Experiment 8

Intermediate Code Generation

YACC Code:

```
%{
#include"icg.tab.h"
#include<stdio.h>
char addtotable(char,char,char);
int index1=0;
char temp = 'A'-1;
struct expr{
char operand1;
char operand2;
char operator;
char result;
};
%}
%union{
char symbol;
}

%left '+' '-'
%left '/' '*'
%token <symbol> LETTER NUMBER
%type <symbol> exp

%%
statement: LETTER '=' exp ';'
{addtotable((char)$1,(char)$3,'=');};
exp: exp '+' exp {$$ =
addtotable((char)$1,(char)$3,'+');}
   |exp '-' exp {$$ =
addtotable((char)$1,(char)$3,'-');}
   |exp '/' exp {$$ =
addtotable((char)$1,(char)$3,'/');}
   |exp '*' exp {$$ =
addtotable((char)$1,(char)$3,'*');}
   |'(' exp ')' {$$= (char)$2;}
   |NUMBER {$$ = (char)$1;}
   |LETTER {(char)$1;};
%%

struct expr arr[20];
void yyerror(char *s){
   printf("Errror %s",s);
}

char addtotable(char a, char b, char o){
   temp++;
   arr[index1].operand1 =a;
   arr[index1].operand2 = b;
   arr[index1].operator = o;
   arr[index1].result=temp;
   index1++;
   return temp;
}

void threeAdd(){

   int i=0;
   char temp='A';
   while(i<index1){
      printf("%c:=\t",arr[i].result);
      printf("%c\t",arr[i].operand1);
      printf("%c\t",arr[i].operator);
      printf("%c\t",arr[i].operand2);
      i++;
      temp++;
      printf("\n");
   }
}

void fouradd(){
   int i=0;
   char temp='A';
   while(i<index1){
```

```
        printf("%c\t",arr[i].operator);
        printf("%c\t",arr[i].operand1);
        printf("%c\t",arr[i].operand2);
        printf("%c",arr[i].result);
        i++;
        temp++;
        printf("\n");
    }

}

int find(char l){
    int i;
    for(i=0;i<index1;i++)
        if(arr[i].result==l) break;
    return i;
}

void triple(){
    int i=0;
    char temp='A';
    while(i<index1){
        printf("%c\t",arr[i].operator);
        if(!isupper(arr[i].operand1))
        printf("%c\t",arr[i].operand1);
        else{
            printf("pointer");
            printf("%d\t",find(arr[i].operand1));
        }
        if(!isupper(arr[i].operand2))
        printf("%c\t",arr[i].operand2);
        else{
            printf("pointer");
            printf("%d\t",find(arr[i].operand2));
        }
        i++;
        temp++;
        printf("\n");
    }
}

int yywrap(){
    return 1;
}

int main(){
    printf("Enter the expression: ");
    yyparse();
    threeAdd();
    printf("\n");
    fouradd();
    printf("\n");
    triple();
    return 0;
}
```

Lex Code:

```
%{
#include "icg.tab.h"
extern char yyval;
%}

%%

[0-9]+
{yylval.symbol=(char)(yytext[0]);return
NUMBER;}
[a-z] {yylval.symbol=
(char)(yytext[0]);return LETTER;}
. {return yytext[0];}
\n {return 0;}
%%
```

```
C:\Users\HP\Desktop\YACC>bison -d icg.y

C:\Users\HP\Desktop\YACC>flex icg.l

C:\Users\HP\Desktop\YACC>gcc lex.yy.c icg.tab.c
icg.tab.c: In function 'yyparse':
icg.tab.c:617:16: warning: implicit declaration of function 'yylex' [-Wimplicit-function-declaration]
 # define YYLEX yylex ()
                ^~~~~~
icg.tab.c:1262:16: note: in expansion of macro 'YYLEX'
       yychar = YYLEX;
                ^~~~~
icg.tab.c:1432:7: warning: implicit declaration of function 'yyerror'; did you mean 'yyerrok'? [-Wimplicit-function-declaration]
       yyerror (YY_("syntax error"));
       ^~~~~~~~
       yyerrok
icg.y: At top level:
icg.y:50:6: warning: conflicting types for 'yyerror'
 void yyerror(char *s){
      ^~~~~~~~
icg.tab.c:1432:7: note: previous implicit declaration of 'yyerror' was here
       yyerror (YY_("syntax error"));
       ^~~~~~~~
icg.y: In function 'triple':
icg.y:108:13: warning: implicit declaration of function 'isupper' [-Wimplicit-function-declaration]
         if(!isupper(arr[i].operand1))
             ^~~~~~~

C:\Users\HP\Desktop\YACC>a.exe
Enter the expression: a=b*c+d
Errror syntax errorA:=  b       *       c
B:=     A       +       d

*       b       c       A
+       A       d       B

*       b       c
+       pointer0        d
```