18-447 Intro to Computer Architecture, Spring 2012
Final Exam

Instructor: Onur Mutlu
Teaching Assistants: Chris Fallin, Lavanya Subramanian, Abeer Agrawal
Date: May 10, 2012

Name:

Problem I (190 Points)	:	
Problem II (60 Points)	:	
Problem III (60 Points)	:	
Problem IV (30 Points)	:	
Problem V (45 Points)	:	
Problem VI (60 Points)	:	
Legibility and Name (2 Points)	:	
Total (447 Points)	:	

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 3 hours.
4. Clearly indicate your final answer.
5. Please show your work when needed.
6. Please write your initials on every odd page.
7. Please make sure that your answers to all questions (and all supporting work that is required) are contained in the space provided.

- **Be cognizant of time.** Do not spend too much time on one question.
- **Be concise.** You will be penalized for verbosity.
- **Show work when needed.** You will receive partial credit at our discretion.
- **Write legibly.** Show your final answer.

I. Potpourri (190 Points)

1) Superscalar and Out-of-Order Processing (10 Points)

Please explain the following concepts, each in no more than 20 words:

Superscalar processing:

Out-of-order execution:

Please answer the questions below:

(a) Are the concepts of superscalar processing and out-of-order execution orthogonal? **YES** **NO**

Why or why not? Explain and justify in no more than 20 words.

(b) Assume your friend at a processor design company designed a 1-instruction-wide processor with out-of-order execution. Every instruction in this machine takes a single cycle.

What would you suggest to your friend to simplify the design of the above processor?

2) Out-of-Order Execution (10 points)

Assume the execution latency of the longest-latency instruction in a 4-wide superscalar, out-of-order machine implementing Tomasulo's algorithm is 1000 cycles.

How large should the instruction window be such that the decode of instructions does not stall in the presence of this longest-latency instruction?

Assume we would like to sustain 4 executed instructions per cycle, each with two source registers at most, and we have 4 general-purpose functional units. How many total tag comparators are needed in the reservation stations to support this in this machine? (Each tag comparator can perform only one comparison per cycle.)

3) Register versus Memory Dependences (10 Points)

What is the fundamental difference between dependencies through registers and dependencies through memory? Explain in no more than 20 words.

4) DRAM Chips (5 Points)

In class we discussed that the row address and the column address are sent at different times to the DRAM chip. Why is this so?

5) Cache Replacement (15 Points)

In class, we discussed a cache replacement algorithm called Belady's OPT, published by Belady in 1966. Belady's OPT chooses the victim block such that it is the block that is going to be referenced furthest in the future by the program.

(a) Suppose you want to design a processor that implements this algorithm. How would you design this processor?

(b) Is this algorithm optimal in minimizing cache miss rate? Circle one: **YES** **NO**

Explain briefly why or why not:

(c) Is this algorithm optimal in minimizing the execution time of a program? Circle one: **YES** **NO**

Explain briefly why or why not:

6) VLIW vs. Superscalar vs. Array Processor (20 Points)

(a) What do VLIW, superscalar execution, and array processing concepts have in common?

(b) Provide two reasons why a VLIW microarchitecture is simpler than a "same-width" superscalar microarchitecture:

(c) Provide two reasons why a superscalar microarchitecture could provide higher performance than a "same-width" VLIW microarchitecture:

(d) Provide a reason why a VLIW processor is more flexible than an array processor:

(e) Provide a reason why an array processor is simpler than a "same-width" VLIW processor. Explain.

7) CRAY-1 (5 Points)

In lecture we discussed that, CRAY-1, a fast vector processor, had the fastest scalar unit of its time (circa early 1970s). Why so? Explain in less than 20 words.

8) Bloom Filters (15 Points)

Consider a Bloom filter with a bit vector v consisting of m bits. This Bloom filter uses k hash functions $h_1, \ldots, h_k$ that map any element e to a bit index between 0 and $m-1$. To represent that an element e is a member of the set, the Bloom filter sets the k bits at indices $h_1(e), \ldots, h_k(e)$ to 1. An element is considered part of the set if and only if all k bits to which the hash functions map the element are set to 1.

