

15740-18740 Computer Architecture, Fall 2010
Midterm Exam II

Instructor: Onur Mutlu

Teaching Assistants: Evangelos Vlachos, Lavanya Subramanian, Vivek Seshadri

Date: November 22, 2010

Name:

Problem 1 (60 points)	:	<input type="text"/>
Problem 2 (30 points)	:	<input type="text"/>
Problem 3 (18 points)	:	<input type="text"/>
Problem 4 (20 points)	:	<input type="text"/>
Problem 5 (24 points)	:	<input type="text"/>
Problem 6 (24 points)	:	<input type="text"/>
Bonus Problem 7 (24 points)	:	<input type="text"/>
Legibility and Name (5 points)	:	<input type="text"/>
Total (181 + 24 points)	:	<input type="text"/>

Instructions:

1. This is a closed book exam. You are allowed to have two letter-sized cheat sheets.
2. No electronic devices may be used.
3. This exam lasts 1 hour 50 minutes.
4. Clearly indicate your final answer.
5. Please show your work when needed.
6. Please write your name on every sheet.
7. Please make sure that your answers to all questions (and all supporting work that is required) are contained in the space provided.

Problem 1: Potpourri (60 points)

1) Always Not Taken (6 points)

A processor employs a static branch prediction technique which always predicts a conditional branch to be not taken. Vivek runs a set of applications on this processor and determines the conditional branch prediction accuracy to be 30%.

a) What can be done to improve conditional branch prediction accuracy for this set of applications without changing the processor?

b) What other benefits does your scheme provide (if anything else)? Be precise.

2) Cache Logic (6 points)

Assume a 32-bit address space, a cache size of 64KB and a cache block size of 64 bytes.

a) How many tag comparators do you need to build this cache, if the cache is direct-mapped?

b) How many tag comparators do you need to build this cache, if this cache is 4-way set-associative?

c) How many tag comparators do you need to build this cache, if this cache is fully associative?

3) DRAM Refresh (3 points)

Why is there a need to perform “refresh” operations in DRAM?

4) DRAM Scheduling (6 points)

What is the purpose of request batching in DRAM scheduling (think about parallelism aware batch scheduling)?

Where else can the idea of request batching be applied? Be brief, but specific.

5) Execution-based Prefetching (9 points)

In lecture, we talked about the idea of creating a “thread” for pre-executing a piece of the program solely for prefetching purposes. Assume we have a mechanism that creates such threads, which we called “speculative threads” in lecture, as opposed to the “main program thread” which is the original application. These “speculative threads” can be executed

1. on a core separate from the core where the main program thread is running.
2. on a separate hardware thread context in the same core where the main program thread is running.
3. or on the same hardware context in the same core where the main program thread is running, when the main program thread is idle due to long-latency cache misses.

Option 1 (Separate core)

Advantage:

Name:

4

Disadvantage:

Option 2 (Separate hardware thread context on the same core)

Advantage:

Disadvantage:

Option 3 (On the same hardware thread context during idle cycles)

Advantage:

Disadvantage:

6) Superscalar Decode (10 points)

Suppose you are building a superscalar processor that can issue 4 x86 instructions in parallel. You find that decoding takes a significant latency. Explain what characteristic of the x86 ISA causes this. Why?

Briefly describe one technique that reduces the latency of the x86 multiple instruction decoder.

What are the disadvantages of the technique?

7) Trace Caches (10 points)

What does branch promotion allow that enhances the performance of a trace cache?

How is the promoted branch handled at fetch and execute stages of the processor?

A curious scientist discovers that the branch prediction accuracy on a processor with a trace cache that employs branch promotion is higher than that on a processor with a trace cache that does not employ branch promotion. Explain why this is the case.

8) Predicated Execution (5 points)

Assume the following branch is predicted perfectly (100% accuracy) when it is executed on a 8-wide superscalar, out-of-order processor.

```
if (a == b) {  
    x = 1;  
} else {  
    x = 3;  
}
```

Would you ever want to convert this code into predicated code? Circle one: YES NO
Why, why not? Explain clearly.

