

Минобрнауки России
Санкт-Петербургский государственный политехнический университет
Институт информационных технологий и управления
Кафедра «Информационные и управляющие системы»

КУРСОВОЙ ПРОЕКТ

Разработка учебной системы программирования
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
План работы.....	3
Эквивалент на Ассемблере	4
Модификация грамматики языка	5
Матрица смежности.....	9
Таблица синтаксических правил	10
Таблица входов в правила	12
Модификация функций компилятора	13
Выводы.....	19

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 функциональностью. 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	Начало программы
	BALR	RBASE, 0	Загрузка регистра базы
	USING	*, RBASE	Назначить регистр базой
	LH	3, B	Загрузка переменной B в регистр
	LH	4, A	Загрузка переменной A в регистр
	OR	3, 4	Логическое «ИЛИ» регистров
	SRL	4, 3	Сдвиг операнда вправо
	SLL	3, 2	Сдвиг операнда влево
	LH	4, TMP	Загрузка маски в регистр
	NR	3, 4	Логическое «И» регистров
	STH	3, C	Формирование результата
	BCR	15, RVIX	Выход из программы
A	DS	0H	Выравнивание адреса
	DC	BL2'10'	Инициализация переменной
B	DS	0H	Выравнивание адреса
	DC	BL2'101'	Инициализация переменной
C	DS	0H	Выравнивание адреса
	DS	BL2	Объявление без инициализации
TMP	DS	0H	Выравнивание адреса
	DC	BL2'111'	Инициализация переменной
RBASE	EQU	5	
RVIX	EQU	14	
	END	EX07	Конец программы

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 следующей строке происходит запись содержимого из регистра в память.

Далее идёт объявление переменных. BL2 – значение переменной в логическом виде (выделяется 16 разрядов). Команда DS 0H служит для выравнивания адреса на границу полуслова. Параметр 0 позволяет осуществить выравнивания без выделения памяти.

Последние 3 строки программы – эпилог.

Модификация грамматики языка

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

1. $\langle \text{PRO} \rangle ::= \langle \text{OPR} \rangle \langle \text{TEL} \rangle \langle \text{OEN} \rangle$
2. $\langle \text{OPR} \rangle ::= \langle \text{IPR} \rangle : \text{PROC_OPTIONS}(\text{MAIN}) ;$
3. $\langle \text{IPR} \rangle ::= \langle \text{IDE} \rangle$
4. $\langle \text{IDE} \rangle ::= \langle \text{BUK} \rangle \mid \langle \text{IDE} \rangle \langle \text{BUK} \rangle \mid \langle \text{IDE} \rangle \langle \text{CIF} \rangle$
5. $\langle \text{BUK} \rangle ::= \text{A} \mid \text{B} \mid \text{C} \mid \text{D} \mid \text{E} \mid \text{M} \mid \text{P} \mid \text{X}$
6. $\langle \text{CIF} \rangle ::= 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$
7. $\langle \text{TEL} \rangle ::= \langle \text{ODC} \rangle \mid \langle \text{TEL} \rangle \langle \text{ODC} \rangle \mid \langle \text{TEL} \rangle \langle \text{OPA} \rangle$
 $\mid \langle \text{TEL} \rangle \langle \text{OPL} \rangle$
8. $\langle \text{ODC} \rangle ::= \text{DCL_IPE_BIN_FIXED}(\langle \text{RZR} \rangle) ; \mid$
 $\text{DCL_IPE_BIN_FIXED}(\langle \text{RZR} \rangle) \text{INIT}(\langle \text{LIT} \rangle) ;$
 $\mid \text{DCL_IPE_BIT}(\langle \text{RZR} \rangle) ; \mid \text{DCL_IPE_BIT}(\langle \text{RZR} \rangle) \text{INIT}(\langle \text{LIT} \rangle) ;$
9. $\langle \text{IPE} \rangle ::= \langle \text{IDE} \rangle$
10. $\langle \text{RZR} \rangle ::= \langle \text{CIF} \rangle \mid \langle \text{RZR} \rangle \langle \text{CIF} \rangle$
11. $\langle \text{LIT} \rangle ::= \langle \text{MAN} \rangle \text{B}$
12. $\langle \text{MAN} \rangle ::= 1 \mid \langle \text{MAN} \rangle 0 \mid \langle \text{MAN} \rangle 1$
13. $\langle \text{OPA} \rangle ::= \langle \text{IPE} \rangle = \langle \text{AVI} \rangle ;$
- 13.1 $\langle \text{OPL} \rangle ::= \langle \text{IPE} \rangle = \langle \text{LVI} \rangle ;$
14. $\langle \text{AVI} \rangle ::= \langle \text{LIT} \rangle \mid \langle \text{IPE} \rangle \mid \langle \text{AVI} \rangle \langle \text{ZNK} \rangle \langle \text{LIT} \rangle \mid$
 $\langle \text{AVI} \rangle \langle \text{ZNK} \rangle \langle \text{IPE} \rangle$
- 14.1 $\langle \text{LVI} \rangle ::= \langle \text{IPE} \rangle \mid \langle \text{LVI} \rangle \langle \text{ZNKL} \rangle \langle \text{IPE} \rangle \mid \langle \text{SSTR} \rangle \mid (\langle \text{LVI} \rangle)$
15. $\langle \text{ZNK} \rangle ::= + \mid -$
- 15.1 $\langle \text{ZNKL} \rangle ::= !!$
- 15.2 $\langle \text{SSTR} \rangle ::= \text{SUBSTR}(\langle \text{LVI} \rangle, \langle \text{RZR} \rangle, \langle \text{RZR} \rangle)$
16. $\langle \text{OEN} \rangle ::= \text{END_IPR}$

