CS323 Documentation

About 2 pages

1. Problem Statement

Our project seeks to address the issue of deriving assembly code from a given program that's been analyzed lexically and syntactically. The process of generating assembly code is largely comprised of two parts: symbol table handling and the generation of assembly code given a simplified version of our initial Rat24S language. Symbol table handling deals with the interaction of the symbol table; determining the correct circumstances under which identifiers are inserted into the table along with their corresponding memory address and data type. Symbol table handling also includes error checking and returning meaningful error messages when needed so that the symbol table can best avoid conflicts and errors. Generating assembly code is done by utilizing a modified version of the functions created for syntax analysis. The assembly code generator will generate the corresponding assembly instructions for each syntactic parsing function, and will maintain that within an array that contains data on the instruction address, operation, and operand for said assembly code instruction. At the end of the program, both the symbol table and assembly code instruction array should be printed.

2. How to use your program

3.

1. *Extract the files within the zipped folder.*
2. *Check that the assembly_code_generator.py, lexical_analyzer.py, and all the test case files are in the same directory.*
3. *Open up terminal/command prompt in that directory, type: python main.py and hit enter.*
4. *It should export the assembly code generator output into individual text files corresponding to the test files: "test_case_one_assembly.txt", "test_case_two_assembly.txt", and "test_case_three_assembly.txt".*
5. *It will also print to the console the assembly code generator output (symbol table and instruction array).*

6. *You can modify the input of the test case by directly modifying the text file itself or replacing it with your own file with the same file name.*
7. *The python version we are using is python 3.12*

4. Design of your program

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

Major components of the program include a lexical analyzer which performs a lexical analysis of the input, breaking it into lexemes and tokens. Within the assembly code generator file, there is a simplified syntax analyzer along with new data structures and functions intended to create assembly code. Notable data structures utilized include python dictionaries within the lexical analyzer as well as the assembly code generator. In addition, lists, 1-dimensional and 2-dimensional, were utilized. Notable algorithms include the recursive algorithm utilized within the simplified syntax analyzer for the assembly code generator.

5. 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

6. Any shortcomings

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

Shortcomings in our project include the lack of data type checking to see if there were any disallowed operations on certain data types occurring. Our program does not check if arithmetic operations were performed on booleans, or that types must match for arithmetic operations.