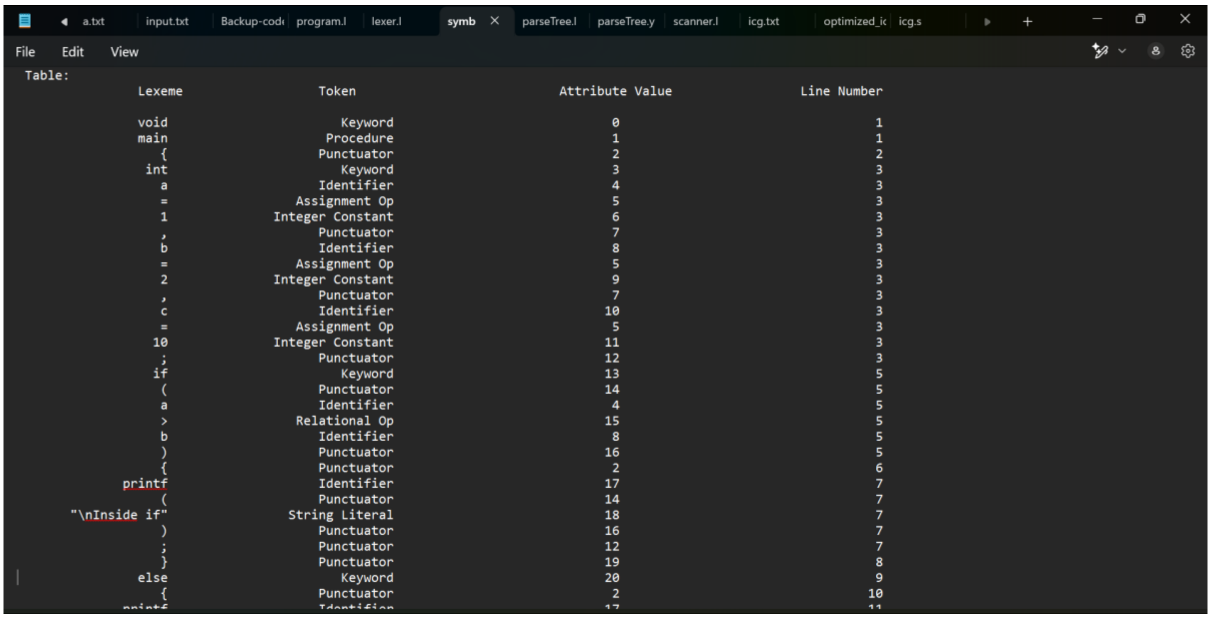
C Compiler All Phases

1. Lexical Analyzer

```
flex lexAnalyzer.l
gcc lex.yy.c
a.exe < testCases/ifelse.c
```

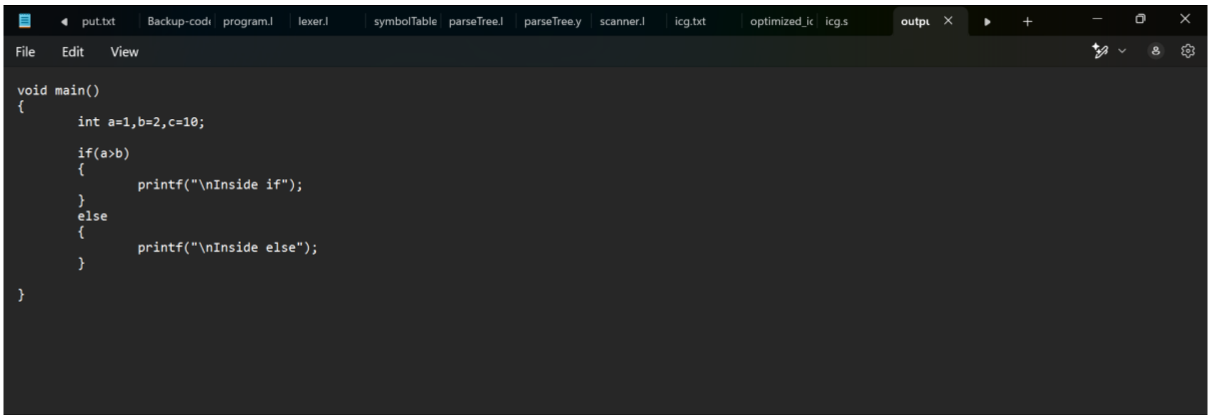
The Symbol Table will be generated in symbolTable.txt file. There is also a lex file to remove comments from a given .c file



```
 Table:
                Lexeme                  Token                          Attribute Value                 Line Number

                void                  Keyword                                0                               1
                main                Procedure                                1                               1
                   {                Punctuator                               2                               2
                 int                  Keyword                                3                               3
                   a               Identifier                                4                               3
                   =            Assignment Op                                5                               3
                   1         Integer Constant                                6                               3
                   ,               Punctuator                                7                               3
                   b               Identifier                                8                               3
                   =            Assignment Op                                5                               3
                   2         Integer Constant                                9                               3
                   ,               Punctuator                                7                               3
                   c               Identifier                               10                               3
                   =            Assignment Op                                5                               3
                  10         Integer Constant                               11                               3