(a) Let $m = 8$ and $k = 2$. Define hash functions $h_1(x) = x \bmod 8$, and $h_2(x) = \lfloor \frac{x \bmod 16}{2} \rfloor$. Show the contents of the Bloom filter's bit vector below after we add the elements 8, 9 and 4.

Bit Index	Value
0	
1	
2	
3	
4	
5	
6	
7	

(b) Find an element e, $e \notin \{4, 8, 9\}$, such that the Bloom filter with the hash functions and bit vector above indicates that e is in the set. Write e below:

9) Update vs. Invalidate Based Coherence (5 Points)

In a multiprocessor system with private L1 caches and a shared L2 cache, if shared data is read by all cores every cycle, but written once per 1000 cycles by a single core, what kind of coherence protocol would you use?

UPDATE **INVALIDATE**

Why?

10) Directory Based Coherence (5 Points)

Assume we use the directory based coherence mechanism we discussed in class. Say, for cache block A, the bit vector stored in the directory is all zeros. What does this tell you about the cache block?

11) Coherence Protocols (20 Points)

Suppose we have a multiprocessor system with 512 processors. Each processor has a 1 Megabyte private writeback cache with 64-byte cache blocks. The main memory size is 1 Gigabyte.

(a) If we implement a snoopy bus based MESI cache coherence protocol, how many bits of state do we need in the entire system for the coherence protocol implementation?

Where do these bits reside?

(b) If we instead implement a directory based cache coherence protocol *as we discussed in class*, how many bits of state do we need in the entire system for the coherence protocol implementation?

Where do these bits reside?

Which of the above two protocols would you choose for this system?

Snoopy **Directory**

Why?

12) Multiprocessing (10 Points)

What differentiates a tightly coupled multiprocessor from a loosely coupled one?

Assume you are designing a loosely coupled multiprocessor. What type of cache coherence protocol would you implement?

Why?

13) Programmer versus (Micro)architect (15 Points)

Choices made by a computer architect significantly affect the difficulty the programmer experiences in writing programs or the microarchitect experiences in designing hardware. Over the past 27 lectures this semester, we explored many concepts where this tradeoff was evident.

We discussed many techniques which made programmer's job easier while making hardware designer's job more difficult. Please list five such techniques below (no more than 5 words for each technique's name):

14) Topologies (10 Points)

Suppose you would like to connect 625 processors, and you are considering three different topologies: bus, point-to-point network, and mesh.

Describe one disadvantage of each:

A Single Bus:

A Point-to-Point Network:

A 25x25 Mesh:

Which one would you choose? Why?

15) Hot Potato Routing (10 Points)

In class, we discussed the idea of hot potato routing, introduced by Paul Baran in 1962 as a means to handle contention when two packets need to use the same output link. This idea is also called misrouting or deflection routing.

(a) What is the main benefit of hot potato routing?

(b) What is the main disadvantage of hot potato routing?

(c) Why do you think this idea is also called "misrouting"?

16) INV Bits in Runahead Execution (20 points)

(a) What is the purpose of the INV bits in a runahead execution processor?

(b) Suppose we did not have the hardware budget to have INV bits in a runahead processor. Describe how you would design a runahead processor to deal with the problem INV bits are otherwise used to solve.

(c) Give one reason why the runahead processor design you propose would have higher performance (if possible) than the runahead processor that has INV bits. If this is not possible, write "impossible".

(d) Give one reason why the runahead processor design you propose would have lower performance (if possible) than the runahead processor that has INV bits. If this is not possible, write "impossible".

17) Runahead Execution (5 Points)

A 1-wide runahead execution processor that incurs 100 cycles on an L2 miss, executes the following code segment:

LOOP:
lw $t1 ← 0($s0) //L2 Miss
lw $t2 ← 8($t1)
add $t3 ← $t1, $t2
lw $s0 ← 0($t3)
jmp LOOP

The first load (lw $t1 ← 0($s0)) is an L2 miss. When this miss becomes the oldest instruction in the reorder buffer, the processor enters runahead mode. How many L2 misses are generated in runahead mode?

