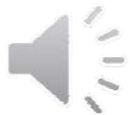


# Chapter 5

Large and Fast:  
Exploiting Memory  
Hierarchy



# Cache Coherence Problem

- Suppose two CPU cores share a physical address space
  - Write-through caches

Time step	Event	CPU A's cache	CPU B's cache	Memory
0				0
1	CPU A reads X	0		0
2	CPU B reads X	0	0	0
3	CPU A writes 1 to X	1	0	1



# Coherence Defined

- Informally: Reads return most recently written value
- Formally:
  - $P$  writes  $X$ ;  $P$  reads  $X$  (no intervening writes)  
⇒ read returns written value
  - $P_1$  writes  $X$ ;  $P_2$  reads  $X$  (sufficiently later)  
⇒ read returns written value
    - c.f. CPU B reading  $X$  after step 3 in example
  - $P_1$  writes  $X$ ,  $P_2$  writes  $X$   
⇒ all processors see writes in the same order
    - End up with the same final value for  $X$



# Cache Coherence Protocols

- Operations performed by caches in multiprocessors to ensure coherence
  - Migration of data to local caches
    - Reduces bandwidth for shared memory
  - Replication of read-shared data
    - Reduces contention for access
- Snooping protocols
  - Each cache monitors bus reads/writes
- Directory-based protocols
  - Caches and memory record sharing status of blocks in a directory



# Invalidating Snooping Protocols

- Cache gets exclusive access to a block when it is to be written
  - Broadcasts an invalidate message on the bus
  - Subsequent read in another cache misses
    - Owning cache supplies updated value

CPU activity	Bus activity	CPU A's cache	CPU B's cache	Memory
				0
CPU A reads X	Cache miss for X	0		0
CPU B reads X	Cache miss for X	0	0	0
CPU A writes 1 to X	Invalidate for X	1		1
CPU B read X	Cache miss for X	1	1	1

