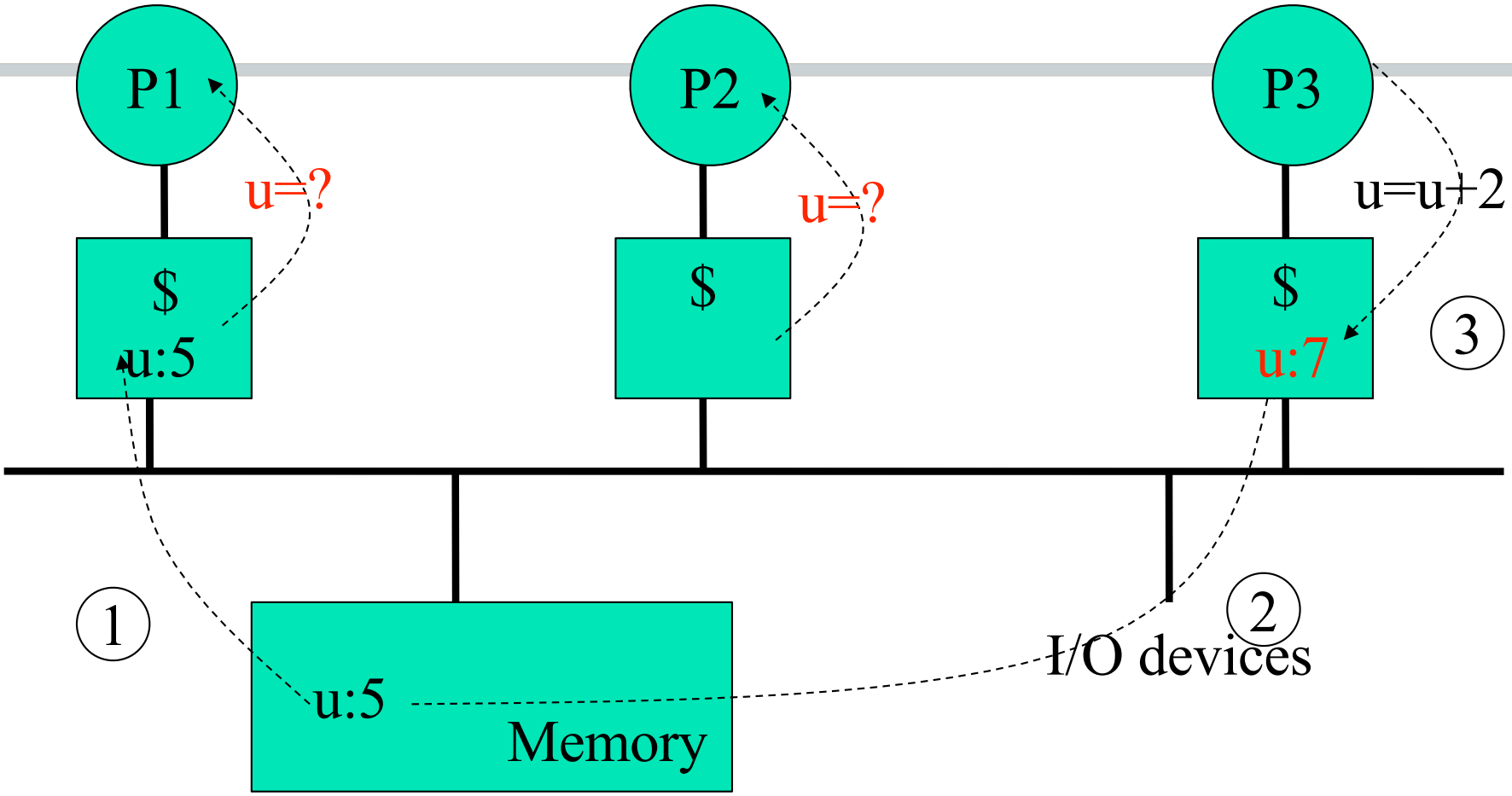
Review: Cache Coherence

- Problem: shared-memory multi-processor system with different cached copies of the same shared value
- Hardware-based protocol
  - Write invalidate, write update
  - Snooping-based, directory-based
- Key elements (write-invalidate)
  - Exclusive access to a share element before updating
  - Locating up-to-date copy
  - Broadcasting and snooping