II. Parallel Speedup (60 Points)

You are a programmer at a large corporation, and you have been asked to parallelize an old program so that it runs faster on modern multicore processors.

(a) You parallelize the program and discover that its speedup over the single-threaded version of the same program is significantly less than the number of processors. You find that many cache invalidations are occuring in each core's data cache. What program behavior could be causing these invalidations (in 20 words or less)?

(b) You modify the program to fix this first performance issue. However, now you find that the program is slowed down by a global state update that must happen in only a single thread after every parallel computation. In particular, your program performs 90% of its work (measured as processor-seconds) in the parallel portion and 10% of its work in this serial portion. The parallel portion is perfectly parallelizable. What is the maximum speedup of the program if the multicore processor had an infinite number of cores?

(c) How many processors would be required to attain a speedup of 4?

(d) In order to execute your program with parallel and serial portions more efficiently, your corporation decides to design a custom heterogeneous processor.

- This processor will have one large core (which executes code more quickly but also takes greater die area on-chip) and multiple small cores (which execute code more slowly but also consume less area), all sharing one processor die.
- When your program is in its parallel portion, all of its threads execute **only** on small cores.
- When your program is in its serial portion, the one active thread executes on the large core.
- Performance (execution speed) of a core is proportional to the square root of its area.
- Assume that there are 16 units of die area available. A small core must take 1 unit of die area. The large core may take any number of units of die area n^2, where n is a positive integer.
- Assume that any area not used by the large core will be filled with small cores.

How large would you make the large core for the fastest possible execution of your program?

What would the same program's speedup be if all 16 units of die area were used to build a homogeneous system with 16 small cores, the serial portion ran on one of the small cores, and the parallel portion ran on all 16 small cores?

Does it make sense to use a heterogeneous system for this program which has 10% of its work in serial sections?
YES **NO**

Why or why not?

(e) Now you optimize the serial portion of your program and it becomes only 4% of total work (the parallel portion is the remaining 96%). What is the best choice for the size of the large core in this case?

What is the program's speedup for this choice of large core size?

What would the same program's speedup be for this 4%/96% serial/parallel split if all 16 units of die area were used to build a homogeneous system with 16 small cores, the serial portion ran on one of the small cores, and the parallel portion ran on all 16 small cores?

Does it make sense to use a heterogeneous system for this program which has 4% of its work in serial sections?

YES **NO**

Why or why not?

III. VLIW (60 Points)

You are using a tool that transforms machine code that is written for the MIPS ISA to code in a VLIW ISA. The VLIW ISA is identical to MIPS except that multiple instructions can be grouped together into one *VLIW instruction*. Up to N MIPS instructions can be grouped together (N is the machine width, which depends on the particular machine). The transformation tool can reorder MIPS instructions to fill VLIW instructions, as long as loads and stores are not reordered relative to each other (however, independent loads and stores can be placed in the same VLIW instruction). You give the tool the following MIPS program (we have numbered the instructions for reference below):

(01) lw $t0 ← 0($a0)
(02) lw $t2 ← 8($a0)
(03) lw $t1 ← 4($a0)
(04) add $t6 ← $t0, $t1
(05) lw $t3 ← 12($a0)
(06) sub $t7 ← $t1, $t2
(07) lw $t4 ← 16($a0)
(08) lw $t5 ← 20($a0)
(09) srlv $s2 ← $t6, $t7
(10) sub $s1 ← $t4, $t5
(11) add $s0 ← $t3, $t4
(12) sllv $s4 ← $t7, $s1
(13) srlv $s3 ← $t6, $s0
(14) sllv $s5 ← $s0, $s1
(15) add $s6 ← $s3, $s4
(16) add $s7 ← $s4, $s6
(17) srlv $t0 ← $s6, $s7
(18) srlv $t1 ← $t0, $s7

(a) Draw the dataflow graph of the program in the space below. Represent instructions as numbered nodes (01 – 18), and flow dependences as directed edges (arrows).

