Минобрнауки России

Санкт-Петербургский государственный политехнический университет

Институт информационных технологий и управления

**Кафедра «Информационные и управляющие системы»**

**КУРСОВОЙ ПРОЕКТ**

**Построение компилятора с языка высокого уровня.**

по дисциплине «Системы программирования»

Выполнили

студенты гр.5084/12 А.А.Лукашин

К.С.Шубин

Руководитель

доцент В.Я.Расторгуев

«\_\_» \_\_\_\_\_\_\_\_\_\_ 2013 г.

Санкт-Петербург

2013

Оглавление

[Задание 3](#_Toc356379044)

[План работы 3](#_Toc356379045)

[Эквивалент на Ассемблере 4](#_Toc356379046)

[Модификация грамматики языка 6](#_Toc356379047)

[Матрица смежности 10](#_Toc356379048)

[Таблица синтаксических правил 11](#_Toc356379049)

[Таблица входов в правила 13](#_Toc356379050)

[Модификация функций компилятора 14](#_Toc356379051)

[Выводы 28](#_Toc356379052)

# Задание

Усовершенствовать компилятор с языка высокого уровня (ЯВУ) для получения эквивалента исходного текста на ассемблере.

*Вариант №7:*

Код на языке PL1

EX07: 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 EX07;

# План работы

Необходимо доработать компилятор с ЯВУ, дополнив его новой функциональность. В новую функциональность входят:

1. новый тип данных (bit);
2. операция конкатенации (!!);
3. операция взятия подстроки битовой строки (substr).

Для решения этой задачи необходимо:

1. модифицировать грамматику языка;
2. изменить и дополнить матрицу смежности;
3. расширить таблицу синтаксических правил;
4. расширить таблицу входов в правила;
5. добавить и изменить необходимые функции компилятора.

# Эквивалент на Ассемблере

После компиляции с языка PL1 на Ассемблер должен получиться следующий код:

EX07 START 0

BALR RBASE,0

USING \*,RBASE

LH 3,B

LH 4,A

SRL 4,3

OR 3,4

SLL 3,2

LH 4,TMP

NR 3,4

SRL 3,16

STH 3,C

BCR 15,RVIX

A DC F'3'

DC F'2'

DC BL2'10'

B DC F'3'

DC F'3'

DC BL2'101'

C DC F'16'

DC F'0'

DS BL2

TMP DC F'16'

DC F'3'

DC BL2'111'

RBASE EQU 5

RVIX EQU 14

END EX07

Первые 3 строки программы – пролог.

Следующие 4 строки относятся к операции конкатенации. Переменная B записывается в 3 регистр, переменная A – в 4 регистр. Далее происходит сдвиг содержимого регистра 4 на 3 разряда вправо (SRL). Затем регистровая операция ИЛИ (OR) между 3 и 4 регистрами, результат записывается в 3 регистр. Для хранения переменных используется 16-разрядная модель данных (полуслово). Поэтому в качестве команды загрузки переменной в регистр используется LH.

Следующие 3 строки отвечают за операцию взятия подстроки. Происходит сдвиг влево содержимого 3 регистра на 2 разряда (второй параметр операции substr). Затем в 4 регистр загружается маска tmp. Первые 3 разряда маски (третий параметр substr) – единицы, остальное – нули. Затем происходит регистровая операции И между 3 и 4 регистрами (NR).

В следующих 2 строках происходит запись содержимого из регистра в память. Т.к. операция STH осуществляет выгрузку правого полуслова регистра в память, сначала содержимого 3 регистра сдвигается на 16 разрядов вправо.

Далее идёт объявление переменных. Оно состоит из 3 частей. Первая – резервирование памяти в соответствии с заявленным требованием в разделе DCL программы на ЯВУ (bit (3)). Вторая – реально требуемая память. Третья – BL2 – значение переменной в логическом виде (выделяется 16 разрядов).

Последние 3 строки программы – эпилог.

