ECE 252 / CPS 220 Homework #3

Due in class on Wednesday, October 7

Total Points 80

Explain all of your answers to get full credit!

Dynamic ILP

1) (10 points) H&P 2.7

2) (10) Give a short example of assembly code that is not helped at all by dynamic scheduling (as compared to in-order execution). Explain why dynamic scheduling does not help its performance.

3) (10) Some researchers have proposed pipelining wakeup/select into more than one pipeline stage, in order to allow it to take more time (in nanoseconds) without impacting the clock period. How can pipelining wakeup/select degrade performance?

4) (10) In most microprocessors, globally-viewable status registers (like the processor status word) that are implicitly read by almost all instructions are considered not renameable. (a) Why would architects choose not to rename these registers? (It's possible to rename them, but architects generally choose not to.) (b) If these registers don't get renamed, then explain how one way in which the pipeline could correctly deal with an instruction that writes to one of these registers. You must assume that the core is out-of-order and renames all other registers.

5) (10) The Pentium4 paper discusses how the processor speculatively schedules (wakes up) instructions that are dependent on a load that issues. (a) What exactly is the processor predicting? (b) How does this speculation help performance? (c) In case of a misprediction, what must the processor do to recover (i.e., hide the impact of the mis-speculation)?

6) (10) Several architects have proposed register-file caches. The idea is to have a very large register file which would be too big to access in a single cycle. However, by putting a smaller cache of registers in front of it, the goal is to achieve single-cycle access in the common case (for register-file cache hits). This is not unlike the 2-level store queue in the Checkpoint Processing and Recovery paper, but for the register file instead of the store queue. Explain how a core would use a register file cache, including what happens on register file cache misses and how/when you'd choose to move registers in/out of the register file cache.

Dynamic Scheduling with SimpleScalar

7) (20) Start with the sim-outorder simulator and just use the gcc and go benchmarks. You will NOT have to modify the sim-outorder.c code for this assignment (and thus you don't need to turn in any code), but you will have to feed it different command line parameters to configure it. If you run sim-outorder without any input parameters, it will spit out all of the possibilities, which should help you to figure out how to specify the configurations in the following experiments.

Experiment #1: Compare in-order versus out-of-order execution (hint: the default is out-of-order, and there's a flag that can change this). Don't change any other flags. What do you observe?

For the rest of the experiments, assume an out-of-order core. Use the default parameter values for any parameters not discussed in the question.

Experiment #2: Evaluate the importance of the RUU size, by comparing a size of 16 vs. a size of 32 vs. a size of 64. As with all experiments here, don't change anything else. Explain your results - that is, why did the changes in RUU size have a small/big impact?

Experiment #3: Evaluate the importance of superscalar width by comparing a 3-wide to a 6-wide. Remember that you want to balance the widths of decode, issue, execute, and commit (i.e., the pipeline should either be 3-wide at all stages or 6-wide at all stages). Is the performance benefit of going from 3-wide to 6-wide worth the hardware and power costs? Explain why or why not.

Experiment #4: For a 3-wide pipeline, evaluate the impact of the number of integer ALUs by comparing a processor with 2 to a processor with 3. What does this result tell you about the number of ALUs necessary to avoid structural hazards?

Analysis: Given what you learned from these 4 experiments, explain (a) which design decisions (of the 4 you explored in these experiments) are most important for performance and (b) where you might choose a lesser performing design point for reasons of power-efficiency (performance per watt[1]) and cost-effectiveness. Justify your answers!

1. In general, having more hardware leads to more power consumption. Larger storage structures consume more power. If you have questions about power-efficiency, please ask.