9) Prefetching Algorithms (5 points)

Recall that content directed prefetching identifies pointers in a fetched cache line and generates prefetch requests for those pointer addresses. What are two advantages this prefetching mechanism has that Markov and Stream prefetchers do not have?

Problem 2: Multi-core Cache Partitioning (30 points)

Suppose we have a system with 32 cores that are sharing a physical second-level cache. Assume each core is running a single single-threaded application, and all 32 cores are concurrently running applications. Assume that the page size of the architecture is 8KB, the block size of the cache is 128 bytes, and the cache uses LRU replacement. We would like to ensure each application gets a dedicated amount of cache space in this shared cache without interference from other cores. We would like to enforce this using the OS-based page coloring mechanism to partition the cache, which we discussed in lecture. (Hint: the operating system ensures, using virtual memory mechanisms, that the applications do not contend for the same space in the cache)

a) What is the minimum size the L2 cache needs to be such that each application is allocated its dedicated space in the cache via page coloring? Show your work. (4 points)

b) Assume the cache is 4MB, 32-way associative. Can the operating system ensure that the cache is partitioned such that no two applications interfere for cache space? Show your work. (4 Points)

c) Assume you would like to design a 32MB shared cache such that the operating system has the ability to ensure that the cache is partitioned such that no two applications interfere for cache space. What is the minimum associativity of the cache such that this is possible? Show your work. (4 points)

d) What is the maximum associativity of the cache such that this is possible? Show your work. (4 points)

e) Suppose we decide to change the cache design and use utility based cache partitioning (UCP) to partition the cache, instead of OS-based page coloring. Assume we would like to design a 4MB cache with a 128-byte block size. What is the minimum associativity of the cache such that each application is guaranteed a minimum amount of space without interference? (3 points)

f) Is it desirable to implement UCP on a cache with this minimum associativity? Why, why not? Explain. (3 points)

g) What is the maximum associativity of a cache that uses UCP such that each application is guaranteed a minimum amount of space without (4 points) interference?

h) Is it desirable to implement UCP on a cache with this maximum associativity? Why, why not? Explain. (4 points)

Problem 3: Where to Prefetch into? (18 points)

Lavanya, Vivek, and Evangelos are evaluating the design decisions for a new prefetcher to be implemented on their new single-core system with a two-level cache hierarchy. Lavanya designs a prefetcher that places the prefetched cache blocks into the L1 cache. Vivek modifies the design such that the prefetcher places the prefetched blocks into the L2 cache instead of the L1 (all else remains the same).

a) On workload A, Lavanya's design performs better than Vivek's design. Is this possible?

Circle one: YES NO

Explain why. (6 points)

b) On workload B, Vivek's design performs better than Lavanya's. Is this possible? Circle one: YES NO

Circle one: YES NO

Explain why. (6 points)

c) Evangelos modifies the design such that the prefetcher places the prefetched blocks into a separate prefetch buffer accessed in parallel with the L2 cache. To his surprise, he finds that this design degrades performance compared to both Lavanya and Vivek's designs on both workloads A and B. Is this possible? Circle one: YES NO

Explain clearly why this could happen, if it is possible. (6 points)

Problem 4: Register File (20 points)

a) You are asked to design a 16-wide-issue superscalar machine. Assuming each instruction sources two register operands and writes to one destination register, how many read and write ports are needed in the register file? (State your assumptions, if needed, clearly) (2 points)

b) Why is having so many ports to the register file undesirable? (3 points)

c) In class, we discussed several techniques to reduce the port requirements into the register file. Describe one such technique. (5 points)

d) Describe how and why it reduces number of ports into the register file? (5 points)

e) What additional complexities or overheads does the technique introduce that were not present in the baseline processor? (5 points)

Problem 5: DRAM Read-Write Switching Penalty (24 points)

The DRAM controller needs to service both read and write requests to DRAM. The read requests are load misses in the last level cache and the write requests are writebacks from the last level cache. At any point in time, the controller will likely have a set of read and write requests to be serviced. Whenever a read request is followed by a write request or vice versa, the controller has to stall for some number of DRAM cycles during which it cannot issue any requests to DRAM.

