Санкт-Петербургский политехнический университет Петра Великого

Кафедра компьютерных систем и программных технологий

**Отчёт по лабораторной работе №2.2**

**Дисциплина**: Транслирующие системы

**Тема**: «Индивидуальное задание (утилиты YACC и LEX)»

Выполнил студент гр. 3530901/70203 И.Д. Иванов

(подпись)

Преподаватель В.Н. Цыган (подпись)

“\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_ 2019 г.

Санкт-Петербург

2019

1. **Цель работы:**

Закрепление полученных навыков применения утилит yacc и lex для генерирования синтаксических анализаторов.

1. **Задание**

Построить транслятор оператора if-else языка С в язык a86. Тип данных – только целые. Условное выражение – арифметическое бесскобочное с сохранением приоритетов. Инструкции на ветвях if-else – простейшие операторы присваивания вида:

* id = id
* id = const
* id = op1 операция op2

Предусмотреть 1-2 собственных диагностических сообщения.

1. **Выполнение работы**

**Входная грамматика:**

prog : statement | statement prog;

statement : if\_start cond body else\_start body | if\_start cond body;

if\_start : \_IF;

else\_start : \_ELSE;

cond : '(' exp ')' ;

va : VAR\_NAME | VAR\_VALUE ;

exp\_md : va '\*' va | va '/' va | exp\_md '\*' va | exp\_md '/' va ;

exp : va | exp '+' va | exp '-' va | exp '+' exp\_md | exp '-' exp\_md | exp\_md

body : '{' nest\_code '}' | '{' '}';

nest\_code : assign | statement | assign nest\_code | statement nest\_code;

assign : VAR\_NAME '=' va ';' | VAR\_NAME '=' exp ';' ;

**Лексический анализатор:**

%{

#include "y.tab.h"

%}

%%

"eles" {yyerror("Warning: eles, did you mean else?");yylval.value = strdup(yytext); return VAR\_NAME;}

"if" {return \_IF;}

"else" {return \_ELSE;}

[a-zA-Z][0-9a-zA-Z\_]\* { yylval.value = strdup(yytext); return VAR\_NAME; }

[0-9]+ { yylval.value = strdup(yytext); return VAR\_VALUE;}

"+" |

"-" |

"\*" |

"/" |

"=" |

"(" |

")" |

"{" |

"}" |

";" |

":" {return yytext[0];}

[ \n\t] ;

%%

int yywrap() {return 1;}

**Синтаксический анализатор:**

%union {

char\* value;

};

%{

#include <stdlib.h>

#include <string.h>

int counter=0;

int else\_counter=0;

int md\_counter=0;

%}

%token \_IF \_ELSE

%token <value>VAR\_NAME

%token <value>VAR\_VALUE

%start prog

%%

prog : statement {printf("L:%d\n",counter+1);}

| statement prog ;

statement : if\_start cond body else\_start body | if\_start cond body

| if\_start body {yyerror("Error: missing if condition");exit(1);};

if\_start : \_IF {counter++; else\_counter=0; printf("L:%d\n",counter);};

else\_start: \_ELSE {counter++; else\_counter++; printf(" jmp L%d\n", counter+1);

printf("L:%d\n",counter);};

cond : '(' exp ')' {printf(" mov bx, %s\n cmp ax, bx\n", "0");

printf(" je L%d\n", counter+1); md\_counter=0;};

va : VAR\_NAME {$<value>$ = $<value>1;}

| VAR\_VALUE {$<value>$ = $<value>1;};

exp\_md : va '\*' va {

printf(" mov a%d, %s\n", md\_counter,$<value>1);

printf(" mul a%d, %s\n", md\_counter, $<value>3);$<value>$=md\_counter;}

| va '/' va {

printf(" mov a%d, %s\n", md\_counter, $<value>1);

printf(" div a%d, %s\n", md\_counter, $<value>3);$<value>$=md\_counter;}

| exp\_md '\*' va {

printf(" mul a%d, %s\n", md\_counter, $<value>3);$<value>$=md\_counter;}

| exp\_md '/' va {

printf(" div a%d, %s\n", md\_counter, $<value>3);$<value>$=md\_counter;}

exp : va {printf(" mov ax, %s\n", $<value>1);}

| exp '+' va { printf(" add ax, %s\n", $<value>3);}

| exp '-' va { printf(" sub ax, %s\n", $<value>3);}

| exp '+' exp\_md { printf(" add ax, a%d\n", $<value>3); md\_counter++;}

| exp '-' exp\_md { printf(" sub ax, a%d\n", $<value>3); md\_counter++;}

| exp\_md {printf(" mov ax, a%d\n", $<value>1); md\_counter++;};