(b) When you run the tool with its settings targeted for a particular VLIW machine, you find that the resulting VLIW code has 9 VLIW instructions. What minimum value of N must the target VLIW machine have?

(c) Write the MIPS instruction numbers (from the code above) corresponding to each VLIW instruction, for this value of N. When there is more than one MIPS instruction that could be placed into a VLIW instruction, choose the instruction that comes earliest in the original MIPS program.

	MIPS Inst. No.	MIPS Inst. No.	MIPS Inst. No.	MIPS Inst. No.	MIPS Inst. No.	MIPS Inst. No.	MIPS Inst. No.	MIPS Inst. No.	MIPS Inst. No.	MIPS Inst. No.
VLIW Instruction 1:										
VLIW Instruction 2:										
VLIW Instruction 3:										
VLIW Instruction 4:										
VLIW Instruction 5:										
VLIW Instruction 6:										
VLIW Instruction 7:										
VLIW Instruction 8:										
VLIW Instruction 9:										

(d) You find that the code is still not fast enough when it runs on the VLIW machine, so you contact the VLIW machine vendor to buy a machine with a larger machine width N. What minimum value of N would yield the maximum possible performance (i.e., the fewest VLIW instructions), assuming that all MIPS instructions (and thus VLIW instructions) complete with the same fixed latency and assuming no cache misses?

(e) Write the MIPS instruction numbers corresponding to each VLIW instruction, for this optimal value of N. Again, as in part (**c**) above, pack instructions such that when more than one instruction can be placed in a given VLIW instruction, the instruction that comes first in the original MIPS code is chosen.

	MIPS Inst. No.	MIPS Inst. No.	MIPS Inst. No.	MIPS Inst. No.	MIPS Inst. No.	MIPS Inst. No.	MIPS Inst. No.	MIPS Inst. No.	MIPS Inst. No.	MIPS Inst. No.
VLIW Instruction 1:										
VLIW Instruction 2:										
VLIW Instruction 3:										
VLIW Instruction 4:										
VLIW Instruction 5:										
VLIW Instruction 6:										
VLIW Instruction 7:										
VLIW Instruction 8:										
VLIW Instruction 9:										

(f) A competing processor design company builds an in-order superscalar processor with the same machine width N as the width you found in part (**b**) above. The machine has the same clock frequency as the VLIW processor. When you run the original MIPS program on this machine, you find that it executes slower than the corresponding VLIW program on the VLIW machine in part (**b**). Why could this be the case?

(g) When you run some other program on this superscalar machine, you find it runs faster than the corresponding VLIW program on the VLIW machine. Why could this be the case?

IV. Memory System (30 Points)

A machine with a 4 GB DRAM main memory system has 4 channels, 1 rank per channel and 4 banks per rank. The cache block size is 64 bytes.

(a) You are given the following byte addresses and the channel and bank to which they are mapped:

Byte: `0x0000` ⇒ Channel 0, Bank 0
Byte: `0x0100` ⇒ Channel 0, Bank 0
Byte: `0x0200` ⇒ Channel 0, Bank 0
Byte: `0x0400` ⇒ Channel 1, Bank 0
Byte: `0x0800` ⇒ Channel 2, Bank 0
Byte: `0x0C00` ⇒ Channel 3, Bank 0
Byte: `0x1000` ⇒ Channel 0, Bank 1
Byte: `0x2000` ⇒ Channel 0, Bank 2
Byte: `0x3000` ⇒ Channel 0, Bank 3

Determine which bits of the address are used for each of the following address components. Assume row bits are higher order than column bits:

- Byte on bus
  Addr [2 : 0]
- Channel bits (channel bits are contiguous)
  Addr [_____ : _____]
- Bank bits (bank bits are contiguous)
  Addr [_____ : _____]
- Column bits (column bits are contiguous)
  Addr [_____ : _____]
- Row bits (row bits are contiguous)
  Addr [_____ : _____]

(b) Two applications App 1 and App 2 share this memory system (using the address mapping scheme you determined in part (**a**)). The memory scheduling policy employed is FR-FCFS. The following requests are queued at the memory controller request buffer at time t. Assume the first request (A) is the oldest and the last one (A + 15) is the youngest.

