CS/ECE 752: Advanced Computer Architecture I

Prof. David A. Wood

Final Exam
December 18, 2006
Approximate Weight: 20%

CLOSED BOOK
TWO SHEETS OF NOTES

NAME: _____

DO NOT OPEN THE EXAM UNTIL TOLD TO DO SO!

Read over the entire exam before beginning. Verify that your exam includes all 9 pages. Budget your time according to the weight of the questions, and your ability to answer them. Limit your answers to the space provided, if possible. If not, write on the BACK OF THE SAME SHEET. Use the back of the sheet for scratch work. WRITE YOUR NAME ON EACH SHEET.

Problem	Possible Points	Points
Problem 1	10	
Problem 2	30	
Problem 3	10	
Problem 4	10	
Problem 5	15	
Problem 6	25	
Total	**100**	

NAME: _____

Problem 1: (10 points)

Circle either the word True or the word False for the questions below.

True or False: Compiler optimizations such as loop unrolling and if-conversion can improve program performance on both statically scheduled processors and dynamically scheduled processors.

True or False: Non-blocking caches improve performance by reducing the latency of cache misses.

True or False: The Sun Niagara processor uses the ICOUNT policy to select the which thread should run next.

True or False: Software pipelining is the preferred optimization if a loop has a loop-carried dependence.

True or False: Loop blocking is a compiler optimization that primarily tries to increase a program's spatial locality.

True or False: Seznec's skewed-associative caches try to reduce conflict misses by hashing the address bits in the cache tag.

True or False: Stream buffers are one way to implement non-binding prefetching.

True or False: Virtual address aliases are not a problem for a physically-tagged cache if the virtual memory page size times the associativity are less than or equal to the cache size.

True or False: TLB Reach can be increased both by increasing the page size and by increasing the number of TLB entries.

True or False: Implementing TLB reload (i.e., TLB miss handling) in hardware always results in lower TLB miss penalties than implementing it in software.

Problem 2: (30 points)

Consider a memory hierarchy with the following components and properties:

- An L1 data cache with 64 KByte capacity, 32 byte blocks, direct-mapped placement, a write-through policy, and physical tags. An L1 hit takes 2 cycles.
- A two-entry victim cache (i.e., victim buffer) between the L1 and L2 with fully-associative placement, LRU replacement, and a "swap-on-hit" policy. A reference that misses in the L1 cache but hits in the victim cache takes a total of 4 cycles.
- An L2 unified cache with 1 Mbyte capacity, 128 byte (address) blocks, 32 byte subblocks, 4-way set-associative, LRU replacement policy, a writeback policy, and physical tags. A reference that misses in both the L1 cache and victim cache but hits in the L2 cache takes a total of 15 cycles.
- A main memory system with 4 Gbyte capacity. A reference that misses in all caches and is satisfied by the main memory takes a total of 250 cycles.
- Physical addresses are 32 bits and the smallest addressable unit is one byte.
- Ignore the instruction cache and TLB.

Part A: (15 points)

How many bits are required to implement this memory system? Be sure to include the state needed to implement the various policies. Complete the table below. Show your work in the space below the table, using additional pages if necessary.

Table 1:

Cache	Bits per Block	Blocks per set	Bits per set	Total bits in cache
L1 Cache				
Victim Cache				
L2 Cache				

NAME: _____

Part B: (15 points)

Consider only the L1 data cache and victim cache and assume that all entries in both caches are initially invalid. For the memory reference stream below, determine which references hit or miss at each level (accessed from top to bottom). Assume that the victim cache is only accessed on L1 misses and all accesses that miss in the victim cache will hit in the L2 cache. Assume byte addressing and all accesses are 32-bit loads.

Table 2:

Memory Reference	L1 Index	Hits in?	Victim Cache Contents MRU, LRU
0x00000000	0	L2	invalid, invalid
0x00010010			
0x00020020			
0x00040040			
0x00000000			
0x00010010			
0x00020020			
0x00040040			
0xEEE00000			
0x00000000			
0x00010010			
0x00020020			
0x00040040			

Complete the tables to the left. In Table 2,, indicate in which cache each reference hits. In Table 3, indicate the number of hits, misses, and cycles for each cache.

Hint: Calculate the L1 index bits for each access in the middle column. Then track which entries are added to or removed from the victim cache.

Table 3:

Cache	Number of Hits	Number of Misses	Hit Latency	Total Hit Cycles
L1 Cache			2	
Victim Cache			4	
L2 Cache			15	
All				

Problem 3: (10 points)

What is the difference between a *store queue*, a *write buffer*, and a *writeback buffer*? Explain what each one does and identify their key requirements. Which, if any, of these are visible to the instruction set architecture (i.e., can affect the execution of any program)?

NAME: _____

Problem 4: (10 points)

Part A: (5 points)

What is the key architectural feature needed to support strip mining for arbitrary length arrays? Explain how it is used in strip mining.

Part B: (5 points)

Mark Hill's original 3C's classification broke cache misses into three categories. Norm Jouppi later added a fouth category. What are the four categories? Identify the addition and explain why it was omitted from the original classification.