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",
- "<ZNKL>" - 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>
28. DCL_IPE_BIN_FIXED(<RZR>); -> <ODC>
29. DCL_IPE_BIN_FIXED(<RZR>) INIT(<LIT>); -> <ODC>
30. DCL_IPE_BIT(<RZR>); -> <ODC>
31. DCL_IPE_BIT(<RZR>) INIT(<LIT>); -> <ODC>
32. <IDE> -> <IPE>
33. <CIF> -> <RZR>
34. <RZR><CIF> -> <RZR>
35. <MAN>B -> <LIT>
36. 1 -> <MAN>
37. <MAN>0 -> <MAN>
38. <MAN>1 -> <MAN>
39. <IPE>=<AVI>; -> <OPA>
40. <IPE>=<LVI>; -> <OPL>
41. <LIT> -> <AVI>
42. <IPE> -> <AVI>
43. <AVI><ZNK><LIT> -> <AVI>
44. <AVI><ZNK><IPE> -> <AVI>
45. <IPE> -> <LVI>
46. <LVI><ZKNL><IPE> -> <LVI>
47. <SSTR> -> <LVI>
48. (<LVI>) -> <LVI>
49. + -> <ZNK>
50. - -> <ZNK>
51. !! -> <ZKNL>
52. SUBSTR(<LVI>, <RZR>, <RZR>) -> <SSTR>
53. 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>
 - L-> <CIF> -> <IDE>
 - L-> <IPR>
 - L-> <IPE>
4. <BUK> -> <IDE>
5. A -> <BUK>
6. B -> <BUK>
7. C -> <BUK>
8. D -> CL_IPE_BIN_FIXED(<RZR>)INIT(<LIT>); -> <ODC>
 - | L-> ; -> <ODC>
 - | L->T(<RZR>)INIT(<LIB>); -> <ODC>
 - | L-> ; -> <ODC>
 - L <BUK>
9. E -> ND_IPR -> <OEN>
 - L-> <BUK>
10. M -> <BUK>
11. P -> <BUK>
12. X -> <BUK>
13. 0 -> <CIF>
14. 1 -> <CIF>
 - L-> <MAN>
15. 2 -> <CIF>
16. 3 -> <CIF>
17. 4 -> <CIF>
18. 5 -> <CIF>
19. 6 -> <CIF>
20. 7 -> <CIF>
21. 8 -> <CIF>
22. 9 -> <CIF>
23. <ODC> -> <TEL>
24. <TEL> -> <ODC> -> <TEL>
 - L-> <OPA> -> <TEL>
 - L-> <OPL> -> <TEL>
25. <CIF> -> <RZR>
26. <RZR><CIF> -> <RZR>
27. <MAN> -> B -> <LIT>
 - L-> 0 -> <MAN>
 - L-> 1 -> <MAN>
28. <IPE> -> =<AVI>; -> <OPA>
 - | L-> <LVI>; -> <OPL>
 - L-> <AVI>
 - L-> <LVI>
29. <LIT> -> <AVI>
30. <AVI><ZNK> -> <LIT> -> <AVI>
 - L-> <IPE> -> <AVI>
31. <LVI><ZKNL><IPE> -> <LVI>
32. <SSSTR> -> <LVI>
33. (<LVI>) -> <LVI>
34. + -> <ZNK>
35. - -> <ZNK>
36. !! -> <ZKNL>
37. SUBSTR(<LVI>, <RZR>, <RZR>) -> <SSSTR>