Cache Coherence Problem Example



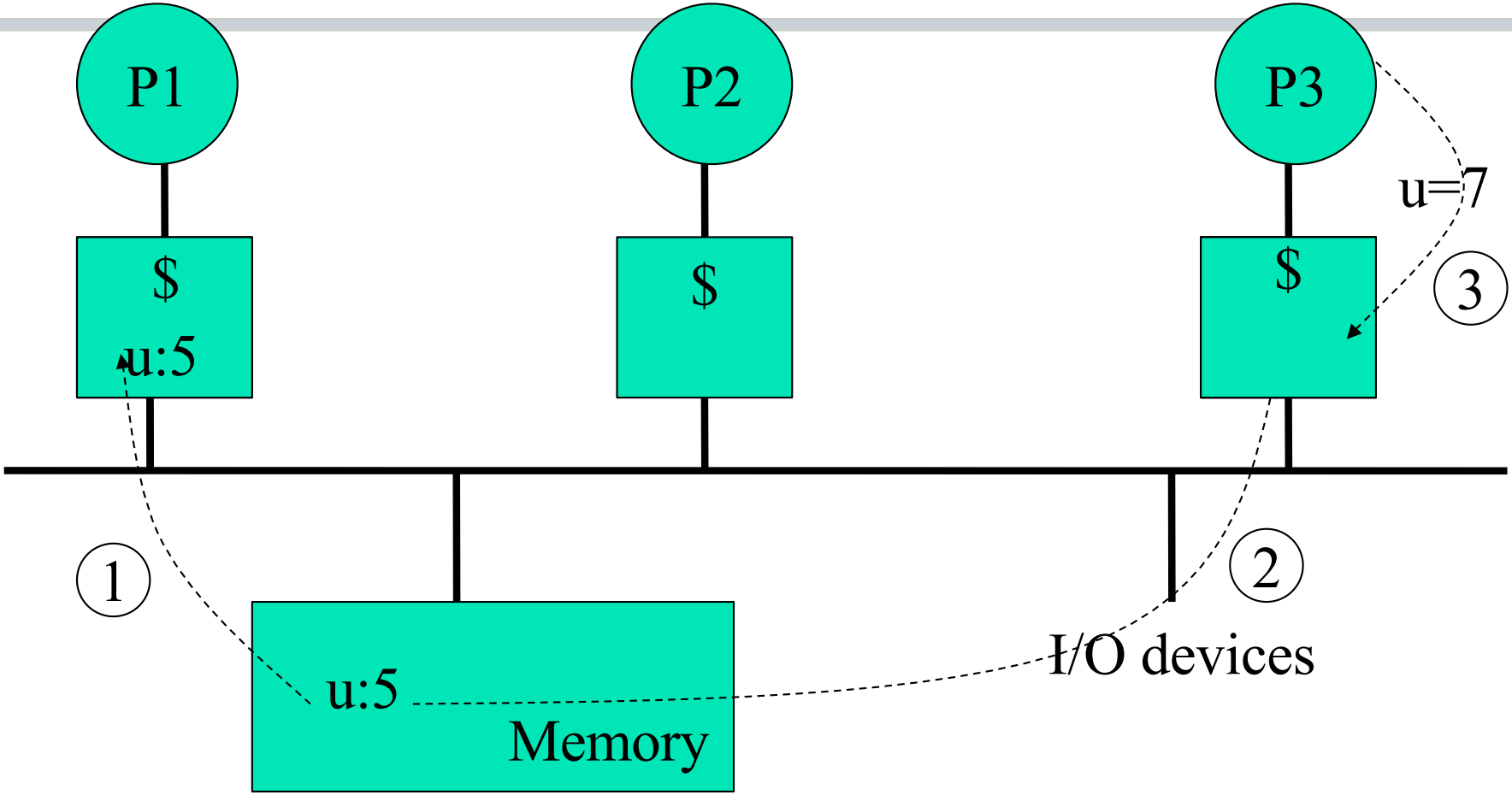
© Parallel Computer Architecture, Culler, Singh & Gupta, MKP

