S. B. JAIN INSTITUTE OF TECHNOLOGY, MANAGEMENT & RESEARCH, NAGPUR.

Practical No. 6

AIM: Construct a Program to generate Intermediate code using three address statements for logical expression.

Name of Student: Shrutika Pradeep Bagdi

Roll No.: CS22130

Semester/Year: 6th Semester/3rd Year

Academic Session: 2024-25

Date of Performance:

Date of Submission:

AIM: Construct a Program to generate Intermediate statements using three address codes for logical expressions.

OBJECTIVE / EXPECTED LEARNING OUTCOME:

The objectives and expected learning outcome of this practical are:

- To demonstrate how to convert logical expression into TAC (Three Address Code) using SDTS in YACC.

HARDWARE AND SOFTWARE REQUIRMENTS:

Hardware Requirement:

- Processor: Dual Core
- RAM: 1GB
- Hard Disk Drive: > 80 GB

THEORY:

1) SDTS scheme for generation of three address statements for logical expression.

2) Use of Backpatch and Merge routines

3) Data structures required to generate Three address code

4) Role of NEXTQUAD

5) Use of GENCODE function

ALGORITHM / PROCEDURE:

CODE:

Lex Code:

csc15@linux-p2-1272il: ~/CS22130

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

%option noyywrap

%%

[ \t\n]         { /* Ignore whitespace */ }
[0-9]+          { yylval.str = strdup(yytext); return NUM; }
[a-zA-Z][a-zA-Z0-9]* { yylval.str = strdup(yytext); return ID; }
"="             { return ASSIGN; }
"+"             { return PLUS; }
"-"             { return MINUS; }
"*"             { return MUL; }
"/"             { return DIV; }
";"             { return SEMI; }
"("             { return LPAREN; }
")"             { return RPAREN; }
.               { printf("Unexpected character: %s\n", yytext); exit(1); }

%%
```

Yacc Code:

csc15@linux-p2-1272il: ~/CS22130

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

int temp_var_count = 1; // Counter for temporary variables

void generate_code(char* result, char* op1, char* op, char* op2) {
    printf("%s = %s %s %s\n", result, op1, op, op2);
}

void generate_assignment(char* id, char* value) {
    printf("%s = %s\n", id, value);
}

// Explicit declaration of yyerror to avoid implicit declaration warning
void yyerror(const char* msg);

%}

%union {
    char* str;
}

%token <str> ID NUM
%token PLUS MINUS MUL DIV ASSIGN SEMI LPAREN RPAREN
%left PLUS MINUS
%left MUL DIV
%type <str> expr stmt

%%

stmt : ID ASSIGN expr SEMI {
            generate_assignment($1, $3);
            free($1);
            free($3);
        }
     ;
```

\

```
expr : expr PLUS expr {
            $$ = (char*) malloc(10);
            sprintf($$, "t%d", temp_var_count++);
            generate_code($$, $1, "+", $3);
            free($1);
            free($3);
        }
      | expr MINUS expr {
            $$ = (char*) malloc(10);
            sprintf($$, "t%d", temp_var_count++);
            generate_code($$, $1, "-", $3);
            free($1);
            free($3);
        }
      | expr MUL expr {
            $$ = (char*) malloc(10);
            sprintf($$, "t%d", temp_var_count++);
            generate_code($$, $1, "*", $3);
            free($1);
            free($3);
        }
      | expr DIV expr {
            $$ = (char*) malloc(10);
            sprintf($$, "t%d", temp_var_count++);
            generate_code($$, $1, "/", $3);
            free($1);
            free($3);
        }
      | ID { $$ = strdup($1); }
      | NUM { $$ = strdup($1); }
      | LPAREN expr RPAREN { $$ = $2; }
      ;

%%
```

```
int main() {
    printf("Enter an expression (e.g., a = b + c * d;):\n");
    yyparse();
    return 0;
}

void yyerror(const char* msg) {
    fprintf(stderr, "Syntax Error: %s\n", msg);
}
```

OUTPUT:

```
csc15@linux-p2-1272il:~/CS22130$ vi Practical6.l
csc15@linux-p2-1272il:~/CS22130$ vi Practical6.y
csc15@linux-p2-1272il:~/CS22130$ yacc -d Practical6.y
csc15@linux-p2-1272il:~/CS22130$ flex Practical6.l
csc15@linux-p2-1272il:~/CS22130$ cc lex.yy.c y.tab.c
csc15@linux-p2-1272il:~/CS22130$ ./a.out
Enter an expression (e.g., a = b + c * d;):
a=b+c*d;
t1 = c * d
t2 = b + t1
a = t2
```

CONCLUSION:

DISCUSSION AND VIVA VOCE:

- **Q1:** What is Intermediate Code?
- **Q2:** What is the need of Intermediate code?
- **Q3:** What are the types of Intermediate code?
- **Q4:** Why Three Address Code (TAC) preferred to use as Intermediate Code?
- **Q5:** Which are the optional phases of Compiler?

REFERENCE:

- **Book:** Alfred V. Aho, Monica S. Lam, Ravi Sethi, Jeffrey D. Ullman, "Compilers Principles, Techniques and Tools", Pearson Education, 2nd edition. 2010.
- **Book:** Compiler Design by O.G. Kakde, Laxmi Publications, 2006.
- Lab Manual of Compiler Design (Institute of Aeronautical Engineering, Dundigal, Hyderabad).
- Language Processors Lab Manual (MIT, Manipal).