' },

{/\*. 46 .\*/ " " , 0 , 'T' },

{/\*. 47 .\*/ "; " , 0 , 'T' },

{/\*. 48 .\*/ "L " , 0 , 'T' },

{/\*. 49 .\*/ "F " , 0 , 'T' },

{/\*. 50 .\*/ "= " , 0 , 'T' },

{/\*. 51 .\*/ "H " , 0 , 'T' },

{/\*. 52 .\*/ "\* " , 198 , 'T' },

**{/\*. 53 .\*/ "! " , 217 , 'T' },**

**{/\*. 54 .\*/ "U " , 0 , 'T' },**

**{/\*. 55 .\*/ ", " , 0 , 'T' }**

};

char TPR [ NVXOD ] [ NNETRM ] =

{

/\*

\_\_\_\_\_\_\_\_\_\_ \_\_\_ \_\_\_ \_\_\_ \_\_\_ \_\_\_ \_\_\_ \_\_\_ \_\_\_ \_\_\_ \_\_\_ \_\_\_ \_\_\_ \_\_\_ \_\_\_ \_\_\_

| AVI:BUK:CIF:IDE:IPE:IPR:LIT:MAN:ODC:OEN:OPA:OPR:PRO:RZR:TEL:ZNK|

|\_\_\_\_\_\_\_\_\_\_:\_\_\_:\_\_\_:\_\_\_:\_\_\_:\_\_\_:\_\_\_:\_\_\_:\_\_\_:\_\_\_:\_\_\_:\_\_\_:\_\_\_:\_\_\_:\_\_\_:\_\_\_| \*/

{/\*AVI\*/ 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },

{/\*BUK\*/ 0 , 0 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },

{/\*CIF\*/ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1 , 0 , 0 },

{/\*IDE\*/ 0 , 0 , 0 , 1 , 1 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },

{/\*IPE\*/ 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 0 , 0 },

{/\*IPR\*/ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 0 },

{/\*LIT\*/ 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },

{/\*MAN\*/ 0 , 0 , 0 , 0 , 0 , 0 , 1 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },

{/\*ODC\*/ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1 , 0 },

{/\*OEN\*/ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },

{/\*OPA\*/ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },

{/\*OPR\*/ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1 , 0 , 0 , 0 },

{/\*PRO\*/ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },

{/\*RZR\*/ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1 , 0 , 0 },

{/\*TEL\*/ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1 , 0 },

{/\*ZNK\*/ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },

/\*

\_\_\_\_\_\_\_\_\_\_ \_\_\_ \_\_\_ \_\_\_ \_\_\_ \_\_\_ \_\_\_ \_\_\_ \_\_\_ \_\_\_ \_\_\_ \_\_\_ \_\_\_ \_\_\_ \_\_\_ \_\_\_

| AVI:BUK:CIF:IDE:IPE:IPR:LIT:MAN:ODC:OEN:OPA:OPR:PRO:RZR:TEL:ZNK|

|\_\_\_\_\_\_\_\_\_\_:\_\_\_:\_\_\_:\_\_\_:\_\_\_:\_\_\_:\_\_\_:\_\_\_:\_\_\_:\_\_\_:\_\_\_:\_\_\_:\_\_\_:\_\_\_:\_\_\_:\_\_\_| \*/

{/\* A\*/ 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },

{/\* B\*/ 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },

{/\* C\*/ 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },

{/\* D\*/ 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },

{/\* E\*/ 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 },

{/\* M\*/ 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },

{/\* P\*/ 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },

{/\* X\*/ 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },

{/\* 0\*/ 0 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },

{/\* 1\*/ 0 , 0 , 1 , 0 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },

{/\* 2\*/ 0 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },

{/\* 3\*/ 0 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },

{/\* 4\*/ 0 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },

{/\* 5\*/ 0 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1 , 0 , 0 },

{/\* 6\*/ 0 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1 , 0 },

{/\* 7\*/ 0 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },

/\*

\_\_\_\_\_\_\_\_\_\_ \_\_\_ \_\_\_ \_\_\_ \_\_\_ \_\_\_ \_\_\_ \_\_\_ \_\_\_ \_\_\_ \_\_\_ \_\_\_ \_\_\_ \_\_\_ \_\_\_ \_\_\_

| AVI:BUK:CIF:IDE;IPE:IPR:LIT:MAN:ODC:OEN:OPA:OPR:PRO:RZR:TEL:ZNK|

|\_\_\_\_\_\_\_\_\_\_:\_\_\_:\_\_\_:\_\_\_:\_\_\_:\_\_\_:\_\_\_:\_\_\_:\_\_\_:\_\_\_:\_\_\_:\_\_\_:\_\_\_:\_\_\_:\_\_\_:\_\_\_| \*/

{/\* 8\*/ 0 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },

{/\* 9\*/ 0 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },

{/\* +\*/ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1 },

{/\* -\*/ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1 },

{/\* :\*/ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },

{/\* I\*/ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },

{/\* R\*/ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },

{/\* N\*/ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },

{/\* O\*/ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },

{/\* T\*/ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },

{/\* S\*/ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },

{/\* (\*/ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },

{/\* )\*/ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },

{/\* ;\*/ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },

{/\* \*/ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },

{/\* L\*/ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },

/\*

\_\_\_\_\_\_\_\_\_\_ \_\_\_ \_\_\_ \_\_\_ \_\_\_ \_\_\_ \_\_\_ \_\_\_ \_\_\_ \_\_\_ \_\_\_ \_\_\_ \_\_\_ \_\_\_ \_\_\_ \_\_\_

| AVI:BUK:CIF:IDE:IPE:IPR:LIT:MAN:ODC:OEN;OPA:OPR:PR0:RZR:TEL:ZNK|

|\_\_\_\_\_\_\_\_\_\_:\_\_\_:\_\_\_:\_\_\_:\_\_\_:\_\_\_:\_\_\_:\_\_\_:\_\_\_:\_\_\_:\_\_\_:\_\_\_:\_\_\_:\_\_\_:\_\_\_:\_\_\_| \*/

{/\* F\*/ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },

{/\* =\*/ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },

{/\* H\*/ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },

{/\* \*\*/ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1 },

**{/\* !\*/ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1 },**

**{/\* U\*/ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 },**

**{/\* ,\*/ 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 }**

/\*|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_| \*/

};

# Модификация алгоритма исходного макета

*Непосредственно реализованный алгоритм для операций конкатенация и подстрока:*

void compress\_ISXTXT() /\* Программа уплотнения \*/

/\* исходного текста путем \*/

/\* удаления "лишних" \*/

/\* пробелов, выполняющая \*/

/\* роль примитивного лек- \*/

/\* сического анализатора \*/

{

I3 = 0;

for ( I1 = 0 ; I1 < NISXTXT ; I1++ )

{

for ( I2 = 0 ; I2 < 80 ; I2++ )

if ( ISXTXT [ I1 ][ I2 ] != '\x0' )

{

if ( ISXTXT [ I1 ][ I2 ] == ' ' &&

( PREDSYM == ' ' || PREDSYM == ';' ||

PREDSYM == ')' || PREDSYM == ':' ||

PREDSYM == '(' **|| PREDSYM == ','**

)

)

{

PREDSYM = ISXTXT [ I1 ][ I2 ];

goto L2;

}

**if ( ISXTXT [ I1 ][ I2 ] == '!' &&**

**( PREDSYM == '!') )**

**{**

**PREDSYM = ISXTXT [ I1 ][ I2 ];**

**goto L2;**

**}**

if

(

( ISXTXT [ I1 ][ I2 ] == '+' ||

ISXTXT [ I1 ][ I2 ] == '-' ||

ISXTXT [ I1 ][ I2 ] == '=' ||

ISXTXT [ I1 ][ I2 ] == '(' ||

ISXTXT [ I1 ][ I2 ] == ')' ||

ISXTXT [ I1 ][ I2 ] == '\*' **||**

**ISXTXT [ I1 ][ I2 ] == '!'**

)

&&

PREDSYM == ' '

)

{

I3-- ;

goto L1;

}

if ( ISXTXT [ I1 ][ I2 ] == ' ' &&

( PREDSYM == '+' || PREDSYM == '-' ||

PREDSYM == '=' || PREDSYM == '\*' **|| PREDSYM == '!'**

)

)

{

goto L2;

}

L1:

PREDSYM = ISXTXT [ I1 ][ I2 ];

STROKA [ I3 ] = PREDSYM;

I3++ ;

L2: continue;

}

else

break;

}

STROKA [I3] = '\x0';

}

int ODC1 ()

{

int i;

FORM (); /\* форматирование ПЛ1-опе-\*/

/\* ратора DCL \*/

for ( i = 0; i < ISYM; i++ ) /\* если фиксируем повтор- \*/

{ /\* повторное объявление \*/

if ( !strcmp ( SYM [i].NAME, FORMT [1] ) && /\* второго терма оператора\*/

strlen ( SYM [i].NAME ) == /\* DCL, то \*/

strlen ( FORMT [1] )

)

return 6; /\* завершение программы \*/

/\* по ошибке \*/

}

strcpy ( SYM [ISYM].NAME, FORMT [1] ); /\* при отсутствии повтор- \*/

/\* ного объявления иденти-\*/

/\* фикатора запоминаем его\*/

/\* вместе с разрядностью в\*/

/\* табл.SYM \*/

if ( !strcmp ( FORMT [2], "BIN" ) && /\* если идентификатор оп- \*/

!strcmp ( FORMT [3], "FIXED" ) )/\* ределен как bin fixed, \*/

{

strcpy ( SYM [ISYM].RAZR, FORMT [4] );

SYM [ISYM].TYPE = 'B'; /\* то устанавливаем тип \*/

/\* идентификатора = 'B' и \*/

/\* если идентификатор \*/

/\* имеет начальную иници- \*/

if ( !strcmp ( FORMT [5], "INIT" ) ) /\* ализацию, то запомина- \*/

strcpy ( SYM [ISYM++].INIT, FORMT [6] ); /\* ем в табл. SYM это на- \*/

/\* чальное значение, а \*/

else /\* иначе \*/

strcpy ( SYM [ISYM++].INIT, "0B" ); /\* инициализируем иденти- \*/

/\* фикатор нулем \*/

return 0;

}

**else if ( !strcmp ( FORMT [2], "BIT" ) )/\* ределен как bit, \*/**

**{**

**strcpy ( SYM [ISYM].RAZR, FORMT [3] );**

**SYM [ISYM].TYPE = 'T'; /\* то устанавливаем тип \*/**

**/\* идентификатора = 'T' и \*/**

**/\* если идентификатор \*/**

**/\* имеет начальную иници- \*/**

**if ( !strcmp ( FORMT [4], "INIT" ) ) /\* ализацию, то запомина- \*/**

**strcpy ( SYM [ISYM++].INIT, FORMT [5] ); /\* ем в табл. SYM это на- \*/**

**/\* чальное значение, а \*/**

**else /\* иначе \*/**

**strcpy ( SYM [ISYM++].INIT, "B" ); /\* инициализируем иденти- \*/**

**/\* фикатор нулем \*/**

**return 0;**

**}**

else /\* иначе \*/

{

SYM [ISYM].TYPE = 'U'; /\* устанавливаем тип иден-\*/

/\* тификатора = 'U' и \*/

return 2; /\* завершаем программу \*/

/\* по ошибке \*/

}

/\* успешное завешение \*/

/\* программы \*/

}

int AVI2 ()

{

char i;

FORM (); /\*форматируем правую часть\*/

/\*арифметического ПЛ1-опе-\*/

/\*ратора присваивания \*/

if ( IFORMT == 1 ) /\* если правая часть одно-\*/

{ /\* термовая, то: \*/

for ( i = 0; i < ISYM; i++ ) /\* ищем этот терм в табли-\*/

{ /\* це имен и \*/

if ( !strcmp ( SYM [i].NAME, FORMT [0] ) &&/\* если находим, то: \*/

strlen ( SYM [i].NAME ) ==

strlen ( FORMT [0] )

)

{

if ( SYM [i].TYPE == 'B' ) /\* в случае типа=bin fixed\*/

{

if ( strcmp ( SYM [i].RAZR, "15" ) /\* и разрядности <= 15 \*/

<= 0 )

memcpy ( ASS\_CARD.\_BUFCARD.OPERAC, /\* формируем код ассембле-\*/

"LH", 2 );/\* ровской операции LH, \*/

else

memcpy ( ASS\_CARD.\_BUFCARD.OPERAC, /\* а при разрядности >15 \*/

"L", 1 );/\* формируем код ассембле-\*/

/\* ровской операции L \*/

strcpy ( ASS\_CARD.\_BUFCARD.OPERAND, /\* формируем \*/

"RRAB," );/\* первый и \*/

strcat ( ASS\_CARD.\_BUFCARD.OPERAND, /\* второй операнды ассемб-\*/

FORMT [0]);/\* леровской операции \*/

ASS\_CARD.\_BUFCARD.OPERAND [ strlen /\* вставляем разделитель \*/

( ASS\_CARD.\_BUFCARD.OPERAND ) ] = ' ';

memcpy ( ASS\_CARD.\_BUFCARD.COMM, /\* и построчный коментарий\*/

"Загрузка переменной в регистр", 29 );