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	IFE	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
{/* ZNKL */	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 },

/*
{ /*. 255      .*/ 256 , 0 , "!" , 0 },
{ /*. 256      .*/ 257 , 255 , "!" , 0 },
{ /*. 257      .*/ 258 , 256 , "ZKL" , 0 },
{ /*. 258      .*/ 0 , 257 , "*" , 0 },

/*
{ /*. 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, введенные 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;
    char current = 0;
    for (I1 = 0; I1 < NISXTXT; I1++) {
        for (I2 = 0; I2 < 80; I2++){
            current = ISXTXT[I1][I2];
            if (current == '\x0') {
                break;
            }
            if (current == '\n') {
                continue;
            }
            if (current == ' ' && (PREDSYM == ' ' || PREDSYM == ';' || PREDSYM == ')' || PREDSYM == ':') {
                PREDSYM = current;
                continue;
            }
            if ((current == '!' || current == '+' || current == '-' || current == '=' || current == '(' || current == ')' || current == '*' && PREDSYM == ' ') {
                I3--;
                PREDSYM = current;
                STROKA[I3] = PREDSYM;
                I3++;
                continue;
            }
            if (current == ' ' && (PREDSYM == '+' || PREDSYM == '-' || PREDSYM == '=' || PREDSYM == '*' || PREDSYM == '!')) {
                continue;
            }
            PREDSYM=current;
            STROKA[I3]=current;
            I3++;
        }
        STROKA[I3] = '\x0';
    }
}
```

2) 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", 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;
}
```

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

```
int ODC1() {
    int i;
    FORM();
    for (i = 0; i < ISYM; i++)
    {
        if (!strcmp(SYM[i].NAME, FORMT[1]) && strlen(SYM[i].NAME) == strlen(FORMT[1]))
            return 6;
    }
    strcpy(SYM[ISYM].NAME, FORMT[1]);
    strcpy(SYM[ISYM].RAZR, FORMT[4]);
    /* фикатора запоминаем его*/
    /* вместе с разрядностью в*/
    /* табл.SYM */
    if (!strcmp(FORMT[2], "BIN") && !strcmp(FORMT[3], "FIXED")
    {
        SYM[ISYM].TYPE = 'B';
        goto ODC11;
    } else if (!strcmp(FORMT[2], "BIT"))
    {
        SYM[ISYM].TYPE = 'I';
        goto ODC12;
    } else
    {
        SYM[ISYM].TYPE = 'U';
        return 2;
    }
    ODC11:
    if (!strcmp(FORMT[5], "INIT"))
        strcpy(SYM[ISYM++].INIT, FORMT[6]);
    else
        strcpy(SYM[ISYM++].INIT, "0B");
    return 0;
    ODC12:
    strcpy(SYM[ISYM].RAZR, FORMT[3]);
    if (!strcmp(FORMT[4], "INIT"))
        strcpy(SYM[ISYM++].INIT, FORMT[5]);
    else
        strcpy(SYM[ISYM++].INIT, "0B");
    return 0;
}
```

4) Функция формирования пролога

```
int OPR2() {
    char i = 0;
    FORM();
    while (FORMT[0][i] != '\x0')
        ASS_CARD._BUFCARD.METKA[i++] = FORMT[0][i];
    memcpy(ASS_CARD._BUFCARD.OPERAC, "START", 5);
    memcpy(ASS_CARD._BUFCARD.OPERAND, "0", 1);
    memcpy(ASS_CARD._BUFCARD.COMM, "Nacalo programmy", 16);
    ZKARD();
    memcpy(ASS_CARD._BUFCARD.OPERAC, "BALR", 4);
    memcpy(ASS_CARD._BUFCARD.OPERAND, "RBASE,0", 7);
    memcpy(ASS_CARD._BUFCARD.COMM, "Zagruzit' registr bazy", 22);
    ZKARD();
    memcpy(ASS_CARD._BUFCARD.OPERAC, "USING", 5);
    memcpy(ASS_CARD._BUFCARD.OPERAND, "*", RBASE, 7);
    memcpy(ASS_CARD._BUFCARD.COMM, "Naznagit' registr bazoy", 23);
    ZKARD();
    return 0;
}
```