- Assume write back caches, similar problem for write through caches

Cache Coherent Protocols

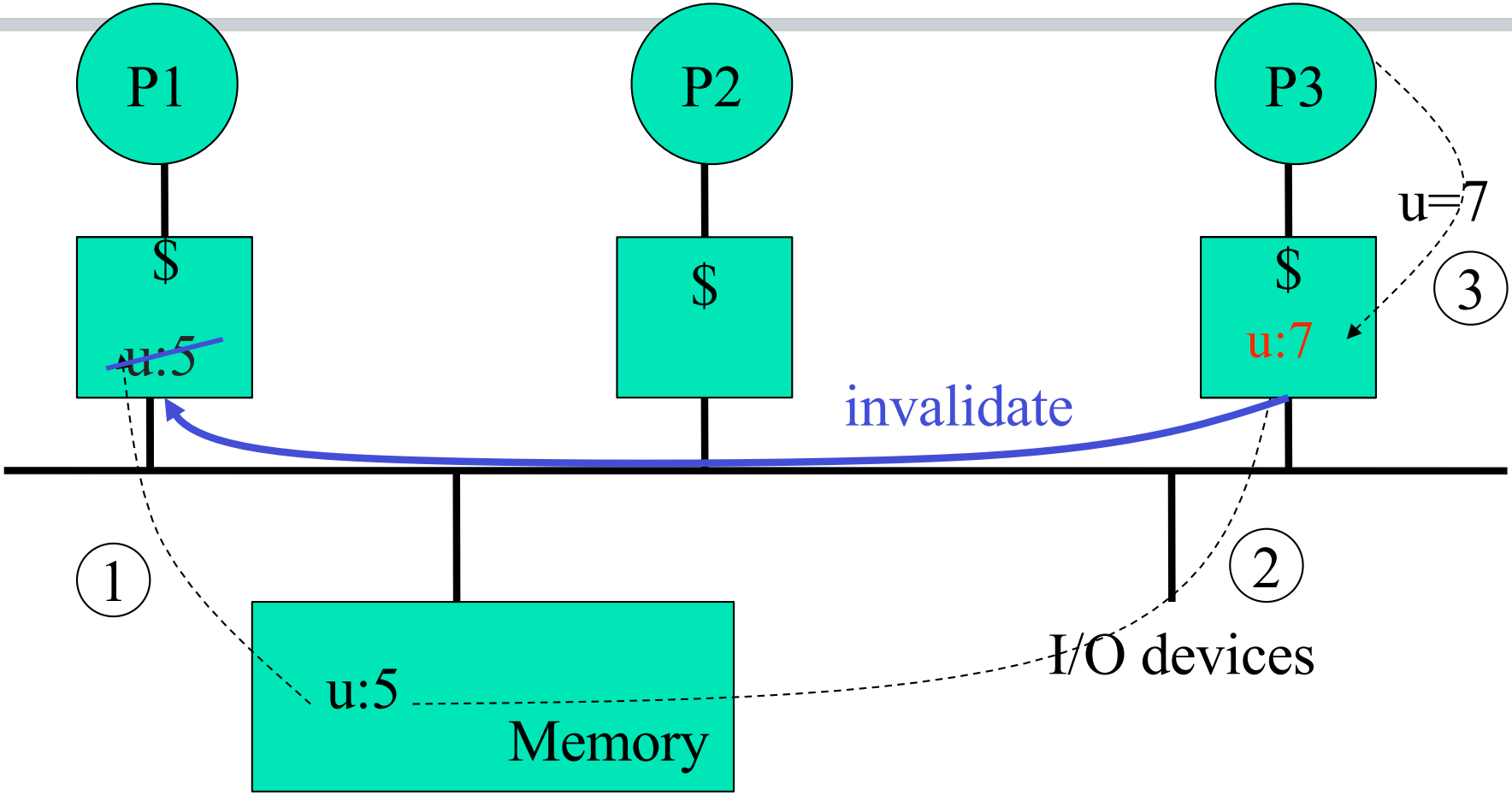
- A HW protocol to maintain coherent caches
- Key: eliminating incoherent copies
  - Write invalidate: get an exclusive access before updating a shared block
    - All other cached copies are invalidated
- Key: bookkeep status of shared blocks
  - Shared, exclusive, invalid, etc.
  - Snooping: local cache events broadcasted; each cache monitors and responds to broadcasts