# Модификация грамматики языка

Ниже представлена грамматика в модифицированном виде. Серым выделены добавленные правила:

1. <PRO> ::= <OPR><TEL><OEN>

2. <OPR> ::= <IPR>:PROC\_OPTIONS(MAIN);

3. <IPR> ::= <IDE>

4. <IDE> ::= <BUK> | <IDE><BUK> | <IDE><CIF>

5. <BUK> ::= A | B | C | D | E | M | P | X

6. <CIF> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

7. <TEL> ::= <ODC> | <TEL><ODC> | <TEL><OPA>

| <TEL><OPL>

8. <ODC> ::= DCL\_<IPE>\_BIN\_FIXED(<RZR>); |

DCL\_<IPE>\_BIN\_FIXED(<RZR>)INIT(<LIT>);

| DCL\_<IPE>\_BIT(<RZR>); | DCL\_<IPE>\_BIT(<RZR>)INIT(<LIT>);

9. <IPE> ::= <IDE>

10. <RZR> ::= <CIF> | <RZR><CIF>

11. <LIT> ::= <MAN>B

12. <MAN> ::= 1 | <MAN>0 | <MAN>1

13. <OPA> ::= <IPE>=<AVI>;

13.1 <OPL> ::= <IPE>=<LVI>;

14. <AVI> ::= <LIT> | <IPE> | <AVI><ZNK><LIT> |

<AVI><ZNK><IPE>

14.1 <LVI> ::= <IPE> | <LVI><ZNKL><IPE> | <SSTR> | (<LVI>)

15. <ZNK> ::= + | -

15.1 <ZNKL> ::= !!

15.2 <SSTR> ::= SUBSTR(<LVI>, <RZR>, <RZR>)

16. <OEN> ::= END\_<IPR>

Здесь использованы следующие метасимволы и символы:

* "<" и ">" - левый и правый ограничители нетерминального символа,
* "::=" - метасимвол со смыслом "равно по определению",
* "|" - метасимвол альтернативного определения "или",
* "\_" - терминальный символ со смыслом "пробел",
* "<PRO>" - нетерминал "программа",
* "<OPR>" - нетерминал "оператор пролога программы",
* "<IPR>" - нетерминал "имя программы",
* "<IDE>" - нетерминал "идентификатор",
* "<BUK>" - нетерминал "буква",
* "<CIF>" - нетерминал "цифра",
* "<TEL>" - нетерминал "тело программы",
* "<ODC>" - нетерминал "оператор declare",
* "<IPE>" - нетерминал "имя переменной",
* "<RZR>" - нетерминал "разрядность",
* "<LIT>" - нетерминал "литерал",
* "<MAN>" - нетерминал "мантисса",
* "<OPA>" - нетерминал "оператор присваивания арифметический",
* "<OPL>" - нетерминал "оператор присваивания логический",
* "<AVI>" - нетерминал "арифметическое выражение",
* "<LVI>" - нетерминал "логическое выражение"**,**
* "<ZNK>" - нетерминал "знак",
* "<ZKL>" - нетерминал "знак логический",
* "<LOP>" - нетерминал "логическая операция"**,**
* "<OEN>" - нетерминал "оператор эпилога программы".
* "<SUS>" - нетерминал "оператор подстрока",

Прежде чем приступать к редактированию матрицы смежности, представим данные правила в виде продукции. Для этого «перевернём» их слева направо:

