Don Bosco Institute of Technology, Kulra(W) Department of Computer Engineering

CSL601: System Programming and Compiler Construction Lab2022-23

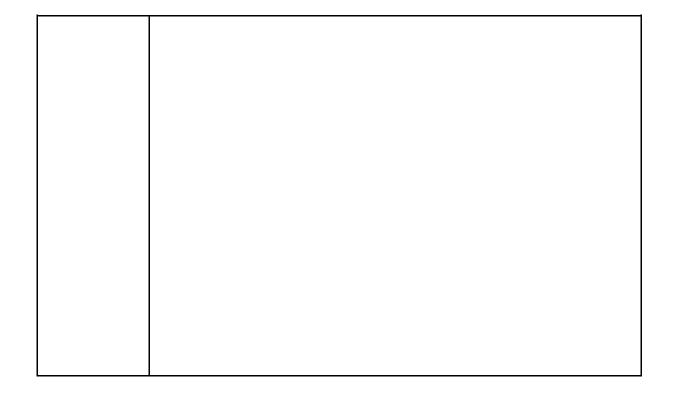
Experiment No.:	07
Experiment Title:	Intermediate code Generator Task:
	Implement Intermediate Code Generation and convert infix expression to postfix expression.(Yacc parser)
Student Name	Siddhanth Naidu
Roll No.	43
Objectives :	To understand parser generator tool : YACC and Intermediate code generation phase of compiler

Theory /Algorithm :

Intermediate code era may be a stage within the compiler plan handle where the compiler takes the source code and changes over it into an halfway representation that's more appropriate for advance investigation and optimization. The prepare of middle of the road code generation involves several steps, counting lexical examination, language structure examination, and semantic examination.

After the AST is generated, the compiler can begin the middle code era stage. This stage includes changing over the AST into a more machine-independent representation, regularly utilizing three-address code or quadruples. The intermediate code era stage includes a few errands, counting expression assessment, control-flow investigation, and code optimization. Expression assessment includes changing over expressions into three-address code or quadruples.

Code optimization includes changing the middle of the road code to make strides execution or decrease code size. Once the middle code is produced, it can be assist optimized utilizing strategies such as consistent collapsing, quality reduction, and loop optimization. In conclusion, middle code era is an fundamental stage within the compiler plan prepare. It includes changing over the AST into a more machine-independent representation utilizing three-address code or quadruples. The middle code can at that point be assist optimized to progress execution or diminish code estimate, making it simpler for the compiler to create productive machine code.



```
Program
Code
              scan.l
              % {
              #include"y.tab.h" extern
              int yylval;
              % }
              %%
              [0-9]+ {yylval=atoi(yytext); return NUMBER;}
              \n
                      return 0;
                            *yytext;
              [\t];
              . return
              %%
              int yywrap(){
                  return 1;
              }
              postfix.y 1>
              % {
              #include<stdio
              % }
              %token
              NUMBER
```

```
%left '+''-'
%left '*''/'
%righ NEGATIVE

%%
S:

E
{printf("\n");}
;
```

```
E:
                     E '+' E {printf("+");}
                         E '*' E {printf("*");}
                         E '-' E {printf("-");}
                         E '/' E {printf("/");}
                         '(' E ')'
                         '-' E %prec NEGATIVE {printf("-");}
                       NUMBER
                                    {printf("%d", yylval);}
               %%
               int main(){ printf("\nEnter infix expression
                   => "); yyparse();
               }
               int yyerror (char *msg) { return printf
                   ("Error: %s\n", msg);
               }
Input to the
                 1. 1+2/7
Program:
                 2. 9 - 4 * 3 /7 + 5
                 3. 12 + 3
                 4. 12 - 3
                 5. 1++2
```

```
Output of the [15:34] 

Bash
                                                                        84ms
               ~/Projects/Sem6/spcc/exp7
program:
               ) ./a.out
               Enter infix expression => 1+2 / 7
               127/+
               [15:35] 🗷 Bash
                                                                   15s 123ms
               ~/Projects/Semó/spcc/exp7
               [15:35] 🗷 Bash
                                                                       107ms
               ~/Projects/Sem6/spcc/exp7
               ) ./a.out
               Enter infix expression \Rightarrow 9 - 4 * 3 /7 + 5
               943*7/-5+
               [15:36] 🗷 Bash
                                                                   37s 867ms
               ~/Projects/Sem6/spcc/exp7
               [15:36] 🗷 Bash
                                                                        94ms
               ~/Projects/Sem6/spcc/exp7
               ) ./a.out
               Enter infix expression => 1 2 + 3
               Error: syntax error
               [15:36] 🗷 Bash
                                                                    6s 442ms
               ~/Projects/Sem6/spcc/exp7
               [15:37] B Bash
                                                                        87ms
               ~/Projects/Sem6/spcc/exp7
               ) ./a.out
               Enter infix expression => 12 - 3
               123-
               [15:37] B Bash
                                                                    3s 923ms
               ~/Projects/Semó/spcc/exp7
               [15:38] Bash
                                                                        83ms
               ~/Projects/Sem6/spcc/exp7
               ) ./a.out
               Enter infix expression => 1 + + 2
               1Error: syntax error
               [15:39] 🖪 Bash
                                                                    8s 663ms
               ~/Projects/Semó/spcc/exp7
Outcome of
               In this experiment these two tools were used to create a
               compiler to convert arithmetic expressions written in human
the
Experiment:
               readable infix form to These expressions should also be valid
               for any programming language.
```

	In this experiment these two tools were used to create a compiler to convert arithmetic expressions written in human readable infix form to These expressions should also be valid for any programming language.
Reference:	<u>class notes</u>

Course in-charge-Mayura Gavhane