Example Revisited: Invalidate/Snoopy



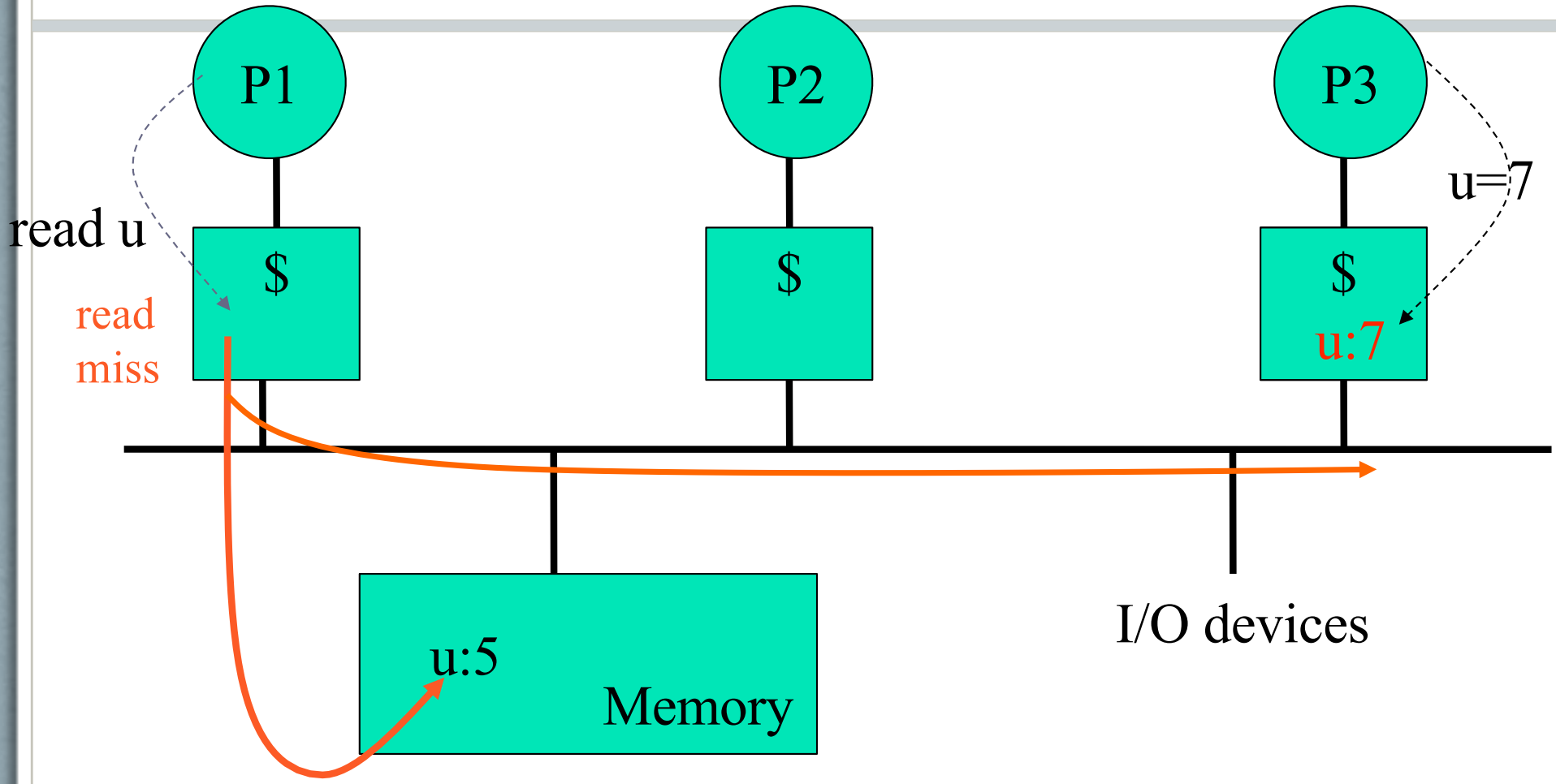
- Write back cache

Example Revisited: Invalidate/Snoopy



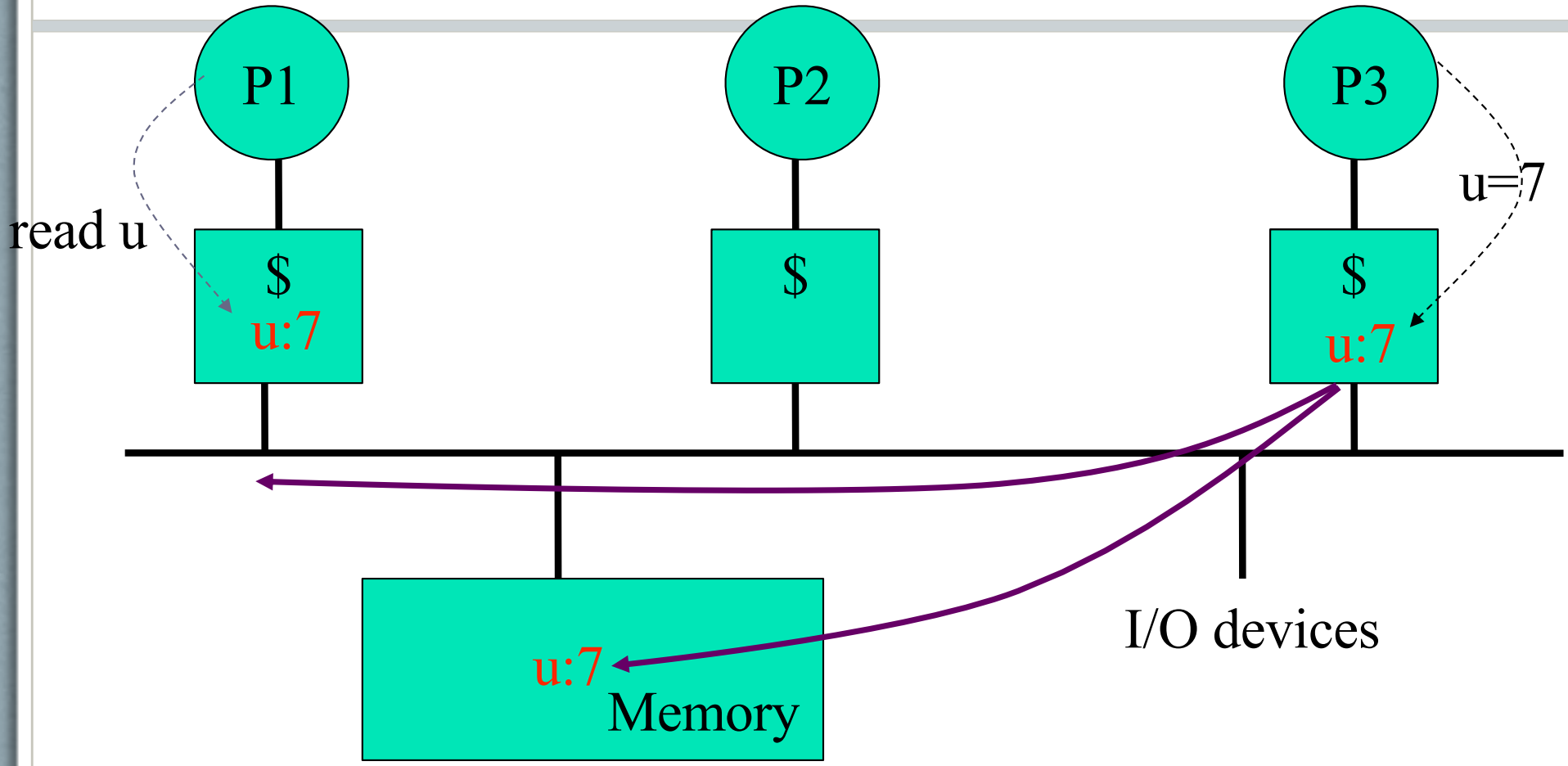
- Invalidate before P3 updating to get an exclusive access to the shared data

