**CS323 Documentation**

About 2 pages

1. Problem Statement

<write the problem statement here. You can mostly get it from the assignment itself>

**In this assignment, we are tasked with symbol table handling and generating an assembly code for the simplified version of Rat24F. We are to place all the identifiers declared in the program in a symbol table and accessed by the symbol table handing 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 parser should also produce the assembly code instructions. The instructions are kept in an array, and at the end, the content of the array is printed out to produce the listing of assembly code. The array should have at least 1000 assembly instructions, and the instruction starts from 1. The listing should include an array index for each entry so that it serves as a label to jump to.**

2. How to use your program

<write detailed steps how to execute your program>

1. **Download the zip folder and ensure there is the “main.py” file and there are 6 text files named “input(2-3).txt” and “output(2-3).txt”**
2. **Execute the main.py file, and it will read the “input.txt” run the lexer function and syntax analyzer, and store the results in “output.txt”**
3. **To test the other input text files, change the file input and output in the source code**
   1. **Variables labeled Input\_file and Outout\_file change the input and output files written to and from**
4. **Open the output file to see your results**

3. Design of your program

< write major components of your program. Also, data structures you are utilizing, particular algorithms you have chosen etc. >

**When we were writing the program, we were tasked with creating the functions for compound, get, and put. These functions are supposed to return our assembly instructions. We also created a table using multiple lists within a list to keep track of all the memory addresses for each identifier, the set of instructions, and the operand. We would call the generate instruction function that was provided to us for the assignment, if, while, compound, get, and put statements. There was not much to implement outside of the if-else functionality since the majority of the code was given, and we just needed to place the function calls in our preexisting syntax analyzer code. The output is just the assembly code listing and the symbol table. We also made modifications to other functions to ensure that only variables declared in the initialization section and stored in the symbol table can be used throughout the rest of the code.**

4. Any Limitation

<All features are running according to the assignment but you limit your program due to resource limitations, such as Maximum number of lines in the source code, size of the identifier, integer etc. Say ‘None’ if there is no limitation>

**None**

5. Any shortcomings

<Anything you could NOT implement although that is required by the Assignment. Say ‘None’ if there is no shortcoming>

**None**