1. <OPR><TEL><OEN> -> <PRO>
2. <IPR>:PROC\_OPTIONS(MAIN); -> <OPR>
3. <IDE> -> <IPR>
4. <BUK> -> <IDE>
5. <IDE><BUK> -> <IDE>
6. <IDE><CIF> -> <IDE>
7. A -> <BUK>
8. B -> <BUK>
9. C -> <BUK>
10. D -> <BUK>
11. E -> <BUK>
12. M -> <BUK>
13. P -> <BUK>
14. X -> <BUK>
15. 0 -> <CIF>
16. 1 -> <CIF>
17. 2 -> <CIF>
18. 3 -> <CIF>
19. 4 -> <CIF>
20. 5 -> <CIF>
21. 6 -> <CIF>
22. 7 -> <CIF>
23. 8 -> <CIF>
24. 9 -> <CIF>
25. <ODC> -> <TEL>
26. <TEL><ODC> -> <TEL>
27. <TEL><OPA> -> <TEL>
28. <TEL><OPL> -> <TEL>
29. DCL\_<IPE>\_BIN\_FIXED(<RZR>); -> <ODC>
30. DCL\_<IPE>\_BIN\_FIXED(<RZR>)INIT(<LIT>); -> <ODC>
31. DCL\_<IPE>\_BIT(<RZR>); -> <ODC>
32. DCL\_<IPE>\_BIT(<RZR>)INIT(<LIT>); -> <ODC>
33. <IDE> -> <IPE>
34. <CIF> -> <RZR>
35. <RZR><CIF> -> <RZR>
36. <MAN>B -> <LIT>
37. 1 -> <MAN>
38. <MAN>0 -> <MAN>
39. <MAN>1 -> <MAN>
40. <IPE>=<AVI>; -> <OPA>
41. <IPE>=<LVI>; -> <OPL>
42. <LIT> -> <AVI>
43. <IPE> -> <AVI>
44. <AVI><ZNK><LIT> -> <AVI>
45. <AVI><ZNK><IPE> -> <AVI>
46. <IPE> -> <LVI>
47. <LVI><ZNKL><IPE> -> <LVI>
48. <SSTR> -> <LVI>
49. (<LVI>) -> <LVI>
50. + -> <ZNK>
51. - -> <ZNK>
52. !! -> <ZNKL>
53. SUBSTR(<LVI>, <RZR>, <RZR>) -> <SSTR>
54. END\_<IPR> -> <OEN>

Теперь, просматривая каждую из продукций слева-направо, сгруппируем продукции, имеющие общие части в "кусты", в которых роль "ствола" играют общие части продукций, а роль "ветвей" – различающиеся части продукций.

1. <OPR><TEL><OEN> -> <PRO>
2. <IPR>:PROC\_OPTIONS(MAIN); -> <OPR>
3. <IDE> -> <BUK> -> <IDE>

└-> <CIF> -> <IDE>

└-> <IPR>

└-> <IPE>

1. <BUK> -> <IDE>
2. A -> <BUK>
3. B -> <BUK>
4. C -> <BUK>
5. D -> CL\_<IPE>\_BIN\_FIXED(<RZR>)INIT(<LIT>); -> <ODC>

| | └-> ; -> <ODC>

| └->T(<RZR>)INIT(<LIB>); -> <ODC>

| └-> ; -> <ODC>

└ <BUK>

1. E -> ND\_<IPR> -> <OEN>

└-> <BUK>

1. M -> <BUK>
2. P -> <BUK>
3. X -> <BUK>
4. 0 -> <CIF>
5. 1 -> <CIF>

└-> <MAN>

1. 2 -> <CIF>
2. 3 -> <CIF>
3. 4 -> <CIF>
4. 5 -> <CIF>
5. 6 -> <CIF>
6. 7 -> <CIF>
7. 8 -> <CIF>
8. 9 -> <CIF>
9. <ODC> -> <TEL>
10. <TEL> -> <ODC> -> <TEL>

└-> <OPA> -> <TEL>

└-> <OPL> -> <TEL>

1. <CIF> -> <RZR>
2. <RZR><CIF> -> <RZR>
3. <MAN> -> B -> <LIT>

└-> 0 -> <MAN>

└-> 1 -> <MAN>

1. <IPE> -> =<AVI>; -> <OPA>

| └-> <LVI>; -> <OPL>

└-> <AVI>

└-> <LVI>

1. <LIT> -> <AVI>
2. <AVI><ZNK> -> <LIT> -> <AVI>

└-> <IPE> -> <AVI>

1. <LVI><ZNKL><IPE> -> <LVI>
2. <SSTR> -> <LVI>
3. (<LVI>) -> <LVI>
4. + -> <ZNK>
5. - -> <ZNK>
6. !! -> <ZNKL>
7. SUBSTR(<LVI>, <RZR>, <RZR>) -> <SSTR>

Таким образом, в грамматику языка были добавлены нетерминалы OPL (операнд присваивания логический), LVI (логическое выражение), ZKL (знак логический) и SUS (подстрока) и терминалы U, ! и запятая.