                   ;               Punctuator                               12                               3
                  if                  Keyword                               13                               5
                   (               Punctuator                               14                               5
                   a               Identifier                                4                               5
                   >            Relational Op                               15                               5
                   b               Identifier                                8                               5
                   )               Punctuator                               16                               5
                   {               Punctuator                                2                               6
              printf               Identifier                               17                               7
                   (               Punctuator                               14                               7
      "\nInside if"            String Literal                               18                               7
                   )               Punctuator                               16                               7
                   ;               Punctuator                               12                               7
                   }               Punctuator                               19                               8
                else                  Keyword                               20                               9
                   {               Punctuator                                2                              10
```

```
flex commentRemover.l
gcc lex.yy.c
a.exe < TestCases/ifelse.c
```

A new file called output.c will be created.

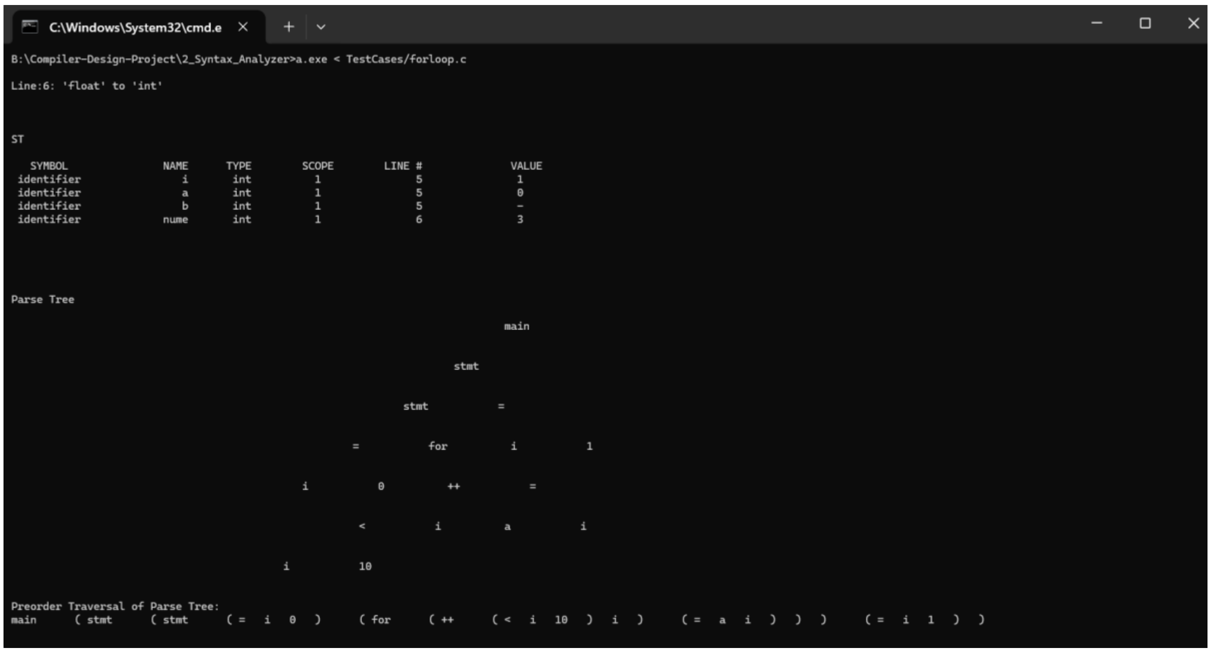


```
void main()
{
        int a=1,b=2,c=10;

        if(a>b)
        {
                printf("\nInside if");
        }
        else
        {
                printf("\nInside else");
        }

}
```