Example Revisited: Invalidate/Snoopy



- Invalidated copy triggers read miss: broadcast to all processors

Example Revisited: Invalidate/Snoopy



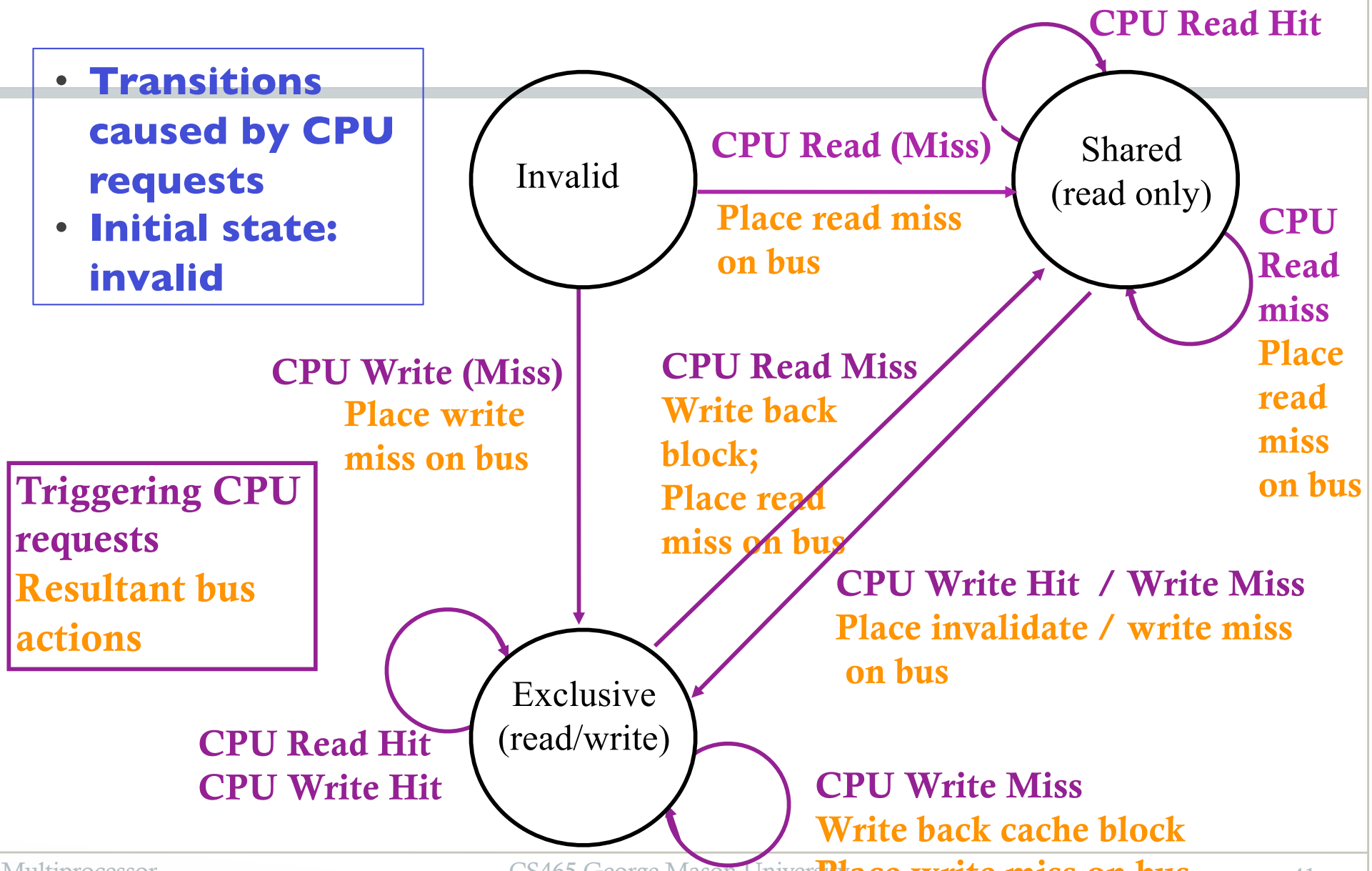
- P3 responds with updated shared data

Write-Invalidate Snoopy Protocol

- Assumption
  - Invalidation protocol, write-back cache (write allocate)
- Every cache maintains a state machine for each block, with states:
  - Invalid: not up-to-date (cache misses)
  - Shared: clean and up-to-date
  - Exclusive: cache has only copy, can update, and dirty
- Transitions can be triggered by either of these:
