GOA COLLEGE OF ENGINEERING

"Bhausaheb Bandodkar Technical Education Complex"

Experiment No: 9

Intermediate Code Generator

Aim: Write a YACC program to implement Intermediate Code generator for given input.

Theory:

In the analysis-synthesis model of a compiler, the front end of a compiler translates a source program into an independent intermediate code, then the back end of the compiler uses this intermediate code to generate the target code (which can be understood by the machine).

The benefits of using machine independent intermediate code are:

- Because of the machine independent intermediate code, portability will be enhanced.For ex, suppose, if a compiler translates the source language to its target machine language without having the option for generating intermediate code, then for each new machine, a full native compiler is required. Because, obviously, there were some modifications in the compiler itself according to the machine specifications.
- Retargeting is facilitated
- It is easier to apply source code modification to improve the performance of source code by optimising the intermediate code.

Lex Program:

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

%%
if {return IF;}
[\t ]+
[-+*(){}=;\n] {return *yytext;}
[a-zA-Z0-9]+ {yylval.string_value = strdup(yytext );return ID;}
">"|"<"|">="|"<="|"!="|"==" {yylval.string_value = strdup(yytext );return OP;}
%%
```

Yacc Program:

```
%{
#include<stdio.h>
#include<string.h>
void yyerror(char*);
void Gen(char*);
char temp[50];
int i=0;
int j=0;
%}
```

```
%union
{
char *string_value;
}

%type <string_value> EXP
%type <string_value> COND
%type <string_value> START
%type <string_value> STMT
%token <string_value> ID
%token <string_value> OP
%token IF

%%
START :
        IF '(' COND ')' '{' STMT '}' { sprintf(temp,"\nl%d=if(%s){%s}",i,$3,$6); $$="t";i++;Gen(temp); }
        ;
STMT :
        START
        |ID '=' EXP ';' STMT { sprintf(temp,"\nt%d=%s=%s",j,$1,$3); sprintf($$, "t%d", j); j++; Gen(temp); }
        |
        ;
EXP :
        EXP '+' EXP { sprintf(temp,"\nt%d=%s+%s",j,$1,$3); sprintf($$, "t%d", j); j++; Gen(temp); }
        | EXP '-' EXP { sprintf(temp,"\nt%d=%s-%s",j,$1,$3); sprintf($$, "t%d", j); j++; Gen(temp); }
        | EXP '*' EXP { sprintf(temp,"\nt%d=%s*%s",j,$1,$3); sprintf($$, "t%d", j); j++; Gen(temp); }
        | ID { $$=$1; }
        ;
COND :
        ID OP ID { sprintf(temp,"\nt%d=%s%s%s",j,$1,$2,$3);sprintf($$, "t%d", j); j++; Gen(temp); }
        ;
%%

void Gen(char *val)
{
FILE *f;
f=fopen("output.txt","a");
fputs(val,f);
fclose(f);
}

int yywrap() { return 1; }

void yyerror(char *s)
```

```
{
return;
}

int main( int argc, char **argv ) {
yyparse();
return 1;
}
```

Conclusion:

The yacc program to implement intermediate code generation has been successfully executed.