**CS323 Documentation**

1. **Problem Statement**

*Alter Rat15s to include semantic for assembly instructions.*

1. **How to use your program**

*Run the file “cpsc323ver3.exe”*

*When prompted, enter the name of text file to be read.*

*A file named “FileOut.txt” will be generated by the program.*

*The “FileOut.txt” file will contain the results generated by the program using the specified text file.*

1. **Design of your program**

*Program design for assignment 3*

*Structs are used for the symbol table and instruction tables. All identifier are stored in the symbol along with their declared type and memory location.*

*The ispresent() function is used to determine if an identifier present in is symbol table and will return memory location of identifiers in the table. Identifiers not in the present table will return a value of -1.*

*Back\_patch() function is a reproduction of the code from “assignment 3 solutions” handout.*

*<Opt Function Definitions> has been commented out to meet the requirements of this assignment.*

*For the condition statement, relop for <=, =>, != generated two instructions. In “<=” instruction GRT is used followed by a JUMPZ which is paired with EQU instruction followed by JUMPZ. The idea is that meeting either of those conditions will allow it to exit the loop. For “!=” the instruction pair used GRT and LES instructions each followed by the conditional JUMPZ command.*

*For the “If else endif” statement function, the intended design was to use a combination of conditional unconditional jumps to achieve this function. First a conditional jump instruction is generated to be used if the condition was meet. A returned 0 which instructed the program to jump to the second statement. If the condition was 1 the unconditional jump would not have been used which allows the execution of the instructions following the condition. At the end of the statement an unconditional jump is used to jump to the following label location for the purpose of skipping the first set of statement instructions.*

*Program design for assignment 2*

*A recursive descent parser (RDP) is used to parse the sentences. A doubly linked list is used to temporarily contain tokens from each sentence until it is parsed by RDP. Syntax rules used are those contained in project description.*

*Program design from assignment 1*

*2d arrays is used to store two FSM tables, one for the identifier FSM and another for the combined table for real and integer.*

*Regular Expressions used are as follows;*

*Integer = d+*

*Real = d+.d+*

*Identifier = L((L|d|\_)\*(L|d))\**

*A line of source code is read from a specified text file into a string called “sentence”. The line is parsed into tokens using special characters as a delimiter. Which is identified by the isspecial() function. The token is passed to the lexer() function. The special character is then itself passed to the lexer() function. The lexer () function checks the FSM table to determine its appropriate state. Once the state is determined, the lexer() functions calls the statetoken() function and passes the state information with the token. The statetoken() displays and writes to file the appropriate token type for the lexeme. The process is repeated until all portions of the string have been consumed. When the string has been completed process the next line of source code is read into the string. This process is repeated until all lines in the text files have been read and feed thru lexer.*

1. **Any Limitation**

*Array size declared as 300 for the symbol table and instruction table. The jumpstack array is declared to be size 100.*

1. **Any shortcomings**

* *“If else endif” statement is not working properly. Unconditional “JUMP” instruction in the “if else endif” does not generate a forward location address to jump to.*
* *No sematic action produced for compound statement.*