ZKARD (); /\* запомнить операцию ас- \*/

/\* семблера и \*/

return 0; /\* завершить программу \*/

}

**else if ( SYM [i].TYPE == 'T' )**

**{**

**return 0;**

**}**

else

return 3; /\* если тип терма не bin \*/

/\* fixed,то выход по ошиб-\*/

/\* ке \*/

}

}

return 4; /\* если терм-идентификатор\*/

/\* неопределен, то выход \*/

/\* по ошибке \*/

}

else /\* если правая часть ариф-\*/

/\* метического выражения \*/

/\* двухтермовая, то: \*/

{

for ( i = 0; i < ISYM; i++ ) /\* если правый терм ариф- \*/

{ /\* метического выражения \*/

if ( !strcmp ( SYM [i].NAME, /\*определен в табл.SYM,то:\*/

FORMT [IFORMT-1] ) &&

strlen ( SYM [i].NAME ) ==

strlen ( FORMT [IFORMT-1] )

)

{

if ( SYM [i].TYPE == 'B' ) /\* если тип правого опе- \*/

{ /\* ранда bin fixed, то: \*/

if ( STROKA [ DST [I2].DST4 - /\* если знак опер."+",то: \*/

strlen( FORMT [IFORMT-1] ) ] == '+' )

{

if ( strcmp ( SYM [i].RAZR, "15" ) /\* если разрядность прав. \*/

<= 0 )/\* операнда <= 15, то: \*/

memcpy ( ASS\_CARD.\_BUFCARD.OPERAC,

"AH", 2 );/\* формируем код ассембле-\*/

else /\* ровской операции "AH",а\*/

memcpy ( ASS\_CARD.\_BUFCARD.OPERAC,

"A", 1 );/\* иначе - "A" \*/

}

else

{

if ( STROKA [ DST [I2].DST4 - /\* если же знак операции \*/

strlen ( FORMT [IFORMT-1] ) ] == /\* арифметического выра- \*/

'-' )/\* жения "-", то: \*/

{

if ( strcmp ( SYM [i].RAZR, "15" )/\* при разрядности ариф- \*/

<= 0 )/\* метич.выраж.<= 15 \*/

memcpy( ASS\_CARD.\_BUFCARD.OPERAC,/\* формируем код ассембле-\*/

"SH", 2 );/\* ровской операции "SH",F\*/

else

memcpy( ASS\_CARD.\_BUFCARD.OPERAC,/\* иначе - "S" \*/

"S", 1 );

}

else

return 5; /\* если знак операции не \*/

/\* "+" и не "-", то завер-\*/

/\* шение программы по \*/

/\* ошибке \*/

}

/\* формируем: \*/

strcpy ( ASS\_CARD.\_BUFCARD.OPERAND, /\* - первый операнд ассем-\*/

"RRAB," );/\*блеровской операции; \*/

strcat ( ASS\_CARD.\_BUFCARD.OPERAND, /\* - второй операнд ассем-\*/

FORMT [IFORMT-1] );/\*блеровской операции; \*/

ASS\_CARD.\_BUFCARD.OPERAND [ strlen

( ASS\_CARD.\_BUFCARD.OPERAND )] =/\* - разделяющий пробел; \*/

' ';

memcpy ( ASS\_CARD.\_BUFCARD.COMM,

"Формирование промежуточного значения",/\* - построчный коментарий\*/

36 );

ZKARD (); /\* запоминание ассембле- \*/

/\* ровской операции \*/

return 0; /\* успешное завершение \*/

/\* пограммы \*/

}

**else if ( SYM [i].TYPE == 'T' )**

**{**

**if ( STROKA [ DST [I2].DST4 - /\* если знак опер."!",то: \*/**

**strlen( FORMT [IFORMT-1] ) ] == '!' )**

**{**

**strcpy ( SYM [ISYM].NAME, "MID" );**

**strcpy ( SYM [ISYM].RAZR, "16" );**

**SYM [ISYM].TYPE = 'T';**

**strcpy ( SYM [ISYM++].INIT, "B" );**

**memcpy ( ASS\_CARD.\_BUFCARD.OPERAC,**

**"L", 1 );**

**strcpy ( ASS\_CARD.\_BUFCARD.OPERAND,**

**"RRAB," );**

**strcat ( ASS\_CARD.\_BUFCARD.OPERAND,**

**"@V");**

**strcat ( ASS\_CARD.\_BUFCARD.OPERAND,**

**FORMT [IFORMT - 1]);**

**ASS\_CARD.\_BUFCARD.OPERAND [ strlen**

**( ASS\_CARD.\_BUFCARD.OPERAND ) ] = ' ';**

**ZKARD ();**

**memcpy ( ASS\_CARD.\_BUFCARD.OPERAC,**

**"L", 1 );**

**strcpy ( ASS\_CARD.\_BUFCARD.OPERAND,**

**"RSH," );**

**strcat ( ASS\_CARD.\_BUFCARD.OPERAND,**

**"@L");**

**strcat ( ASS\_CARD.\_BUFCARD.OPERAND,**

**FORMT [IFORMT - 2]);**

**ASS\_CARD.\_BUFCARD.OPERAND [ strlen**

**( ASS\_CARD.\_BUFCARD.OPERAND ) ] = ' ';**

**ZKARD ();**

**memcpy ( ASS\_CARD.\_BUFCARD.OPERAC,**

**"SLR", 3 );**

**strcpy ( ASS\_CARD.\_BUFCARD.OPERAND,**

**"RRAB," );**

**strcat ( ASS\_CARD.\_BUFCARD.OPERAND,**

**"0(0,6)");**

**ASS\_CARD.\_BUFCARD.OPERAND [ strlen**

**( ASS\_CARD.\_BUFCARD.OPERAND ) ] = ' ';**

**ZKARD ();**

**memcpy ( ASS\_CARD.\_BUFCARD.OPERAC,**

**"O", 1 );**

**strcpy ( ASS\_CARD.\_BUFCARD.OPERAND,**

**"RRAB," );**

**strcat ( ASS\_CARD.\_BUFCARD.OPERAND,**

**"@V");**

**strcat ( ASS\_CARD.\_BUFCARD.OPERAND,**

**FORMT [IFORMT - 2]);**

**ASS\_CARD.\_BUFCARD.OPERAND [ strlen**

**( ASS\_CARD.\_BUFCARD.OPERAND ) ] = ' ';**

**ZKARD ();**

**memcpy ( ASS\_CARD.\_BUFCARD.OPERAC,**

**"ST", 2 );**

**strcpy ( ASS\_CARD.\_BUFCARD.OPERAND,**

**"RRAB," );**

**strcat ( ASS\_CARD.\_BUFCARD.OPERAND,**

**"@VMID");**

**ASS\_CARD.\_BUFCARD.OPERAND [ strlen**

**( ASS\_CARD.\_BUFCARD.OPERAND ) ] = ' ';**

**ZKARD ();**

**memcpy ( ASS\_CARD.\_BUFCARD.OPERAC,**

**"A", 1 );**

**strcpy ( ASS\_CARD.\_BUFCARD.OPERAND,**

**"RSH," );**

**strcat ( ASS\_CARD.\_BUFCARD.OPERAND,**

**"@L");**

**strcat ( ASS\_CARD.\_BUFCARD.OPERAND,**

**FORMT [IFORMT - 1]);**

**ASS\_CARD.\_BUFCARD.OPERAND [ strlen**

**( ASS\_CARD.\_BUFCARD.OPERAND ) ] = ' ';**

**ZKARD ();**

**memcpy ( ASS\_CARD.\_BUFCARD.OPERAC,**

**"ST", 2 );**

**strcpy ( ASS\_CARD.\_BUFCARD.OPERAND,**

**"RSH," );**

**strcat ( ASS\_CARD.\_BUFCARD.OPERAND,**

**"@LMID");**

**ASS\_CARD.\_BUFCARD.OPERAND [ strlen**

**( ASS\_CARD.\_BUFCARD.OPERAND ) ] = ' ';**

**ZKARD ();**

**return 0;**

**}**

}

else

return 3; /\* если тип правого опе- \*/

/\* ранда арифметического \*/

/\* выражения не bin fixed,\*/

/\* то завершение програм- \*/

/\* мы по ошибке \*/

}

}

return 4; /\* если правый операнд \*/

/\* арифметического выраже-\*/

/\*ния не определен в табл.\*/

/\* SYM, то завершить про- \*/

/\* грамму по ошибке \*/

}

}

int OEN2 ()

{

char RAB [20];

char TEMP [20];

char j = 0;

char i = 0;

FORM (); /\* форматируем ПЛ1-опера- \*/

/\* тор END \*/

memcpy ( ASS\_CARD.\_BUFCARD.OPERAC, "BCR", 3 ); /\* формируем код безуслов-\*/

/\*ного возврата управления\*/

/\* в вызывающую программу \*/

memcpy ( ASS\_CARD.\_BUFCARD.OPERAND,"15,14", 5 );/\* операнды команды и \*/

memcpy ( ASS\_CARD.\_BUFCARD.COMM, /\* поле построчного комен-\*/

"Выход из программы", 18 );/\* тария \*/

ZKARD (); /\* запомнить опреацию \*/

/\* Ассемблера \*/

/\* далее идет блок форми- \*/

/\* рования декларативных \*/

/\* псевдоопераций DC для \*/

/\* каждого идентификатора,\*/

/\* попавшего в табл.SYM \*/

for ( i = 0; i < ISYM; i++ )

{ /\* если строка табл.SYM \*/

if ( isalpha ( SYM [i].NAME [0] ) ) /\* содержит идентификатор,\*/

/\* т.е.начинается с буквы,\*/

{ /\* то: \*/

if ( SYM [i].TYPE == 'B' ) /\* если тип оператора bin \*/

/\* fixed, то: \*/

{

strcpy ( ASS\_CARD.\_BUFCARD.METKA, /\* пишем идентификатор в \*/

SYM [i].NAME ); /\* поле метки псевдоопера-\*/

/\* ции DC \*/

ASS\_CARD.\_BUFCARD.METKA [ strlen

( ASS\_CARD.\_BUFCARD.METKA ) ] = ' '; /\* пишем разделитель полей\*/

memcpy ( ASS\_CARD.\_BUFCARD.OPERAC, /\* пишем код псевдоопера- \*/

"DC", 2 ); /\* ции DC \*/

if ( strcmp ( SYM [i].RAZR, "15" ) <= 0 ) /\* формируем операнды псе-\*/

/\* вдооперации DC \*/

strcpy ( ASS\_CARD.\_BUFCARD.OPERAND, /\* для случая полуслова \*/

"H\'" );

else /\* или \*/

strcpy ( ASS\_CARD.\_BUFCARD.OPERAND, /\* для случая слова \*/ "F\'" );

//Dos command

// strcat ( ASS\_CARD.\_BUFCARD.OPERAND, /\* формируем цифровую \*/

// ltoa ( VALUE (SYM [i].INIT), /\* часть операнда псевдо- \*/

// &RAB [0], 10) ); /\* операции, \*/

//let's do that in Unix!

strcat(ASS\_CARD.\_BUFCARD.OPERAND, gcvt(VALUE(SYM[i].INIT), 10, &RAB[0]));