In a modern DRAM, switching from a read to write takes 2 DRAM cycles and switching from a write to read takes 6 DRAM cycles. Assume that there is only one bank in the system and a row-buffer hit takes 14 DRAM cycles and a row-buffer conflict takes 34 DRAM cycles. At some point in time, the following requests are in the DRAM controller queue. A request is represented as (Operation row-number, column-number). The leftmost request is the oldest request and the rightmost is the youngest.

Rd 1,1 ← Wr 2,3 ← Rd 3,1 ← Wr 4,7 ← Rd 5,9 ← Wr 5,5

a) A memory controller does not distinguish between reads and writes and simply uses FR-FCFS (row-hit first scheduling policy). Determine the total number of DRAM cycles taken to service the above requests in the request queue. Assume that the first scheduled request is a row conflict and has no switching penalty. Show your work for partial credit. (4 points)

b) Assume you modified the FR-FCFS scheduler such that it always prioritizes write requests (and everything else remains the same). Determine the total number of DRAM cycles taken to service the above requests. Show your work clearly. (4 points)

c) Is prioritizing write requests (as in (b)) a good idea? Why, why not? (4 points)

d) You are now given the task of improving the scheduler. Describe how you would design the controller such that the system performance is maximized in the presence of read-write penalty. Describe all modifications you would make (to the scheduling mechanism, buffering, etc.) to make your solution viable and implementable. (8 points)

e) Determine the total number of DRAM cycles taken to service the above requests in the request queue, with your new design. Show your work clearly. (4 points)

Problem 6: Hardware/Software Cooperation (24 points)

In class, we have talked about several hardware/software cooperative techniques, solving different problems. Pick your favorite (if you do not have a favorite, pick one you feel the most comfortable explaining to us – we will call it your favorite). Explain how it works by answering the following questions, clearly focusing on the technique's advantages and disadvantages (make sure your comparison points are clear).

a) The name of the technique: (2 points)

b) The problem the technique solves (be specific): (4 points)

c) Basic idea of the technique (be clear – we are looking for the high level idea and insight, not implementation details): (5 points)

d) What is an alternative technique to solve the same problem. Briefly explain the key idea of the technique (be clear) (5 points)

Name:

15

e) Advantages of your favorite technique over the alternative: (4 points)

f) Disadvantages of your favorite technique over the alternative: (4 points)

Bonus (Predicated Execution)

Predicated execution converts control dependencies into data dependencies, thereby eliminating branches. The following is a piece of code with a single branch in it.

```
if (a == c) {
    x = a;
}
else {
    y = b;
}
z = x + y;
```

Assume the following mnemonics:

1. MOV R1, R2 *// Copy the value of R2 to R1*
2. ADD R1, R2, R3 *// R1 = R2 + R3*
3. BEQ label, R1, R2 *// If (R1 == R2), jump to label*
4. JMP label *// Jump to label unconditionally*
5. SETEQ p1, R1, R2 *// If R1 == R2, set p1 to true; else, set p1 to false*
6. p1 INSTRUCTION *// If p1 is true, execute INSTRUCTION; else, NOP*
7. !p1 INSTRUCTION *// If p1 is false, execute INSTRUCTION; else, NOP*
 // INSTRUCTION can be either instruction 1 or 2

Assume the following mapping between variables and registers:

a → R1, b → R2, c → R3
x → R4, y → R5, z → R6

a) Write an assembly code equivalent to the above program using conditional branches (you can use only instructions 1, 2, 3 and 4). (3 points)

b) Write an assembly code equivalent to the above program using predicated instructions (you can use all instructions except 3 and 4). (5 points)

c) Briefly explain how predicated execution affects each of the following stages of the super-scalar pipeline.

Fetch (4 points)

Dependency Check (as part of Decode) (4 points)

Rename (4 points)

Commit (4 points)

Name:

Name:

Name: