Cache Coherence

» Two necessary conditions:
  1. Write propagation: Writes eventually become visible to other processors
  2. Write serialization: All processors observe writes to one location appear to happen in a consistent order

» MSI protocol provides a sufficient condition via single-writer multi-reader policy
  - Only one cache may have write permission at any given point in time
  - Multiple caches can have read-only permission at a given point in time
Write-back caches: MSI

» Three stable states per cache-line
  - Invalid (I): Cache does not have a copy
  - Shared (S): Cache has read-only copy; clean
  - Modified (M): Cache has only copy; writable; (potentially) dirty

» Processor-initiated actions:
  - Read: needs to upgrade permission to S
  - Write: needs to upgrade permission to M
  - Evict: relinquish permissions (caused by access to a different cache line)
MSI directory states

» Uncached (Un): No cache has a valid copy

» Shared (Sh): One or more caches in S state. Must track sharers.

» Exclusive (Ex): One of the caches in M state. Must track owner.

» Does the directory need transient states?
  - Yes on downgrades/invalidations, to guarantee serialization
Memory Consistency

» Why care about it?
  - Allows us to reason about multiprocessor behavior

» Note that coherence != consistency
  - Coherence: Makes processors have an up-to-date view of a single memory location
  - Consistency: Deals with memory operations on multiple memory locations
Consistency Choices

» Sequential Consistency
  - Arbitrary order-preserving interleaving of memory references of sequential programs
  - Easiest to understand, but hard to make performant processors!

» Relax consistency orderings for processor optimizations
  - TSO: stores can be ordered after later loads -> consequence of store buffers
  - Non-blocking caches, speculative execution, ...