ASS\_CARD.\_BUFCARD.OPERAND [ strlen /\* замыкающий апостроф \*/

( ASS\_CARD.\_BUFCARD.OPERAND ) ] = '\''; /\* и \*/

memcpy ( ASS\_CARD.\_BUFCARD.COMM, /\* поле построчного комен-\*/

"Определение переменной", 22 ); /\* тария \*/

ZKARD (); /\* запомнить операцию \*/

/\* Ассемблера \*/

}

**else if ( SYM [i].TYPE == 'T' ) /\* если тип оператора bit \*/**

**/\* то: \*/**

**{**

**strcpy ( ASS\_CARD.\_BUFCARD.METKA, SYM [i].NAME );**

**ASS\_CARD.\_BUFCARD.METKA [ strlen( ASS\_CARD.\_BUFCARD.METKA ) ] = ' ';**

**memcpy ( ASS\_CARD.\_BUFCARD.OPERAC,"DC", 2 );**

**strcpy ( ASS\_CARD.\_BUFCARD.OPERAND, "F\'" ); /\* поле макс. длинны \*/**

**strcat(ASS\_CARD.\_BUFCARD.OPERAND, SYM [i].RAZR);**

**ASS\_CARD.\_BUFCARD.OPERAND [ strlen**

**( ASS\_CARD.\_BUFCARD.OPERAND ) ] = '\'';**

**ZKARD ();**

**j = 0;**

**while ( SYM[i].INIT[j] != 'B' )**

**{**

**TEMP[j] = SYM[i].INIT[j];**

**j++;**

**}**

**TEMP[j] = 0;**

**memcpy ( ASS\_CARD.\_BUFCARD.METKA,"@L", 2 );**

**strcpy ( ASS\_CARD.\_BUFCARD.METKA + 2, SYM [i].NAME );**

**ASS\_CARD.\_BUFCARD.METKA [ strlen( ASS\_CARD.\_BUFCARD.METKA ) ] = ' ';**

**memcpy ( ASS\_CARD.\_BUFCARD.OPERAC,"DC", 2 );**

**strcpy ( ASS\_CARD.\_BUFCARD.OPERAND, "F\'" ); /\* поле факт. длинны \*/**

**strcat(ASS\_CARD.\_BUFCARD.OPERAND, gcvt(j, 10, &RAB[0]));**

**ASS\_CARD.\_BUFCARD.OPERAND [ strlen**

**( ASS\_CARD.\_BUFCARD.OPERAND ) ] = '\'';**

**ZKARD ();**

**memcpy ( ASS\_CARD.\_BUFCARD.METKA,"@V", 2 );**

**strcpy ( ASS\_CARD.\_BUFCARD.METKA + 2, SYM [i].NAME );**

**ASS\_CARD.\_BUFCARD.METKA [ strlen( ASS\_CARD.\_BUFCARD.METKA ) ] = ' ';**

**memcpy ( ASS\_CARD.\_BUFCARD.OPERAC,"DC", 2 );**

**strcpy ( ASS\_CARD.\_BUFCARD.OPERAND, "BL4\'" );**

**strcat(ASS\_CARD.\_BUFCARD.OPERAND, TEMP ); /\* поле значения \*/**

**ASS\_CARD.\_BUFCARD.OPERAND [ strlen**

**( ASS\_CARD.\_BUFCARD.OPERAND ) ] = '\'';**

**memcpy ( ASS\_CARD.\_BUFCARD.COMM,**

**"Определение переменной", 22 );**

**ZKARD ();**

**}**

}

}

/\* далее идет блок декла- \*/

/\* ративных ассемблеровс- \*/

/\* ких EQU-операторов, оп-\*/

/\* ределяющих базовый и \*/

/\* рабочий регистры общего\*/

/\* назначения \*/

memcpy ( ASS\_CARD.\_BUFCARD.METKA, "RBASE", 5 ); /\* формирование EQU-псев- \*/

memcpy ( ASS\_CARD.\_BUFCARD.OPERAC, "EQU",3 ); /\* дооперации определения \*/

memcpy ( ASS\_CARD.\_BUFCARD.OPERAND, "15", 2 ); /\* номера базового регист-\*/

/\* ра общего назначения \*/

/\* и \*/

ZKARD ();

/\* запоминание ее \*/

memcpy ( ASS\_CARD.\_BUFCARD.METKA, "RRAB", 4 ); /\* формирование EQU-псев- \*/

memcpy ( ASS\_CARD.\_BUFCARD.OPERAC, "EQU",3 ); /\* дооперации определения \*/

memcpy ( ASS\_CARD.\_BUFCARD.OPERAND, "5", 1 ); /\* номера базового регист-\*/

/\* ра общего назначения \*/

/\* и \*/

ZKARD ();

**memcpy ( ASS\_CARD.\_BUFCARD.METKA, "RSH", 4 ); /\* формирование EQU-псев- \*/**

**memcpy ( ASS\_CARD.\_BUFCARD.OPERAC, "EQU",3 ); /\* дооперации определения \*/**

**memcpy ( ASS\_CARD.\_BUFCARD.OPERAND, "6", 1 ); /\* номера базового регист-\*/**

**/\* ра общего назначения \*/**

**/\* и \*/**

**ZKARD (); /\* запоминание ее \*/**

memcpy ( ASS\_CARD.\_BUFCARD.OPERAC, "END", 3 ); /\* формирование кода ас- \*/

/\* семблеровской псевдо- \*/

/\* операции END, \*/

i = 0;

while ( FORMT [1][i] != '\x0' ) /\* ее операнда \*/

ASS\_CARD.\_BUFCARD.OPERAND [i] = FORMT [1][i++];/\* и \*/

memcpy ( ASS\_CARD.\_BUFCARD.COMM, /\* построчного коментария \*/

"Конец программы", 15 );

ZKARD (); /\* запоминание псевдоопе- \*/

/\* рации \*/

return 0; /\* завершение программы \*/

}

int OPA2 ()

{

char RAB [20];

int temp1;

int temp2;

int i;

FORM (); /\*форматируем ПЛ1-оператор\*/

/\*присваивания арифметич. \*/

for ( i = 0; i < ISYM; i++ )

{ /\* если идентификатор пра-\*/

/\* вой части оператора оп-\*/

if ( !strcmp ( SYM [i].NAME, FORMT [0] ) && /\* ределен ранее через \*/

strlen ( SYM [i].NAME ) == /\* оператор DCL, то: \*/

strlen ( FORMT [0] )

)

{

if ( SYM [i].TYPE == 'B' ) /\* если этот идентификатор\*/

{ /\* имеет тип bin fixed,то:\*/

if ( strcmp ( SYM [i].RAZR, "15" ) /\* если bin fixed (15),то:\*/

<= 0 )

memcpy ( ASS\_CARD.\_BUFCARD.OPERAC, /\* сформировать команду \*/

"STH", 3 );/\* записи полуслова \*/

else /\* иначе: \*/

memcpy ( ASS\_CARD.\_BUFCARD.OPERAC, /\* команду записи слова \*/

"ST", 2 );

strcpy ( ASS\_CARD.\_BUFCARD.OPERAND, /\* доформировать \*/

"RRAB," );/\* операнды \*/

strcat ( ASS\_CARD.\_BUFCARD.OPERAND, /\* команды \*/

FORMT [0]) ;

ASS\_CARD.\_BUFCARD.OPERAND [ strlen /\* и \*/

( ASS\_CARD.\_BUFCARD.OPERAND ) ] = ' ';

memcpy ( ASS\_CARD.\_BUFCARD.COMM, /\* построчный коментарий \*/

"Формирование значения арифм.выражения",

37 );

ZKARD (); /\* запомнить операцию \*/

/\* Ассемблера и \*/

return 0; /\* завершить программу \*/

}

**else if ( SYM [i].TYPE == 'T' )**

**{**

**if (!strcmp(FORMT [1], "SUBSTR") )**

**{**

**temp1 = atoi(FORMT [IFORMT - 3]);**

**strcpy ( SYM [ISYM].NAME, "SL" );**

**strcpy ( SYM [ISYM].RAZR, "33" );**

**SYM [ISYM].TYPE = 'B';**

**strcpy ( SYM [ISYM++].INIT, gcvt(temp1 - 1, 10, &RAB[0]) );**

**temp2 = atoi(FORMT [IFORMT - 2]);**

**strcpy ( SYM [ISYM].NAME, "NL" );**

**strcpy ( SYM [ISYM].RAZR, "33" );**

**SYM [ISYM].TYPE = 'B';**

**strcpy ( SYM [ISYM++].INIT, gcvt(temp2, 10, &RAB[0]) );**

**strcpy ( SYM [ISYM].NAME, "MSK" );**

**strcpy ( SYM [ISYM].RAZR, "33" );**

**SYM [ISYM].TYPE = 'X';**

**for(temp1 = 0; temp1 < temp2; ++temp1)**

**{**

**RAB[temp1] = '1';**

**}**

**RAB[temp1++] = 'B';**

**RAB[temp1] = 0;**

**strcpy ( SYM [ISYM++].INIT, RAB );**

**memcpy ( ASS\_CARD.\_BUFCARD.OPERAC,"L", 1 );**

**strcpy ( ASS\_CARD.\_BUFCARD.OPERAND, RRAB," );**

**strcat ( ASS\_CARD.\_BUFCARD.OPERAND, "@VMID");**

**ASS\_CARD.\_BUFCARD.OPERAND [ strlen**

**( ASS\_CARD.\_BUFCARD.OPERAND ) ] = ' ';**

**ZKARD ();**

**memcpy ( ASS\_CARD.\_BUFCARD.OPERAC,"L", 1 );**

**strcpy ( ASS\_CARD.\_BUFCARD.OPERAND, RSH," );**

**strcat ( ASS\_CARD.\_BUFCARD.OPERAND, "SL");**

**ASS\_CARD.\_BUFCARD.OPERAND [ strlen**

**( ASS\_CARD.\_BUFCARD.OPERAND ) ] = ' ';**

**ZKARD ();**

**memcpy ( ASS\_CARD.\_BUFCARD.OPERAC, "SLL", 3 );**

**strcpy ( ASS\_CARD.\_BUFCARD.OPERAND, RRAB," );**

**strcat ( ASS\_CARD.\_BUFCARD.OPERAND, 0(0,6)");**

**ASS\_CARD.\_BUFCARD.OPERAND [ strlen**

**( ASS\_CARD.\_BUFCARD.OPERAND ) ] = ' ';**

**ZKARD ();**

**memcpy ( ASS\_CARD.\_BUFCARD.OPERAC,"N", 1 );**

**strcpy ( ASS\_CARD.\_BUFCARD.OPERAND, RRAB," );**

**strcat ( ASS\_CARD.\_BUFCARD.OPERAND, MSK");**

**ASS\_CARD.\_BUFCARD.OPERAND [ strlen**

**( ASS\_CARD.\_BUFCARD.OPERAND ) ] = ' ';**

**ZKARD ();**

**memcpy ( ASS\_CARD.\_BUFCARD.OPERAC,"ST", 2 );**

**strcpy ( ASS\_CARD.\_BUFCARD.OPERAND, "RRAB," );**

**strcat ( ASS\_CARD.\_BUFCARD.OPERAND, "@V");**

**strcat ( ASS\_CARD.\_BUFCARD.OPERAND, FORMT [0]);**

**ASS\_CARD.\_BUFCARD.OPERAND [ strlen**

**( ASS\_CARD.\_BUFCARD.OPERAND ) ] = ' ';**

**ZKARD ();**

**memcpy ( ASS\_CARD.\_BUFCARD.OPERAC,**

**"L", 1 );**

**strcpy ( ASS\_CARD.\_BUFCARD.OPERAND,**

**"RSH," );**

**strcat ( ASS\_CARD.\_BUFCARD.OPERAND, "NL");**

**ASS\_CARD.\_BUFCARD.OPERAND [ strlen ( ASS\_CARD.\_BUFCARD.OPERAND ) ] = ' ';**

**ZKARD ();**

**memcpy ( ASS\_CARD.\_BUFCARD.OPERAC,"ST", 2 );**

**strcpy ( ASS\_CARD.\_BUFCARD.OPERAND, "RSH," );**

**strcat ( ASS\_CARD.\_BUFCARD.OPERAND, "@L");**

**strcat ( ASS\_CARD.\_BUFCARD.OPERAND, FORMT [0]);**

**ASS\_CARD.\_BUFCARD.OPERAND [ strlen**

**( ASS\_CARD.\_BUFCARD.OPERAND ) ] = ' ';**

**ZKARD ();**

**return 0;**

}

}

else /\* если идентификатор не \*/

/\* имеет тип bin fixed,то:\*/

return 3; /\* завершение с диагности-\*/

/\* кой ошибки \*/

}

}

return 4; /\* если идентификатор ра- \*/

/\* нее не определен через \*/

/\* ПЛ1-оператор DCL,то за-\*/

/\* вершение с диагностикой\*/

/\* ошибки \*/

}

# Заключение

*В процессе выполнения работы, с проблемами не столкнулись.*

*Плюсы данной реализации:*

1. *Сохранение функциональных способностей в старой системе и добавление новых.*
2. *Полноценная работа программы с битовыми строками длиной не более 16 бит.*
3. *Независимая реализация операций подстроки и конкатенации – при изменении входных данных потребуется меньшее колличество изменений.*

*Минусы данной реализации:*

1. *Алгоритм не рассматривает ситуации выхода за пределы доспустимых длин строк (при конкатенации строк не проверяется длина результирующей строки, как и при взятии подстроки не проверяется что строка-аргумент имеет длину не меньше заданного колличесва вырезаемых символов.*

# Разработка элементов учебной системы программирования

по дисциплине «Системы программирования».

# Компилятор ассемблера.

Вариант №7

Выполнили студенты гр. 5084/13 «подпись» Благодиров А.В.

«подпись» Черкалова В.В.

«подпись» Афанасьева В.Ю.

Руководитель доцент «подпись» Расторгуев В.Я.

