

CS 573

Midterm Exam

Solutions

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 social security number, but not your name, on each sheet of paper you turn in
- show your work whenever appropriate. There can be no partial credit unless I see how answers were arrived
- be succinct. You may lose points for facts that, while true, are not relevant to the question at hand

You have until 12:20 to finish the exam. The questions are equally weighted.

1. Register renaming was originally created as a way to let a machine with a small number of registers perform out-of-order execution.
 - (a) Why is out-of-order difficult to implement on a machine with only a small number of registers, especially a two-operand architecture like Intel's, without doing register renaming?
With only a few registers, you can only specify a few possible result registers simultaneously, so you can't maintain many instructions in flight.
 - (b) RISC instruction sets like MIPS have a large number of registers. However, when out-of-order execution is implemented on them, they also frequently implement register renaming. Why?
While a RISC machine has many registers, some of them are used a lot and end up as the result of many instructions (examples are the stack pointer and the register used for function returns). So while there are many registers available to use, you only end up with a few actually in flight.
2. In imitating the IA-32 in the assignment, I used a two-operand ISA instruction set with a very limited number of registers.
 - (a) How much easier would it be to write code for a two-operand computer with many more registers?
It would help a lot. The individual instructions wouldn't be any more flexible, but you could still get a lot more "global" flexibility since you've got a lot of freedom to use a lot of registers.
 - (b) How much easier would it be to write code for a three-operand computer with only four registers?
Wouldn't make much difference. Since you've only got a few registers for results, you're still in a straitjacket.
3. If I had included branch instructions in the computer used in the homework assignment, they would have been based on condition codes (like IA-32 really is). This poses a problem with out-of-order execution, of course, since the last instruction executed before a branch takes place may not be the last instruction in the program text before the branch. How could you keep track of what instruction is going to provide condition codes to a branch?
As you decode each instruction, mark the last one decoded as the CC source. When a conditional branch is encountered, wait for the current CC source to provide condition codes.

4. The Average Memory Access Time (AMAT) of a memory system with a one-level cache is

$$AMAT = T_c + (1 - H)T_m$$

where

T_c is the access time for the cache

H is the cache hit rate (the probability that a memory access will find its data in the cache)

T_m is the memory access time

(this definition is based on a memory system in which an attempt is made to locate data in the cache first, and then goes out to memory if it did not find the data in the cache).

Suppose switching from a direct-mapped cache to a two-way set-associative cache increases T_c by 10%. As a function of T_m , how much better does the hit rate have to be for the change to show a performance improvement? Your answer should express the necessary difference in the hit rates as a function of T_c and T_m .

$$\begin{aligned} 1.1T_c + (1 - H_2)T_m &< T_c + (1 - H_1)T_c \\ .1T_c &< (1 - H_1)T_m - (1 - H_2)T_m \\ .1T_c &< (H_2 - H_1)T_m \\ \frac{T_c}{10T_m} &< (H_2 - H_1) \end{aligned}$$