Extra Credit Programming Assignment
Due Thursday, March 15, 2007 at 11:59pm

1 Introduction

This extra credit programming assignment is intended to be a fun project with which to end the quarter. There are two choices for this extra credit assignment:

1. implement some optimizations in your compiler; or
2. write a Cool program.

You may submit either an optimizer or a test program, but not both.

You may work in a group of one or two people.

2 Optimizer Project

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; see the ASU chapters 9 and 10 for ideas. 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**.

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!!*

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.

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 Test Program Project

Extra credit will be awarded for submitted Cool programs that reveal bugs or strange behavior in optimizing Cool compilers. The more projects your program breaks, the more it is worth! Test cases cannot be based, even loosely, on pre-existing Cool programs, such as those in the `examples` directory.

Test programs will be worth up to 3% of the total grade, but should also be much easier to write than optimizers.

4 Grading

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.

5 Submission

5.1 Optimizer Project

Submit your optimizer by running "`/usr/class/cs143/bin/submit`" and selecting assignment "`PAX1`", which will collect the same files as for the code generator assignment (PA5).

- ☐ Include a write-up of the optimizations you performed in the README.
- ☐ Make sure all your code for the optimizer and code generator is in
  - cool-tree.h, cgen.h, cgen.cc, cgen_supp.cc, and emit.h for the C++ version; or
  - cool-tree.java, CgenClassTable.java, CgenNode.java, CgenSupport.java, BoolConst.java, IntSymbol.java, StringSymbol.java, TreeConstants.java, and additional .java files you may have added for the Java version.
- ☐ Be sure to answer 'yes' to the submission prompt for files that contain your code.

5.2 Test Program Project

Submit your test program by running "`/usr/class/cs143/bin/submit`" and selecting assignment "`PAX2`", which will collect all .cl files. Your submission will be treated as one test program; in other words, all .cl files will be compiled together into one program. Please include comments in your source file that describe your test program.