GOA COLLEGE OF ENGINEERING

"Bhausaheb Bandodkar Technical Education Complex"

Experiment No: 10

Simple Code Generator

Aim: Write a YACC program to implement Simple Code Generator for given input.

Theory:

Code generation can be considered as the final phase of compilation. Through post code generation, optimization process can be applied on the code, but that can be seen as a part of code generation phase itself. The code generated by the compiler is an object code of some lower-level programming language, for example, assembly language. We have seen that the source code written in a higher-level language is transformed into a lower-level language that results in a lower-level object code, which should have the following minimum properties:

- It should carry the exact meaning of the source code.
- It should be efficient in terms of CPU usage and memory management.

Lex Program:

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

%%
[\t ]+ /* ignore the blank spaces */ ;
[-+*=\n,;(){}] { return *yytext; }
[a-zA-Z]+ { // return valid tokens to yacc program
yylval.string_value = strdup(yytext );
return ID;
};
[0-9]+ { // return valid tokens to yacc program
yylval.string_value = strdup(yytext );
return INTEGER;
};
%%
```

Yacc Program:

```
%{
#include<stdio.h>
#include<string.h>
void yyerror(char*);
void Gen(char*);
int j=0,l=0;
char temp[50];
extern FILE *yyin;
```

```
%}

%union {
   char *string_value;
}

%type <string_value> EXP START
%token <string_value> ID
%token <string_value> INTEGER

%%
STMT :
      STMT START
      |
      ;
START :
      ID '=' EXP { sprintf(temp,"\n%s=%s",$1,$3); $$="t"; Gen(temp); sprintf(temp,"\tmov %s,%s",$1,$3); Gen(temp); }
      ;
EXP :
      EXP '+' EXP {sprintf(temp,"\nt%d=%s+%s",j,$1,$3); Gen(temp);sprintf(temp,"\t mov R%d,%s | add R%d,%s | mov t
%d,R%d",l,$1,l,$3,j,l);sprintf($$, "t%d", j);l++;j++;Gen(temp);}
      | EXP '-' EXP {sprintf(temp,"\nt%d=%s-%s",j,$1,$3); Gen(temp);sprintf(temp,"\tmov R%d,%s | sub R%d,%s | mov t
%d,R%d",l,$1,l,$3,j,l);sprintf($$, "t%d", j);l++;j++;Gen(temp);}
      | EXP '*' EXP {sprintf(temp,"\nt%d=%s*%s",j,$1,$3); Gen(temp);sprintf(temp,"\tmov R%d,%s | mul R%d,%s | mov t
%d,R%d",l,$1,l,$3,j,l);sprintf($$, "t%d", j);l++;j++;Gen(temp);}
      | ID { $$=$1; }
      | INTEGER { $$=$1; }
      ;
%%

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

int yywrap() { return 1; }

void yyerror(char *s)
{
}
```

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

Conclusion:

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