

CS 573
Midterm Exam
March 16, 2009

The following exam is open book and open notes. You may feel free to use whatever additional reference material you wish, but **no electronic aids** are allowed. Please note the following instructions. There will be a ten point deduction for failure to comply with them:

- start each problem on a new sheet of paper
- write your Banner ID number, but not your name, on each sheet of paper you turn in

Also, please note the following:

- show your work whenever appropriate. There can be no partial credit unless you show how you derived your answers
- be succinct. You may lose points for facts that, while true, are not relevant to the question at hand

You have until 2:30 to finish the exam. The questions are equally weighted.

1. The VAX used a very general “three-address” instruction format: an arithmetic instruction could specify the locations of both of its operands, as well as its result, separately. In many cases, all three operands would be local scalar variables.
 - (a) Assuming local variables are in memory on the stack, what addressing mode would be used to read the operands and write the result (yes, I’m expecting the same mode for the operands and the destination)?
 - (b) Discuss how an arithmetic instruction using this addressing mode can be emulated by a sequence of four or less instructions in a RISC-style (*i.e.* three operand, load-store) architecture.
 - (c) It can be anticipated that the RISC-style sequence might be easier to optimize than the VAX. Why? (of course, it would also be possible to optimize the VAX. However, identifying opportunities for optimization might be a little bit more difficult).
2. The Pentium 4 implements register renaming using what amounts to two sets of pointers. One set (the Frontend RAT) is used to allocate new physical registers as architected registers occur as results in the code stream, and to identify dependencies. The second set (the Retirement RAT) is used to identify architected registers with physical registers in retired instructions.

Suppose a program encounters a page fault. Some of the instructions which occurred before the faulting instruction in the code stream haven’t executed yet, while some instructions following it have completed (but have not been retired, since it uses in-order retirement).

Outline what the processor should do to ensure that a precise interrupt occurs, and the page fault is seen to be associated with the correct instruction.
3. After the sorting program in HW2 had been running for a while, it settled on the following behavior for the branch instructions:
 - The swap branch was never taken.
 - The inner loop branch was almost always taken, however, occasionally, it would not be taken (so there would be two branches in a row not taken), after which the outer loop branch would be taken.

A classmate has come to you, very excitedly describing an idea for a new branch predictor: a single two-bit saturating counter (so it’s a lot like Smith’s Strategy 7, except that there’s only a single counter instead of a table of counters).

How well would you anticipate this branch predictor would work on the code from HW2?

4. The standard example used to demonstrate pathological behavior in a two-way set-associative cache is the loop

```
for (i = 0; i < SIZE; i++)  
    a[i] = b[i] + c[i];
```

when $a[i]$, $b[i]$, and $c[i]$ all map to the same cache index, and a cache line is larger than an array element.

- (a) Describe why this code has a data cache hit rate of virtually 0.
 - (b) Predict how the indirect indexed cache you simulated on HW3 will behave on this code. What is the expected mean number of collision chain probes? Assume the cache line is *much* larger than an array element – *i.e.*, enough larger that you don't need to consider the first three misses on each cache line.
5. Under what circumstances might a hashed page table contain two entries with the same physical frame number, but different virtual page number? (no, “there's a bug” is not the answer I'm looking for!)