# Матрица смежности

Были исправлены некоторые глобальные переменные:

#define NVXOD 60 /\* - табл.входов; \*/

#define NNETRM 20 /\* - списка нетерминалов; \*/

Ниже представим изменённые фрагменты матрицы смежности (заголовки добавленных строк и столбцов выделены серым):

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

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

|\_\_\_\_\_\_\_\_\_\_:\_\_\_:\_\_\_:\_\_\_:\_\_\_:\_\_\_:\_\_\_:\_\_\_:\_\_\_:\_\_\_:\_\_\_:\_\_\_:\_\_\_:\_\_\_:\_\_\_:\_\_\_:\_\_\_:\_\_\_:\_\_\_:\_\_\_|

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

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

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

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

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

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

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

…

Единица в матрице означает, что правило, начинающееся с терминала или нетерминала в строке, завершится нетерминалом в столбце. Например, правило, начинающееся с «S» завершится нетерминалом «SUS».

# Таблица синтаксических правил

Была исправлена глобальная переменная:

#define NSINT 262 /\* - табл.синтакс.правил; \*/

Добавленные и изменённые фрагменты таблицы синтаксических правил приводятся ниже:

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

| NN : посл : пред : дер : альт |

|\_\_\_\_\_\_\_\_\_\_:\_\_\_\_\_\_\_\_\_:\_\_\_\_\_\_\_:\_\_\_\_\_\_\_:\_\_\_\_\_\_|

/\* 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 },

…

/\* вход с символа - IPE \*/

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

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

{/\*. 155 .\*/ 156 , 154 , "AVI" , 218 },

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

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

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

{/\*. 159 .\*/ 160 , 153 , "AVI" , 222 },

{/\*. 160 .\*/ 0 , 159 , "\* " , 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" , 224 },

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

{/\*. 185 .\*/ 0 , 184 , "\* " , 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 , 204 , "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 .\*/ 217 , 215 , "ODC" , 0 },

{/\*. 217 .\*/ 0 , 216 , "\* " , 0 },

{/\*. 218 .\*/ 219 , 154 , "LVI" , 0 },

{/\*. 219 .\*/ 220 , 218 , "; " , 0 },

{/\*. 220 .\*/ 221 , 219 , "OPL" , 0 },

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

{/\*. 222 .\*/ 223 , 153 , "LVI" , 0 },

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

{/\*. 224 .\*/ 225 , 179 , "OPL" , 0 },

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

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

/\* вход с символа - LVI \*/

{/\*. 227 .\*/ 228 , 0 , "LVI" , 0 },

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

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

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

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

/\* вход с символа - SUS \*/

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

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

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

/\* вход с символа – ( \*/

{/\*. 235 .\*/ 236 , 0 , "( " , 0 },

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

{/\*. 237 .\*/ 238 , 236 , ") " , 0 },

{/\*. 238 .\*/ 239 , 237 , "LVI" , 0 },

{/\*. 239 .\*/ 0 , 238 , "\* " , 0 },

/\* вход с символа – S \*/

{/\*. 240 .\*/ 241 , 0 , "S " , 0 },

{/\*. 241 .\*/ 242 , 240 , "U " , 0 },

{/\*. 242 .\*/ 243 , 241 , "B " , 0 },

{/\*. 243 .\*/ 244 , 242 , "S " , 0 },

{/\*. 244 .\*/ 245 , 243 , "T " , 0 },

{/\*. 245 .\*/ 246 , 244 , "R " , 0 },

{/\*. 246 .\*/ 247 , 245 , "( " , 0 },

{/\*. 247 .\*/ 248 , 246 , "LVI" , 0 },

{/\*. 248 .\*/ 249 , 247 , ", " , 0 },

{/\*. 249 .\*/ 250 , 248 , "RZR" , 0 },

{/\*. 250 .\*/ 251 , 249 , ", " , 0 },

{/\*. 251 .\*/ 252 , 250 , "RZR" , 0 },

{/\*. 252 .\*/ 253 , 251 , ") " , 0 },

{/\*. 253 .\*/ 254 , 252 , "SUS" , 0 },

{/\*. 254 .\*/ 0 , 253 , "\* " , 0 },

/\* вход с символа – ! \*/

{/\*. 255 .\*/ 256 , 0 , "! " , 0 },

{/\*. 256 .\*/ 257 , 255 , "! " , 0 },

{/\*. 257 .\*/ 258 , 256 , "ZKL" , 0 },

{/\*. 258 .\*/ 0 , 257 , "\* " , 0 },

/\* вход с символа - U \*/

{/\*. 259 .\*/ 260 , 0 , "U " , 0 },

{/\*. 260 .\*/ 261 , 259 , "BUK" , 0 },

{/\*. 261 .\*/ 0 , 260 , "\* " , 0 }

Добавленные строки таблицы описывают новые правила языка.

