CSE P548: Computer Architecture
Assignment 3
Due: Tuesday, May 6

Many architects, including our authors, claim that superpipelining and increasing the instruction issue width are duals of each other with respect to performance. The goal of this assignment is to examine that hypothesis, confirm or refute it, and, in the process, gain greater understanding of the performance benefits and drawbacks of the two techniques. Specifically, you should compare two different implementations, one that issues twice as many instructions per cycle as the base (default) processor of the last assignment, the other that can issue them at twice the rate.

You should work in teams of two. I would like you to have the experience of working with a different partner, but if your work and travel schedules make this too inconvenient, working with the same person is an option. Your call.

Super pipelined computer

Design a superpipelined processor that has about double the number of stages of Blis's default processor. The decode, register renaming, issue, register read, and commit stages can be superpipelined. Since some of today's processors can execute in half a cycle, don't add more stages to any of the execute stages. The fetching part of the fetch stage can also be divided as well, by changing the I-cache hit latency. Mark Oskin says that handling branch prediction in multiple stages may be tricky ("hairy" was the exact word he used), so, although that would be a much more realistic design, you can choose not to do it if you want. Spread your new stages out evenly over the old stages.

Assessing all level-1 caches should now take at least 2 cycles, and the miss penalties of all caches should double as well.

Stages can be added by manipulating the SuperScalarProcessor object and inserting additional DelayStage objects into the design. The procedure Finalize (located in Component/Core/ instruction.cxx) stipulates hints to the instruction scheduler as to when it can schedule dependent instructions. Use it to determine delays between instructions and to throttle back the number of stages of forwarding that can be done in a single cycle. Evan will undoubtedly email other guidelines if you need them.

Because you are superpipelining, the processor cycle time will be half that of the base machine. This is a very rough estimation that ignores some of the effects of latch overhead and clock skew.

Super-wide-issue computer

For the super-wide-issue machine, you will want the capability of issuing up to eight instructions per cycle. Because of the extra time needed to bus register values from the register file to the functional units, the cycle time of this implementation should be 1.4 times the cycle time of the base processor. This is a rough estimate based on the additional time needed to access the instruction queue when doubling issue width. We are assuming it applies to all the hardware data structures.

Both

Both implementations should use your tournament branch predictor (which I'm assuming was the best performer of the ones you evaluated).

You can take some leeway in designing your superpipelined and super-issue-width processors. Not all of you will choose exactly the same designs. Deciding upon a particular implementation is part of the exercise.

Evaluation

Evaluate the performance effect of your superpipelining and super-issue-width implementations relative to Blis's base processor. Assuming that the cycles times of both the super-issue-width processor and the superpipelining processor change to reflect the implementations you decide upon, which has the better overall bottomline performance? To what do you attribute the differences in performance? Show data from component metrics that support your analysis. Where are the stalls and why does the processor stall? Be creative here. (You may need to implement additional metrics to do this. Be sure that you completely understand the metrics that the simulator currently calculates.)

The write-up

Write a short report in the style of technical papers that explains your findings.