  - CPU requests (local events)
  - Broadcasted messages (non-local events)

Protocol State Machine - CPU

- **Transitions caused by CPU requests**
- **Initial state: invalid**

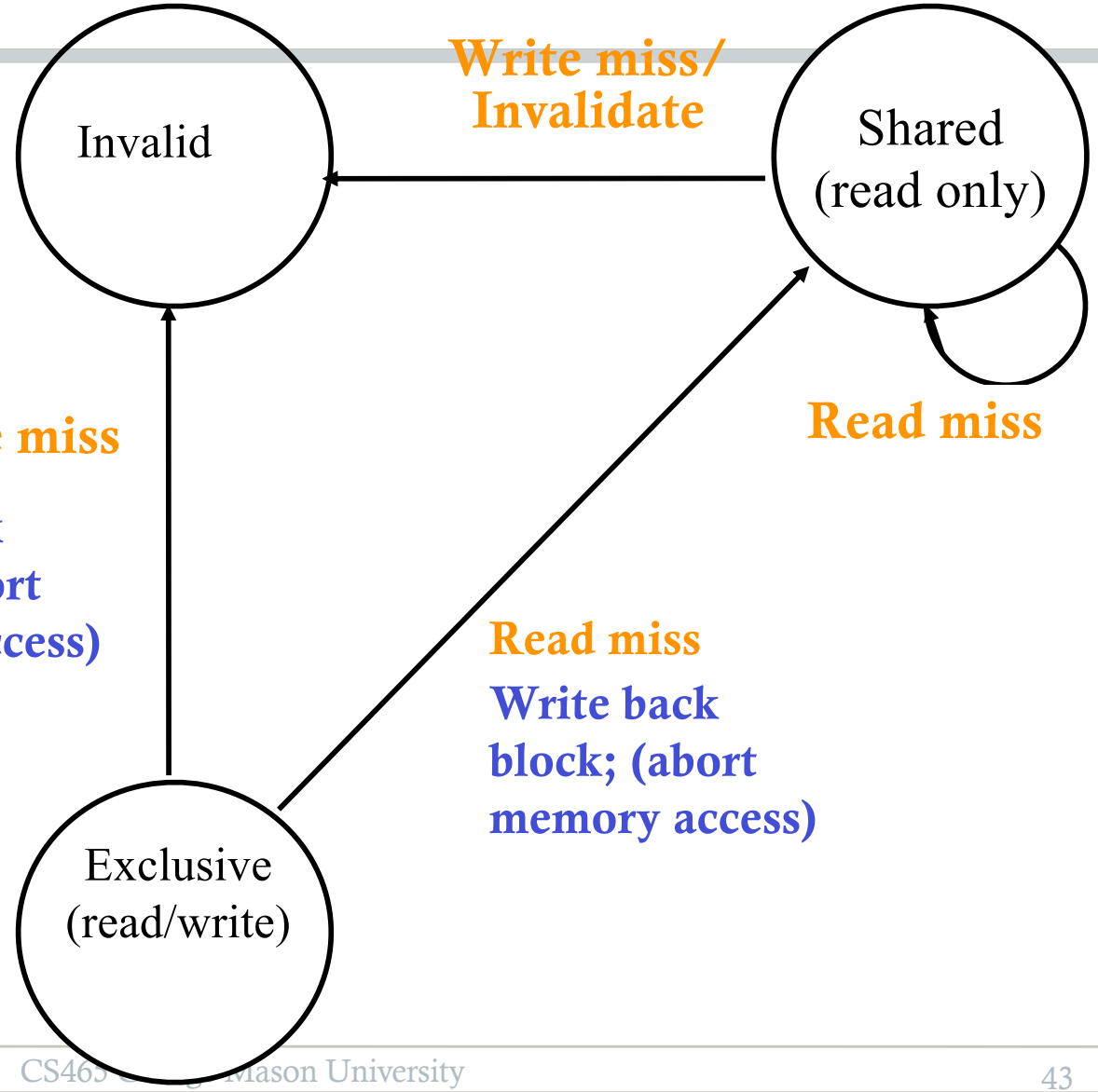


Protocol State Machine

- Why misses for valid (shared or exclusive) blocks?
  - Conflict misses (address collision) are possible (one state machine per block)
- New state after misses?
  - Depends on CPU request type
    - Read misses ➔ shared
    - Write misses➔ exclusive
- Bus activity after misses?
  - Misses are always broadcasted
  - Exclusive means dirty block ➔ write back needed