# Таблица входов в правила

Ниже представлен фрагменты таблицы входов в правила, введённые в соответствии с новыми возможностями языка:

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

| NN | символ | вход| тип |

|\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_|\_\_\_\_\_\_|

{/\*. 21 .\*/ "OPL" , 0 , 'N' },

{/\*. 22 .\*/ "LVI" , 227 , 'N' },

{/\*. 23 .\*/ "ZKL" , 0 , 'N' },

{/\*. 24 .\*/ "SUS" , 232 , 'N' },

…

{/\*. 52 .\*/ "S " , 240 , 'T' },

{/\*. 53 .\*/ "( " , 235 , 'T' },

{/\*. 62 .\*/ "U " , 259 , 'T' },

{/\*. 63 .\*/ "! " , 255 , 'T' },

{/\*. 64 .\*/ ", " , 0 , 'T' }

# Модификация функций компилятора

С целью расширения функциональности языка в следующие функции компилятора были внесены изменения:

1. Функция уплотнения

void compress\_ISXTXT()

{

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 = 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';

}

1. Функция формирования лексем из уплотненного текста

void FORM()

{

int i, j;

for (IFORMT = 0; IFORMT < MAXFORMT; IFORMT++)

memcpy(FORMT[IFORMT], "\x0\x0\x0\x0\x0\x0\x0\x0\x0", 9);

IFORMT = 0;

j = DST[I2].DST2;

FORM1:

for (i = j; i <= DST[I2].DST4 + 1; i++) {

if (STROKA[i] == ':' || STROKA[i] == ' ' || STROKA[i] == '(' || STROKA[i] == '!'

|| STROKA[i] == ')' || STROKA[i] == ';' || STROKA[i] == '+'

|| STROKA[i] == '-' || STROKA[i] == '=' || STROKA[i] == '\*') {

FORMT[IFORMT][i - j] = '\x0';

IFORMT++;

j = i + 1;

goto FORM1;

} else

FORMT[IFORMT][i - j] = STROKA[i];

}

return;

}

1. Функция вычисления нетерминала ODC (оп. DCL) на первом проходе

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- \*/

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

{

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

/\* идентификатора = 'B' и \*/

goto ODC11;

/\* идем на продолжение об-\*/

/\* работки, а \*/

} else if (!strcmp(FORMT[2], "BIT")) /\* если идентификатор определен как bit, \*/

{

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

/\* идентификатора = 'L' и \*/

goto ODC12;

/\* идем на продолжение об-\*/

/\* работки, а \*/

} else /\* иначе \*/

{

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

/\* тификатора = 'U' и \*/

return 2; /\* завершаем программу \*/

/\* по ошибке \*/

}

ODC11: /\* 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; /\* успешное завешение \*/

/\* программы \*/

ODC12: /\* если идентификатор \*/

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

/\* 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, "0B"); /\* инициализируем иденти- \*/

/\* фикатор нулем \*/

return 0; /\* успешное завешение \*/

/\* программы \*/

}

1. Функция формирования пролога

int OPR2() {

char i = 0;

FORM(); /\* форматируем оператор \*/

/\* ПЛ1 - "начало процедур-\*/

/\* ного блока" \*/

while (FORMT[0][i] != '\x0')

ASS\_CARD.\_BUFCARD.METKA[i++] = FORMT[0][i]; /\* нулевой терм используем\*/

/\* как метку в START-псев-\*/

/\* дооперации Ассемблера \*/

memcpy(ASS\_CARD.\_BUFCARD.OPERAC, "START", 5);/\* достраиваем код и опе- \*/

memcpy(ASS\_CARD.\_BUFCARD.OPERAND, "0", 1); /\* ранды в START-псевдо-\*/

memcpy(ASS\_CARD.\_BUFCARD.COMM, /\* операции Ассемблера \*/

"Nacalo programmy", 16);