2. Syntax Analyzer

```
flex parseTree.l
yacc -d parseTree.y
gcc lex.yy.c y.tab.c
a.exe < TestCases/forloop.c
```

Parse tree will be printed in the terminal with its preorder traversal.

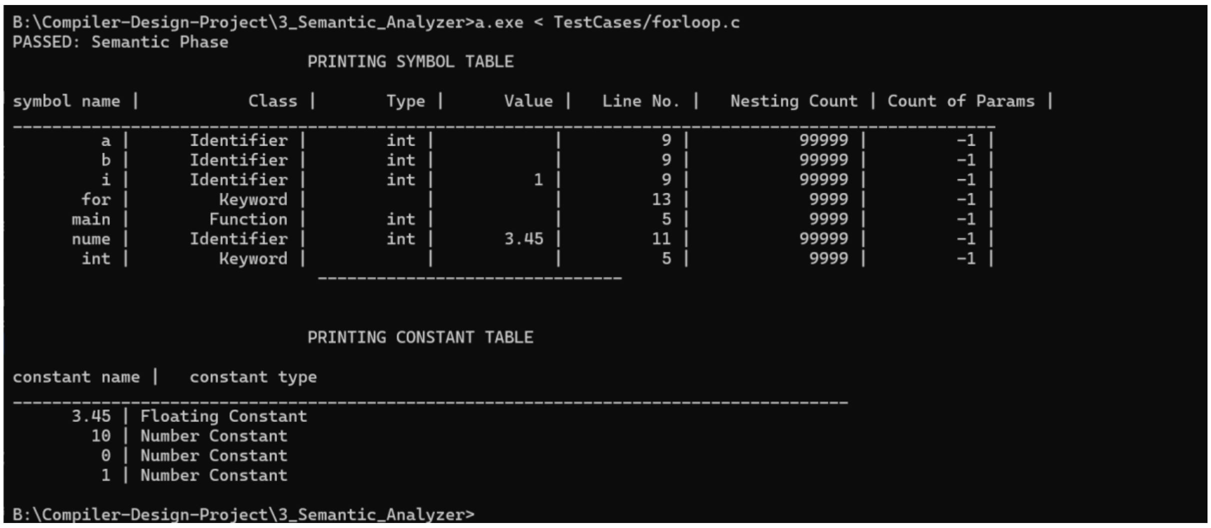


```
B:\Compiler-Design-Project\2_Syntax_Analyzer>a.exe < TestCases/forloop.c

Line:6: 'float' to 'int'



ST

   SYMBOL              NAME      TYPE        SCOPE        LINE #              VALUE
 identifier               i       int          1               5               1
 identifier               a       int          1               5               0
 identifier               b       int          1               5               -
 identifier            nume       int          1               6               3





Parse Tree

                                                                               main

                                                                       stmt

                                                               stmt           =

                                                       =           for          i           1

                                               i           0          ++           =

                                                        <           i          a           i

                                            i           10

Preorder Traversal of Parse Tree:
main     ( stmt     ( stmt     ( =   i   0   )     ( for     ( ++     ( <   i   10   )   i   )     ( =   a   i   )   )   )     ( =   i   1   )   )
```

3. Semantic Analyzer

```
flex scanner.l
yacc -d parser.y
gcc lex.yy.c y.tab.c
a.out < TestCases/forloop.c
```

Parsing result will be printed in the terminal.