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

```

int OEN2() {
    char RAB[20];
    char i = 0;
    FORM();
    memcpy(ASS_CARD._BUFCARD.OPERAC, "BCR", 3);
    memcpy(ASS_CARD._BUFCARD.OPERAND, "15,RVIX", 7);
    memcpy(ASS_CARD._BUFCARD.COMM,
        "Vyhod iz programmy", 18);
    ZKARD();
    for (i = 0; i < ISYM; i++) {
        if (isalpha ( SYM [i].NAME [0] ))
        {
            if (SYM[i].TYPE == 'B')
            {
                strcpy(ASS_CARD._BUFCARD.METKA, SYM[i].NAME);
                ASS_CARD._BUFCARD.METKA[strlen(ASS_CARD._BUFCARD.METKA)] = ' ';
                memcpy(ASS_CARD._BUFCARD.OPERAC, "DC", 2);
                if (strcmp(SYM[i].RAZR, "15") <= 0)
                    strcpy(ASS_CARD._BUFCARD.OPERAND, "H\\");
                else
                    strcpy(ASS_CARD._BUFCARD.OPERAND, "F\\");
                strcat(ASS_CARD._BUFCARD.OPERAND, gcvt(VALUE(SYM[i].INIT), 10,
                    ASS_CARD._BUFCARD.OPERAND[strlen(ASS_CARD._BUFCARD.OPERAND)] =
                    '\n');
                memcpy(ASS_CARD._BUFCARD.COMM,
                    "Определение переменной", 22);
                ZKARD();
            } else if (SYM[i].TYPE == 'L')
            {
                strcpy(ASS_CARD._BUFCARD.METKA, SYM[i].NAME);
                ASS_CARD._BUFCARD.METKA[strlen(ASS_CARD._BUFCARD.METKA)] = ' ';
                int a = 0;
                while (SYM[i].INIT[a] != 'B') {
                    a++;
                }
                SYM[i].INIT[a] = 0;

                if (a == 1) {
                    memcpy(ASS_CARD._BUFCARD.OPERAC, "DS", 2);
                    strcpy(ASS_CARD._BUFCARD.OPERAND, "0H");
                    ASS_CARD._BUFCARD.OPERAND[strlen(ASS_CARD._BUFCARD.OPERAND)]
                    =
                    ' ';
                    ZKARD();
                    memcpy(ASS_CARD._BUFCARD.OPERAC, "DS", 2);
                    memcpy(ASS_CARD._BUFCARD.OPERAND, "BL2", 3);
                    ZKARD();
                    strcpy(ASS_CARD._BUFCARD.METKA, "TMP");
                    ASS_CARD._BUFCARD.METKA[strlen(ASS_CARD._BUFCARD.METKA)] = '
';
                    memcpy(ASS_CARD._BUFCARD.OPERAC, "DS", 2);
                    strcpy(ASS_CARD._BUFCARD.OPERAND, "0H");
                    ASS_CARD._BUFCARD.OPERAND[strlen(ASS_CARD._BUFCARD.OPERAND)]
                    = ' ';
                    ZKARD();
                    memcpy(ASS_CARD._BUFCARD.OPERAC, "DC", 2);
                    strcpy(ASS_CARD._BUFCARD.OPERAND, "BL2\\");
                    strcat(ASS_CARD._BUFCARD.OPERAND, "111");
                    strcat(ASS_CARD._BUFCARD.OPERAND, "\\");
                    ASS_CARD._BUFCARD.OPERAND[strlen
                    (ASS_CARD._BUFCARD.OPERAND)] = ' ';
                    ZKARD();
                } else {
                    memcpy(ASS_CARD._BUFCARD.OPERAC, "DS", 2);
                    strcpy(ASS_CARD._BUFCARD.OPERAND, "0H");
                    ASS_CARD._BUFCARD.OPERAND[strlen(ASS_CARD._BUFCARD.OPERAND)]
                    = ' ';
                    ZKARD();
                    memcpy(ASS_CARD._BUFCARD.OPERAC, "DC", 2);
                    strcpy(ASS_CARD._BUFCARD.OPERAND, "BL2\\");
                    strcat(ASS_CARD._BUFCARD.OPERAND, SYM[i].INIT);
                    ASS_CARD._BUFCARD.OPERAND[strlen(ASS_CARD._BUFCARD.OPERAND)]
                    = '\\';
                    ZKARD();
                }
            }
        }
    }
}

