Name: ____________________________________    unique name: _______________

Sign the honor code:

I have neither given nor received aid on this exam nor observed anyone else doing so.

Scores:

<table>
<thead>
<tr>
<th>#</th>
<th>Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>/ 38</td>
</tr>
<tr>
<td>2</td>
<td>/ 16</td>
</tr>
<tr>
<td>3</td>
<td>/ 8</td>
</tr>
<tr>
<td>4</td>
<td>/ 26</td>
</tr>
<tr>
<td>5</td>
<td>/ 12</td>
</tr>
<tr>
<td>Total</td>
<td>/</td>
</tr>
</tbody>
</table>

NOTES:

• Closed book, closed notes.
• Calculators are allowed, but no PDAs, Portables, Cell phones, etc.
• Don’t spend too much time on any one problem.
• You have about 90 minutes for the exam (avg. 22 minutes per problem).
• There are 12 pages in the exam (including this one). Please ensure you have all pages.
• **Be sure to show work and explain what you’ve done when asked to do so.**
1) Interconnection networks [38 points]

a) Consider a 16-node 2D mesh network that uses X-Y minimal routing. The routers use a conventional 5-stage pipeline (BW, RC, VA, SA, ST) and the link latency is a single cycle. The network’s flit size is 32 bits and the total size of all network messages (including head and tail flits) is 64 bytes. Each router has up to 5 inputs (N, S, E, W, ingress), and 5 outputs (N, S, E, W, egress). Messages must pass through the router pipeline at their ingress and egress nodes. The interconnect is clocked at 1GHz.

i) What is the maximum hop count (number of links) a packet must traverse in the network? [3 points]

6 hops. E.g., lower left corner to upper right corner.

(ii) How many cycles would it take to transfer entire message between source and destination pairs having maximum hop count. [3 points]

7 routers * 5 cycles + 6 links * 1 cycle + serialization (64 bytes / 4 bytes per cycle).

57 cycles

(iii) How many cycles would it take to send a message between the same sender and receiver pair if the topology of the network was a 2D torus instead of a mesh? [3 points]

Now its just 2 hops.

3 routers * 5 cycles + 2 links * 1 cycle + serialization (64 bytes / 4 bytes per cycle).

33 cycles

(iv) What is the bisection bandwidth of the 2D mesh network? [3 points]

A bisection cuts 4 links. 4 links * 4 bytes / cycle * 1GHz = 16 Gbytes / sec
b) Consider a network implements both X-Y and Y-X routing algorithms and chooses the algorithm adaptively depending on the congestion in the network.

(i) What is the potential problem that network might suffer if it routes using both algorithms simultaneously? [4 points]

Deadlock. Packets using each of the two algorithms can form a dependency cycle

(ii) Explain a fix that can be implemented to avoid this problem. [4 points]

Use each algorithm on a different virtual channel

c) What is livelock? Explain a congestion scenario in a network that implements non-minimal adaptive routing that can lead to a livelock. Then, explain a mechanism that can be employed to prevent livelock in such a network. [4 points]

In an adaptively routed network, livelock can arise if packets route in a circle because paths out of the circle are blocked by congestion. An age-based priority mechanism can ensure that, ultimately, the eldest packet in the network wins all arbitration and successfully routes to its destination.
d) State one advantage and one disadvantage of a 3D torus topology over a 2D mesh topology. [6 points]

Advantage: more route diversity; shorter routes for a given size network
Disadvantage: higher radix, and hence more complex, routers

e) State one advantage and one disadvantage of each of the following routing implementations.

i) Source table routing [4 points]

Advantage: Eliminates need for route computation state in each router
Disadvantage: Entire route must be recorded in packet, lengthening packet. Can’t implement adaptive routing

ii) Node table routing [4 points]

Advantage: Can implement adaptive routing
Disadvantage: requires route computation stage at each hop
2) DeNovo [16 points]

a) State three advantages of the DeNovo protocol over conventional coherence protocols, such as MESI. [6 points]

1. No transient states
2. No ownership required for writes
3. Lower overhead since there is no need to maintain sharer lists

b) For what purpose do the authors of DeNovo introduce self-invalidations of data? [4 points]

Self-invalidations remove the need for a directory to track sharers.

c) What performance pathology can arise in DeNovoSync when more than two nodes contend for a test-and-test-and-set lock? How does the DeNovoSync design seek to mitigate the negative effects of this performance pathology? [6 points]

Readers require registered state for synchronization variables in DeNovoSync. If multiple readers contend for a lock, the lock variable will ping-pong back and forth between caches as each node spins waiting for the lock to be released. DeNovoSync adds an exponential back-off to such registered read requests to limit the frequency with which blocks ping-pong back and forth.
3) COMA [8 points]

a) Briefly explain how a Cache Only Memory Architecture (COMA) system differs from cache-coherent non-uniform memory architecture (CC-NUMA). [4 points]

*Cache lines do not have a fixed storage location in main memory in COMA systems. Instead, all of memory behaves like a cache and blocks migrate to the node in the system that is accessing the block.*

b) Why is eviction of an exclusive copy of a cache block from a node more complicated in a COMA system than in a conventional MESI-like coherence protocol? [4 points]

*In COMA, a block has no pre-defined home location to which it can be evicted. Hence, the protocol for evicting a block must find a free location somewhere in the system into which it can evict the block.*
4) Memory Consistency [26 points]

a) Consider the following code executed on four processors. Assume all registers on all cores are initialized to zero.

```
Core 1          Core 2          Core 3          Core 4
X ST A = 1      ST B = 1      LD r1 = A      LD r3 = B
               LD r2 = B      LD r4 = A
```

Unlike the Sun Total Store Order consistency model, the IBM Power consistency model does not require store atomicity.

