

California Institute of Technology  
Department of Computer Science  
Computer Architecture

CS184b, Winter 2000

Assignment 4: Control Issues

Tuesday, January 23

**Due:** Tuesday, January 30, 5:00PM**Part A**

1. Formulate a test program (kernel) which will allow you to estimate branch timing details (*i.e.* stall cycles on taken and non-taken branches). Run your test program against the scalar, pipelined simplescalar and report the results. Include your test program and a brief explanation with your writeup.
  - If you write your test kernel in C, you may want to force `gcc` to produce assembly and look at that. (use the `-S` option to `gcc` to get it to write out a symbolic assembly file (`.s`) rather than assembling through to a binary (`.o`)).
  - Alternately, you may want to write the test loop in assembly. (Personally, I'd compile some C down to a `.s` file just to see what the assembly looks like for reference; then maybe use the startup and data code broiler plate from C so I didn't have to write all that from scratch).
2. What is the impact of branches (and branch microarchitecture) on the CPI of your program running on the scalar, pipelined simplescalar using static branch prediction.
  - Use your parameters from problem 1.
  - I expect an answer like 0.3 CPI (*e.g.* 1.2 CPI with no branch effects and 1.5 CPI with branch effects).
  - You'll get effects from branch prediction (compare best of taken, not taken with perfect) and branch delay slots.
3. Let's explore the role of exceptions. Consider a machine without hardware exception support. How would the code have to be different to support the same functional power as a machine with exceptions?
  - Specific let's focus on page faults. Instead of generating a fault, you have an instruction, `TSTPP SRC,RES`, which tests if `SRC` is on a given page and putting the result in `RES`. Let's assume you can call a page-fault handler directly as a procedure call which takes the faulting address whenever you need to handle this condition. How would this change a typical code sequence (use the sequence from HP3.1 for a concrete example). You may have to make additional assumptions about the architecture to make this work; state your assumptions.

- Since page faults are infrequent, how would you further (or alternately) change the ISA and/or microarchitecture to amortize out the costs implied by your solution above. You don't need to give a detailed solution, just sketch the basic idea, instructions, and the key structural components of the solution. ( *e.g.* you can just identify the hazards and bypass cases which would need to be handled, but you don't need to give the detailed condition equations.)

**Part B**

Work the following problems from Hennessey and Patterson:

- 4.3
- 4.4
- 4.6