Computer Systems Architecture
Assignment 6 Solution
Due: Tuesday, May 27

SMT should speculate! I've constructed the analysis as a list of points you could have made in your report.

- Throughput on the SMT is high, 5.2 IPC in a machine that has 6 integer units (the statistics were gathered on an integer workload). It is possible that, given such a high throughput, that wrong-path speculative instructions might be displacing correct-path speculative instructions.
- Speculative instructions on an SMT comprise the majority of instructions fetched, executed and committed. This can be seen in a variety of ways. 57% of fetch IPC, 53% of instructions issued to the functional units, and 52% of commit IPC are speculative. At any given time, more than half of the hardware contexts (4.7) are speculating. Most of the instructions fetched are speculative (61%). Given the magnitude of these numbers and the accuracy of today's branch prediction hardware (88% in this case), it would be extremely surprising if ceasing to speculate or significantly reducing the number of speculative instructions improved performance. Remember that the alternative is to stall until the branch condition is determined; for this simulation that was 10 cycles on average – that's a long time to stall.
- The data show that speculation is not particularly wasteful of hardware resources on SMT. Branch prediction accuracy in this experiment was 88% and only 9.7% of fetched instructions were flushed from the pipeline. 73% of these wrong-path-speculative instructions were removed from the pipeline *before* they reached the functional units, only consuming resources in the form of integer instruction queue entries, renaming registers, and fetch bandwidth. Both the instruction queue (IQ) and the pool of renaming registers are adequately sized: the IQ is only full 4.3% of cycles and renaming registers are exhausted only 0.3% of cycles. Thus, IQ entries and renaming registers are not highly contended. This leaves fetch bandwidth as the only resource that speculation wastes significantly and suggests that modifying the fetch policy might improve performance. But it might not. We can't tell that from our data.
- Since SMT fetches from *each* thread only once over 5.4 cycles on average (as opposed to almost every cycle for the single-threaded superscalar (every 1.4 cycles)), it speculates less aggressively past branches (on average, past 1.4 branches, compared to 3.3 branches on the superscalar). This causes the percentage of speculative instructions fetched to decline from 94% on the superscalar to 61% on SMT. More important, it also reduces the percentage of speculative instructions on the wrong path (9.7% for SMT versus 25.8% for the superscalar). Because an SMT processor makes less progress down speculative paths, it avoids multiple levels of speculative branches which impose higher (compounded) misprediction rates. For the SPECInt95 benchmarks 19% of speculative instructions on SMT are wrong path, compared to 28% on a superscalar. Therefore, SMT receives significant benefit from speculation at lower cost, compared to a superscalar.

The data also confirm the superscalar's need to speculate.

- Instruction throughput on the superscalar is low (2.8 IPC) even with speculation. This argues that speculative instructions are filling otherwise empty instructions slots and that leaving them empty will only lower IPC.
- The superscalar speculates the vast majority of the time: .9 hardware contexts (remember, there is only one!) are executing speculative instructions on average and 94% of fetched instructions are speculative. This means that speculating is more important to a superscalar. The high branch prediction accuracy (92%) shows that this large amount of speculation works out.

Without speculation SMT's IPC drops to 4.2 (data obtained in another simulation which you did not see, because it would have given away the store).