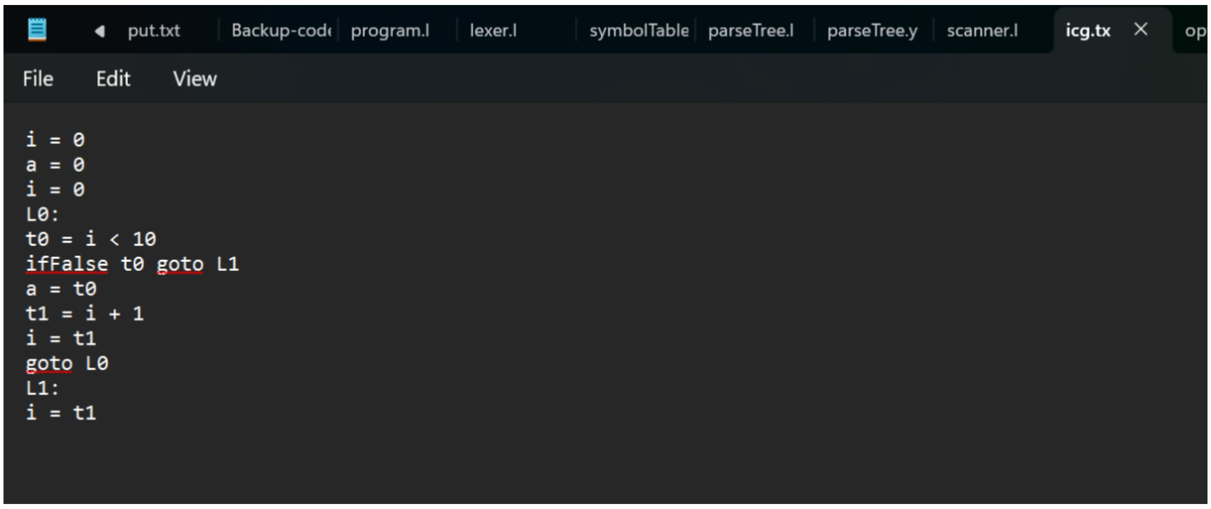


```
B:\Compiler-Design-Project\3_Semantic_Analyzer>a.exe < TestCases/forloop.c
PASSED: Semantic Phase
                              PRINTING SYMBOL TABLE

symbol name |           Class |       Type |      Value |   Line No. |   Nesting Count | Count of Params |
-----------------------------------------------------------------------------------------------------
          a |      Identifier |        int |            |          9 |           99999 |          -1 |
          b |      Identifier |        int |            |          9 |           99999 |          -1 |
          i |      Identifier |        int |          1 |          9 |           99999 |          -1 |
        for |         Keyword |            |            |         13 |            9999 |          -1 |
       main |        Function |        int |            |          5 |            9999 |          -1 |
       nume |      Identifier |        int |       3.45 |         11 |           99999 |          -1 |
        int |         Keyword |            |            |          5 |            9999 |          -1 |
                                -------------------------------


                              PRINTING CONSTANT TABLE

constant name |   constant type

--------------------------------------------------------------------------------------
       3.45 | Floating Constant
         10 | Number Constant
          0 | Number Constant
          1 | Number Constant

B:\Compiler-Design-Project\3_Semantic_Analyzer>
```

