- Your instructor for the week Nov 13-17:

Luke McDowell
lucasm@cs.washington.edu

  - Material in this lecture in Henessey and Patterson, Chapter 8
    - 654-665
  - Plus extra info on the Illinois protocol (not fully described in text)

  - Some material from David Patterson's slides for CS 252 at Berkeley

Review

- Cache Coherency Problem
- Different solutions depending on how memory is organized
  - Snooping where we have a single bus connected to single main memory
  - Directory schemes where memory is distributed
- Snooping
  - Write-Invalidate vs. Write-Update
  - We consider only Write-Invalidate schemes

An Example Snoopy Protocol

- Invalidation protocol, write-back cache
- Each block of memory is in one state:
  - Clean in all caches and up-to-date in memory (Shared)
  - OR Dirty in exactly one cache (Exclusive)
  - OR Not in any caches
- Each cache block is in one state (track these):
  - Shared : block can be read
  - OR Exclusive : cache has only copy, its writeable, and dirty
  - OR Invalid : block contains no data
- Read misses: cause all caches to snoop bus
- Writes to clean line are treated as misses

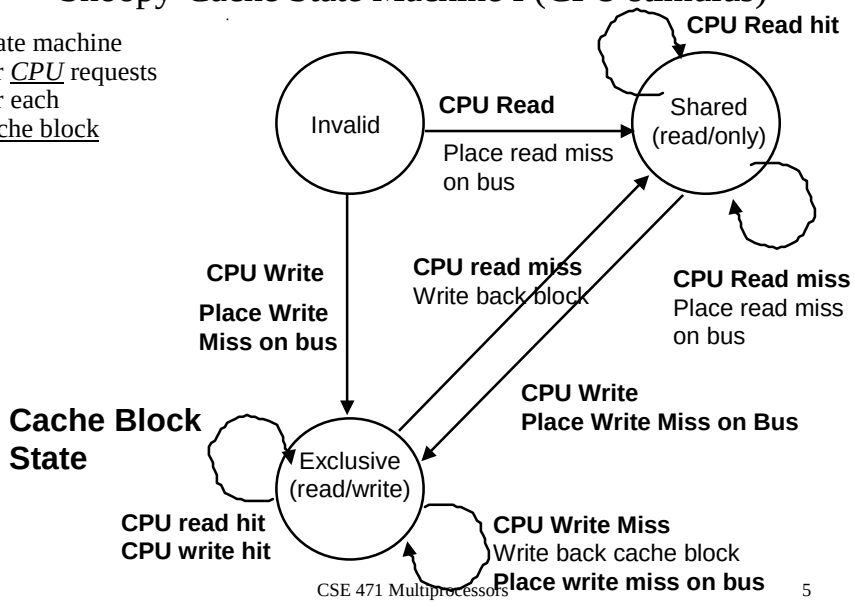
(slide from Patterson CS 252)

State transitions for a given cache block

- Events incurred by processor associated with the cache
  - Read miss, write miss, write on clean block
- Events incurred by snooping on the bus as result of other processor actions, e.g.,
  - Read miss by Q might make P's block transit from Exclusive to Shared
  - Write miss by Q might make P's block transit from Exclusive to Invalid (write invalidate protocol)

Snoopy-Cache State Machine I (CPU stimulus)

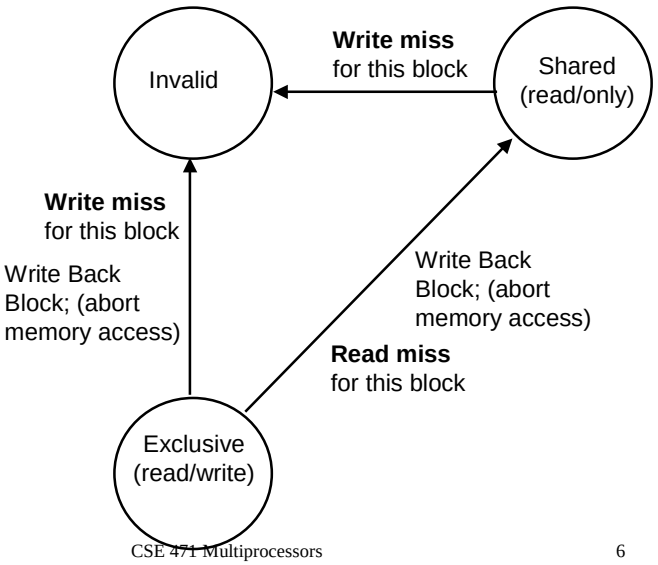
- State machine for *CPU* requests for each cache block



(slide from Patterson CS 252)

Snoopy-Cache State Machine II (Bus Stimulus)

- State machine for *bus* requests for each cache block



Important Note:

The CPU stimulus and Bus Stimulus graphs combine to form a single state machine – drawn separately for ease of understanding.

