Michael Cingari (Section 3) Owen Salcido & Brian Chung (Section 4)
Monday, December 16, 2019
Anthony Le, CPSC 323

Assignment 3 Documentation

1. Problem Statement:

The assignment was to create a program that handles symbol handling as well as generating assembly code from high-level code. The program reads through an input file (input.txt) and outputs assembly code in the console. The program is built upon our top-down parser, so it continues to checks if the given input code is syntactically accepted while converting the code into generalized assembly.

2. How to use the program:

Run the main.py python file with python3. ~ Represents the directory path to folder; all python files are contained in folder 'code.' This will print the assembly code from the input found in ~/code/input.txt.

Note: The python external library '*pandas*' is required to read the CSV file that contains the 2D table for the table-driven predictive parser. Please view requirements.txt.

Windows

- Open Command line: Start menu -> Run and type cmd
- Type: py ~/code/main.py

Mac OS X

- Open Command line: Finder -> Go menu -> Applications -> Terminal
- Type: python3 ~/code/main.py

Linux

- Open a command prompt (e.g. terminal)
- Type: python ~/code/main.py

3. Design of program:

In a nutshell, the program reads through all valid lines/items in an input file and runs the table-driven predictive parser. During the top-down parser, the function stores the current state that is found on the parse tree. The program reads the input.txt file that contains the user code. The program will pre-process the code removing any comments as well as formatting the code such that the parser can adequately comprehend the user's input code. The program checks if the program is syntactically correct by indexing the production rule table. (Grammar for the input code can be found in Grammar.txt)

Because it is a top-down driven parser, it is slightly challenging to manage the current states of the code. To overcome this program, the program will store the last seen production rule and compare it against accepted or key states. If the state is a key state (determined if new rules shall apply), the program will store that state. Given the state and the next input on the input string, the program will follow rule sets and output assemble code accordingly.

To manage the memory or where each variable would be stored to each memory address, the program also saves any declared identifier into a dictionary of variables. This dictionary holds the memory address as well as the variable type. The function finally outputs all the assembly code and variable information to the console. The program will also output if the input code was syntactically correct.

4. Any Limitations:

Currently, the program can only handle integer declarations and allocations; no other data types have been implemented for the scope of the assignment. The program can also convert assignment, addition, subtraction, multiplication, and division operators.

The program cannot code all the functions of the proper coding language. Loops, if statements, and functions have not been implemented.

5. Any Shortcomings:

None.