body : '{' nest\_code '}' | '{' '}';

nest\_code : assign | statement | assign nest\_code | statement nest\_code;

assign : VAR\_NAME '=' va ';' {printf(" mov %s, %s\n", $<value>1, $<value>3); md\_counter=0;}

| VAR\_NAME '=' exp ';' {printf(" mov %s, ax\n", $<value>1);md\_counter=0;}

| VAR\_VALUE '=' va ';' {yyerror("Error : You can't assing value to an integer");exit(1);}

| VAR\_VALUE '=' exp ';' {yyerror("Error : You can't assing value to an integer");exit(1);};

%%

Транслятор обрабатывает любое число последовательных операторов if/if-else. Переменная *counter* является счетчиком блоков if/else, она необходима для формирования номеров меток. Переменная *else\_counter* – счётчик количества операторов else у одного if. Переменная md\_counter необходима для грамотной обработки арифметических выражений с сохранением приоритетов.

В таблице 1 приведены эквивалентные записи операторов и действий на языках C и a86.

*Таблица 1*

|  |  |
| --- | --- |
| Язык С | Ассемблер a86 |
| if(5-3/4\*asd) | mov ax, 5  mov a0, 3  div a0, 4  mul a0, asd  sub ax, a0  mov bx, 0  cmp ax, bx |
| if(…){  …  }  else{  …  } | L:1  …  mov bx, 0  cmp ax, bx  je L2  …  jmp L3  L:2  …  L:3  … |
| asdf = 4+15-10+id7\*100 | mov ax, 4  add ax, 15  sub ax, 10  mov a0, id7  mul a0, 100  add ax, a0  mov asdf, ax |

**Пример работы транслятора:**

Входной файл:

if(4+15-10+var\*100){

id2=id15;

}

else{

id1=id3;

}

if(5-3/4\*asd){

id5=id2\*4;

id3=2;

id2=4\*2-100/id3;

}

if(123\*s1/111+433\*s2\*s3/s4+54){

}

else{

asdf=4+15-10+id7\*100;

}

Выходной файл:

L:1

mov ax, 4

add ax, 15

sub ax, 10

mov a0, var

mul a0, 100

add ax, a0

mov bx, 0

cmp ax, bx

je L2

mov id2, id15

jmp L3

L:2

mov id1, id3

L:3

mov ax, 5

mov a0, 3

div a0, 4

mul a0, asd

sub ax, a0

mov bx, 0

cmp ax, bx

je L4

mov a0, id2

mul a0, 4

mov ax, a0

mov id5, ax

mov id3, 2

mov a0, 4

mul a0, 2

mov ax, a0

mov a1, 100

div a1, id3

sub ax, a1

mov id2, ax

L:4

mov a0, 123

mul a0, s1

div a0, 111

mov ax, a0

mov a1, 433

mul a1, s2

mul a1, s3

div a1, s4

add ax, a1

add ax, 54

mov bx, 0

cmp ax, bx

je L5

jmp L6

L:5

mov ax, 4

add ax, 15

sub ax, 10

mov a0, id7

mul a0, 100

add ax, a0

mov asdf, ax

L:6

Были предусмотрены следующие диагностические сообщения:

1. При попытке задать какое-либо значение не переменной, а числу:

Входной файл:

if(4+15-10+var\*100){

5=4+id2;

}

else{

id1=id3;

}

Результат работы программы:

https://sun9-63.userapi.com/c854524/v854524953/170169/OAf1yyruLo4.jpg

2. При пропуске условия после объявления оператора if:

Входной файл:

if(5-3/4\*asd){

id5=id2\*4;

}

if{

id=12\*5+4;

}

else{

asdf=4+15-10+id7\*100;

}

Результат работы программы:

https://sun9-10.userapi.com/c854524/v854524953/17018e/wI_am86Elrk.jpg

1. **Вывод:**

В результате выполнения данной работы были закреплены полученные навыки применения утилит yacc и lex для генерирования синтаксических анализаторов. Был реализован транслятор, соответствующий установленным требованиям.

Наиболее трудной частью работы мне показалась организация сохранения приоритетов при обработке арифметических выражений.