(slide from Patterson CS 252)

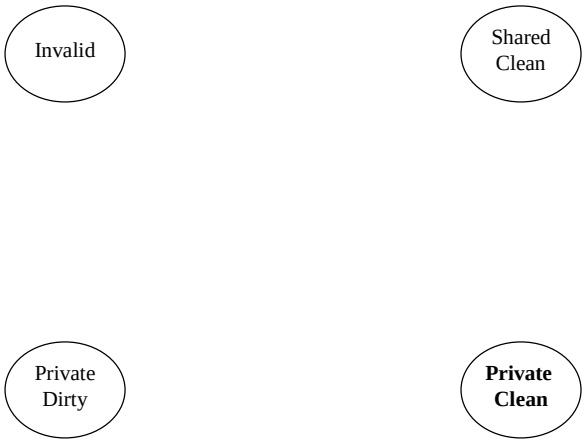
Some confusion with Snoopy Protocols

- States are stored with each cache block
  - No state stored in main memory
- State applies to a cache block, **not** to a specific block of memory
  - Remember that a particular cache block may hold different blocks of memory at different times, as indicated by the cache tags
- This can make some parts of the snooping state diagrams confusing at first
  - Can get a "Read miss" while in the Shared state?
    (because the processor needs to put a different block of memory into that cache block)

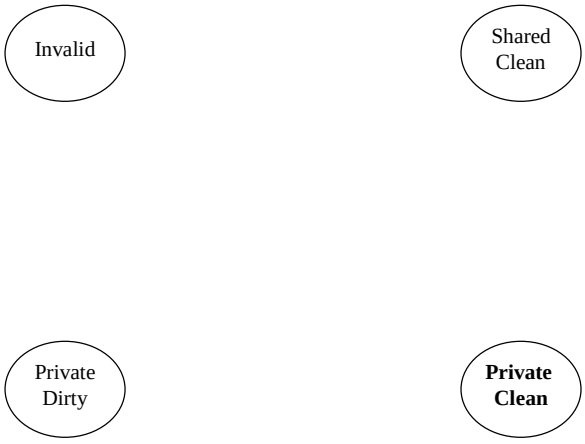
Illinois protocol: more advanced

- Two Changes:
  - 1. Add 4th state ("private & clean") to enhance performance
    - Exclusive state is now "private and dirty"
    - On a write to a block in "private and clean" state, no need to send an invalidation message (occurs often for private variables).
  - 2. On a read miss when block is in shared state:
    - Cache will provide data instead of main memory
      - Often faster than going to main memory
    - If more than one cache, which one?
      - Answer: the first to grab the bus
    - Important detail: If data comes from main memory, requesting processor knows it has the only copy

Illinois 4-state protocol (CPU stimulus)



Illinois 4-state protocol (Bus stimulus)



Performance of snoopy protocols

- Performance depends on the length of a *write run*
- Write run: sequence of write references by 1 processor to a shared address (or shared block) uninterrupted by either access by another processor or replacement
  - long write runs better to have write invalidate
  - short write runs better to have write update
- There have been proposals to make the choice between protocols at run time

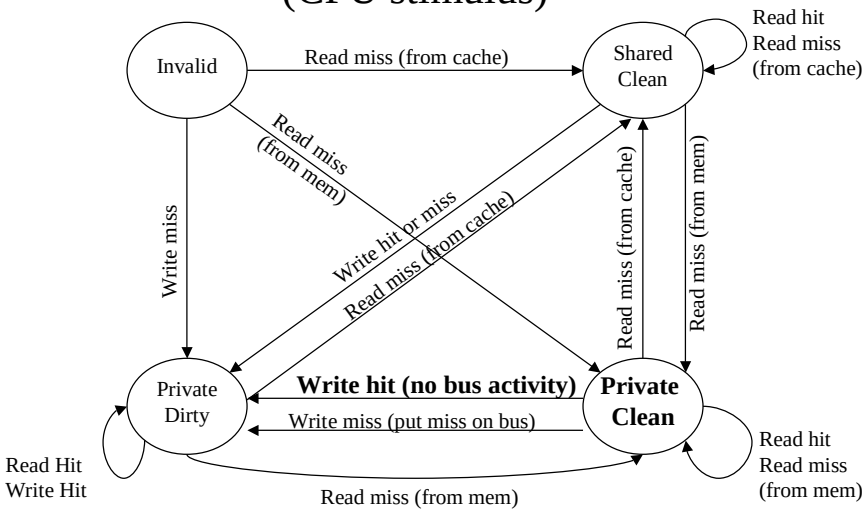
Wrinkle #1: snooping interfering with CPU

- each bus transaction causes a check on cache tags
  - could interfere with CPU's cache access, stalling processor
- fix #1: extra set of tags for snooping activity
  - on cache miss, processor must arbitrate for and update both sets
  - if snoop finds match, it must also update both tags for invalidates or to update shared bit
- fix #2:
  - exploit multilevel cache, and cache inclusion property
    - every entry in L1 cache also in L2 cache
    - snoop uses L2 cache, CPU uses L1 cache
    - snoop hit: snoop may need to update L1 cache as well
  - can do fix #1 and fix #2 combined…

Wrinkle #2: atomicity of operations

- protocols assume that operations are atomic
  - e.g., assumes write miss can be detected, acquire bus, and receive response as a single atomic action
- introduces possibility that protocol can deadlock
  - to fix, need to augment protocol to deal with non-atomic writes without adding deadlock
  - a topic for a different course…

Illinois 4-state protocol (CPU stimulus)



Illinois 4-state protocol (Bus stimulus)