COVER PAGE

CS323 Programming Assignments

1. Your Name:	Dennis Wu
2. Assignment Number:	3
3. Due Date:	May 11, 2016
4. Turn-In Date:	May 11, 2016
5. Executable File Name:	Project1.exe
6. Lab Room:	CS-408
7. OS:	Windows 10 64-bit

GRADE:

COMMENTS:

2. Problem Statement

The third assignment uses code from both assignment 1: Lexer and assignment 2: Syntax to generate a symbol table and assembly code for the Rat16S. The program reads the Rat16S source code, calls the lexer function to create the tokens/lexeme, and then goes through the grammar rules using the syntax function. If the code contains no grammar error(s), the program then creates a text file containing a symbol table and assembly code.

3. How to use your program

Put the executable file (**Project1.exe**) and your Rat16S code (**code.txt**) into the same folder location. Double-click the exe file and the program will generate two files (**tokens.txt** and **assembly.txt**). If the Rat16S code contains any errors, the program will abort and the error message will be displayed inside the assembly.txt file. If there are no errors, the assembly.txt file will contain both the assembly code and the symbol table.

To run another test case, remove the current test case (**code.txt**) file and insert the next test case. Make sure to **rename** the new test case file to: **code.txt**. Running the program will override the existing output file.

Note: Please make sure that the end of the source code contains an additional <enter>, if not, tokens may be repeated.

4. Design of your program

// assignment 1: lexer function to generate token/lexeme
// assignment 2: syntax production rules

// assignment 3:

Function: Check Bool Math
Input: <nothing>
Output: <nothing>
This function is called whenever there is a math operation. The function checks the identifier's type (saved as 'x' and 'y') to see if either are type boolean. If so, an error is thrown and the parsing stops.

Pseudo:

```
Check for PushM
   save instruction table item into temp
         loop thru the entire symbol table
            if the matching item type is Boolean, return false
```

Function: Symbol Entry
Input: string lexeme, string token
Output: <nothing>
This function first calls check_symbol_table to see if the identifier name already exists or not. If the identifier is new, the function will store the identifier name, memory location, and type into the symbol table. If the item is already in the symbol table, an error will be thrown.

Pseudo:

```
Check if lexeme is already in the symbol table
        if not in table, then
                store lexeme, memory address, and type
                increment memory address
                increment symbol item count

If it is already in the symbol table
        output error
```

Function: Check Symbol Table
Input: string lexeme
Output: Boolean results
This function checks to see if the current identifier is already in the table.

Pseudo:

```
If symbol items > 0
        loop through entire symbol items table
                if input string matches a symbol table item
                        return false.
```

Function: Print Tables
Input: <nothing>
Output: <nothing>
This function prints the assembly code and symbol table into the text file: assembly.txt.

Pseudo:

```
loop through instruction items
        output the instruction number, instruction
        check for 999999 inside table
```

if so, display blank
if not, display integer

loop through symbol items
output symbol identifier, memory location and type

5. Any Limitation

None

6. Any shortcomings

- Back_patch function only stores one value and displays -999 for others
- an integer value could be assigned into a boolean variable
- parsing logic may not be 100% from 2nd assignment

Test Case 1:

assembly.txt - Notepad

File Edit Format View Help

```
          Assembly Code Listing

1               PUSHM           5001
2               PUSHM           5002
3               PUSHM           5003
4               MUL
5               ADD
6               POPM            5000


                        Symbol Table

Identifier              Memory Location                 Type
x                               5000                    integer
a                               5001                    integer
b                               5002                    integer
c                               5003                    integer
```

