EECS 570 *Midterm Exam*

Winter 2024

Name: _____ unique name: _____

Sign the honor code:

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

_____

Scores:

#	Points
1	**/ 30**
2	**/ 30**
3	**/ 20**
4	**/ 20**
Total	***/ 100***

NOTES:

- Closed book. One 8.5" x 11" page of notes is allowed.
- Calculators are allowed
- Don't spend too much time on any one problem.
- You have about 120 minutes for the exam (avg. 30 minutes per problem).
- There are 10 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. Short Answers [30 points]

a. Dennard Scaling [3 points]

Moore's law is an observation that the number of transistors on an integrated circuit will double every two years with minimal rise in cost. This is possible because of Dennard scaling, where when the wire width scales down from W to W/α and the gate width scales from G to G/α, voltage decreases from V to V/α.

(i) transistor density scales from D to _____

(ii) Power per transistor scales from P to _____

(iii) Power density scales from p to _____

b. Post Dennard Scaling [4 points]

As the voltage keeps decreasing, the leak current increases as Vdd gets closer to Vth. In this post-Dennard scaling scenario, when the wire width scales down from W to W/α and the gate width scales from G to G/α, voltage stays constant at V.

(i) transistor density scales from D to _____

(ii) Power per transistor scales from P to _____

(iii) Power density scales from p to _____

(iv) Post-Dennard scaling creates a Power / ILP / frequency (circle one) wall that kills Moore's Law.

c. Message Passing vs. Shared Memory [11 points]:

Compare a message-passing model against a shared memory model for multiple core systems. Please circle the model or models that fit the description

Programming model and HW-SW abstraction

Different threads/cores see unified address space.	Shared mem / message passing
Requires explicit communication defined by SW.	Shared mem / message passing

Implementation

Implementation of synchronization is complex.	Shared mem / message passing
High API overhead	Shared mem / message passing

Flexibility

Different cores can run different OS.	shared mem / message passing
Easy to optimize for specific workloads.	Shared mem / message passing

Give an example use case of the message passing model and an example use case of the shared memory model. Explain why each model is used by connecting their pros and cons. [5 points]

d. GPU Architecture [12 points]

SIMT (single instruction multiple threads) vs. SIMD (single instruction multiple data)

(circle the model or models that fit the description [5 points])

Programming model used on:

CPU	SIMD / SIMT
GPU	SIMD / SIMT
Provides better support for conditional control flow	SIMD / SIMT
Executes in locked steps.	SIMD / SIMT
Need explicit synchronization	SIMD / SIMT

b. CUDA memory model: circle which scopes the following memory space is defined in [4 points]

Register	per instruction	/	per block	/	per grid
Local mem	per instruction	/	per block	/	per grid
Shared mem	per instruction	/	per block	/	per grid
Global mem	per instruction	/	per block	/	per grid

c. Warp Scheduling [3 points]:

GPU SMs perform a context switch to schedule a different warp. True / False

Register values of <u>(the warp currently being executed / all the warps)</u> stay in register file. (select one that makes the sentence correct)

Conditional branches are handled with _____

2. Cache Coherence [30 points]
(a) It is common in shared memory programs for one processor to do a single write to a cache block read by many processors reading frequently. In the base MSI protocol this causes every processor to invalidate the cache line, flood the bus with load / BusRd messages, and wait for the writer in the M state to downgrade itself and share the updated cache block.

We propose an optimization where a processor that intends on doing a single write to a cache block can warn all the readers ahead of time, and broadcast the update when it is finished. This will reduce the storm of BusRd messages after invalidations in the MSI protocol. This requires two new states to be added to the MSI protocol (a standard MSI diagram is included below for reference.)

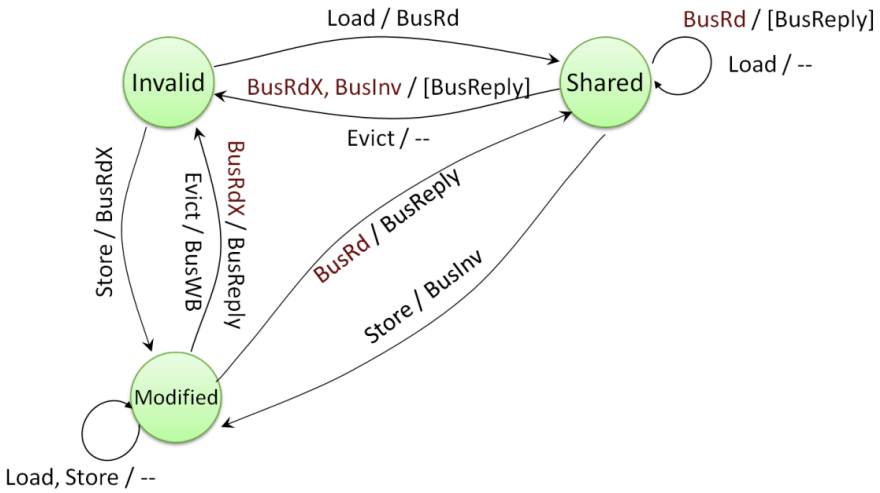
The two new states are:

Borrowed (B) - the processor intends on performing a single write operation so it "borrows" the cacheline from processors in the modified or shared state. After a single write to the cache block the borrowed state downgrades itself to the shared state and sends a BusReply message to other processors waiting for the new cacheline.