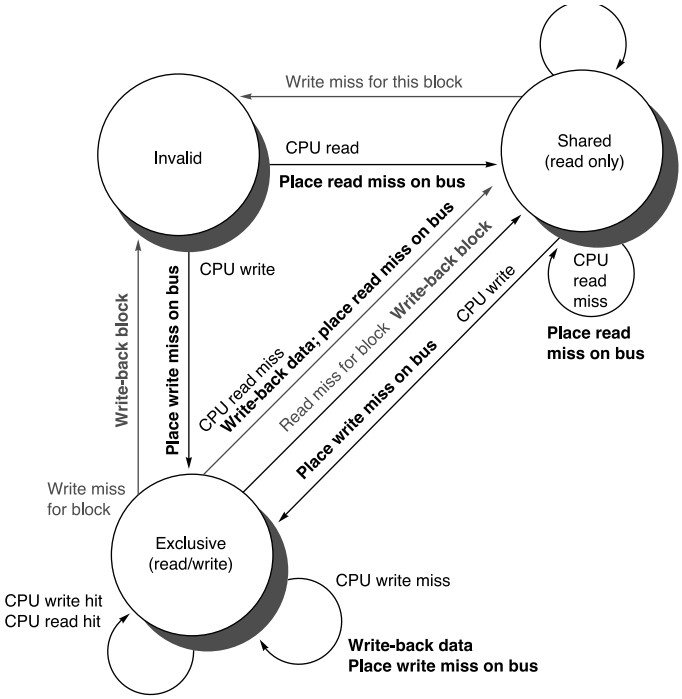
NAME: _____

Problem 5: (15 points)

Consider a multiprocessor system that uses broadcast snooping cache coherence. Four processors, P1, P2, P3, and P4 perform the following sequence of loads and stores to/from lines A and B. Assume A and B do not conflict in the data caches. Assume the protocol described in the book (reprinted on the next page), which uses the three states: Invalid (I), Shared (S), and Exclusive (E). The table below shows the state of the memory system as time flows down. The cache blocks are represented with the following notation: *address:(state, data)*. For example, *A:(I,0)* means that cache block A is in state I with data value 0. A data value of x means the value is unknown or undefined. Complete the table below, updating the cache and memory states in response to the sequence of loads and stores. Indicate actions taken by the cache and memory controllers: hits, requests to get a block shared or exclusive, and responses to requests. You may use arrows (as shown) to indicate that the state has not changed in that cycle.

P1	P2	P3	P4	MEM
A:(I,x) B:(I,x)	A:(I,x) B:(I,x)	A:(I,x) B:(I,x)	A:(I,x) B:(I,x)	A:0 B:0
load A miss, get A shared A:(S,0) B:(I,x)	↓	↓	↓	respond with A A:0 B:0
↓	**load B** miss, get B shared A:(I,x) B:(S,0)	↓	↓	respond with B A:0 B:0
	load A			
	store B = 1			
		load A		
load B				
			store A = 3	
load A				
			load B	

NAME: _____


Write miss for this block
Invalid
CPU read
Place read miss on bus
Shared
(read only)
CPU write
Place write miss on bus
Write-back block
CPU read miss
Write-back data; place read miss on bus
Read miss for block
Write-back block
Place write miss on bus
CPU write
CPU
read
miss
Place read
miss on bus
Write miss
for block
Exclusive
(read/write)
CPU write hit
CPU read hit
CPU write miss
Write-back data
Place write miss on bus

Problem 6: (25 points)

Consider the following code sequence executing on a MIPS R10000 processor.

```
L0:   lw     R6, 0(R1)       // notation: lw dest, imm(src1)
L1:   lw     R2, 100(R1)
L2:   add    R1, R6, R3      // notation: add dest, src1, src2
L3:   bnez   R2, L7
L4:   add    R6, R1, R5
L5:   sub    R1, R2, R4
L6:   jump   end
L7:   add    R3, R6, R1
L8:   lw     R0, 8(R2)
L9:   lw     R1, 16(R2)
L10:  sub    R4, R6, R1
end:  halt
```

Execution follows this sequence: instruction dispatch begins at L0 and advances to the branch instruction, which is predicted taken and control transfers to the code at L7; a misprediction is detected just before dispatch of the sub instruction at L10, and execution restarts at L4. The table to the left shows the register map *prior* to renaming the instruction marked by the label. For example, the state in the column L0 is the state of the register map just before the first instruction is remapped. The table to the right is the active list (reorder buffer) before the first instruction is dispatched. The free list (below) shows which physical registers are available. **Fill in the tables assuming that instruction dispatch has continued to the instruction at the label 'end'. Further assume that the 'lw' instruction at L1 has NOT completed execution, but that all other independent instructions have completed. Also, show the state of the free list at this point.**

Register Map				
Logical Register	Physical Register			
	L0	L7	L10	end
R1	7			
R2	9			
R3	11			
R4	13			
R5	15			
R6	1			

Active List (ROB)			
PC (use label)	Result Register	Previous Mapping	Done

Head@L0

Tail@L0

14	12	10	8	6	4	2	0	3	5	7	15

RD_PTR WR_PTR

Free List