Programming Assignment 5 - Extra Credit
Tuesday, June 2, 2015 at 11:59pm

1 Introduction

This extra credit programming assignment is intended to be a fun project with which to end the quarter. You may work in a group of one or two people.

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. For example, 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 (C++) and Flags.cgen_optimize (Java). 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.

3 Files and Directories

To get started, log in to one of the corn machines and run one of the following commands:

/usr/class/cs143/bin/pa_fetch PA5 <project_directory> for the optimizer (C++ version)

/usr/class/cs143/bin/pa_fetch PA5J <project_directory> for the optimizer (Java version)
The project directory you specify will be created if necessary, and will contain a few files for you to edit and a bunch of symbolic links for things you should not be editing. (In fact, if you make and modify private copies of these files, you may find it impossible to complete the assignment.) See the instructions in the README file.

For both the C++ and Java optimizers, you will probably want to start by copying over your code from PA4 and making sure you have something that is functionally correct before you start implementing optimizations.

4 Submission

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.

Submit your optimizer by running “make submit” from your PA5 or PA5J directory. If you added source files, make sure they’re in the list and that they are built properly by your Makefile. Include a description of your optimizations in the README file.