Exercise 6
Three Address Code Generation

Praveen Kumar R
312217104114

March 9, 2020

Aim

To generate three address code for while and switch statements in pascal language using lex and yacc tool.

Lex Code

```
%{
#include<stdio.h>
#include<string.h>
#include "y.tab.h"
%}
term ([a-zA-Z\_][a-zA-Z\_0-9]*|[0-9]+)
relop ("<"|"<="|">"|">="|"=="|"!=")
op ("+"|"-"|"*"|"/"|"%")
%%
"while" { return WHILE; }
"do" { return DO; }
"switch" { return SWITCH; }
"case" { return CASE; }
"default" { return DEFAULT; }
"break" { return BREAK; }
{term} { yylval.str = strdup(yytext); return TERM; }
{relop} { yylval.str = strdup(yytext); return RELOP; }
{op} { yylval.str = strdup(yytext); return OP; }
[ \t\n]+ { }
. { return *yytext; }
%%
```

YACC Code

```
%{
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
int yylex(void);
#include "y.tab.h"
extern FILE *yyin;
int cc = 1, tc = 1, sc = 0, currcase = 1;
%}
%token TERM RELOP OP WHILE DO SWITCH CASE DEFAULT BREAK
%union
{
    int intval;
    float floatval;
    char *str;
}
%type<str> TERM RELOP OP
%%
line: /* empty */
    | TERM '=' TERM OP TERM ';' { printf("t%d := %s %s %s\n%s := t%d\n", tc, $3, $4,
                                        $5, $1, tc);
                                tc++; }
                            line
    | TERM '=' TERM RELOP TERM ';' { printf("t%d := %s %s %s\n%s := t%d\n", tc, $3,
                                        $4, $5, $1, tc);
                                    tc++; }
                            line
```

```
    | TERM '=' TERM ';' { printf("%s := %s\n", $1, $3); }
                  line
    | WHILE TERM RELOP TERM DO '{' { printf("LABEL%d: if not %s %s %s goto FALSE%d
                                        \nTRUE%d: ", cc, $2, $3, $4, cc, cc); }
                        line '}' { printf("FALSE%d: ", cc); cc++; }
                        line
    | WHILE TERM OP TERM DO '{' { printf("LABEL%d: if not %s %s %s  goto FALSE%d\n
                                        TRUE%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 '(' TERM RELOP TERM ')' '{' { printf("t%d := %s %s %s\n", tc, $3, $4, $5);
                                        sc = tc;
                                        tc++; }
                            cases '}' { printf("NEXT%d: ", cc); cc++; }
                            line
    | SWITCH '(' TERM OP TERM ')' '{' { printf("t%d := %s %s %s\n", tc, $3, $4, $5);
                                      sc = tc;
                                      tc++; }
                          cases '}' { printf("NEXT%d: ", cc); cc++; }
                          line
    | SWITCH '(' TERM ')' '{' { printf("t%d := %s\n", tc, $3); sc = tc; tc++; }
                  cases '}' { printf("NEXT%d: ", cc); cc++; }
                  line
    | BREAK ';' line { printf("goto NEXT%d\n", cc); }
cases: /* empty */
     | CASE TERM ':' { printf("CASE%d: if t%d != %s goto CASE%d\n ",
                                    currcase, sc, $2,currcase+1); currcase++; }
       line cases
     | DEFAULT{printf("CASE%d: ",currcase);} ':' line { printf("goto NEXT%d\n", cc); }
       cases
%%
int yyerror(char* s)
{
  fprintf(stderr, "%s\n", s);
  return 0;
}
int yywrap()
{
  return 1;
}
int main()
{
  char inputFile[100];
  printf("Enter the input file: ");
  scanf("%s",inputFile);
  yyin = fopen(inputFile, "r");
  yyparse();
  printf("\n");
  return 0;
}
```

Sample Input & Output 1

input.in

```
while i < 10 do
{
    a =0;
    i = i +1;
}

switch(i + j) {
 case 1: x = y + z; break;
 case 2: u = v + w; break;
 default: p = q + r;
}
a = 5;
```

```
praveen@praveen/CompilerDesign/LALRparser:~$ lex TAC.l
praveen@praveen/CompilerDesign/LALRparser:~$ yacc -d TAC.y
praveen@praveen/CompilerDesign/LALRparser:~$ gcc y.tab.c lex.yy.c
praveen@praveen/CompilerDesign/LALRparser:~$ ./a.out
```

```
Enter the input file: input.in
LABEL1: if not i < 10 goto FALSE1
TRUE1: a := 0
       t1 := i + 1
       i := t1
FALSE1: t2 := i + j
CASE1: if t2 != 1 goto CASE2
       t3 := y + z
       x := t3
       goto NEXT2
CASE2: if t2 != 2 goto CASE3
       t4 := v + w
       u := t4
       goto NEXT2
CASE3: t5 := q + r
       p := t5
       goto NEXT2
NEXT2: a := 5
```