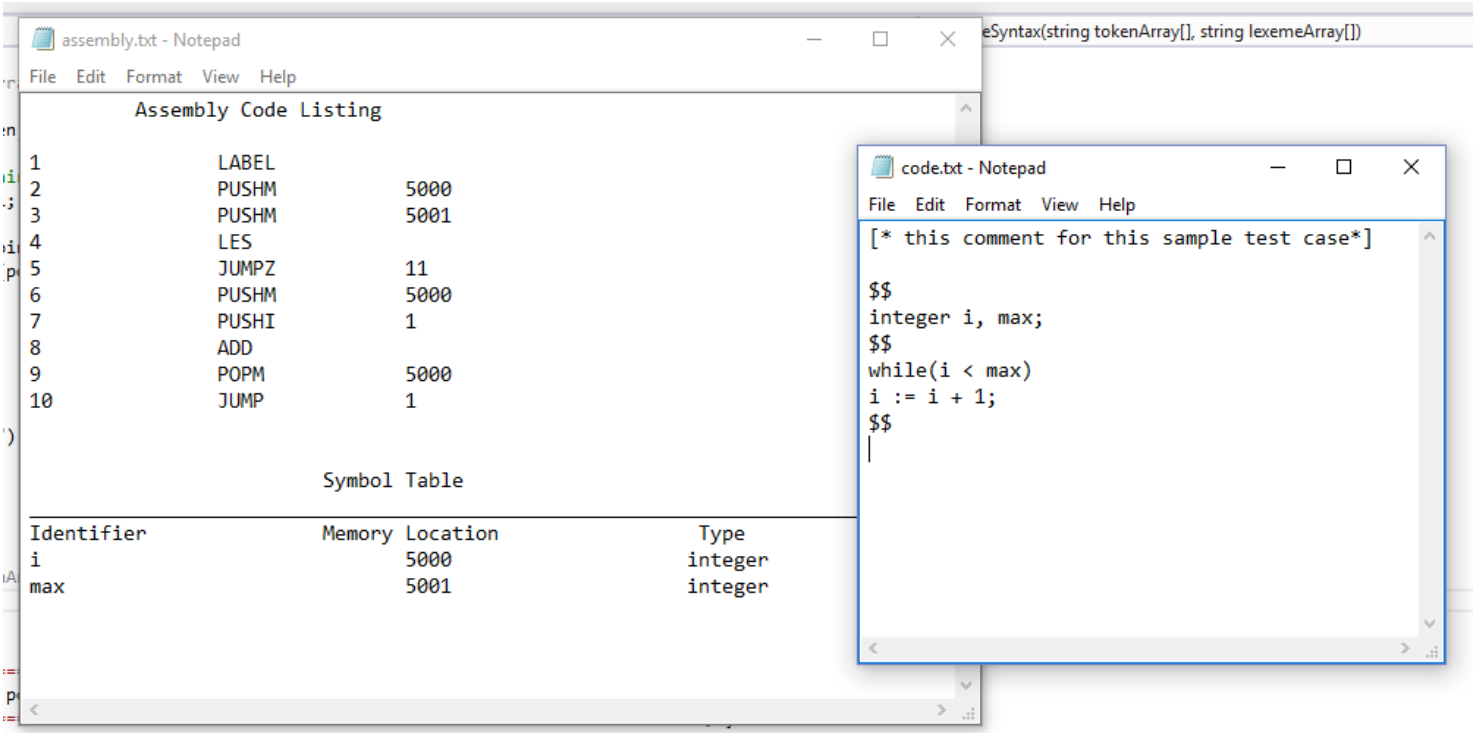
code.txt - Notepad

File Edit Format View Help

```
[* this comment for this sample test case*]

$$
integer x, a, b, c;
$$

x:= a + b * c;
$$
```

```
"Error with parsing, see output file." << endl;

le << "Error with the syntax, " << errorMsg << endl << "Now quitting...\n";
"pause");
```

Test Case 2:



assembly.txt - Notepad

File Edit Format View Help

```
        Assembly Code Listing

1               LABEL
2               PUSHM           5000
3               PUSHM           5001
4               LES
5               JUMPZ           11
6               PUSHM           5000
7               PUSHI           1
8               ADD
9               POPM            5000
10              JUMP            1


                        Symbol Table

Identifier              Memory Location                 Type
i                               5000                    integer
max                             5001                    integer
```

code.txt - Notepad

File Edit Format View Help

```
[* this comment for this sample test case*]

$$
integer i, max;
$$
while(i < max)
i := i + 1;
$$
```

Test Case 3:

assembly.txt - Notepad

File Edit Format View Help

```
        Assembly Code Listing

1               PUSHI           0
2               POPM            5002
3               PUSHI           1
4               POPM            5000
5               STDIN
6               POPM            5001
7               LABEL
8               PUSHM           5000
9               PUSHM           5001
10              LES
11              JUMPZ           21
12              PUSHM           5002
13              PUSHM           5000
14              ADD
15              POPM            5002
16              PUSHM           5000
17              PUSHI           1
18              ADD
19              POPM            5000
20              JUMP            7
21              PUSHM           5002
22              PUSHM           5001
23              ADD
24              STDOUT


                        Symbol Table

Identifier              Memory Location                 Type
i                               5000                    integer
max                             5001                    integer
sum                             5002                    integer
```

code.txt - Notepad

File Edit Format View Help

```
[* this comment for this sample test case*]

$$
integer i, max, sum;
$$
sum := 0;
i := 1;

scanf(max);

while (i < max)
{ sum := sum + i;
i := i + 1;
}
printf (sum+max);

$$
```