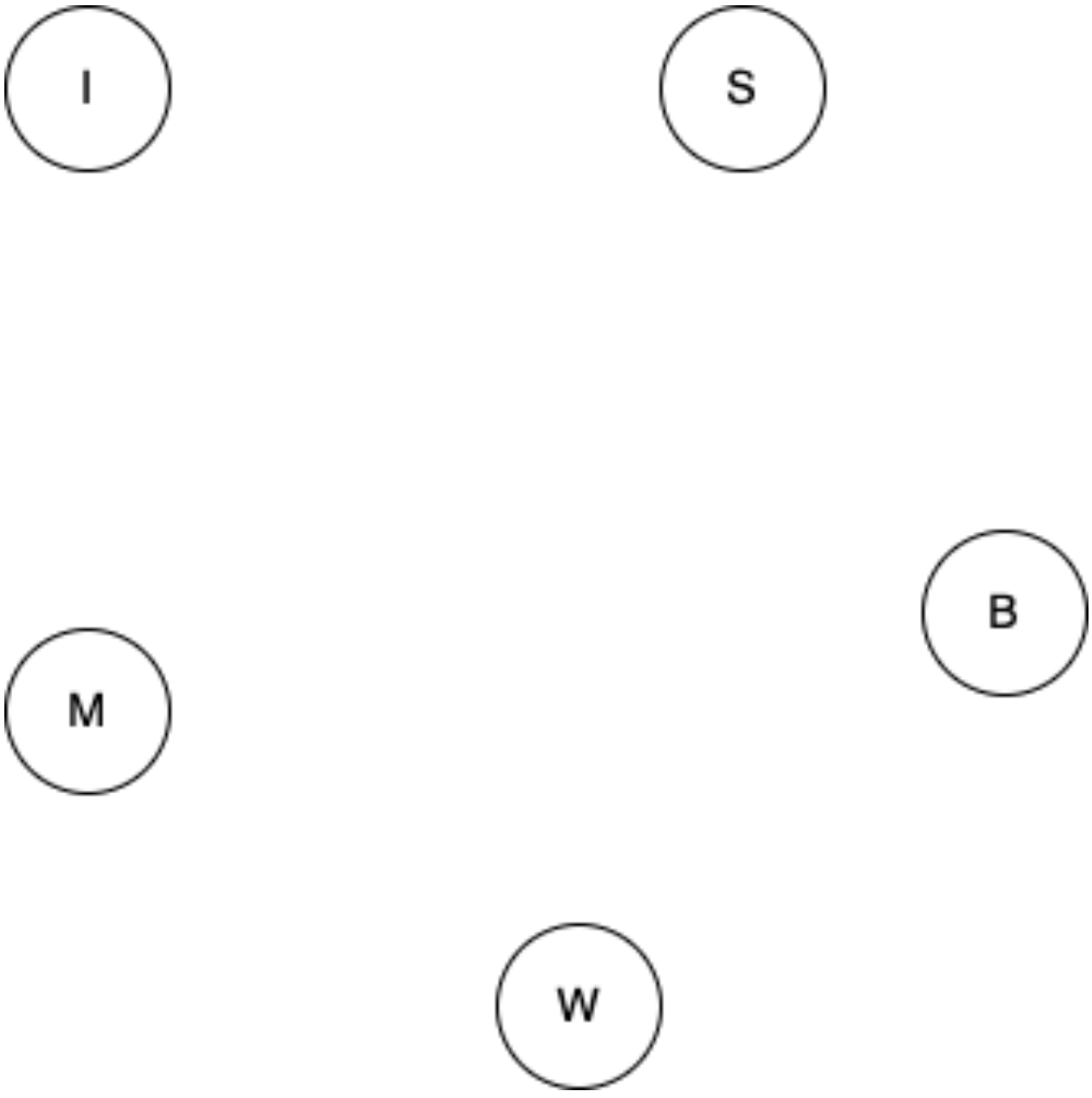
Waiting (W) - processors that are "lending" the cache block to the processor in the borrowed state will stay in the waiting state until they receive a BusReply or BusRdX message.

