Programming Assignment 5 - Extra Credit
Tuesday, June 6, 2023 at 11:59pm

IMPORTANT: Due to the constraints on getting course grades calculated, no late submissions will be accepted for this assignment. Consider putting in "safety submissions" as you go so that you don't miss the deadline.

1 Introduction

This extra credit programming assignment is intended to be a fun project with which to end the quarter – the goal here is to implement some optimizations in your compiler.

The final curve for the course will be determined *before* including the extra credit. In other words, if you elect not to do extra credit work, you will not be at a disadvantage in the final grading with respect to those who do.

This extra-credit option is open-ended; you can do as much as you like. We will award credit for results. A project that merely attempts, but does not complete, an optimization phase may receive as little as no extra credit.

We have not implemented an optimization phase in coolc, so we have no skeleton code to give you—you are on your own. If you want to do an optimization phase, you are encouraged to talk it over with one of the course staff first. *Under absolutely no circumstances should you try optimization before your code generator is finished!!*

2 Details

Extra credit will be awarded for projects that, in addition to code generation, perform some significant optimization of the code. The amount of extra credit depends on how well the optimization is written, documented, and demonstrated. Two critical factors are

1. correctness (the optimizations don't result in incorrect programs); and
2. the percentage speed-up your optimized code achieves over coolc, as measured by a weighted sum of the instructions executed on spim over a suite of benchmarks of our choosing.

To find out how many instructions a Cool program executes, run spim with the `-keepstats` option.

There are many possible optimizations to implement. Assuming your initial code generator is straightforward (like coolc's), then two directions that may yield significant improvement are (1) improving register usage and (2) specializing the implementation of the basic classes **Int** and **String**.

There is a `-O` flag that controls the global variable **cgen_optimize** found in src/handle_flags.cc. You may use this flag to switch between generating normal code and optimized code. For this project, we will always run your compiler with the `-O` flag on. The total extra credit for doing optimization will not exceed 6% of the total grade for the course. Roughly speaking, the extra credit is worth up to about half of one of the two large programming assignments.

3 Files and Directories

There is no separate starter code for PA5; you should just re-submit your code **using the instructions below to ensure it gets picked up as a PA5 (as opposed to PA4) submission** after adding in the optimizer. Include a short description of your optimizations in the README file.

4 Submission

Before you submit your optimizer, ensure the following:

- If you added source files, make sure they are built properly by your Makefile. Ensure that your code compiles and runs correctly on myth using **your** Makefile.
- Include a brief description of your optimizations in the README file.
- Open a shell in the "root" directory of your submission (i.e the one containing the Makefile) and run the submission script as follows:

  ```
  /afs/ir/class/cs143/bin/pa_submit PA5
  ```

  It is your responsibility to double check that the submission script picks up all the files needed to build and run your optimizing code generator.