Give an example of values for r1, r2, r3, and r4 that can arise under IBM Power consistency that are prohibited under Sun TSO. [4 points]

- \( r_1 = 1, r_2 = x, r_3 = 1, r_4 = 0. \)
- \( r_1 = 1, r_2 = 0, r_3 = 1, r_4 = x. \)

b) Three threads (t1, t2, t3) access two synchronization variables (x, y) and one data variable (d). Assume that initially \( d=y=z=0. \)

```
Core 1          Core 2          Core 3
(1) St d = 1    (3) while (Ld x == 0);  (5) while (Ld y == 0);
(2) St x = 1    (4) St y = 1      (6) assert (d == 1)
```

Under release consistency, indicate which of the loads and stores (1) – (6) must be annotated as acquires and releases to assure that the assert in line (6) will always be true. [4 points]

Either: (2) and (4) release. (5) acquire.
Either: (2) release, (3) and (5) acquire.
c) Consider the following C++ 11 program:

```cpp
int x = 0; // x, and flag are shared by both threads

int flag = 0;

Thread 1
x = 1;
flag = 1;

Thread 2
while (flag == 0);
printf("%d
", x);
```

What value(s) does C++11 require this program to print to the standard output? [4 points]

The behavior of the program is undefined, as there is a data race on the variable "flag", which was not declared atomic. Hence, C++11 does not require the program to print anything at all.

d) Conventional high-performance out-of-order processors, such as the MIPS R10K, allow load instructions in the re-order buffer to execute speculatively out of order with respect to one another and with respect to store instructions. Nevertheless, a series of research proposals, (e.g., InvisiFence) have proposed additional hardware mechanisms that improve performance over the MIPS R10K speculation mechanisms. Explain briefly why MIPS R10K’s speculation mechanisms are not sufficient to hide all consistency-related stalls. Include an example of an execution scenario where a MIPS R10K-like mechanism will stall, but InvisiFence-like hardware can eliminate the stall. [6 points]

The out-of-order instruction window of R10K is insufficient to hide the latency of store misses. The out-of-order window cannot be scaled large enough to hide these latencies. Loads that reach the head of the re-order buffer must stall. InvisiFence and similar mechanisms allow loads to retire speculatively when the post-retirement store-buffer is non-empty, where MIPS R10K would stall.
e) Consider the following code executed on two processors. x and y are memory locations, r1 through r6 are registers. Assume all registers and memory locations are initialized to zero.

Core 1

(1) r1 = & x;  //address of x in r1
(2) St [r1] = 1 //store 1 to x
(3) MEMBAR     //memory barrier
(4) r2 = & y;  //address of y in r2
(5) St [r2] = 1 //store 1 to y

Core 2

(6) r3 = & y;  //address of y in r3
(7) Ld r4 = [r3] //load y into r4
(8) r5 = & x;  //address of x in r5
(9) Ld r6 = [r5] //load x into r6

i) Under Sun’s RMO consistency model, it is possible for this program to execute with the final values r4=1 and r6=0. Where must a MEMBAR be inserted to prevent that outcome? [4 points]

Add a MEMBAR anywhere between (7) and (9).

ii) Propose a change to the program on Core 2 that prevents the final values r4=1 and r6=0 without adding any MEMBAR instructions and without changing the order of the two loads. [4 points]

Change line (8) so it has a data dependency on line (7). Any sequence of code that creates a data dependency between (7) and (9) will require them to execute in order. Here is example pseudo-code:

(8) r5 = &y | (r4 XOR r4)
5) **Design: Token Coherence [12 points]**

Conventional coherence protocols keep track of read and write permissions explicitly by tracking the state of each block in the coherence directory. Nodes must explicitly communicate with the directory to obtain read or write permission, and the directory guarantees that, when a node has write permission, no other node may read and vice versa.

Rather than track coherence permission via an explicit state machine, recent work on *Token Coherence* proposes that coherence permissions should instead be represented by a set of *tokens* for every memory address, one token per node in the system. These tokens conceptually originate in memory when a block is invalid, and are never created or destroyed, but instead can be transferred between nodes in the system through coherence-like messages. Tokens are tracked in the cache (much like coherence state) when a cache has a valid copy of a line. The scheme uses these tokens to determine if a node has an exclusive copy of a block or there are possibly multiple shared copies.

Propose a design for a token-based coherence system. Be sure to explain:

1. What must a node do to obtain read permission? How does your protocol guarantee that no node may write while some nodes have read permission?

2. What must a node do to obtain write permission? How does your protocol guarantee that there are no readers when a node has write permission?

3. How can your design support two-hop read requests?

4. How does your design ensure that a node that wishes to read or write will eventually get permission to complete its operations (in other words, how do you ensure there are no deadlocks or livelocks)?
(additional space for question 5)