**ASSIGNMENT: 6**

**Problem Statement:**

Intermediate code generation using LEX &YACC for Control Flow and Switch Case statements.

**/\*(LexProgram :intar.l)\*/**

ALPHA [A-Za-z]

DIGIT [0-9]

%%

{ALPHA}({ALPHA}|{DIGIT})\* return ID;

{DIGIT}+ {yylval=atoi(yytext); return NUM;}

[\n\t] yyterminate();

. returnyytext[0];

%%

**/\*(YaccProgram :intar.y)\*/**

%token ID NUM

%right '='

%left '+' '-'

%left '\*' '/'

%left UMINUS

%%

S:ID{push();} '='{push();} E{codegen\_assign();}

;

E:E'+'{push();} T{codegen();}

|E'-'{push();} T{codegen();}

| T

;

T:T'\*'{push();} F{codegen();}

|T'/'{push();} F{codegen();}

|F

;

F:'(' E ')'

|'-'{push();} F{codegen\_umin();} %prec UMINUS

|ID{push();}

|NUM{push();}

;

%%

#include "lex.yy.c"

#include<ctype.h>

charst[100][10];

int top=0;

char i\_[2]="0";

char temp[2]="t";

main()

{

printf("Enter the expression : ");

yyparse();

}

push()

{

strcpy(st[++top],yytext);

}

codegen()

{

strcpy(temp,"t");

strcat(temp,i\_);

printf("%s = %s %s %s\n",temp,st[top-2],st[top-1],st[top]);

top-=2;

strcpy(st[top],temp);

i\_[0]++;

}

codegen\_umin()

{

strcpy(temp,"t");

strcat(temp,i\_);

printf("%s = -%s\n",temp,st[top]);

top--;

strcpy(st[top],temp);

i\_[0]++;

}

codegen\_assign()

{

printf("%s = %s\n",st[top-2],st[top]);

top-=2;

}

/\*

**Output:**

nn@linuxmint ~ $ lexintar.l

nn@linuxmint ~ $ yaccintar.y

nn@linuxmint ~ $ gccy.tab.c -ll -ly

nn@linuxmint ~ $ ./a.out

Enter the expression : a=(k+8)\*(c-s)

t0 = k + 8

t1 = c - s

t2 = t0 \* t1

a = t2

\*/

**/\*if condition lex program\*/**

ALPHA [A-Za-z]

DIGIT [0-9]

%%

if return IF;

then return THEN;

else return ELSE;

{ALPHA}({ALPHA}|{DIGIT})\* return ID;

{DIGIT}+ {yylval=atoi(yytext); return NUM;}

[ \t] ;

\n yyterminate();

. returnyytext[0];

%%

**/\*if condition yacc program\*/**

%token ID NUM IF THEN ELSE

%right '='

%left '+' '-'

%left '\*' '/'

%left UMINUS

%%

S : IF '(' E ')'{lab1();} THEN E ';'{lab2();} ELSE E ';'{lab3();}

;

E :V '='{push();} E{codegen\_assign();}

| E '+'{push();} E{codegen();}

| E '-'{push();} E{codegen();}

| E '\*'{push();} E{codegen();}

| E '/'{push();} E{codegen();}

| '(' E ')'

| '-'{push();} E{codegen\_umin();} %prec UMINUS

| V

| NUM{push();}

;

V : ID {push();}

;

%%

#include "lex.yy.c"

#include<ctype.h>

charst[100][10];

int top=0;

char i\_[2]="0";

char temp[2]="t";

int label[20];

intlnum=0;

intltop=0;

main()

{

printf("Enter the expression : ");

yyparse();

}

push()

{

strcpy(st[++top],yytext);

}

codegen()

{

strcpy(temp,"t");

strcat(temp,i\_);

printf("%s = %s %s %s\n",temp,st[top-2],st[top-1],st[top]);

top-=2;

strcpy(st[top],temp);

i\_[0]++;

}

codegen\_umin()