A B A + 1 A + 2 A + 3 B + 10 A + 4 B + 12 A + 5 A + 6 A + 7
A + 8 A + 9 A + 10 A + 11 A + 12 A + 13 A + 14 A + 15

These are cache block addresses, not byte addresses. Note that requests to A + x are from App 1, while requests to B + x are from App 2. Addresses A and B are row-aligned (i.e., they are at the start of a row) and are at the same bank but are in different rows.

Assuming row-buffer hits take T time units to service and row-buffer conflicts/misses take 2T time units to service, what is the slowdown (compared to when run alone on the same system) of

i) App 1?

ii) App 2?

(c) Which application slows down more?

Why?

(d) In class, we discussed memory channel partitioning and memory request scheduling as two solutions to mitigate interference and application slowdowns in multicore systems. Propose another solution to reduce the slowdown of the more-slowed-down application, *without* increasing the slowdown of the other application? Be concrete.

V. Transient Faults (45 Points)

In class and in labs, we have implicitly assumed that the circuits in the processor are always *correct*. However, in the real world, this is not always the case. One common type of physical problem in a processor is called a *transient* fault. A transient fault simply means that a bit in a flip-flop or register incorrectly changes (i.e., flips) to the opposite state (e.g., from 0 to 1 or 1 to 0) temporarily.

Consider how a transient fault could affect each of the following processor structures. For each part of this question, answer

(i) does a transient fault affect the *correctness* of the processor (will the processor still give correct results for **any** program if such a fault occurs in this structure)?

(ii) *if the fault does not affect correctness*, does it affect the *performance* of the processor?

Valid answers for each question are "can affect" or "cannot affect". Answer for *performance* in a particular situation only if correctness is not affected. When in doubt, state your assumptions.

(a) A bit in the global history register of the global branch predictor:

Flipped from 0 to 1:

Correctness: CAN AFFECT CANNOT AFFECT

If correctness is affected, why?

Performance: CAN AFFECT CANNOT AFFECT

Flipped from 1 to 0:

Correctness: CAN AFFECT CANNOT AFFECT

If correctness is affected, why?

Performance: CAN AFFECT CANNOT AFFECT

(b) A bit in a pipeline register (prior to the stage where branches are resolved) that when set to '1' indicates that a branch should trigger a flush due to misprediction, when that pipeline register is currently holding a branch instruction:

Flipped from 0 to 1:

Correctness: CAN AFFECT CANNOT AFFECT

If correctness is affected, why?

Performance: CAN AFFECT CANNOT AFFECT

Flipped from 1 to 0:

Correctness: CAN AFFECT CANNOT AFFECT

If correctness is affected, why?

Performance: CAN AFFECT CANNOT AFFECT

(c) A bit in the state register of the microsequencer in a microcoded design:

Flipped from 0 to 1:

Correctness: CAN AFFECT CANNOT AFFECT

If correctness is affected, why?

Performance: CAN AFFECT CANNOT AFFECT

Flipped from 1 to 0:

Correctness: CAN AFFECT CANNOT AFFECT

If correctness is affected, why?

Performance: CAN AFFECT CANNOT AFFECT

(d) A bit in the tag field of a register alias table (RAT) entry in an out-of-order design:

Flipped from 0 to 1:

Correctness: CAN AFFECT CANNOT AFFECT

If correctness is affected, why?

Performance: CAN AFFECT CANNOT AFFECT

Flipped from 1 to 0:

Correctness: CAN AFFECT CANNOT AFFECT

If correctness is affected, why?

Performance: CAN AFFECT CANNOT AFFECT

(e) The dirty bit of a cache block in a write-back cache:

Flipped from 0 to 1:

Correctness: CAN AFFECT CANNOT AFFECT

If correctness is affected, why?

Performance: CAN AFFECT CANNOT AFFECT

Flipped from 1 to 0:

Correctness: CAN AFFECT CANNOT AFFECT

If correctness is affected, why?

