

Name _____

Email _____@mit.edu

Computer System Architecture
6.823 Quiz #3
April 24th, 2015
Professors Daniel Sanchez and Joel Emer

This is a closed book, closed notes exam.

80 Minutes
14 Pages

Notes:

- Not all questions are of equal difficulty, so look over the entire exam and budget your time carefully.
- Please carefully state any assumptions you make.
- Show your work to receive full credit.
- Please write your name on every page in the quiz.
- You must not discuss a quiz's contents with other students who have not yet taken the quiz.

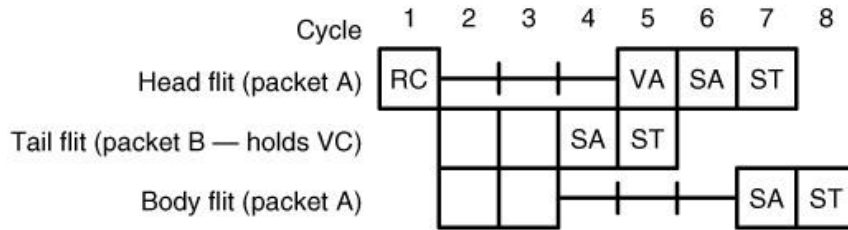
Part A	_____	32 Points
Part B	_____	38 Points
Part C	_____	30 Points

TOTAL _____ **100 Points**

Part A: Network-on-chip (32 Points)

Question 1 (6 points)

Consider the router in Handout 10. Assume this router **has one virtual channel per physical link**. Suppose two packets, A and B, are traversing the router. Both are routed to output unit 2, as shown in the following waterfall diagram.



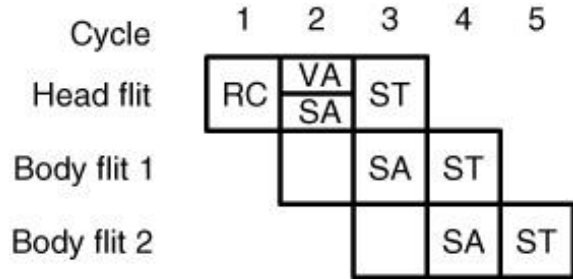
Before cycle 1, packet B's head flit has finished RC and VA. In the following cycles, packet B's four flits traverse SA and ST without stalls. Packet A's head flit completes routing computation at cycle 1 and tries to allocate an output virtual channel starting at cycle 2. Unfortunately, the only output virtual channel compatible with its route is occupied by packet B, so packet A's head flit fails to allocate a VC and is unable to make progress until packet B's tail flit releases the VC.

Fill in the following table showing the state of packet A's input virtual channel.

Cycle	G	R	O
1	R	-	-
2			
3			
4			
5			
6			
7			
8			

Question 2 (8 points)

Suppose the router in Handout 10 is improved with **speculative switch allocation**. Head flits attempt VC and switch allocation in the same cycle. If both succeed, the head flit traverses the switch on the next cycle, as shown in the waterfall diagram below.



Consider the same scenario as in question 1, with packets A and B going to the same output unit. Assume that **non-speculative switch allocation requests are always prioritized over speculative ones** (i.e., those from flits without a VC). Fill in the following waterfall diagram to show how packet A is routed.

Cycle	1	2	3	4	5	6	7	8
A: Head Flit	RC							
A: Body Flit 1								
A: Body Flit 2								
B: Body Flit 1	SA	ST						
B: Body Flit 2	-	SA	ST					
B: Body Flit 3	-	-	SA	ST				
B: Tail Flit	-	-	-	SA	ST			

Question 4 (8 points)

Ben Bitdiddle wants to implement the Valiant routing algorithm, which routes each packet through a randomly-chosen intermediate node. He uses routers with two virtual channels per physical link. He decides to use X-Y routing between the source node and intermediate node, and Y-X routing between the intermediate node and the destination node. However, Alyssa points out this routing algorithm will not work without further modification. Explain why this is the case and provide a solution for Ben.

Part B: Cache Coherence (38 Points)

Ben Bitdiddle is designing a snoopy-based, write-invalidate MSI protocol for write-back caches. Under the standard MSI protocol, when a cache observes a Bus Read Exclusive message (BusRdX), it has to invalidate its own copy of the cache block. Ben instead proposes an optimization, called delayed invalidation, to potentially reduce the number of read misses. The optimization works as follows:

Delayed invalidation: When a cache observes a Bus Read Exclusive message (BusRdX) and it has a copy of the block in the Shared (S) state, the cache delays the invalidation of the block until before a cache miss happens. In other words, the cache will treat any subsequent requests from its own processor as if the BusRdX had not happened, until one of those requests causes a miss. At that point, all pending invalidations are performed before processing the miss.

Question 1 (14 Points)

Suppose processors P1 and P2 have private, snoopy caches. Both caches are initially empty. Consider the following sequence of accesses:

I0	P2: read	A
I1	P1: write	A
I2	P2: read	A
I3	P1: write	A
I4	P2: read	A
I5	P2: read	B
I6	P2: read	A

Assume blocks A and B do not conflict in the cache. Compare Ben's delayed invalidation optimization with the standard MSI protocol by filling the states (on the next page) for each cache block after each operation is done and calculate the number of misses in both cases.