«\_\_\_»\_\_\_\_\_\_\_\_2012

# Введение

Данная курсовая работа имеет своей целью получение практических навыков построения компилятора с языка высокого уровня (ЯВУ), являющегося одним из элементов системы программирования, образующих в совокупности следующий технологический конвейер:



При этом предполагается то, что данная система программирования работает на технологической ЭВМ (IBM PC) и является по существу кросс-системой для объектной ЭВМ (ЕС ЭВМ). В этой системе:

* в качестве языка высокого уровня (ЯВУ) выбран язык, образованный из подмножества языковых конструкций ПЛ1, а исходная программа готовится в виде текстового файла технологической ЭВМ с расширением \*.pli;
* язык АССЕМБЛЕРА сформирован из языковых конструкций АССЕМБЛЕРА ЕС ЭВМ, а ассемблеровский эквивалент исходной программы формируется в виде текстового файла технологической ЭВМ с расширением \*.ass;
* объектный эквивалент исходной программы готовится в формате объектных файлов операционной системы ОС ЕС ЭВМ и хранится в виде двоичного файла технологической ЭВМ с расширением \*.tex;
* загрузочный эквивалент исходной программы представляет собой машинный код ЕС ЭВМ, запоминаемый в области ОЗУ технологической ЭВМ, являющейся зоной загрузки для эмулятора объектной ЭВМ.

# Постановка задачи

Необходимо выполнить доработку элементов макета учебной системы программирования до уровня, позволяющего обрабатывать “новые” для макета конструкции языка высокого уровня, примененные в соответствующем варианте:



Где на входе имеется исходный код программы на ассемблере ЭВМ IBM 370:

EX07: PROC OPTIONS ( MAIN );

DCL A BIT ( 3 ) INIT ( '10'B );

DCL B BIT ( 3 ) INIT ( '101'B );

DCL C BIT ( 16 );

C = SUBSTR (( B !! A ), 2, 3 );

END EX07;

На выходе строится эквивалент программы на ассемблере архитектуры IBM 370:

EX07 START 0

BALR RBASE,0

USING \*,RBASE

L RRAB,@VA

L RSH,@LB

SLR RRAB,0(0,6)

O RRAB,@VB

ST RRAB,@VMID

A RSH,@LA

ST RSH,@LMID

L RRAB,@VMID

L RSH,SL

SLL RRAB,0(0,6)

N RRAB,MSK

ST RRAB,@VC

L RSH,NL

ST RSH,@LC

BCR 15,14

A DC F'3'

@LA DC F'2'

@VA DC BL4'10'

B DC F'3'

@LB DC F'3'

@VB DC BL4'101'

C DC F'16'

@LC DC F'0'

@VC DC BL4''

MID DC F'16'

@LMID DC F'0'

@VMID DC BL4''

SL DC F'1'

NL DC F'3'

MSK DC BL4'111'

RBASE EQU 15

RRAB EQU 5

RSH EQU 6

END EX07

На выходе эквивалент исходной программы в виде байт-кода для IBM 370:

Offset 0 1 2 3 4 5 6 7 8 9 A B C D E F

00000000 02 45 53 44 │ 40 40 40 40 │ 40 40 00 10 │ 40 40 00 01 .ESD@@@@@@..@@..

00000010 45 58 30 37 │ 40 40 40 40 │ 00 00 00 00 │ 40 00 00 78 EX07@@@@....@..x

00000020 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

00000030 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

00000040 40 40 40 40 │ 40 40 40 40 │ 45 58 30 37 │ 40 40 40 40 @@@@@@@@EX07@@@@

00000050 02 54 58 54 │ 40 00 00 00 │ 40 40 00 02 │ 40 40 00 01 .TXT@...@@..@@..

00000060 05 F0 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 .ð@@@@@@@@@@@@@@

00000070 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

00000080 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

00000090 40 40 40 40 │ 40 40 40 40 │ 45 58 30 37 │ 40 40 40 40 @@@@@@@@EX07@@@@

000000A0 02 54 58 54 │ 40 00 00 02 │ 40 40 00 04 │ 40 40 00 01 .TXT@...@@..@@..

000000B0 58 50 F0 42 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 XPðB@@@@@@@@@@@@

000000C0 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

000000D0 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

000000E0 40 40 40 40 │ 40 40 40 40 │ 45 58 30 37 │ 40 40 40 40 @@@@@@@@EX07@@@@

000000F0 02 54 58 54 │ 40 00 00 06 │ 40 40 00 04 │ 40 40 00 01 .TXT@...@@..@@..

00000100 58 60 F0 4A │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 X`ðJ@@@@@@@@@@@@

00000110 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

00000120 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

00000130 40 40 40 40 │ 40 40 40 40 │ 45 58 30 37 │ 40 40 40 40 @@@@@@@@EX07@@@@

00000140 02 54 58 54 │ 40 00 00 0A │ 40 40 00 04 │ 40 40 00 01 .TXT@...@@..@@..

00000150 88 50 60 00 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 .P`.@@@@@@@@@@@@

00000160 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

00000170 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

00000180 40 40 40 40 │ 40 40 40 40 │ 45 58 30 37 │ 40 40 40 40 @@@@@@@@EX07@@@@

00000190 02 54 58 54 │ 40 00 00 0E │ 40 40 00 04 │ 40 40 00 01 .TXT@...@@..@@..

000001A0 56 50 F0 4E │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 VPðN@@@@@@@@@@@@

000001B0 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

000001C0 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

000001D0 40 40 40 40 │ 40 40 40 40 │ 45 58 30 37 │ 40 40 40 40 @@@@@@@@EX07@@@@

000001E0 02 54 58 54 │ 40 00 00 12 │ 40 40 00 04 │ 40 40 00 01 .TXT@...@@..@@..

000001F0 50 50 F0 66 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 PPðf@@@@@@@@@@@@

00000200 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

00000210 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

00000220 40 40 40 40 │ 40 40 40 40 │ 45 58 30 37 │ 40 40 40 40 @@@@@@@@EX07@@@@

00000230 02 54 58 54 │ 40 00 00 16 │ 40 40 00 04 │ 40 40 00 01 .TXT@...@@..@@..

00000240 5A 60 F0 3E │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 Z`ð>@@@@@@@@@@@@

00000250 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

00000260 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

00000270 40 40 40 40 │ 40 40 40 40 │ 45 58 30 37 │ 40 40 40 40 @@@@@@@@EX07@@@@

00000280 02 54 58 54 │ 40 00 00 1A │ 40 40 00 04 │ 40 40 00 01 .TXT@...@@..@@..

00000290 50 60 F0 62 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 P`ðb@@@@@@@@@@@@

000002A0 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

000002B0 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

000002C0 40 40 40 40 │ 40 40 40 40 │ 45 58 30 37 │ 40 40 40 40 @@@@@@@@EX07@@@@

000002D0 02 54 58 54 │ 40 00 00 1E │ 40 40 00 04 │ 40 40 00 01 .TXT@...@@..@@..

000002E0 58 50 F0 66 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 XPðf@@@@@@@@@@@@

000002F0 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

00000300 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

00000310 40 40 40 40 │ 40 40 40 40 │ 45 58 30 37 │ 40 40 40 40 @@@@@@@@EX07@@@@

00000320 02 54 58 54 │ 40 00 00 22 │ 40 40 00 04 │ 40 40 00 01 .TXT@.."@@..@@..

00000330 58 60 F0 6A │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 X`ðj@@@@@@@@@@@@

00000340 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

00000350 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

00000360 40 40 40 40 │ 40 40 40 40 │ 45 58 30 37 │ 40 40 40 40 @@@@@@@@EX07@@@@

00000370 02 54 58 54 │ 40 00 00 26 │ 40 40 00 04 │ 40 40 00 01 .TXT@..&@@..@@..

00000380 89 50 60 00 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 .P`.@@@@@@@@@@@@

00000390 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

000003A0 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

000003B0 40 40 40 40 │ 40 40 40 40 │ 45 58 30 37 │ 40 40 40 40 @@@@@@@@EX07@@@@

000003C0 02 54 58 54 │ 40 00 00 2A │ 40 40 00 04 │ 40 40 00 01 .TXT@..\*@@..@@..

000003D0 54 50 F0 72 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 TPðr@@@@@@@@@@@@

000003E0 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

000003F0 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

00000400 40 40 40 40 │ 40 40 40 40 │ 45 58 30 37 │ 40 40 40 40 @@@@@@@@EX07@@@@

00000410 02 54 58 54 │ 40 00 00 2E │ 40 40 00 04 │ 40 40 00 01 .TXT@...@@..@@..

00000420 50 50 F0 5A │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 PPðZ@@@@@@@@@@@@

00000430 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

00000440 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

00000450 40 40 40 40 │ 40 40 40 40 │ 45 58 30 37 │ 40 40 40 40 @@@@@@@@EX07@@@@

00000460 02 54 58 54 │ 40 00 00 32 │ 40 40 00 04 │ 40 40 00 01 .TXT@..2@@..@@..

00000470 58 60 F0 6E │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 X`ðn@@@@@@@@@@@@

00000480 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

00000490 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

000004A0 40 40 40 40 │ 40 40 40 40 │ 45 58 30 37 │ 40 40 40 40 @@@@@@@@EX07@@@@

000004B0 02 54 58 54 │ 40 00 00 36 │ 40 40 00 04 │ 40 40 00 01 .TXT@..6@@..@@..

000004C0 50 60 F0 56 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 P`ðV@@@@@@@@@@@@

000004D0 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

000004E0 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

000004F0 40 40 40 40 │ 40 40 40 40 │ 45 58 30 37 │ 40 40 40 40 @@@@@@@@EX07@@@@

00000500 02 54 58 54 │ 40 00 00 3A │ 40 40 00 02 │ 40 40 00 01 .TXT@..:@@..@@..

00000510 07 FE 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 .þ@@@@@@@@@@@@@@

00000520 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

00000530 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

00000540 40 40 40 40 │ 40 40 40 40 │ 45 58 30 37 │ 40 40 40 40 @@@@@@@@EX07@@@@

00000550 02 54 58 54 │ 40 00 00 3C │ 40 40 00 04 │ 40 40 00 01 .TXT@..<@@..@@..

00000560 00 00 00 03 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 ....@@@@@@@@@@@@

00000570 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

00000580 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

00000590 40 40 40 40 │ 40 40 40 40 │ 45 58 30 37 │ 40 40 40 40 @@@@@@@@EX07@@@@

000005A0 02 54 58 54 │ 40 00 00 40 │ 40 40 00 04 │ 40 40 00 01 .TXT@..@@@..@@..

000005B0 00 00 00 02 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 ....@@@@@@@@@@@@

000005C0 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

000005D0 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

000005E0 40 40 40 40 │ 40 40 40 40 │ 45 58 30 37 │ 40 40 40 40 @@@@@@@@EX07@@@@

000005F0 02 54 58 54 │ 40 00 00 44 │ 40 40 00 04 │ 40 40 00 01 .TXT@..D@@..@@..

00000600 80 00 00 00 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 ....@@@@@@@@@@@@

00000610 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

00000620 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

00000630 40 40 40 40 │ 40 40 40 40 │ 45 58 30 37 │ 40 40 40 40 @@@@@@@@EX07@@@@

00000640 02 54 58 54 │ 40 00 00 48 │ 40 40 00 04 │ 40 40 00 01 .TXT@..H@@..@@..

00000650 00 00 00 03 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 ....@@@@@@@@@@@@

00000660 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

00000670 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

00000680 40 40 40 40 │ 40 40 40 40 │ 45 58 30 37 │ 40 40 40 40 @@@@@@@@EX07@@@@

00000690 02 54 58 54 │ 40 00 00 4C │ 40 40 00 04 │ 40 40 00 01 .TXT@..L@@..@@..

000006A0 00 00 00 03 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 ....@@@@@@@@@@@@

000006B0 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

000006C0 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

000006D0 40 40 40 40 │ 40 40 40 40 │ 45 58 30 37 │ 40 40 40 40 @@@@@@@@EX07@@@@

000006E0 02 54 58 54 │ 40 00 00 50 │ 40 40 00 04 │ 40 40 00 01 .TXT@..P@@..@@..

000006F0 A0 00 00 00 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40  ...@@@@@@@@@@@@

00000700 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

00000710 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

00000720 40 40 40 40 │ 40 40 40 40 │ 45 58 30 37 │ 40 40 40 40 @@@@@@@@EX07@@@@

00000730 02 54 58 54 │ 40 00 00 54 │ 40 40 00 04 │ 40 40 00 01 .TXT@..T@@..@@..

00000740 00 00 00 10 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 ....@@@@@@@@@@@@

00000750 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

00000760 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

00000770 40 40 40 40 │ 40 40 40 40 │ 45 58 30 37 │ 40 40 40 40 @@@@@@@@EX07@@@@

00000780 02 54 58 54 │ 40 00 00 58 │ 40 40 00 04 │ 40 40 00 01 .TXT@..X@@..@@..

00000790 00 00 00 01 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 ....@@@@@@@@@@@@

000007A0 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

000007B0 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

000007C0 40 40 40 40 │ 40 40 40 40 │ 45 58 30 37 │ 40 40 40 40 @@@@@@@@EX07@@@@

000007D0 02 54 58 54 │ 40 00 00 5C │ 40 40 00 04 │ 40 40 00 01 .TXT@..\@@..@@..

000007E0 00 00 00 00 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 ....@@@@@@@@@@@@

000007F0 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

00000800 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

00000810 40 40 40 40 │ 40 40 40 40 │ 45 58 30 37 │ 40 40 40 40 @@@@@@@@EX07@@@@

00000820 02 54 58 54 │ 40 00 00 60 │ 40 40 00 04 │ 40 40 00 01 .TXT@..`@@..@@..

