Simple Compiler v3 Assignment 1 – Front End and LLVM IR

Be sure to fully review and comply with the SimpleO Language Specification document for this and all future SimpleCompilerv3 assignments.

In this assignment, the following functionality will be developed.

Completion of the front-end of the compiler to output syntactically correct LLVM IR (as checked by llc and our test cases evaluator program) for the following functionality:

1. Returning immediate values (e.g., "return 10;")
2. Returning immediate expression values for addition (e.g., "return 10 + 20")
3. Returning immediate expression values for subtraction (e.g., "return 20 – 10")
4. Returning immediate expression values for multiplication (e.g., "return 10 * 20")
5. Variable declaration with explicit assignment of integers only (e.g., "int x = 10;")
6. Returning expression values for the above binary operations with the use of variables, and variables mixed with immediates (see test-cases directory)
7. Returning expression values for the above unary operations with the use of variables, and immediates (see test-cases directory)

****All provided test cases must work without error or warning to receive full credit. These test cases are not comprehensive and there are likely edge cases that may or may not work that are not covered by the test cases. However, the test cases provide sufficient coverage for the functionality requested (and they align with the SimpleO Language Specification document). The test cases directory is cumulative and later assignments include all test cases from the previous assignments to make sure you don't break functionality as you develop your compiler. You may develop your own test cases, though I will only be using the supplied test cases to evaluate your submissions for each assignment.

****Do not modify the MakeFile, shell scripts, file names, or directory structure. Any modification can lead to it not working in my environment when I grade it and you will not receive credit if I cannot evaluate your compiler. Only Ubuntu 22.04 LTS is supported. No other environments have been validated and are not supported.

Be sure to follow a professional coding standard, including the naming of your variables, etc. You should use descriptive names that are appropriate. Nonsensical or whimsical names will lose points.

Additionally, you must attest and add the following honor statement to your Canvas submission (in the Canvas submission comments):

I, <state your name>, attest that this submission is my own work. I have not worked with anyone else on my implementation and have not used AI or other unpermitted sources to help me with my solution, in accordance with our definition of class cheating shown in Lecture 1.

Any submission that does not include this attestation will not receive any credit.