Assume we use the standard MSI protocol. Fill in the following table.

Standard MSI Protocol				
	Processor P1's Cache		Processor P2's Cache	
Initial State	A: I	B: I	A: I	B: I
After P2 reads A	A: I	B: I	A: S	B: I
After P1 writes A	A:	B:	A:	B:
After P2 reads A	A:	B:	A:	B:
After P1 writes A	A:	B:	A:	B:
After P2 reads A	A:	B:	A:	B:
After P2 reads B	A:	B:	A:	B:
After P2 reads A	A:	B:	A:	B:

How many misses occur in the two caches?

Assume we adopt Ben's delayed invalidation optimization. Fill in the following table. If there is a delayed invalidation, write it in the invalidation queue (the "Inv Queue" column). For example, "Inv L" means there is a delayed invalidation on block L.

MSI Protocol with Delayed Invalidation						
	Processor P1's Cache			Processor P2's Cache		
	MSI state		Inv Queue	MSI state		Inv Queue
Initial State	A: I	B: I		A: I	B: I	
After P2 reads A	A: I	B: I		A: S	B: I	
After P1 writes A	A:	B:		A:	B:	
After P2 reads A	A:	B:		A:	B:	
After P1 writes A	A:	B:		A:	B:	
After P2 reads A	A:	B:		A:	B:	
After P2 reads B	A:	B:		A:	B:	
After P2 reads A	A:	B:		A:	B:	

How many misses occur in the two caches?

Question 2 (6 Points)

Does Ben's delayed invalidation optimization violate cache coherence rules? Please explain your answer in one or two sentences.

Question 3 (10 Points)

Suppose the original system guarantees sequential consistency. Does adding the delayed invalidation optimization break sequential consistency? Please explain your answer in one or two sentences. If your answer is yes, please provide a sequence of load/store operations that violates sequential consistency.

Question 4 (8 Points)

Ben only applies delayed invalidation on cache blocks that are in the S state. When a cache observes a Bus Read Exclusive message (BusRdX) and the associated cache block is in the Modified (M) state, it sends out the data in response to a BusRdX message and changes the cache state to Invalid (I).

Is it possible to delay invalidation when the cache block is in the Modified (M) state? If it is not, please explain why. If it is possible, please describe how to make delayed invalidations work when the block is in the M state. In other words, please describe the actions the cache needs to take when the cache observes a BusRdX message, how to handle subsequent read and write accesses if the invalidation is delayed, and when the invalidation needs to be processed.

Part C: Synchronization and Consistency (30 Points)

Please use Handout 14 to answer the questions in this part.

Question 1 (12 Points)

Ben designs an architecture that does not have the atomic compare-and-swap (CAS) instruction but has load-reserve (LR) and store-conditional (SC) instructions.

Help Ben implement a Boolean compare-and-swap instruction BCAS *old, new, Imm(base)* using load-reserve and store-conditional instructions:

```
LR rs, Imm(rt):
    <flag, addr> ← <1, rt + Imm>
    rs ← Memory[rt + Imm]

SC rs, Imm(rt):
    If <flag, addr> == <1, rt + Imm>:
        Memory[rt + Imm] ← rs
        rs ← 1                # Succeed
    Else:
        rs ← 0                # Fail
```

BCAS is a simplified CAS instruction that only deals with values 0 and 1. You can use temporary registers (*tmp1, tmp2, tmp3...*) and any algorithmic, logical, memory, and branch instructions in the MIPS instruction set.

Name _____

Question 1 Cont.

(This is an extra page in case you need more space for Question 1.)

Question 2 (6 Points)

Suppose the hardware where the shared-memory queue from Handout 14 is executed has a weak consistency model that relaxes all the orderings of reads and writes. Give an example of memory orderings between the producer and consumer that would result in incorrect behavior. *Please fully explain your answer to get full credit.*

Your memory ordering example should look something like:

P1, C2, P2, C4, P4, C5, C7, C9, C10

Question 3 (12 Points)

Please add the minimum number of memory fences ($FENCE_{WR}$, $FENCE_{RW}$, $FENCE_{WW}$, or $FENCE_{RR}$) to the producer and consumer codes to ensure correctness with a weak consistency model. Please explain your answer fully.

Code for producer to enqueue a message:

```

P1: LD R3, 0(R2) # get tail pointer

P2: ST R1, 0(R3) # write message to tail

P3: ADD R3, R3, 4 # update tail pointer

P4: ST R3, 0(R2)

```

Code for consumer to dequeue a message:

```

C1: SpinLock: MOV R6, R0 # set R6 to 0

C2: CAS R6, R5, 0(R4) # try to acquire lock

C3: BNEZ R6, SpinLock

C4: LD R7, 0(R2) # get head pointer

C5: Retry: LD R8, 0(R3) # get tail pointer

C6: BEQ R7, R8, Retry # is there a message?

C7: LD R1, 0(R7) # read message from queue

C8: ADD R7, R7, 4 # update head pointer

C9: ST R7, 0(R2)

C10: ST R0, 0(R4) # release lock

```

Name _____

Question 3 Cont.

(This is an extra page in case you need more space for Question 3.)