00000830 00 00 00 10 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 ....@@@@@@@@@@@@

00000840 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

00000850 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

00000860 40 40 40 40 │ 40 40 40 40 │ 45 58 30 37 │ 40 40 40 40 @@@@@@@@EX07@@@@

00000870 02 54 58 54 │ 40 00 00 64 │ 40 40 00 04 │ 40 40 00 01 .TXT@..d@@..@@..

00000880 00 00 00 01 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 ....@@@@@@@@@@@@

00000890 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

000008A0 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

000008B0 40 40 40 40 │ 40 40 40 40 │ 45 58 30 37 │ 40 40 40 40 @@@@@@@@EX07@@@@

000008C0 02 54 58 54 │ 40 00 00 68 │ 40 40 00 04 │ 40 40 00 01 .TXT@..h@@..@@..

000008D0 00 00 00 00 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 ....@@@@@@@@@@@@

000008E0 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

000008F0 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

00000900 40 40 40 40 │ 40 40 40 40 │ 45 58 30 37 │ 40 40 40 40 @@@@@@@@EX07@@@@

00000910 02 54 58 54 │ 40 00 00 6C │ 40 40 00 04 │ 40 40 00 01 .TXT@..l@@..@@..

00000920 00 00 00 01 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 ....@@@@@@@@@@@@

00000930 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

00000940 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

00000950 40 40 40 40 │ 40 40 40 40 │ 45 58 30 37 │ 40 40 40 40 @@@@@@@@EX07@@@@

00000960 02 54 58 54 │ 40 00 00 70 │ 40 40 00 04 │ 40 40 00 01 .TXT@..p@@..@@..

00000970 00 00 00 03 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 ....@@@@@@@@@@@@

00000980 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

00000990 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

000009A0 40 40 40 40 │ 40 40 40 40 │ 45 58 30 37 │ 40 40 40 40 @@@@@@@@EX07@@@@

000009B0 02 54 58 54 │ 40 00 00 74 │ 40 40 00 04 │ 40 40 00 01 .TXT@..t@@..@@..

000009C0 60 00 00 00 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 `...@@@@@@@@@@@@

000009D0 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

000009E0 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

000009F0 40 40 40 40 │ 40 40 40 40 │ 45 58 30 37 │ 40 40 40 40 @@@@@@@@EX07@@@@

00000A00 02 45 4E 44 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 .END@@@@@@@@@@@@

00000A10 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

00000A20 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

00000A30 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

00000A40 40 40 40 40 │ 40 40 40 40 │ 45 58 30 37 │ 40 40 40 40 @@@@@@@@EX07@@@@

# Анализ поставленной задачи

*В компиляторе с ассемблера были добавлены следующие функции:*

*N регистр регистр – логическое И*

*O регистр регистр – логическое ИЛИ*

*SLR регистр метка – сдвиг вправо*

*SLL. регистр метка – сдвиг влево*

*При использовании описаннного в компиляторе с ЯВУ алгоритма данных операций достаточно, чтобы эффективно реализовать необходимые функции.*

*Кроме того, добавлена обработка нового типа данных – BR4 – битовая строка.*

# Входные ограничения

*Для данной части системы единственным ограничением можно считать корректную программу, поданную на вход (то есть использование в ней только описанных в компиляторе функций.*

# Модификация базы данных исходного макета

*Добавляем описание новых функций:*

struct TMOP /\*структ.стр.табл.маш.опер\*/

{

unsigned char MNCOP [5]; /\*мнемокод операции \*/

unsigned char CODOP ; /\*машинный код операции \*/

unsigned char DLOP ; /\*длина операции в байтах \*/

int (\*BXPROG)() ; /\*указатель на подпр.обраб\*/

} T\_MOP [NOP] = /\*об'явление табл.маш.опер\*/

{

{{'B','A','L','R',' '} , '\x05' , 2 , FRR} , /\*инициализация \*/

{{'B','C','R',' ',' '} , '\x07' , 2 , FRR} , /\*строк \*/

{{'S','T',' ',' ',' '} , '\x50' , 4 , FRX} , /\*таблицы \*/

{{'L',' ',' ',' ',' '} , '\x58' , 4 , FRX} , /\*машинных \*/

{{'A',' ',' ',' ',' '} , '\x5A' , 4 , FRX} , /\*операций \*/

{{'S',' ',' ',' ',' '} , '\x5B' , 4 , FRX} , /\* \*/

**{{'S','L','R',' ',' '} , '\x88' , 4 , FRX} , /\* \*/**

**{{'S','L','L',' ',' '} , '\x89' , 4 , FRX} , /\* \*/**

**{{'N',' ',' ',' ',' '} , '\x54' , 4 , FRX} , /\* \*/**

**{{'O',' ',' ',' ',' '} , '\x56' , 4 , FRX} /\* \*/**

};

\*/

# Модификация алгоритма исходного макета

*Описываем переходы к самим функциям:*

int SDC() /\*подпр.обр.пс.опер.DC \*/

{

char \*RAB; /\*рабочая переменная \*/

int temp = 0;

int i;

unsigned int msk = 0x80000000;

RX.OP\_RX.OP = 0; /\*занулим два старших \*/

RX.OP\_RX.R1X2 = 0; /\*байта RX.OP\_RX \*/

if

( /\* если операнд начинается\*/

!memcmp(TEK\_ISX\_KARTA.STRUCT\_BUFCARD.OPERAND,/\* с комбинации \*/

"F'", 2) /\* F', \*/

) /\* то \*/

{

RAB=strtok /\*в перем. c указат.RAB \*/

( /\*выбираем первую лексему \*/

(char\*)TEK\_ISX\_KARTA.STRUCT\_BUFCARD.OPERAND+2,/\*операнда текущей карты \*/

"'" /\*исх.текста АССЕМБЛЕРА \*/

);

RX.OP\_RX.B2D2 = atoi ( RAB ); /\*перевод ASCII-> int \*/

RAB = (char \*) &RX.OP\_RX.B2D2; /\*приведение к соглашениям\*/

swab ( RAB , RAB , 2 ); /\* ЕС ЭВМ \*/ }

**else if (!memcmp(TEK\_ISX\_KARTA.STRUCT\_BUFCARD.OPERAND, "BL4'", 4))**

**{**

**RAB=strtok((char\*)TEK\_ISX\_KARTA.STRUCT\_BUFCARD.OPERAND+4, "'");**

**for (i = 0; i< strlen(RAB); ++i)**

**{**

**if (RAB[i] == '1') /\*перевод ASCII-> int \*/**

**{**

**temp = temp msk; |**

**}**

**msk >>= 1;**

**}**

**RAB = (char \*) &temp;**

**memcpy ( RX.BUF\_OP\_RX + 3 , RAB, 1);**

**memcpy ( RX.BUF\_OP\_RX + 2 , RAB + 1 , 1); /\*приведение к соглашениям\*/**

**memcpy ( RX.BUF\_OP\_RX + 1 , RAB + 2, 1); /\* ЕС ЭВМ \*/**

**memcpy ( RX.BUF\_OP\_RX , RAB + 3 , 1);**

**}**

else /\*иначе \*/

return (1); /\*сообщение об ошибке \*/

STXT (4); /\*формирование TXT-карты \*/

return (0); /\*успешн.завершение подпр.\*/

}

if ( isalpha ( (int) \*METKA2 ) **|| \*METKA2 == '@' ) /**\*если лексема начинается \*/

{ /\*с буквы, то: \*/

for ( J=0; J<=ITSYM; J++ ) /\* все метки исх.текста в \*/

{ /\* табл. T\_SYM сравниваем \*/

/\* со знач.перем. \*МЕТКА \*/

METKA = strtok (

(char\*) T\_SYM[J].IMSYM , " "

);

if( !strcmp ( METKA , METKA2 ) ) /\* и при совпадении: \*/

{ /\* установить нач.знач.: \*/

NBASRG = 0; /\* номера базов.регистра\*/

DELTA = 0xfff - 1; /\* и его значен.,а также\*/

ZNSYM = T\_SYM[J].ZNSYM; /\* смещен.втор.операнда \*/

for ( I=0; I<15; I++ ) /\*далее в цикле из всех \*/

{ /\*рег-ров выберем базовым \*/

if ( T\_BASR[I].PRDOST == 'Y' /\* призн.активности, \*/

&& ZNSYM - T\_BASR[I].SMESH >= 0 /\* значенение, меньшее по \*/

&& ZNSYM - T\_BASR[I].SMESH < DELTA )

{NBASRG = I + 1;

DELTA = ZNSYM - T\_BASR[I].SMESH; } }

if ( NBASRG == 0 || DELTA > 0xfff ) /\*если баз.рег.не выбр.,то\*/

return(5); /\* заверш.подпр.по ошибке \*/

else /\*иначе \*/

{

/\* сформировыать машинное \*/

B2D2 = NBASRG << 12; /\* представление второго \*/

B2D2 = B2D2 + DELTA; /\* операнда в виде B2D2 \*/

PTR = (char \*)&B2D2; /\* и в соглашениях ЕС ЭВМ \*/

swab ( PTR , PTR , 2 ); /\* с записью в тело ком-ды\*/

RX.OP\_RX.B2D2 = B2D2;

}

goto SRX2; /\*перех.на форм.первого \*/

} /\* опреранда машинной ком\*/

}

return(2); /\*сообщ."необ'явл.идентиф"\*/

}

else

{

**METKA = strtok (METKA2 , "(");**

**DELTA = atoi(METKA);**

**METKA2 = METKA2 + strlen(METKA) + 1; /\* Разборе конструкции \*/**

**METKA = strtok (METKA2, ","); /\* вида 0(0,0) \*/**

**METKA2 = METKA2 + strlen(METKA) + 1;**

**METKA = strtok (METKA2, ")");**

**NBASRG = atoi(METKA);**

**/\* сформировыать машинное \*/**

**B2D2 = NBASRG << 12; /\* представление второго \*/**

**B2D2 = B2D2 + DELTA; /\* операнда в виде B2D2 \*/**

**PTR = (char \*)&B2D2; /\* и в соглашениях ЕС ЭВМ \*/**

**swab ( PTR , PTR , 2 ); /\* с записью в тело ком-ды\*/**

**RX.OP\_RX.B2D2 = B2D2;**

**goto SRX2;**

}

# Заключение

*Плюсы данной реализации:*

1. *Простота, вносится минимум изменений*
2. *Не затрагиваются уже существующая функциональность*

*Минусы данной реализации:*

1. *необходимо в алгоритме ограничиться данными функциями*

# Разработка элементов учебной системы программирования

по дисциплине «Системы программирования».

# Загрузчик, Эмулятор, Отладчик.

Вариант №7

Выполнили студенты гр. 5084/13 «подпись» Благодиров А.В.

«подпись» Черкалова В.В.

«подпись» Афанасьева В.Ю.

Руководитель доцент «подпись» Расторгуев В.Я.

«\_\_\_»\_\_\_\_\_\_\_\_2012

# Введение

Данная курсовая работа имеет своей целью получение практических навыков построения компилятора с языка высокого уровня (ЯВУ), являющегося одним из элементов системы программирования, образующих в совокупности следующий технологический конвейер:



При этом предполагается то, что данная система программирования работает на технологической ЭВМ (IBM PC) и является по существу кросс-системой для объектной ЭВМ (ЕС ЭВМ). В этой системе:

* в качестве языка высокого уровня (ЯВУ) выбран язык, образованный из подмножества языковых конструкций ПЛ1, а исходная программа готовится в виде текстового файла технологической ЭВМ с расширением \*.pli;
* язык АССЕМБЛЕРА сформирован из языковых конструкций АССЕМБЛЕРА ЕС ЭВМ, а ассемблеровский эквивалент исходной программы формируется в виде текстового файла технологической ЭВМ с расширением \*.ass;
* объектный эквивалент исходной программы готовится в формате объектных файлов операционной системы ОС ЕС ЭВМ и хранится в виде двоичного файла технологической ЭВМ с расширением \*.tex;
* загрузочный эквивалент исходной программы представляет собой машинный код ЕС ЭВМ, запоминаемый в области ОЗУ технологической ЭВМ, являющейся зоной загрузки для эмулятора объектной ЭВМ.

# Постановка задачи

Необходимо выполнить доработку элементов макета учебной системы программирования до уровня, позволяющего обрабатывать “новые” для макета конструкции языка высокого уровня, примененные в соответствующем варианте:



Где на входе исходный текст программы в виде байт-кода ЭВМ IBM 370:

Offset 0 1 2 3 4 5 6 7 8 9 A B C D E F

00000000 02 45 53 44 │ 40 40 40 40 │ 40 40 00 10 │ 40 40 00 01 .ESD@@@@@@..@@..

