Practical 6

Name: Rajatkumar Patel
Roll No.: 18BCE191

Aim

To generate Three Address code for the assignment statement.

Code

Input

Practical-6.l file

```
%{

#include <stdio.h>
#include <stdlib.h>
#include "y.tab.h"

%}

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

. {return yytext[0];}
%%

yywrap(){
  return 1;
}
```

Practical-6.y file

```
%{

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

void convertToThreeAddressCode();
char addToTable(char,char,char);
int i = 0;
char tmp='1';
struct exp{
  char op1,op2,op;
};

%}

%union
{
  char symbol;
}

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

stmt: LETTER '=' expr ';' {addToTable($1,'=',$3);}
       |expr ';'
       ;

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

%%

yyerror(char *s){
       printf("%s",s);
       exit(0);
```

```
}

struct exp code[20];

char addToTable(char op1,char op,char op2){
	code[i].op1=op1;
	code[i].op=op;
	code[i].op2=op2;

	i++;
	return tmp++;
}

void convertToThreeAddressCode(){
	printf("\n\n\t\tTHREE ADDRESS CODE \n\n");
	int cnt=0;

	char tmp='1';
	while(cnt < i){
		if(code[cnt].op != '=')
			printf("\tt%c : = \t",tmp++);

		if(isalpha(code[cnt].op1))
			printf("\t%c\t",code[cnt].op1);
		else if(code[cnt].op1 >='1' && code[cnt].op1 <='9')
			printf("\tt%c\t",code[cnt].op1);

		printf("%c",code[cnt].op);

		if(isalpha(code[cnt].op2))
			printf("\t%c\n",code[cnt].op2);
		else if(code[cnt].op2 >='1' && code[cnt].op2 <='9')
			printf("\tt%c\n",code[cnt].op2);

		cnt++;
	}
}

main(){
	printf("\nEnter the expression \n");
	yyparse();
	convertToThreeAddressCode();

}
```

Output

```
(base) rajat@rajat-VivoBook-S14-X430UA:/Rajat1/Books/Compiler Construction/Practicals/Practical6$ flex Practical-6.l
(base) rajat@rajat-VivoBook-S14-X430UA:/Rajat1/Books/Compiler Construction/Practicals/Practical6$ yacc Practical-6.y -d
(base) rajat@rajat-VivoBook-S14-X430UA:/Rajat1/Books/Compiler Construction/Practicals/Practical6$ gcc lex.yy.c y.tab.c -w
(base) rajat@rajat-VivoBook-S14-X430UA:/Rajat1/Books/Compiler Construction/Practicals/Practical6$ ./a.out

Enter the expression
x=a+(b/c*(d+e));


                THREE ADDRESS CODE

        t1 : =          b       /       c
        t2 : =          d       +       e
        t3 : =          t1      *       t2
        t4 : =          a       +       t3
        x       =       t4
(base) rajat@rajat-VivoBook-S14-X430UA:/Rajat1/Books/Compiler Construction/Practicals/Practical6$ _
```

Conclusion

Implemented algorithm to convert the expression to three address code. Learned about how expressions are converted to three address code which is further used for conversion to assembly code.