Performance: CAN AFFECT CANNOT AFFECT

(f) A bit in the application id field of an application-aware memory scheduler's request buffer entry:

Flipped from 0 to 1:

Correctness: CAN AFFECT CANNOT AFFECT

If correctness is affected, why?

Performance: CAN AFFECT CANNOT AFFECT

Flipped from 1 to 0:

Correctness: CAN AFFECT CANNOT AFFECT

If correctness is affected, why?

Performance: CAN AFFECT CANNOT AFFECT

(g) Reference bit in a page table entry:

Flipped from 0 to 1:

Correctness: CAN AFFECT CANNOT AFFECT

If correctness is affected, why?

Performance: CAN AFFECT CANNOT AFFECT

Flipped from 1 to 0:

Correctness: CAN AFFECT CANNOT AFFECT

If correctness is affected, why?

Performance: CAN AFFECT CANNOT AFFECT

(h) A bit indicating the processor is in runahead mode:

Flipped from 0 to 1:

Correctness: CAN AFFECT CANNOT AFFECT

If correctness is affected, why?

Performance: CAN AFFECT CANNOT AFFECT

Flipped from 1 to 0:

Correctness: CAN AFFECT CANNOT AFFECT

If correctness is affected, why?

Performance: CAN AFFECT CANNOT AFFECT

(i) A bit in the stride field of a stride prefetcher:

Flipped from 0 to 1:

Correctness: CAN AFFECT CANNOT AFFECT

If correctness is affected, why?

Performance: CAN AFFECT CANNOT AFFECT

Flipped from 1 to 0:

Correctness: CAN AFFECT CANNOT AFFECT

If correctness is affected, why?

Performance: CAN AFFECT CANNOT AFFECT

VI. Prefetching (60 Points)

You and your colleague are tasked with designing the prefetcher of a machine your company is designing. The machine has a single core, L1 and L2 caches and a DRAM memory system.

We will examine different prefetcher designs and analyze the trade-offs involved.

- For all parts of this question, we want to compute prefetch accuracy, coverage and bandwidth overhead after the prefetcher is trained and is in steady state. Therefore, **exclude the first six requests from all computations**.
- If there is a request already outstanding to a cache block, a new request for the same cache block will not be generated. The new request will be merged with the already outstanding request in the MSHRs.

(a) You first design a stride prefetcher that observes the last three cache block requests. If there is a constant stride between the last three requests, it prefetches the next cache block using that stride.

You run an application that has the following access pattern to memory (these are cache block addresses):

```
A A+1 A+2 A+7 A+8 A+9 A+14 A+15 A+16 A+21 A+22 A+23 A+28 A+29 A+30...
```

Assume this pattern continues for a long time.

Compute the coverage of your stride prefetcher for this application.

Compute the accuracy of your stride prefetcher for this application.

(b) Your colleague designs a new prefetcher that, on a cache block access, prefetches the next N cache blocks.

The coverage and accuracy of this prefetcher are 66.67% and 50% respectively for the above application. What is the value of N?

We define the bandwidth overhead of a prefetcher as

$$\frac{\text{Total number of cache block requests with the prefetcher}}{\text{Total number of cache block requests without the prefetcher}} \quad (1)$$

What is the bandwidth overhead of this next-N-block prefetcher for the above application?

(c) Your colleague wants to improve the coverage of her next-N-block prefetcher further for the above application, but is willing to tolerate a bandwidth overhead of at most 2x.
Is this possible? **YES** **NO**
Why or why not?

(d) What is the minimum value of N required to achieve a coverage of 100% for the above application? Remember that you should exclude the first six requests from your computations.

What is the bandwidth overhead at this value of N?

(e) You are not happy with the large bandwidth overhead required to achieve a prefetch coverage of 100% with a next-N-block prefetcher. You aim to design a prefetcher that achieves a coverage of 100% with a 1x bandwidth overhead. Propose a prefetcher design that accomplishes this goal.
Be concrete and clear.

SCRATCH PAD

SCRATCH PAD

SCRATCH PAD

SCRATCH PAD