```

```

    }
}

memcpy(ASS_CARD._BUFCARD.METKA, "RBASE", 5);
memcpy(ASS_CARD._BUFCARD.OPERAC, "EQU", 3);
memcpy(ASS_CARD._BUFCARD.OPERAND, "5", 1);
ZKARD();
memcpy(ASS_CARD._BUFCARD.METKA, "RVIX", 4);
memcpy(ASS_CARD._BUFCARD.OPERAC, "EQU", 3);
memcpy(ASS_CARD._BUFCARD.OPERAND, "14", 2);
ZKARD();
memcpy(ASS_CARD._BUFCARD.OPERAC, "END", 3); /
i = 0;
while (FORMT[1][i] != '\x0') ASS_CARD._BUFCARD.OPERAND[i] = FORMT[1][i++];
memcpy(ASS_CARD._BUFCARD.COMM, "Konec programy", 15);
ZKARD();
return 0;
}

```

6) Функция формирования лексем из уплотненного текста

```

int gen_COD()
{
    int NOSH;
    int (* FUN [NNETRM][2]) () =
    {
        /* 1 */ AVI1, AVI2 },
        /* 2 */ BUK1, BUK2 },
        /* 3 */ CIF1, CIF2 },
        /* 4 */ IDE1, IDE2 },
        /* 5 */ IPE1, IPE2 },
        /* 6 */ IPR1, IPR2 },
        /* 7 */ LIT1, LIT2 },
        /* 8 */ MAN1, MAN2 },
        /* 9 */ ODC1, ODC2 },
        /* 10 */ OEN1, OEN2 },
        /* 11 */ OPA1, OPA2 },
        /* 12 */ OPR1, OPR2 },
        /* 13 */ PRO1, PRO2 },
        /* 14 */ RZR1, RZR2 },
        /* 15 */ TEL1, TEL2 },
        /* 16 */ ZNK1, ZNK2 },
        /* 17 */ OPL1, OPL2 },
        /* 18 */ LV11, LV12 },
        /* 19 */ ZKL1, ZKL2 },
        /* 20 */ SUS1, SUS2 }
    };
    for (I2 = 0; I2 < L; I2++)
        if ((NOSH = FUN[numb(DST[I2].DST1, 3)][0]()) != 0)
            return (NOSH);
    for (I2 = 0; I2 < L; I2++)
        if ((NOSH = FUN[numb(DST[I2].DST1, 3)][1]()) != 0)
            return (NOSH);
    return 0;
}

```

Кроме того были добавлены следующие новые функции:

1) Оператор логического присваивания на первом проходе

```

int OPL1() {
    return 0;
}

```

2) Оператор логического присваивания на втором проходе

```

int OPL2() {
    int i;
    FORM();
    for (i = 0; i < ISYM; i++) {
        if (!strcmp(SYM[i].NAME, FORMT[0]) && strlen(SYM[i].NAME) == strlen(FORMT[0])) {
            if (SYM[i].TYPE == 'L')

```



```

        {
            memcpy(ASS_CARD._BUFCARD.OPERAC, "STH", 3);
            strcpy(ASS_CARD._BUFCARD.OPERAND, "3,");
            strcat(ASS_CARD._BUFCARD.OPERAND, FORMT[0]);
            ASS_CARD._BUFCARD.OPERAND[strlen(ASS_CARD._BUFCARD.OPERAND)] = ' ';
            memcpy(ASS_CARD._BUFCARD.COMM, "Formirovaniye znaceniya
logic.virazeniya", 37);

            ZKARD();
            return 0;
        }
        else
            return 3;
    }
    return 4;
}

```

3) Функция логического выражения на первом проходе

```

int LVI1() {
    return 0;
}