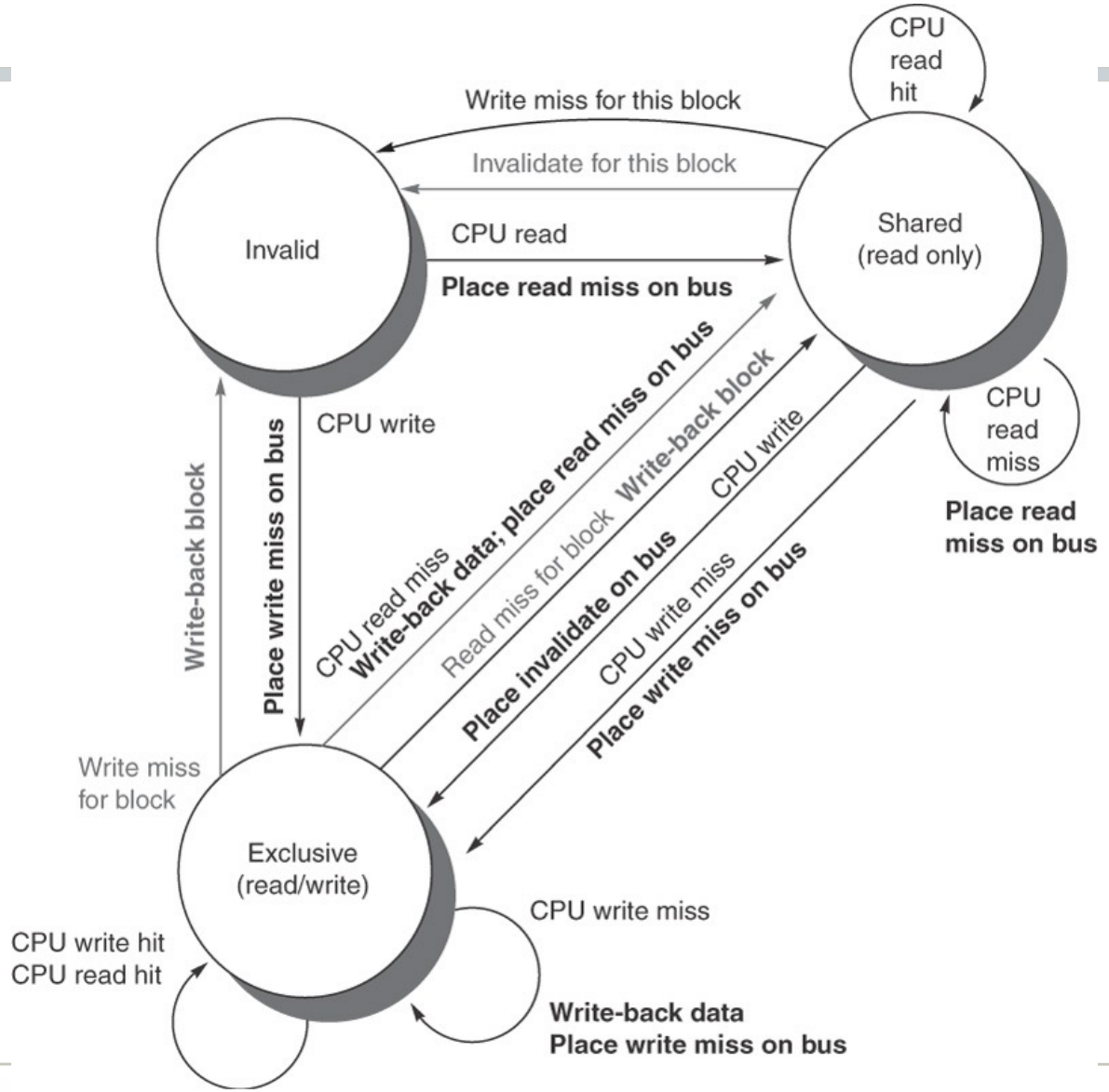
Protocol State Machine - Bus

- **Transitions caused by bus transactions**



Triggering bus transactions
Resultant bus actions

Write Invalidate Snoopy Protocol



Example

	P1			P2			Bus				Memory	
Step	State	Addr	Value	State	Addr	Value	Msg	Proc	Addr	Value	Addr	Value
P1 write 10 to A1												
P1 read A1												
P2 read A1												
P2 write 20 to A1												
P2 write 40 to A2												

Assumes A1 and A2 map to same cache index (conflict),
initial block state: invalid
Write back (write-allocate) cache