ZKARD(); /\* запоминаем карту Ассем-\*/

/\* блера \*/

memcpy(ASS\_CARD.\_BUFCARD.OPERAC, "BALR", 4); /\* формируем BALR-операцию\*/

memcpy(ASS\_CARD.\_BUFCARD.OPERAND, /\* Ассемблера \*/

"RBASE,0", 7);

memcpy(ASS\_CARD.\_BUFCARD.COMM, "Zagruzit' registr bazy", 22);

ZKARD(); /\* и запоминаем ее \*/

memcpy(ASS\_CARD.\_BUFCARD.OPERAC, "USING", 5);/\* формируем USING-псевдо-\*/

memcpy(ASS\_CARD.\_BUFCARD.OPERAND, /\* операцию Ассемблера \*/

"\*,RBASE", 7);

memcpy(ASS\_CARD.\_BUFCARD.COMM, "Naznacit' registr bazoy", 23);

ZKARD(); /\* и запоминаем ее \*/

return 0; /\* завершить подпрограмму \*/

}

1. Функция вычисления нетерминала OEN (оп. END), формирование эпилога

int OEN2() {

char RAB[20];

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,RVIX", 7);/\* операнды команды и \*/

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

"Vyhod iz programmy", 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 == 'L') /\* если тип оператора bit, то: \*/

{

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

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

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

"F\'");/\* первый и \*/

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

SYM[i].RAZR);/\* леровской операции \*/

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

"\'");/\* леровской операции \*/

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

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

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

"Opredelenye peremennoy", 22); /\* тария \*/

ZKARD(); /\* запомнить операцию Ассемблера \*/

int a = 0;

//char \*str;

while(SYM[i].INIT[a] != 'B') {

a++;

}

SYM[i].INIT[a] = 0;

char a2[2];

gcvt((double)a, 2, a2);

if(SYM[i].INIT[0] == '0') {

a2[0] = '0';

}

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

"DC", 2);

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

"F\'");/\* первый и \*/

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

a2);/\* леровской операции \*/

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

"\'");/\* леровской операции \*/

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

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

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

// "Opredelenye peremennoy", 22); /\* тария \*/

ZKARD(); /\* запомнить операцию Ассемблера \*/

if(a == 1) {

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

"DS", 2); /\* ции DC \*/

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

"BL2", 3); /\* ции DC \*/

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

// "Opredelenye peremennoy", 22); /\* тария \*/

ZKARD(); /\* запомнить операцию Ассемблера \*/

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

"TMP"); /\* поле метки псевдооперации 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 \*/

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

"F\'");/\* первый и \*/

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

SYM[i].RAZR);/\* леровской операции \*/

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

"\'");/\* леровской операции \*/

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

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

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

"Opredelenye peremennoy", 22); /\* тария \*/

ZKARD(); /\* запомнить операцию Ассемблера \*/

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

"DC", 2);

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

"F\'");/\* первый и \*/

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

"3");/\* леровской операции \*/

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

"\'");/\* леровской операции \*/

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

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

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

// "Opredelenye peremennoy", 22); /\* тария \*/

ZKARD(); /\* запомнить операцию Ассемблера \*/

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

"DC", 2); /\* ции DC \*/

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

"BL2\'");/\* первый и \*/

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

"111");/\* леровской операции \*/

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

"\'");/\* леровской операции \*/

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

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

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

// "Opredelenye peremennoy", 22); /\* тария \*/

ZKARD(); /\* запомнить операцию Ассемблера \*/

}

else {

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

"DC", 2); /\* ции DC \*/

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

"BL2\'");

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

SYM [i].INIT /\* часть операнда псевдо- \*/

); /\* операции, \*/

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

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

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

// "Opredelenye peremennoy", 22); /\* тария \*/

ZKARD(); /\* запомнить операцию Ассемблера \*/

}

}

}

}

/\* 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, "5", 1); /\* номера базового регист-\*/

/\* ра общего назначения \*/

/\* и \*/

ZKARD(); /\* запоминание ее \*/

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

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

memcpy(ASS\_CARD.\_BUFCARD.OPERAND, "14", 2); /\* 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 \*/

"Konec programmy", 15);