```

4) Функция логического выражения на втором проходе

```

int LVI2() {
    char i;
    FORM();
    if (IFORMT == 1)
    {
        for (i = 0; i < ISYM; i++)
        {
            if (!strcmp(SYM[i].NAME, FORMT[0]) && strlen(SYM[i].NAME) ==
strlen(FORMT[0])) {
                if (SYM[i].TYPE == 'L')
                {
                    memcpy(ASS_CARD._BUFCARD.OPERAC, "LH", 2);
                    strcpy(ASS_CARD._BUFCARD.OPERAND, "3,");
                    strcat(ASS_CARD._BUFCARD.OPERAND, FORMT[0]);
                    ASS_CARD._BUFCARD.OPERAND[strlen(ASS_CARD._BUFCARD.OPERAND)]
= ' ';
                    memcpy(ASS_CARD._BUFCARD.COMM, "Zagruzka peremennoy v
registr", 29);

                    ZKARD();
                    return 0;
                } else
                    return 3;
            }
        }
        return 4; /
    } else
    {
        for (i = 0; i < ISYM; i++)
        {
            if(STROKA[DST[I2].DST4] == ' ')
                return 0;
            if (!strcmp(SYM[i].NAME,
FORMT[IFORMT - 1]) && strlen(SYM[i].NAME) == strlen(FORMT[IFORMT-
1])) {
                memcpy(ASS_CARD._BUFCARD.OPERAC, "LH", 2);
                strcpy(ASS_CARD._BUFCARD.OPERAND, "4,");
                strcat(ASS_CARD._BUFCARD.OPERAND, FORMT[IFORMT - 1]);
                ASS_CARD._BUFCARD.OPERAND[strlen(ASS_CARD._BUFCARD.OPERAND)]= ' ';
                memcpy(ASS_CARD._BUFCARD.COMM, "Formirovaniye promezutocnogo
znaceniya", 36);

                ZKARD();
                memcpy(ASS_CARD._BUFCARD.OPERAC, "SRL", 3);
                strcpy(ASS_CARD._BUFCARD.OPERAND, "4,");
                strcat(ASS_CARD._BUFCARD.OPERAND, "3");
                ASS_CARD._BUFCARD.OPERAND[strlen(ASS_CARD._BUFCARD.OPERAND)]= ' ';
                memcpy(ASS_CARD._BUFCARD.COMM, "Sdvig operanda vpravo", 21);
                ZKARD();
            }
        }
    }
}

```

```

        memcpy(ASS_CARD._BUFCARD.OPERAC, "OR", 2);
        strcpy(ASS_CARD._BUFCARD.OPERAND, "3,");
        strcat(ASS_CARD._BUFCARD.OPERAND, "4");
        ASS_CARD._BUFCARD.OPERAND[strlen(ASS_CARD._BUFCARD.OPERAND)] = ' ';
        memcpy(ASS_CARD._BUFCARD.COMM, "Logicheskoye ILI registrov", 26);
        ZKARD();
        return 0;
    } else
        return 3;
    }
}
return 4;
}
}

```

5) Функция логического знака на первом проходе

```

int ZKL1() {
    return 0;
}

```

6) Функция логического знака на втором проходе

```

int ZKL2() {
    return 0;
}

```

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

```

int SUS1() {
    return 0;
}

```

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

```

int SUS2() {
    memcpy(ASS_CARD._BUFCARD.OPERAC, "SLL", 3);
    strcpy(ASS_CARD._BUFCARD.OPERAND, "3,");
    strcat(ASS_CARD._BUFCARD.OPERAND, "1");
    ASS_CARD._BUFCARD.OPERAND[strlen(ASS_CARD._BUFCARD.OPERAND)] = ' ';
    memcpy(ASS_CARD._BUFCARD.COMM, "Sdvig operanda vlevo", 20);
    ZKARD();
    memcpy(ASS_CARD._BUFCARD.OPERAC, "LH", 2);
    strcpy(ASS_CARD._BUFCARD.OPERAND, "4,");
    strcat(ASS_CARD._BUFCARD.OPERAND, "TMP");
    ASS_CARD._BUFCARD.OPERAND[strlen(ASS_CARD._BUFCARD.OPERAND)] = ' ';
    memcpy(ASS_CARD._BUFCARD.COMM, "Zagruzka maski v reg", 20);
    ZKARD();
    memcpy(ASS_CARD._BUFCARD.OPERAC, "NR", 2);
    strcpy(ASS_CARD._BUFCARD.OPERAND, "3,");
    strcat(ASS_CARD._BUFCARD.OPERAND, "4");
    ASS_CARD._BUFCARD.OPERAND[strlen(ASS_CARD._BUFCARD.OPERAND)] = ' ';
    memcpy(ASS_CARD._BUFCARD.COMM, "Logicheskoye I registr", 21);
    ZKARD();
    return 0;
}

```

Выводы

В рамках первого этапа курсовой работы по написанию компилятора с языка высокого уровня в рамках поставленной задачи 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. В качестве основных проблем можно отметить:

- 1) Избыточное количество глобальных переменных
- 2) Неудобные комментарии
 - ✓ Комментарии написаны на русском языке
 - ✓ Комментарии написаны по столбцам а не по строкам. При форматировании кода они съезжают
- 3) Большое количество операторов goto
- 4) Объявление методов и переменных по правилам транслитерации (было бы удобнее, если бы имена соответствовали английским названиям)