**CS323 Assignment 3 Documentation**

1. **Problem Statement**

The purpose of this assignment is to create assembly code and associated symbol table after lexical and syntactical analysis of the source code. Assembly procedures are written for various

1. **How to use your program**
2. Unzip the file for Assignment3\_Final.
3. If you have your own test file, the easiest way to analyze it would be to replace the contents of the first test1.txt file.
4. Double click the Assignment3.exe. If this works, skip to step 7. If not, continue on to the next step.
5. In windows cmd, enter into Assignment3\_Final folder.
6. Type dir to see if you are in the right folder and Assignment3.exe is available.
7. Type Assignment2.exe in terminal and program should run.
8. Test output files (ex. Assembly\_output\_test1.txt) will show the Assembly code of the corresponding test file (ex. test1.txt).
9. The terminal will also show analysis of the assembly code and symbol table.
10. **Design of your program**

The symbol table was created inside of the SA class that created for the syntactical analyzer. All data and related functions are also in the class. The data structure of the table is an array of structs. The structs have the properties name, type, and memory. Insert, lookup, and print have all been created for the symbol table.

The assembly code was stored in a vector called instruct\_table. The data type for the table was a struct consisting of the properties operation, operand, and instruction number. Print function was created to print all instructions to a txt file and console.

For analysis purposes, a vector was created for the jump stack. This data structure held addresses that would be needed for analysis of the while loop and if statements.

1. **Any Limitations**

The comments will be ignored, but there must be a space after the beginning of the comment symbol (/\* ) and a space before the end of comment symbol. ( \*/).

1. **Any shortcomings**

Because of type casting, all NULL values were output as 0 for the operands in the assembly code. This can cause confusion, as some operands do need to show 0 is the value (such as EQU and PUSHI). These two instructions work properly, but can get confusing, so we can’t always assume 0 is NULL. It would have to be viewed from the perspective of the instructions.