# **Laboratory-5**

## **Question**

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

### **Lex File (lex.l):**

%{

**typedef** **struct** B {

**int** t;

**int** f;

}B;

#include "y.tab.h"

%}

%option header-file="myLexHeader.h"

%%

"#include" **return** INCLUDE;

"<"[a-zA-Z]+\.h">" **return** INCL\_FILE;

"#define" {strcpy(yylval.string,yytext); **return** MACRO;}

(**int**|**char**|**float**) **return** TYPE;

"if" **return** IF;

"else" **return** ELSE;

"while" **return** WHILE;

"main" **return** MAIN;

"return" **return** RETURN;

(<|>|>=|<=|==) {strcpy(yylval.string,yytext); **return** LOGIC\_OPRTR;}

(\+=|\-=|\\*=|\/=) **return** OPRTR\_ASSGN;

(\+\+|\-\-) **return** UNARY;

([\_|a-z]+[0-9]\*)\* {strcpy(yylval.string,yytext); **return** VARIABLE;}

[0-9]\* {strcpy(yylval.string,yytext); **return** CONSTANT;}

[\n\r] {yylineno++;}

[-;+\*/=(),{}] {**return** yytext[0];}

[ \t]+ ;

. **return** ERROR;

%%

**int** yywrap(**void**)

{

**return** 1;

}

### **Yacc File (par.y):**

%{

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <stdbool.h>

**extern** **int** yylex();

**int** yyerror();

**int** temp\_var\_num = 0;

**int** label\_num=0;

**int** else\_label\_num = 0;

**bool** else\_flag=0;

**int** bool\_ir\_index=100;

**int** bool\_ir\_start = 100;

**int** bool\_ir\_if\_top=-1;

**int** bool\_ir\_if\_stack[10];

**int** loop\_stack\_top=-1;

**int** loop\_stack[10];

**char** bool\_code[25][50];

**struct** three\_address\_code {

**char** instr[4][20];

} \*tac\_temp,tac[50];

**int** tac\_len, tac\_temp\_len, tac\_temp\_tot = 1;

**struct** tac\_package {

**int** tac\_len;

**struct** three\_address\_code \*tac;

};

**void** addToThreeAddressArthmtc(**char**\* op, **char**\* arg1, **char**\* arg2, **char**\* result)

{

**if**(tac\_temp\_len >= tac\_temp\_tot-1)

{

tac\_temp = reallocarray(tac\_temp,**sizeof**(**struct** three\_address\_code), tac\_temp\_tot \*= 2);

}

strcpy(tac\_temp[tac\_temp\_len].instr[0],result);

strcpy(tac\_temp[tac\_temp\_len].instr[1],arg1);

strcpy(tac\_temp[tac\_temp\_len].instr[2],arg2);

strcpy(tac\_temp[tac\_temp\_len].instr[3],op);

tac\_temp\_len++;

}

**void** addToThreeAddressBrnch(**char** \*res)

{

**char** \*arg1 = "",\*arg2 = "",\*op = "",\*result = "";

**char** \*init = strtok(res," ");

**if**(res[0] == 'i')

{

arg1 = strtok(NULL," ");

op = strtok(NULL, " ");

arg2 = strtok(NULL, " ");

strtok(NULL, " ");

result = strtok(NULL, " ");

}

**else** **if**(res[0] == 'g')

{

op = init;

result = strtok(NULL, " ");

}

**else** result = strtok(init,":");

strcpy(tac[tac\_len].instr[0],result);

strcpy(tac[tac\_len].instr[1],arg1);

strcpy(tac[tac\_len].instr[2],arg2);

strcpy(tac[tac\_len].instr[3],op);

tac\_len++;

}

**void** generate\_code(**char**\* op, **char**\* arg1, **char**\* arg2, **char**\* result)

{

//printf("%s = %s %s %s\n", result, arg1, op, arg2);

addToThreeAddressArthmtc(op,arg1,arg2,result);

}

**void** backpatch(**int** \*list,**int** next\_ir) {

**char** addr[10];

sprintf(addr,"L%d ",next\_ir-100);

**int** i=0;

**while**(list[i]!=0) {

**char** label[25];

strcat(bool\_code[list[i]-100],addr);

**if**(bool\_code[next\_ir-100][0] != 'L') {

sprintf(label,"L%d: ",next\_ir-100);

strcat(label,bool\_code[next\_ir-100]);

strcpy(bool\_code[next\_ir-100],label);

}

i++;

}

}

%}

%token INCLUDE INCL\_FILE MACRO

%token TYPE IF ELSE VARIABLE CONSTANT

%token WHILE MAIN RETURN

%token LOGIC\_OPRTR OPRTR\_ASSGN UNARY OR AND

%token ERROR

%left '+' '-'

%left '\*' '/'

%left LOGIC\_OPRTR

%right '=' UMINUS

%%

program: program\_body

|

;

program\_body:

include program\_body

| main

;

include: INCLUDE INCL\_FILE

main: TYPE MAIN '(' ')' '{' body '}'

body:

body line

|

;

line: branch

{

**if**(else\_flag)

{

**char** label[5] = {0}, buff[5] = {0};

sprintf(buff,"%d",else\_label\_num);

label[0] = 'G';

strcat(label,buff);

strcpy(tac[tac\_len++].instr[0],label);

else\_flag=0;

else\_label\_num++;

}

}

| assignment ';'

%%

#include "myLexHeader.h"

**int** main(**void**)

{

**for**(**int** i=0;i<tac\_len;i++)

{

**char** \*res = tac[i].instr[0];

**char** \*arg1 = tac[i].instr[1];

**char** \*arg2 = tac[i].instr[2];

**char** \*op = tac[i].instr[3];

**if**(res[0] == 'L' || res[0] == 'G')

{

**if**(op[0] == 'g') printf("%s %s\n", op, res);

**else** **if**(op[0] == '\0') printf("%s:\n", res);

**else** printf("if %s %s %s goto %s\n", arg1, op, arg2, res);

}

**else**

printf("%s = %s %s %s\n", res,arg1,op,arg2);

printf("completed!\n");

**else** printf("failed!\n");}

}

**int** yyerror(**char** \*s)

{

printf("%d : %s %s\n",yylineno,s,yytext);

**return** 1;

}

**Sample.c:**

**int** main(){

**int** x = 4, count = 1;

**if**(x < 5) count = count\*2;

**else** count = count + 2;

**return** 0;

}

### **Output:**

A computer screen shot of a computer

Description automatically generated

## **Result:**

Intermediate code for conditional and looping constructs was generated successfully.