UCS 1602 - Compiler Design

Exercise 7 - Generate three address code for a simple program using LEX and YACC.

Name:	Mahesh Bharadwaj K
Reg No:	185001089
Semester:	VI
Date:	April 14, 2021

Aim:

To generate three address code for a simple program using LEX and YACC.

Program

Lexer file

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

    int debug=0;
%}
term ([a-zA-Z_][a-zA-Z0-9_]*|-?[0-9]+)
relop ("<"|"<="|">"|">="|"=="|"!=")
op ("+"|"-"|"*"|"/"|"%")
bool_op ("!"|"&&"|"||")
%%
";" {return EOS;}
"if" {return IF;}
"else" {return ELSE;}
"while" { return WHILE; }
"do" { return DO; }
"switch" { return SWITCH; }
"case" { return CASE; }
"default" { return DEFAULT; }
"break" { return BREAK; }
{bool_op} {yylval.str = strdup(yytext);return BOOL_OP;}
"=" {yylval.str = strdup(yytext); return ASSIGN_OP;}
{term} { yylval.str = strdup(yytext); return TERM; }
{relop} { yylval.str = strdup(yytext); return REL_OP; }
{op} { yylval.str = strdup(yytext); return ARITH_OP; }
[ \t\n]+ { }
. { return *yytext; }
%%
```

Yacc Program

```
%{
    #include <stdlib.h>
    #include <stdio.h>
    int yylex(void);
    extern FILE* yyin;
    #include "y.tab.h"
    int error = 0;
    /*extern int debug;*/
    int cc = 1, tc = 1, nc = 1, sc = 0;
%}
%token TERM ASSIGN_OP ARITH_OP REL_OP ID BOOL_OP EOS IF ELSE WHILE SWITCH CASE DEFAULT
↪  BREAK DO
%union
{
    int intval;
    float floatval;
    char *str;
}
%type<str> TERM REL_OP ARITH_OP ASSIGN_OP
%%
line: /* empty */
    | TERM ASSIGN_OP TERM ARITH_OP TERM EOS { printf("t%d := %s %s %s\n%s := t%d\n",
    ↪  tc, $3, $4, $5, $1, tc); tc++; } line
    | TERM ASSIGN_OP TERM REL_OP TERM EOS { printf("t%d := %s %s %s\n%s := t%d\n", tc,
    ↪  $3, $4, $5, $1, tc); tc++; } line
    | TERM ASSIGN_OP TERM EOS { printf("%s := %s\n", $1, $3); } line
    | TERM ASSIGN_OP '-' TERM EOS {printf("t%d := -%s\n", tc, $4); } line
    | while_block
    | switch_block

while_block: WHILE TERM REL_OP TERM DO '{' { printf("LABEL%d: if not %s %s %s then
↪  goto FALSE%d\nTRUE%d: ", cc, $2, $3, $4, cc, cc); } line '}' { printf("FALSE%d: ",
↪  cc); cc++; } line
    | WHILE TERM ARITH_OP TERM DO '{' { printf("LABEL%d: if not %s %s %s then goto
    ↪  FALSE%d\nTRUE%d: ", cc, $2, $3, $4, cc, cc); } line '}' { printf("FALSE%d: ",
    ↪  cc); cc++; } line
    | WHILE TERM DO '{' { printf("LABEL%d: if not %s then goto FALSE%d\nTRUE%d: ", cc,
    ↪  $2, cc, cc); } line '}' { printf("FALSE%d: ", cc); cc++; } line