00000010 45 58 30 37 │ 40 40 40 40 │ 00 00 00 00 │ 40 00 00 78 EX07@@@@....@..x

00000020 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

00000030 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

00000040 40 40 40 40 │ 40 40 40 40 │ 45 58 30 37 │ 40 40 40 40 @@@@@@@@EX07@@@@

00000050 02 54 58 54 │ 40 00 00 00 │ 40 40 00 02 │ 40 40 00 01 .TXT@...@@..@@..

00000060 05 F0 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 .ð@@@@@@@@@@@@@@

00000070 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

00000080 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

00000090 40 40 40 40 │ 40 40 40 40 │ 45 58 30 37 │ 40 40 40 40 @@@@@@@@EX07@@@@

000000A0 02 54 58 54 │ 40 00 00 02 │ 40 40 00 04 │ 40 40 00 01 .TXT@...@@..@@..

000000B0 58 50 F0 42 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 XPðB@@@@@@@@@@@@

000000C0 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

000000D0 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

000000E0 40 40 40 40 │ 40 40 40 40 │ 45 58 30 37 │ 40 40 40 40 @@@@@@@@EX07@@@@

000000F0 02 54 58 54 │ 40 00 00 06 │ 40 40 00 04 │ 40 40 00 01 .TXT@...@@..@@..

00000100 58 60 F0 4A │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 X`ðJ@@@@@@@@@@@@

00000110 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

00000120 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

00000130 40 40 40 40 │ 40 40 40 40 │ 45 58 30 37 │ 40 40 40 40 @@@@@@@@EX07@@@@

00000140 02 54 58 54 │ 40 00 00 0A │ 40 40 00 04 │ 40 40 00 01 .TXT@...@@..@@..

00000150 88 50 60 00 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 .P`.@@@@@@@@@@@@

00000160 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

00000170 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

00000180 40 40 40 40 │ 40 40 40 40 │ 45 58 30 37 │ 40 40 40 40 @@@@@@@@EX07@@@@

00000190 02 54 58 54 │ 40 00 00 0E │ 40 40 00 04 │ 40 40 00 01 .TXT@...@@..@@..

000001A0 56 50 F0 4E │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 VPðN@@@@@@@@@@@@

000001B0 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

000001C0 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

000001D0 40 40 40 40 │ 40 40 40 40 │ 45 58 30 37 │ 40 40 40 40 @@@@@@@@EX07@@@@

000001E0 02 54 58 54 │ 40 00 00 12 │ 40 40 00 04 │ 40 40 00 01 .TXT@...@@..@@..

000001F0 50 50 F0 66 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 PPðf@@@@@@@@@@@@

00000200 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

00000210 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

00000220 40 40 40 40 │ 40 40 40 40 │ 45 58 30 37 │ 40 40 40 40 @@@@@@@@EX07@@@@

00000230 02 54 58 54 │ 40 00 00 16 │ 40 40 00 04 │ 40 40 00 01 .TXT@...@@..@@..

00000240 5A 60 F0 3E │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 Z`ð>@@@@@@@@@@@@

00000250 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

00000260 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

00000270 40 40 40 40 │ 40 40 40 40 │ 45 58 30 37 │ 40 40 40 40 @@@@@@@@EX07@@@@

00000280 02 54 58 54 │ 40 00 00 1A │ 40 40 00 04 │ 40 40 00 01 .TXT@...@@..@@..

00000290 50 60 F0 62 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 P`ðb@@@@@@@@@@@@

000002A0 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

000002B0 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

000002C0 40 40 40 40 │ 40 40 40 40 │ 45 58 30 37 │ 40 40 40 40 @@@@@@@@EX07@@@@

000002D0 02 54 58 54 │ 40 00 00 1E │ 40 40 00 04 │ 40 40 00 01 .TXT@...@@..@@..

000002E0 58 50 F0 66 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 XPðf@@@@@@@@@@@@

000002F0 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

00000300 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

00000310 40 40 40 40 │ 40 40 40 40 │ 45 58 30 37 │ 40 40 40 40 @@@@@@@@EX07@@@@

00000320 02 54 58 54 │ 40 00 00 22 │ 40 40 00 04 │ 40 40 00 01 .TXT@.."@@..@@..

00000330 58 60 F0 6A │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 X`ðj@@@@@@@@@@@@

00000340 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

00000350 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

00000360 40 40 40 40 │ 40 40 40 40 │ 45 58 30 37 │ 40 40 40 40 @@@@@@@@EX07@@@@

00000370 02 54 58 54 │ 40 00 00 26 │ 40 40 00 04 │ 40 40 00 01 .TXT@..&@@..@@..

00000380 89 50 60 00 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 .P`.@@@@@@@@@@@@

00000390 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

000003A0 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

000003B0 40 40 40 40 │ 40 40 40 40 │ 45 58 30 37 │ 40 40 40 40 @@@@@@@@EX07@@@@

000003C0 02 54 58 54 │ 40 00 00 2A │ 40 40 00 04 │ 40 40 00 01 .TXT@..\*@@..@@..

000003D0 54 50 F0 72 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 TPðr@@@@@@@@@@@@

000003E0 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

000003F0 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

00000400 40 40 40 40 │ 40 40 40 40 │ 45 58 30 37 │ 40 40 40 40 @@@@@@@@EX07@@@@

00000410 02 54 58 54 │ 40 00 00 2E │ 40 40 00 04 │ 40 40 00 01 .TXT@...@@..@@..

00000420 50 50 F0 5A │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 PPðZ@@@@@@@@@@@@

00000430 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

00000440 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

00000450 40 40 40 40 │ 40 40 40 40 │ 45 58 30 37 │ 40 40 40 40 @@@@@@@@EX07@@@@

00000460 02 54 58 54 │ 40 00 00 32 │ 40 40 00 04 │ 40 40 00 01 .TXT@..2@@..@@..

00000470 58 60 F0 6E │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 X`ðn@@@@@@@@@@@@

00000480 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

00000490 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

000004A0 40 40 40 40 │ 40 40 40 40 │ 45 58 30 37 │ 40 40 40 40 @@@@@@@@EX07@@@@

000004B0 02 54 58 54 │ 40 00 00 36 │ 40 40 00 04 │ 40 40 00 01 .TXT@..6@@..@@..

000004C0 50 60 F0 56 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 P`ðV@@@@@@@@@@@@

000004D0 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

000004E0 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

000004F0 40 40 40 40 │ 40 40 40 40 │ 45 58 30 37 │ 40 40 40 40 @@@@@@@@EX07@@@@

00000500 02 54 58 54 │ 40 00 00 3A │ 40 40 00 02 │ 40 40 00 01 .TXT@..:@@..@@..

00000510 07 FE 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 .þ@@@@@@@@@@@@@@

00000520 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

00000530 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

00000540 40 40 40 40 │ 40 40 40 40 │ 45 58 30 37 │ 40 40 40 40 @@@@@@@@EX07@@@@

00000550 02 54 58 54 │ 40 00 00 3C │ 40 40 00 04 │ 40 40 00 01 .TXT@..<@@..@@..

00000560 00 00 00 03 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 ....@@@@@@@@@@@@

00000570 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

00000580 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

00000590 40 40 40 40 │ 40 40 40 40 │ 45 58 30 37 │ 40 40 40 40 @@@@@@@@EX07@@@@

000005A0 02 54 58 54 │ 40 00 00 40 │ 40 40 00 04 │ 40 40 00 01 .TXT@..@@@..@@..

000005B0 00 00 00 02 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 ....@@@@@@@@@@@@

000005C0 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

000005D0 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

000005E0 40 40 40 40 │ 40 40 40 40 │ 45 58 30 37 │ 40 40 40 40 @@@@@@@@EX07@@@@

000005F0 02 54 58 54 │ 40 00 00 44 │ 40 40 00 04 │ 40 40 00 01 .TXT@..D@@..@@..

00000600 80 00 00 00 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 ....@@@@@@@@@@@@

00000610 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

00000620 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

00000630 40 40 40 40 │ 40 40 40 40 │ 45 58 30 37 │ 40 40 40 40 @@@@@@@@EX07@@@@

00000640 02 54 58 54 │ 40 00 00 48 │ 40 40 00 04 │ 40 40 00 01 .TXT@..H@@..@@..

00000650 00 00 00 03 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 ....@@@@@@@@@@@@

00000660 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

00000670 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

00000680 40 40 40 40 │ 40 40 40 40 │ 45 58 30 37 │ 40 40 40 40 @@@@@@@@EX07@@@@

00000690 02 54 58 54 │ 40 00 00 4C │ 40 40 00 04 │ 40 40 00 01 .TXT@..L@@..@@..

000006A0 00 00 00 03 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 ....@@@@@@@@@@@@

000006B0 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

000006C0 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

000006D0 40 40 40 40 │ 40 40 40 40 │ 45 58 30 37 │ 40 40 40 40 @@@@@@@@EX07@@@@

000006E0 02 54 58 54 │ 40 00 00 50 │ 40 40 00 04 │ 40 40 00 01 .TXT@..P@@..@@..

000006F0 A0 00 00 00 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40  ...@@@@@@@@@@@@

00000700 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

00000710 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

00000720 40 40 40 40 │ 40 40 40 40 │ 45 58 30 37 │ 40 40 40 40 @@@@@@@@EX07@@@@

00000730 02 54 58 54 │ 40 00 00 54 │ 40 40 00 04 │ 40 40 00 01 .TXT@..T@@..@@..

00000740 00 00 00 10 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 ....@@@@@@@@@@@@

00000750 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

00000760 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

00000770 40 40 40 40 │ 40 40 40 40 │ 45 58 30 37 │ 40 40 40 40 @@@@@@@@EX07@@@@

00000780 02 54 58 54 │ 40 00 00 58 │ 40 40 00 04 │ 40 40 00 01 .TXT@..X@@..@@..

00000790 00 00 00 01 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 ....@@@@@@@@@@@@

000007A0 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

000007B0 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

000007C0 40 40 40 40 │ 40 40 40 40 │ 45 58 30 37 │ 40 40 40 40 @@@@@@@@EX07@@@@

000007D0 02 54 58 54 │ 40 00 00 5C │ 40 40 00 04 │ 40 40 00 01 .TXT@..\@@..@@..

000007E0 00 00 00 00 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 ....@@@@@@@@@@@@

000007F0 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

00000800 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

00000810 40 40 40 40 │ 40 40 40 40 │ 45 58 30 37 │ 40 40 40 40 @@@@@@@@EX07@@@@

00000820 02 54 58 54 │ 40 00 00 60 │ 40 40 00 04 │ 40 40 00 01 .TXT@..`@@..@@..

00000830 00 00 00 10 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 ....@@@@@@@@@@@@

00000840 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

00000850 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

00000860 40 40 40 40 │ 40 40 40 40 │ 45 58 30 37 │ 40 40 40 40 @@@@@@@@EX07@@@@

00000870 02 54 58 54 │ 40 00 00 64 │ 40 40 00 04 │ 40 40 00 01 .TXT@..d@@..@@..

00000880 00 00 00 01 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 ....@@@@@@@@@@@@

00000890 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

000008A0 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

000008B0 40 40 40 40 │ 40 40 40 40 │ 45 58 30 37 │ 40 40 40 40 @@@@@@@@EX07@@@@

000008C0 02 54 58 54 │ 40 00 00 68 │ 40 40 00 04 │ 40 40 00 01 .TXT@..h@@..@@..

000008D0 00 00 00 00 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 ....@@@@@@@@@@@@

000008E0 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

000008F0 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

00000900 40 40 40 40 │ 40 40 40 40 │ 45 58 30 37 │ 40 40 40 40 @@@@@@@@EX07@@@@

00000910 02 54 58 54 │ 40 00 00 6C │ 40 40 00 04 │ 40 40 00 01 .TXT@..l@@..@@..

00000920 00 00 00 01 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 ....@@@@@@@@@@@@

00000930 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

00000940 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

00000950 40 40 40 40 │ 40 40 40 40 │ 45 58 30 37 │ 40 40 40 40 @@@@@@@@EX07@@@@

00000960 02 54 58 54 │ 40 00 00 70 │ 40 40 00 04 │ 40 40 00 01 .TXT@..p@@..@@..

00000970 00 00 00 03 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 ....@@@@@@@@@@@@

00000980 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

00000990 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

000009A0 40 40 40 40 │ 40 40 40 40 │ 45 58 30 37 │ 40 40 40 40 @@@@@@@@EX07@@@@

000009B0 02 54 58 54 │ 40 00 00 74 │ 40 40 00 04 │ 40 40 00 01 .TXT@..t@@..@@..

