CS316: Compilers Lab

Programming Assignment 5: Loops Due: 31/3/2022

1 Introduction

Your goal in this step is to generate executable code for loops. You have already generated code for altering the flow of control in case of 'IF' construct in the previous assignment. In this assignment, you will generate code for the 'for' loop, which could have BREAK and CONTINUE statements.

Testing your Tiny code You can test your Tiny code by using the same simulator as in the previous step. Your compiler will be tested against only the inputs that we provide. *However, in the next assignment, there will be hidden test cases* .

Sample inputs and outputs: inputs and outputs.

2 What you need to do

In this step, you will be generating assembly code for 'for' loops, 'while' loops and function calls, as described above. You should correctly be able to handle functions with return values, functions where complex expressions are passed in as arguments (store the result in a temporary, then push that temporary onto the stack as the argument), and recursive functions.

Handling errors All the inputs we will give you in this step will be valid programs. We will also ensure that all expressions are type safe: a given expression will operate on either INTs or FLOATs, but not a mix, and all assignment statements will assign INT results to variables that are declared as INTs (and respectively for FLOATs).

Grading In this step, we will only grade your compiler on the correctness of the generated code. We will run your generated code through the Tiny simulator and check to make sure that you produce the same result as our code. When we say result, we mean the outputs of any WRITE statements in the program (not details such as how many cycles the code uses, how many registers, etc.)

We will not check to see if you generate exactly the same code that we do – no need to diff anything. We only care if your generated code *works correctly.* You may generate slightly different code than we did.

3 What you need to submit

- Place all the necessary code for your compiler that you wrote yourself. You do not need to include the ANTLR jar files if you are using ANTLR.
- A Makefile with the following targets:
  1. `compiler`: this target will build your compiler. If you are using ANTLR, this should create a `.jar` file.
  2. `clean`: this target will remove any intermediate files that were created to build the compiler.
  3. `dev`: this target will print the same information that you printed in previous PA.
- A shell script (this must be written in bash) called `runme` that runs your compiler. This script should take in two arguments: first, the input program file to be compiled and second, the filename where you want to put the compiler. You can assume that we will have run `make compiler` before running this script.

- You should tag your programming assignment submission as cs316pa5submission

While you may create as many other directories as you would like to organize your code or any intermediate products of the compilation process, both your `Makefile` and your `runme` script should be in the root directory of your repository.

Do not submit any binaries. Your git repo should only contain source files; no products of compilation. If you have a folder named `test` in your repo, it will be deleted as part of running our test script (though the deletion won't get pushed) – make sure no code necessary for building/running your compiler is in such a directory.