switch_block: SWITCH '(' TERM REL_OP TERM ')' '{' { printf("t%d := %s %s %s\n", tc,
↪  $3, $4, $5); sc = tc; tc++; } cases_block '}' { printf("NEXT%d: ", cc); cc++; }
↪  line
    | SWITCH '(' TERM ARITH_OP TERM ')' '{' { printf("t%d := %s %s %s\n", tc, $3, $4,
    ↪  $5); sc = tc; tc++; } cases_block '}' { printf("NEXT%d: ", cc); cc++; } line
    | SWITCH '(' TERM ')' '{' { printf("t%d := %s\n", tc, $3); sc = tc; tc++; }
    ↪  cases_block '}' { printf("NEXT%d: ", cc); cc++; } line
    | BREAK EOS line { printf("goto NEXT%d\n", cc); }
cases_block: /* empty */
     | CASE TERM ':' { printf("CASE%d: if not t%d == %s goto CASE%d\n", nc, sc, $2, nc
     ↪  + 1); nc++; } line cases_block
     | DEFAULT { printf("CASE%d: ", nc); nc++; } ':' line { printf("goto NEXT%d\n",
     ↪  cc); } cases_block
%%
int yyerror(char* s)
{
  fprintf(stderr, "%s\n", s);
  return 0;
```

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

int main(int argc, char **argv){
    /*yydebug = 1;*/
    if(argc != 2){
        fprintf(stderr, "Enter file name as argument!\n");
        return 1;
    }
    yyin = fopen(argv[1], "rt");
    if (!yyin){
        fprintf(stderr, "File not found!\n");
        return 2;
    }
    yyparse();
    return 0;
}
```

Output

Figure 1: **Sample Input**

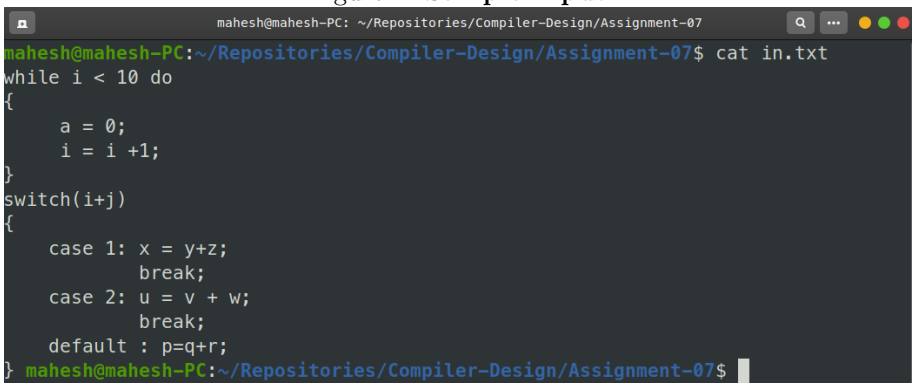
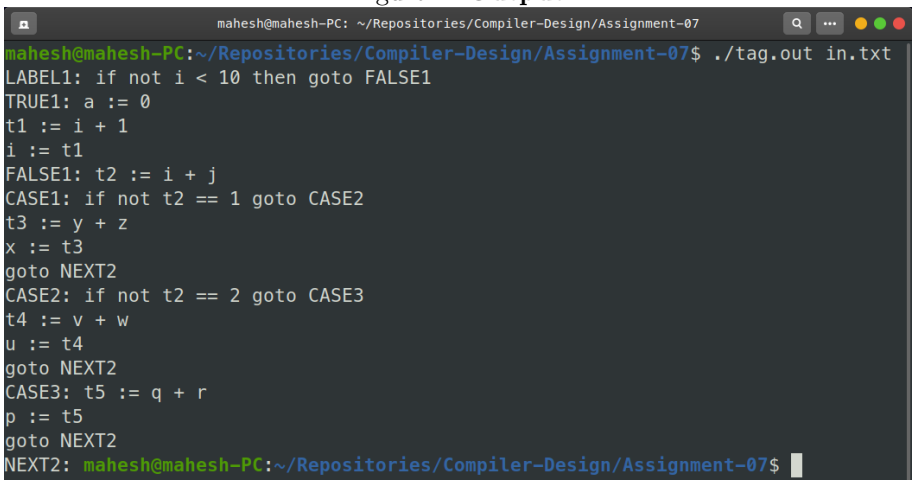


Figure 2: **Output**



Learning Outcomes

1. We learn to write grammar for TAC generation.
2. We learn to Write rules to parse tokens in grammar.
3. We learn to generate three address code using YACC and LEX tool.