000009C0 60 00 00 00 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 `...@@@@@@@@@@@@

000009D0 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

000009E0 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

000009F0 40 40 40 40 │ 40 40 40 40 │ 45 58 30 37 │ 40 40 40 40 @@@@@@@@EX07@@@@

00000A00 02 45 4E 44 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 .END@@@@@@@@@@@@

00000A10 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

00000A20 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

00000A30 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 │ 40 40 40 40 @@@@@@@@@@@@@@@@

00000A40 40 40 40 40 │ 40 40 40 40 │ 45 58 30 37 │ 40 40 40 40 @@@@@@@@EX07@@@@

На выходе пошаговое выполнение инструкций программы с отображением текущей команды, значением регистров и памяти программы:

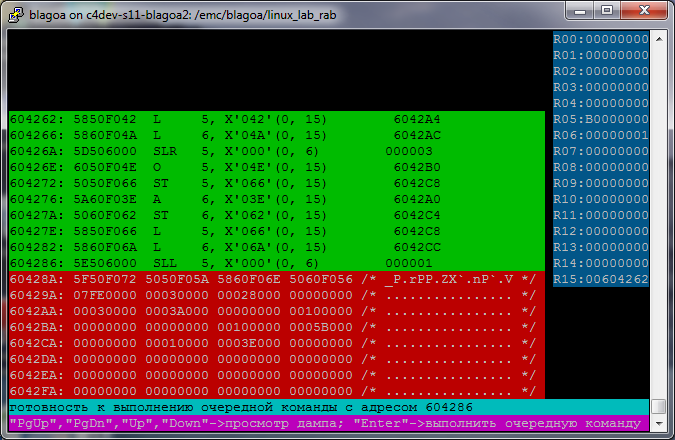


Рис. 1 Вывод загрузчика

# Анализ поставленной задачи

# Входные ограничения

*По сравнению с примером проекта в данном модуле не было внесено изменений во входные ограничения.*

# Модификация базы данных исходного макета

*Добавление в таблицы новых функций:*

struct TMOP /\*структ.стр.табл.маш.опер\*/

{

unsigned char MNCOP [5]; /\*мнемокод операции \*/

unsigned char CODOP ; /\*машинный код операции \*/

unsigned char DLOP ; /\*длина операции в байтах \*/

int (\*BXPROG)() ; /\*указатель на подпр.обраб\*/

} T\_MOP [NOP] = /\*об'явление табл.маш.опер\*/

{

{{'B','A','L','R',' '} , '\x05' , 2 , FRR} , /\*инициализация \*/

{{'B','C','R',' ',' '} , '\x07' , 2 , FRR} , /\*строк \*/

{{'S','T',' ',' ',' '} , '\x50' , 4 , FRX} , /\*таблицы \*/

{{'L',' ',' ',' ',' '} , '\x58' , 4 , FRX} , /\*машинных \*/

{{'A',' ',' ',' ',' '} , '\x5A' , 4 , FRX} , /\*операций \*/

{{'S',' ',' ',' ',' '} , '\x5B' , 4 , FRX} , /\* \*/

**{{'S','L','R',' ',' '} , '\x88' , 4 , FRX} , /\* \*/**

**{{'S','L','L',' ',' '} , '\x89' , 4 , FRX} , /\* \*/**

**{{'N',' ',' ',' ',' '} , '\x54' , 4 , FRX} , /\* \*/**

**{{'O',' ',' ',' ',' '} , '\x56' , 4 , FRX} /\* \*/**

};

# Модификация алгоритма исходного макета

*Реализация непосредственно новых функций:*

**int P\_SLR() /\* п р о г р а м м а реализации семантики \*/**

**{ /\*команды A \*/**

**int sm; /\*рабочая переменная \*/**

**ADDR = VR[B] + VR[X] + D; /\*вычисление абс.адреса и \*/**

**VR[R1] = VR[R1] >> ADDR; /\*сложение с 1-м операндом\*/**

**return 0; /\*успешное заверш. прогр. \*/**

**}**

**int P\_SLL() /\* п р о г р а м м а реализации семантики \*/**

**{ /\*команды A \*/**

**int sm; /\*рабочая переменная \*/**

**ADDR = VR[B] + VR[X] + D; /\*вычисление абс.адреса и \*/**

**VR[R1] = VR[R1] << ADDR & 0xFFFFFFFF; /\*сложение с 1-м операндом\*/**

**return 0; /\*успешное заверш. прогр. \*/**

**}**

**int P\_N(void)**

**{**

**int sm; /\*рабочая переменная \*/**

**ADDR = VR[B] + VR[X] + D; /\*вычисление рабочего \*/**

**sm = ( int ) ( ADDR - I ); /\*адреса и смещения \*/**

**ARG = OBLZ[BAS\_IND + CUR\_IND + sm] \* 0x1000000L+/\*формирование содержимого\*/**

**OBLZ[BAS\_IND + CUR\_IND + sm + 1] \* 0x10000L +/\*второго операнда в сог- \*/**

**OBLZ[BAS\_IND + CUR\_IND + sm + 2] \* 0x100 + /\*лашениях ЕС ЭВМ \*/**

**OBLZ[BAS\_IND + CUR\_IND + sm + 3]; VR[R1] = VR[R1] & ARG;**

**return 0;**

**}**

**int P\_O(void)**

**{**

**int sm; /\*рабочая переменная \*/**

**ADDR = VR[B] + VR[X] + D; /\*вычисление рабочего \*/**

**sm = ( int ) ( ADDR - I ); /\*адреса и смещения \*/**

**ARG = OBLZ[BAS\_IND + CUR\_IND + sm] \* 0x1000000L+/\*формирование содержимого\*/**

**OBLZ[BAS\_IND + CUR\_IND + sm + 1] \* 0x10000L +/\*второго операнда в сог- \*/**

**OBLZ[BAS\_IND + CUR\_IND + sm + 2] \* 0x100 + /\*лашениях ЕС ЭВМ \*/**

**OBLZ[BAS\_IND + CUR\_IND + sm + 3]; VR[R1] = VR[R1] | ARG;**

**return 0;**

**}**

# Заключение

*Плюсы данной реализации:*

1. *Простота, вносится минимум изменений*

*Минусы данной реализации: необходимо в алгоритме ограничиться данными функциями*

1. *При реализации функций на ЯВУ необходимо ограничиваться лишь указанными функциями.*

# Разработка элементов учебной системы программирования

по дисциплине «Системы программирования».

# Компилятор с языка высокого уровня с использованием Flex и Bison.

Вариант №7

Выполнили студенты гр. 5084/13 «подпись» Благодиров А.В.

«подпись» Черкалова В.В.

«подпись» Афанасьева В.Ю.

Руководитель доцент «подпись» Расторгуев В.Я.

«\_\_\_»\_\_\_\_\_\_\_\_2012

# Введение

Данная курсовая работа имеет своей целью получение практических навыков построения компилятора с языка высокого уровня (ЯВУ), являющегося одним из элементов системы программирования, образующих в совокупности следующий технологический конвейер:



При этом предполагается то, что данная система программирования работает на технологической ЭВМ (IBM PC) и является по существу кросс-системой для объектной ЭВМ (ЕС ЭВМ). В этой системе:

* в качестве языка высокого уровня (ЯВУ) выбран язык, образованный из подмножества языковых конструкций ПЛ1, а исходная программа готовится в виде текстового файла технологической ЭВМ с расширением \*.pli;
* язык АССЕМБЛЕРА сформирован из языковых конструкций АССЕМБЛЕРА ЕС ЭВМ, а ассемблеровский эквивалент исходной программы формируется в виде текстового файла технологической ЭВМ с расширением \*.ass;
* объектный эквивалент исходной программы готовится в формате объектных файлов операционной системы ОС ЕС ЭВМ и хранится в виде двоичного файла технологической ЭВМ с расширением \*.tex;
* загрузочный эквивалент исходной программы представляет собой машинный код ЕС ЭВМ, запоминаемый в области ОЗУ технологической ЭВМ, являющейся зоной загрузки для эмулятора объектной ЭВМ.

# Постановка задачи

Необходимо выполнить доработку элементов макета учебной системы программирования до уровня, позволяющего обрабатывать “новые” для макета конструкции языка высокого уровня, примененные в соответствующем варианте:



Где на входе имеется текст программы на ЯВУ PL/I:

ex1:proc options(main);

dcl A bit (3) init (10B);

dcl B bit (3) init (101B);

dcl C bit (16);

C = substr(( B !! A ),2,3);

end ex1;

На выходе строится эквивалент программы на ассемблере архитектуры IBM 370:

ex1 START 0 Programm start

BALR RBASE,0 Base initialization

USING \*,RBASE Base declaration

L RRAB,@VA

L RSH, @LB

SLR RRAB,0(0,6)

O RRAB,@VB

ST RRAB,@VMID

A RSH, @LA

ST RSH, @LMID

L RRAB,@VMID

L RSH, SL

SLL RRAB,0(0,6)

N RRAB,MSK

S RRAB,@VC

L RSH, NL

ST RSH, @LC

BCR 15,RVIX Return from programm

RBASE EQU 15

RVIX EQU 14

RRAB EQU 5

RSH EQU 6

A DC F'3'

@LA DC F'2'

@VA DC BL4'10'

B DC F'3'

@LB DC F'3'

@VB DC BL4'101'

C DC F'16'

@LC DC F'0'

@VC DC BL4''

MID DC F'16'

@LMID DC F'0'

@VMID DC BL4''

SL DC F'1' Variable declaration with initialization

NL DC F'3' Variable declaration with initialization

MSK DC BL4'111'

END Programm end

# Анализ поставленной задачи

*Для реализации поставленной задачи необходимо определить структуру, описывающую битовые строки. То есть такую конструкцию на ассемблере, которая бы соответсвовала коду на PLI:*

*DCL A BIT ( 3 ) INIT ( '10'B );*

*Помимо этого не следует забывать про необходимость созхранить уже существующую функциональность.*

*Поэтому нами была выбрана следующая реализация:*

*A DC F'3'*

*@LA DC F'2'*

*@VA DC BL4'10'*

*Где метка А – заданнная в определении максимальная длина строки.*

*метка @LA – длина строки, записанной в значение*

*метка @VA – значение строки*

*Такая структура позволяет нам при минимальных модификациях (добавление типа BR4) реализовать данную задачу, не теряя существующей функциональности, поскольку обработка имеющихся типов остается неизменной.*

*Также на ЯВУ нами были добавлены команды SUBSTR и !! (конкатенация).*

*Команда SUBSTR принимает на вход три параметра – строку, номер символа, с которого наинается отсчет, и колличество символов (длину подстроки).*

*Будем воспринимать это следующим образом:*

*SUBSTR(‘0111000’,2,3) = ‘111’*

*Команда конкатенации (!!) принимает два аргумента и “склеивает”их. То есть:*

*‘111’ !! ‘10’ = ‘11110’*

*При реализации ланных операций на ассемблере мы импользовали следующие алгоритмы:*

1. *Подстрока:*

* *Считываем номер элемента N, с которого следует взять подстроку*
* *Сдвигаем строку влево на N-1*
* *При компиляции с ЯВУ создаем маску вида “111000000....”, где колличество 1 – длина подстроки*
* *При помощи операции И “отрезаем” конец старой строки*
* *Записываем результат в поле значения*
* *Записываем новую длину в поле длины*

1. *Конкатенация:*

* *Сдвигаем второй элемент вправо на длину первого*
* *При помощи операции ИЛИ получаем результат*
* *Записываем результат в поле значения*
* *Складываем длину первого и второго аргументов*
* *Записываем новую длину в поле длины*

# Входные ограничения

*При постановке задачи были введены следующие ограничения:*

*1) Максимальная длина битовой строки, обрабатываемая с помощью данного алгоритма – 16.*

*2) Реализованный алгоритм не проверяет длину результата конкатенации двух строк, то есть он не обрабатывает каким либо специфическим образом ситуацию, когда строки А и B заданы таким образом, что их конкатенация превышает максимальную длину битовой строки, определенную в пункте 1.*

*3) В реализованном алгоритме не проверяется попытка взять подстроку выходящую за пределы имебщейся строки (например подстроку длиной 3 из строки длины 1).*

# Модификация базы данных исходного макета

*Добавление распознавания новых функций:*

PL\_I.l

**"," {**

**return ',';**

**}**

**!! {**

**yylval=strdup(yytext); return ZNK;**

**}**

[a-zA-Z][a-zA-Z0-9]\* {

if (!memcmp(yytext,"proc", 4)) return PROC;

if (!memcmp(yytext,"options", 7)) return OPTIONS;

if (!memcmp(yytext,"main", 4)) return MAIN;

if (!memcmp(yytext,"end", 3)) return END;

if (!memcmp(yytext,"dcl", 3)) return DCL;

if (!memcmp(yytext,"bin", 3)) return BIN;

**if (!memcmp(yytext,"bit", 3)) return BIT;**

**if (!memcmp(yytext,"substr", 6)) return SUBSTR;**

if (!memcmp(yytext,"fixed", 3)) return FIXED;

if (!memcmp(yytext,"init", 3)) return INIT;

yylval=strdup(yytext); return IDENT;

