**CS323 Documentation**

About 2-3 pages

**1. Problem Statement**

Assignment 3 consists of 1) Symbol table handling and 2) Generating an assembly code for the simplified version of Rat18F. Every identifier declared in the program should be placed in a symbol table and accessed by the symbol table handling procedures. Each entry in the symbol table should hold the lexeme, and a "memory address" where an identifier is placed within the symbol table. The instructions should be kept in an array and at the end, the content of the array is printed out to produce the listing of assembly code. There is also a multitude of things to remove from the 2nd assignment , such as the “real” type, boolean arithmetic, <Function Definitions>. We also have to check so that types are same when assigned to a variable.

**2. How to use your program**

When you first press the executable (“Table.exe”), it asks for the file name. Enter the full filename, including extension. Once you have entered it in, it will either tell you the results of our Tables or that it’s an invalid filename that can’t be opened for reading. The results are recorded into “tableOutput.txt”. The compiler must be run in “Release” mode instead of “Debug” mode if compiling in Visual Studio. “testCase.txt” is the one with complete case. “testCase2.txt” has an error with boolean arithmetic operations in the end of the text file. It still outputs the tables. “testCase3.txt” checks for boolean.

**3. Design of your program**

For semantics.h, we have all the helper function that are used to modify the syntax functions in syntax.cpp. We created two structs to create Objects of the Instruction and Symbol for the Instruction table and the symbol table. We made it into a vector of 1,000 instructions and 100 symbols to store our tables. We also have a jumpstack. We also used the gen\_instr function in the partial solution, so that the syntax.cpp can insert into the instruction table. A function, to insert into the symbol table, was also created. We created a helper function that tells us if that symbol exists inside the symbol table. We have a getaddress function to find the address of the symbol in the symbol table. We also created a jump stack to store the jumps and functions that push and pop from the stack. Finally, we concluded the header file with print function that allows us to output the results.

In the syntax.cpp, we modified it to requirements of Assignment 3. We eliminated the <Function Definitions>, as a result we got rid of <opt function definition>, <function> , and all of the Non-terminals related to <Parameters>. We also got rid of the “real” type in syntax. When it’s the boolean value of true, we do gen\_instr(“PUSHI”, “1”) and replace 0 with 1 for the value of false. We made it so that booleans can’t do arithmetic operations in <CONDITION>. We also made it so that when you assign the variable to a variable. It must be both be the same type.

We also implemented the partial solution from titanium. We implemented the ADD , SUB, DIV , MUL instructions where they are needed. In Relop, we generated instructions based on the two character symbol , pushed in the instruction address and generated an instruction to Jump if it was zero. In <Assign> , we implemented a POPM function for the Identifier. In <Primary> , we implemented PUSHI for integers and PUSHM for identifiers (variables) . In the <print> function, we have an STDOUT. In the <Scan> function , we have STDIN. In <if prime> , we have Jumps and back patches when it hits ifend. The same goes for when it hits whileend in <While>. It generates ADD, SUB, MUL, AND DIV instructions when it needs to be. Those instructions have no operands.

**4. Any Limitation**

Visual Studio 2017 must be run in Release mode for code to compile. Program will terminate at the first finding of a syntax error.

**5. Any shortcomings**

*NONE.*

**COVER PAGE**

CS323 Programming Assignments

**Fill out all entries 1 - 7. If not, there will be deductions!**

**Check one**

**1. Names [ 1. Jarrett Chien ], (MW [ ] or R class [ X ] )**

**[ 2. Eric Edelman ], (MW [ ] or R class [ X ] )**

**[ if 3. Michael Li ], (MW [ ] or R class [ X ] )**

**2. Assignment Number [ 3 ]**

**3. Due Dates Softcopy [ 12/10 ], Hardcopy [ 12/15 ]**

**4. Turn-In Dates Softcopy [ 12/10 ], Hardcopy [ 12/15 ]**

**5. Executable FileName [ SyntaxAnalyser.exe ]**

**(A file that can be executed without compilation by the instructor)**

**6. LabRoom [ CS-200 ]**

**(Execute your program in a lab in the CS building before submission)**

**7. Operating System [ Windows 10 ]**

**To be filled out by the Instructor:**

**GRADE:**

**COMMENTS:**