4. Intermediate Code Generator

```
flex ICG.l
yacc -d ICG.y
gcc lex.yy.c y.tab.c
a.exe < TestCases/forloop.c
```

Output will be in ICG.txt file

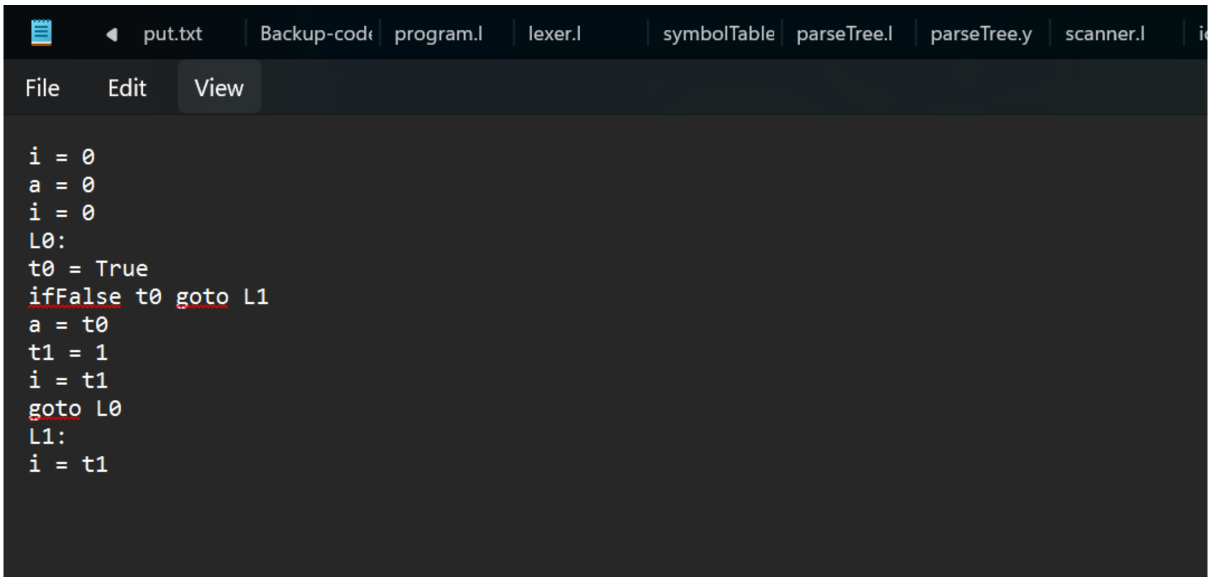


```
i = 0
a = 0
i = 0
L0:
t0 = i < 10
ifFalse t0 goto L1
a = t0
t1 = i + 1
i = t1
goto L0
L1:
i = t1
```