}

# Модификация алгоритма исходного макета

*Реализация обработки новых функций*

PL\_I.y

**int IsBit(char\* pName, int length){**

**char etalon[11];**

**etalon[0] = '@';**

**etalon[1] = 'L';**

**memcpy(&etalon[2], pName, length);**

**return IsDclName( etalon, length + 2);**

**}**

**void addVar(char \*ipe, char \*val, int type){**

**char TEMP [20];**

**int j = 0;**

**memset(&s1[0], ' ', 80);**

**memcpy(&s1[0], ipe, strlen(ipe));**

**memcpy(&s1[9], "DC", 2);**

**if (type == 1)**

**{**

**s1[15]='F';**

**s1[16]='\'';**

**memcpy(&s1[17], val, strlen(val));**

**s1[17+strlen(val)]='\'';**

**memcpy(&s1[30], "Variable declaration with initialization", 40);**

**}**

**else**

**{**

**memcpy(&s1[15], "BL4\'", 4);**

**memcpy(&s1[19], val, strlen(val) );**

**s1[19+strlen(val)]='\'';**

**}**

**memcpy(&DclPart[pDclPart][0], &s1[0], 80);**

**pDclPart++;**

**}**

%debug

%verbose

%token IDENT PROC OPTIONS MAIN END DCL BIN FIXED NUM INIT **BIT SUBSTR**

%left ZNK

%start pro

%%

pro: opr tel oen { pro(); }

;

tel: dec imp

;

dec: odc

| dec odc

;

odc: odi

| odr

;

odi: DCL ipe BIN FIXED '(' rzr ')' INIT '(' lit ')' ';' { odi($2, $6, $10); }

**| DCL ipe BIT '(' rzr ')' INIT '(' lit ')' ';' { odi\_t($2, $6, $10); }**

;

odr: DCL ipe BIN FIXED '(' rzr ')' ';' { odr($2, $6); }

**|DCL ipe BIT '(' rzr ')' ';' { odr\_t($2, $6); }**

;

ipe: IDENT {$$=$1;}

;

rzr: NUM {$$=$1;}

;

lit: NUM {$$=$1;}

;

opr: IDENT ':' PROC OPTIONS '(' MAIN ')' ';' { opr($1); }

;

oen: END IDENT ';' { if ( oen($2) ) YYABORT; }

;

imp: opa

| imp opa

;

opa: ipe '=' avi ';' { if ( opa($1) ) YYABORT; }

**|ipe '=' SUBSTR '(''(' avi ')' ', ' lit ', ' lit ')' ';' { if ( opa\_t($1, $9, $11) ) YYABORT; }**

;

avi: lit { avi\_lit($1); }

| ipe { if ( avi\_ipe($1) ) YYABORT;}

| avi ZNK lit { avi\_avi\_znk\_lit($2, $3); }

| avi ZNK ipe { if ( avi\_avi\_znk\_ipe($2, $3, $1) ) YYABORT; }

;

%%

**void odi\_t(char \*ipe, char \*rzr, char \*lit){**

**char RAB [20];**

**char TEMP [20];**

**int j = 0;**

**memset(&s1[0], ' ', 80);**

**memcpy(&s1[0], ipe, strlen(ipe));**

**memcpy(&s1[9], "DC", 2);**

**s1[15]='F';**

**s1[16]='\'';**

**memcpy(&s1[17], rzr, strlen(rzr));**

**s1[17+strlen(rzr)]='\'';**

**memcpy(&DclPart[pDclPart][0], &s1[0], 80);**

**pDclPart++;**

**j = 0;**

**while ( lit[j] != 'B' )**

**{**

**TEMP[j] = lit[j];**

**j++;**

**}**

**TEMP[j] = 0;**

**memset(&s1[0], ' ', 80);**

**memcpy(&s1[0], "@L", 2);**

**memcpy(&s1[2], ipe, strlen(ipe));**

**memcpy(&s1[9], "DC", 2);**

**s1[15]='F';**

**s1[16]='\'';**

**gcvt(j, 10, &RAB[0]);**

**memcpy(&s1[17], RAB , strlen(RAB));**

**s1[17+strlen(RAB)]='\'';**

**memcpy(&DclPart[pDclPart][0], &s1[0], 80);**

**pDclPart++;**

**memset(&s1[0], ' ', 80);**

**memcpy(&s1[0], "@V", 2);**

**memcpy(&s1[2], ipe, strlen(ipe));**

**memcpy(&s1[9], "DC", 2);**

**memcpy(&s1[15], "BL4\'", 4);**

**memcpy(&s1[19], TEMP, strlen(TEMP));**

**s1[19+strlen(TEMP)]='\'';**

**memcpy(&DclPart[pDclPart][0], &s1[0], 80);**

**pDclPart++;**

**}**

**void odr\_t(char \*ipe, char \*rzr){**

**odi\_t(ipe, rzr, "0B");**

**}**

**int opa\_t(char \*ipe, char \*from, char \*len){**

**char RAB [20];**

**int temp1;**

**int temp2;**

**if (IsDclName(ipe, strlen(ipe))){**

**strcpy(&ErrorMessage[0], " invalid identificator ");**

**strcat(&ErrorMessage[0], ipe);**

**strcat(&ErrorMessage[0], " ");**

**strcat(&ErrorMessage[0], "in left part of opa\n");**

**yyerror(&ErrorMessage[0]);**

**return 1;**

**}**

**temp1 = atoi(from);**

**temp2 = atoi(len);**

**gcvt(temp1 - 1, 10, &RAB[0]);**

**addVar( "SL", RAB, 1);**

**gcvt(temp2, 10, &RAB[0]);**

**addVar( "NL", RAB, 1);**

**for(temp1 = 0; temp1 < temp2; ++temp1)**

**{**

**RAB[temp1] = '1';**

**}**

**RAB[temp1] = 0;**

**addVar( "MSK", RAB, 2);**

**memset(&s1[0], ' ', 80);**

**memcpy(&s1[9], "L", 1 );**

**memcpy(&s1[15], "RRAB," , 5);**

**memcpy(&s1[20], "@VMID" , 5);**

**memcpy(&ImpPart[pImpPart][0], &s1[0], 80);**

**pImpPart++;**

**memset(&s1[0], ' ', 80);**

**memcpy(&s1[9], "L", 1 );**

**memcpy(&s1[15], "RSH," , 4);**

**memcpy(&s1[20], "SL" , 2);**

**memcpy(&ImpPart[pImpPart][0], &s1[0], 80);**

**pImpPart++;**

**memset(&s1[0], ' ', 80);**

**memcpy(&s1[9], "SLL", 3 );**

**memcpy(&s1[15], "RRAB," , 5);**

**memcpy(&s1[20], "0(0,6)" , 6);**

**memcpy(&ImpPart[pImpPart][0], &s1[0], 80);**

**pImpPart++;**

**memset(&s1[0], ' ', 80);**

**memcpy(&s1[9], "N", 1 );**

**memcpy(&s1[15], "RRAB," , 5);**

**memcpy(&s1[20], "MSK" , 3);**

**memcpy(&ImpPart[pImpPart][0], &s1[0], 80);**

**pImpPart++;**

**memset(&s1[0], ' ', 80);**

**memcpy(&s1[9], "ST", 1 );**

**memcpy(&s1[15], "RRAB," , 5);**

**memcpy(&s1[20], "@V" , 2);**

**memcpy(&s1[22], ipe, strlen(ipe));**

**memcpy(&ImpPart[pImpPart][0], &s1[0], 80);**

**pImpPart++;**

**memset(&s1[0], ' ', 80);**

**memcpy(&s1[9], "L", 1 );**

**memcpy(&s1[15], "RSH," , 4);**

**memcpy(&s1[20], "NL" , 2);**

**memcpy(&ImpPart[pImpPart][0], &s1[0], 80);**

**pImpPart++;**

**memset(&s1[0], ' ', 80);**

**memcpy(&s1[9], "ST", 2 );**

**memcpy(&s1[15], "RSH," , 4);**

**memcpy(&s1[20], "@L" , 2);**

**memcpy(&s1[22], ipe, strlen(ipe));**

**memcpy(&ImpPart[pImpPart][0], &s1[0], 80);**

**pImpPart++;**

**return 0;**

**}**

int avi\_avi\_znk\_ipe(char \*znk, char \*ipe, char \*ipe2 ) {

if (IsDclName(ipe, strlen(ipe))){

strcpy(&ErrorMessage[0], " invalid identificator ");

strcat(&ErrorMessage[0], ipe);

strcat(&ErrorMessage[0], " ");

strcat(&ErrorMessage[0], "in avi of opa\n");

yyerror(&ErrorMessage[0]);

return 1;

}

**if(!memcmp(znk, "!!", 2)) {**

**odr\_t( "MID", "16");**

**memset(&s1[0], ' ', 80);**

**memcpy(&s1[9], "L", 1 );**

**memcpy(&s1[15], "RRAB," , 5);**

**memcpy(&s1[20], "@V" , 2);**

**memcpy(&s1[22], ipe, strlen(ipe));**

**memcpy(&ImpPart[pImpPart][0], &s1[0], 80);**

**pImpPart++;**

**memset(&s1[0], ' ', 80);**

**memcpy(&s1[9], "L", 1 );**

**memcpy(&s1[15], "RSH," , 4);**

**memcpy(&s1[20], "@L" , 2);**

**memcpy(&s1[22], ipe2, strlen(ipe2));**

**memcpy(&ImpPart[pImpPart][0], &s1[0], 80);**

**pImpPart++;**

**memset(&s1[0], ' ', 80);**

**memcpy(&s1[9], "SLR", 3 );**

**memcpy(&s1[15], "RRAB," , 5);**

**memcpy(&s1[20], "0(0,6)" , 6);**

**memcpy(&ImpPart[pImpPart][0], &s1[0], 80);**

**pImpPart++;**

**memset(&s1[0], ' ', 80);**

**memcpy(&s1[9], "O", 1 );**

**memcpy(&s1[15], "RRAB," , 5);**

**memcpy(&s1[20], "@V" , 2);**

**memcpy(&s1[22], ipe2, strlen(ipe2));**

**memcpy(&ImpPart[pImpPart][0], &s1[0], 80);**

**pImpPart++;**

**memset(&s1[0], ' ', 80);**

**memcpy(&s1[9], "ST", 2 );**

**memcpy(&s1[15], "RRAB," , 5);**

**memcpy(&s1[20], "@VMID" , 5);**

**memcpy(&ImpPart[pImpPart][0], &s1[0], 80);**

**pImpPart++;**

**memset(&s1[0], ' ', 80);**

**memcpy(&s1[9], "A", 1 );**

**memcpy(&s1[15], "RSH," , 4);**

**memcpy(&s1[20], "@L" , 2);**

**memcpy(&s1[22], ipe, strlen(ipe));**

**memcpy(&ImpPart[pImpPart][0], &s1[0], 80);**

**pImpPart++;**

**memset(&s1[0], ' ', 80);**

**memcpy(&s1[9], "ST", 2 );**

**memcpy(&s1[15], "RSH," , 4);**

**memcpy(&s1[20], "@LMID" , 5);**

**memcpy(&ImpPart[pImpPart][0], &s1[0], 80);**

**pImpPart++;**

**return 0;**

**}**

memset(&s1[0], ' ', 80);

if(!memcmp(znk, "+", 1)) {

memcpy(&s1[9], "A", 1);

memcpy(&s1[30], "Variable\'s value adding", 23);

}

if(!memcmp(znk, "-", 1)) {

memcpy(&s1[9], "S", 1);

memcpy(&s1[30], "Variable\'s value substracting", 29);

}

memcpy(&s1[15], "RRAB, ", 5);

memcpy(&s1[20], ipe, strlen(ipe));

memcpy(&ImpPart[pImpPart][0], &s1[0], 80);

pImpPart++;

return 0;

}

int main() {

pAssProg=0;

memset(&DclPart[0][0], ' ', 80);

memcpy(&DclPart[0][0], "RBASE EQU 15", 17);

memset(&DclPart[1][0], ' ', 80);

memcpy(&DclPart[1][0], "RVIX EQU 14", 17);

memset(&DclPart[2][0], ' ', 80);

memcpy(&DclPart[2][0], "RRAB EQU 5", 16);

**memset(&DclPart[3][0], ' ', 80);**

**memcpy(&DclPart[3][0], "RSH EQU 6", 16);**

**pDclPart=4;**

pImpPart=0;

# Заключение

*В процессе выполнения работы с проблемами не столкнулись. Для реализации данного задания данная среда более интуитивно понятна и проста, чем первый вариант реализации.*

*Плюсы данной реализации:*

1. *Проще в реализации и понимании по сравнению с первым вариантом реализации*
2. *Сохранение функциональных способностей в старой системе и добавление новых.*
3. *Полноценная работа программы с битовыми строками длиной не более 16 бит.*
4. *Независимая реализация операций подстроки и конкатенации – при изменении входных данных потребуется меньшее колличество изменений.*

*Минусы данной реализации:*

1. *Алгоритм не рассматривает ситуации выхода за пределы доспустимых длин строк (при конкатенации строк не проверяется длина результирующей строки, как и при взятии подстроки не проверяется что строка-аргумент имеет длину не меньше заданного колличесва вырезаемых символов.*