Submission Instructions:

1. Prepare a PDF document with your solutions to the questions below. You may use the LaTeXtemplate that is uploaded to Canvas, or any other tool. Please do not submit hand-written solutions since it will be hard to grade.
2. Ensure that each answer is clearly marked, and starting on a new page. This includes subparts of a question (e.g., Q2.1 on P3 and Q2.2 on P4, or Q1 on P1-2, Q2.1 on P3-4 and so on). Your answer may use multiple pages, but ensure that no single page has multiple answers.
3. Include the following acknowledgement on the first page of your submission, and sign it with your full name and date:

   "I affirm that the work I am submitting is my own. I have not copied or used any portion of another student's work. I have not used external sources without proper acknowledgement. If I have used AI tools, I will acknowledge the version used and explain how I used it."

   Full Name:

   Date:
4. Submit the PDF on Gradescope, matching each question to the corresponding page.

1 Memory Consistency [6]

You are an architect at a startup that is going to produce multicore processors based on an Instruction-Set-Architecture, whose **memory consistency model** is described as follows:

The memory model is defined in terms of the **global memory order**, a total ordering of the memory operations produced by all threads. The memory instructions of the ISA and the memory operations they generate are as follows (all memory instructions operate at word-granularity).

Memory operations generated by instructions:

- A `ld` instruction which *reads* from the specified word address.
- A `st` instruction which *writes* to the specified word address.

Preserved Program Order

The **global memory order** respects some but not all program orders. The subset of program order respected by the **global memory order** is known as the **preserved program order**. The preserved program order for this memory consistency model is defined as follows:

Memory operation A *precedes memory operation* B *in* ***preserved program order*** *(and hence also in* ***global memory order****), if* A *precedes* B *in program order and any of the following hold:*

- A is a read and B is a read or a write.
- A is a write and B is a write.

Axiom

Finally, an execution is said to obey the memory model only if there exists a global memory order complying with the preserved program order and satisfying the load value axiom.

Load Value Axiom: A load returns the value written to that word by the store that is latest in the global memory order among the following stores:

- Stores that write to that word address and which precede the load in global memory order.
- Stores that write to that word address and which precede the load in program order.

For the memory model defined as above, and assuming that the multicore processor is an out-of-order processor that employs a directory based coherence protocol, answer the following questions considering both the out-of-order processor pipeline and the cache coherence protocol.

(a) What memory consistency model does the multicore processor enforce? Justify your answer with reasons. [1]

(b) Demonstrate the properties of that memory model using two different litmus tests, i.e., short multi-threaded code sequences like the ones discussed in class. [2]

(c) How should load and store instructions be implemented so that the implementation adheres to the memory consistency model described above, whilst maximizing performance? (Note: maximizing performance is important.) [3]

2 State Transitions [6]

Describe the states and draw the state transition diagram of a **bus-based update-based** coherence protocol that allows for private read/write variables and shared read-only variables to be efficiently handled. The diagram should illustrate state transitions in response to both local events coming from the processor and non-local events on the bus.

Make sure to first describe the states of the protocol and their meanings before describing the state transitions. You can assume that each state transition is atomic – i.e., you don't need to worry about transient states. You can ignore cache evictions.

In order to draw the state-transition diagram, you can use the LaTeX package **`tikz`***, or you can use Google Drawings or any other similar tool. Please try and refrain from including hand-drawn diagrams in your final submission.*

3 Sharing Vector [4]

You are the lead designer of XYZ Computer which ships **10-core shared memory multi-processors** with local caches kept coherent using a **directory based coherence protocol**. Accordingly, each directory entry has a 10-bit sharing vector. Now, XYZ Computer are planning on a 16-core multi-processor. However, due to space constraints, XYZ Computer does not want to increase the size of the sharing vector. How would you adapt the coherence protocol and make it work for the 16-core multi-processor, with only 10-bit sharing vector per directory entry?