COMP 520 Compiler Design
Peephole Contest

Due: Friday, April 13th 11:59 PM

Overview

In class we study the JOOS language and corresponding Java bytecode. Before getting started on this assignment, you may want to review the lecture slides (bytecode and optimization) to make sure you are familiar with the material. The purpose of this assignment is to increase your familiarity with Java bytecode, and to introduce you to practical peephole optimization. This assignment is to be completed in your `GoLite` project groups.

To get you started, we have prepared a template consisting of the JOOS source code, a small set of benchmarks, and various helper scripts for tuning your optimizations. It is available at `https://github.com/comp520/Peephole-Template`. Inside the template you will find all that you need for this assignment, including the `patterns.h` file inside the `JOOSA-src` directory. We saw how these patterns worked in class, and you can review the slides for inspiration.

Objective

The objective of this assignment is to create `.j` files which have the fewest number of bytecode instructions as possible. For each pattern that you add to `patterns.h` you must:

1. Ensure that it is sound
2. Check that it improves the generated code in some fashion (to ensure that peephole optimizer reaches a fixed point)
3. Put a comment which clearly describes the pattern, and argues why it is sound. Unsound patterns will result in deductions of marks, and will be removed for the contest

We have also included a set of 6 benchmarks to help you design and test your optimizations. As you work on your submission, your workflow will likely be as follows:

1. Generate the bytecode for all benchmarks (`count.sh` script)
2. Analyze the generated bytecode (`.j` files) for inefficiencies
3. Design a pattern that improves the code size
4. Test for (a) soundness; and (b) code size improvement

We will test your optimizer on these benchmarks, as well as some others that we keep hidden. This is similar to real world optimization, where you must ensure that your patterns work both on your test set and also arbitrary user defined programs.

Submission

On myCourses, hand in a your `patterns.h` file. Your code should be self-contained and work on the standard JOOS compiler without modification.