Date : 29/03/2022

CSPC62 : COMPILER DESIGN
LAB-6

Roll no. : **106119100**

Name : **Rajneesh Pandey**

Section : **CSE-B**

Generate intermediate code for if, if-else and while loop in C.

Code: If-else

lexer.l

```
%{
#include "parser.tab.h"
#include <stdlib.h>
#include <string.h>
#include <math.h>
%}


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();
.                  return yytext[0];
%%
```

parser.y

```
%{
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int yylex();
void yyerror (char const *s) {
   fprintf (stderr, "%s\n", s);
 }
void push();
```

```
void pusha();
void pushab();
void abc();
void abcde();
void second();
void second1();
void first();
void third();
void codegen();
void codegen_umin();
void codegen_assigna();
void codegen_assign();
void codegen_assignb();
void lab();
void lab1();
void lab2();
void lab3();

%}

%token ID NUM IF THEN ELSE
%right '='
%left '+' '-'
%left '*' '/'
%left UMINUS
%%

S : IF '(' Y ')'{lab();} THEN '{' X '}'{lab2();} ELSE '{' X '}' {lab3();}
  ;
X : E ';'|X X;
Y : B {abc();codegen_assigna();first();}
  | B '&''&'{abc();codegen_assigna();second();} Y
  | B {abc();codegen_assigna();third();}'|''|' Y
  | '!'B{abcde();codegen_assigna();first();}
  ;

B : V '='{push();}'='{push();}D
```

```
  | V '>'{push();}F
  | V '<'{push();}F
  | V '!'{push();}'='{push();}D
  |'(' B ')'
  | V{pushab();}

;
F :'='{push();}D
  |D{pusha();}
;
D :NUM{push();}
  |ID{push();}
;
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();}
  | S
;
V : ID {push();}
  ;
%%

#include "lex.yy.c"
#include<ctype.h>
char st[100][10];
int top=0;
char i_[2]="0";
char temp[2]="t";
int abcd=0;
int label[20];
int lnum=0;
```

```
int ltop=0;
int i=0;
int main()
 {
 printf("Enter the expression : ");
 yyparse();
 }

int yywrap(){return(1);}

void pusha()
{
strcpy(st[++top]," " );
}
void pushab()
{
strcpy(st[++top]," ");
strcpy(st[++top]," ");
strcpy(st[++top]," ");
}
void push()
 {
  strcpy(st[++top],yytext);
 }

void abc()
{
abcd++;
printf("\nX%d : if ",abcd);
}
void abcde()
{
abcd++;
printf("\nX%d :not ",abcd);
}
void second1()
{
```

```
printf("\nif x%d true goto L%d\n",abcd,lnum);
printf("\nif x%d false goto L%d\n",abcd,++lnum);
lnum=lnum-1;
}
void second()
{
int xyz=0;
xyz=abcd+1;
printf("falg=true else flag=false");
printf("\n if flag(true) goto x%d",xyz);
printf("\n if flag(false) goto L1");
}
void first()
{
printf("flag=true else flag=false");
printf("\n if flag(true) goto  L0");
printf("\n if flag(false) goto L1");
}
void third()
{
int xyz=0;
xyz=abcd+1;
printf("flag=true else flag=false");
printf("\n if flag(true) goto L0 ");
printf("\n if flag(false) goto x%d",xyz);
}
void 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]++;
 }

void codegen_umin()
```

```
 {
 strcpy(temp,"t");
 strcat(temp,i_);
 printf("%s = -%s\n",temp,st[top]);
 top--;
 strcpy(st[top],temp);
 i_[0]++;
 }

void codegen_assigna()
{
printf("%s %s %s %s ",st[top-3],st[top-2],st[top-1],st[top]);
top-=3;
}
void codegen_assign()
 {
 printf("%s = %s\n",st[top-2],st[top]);
 top-=2;
 }
void codegen_assignb()
{
printf("%s %s %s ",st[top-3],st[top-2],st[top-1]);
top-=3;
}

void lab()
{
printf("\nL0 :\n");
}
void lab1()
{
 strcpy(temp,"t");
 strcat(temp,i_);
 printf("\n%s = not arguement \n",temp);
 printf("if %s goto L%d\n",temp,lnum);
 i_[0]++;
 label[++ltop]=lnum;
```

```
}

void lab2()
{
int x;
lnum++;
x=label[ltop--];
printf("goto L2\n");
printf("L%d: \n",++x);
label[++ltop]=lnum;
}

void lab3()
{
int y;
y=label[ltop--];
printf("L2: \n");
}
```