More information:

- To transition to the Borrowed state a processor issues a **borrow** command which sends a **BusRdBr** message over the bus.
- A processor cannot transition to the Borrowed state from the Modified state.
- There cannot be multiple processors in the Borrowed state, and if a processor is in the Borrowed state all other processors must be either Invalid or Waiting.
- A processor with an invalid cache block that attempts a *load* to enter the shared state may receive a **BusRdWait** message on the bus telling it to enter the waiting state.
- A processor with an invalid cache block that attempts a *store* or **borrow** will always succeed.
- Lastly, processors in the Waiting state will not perform loads, stores, or borrows.



Given the information above, please complete the new arrows and labels of the state diagram with the same level of detail as the reference MSI protocol for the new MSI+BW protocol. [24 Points]



Add arrows and labels to the diagram that are needed for the new protocol. **You do not need to redraw any arrows or labels from the base MSI protocol.**

(b) What is a potential downside of this optimization, explain in several sentences giving an example sequence of operations? [6 points]

3) Transactional Memory [20 points]

(a) Can transactions be used to replace barrier synchronizations? Justify. [2 points]

(b) Provide an example where two transactions deadlock. [3 points]

(c) Consider two versions of a program, one written using transactions, and another using locks. Assume that both are carefully optimized by expert programmers. Which of the two versions do you expect to perform better, and why? Assume that the runtime overhead of conflict check and versioning is similar to the overhead of lock operations. [3 points]

(d) Two transactions are said to conflict if they execute memory accesses to the same location, and at least one of those accesses is a write. Two transactions are serializable if they appear to have executed one after the other.

Is it true that if two concurrent transactions conflict, then they are not serializable? Justify your answer. Use examples if necessary. [3 points]

(e) A hardware transactional memory (HTM) system can support eager conflict detection by extending the coherence protocol. Would a lazy conflict detection have any advantages in a HTM? Justify your answer. Use examples if necessary. [3 points]

(f) Describe at least three unique challenges that may arise in supporting HTM in a GPU architecture, when compared to a multi-core CPU. [6 points]

4) Directory Protocols [20 Points]

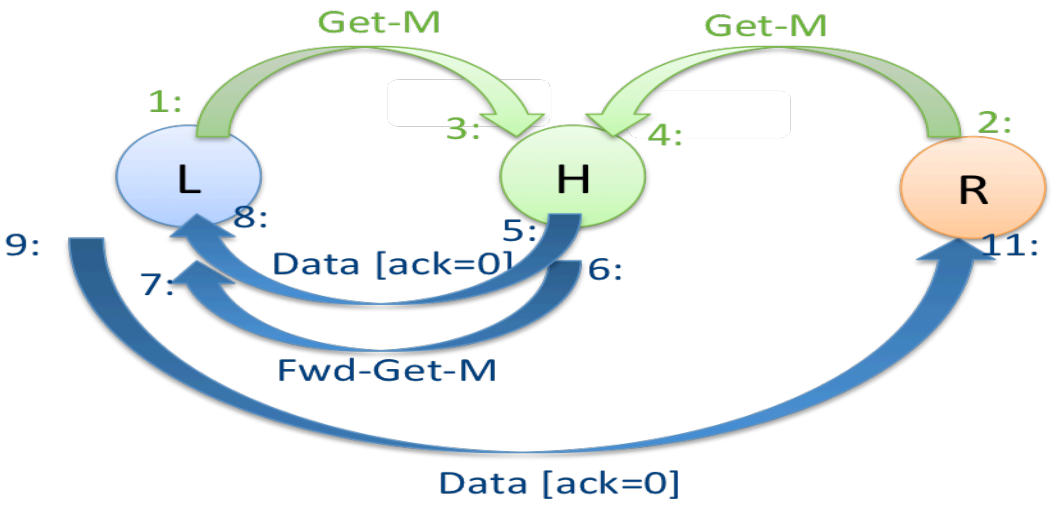
a. What are the 2 major bottlenecks to scaling a snooping bus-based coherence protocol? [2 points]

b. What is the main difference between a 4-hop and 3-hop read transaction? [2 points]

c. Cruise missle invalidations are used to improve the performance of directory protocols. Describe how a cruise missle invalidation works and what it improves in the system. [4 points]

d. When comparing a centralized versus distributed directory, what are the pros and cons of the design tradeoff. [4 points]

e. Below is the state diagram for a store-store race condition in a directory protocol. Each edge is labeled with the order the edge entered and exited the network. Below the diagram please fill out the state of the block at each node after the timestamp completes. You may leave any blocks blank if it doesn't change. [8 points]



After Cycle	State in L	State in H	State in R
0	I	I	I
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			