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

/\* рации \*/

return 0; /\* завершение программы \*/

}

1. Функция формирования лексем из уплотненного текста

int gen\_COD() /\*интерпретации строк сте-\*/

{ /\*ка достижений в направ-\*/

int NOSH; /\*лении от дна к вершине. \*/

int (\* FUN [NNETRM][2]) () = /\*При этом каждая строка \*/

{ /\*воспринимается как кома-\*/

{/\* 1 \*/ AVI1, AVI2 }, /\*нда абстрактной ЭВМ со \*/

{/\* 2 \*/ BUK1, BUK2 }, /\*следующими полями: \*/

{/\* 3 \*/ CIF1, CIF2 },

{/\* 4 \*/ IDE1, IDE2 }, /\* - DST.DST1 - код опера-\*/

{/\* 5 \*/ IPE1, IPE2 }, /\*ции; \*/

{/\* 6 \*/ IPR1, IPR2 },

{/\* 7 \*/ LIT1, LIT2 }, /\* - DST.DST2 - левая гра-\*/

{/\* 8 \*/ MAN1, MAN2 }, /\*ница интерпретируемого \*/

{/\* 9 \*/ ODC1, ODC2 }, /\*фрагмента исх.текста; \*/

{/\* 10 \*/ OEN1, OEN2 },

{/\* 11 \*/ OPA1, OPA2 }, /\* - DST.DST4 -правая гра-\*/

{/\* 12 \*/ OPR1, OPR2 }, /\*ница интерпретируемого \*/

{/\* 13 \*/ PRO1, PRO2 }, /\*фрагмента исх.текста. \*/

{/\* 14 \*/ RZR1, RZR2 },

{/\* 15 \*/ TEL1, TEL2 },

{/\* 16 \*/ ZNK1, ZNK2 },

{/\* 17 \*/ OPL1, OPL2 },

{/\* 18 \*/ LVI1, LVI2 },

{/\* 19 \*/ ZKL1, ZKL2 },

{/\* 20 \*/ SUS1, SUS2 }

};

for (I2 = 0; I2 < L; I2++) /\* организация первого \*/

if ((NOSH = FUN[ /\* прохода семантического \*/

numb(DST[I2].DST1, 3) /\* вычисления \*/

][0]()) != 0

) return (

NOSH ); /\* выход из программы \*/

/\* по ошибке \*/

for ( I2 = 0; I2 < L; I2++ ) /\* организация второго \*/

if ( ( NOSH = FUN [ /\* прохода семантического \*/

numb ( DST [I2].DST1, 3 ) /\* вычисления \*/

][1] ()

) != 0

)

return (NOSH); /\* выход из программы \*/

/\* по ошибке \*/

return 0; /\* успешное завершение \*/

/\* программы \*/

}

Кроме того были добавлены следующие новые функции:

1. Оператор логического присваивания на первом проходе

int OPL1() {

return 0;

}

1. Оператор логического присваивания на втором проходе

int OPL2() {

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 == 'L') /\* 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);

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

"SRL", 3);/\* записи полуслова \*/

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

"3,"); //"RRAB,");/\* операнды \*/

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

"16");

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

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

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

"Sdvig vpravo na poluslovo", 25);

ZKARD(); /\* запомнить операцию \*/

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

"STH", 3);/\* записи полуслова \*/

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

"3,"); //"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 \*/

"Formirovanye znacenya logic.virazenya", 37);

ZKARD(); /\* запомнить операцию \*/

/\* Ассемблера и \*/

return 0; /\* завершить программу \*/

}

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 \*/

}

1. Функция логического выражения на первом проходе

int LVI1() {

return 0;

}

1. Функция логического выражения на втором проходе

int LVI2() {

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 == 'L') /\* в случае типа=bin fixed\*/

{

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

"LH", 2);/\* ровской операции LH, \*/

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

"3,");/\* 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\*/

"Zagruzka peremennoy v registr", 29);

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

/\* семблера и \*/

return 0; /\* завершить программу \*/

} 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(STROKA[DST[I2].DST4] == ')')

return 0;

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 == 'L') /\* если тип правого опе- \*/

{ /\* ранда bin fixed, то: \*/

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

2] == '!') &&

(STROKA[DST[I2].DST4 -

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

}

else

{

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

/\* "+" и не "-", то завер-\*/

/\* шение программы по \*/

/\* ошибке \*/

}

/\* формируем: \*/

memcpy(ASS\_CARD.\_BUFCARD.OPERAC, "LH", 2);/\* формируем код ассембле-\*/

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

"4,");/\*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,

"Formirovanye promezutocnogo znacenya",/\* - построчный коментарий\*/

36);