Input:

if(a>1){a=b+1;}else{b=a+1;}

Run :

```
rajne (main *) if-else
$ bison -d parser.y
parser.y: conflicts: 5 shift/reduce
rajne (main *) if-else
$ flex lexer.l
rajne (main *) if-else
$ gcc parser.tab.c -lm
rajne (main *) if-else
$ ./a.exe
```

Output :

```
rajne (main *) if-else
$ ./a.exe
Enter the expression : if(a>1){a=b+1;}else{b=a+1;}

X1 : if a > 1   flag=true else flag=false
 if flag(true) goto  L0
X1 : if a > 0   falg=true else flag=false
 if flag(true) goto x2
 if flag(false) goto L1
X2 : if b < 0   flag=true else flag=false
 if flag(true) goto L0
 if flag(false) goto x3
X3 : if a < 1   flag=true else flag=false
 if flag(true) goto  L0
 if flag(false) goto L1
L0 :
t0 = x + 1
a = t0
goto L2
L1:
t1 = x + 1
b = t1
L2:
```

Code: While

lexer.l

```
%{
#include"parser.tab.h"
%}
ALPHA [A-Za-z]
DIGIT [0-9]
%%
while                   return WHILE;
{ALPHA}({ALPHA}|{DIGIT})*    return ID;
{DIGIT}+                {yylval=atoi(yytext); return NUM;}
[ \t]                    ;
\n                  yyterminate();
.                   return yytext[0];
%%
```

parser.y

```
%{
#include "lex.yy.c"
#include<ctype.h>
#include <string.h>
int yylex();
void yyerror (char const *s) {
   fprintf (stderr, "%s\n", s);
 }
void push();
void codegen()  ;
void codegen_umin();
void codegen_assign();
void lab1();
void lab2();
void lab3();
%}

%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();}
  ;
%%

char st[100][10];
int top=0;
char i_[2]="0";
char temp[2]="t";

int lnum=1;
int start=1;
int main()
 {
 printf("Enter the expression : ");
 yyparse();
 }

int yywrap(){
  return(1);
```

```
}

void push()
 {
  strcpy(st[++top],yytext);
 }

void 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]++;
 }

void codegen_umin()
 {
 strcpy(temp,"t");
 strcat(temp,i_);
 printf("%s = -%s\n",temp,st[top]);
 top--;
 strcpy(st[top],temp);
 i_[0]++;
 }

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



void lab1()
{
```

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


void lab2()
{
 strcpy(temp,"t");
 strcat(temp,i_);
 printf("%s = not %s\n",temp,st[top]);
 printf("if %s goto L%d\n",temp,lnum);
 i_[0]++;
 }

void lab3()
{
printf("goto L%d \n",start);
printf("L%d: \n",lnum);
}
```

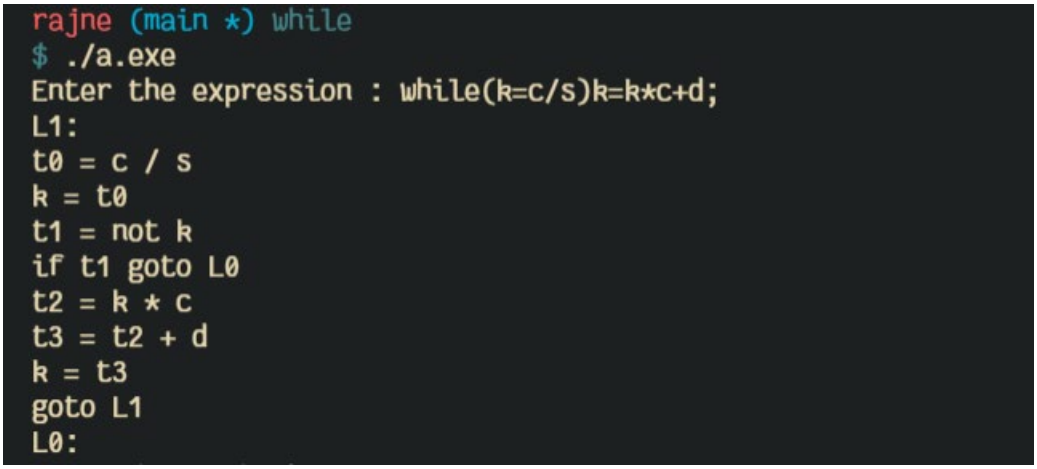
input

while(k=c/s)k=k*c+d;

run

```
rajne (main *) Lab6
$ cd while/
rajne (main *) while
$ bison -d parser.y
rajne (main *) while
$ flex lexer.l
rajne (main *) while
$ gcc parser.tab.c -lm
rajne (main *) while
$ ./a.exe
```

Output


rajne (main *) while
$ ./a.exe
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: