CS/ECE 552: Introduction to Computer Architecture

Prof. David A. Wood

Final Exam
May 17, 2006
12:25-2:25pm, 1221 Computer Sciences
Approximate Weight: 25%

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. It is a long exam, so use your time carefully. 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.

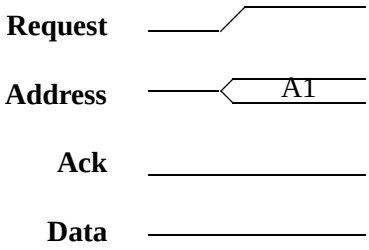
Problem	Possible Points	Points
Problem 1	10	
Problem 2	10	
Problem 3	15	
Problem 4	25	
Problem 5	20	
Problem 6	20	
Total	**100**	

NAME: _____

Problem 1: (10 points)

Part A: (5 points) PCI and its successor PCI Express are two standard I/O interconnects for systems based around the Intel x86 platform. PCI is a parallel I/O architecture with a maximum *usable* bandwidth of 64 bits per cycle at 266 MHz. PCI Express is a serial I/O architecture that has a *raw* bandwidth of one bit per cycle *per lane* at 2.5 GHz. A given PCI Express channel may have between 1 and 32 lanes (denoted 1x, 2x, ... 32x). Which provides more *usable* bandwidth: the parallel 64-bit PCI or an 8x PCI Express channel? Show your work.

Part B: (5 points) A four-cycle handshake is used when two modules run on completely different clocks (i.e., are asynchronous) and thus can't assume when the other will see their request or response. In the example below, the master initiates the request by asserting Request and driving Address. The slave must respond by asserting Ack and driving Data. Complete the timing diagram below to show the full four-cycle handshake between the master and the slave.



NAME: _____

Problem 2: (10 points)

Ideally, the 5-stage pipeline discussed in class will complete one instruction every cycle. Stall Cycles Per Instruction (SCPI) is a metric that measures the average number of stalls (i.e., pipeline bubbles) that get introduced per instruction. Thus the processor's CPI = 1 + SCPI. SCPI can be expressed as the sum of the SCPI's of different (independent) factors. For example, CPI = 1 + $SCPI_{data}$ + $SCPI_{branch}$ + $SCPI_{I\text{-}cache}$ +$SCPI_{D\text{-}cache}$ breaks the CPI down into stalls due to data dependence stalls, branch stalls, instruction cache stalls, and data cache stalls.

Part A: (5 points) What is the $SCPI_{D\text{-}cache}$ of a machine with a data cache miss penalty of 101 cycles and a load/store miss rate of 3%, when 33% of instructions are loads and stores?

Part B: (5 points) Explain how multithreading attempts to reduce the effect of $SCPI_{D\text{-}cache}$.

NAME: _____

Problem 3: (15 points)

Part A: (3 points) Show the Booth recoding for the 8-bit multiplier -39_{ten}.

Part B: (3 points) What is the 2-bit modified Booth recoding of the multiplier in Part A?

Part C: (4 points) A 16-bit multiplier has been recoded using the 2-bit modified Booth recoding algorithm. The recoded multiplier is:

```
0 -2 0 2 -1 2 -1 1
```

What is the original 16-bit two's complement multiplier?

Part D: (5 points) What is a Wallace Tree? Where is it used?

NAME: _____

Problem 4: (25 points)

Consider a memory system with the following parameters. The virtual address has 64 bits. The physical address has 39 bits. A unified (i.e., instructions and data) cache is writeback and has 512 kilobyte capacity with 64-byte blocks. The translation lookaside buffer (TLB) has 64 entries, is fully associative, and is accessed *before* the cache. Pages are 16 kilobytes. All addresses are byte addresses.

Part A: (3 points) Show the breakup of a virtual address into virtual page number and byte offset within a page. Indicate the number of bits in each field.

Part B: (3 points) Show the breakup of a physical address into a page frame number and a byte offset with the page frame. Indicate the number of bits in each field.

Part C: (3 points) How many bits of storage does it take to implement this TLB? Assume the minimum number of bits possible to achieve a correct implementation of address translation.

Part D: (3 points) What other bits of state may a TLB maintain? What are these used for?

Part E: (3 points) Suppose the cache is four-way set associative with pseudo-least-recently-used (pLRU) replacement. Show the break up of the physical address into tag, index, and byte within block used to access the cache. Indicate the number of bits in each field.

Part F: (3 points) How many sets are there in the cache?

Part G: (3 points) How many bits per set are needed to implement the pLRU replacement policy?

Part H: (4 points) How many total *bits* does it take to implement the cache (i.e., tags, state, data, LRU, etc.).

NAME: _____

Problem 5: (20 points)

Part A: (2 points) What is the Hamming distance between 01100101_{two} and 01010110_{two}?

Part B: (3 points) What is the minimum Hamming distance needed between any pair of valid code words to detect a single bit error?

Part C: (3 points) What is the minimum Hamming distance needed between any pair of valid code words to correct a single bit error?

Part D: (3 points) What is the minimum Hamming distance needed between any pair of valid code words to correct a single bit error AND detect a double bit error?

Part E: (3 points) The parity check matrix for a SECDED code is given below. In this matrix C_i's denote check bits and b_i's denote information bits. The codewords are stored in memory in the same bit order as shown in the parity check matrix.

	C_1	C_2	b_1	C_3	b_2	b_3	b_4	C_4
H =	1	0	1	0	1	0	1	0
	0	1	1	0	0	1	1	0
	0	0	0	1	1	1	1	0
	1	1	1	1	1	1	1	1

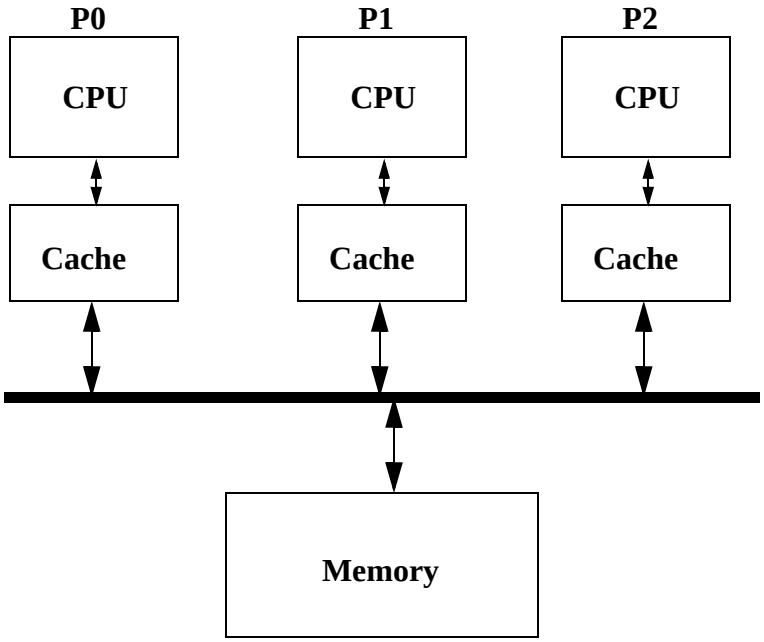
Consider the data word $b_1b_2b_3b_4 = 0111$. Assuming *odd parity is used to compute all check bits*, what is the codeword that is stored in memory for the data word given the above parity check matrix?

Part F: (3 points) Suppose that the word read from the memory is 0101 0101, calculate the syndrome using the parity check matrix above.

Part G: (3 points) What is the procedure for detecting a double error?

Part H: (3 points, extra credit) Was there a double bit error in Part F? If not, what was the value originally written into memory? Like most real systems, ignore the possibility of more than two errors.

Problem 6: (20 points)



Symmetric Multiprocessors (SMPs) use a shared bus to connect the processors to a (logically) shared memory, as illustrated above. Each processor has one or more *writeback* caches to reduce both memory latency and contention for the shared bus. A *write-invalidate* cache coherence protocol is used to ensure that caches maintain a coherent view of the memory state.

Assume a three-state write-invalidate protocol, as discussed in class and in the textbook. The three states are:

```
Modified (M), also known as Read/Write (Dirty)
Shared (S), also known as Read Only (Clean)
Invalid (I)
```

Consider an SMP with the initial memory state as shown on the next page. For simplicity, the table shows only the memory block at address 100 and the block only contains one word. Consider a sequence of processor operations (time flows down the page). For each processor operation, write a one or two sentence description of the actions taken by the coherence protocol (I have completed the first one for you, as an example). Fill in the tables with the state of the caches and memory (for block 100) after each operation has completed.

Recall that $0 in MIPS always contains the value zero.

Initial state:

P0		P1		P2		Memory
State	**Data**	**State**	**Data**	**State**	**Data**	**Data**
I	**0**	**I**	**0**	**I**	**0**	**7**

Operation: Processor P0 executes lw $1, 100($0)

Action: P0 misses in the cache, reads the block from memory, and loads the value 7 into $1

Memory state:

P0		P1		P2		Memory
State	**Data**	**State**	**Data**	**State**	**Data**	**Data**

Operation: Processor P1 executes addi $1, $0, 13 and sw $1, 100($0)

Action:

Memory state:

P0		P1		P2		Memory
State	**Data**	**State**	**Data**	**State**	**Data**	**Data**

Operation: Processor P2 executes lw $1, 100($0)

Action:

Memory state:

P0		P1		P2		Memory
State	**Data**	**State**	**Data**	**State**	**Data**	**Data**

Operation: Processor P1 executes addi $1, $0, 19 and sw $1, 100($0)

Action:

Memory state:

P0		P1		P2		Memory
State	**Data**	**State**	**Data**	**State**	**Data**	**Data**