**CS323 Assignment Documentation**

***<should consists of about 2-3 pages>***

1. **Problem Statement**

*Implement symbol and instruction table handling.*

*Every identifier should be placed into a symbol table and be provided   
a memory address, starting with 10,000. Check to see if an id is already in the table when referencing it. Provide a function to print it out.*

*Add code that will produce the assembly code instructions. The table   
should be printed out to produce the listing of assembly code. Your array should hold at least 1000 assembly instructions. The instruction starts from 1. Also print it out.*

1. **How to use your program**

*By default, the program will try and analyze the file ‘input.txt’. This file should be in the same directory as the program.*

*Should you want to run a differently named file, pass the name of the file to the program as an argument.*

*Upon running, the console will ask whether you want console output of the errors, lexeme-token pairs & production rules. (1: yes, 2: no)*

*Upon completion, you will be informed of the result.*

*An output file, ‘output.txt’ will be generated with all output.*

1. **Design of your program**

***Semantics class***

***Symbol Table*** *– Holds symbols, their type, and their address*

*Gen\_sym: Creates a new entry if not already pre-existing*

*Get\_addr: Retreives the address of a symbol*

*Get\_type: Retreives the type of the address*

***Instruction Table*** *– Holds generated instructions and operands*

*Gen\_instr: Adds an entry to the table*

***Jump Stack*** *– Holds addresses for jumping purposes*

*Push\_jump\_stack: Pushes an address to top of stack*

*Back\_patch: Patches an instruction in the instruction  
table with the address from top of jump stack*

***Changes to syntax\_analyzer:*** *Added use of the newly added semantics class.*

*Now when making a declaration, it calls the appropriate functions to   
ensure the variable hasn’t already been declared.*

*Additionally ensures that a variable has been previously declared prior to use.  
When performing comparisons, assignments, or arithmetic, type matching between expressions is ensured.*

*When making a conditional statement, creates jump points and backpatch them accordingly.*

1. **Any Limitation**

*N/A*

1. **Any shortcomings**

*N/A*