ZKARD(); /\* запоминание ассемблеровской операции \*/

/\* формируем: \*/

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

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

"4,");/\*блеровской операции; \*/

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

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

ASS\_CARD.\_BUFCARD.OPERAND[strlen(ASS\_CARD.\_BUFCARD.OPERAND)]

= /\* - разделяющий пробел; \*/

' ';

memcpy(ASS\_CARD.\_BUFCARD.COMM,

"Sdvig operanda vpravo",/\* - построчный коментарий\*/

21);

ZKARD(); /\* запоминание ассемблеровской операции \*/

/\* формируем: \*/

memcpy(ASS\_CARD.\_BUFCARD.OPERAC, "OR", 2);/\* формируем код ассембле-\*/

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

"3,");/\*блеровской операции; \*/

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

"4");/\*блеровской операции; \*/

ASS\_CARD.\_BUFCARD.OPERAND[strlen(ASS\_CARD.\_BUFCARD.OPERAND)]

= /\* - разделяющий пробел; \*/

' ';

memcpy(ASS\_CARD.\_BUFCARD.COMM,

"Logicheskoye ILI registrov",/\* - построчный коментарий\*/

26);

ZKARD(); /\* запоминание ассемблеровской операции \*/

return 0; /\* успешное завершение \*/

/\* пограммы \*/

} 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 \*/

}

}

1. Функция логического знака на первом проходе

int ZKL1() {

return 0;

}

1. Функция логического знака на втором проходе

int ZKL2() {

return 0;

}

1. Функция вычисления SUS на первом проходе (оп. substr)

int SUS1() {

return 0;

}

1. Функция вычисления SUS на втором проходе (оп. substr)

int SUS2() {

/\* формируем: \*/

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

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

"3,");/\*блеровской операции; \*/

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

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

ASS\_CARD.\_BUFCARD.OPERAND[strlen(ASS\_CARD.\_BUFCARD.OPERAND)]

= /\* - разделяющий пробел; \*/

' ';

memcpy(ASS\_CARD.\_BUFCARD.COMM,

"Sdvig operanda vlevo",/\* - построчный коментарий\*/

20);

ZKARD(); /\* запоминание ассемблеровской операции \*/

/\* формируем: \*/

memcpy(ASS\_CARD.\_BUFCARD.OPERAC, "LH", 2);/\* формируем код ассембле-\*/

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

"4,");/\*блеровской операции; \*/

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

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

ASS\_CARD.\_BUFCARD.OPERAND[strlen(ASS\_CARD.\_BUFCARD.OPERAND)]

= /\* - разделяющий пробел; \*/

' ';

memcpy(ASS\_CARD.\_BUFCARD.COMM,

"Zagruzka maski v reg",/\* - построчный коментарий\*/

20);

ZKARD(); /\* запоминание ассемблеровской операции \*/

/\* формируем: \*/

memcpy(ASS\_CARD.\_BUFCARD.OPERAC, "NR", 2);/\* формируем код ассембле-\*/

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

"3,");/\*блеровской операции; \*/

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

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

ASS\_CARD.\_BUFCARD.OPERAND[strlen(ASS\_CARD.\_BUFCARD.OPERAND)]

= /\* - разделяющий пробел; \*/

' ';

memcpy(ASS\_CARD.\_BUFCARD.COMM,

"Logiceskoye I registr",/\* - построчный коментарий\*/

21);

ZKARD(); /\* запоминание ассемблеровской операции \*/

return 0;

}

# Выводы

В рамках первого этапа курсовой работы по написанию компилятора с языка высокого уровня были выполнены все поставленные задачи:

1. Разработан код на ассемблере, выполняющий действия, эквивалентные описанным на языке PL1
2. Модифицирована грамматика и синтаксические правила языка
3. Существующий компилятор доработан с учетом новых правил

Таким образом можно говорить об успешном завершении первого этапа, результатом которого стал компилятор с языка PL1 на ассемблер в рамках задания.

В качестве самой трудоемкой задачи можно отметить разбор и изменение существующего кода компилятора.