5. Code Optimizer

```
python optimizer.py input.txt
```

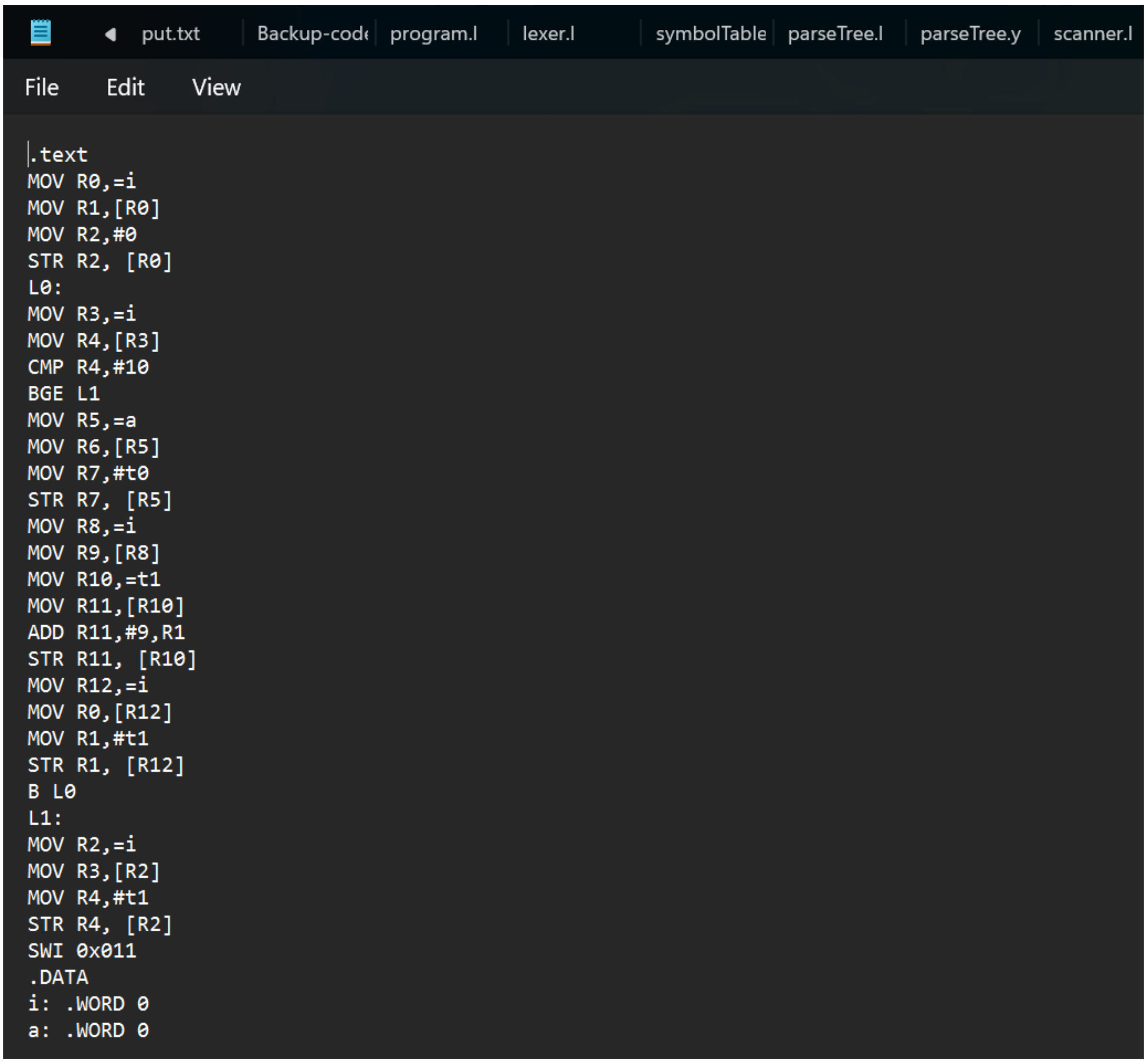
Output will be in optimized_icg.txt



```
i = 0
a = 0
i = 0
L0:
t0 = True
ifFalse t0 goto L1
a = t0
t1 = 1
i = t1
goto L0
L1:
i = t1
```

6. Target Code Generator

```
python assembly.py icg.txt
```



```
.text
MOV R0,=i
MOV R1,[R0]
MOV R2,#0
STR R2, [R0]
L0:
MOV R3,=i
MOV R4,[R3]
CMP R4,#10
BGE L1
MOV R5,=a
MOV R6,[R5]
MOV R7,#t0
STR R7, [R5]
MOV R8,=i
MOV R9,[R8]
MOV R10,=t1
MOV R11,[R10]
ADD R11,#9,R1
STR R11, [R10]
MOV R12,=i
MOV R0,[R12]
MOV R1,#t1
STR R1, [R12]
B L0
L1:
MOV R2,=i
MOV R3,[R2]
MOV R4,#t1
STR R4, [R2]
SWI 0x011
.DATA
i: .WORD 0
a: .WORD 0
```