{

strcpy(temp,"t");

strcat(temp,i\_);

printf("%s = -%s\n",temp,st[top]);

top--;

strcpy(st[top],temp);

i\_[0]++;

}

codegen\_assign()

{

printf("%s = %s\n",st[top-2],st[top]);

top-=2;

}

lab1()

{

lnum++;

strcpy(temp,"t");

strcat(temp,i\_);

printf("%s = not %s\n",temp,st[top]);

printf("if %s gotoL%d\n",temp,lnum);

i\_[0]++;

label[++ltop]=lnum;

}

lab2()

{

int x;

lnum++;

x=label[ltop--];

printf("gotoL%d\n",lnum);

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

label[++ltop]=lnum;

}

lab3()

{

int y;

y=label[ltop--];

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

}

/\* **Output**

lab7@ubuntu:~/ajit if$ lexif.l

lab7@ubuntu:~/ajit if$ yaccif.y

lab7@ubuntu:~/ajit if$ gccy.tab.c -ll -ly

lab7@ubuntu:~/ajit if$ ./a.out

Enter the expression : if(k+8) then k=18;else c=s;

t0 = k + 8

t1 = not t0

if t1 goto L1

k = 18

goto L2

L1:

c = s

L2:

\*/

**/\*while condition lex program\*/**

ALPHA [A-Za-z]

DIGIT [0-9]

%%

while return WHILE;

{ALPHA}({ALPHA}|{DIGIT})\* return ID;

{DIGIT}+ {yylval=atoi(yytext); return NUM;}

[ \t] ;

\n yyterminate();

. returnyytext[0];

%%

**/\*while condition yacc program\*/**

%token ID NUM WHILE

%right '='

%left '+' '-'

%left '\*' '/'

%left UMINUS

%%

S : WHILE{lab1();} '(' E ')'{lab2();} E ';'{lab3();}

;

E :V '='{push();} E{codegen\_assign();}

| E '+'{push();} E{codegen();}

| E '-'{push();} E{codegen();}

| E '\*'{push();} E{codegen();}

| E '/'{push();} E{codegen();}

| '(' E ')'

| '-'{push();} E{codegen\_umin();} %prec UMINUS

| V

| NUM{push();}

;

V : ID {push();}

;

%%

#include "lex.yy.c"

#include<ctype.h>

charst[100][10];

int top=0;

char i\_[2]="0";

char temp[2]="t";

intlnum=1;

int start=1;

main()

{

printf("Enter the expression : ");

yyparse();

}

push()

{

strcpy(st[++top],yytext);

}

codegen()

{

strcpy(temp,"t");

strcat(temp,i\_);

printf("%s = %s %s %s\n",temp,st[top-2],st[top-1],st[top]);

top-=2;

strcpy(st[top],temp);

i\_[0]++;

}

codegen\_umin()

{

strcpy(temp,"t");

strcat(temp,i\_);

printf("%s = -%s\n",temp,st[top]);

top--;

strcpy(st[top],temp);

i\_[0]++;

}

codegen\_assign()

{

printf("%s = %s\n",st[top-2],st[top]);

top-=2;

}

lab1()

{

printf("L%d: \n",lnum++);

}

lab2()

{

strcpy(temp,"t");

strcat(temp,i\_);

printf("%s = not %s\n",temp,st[top]);

printf("if %s gotoL%d\n",temp,lnum);

i\_[0]++;

}

lab3()

{

printf("gotoL%d \n",start);

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

}

/\* **Output**

lab7@ubuntu:~/Desktop/ic/while final$ lexwhile.l

lab7@ubuntu:~/Desktop/ic/while final$ yaccwhile.y

lab7@ubuntu:~/Desktop/ic/while final$ gccy.tab.c -ll -ly

lab7@ubuntu:~/Desktop/ic/while final$ ./a.out

Enter the expression : while(k=c/s)k=k\*c+d;

L1:

t0 = c / s

k = t0

t1 = not k

if t1 goto L0

t2 = k \* c

t3 = t2 + d

k = t3

goto L1

L0:

\*/