Dynamic ILP

1) (10 points) H&P 2.7

2) (10 points) Give a short example of assembly code that is not helped much by dynamic scheduling. Explain why dynamic scheduling does not help its performance.

3) (10 points) Compare the Intel P6 style of renaming to the R10K style of register renaming. What are the advantages and disadvantages of each?

4) (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?

5) (10 points) 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 due to recover (i.e., hide the impact of the mis-speculation)?

6) (20 points) In 1997, Avinash Sodani and Guri Sohi published a paper on “instruction reuse.” The idea is that a given program often dynamically executes the same (static) instructions with the same inputs. They developed an instruction reuse buffer (IRB) that holds recently committed instructions; each entry in the IRB holds the instruction’s PC, input values, and output values. When an instruction executes, we look up its PC and input values in the IRB. If there’s a match, we don’t have to execute the instruction. The processor simply reads the output value from the IRB and uses that value. What benefits does instruction reuse provide for an out-of-order superscalar processor? Are these benefits of a function of the IRB hit rate? What costs does it incur? Is it likely to provide big advantages (compared to processors without instruction reuse)? Consider latency, throughput, power consumption, and hardware costs. Consider the impact of reuse on all pipeline stages.

Dynamic Scheduling with SimpleScalar

7) (30 points) 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.

Experiment #2: Evaluate the importance of the RUU size, by comparing 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 change in RUU size have a small/big impact?

Experiment #3: Evaluate the importance of superscalar width by comparing a 4-wide to an 8-wide. Remember that you want to balance the widths of decode, issue, execute, and commit (i.e., the pipeline should either be 4-wide at all stages or 8-wide at all stages). Is the performance benefit of going from 4-wide to 8-wide worth the hardware and power costs? Explain why or why not.

Experiment #4: For a 4-wide pipeline, evaluate the impact of the number of integer ALUs by comparing a processor with 2 to a processor with 4. 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.