COMP 520 Compiler Design
Peephole Contest

Due: Tuesday, April 11

Overview:

The purpose of this assignment is to get everyone familiar with Java bytecode, and the basics of a peephole optimizer.

We have already studied JOOS and the peephole optimizer in class, and you may want to review the class notes on JOOS and Java bytecode. You should also review, in detail, the slides from Week 8 on optimizations, which give an introduction the the JOOS peephole optimizer.

I have also created for you a directory which has everything you need to build and test your peephole optimizer. You can find this at:
`http://www.cs.mcgill.ca/~cs520/2017/assignments/PeepholeContest/`,
with a .tar.gz file that you can download from
`http://www.cs.mcgill.ca/~cs520/2017/assignments/PeepholeContest.tar.gz`.

If you have already made your own structure, then just take the bits from this that you need. You should take at least the JOOS compiler files (because I gave you some A+ features in it), and the benchmarks.

You should download and untar this file in some directory. Then, you should set an environment variable called PEEPDIR to refer to this directory. All commands have been set to be relative to PEEPDIR.

What you will find in PEEPDIR:

JOOSA-src: The source code for the JOOS compiler. This is mostly the A+ compiler, except for the A+ peephole patterns. Thus, it does a bit more than the A- compiler previously put on the web site. For example, it supports for loops, increment expressions, and proper computation of stack height. These extensions used to be class assignments, but we did different assignments this year.

The key file you need to edit is `patterns.h`. This is where you will add your new patterns. We saw how these patterns worked in our lecture, and you can review the slides.

jasmin.jar: A copy of jasmin, used by the `joosc` script.

jooslib.jar: A copy of the JOOS library. You will need to make sure this is on your classpath if you want to run the programs you are compiling and optimizing. It would be a very good idea to do this, as a sanity check that your optimizations are not changing the behaviour (which would be bad).

joos: This is a script that calls the joos compiler in JOOSA-src/ directory. It appends the externals joos file names to the arguments you give. It produces one `.j` file for each input `.java` file.

joosc: This calls the joos compiler to generate the `.j` files and then calls jasmin to generate the `.class` files. You should be able to run those `.class` files with any Java system. If you are using a cygwin windows system, then use the `joosc.windows` file (rename it to `joosc`).

JOOSexterns: These are the .joos files that define the external signatures. They are included by the scripts.

PeepholeBenchmarks: There are 7 benchmarks in this directory. For each benchmark you should be able to use the Makefile (if there is one), or you can just ues the command: `$PEEPDIR/joosc *.java` or `$PEEPDIR/joosc -O *.java`. Note that your peephole optimizations will be applied only when the `-O` flag is set.

We will test your optimizer on these benchmarks, as well as some others that you don't get ahead of time.

Objective

The objective is to create output .j files which have the fewest number of bytecode instructions.

For each pattern that you add to `patterns.h` you must:

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

You will be asked to submit your `patterns.h` file, so all of your code should be in this file, and it should work in the standard JOOS compiler without modification.

I will provide a script for counting the number of instructions (I am just trying to improve it).