

CS 577 Homework 2 – Generating direct threaded code – due 10am, Thursday, April 17, 2008

On the course web page, you'll find C code for an indirect threaded code interpreter for the same subset of JVM instructions as in the (solution to) homework 1. Again, this subset supports (most) integer arithmetic, integer arrays, static methods, and a minimal set of output facilities. The interpreter is defined by a set of C files (`interp1_itc.c`, `class.[ch]`, `runtime.[ch]`, `basics.[ch]`), and a makefile, which generates an executable `interp1_itc`.

Your primary assignment is to alter `interp1_itc` to use *direct* threading instead of indirect threading. This will involve translating the bytecode for each method to a sequence of code snippet addresses, and modifying the interpreter to dispatch accordingly.

A secondary assignment is to write a benchmark using this JVM subset and exercise it on the various interpreters.

Details of Direct Threading

Modify the existing `itc` interpreter to do direct threading instead, using the `gcc` extensions for taking code addresses. Implement direct threading by modifying the `interp` method to translate the bytecode for each method into a sequence of snippet addresses before executing it. (Obviously this should only be done once per method. The simplest thing is to do this the first time the method is called, and cache the result in the `data` field of the `method` structure, which is reserved for the interpreter to do what it likes with.) It is simplest to allocate the snippet sequence as an array of `u4` values, and declare `pc` as a `u4*`. You'll have to figure out how to handle parameter bytes; don't worry about producing a compact sequence. You may find the auxiliary file `param_size.h` to be useful. Name your resulting interpreter `interp1_dtc.c`

Details of benchmarking

Write one or more benchmark programs exercising the features of the Java subset used by the interpreter. To be useful, benchmarks should run for 10-20 seconds on your preferred platform. Use your benchmark(s) to compare the performance of `interp1.c`, `interp1_itc.c` and your `interp1_dtc.c` on one or more machines of your choice (the more unusual the better!). Also compare the performance of `interp1_itc_cache3.c`, another interpreter provided on the web page, which uses 3-state stack caching. Write a brief summary of your benchmark results; if any of them are counter-intuitive, try to give an explanation.

Your benchmark(s) will be shared with the rest of the class as part of a benchmark suite for testing optimizations in future assignments.

How to submit your homework.

Submit the homework by mail to `apt@cs.pdx.edu` with the subject line "CS577 HW2". The elements of your submission should be plain-text attachments to your mail. Submit:

- A file `interp1_dtc.c` containing your modified interpreter.
- File(s) containing the `.java` source for your benchmark program(s).
- A file `benchmarks.txt` containing a description of your experiences benchmarking `interp1`, `interp1_itc`, `interp1_itc_cache3`, and your `interp1_dtc`.