Example

	P1			P2			Bus				Memory	
Step	State	Addr	Value	State	Addr	Value	Msg	Proc	Addr	Value	Addr	Value
P1 write 10 to A1							WrMs	P1	A1	--		
P1 read A1												
P2 read A1												
P2 write 20 to A1												
P2 write 40 to A2												

Assumes A1 and A2 map to same cache index (conflict),
initial block state: invalid
Write back (write-allocate) cache

Example

	P1			P2			Bus				Memory	
Step	State	Addr	Value	State	Addr	Value	Msg	Proc	Addr	Value	Addr	Value
P1 write 10 to A1	Excl.	A1	10				WrMs	P1	A1	--		
P1 read A1												
P2 read A1												
P2 write 20 to A1												
P2 write 40 to A2												

Assumes A1 and A2 map to same cache index (conflict),
initial block state: invalid
Write back (write-allocate) cache

Example

	P1			P2			Bus				Memory	
Step	State	Addr	Value	State	Addr	Value	Msg	Proc	Addr	Value	Addr	Value
P1 write 10 to A1	Excl.	A1	10				WrMs	P1	A1	--		
P1 read A1	Excl.	A1	10									
P2 read A1												
P2 write 20 to A1												
P2 write 40 to A2												

Assumes A1 and A2 map to same cache index (conflict),
initial block state: invalid
Write back (write-allocate) cache

Example

	P1			P2			Bus				Memory	
Step	State	Addr	Value	State	Addr	Value	Msg	Proc	Addr	Value	Addr	Value
P1 write 10 to A1	Excl.	A1	10				WrMs	P1	A1	--		
P1 read A1	Excl.	A1	10									
P2 read A1							RdMs	P2	A1	--		
P2 write 20 to A1												
P2 write 40 to A2												

Assumes A1 and A2 map to same cache index (conflict),
initial block state: invalid
Write back (write-allocate) cache

Example

	P1			P2			Bus				Memory	
Step	State	Addr	Value	State	Addr	Value	Msg	Proc	Addr	Value	Addr	Value
P1 write 10 to A1	Excl.	A1	10				WrMs	P1	A1	--		
P1 read A1	Excl.	A1	10									
P2 read A1							RdMs WrBk	P2 P1	A1 A1	-- 10		
P2 write 20 to A1												
P2 write 40 to A2												

Assumes A1 and A2 map to same cache index (conflict),
initial block state: invalid
Write back (write-allocate) cache

Example

	P1			P2			Bus				Memory	
Step	State	Addr	Value	State	Addr	Value	Msg	Proc	Addr	Value	Addr	Value
P1 write 10 to A1	Excl.	A1	10				WrMs	P1	A1	--		
P1 read A1	Excl.	A1	10									
P2 read A1	Shrd.	A1	10	Shrd.	A1	10	RdMs WrBk	P2 P1	A1 A1	-- 10	A1	10
P2 write 20 to A1												
P2 write 40 to A2												

Assumes A1 and A2 map to same cache index (conflict),
initial block state: invalid
Write back (write-allocate) cache

Example

	P1			P2			Bus				Memory	
Step	State	Addr	Value	State	Addr	Value	Msg	Proc	Addr	Value	Addr	Value
P1 write 10 to A1	Excl.	A1	10				WrMs	P1	A1	--		
P1 read A1	Excl.	A1	10									
P2 read A1	Shrd.	A1	10	Shrd.	A1	10	RdMs WrBk	P2 P1	A1 A1	-- 10	A1	10
P2 write 20 to A1							Invd.	P2	A1	--		
P2 write 40 to A2												

Assumes A1 and A2 map to same cache index (conflict),
initial block state: invalid
Write back (write-allocate) cache

Example

	P1			P2			Bus				Memory	
Step	State	Addr	Value	State	Addr	Value	Msg	Proc	Addr	Value	Addr	Value
P1 write 10 to A1	Excl.	A1	10				WrMs	P1	A1	--		
P1 read A1	Excl.	A1	10									
P2 read A1	Shrd.	A1	10	Shrd.	A1	10	RdMs WrBk	P2 P1	A1 A1	-- 10	A1	10
P2 write 20 to A1	Invd.	A1	--	Excl.	A1	20	Invd.	P2	A1	--		
P2 write 40 to A2												

Assumes A1 and A2 map to same cache index (conflict),
initial block state: invalid
Write back (write-allocate) cache

Example

	P1			P2			Bus				Memory	
Step	State	Addr	Value	State	Addr	Value	Msg	Proc	Addr	Value	Addr	Value
P1 write 10 to A1	Excl.	A1	10				WrMs	P1	A1	--		
P1 read A1	Excl.	A1	10									
P2 read A1	Shrd.	A1	10	Shrd.	A1	10	RdMs WrBk	P2 P1	A1 A1	-- 10	A1	10
P2 write 20 to A1	Invd.	A1	--	Excl.	A1	20	Invd.	P2	A1	--		
P2 write 40 to A2							WrBk	P2	A1	20		

Assumes A1 and A2 map to same cache index (conflict),
initial block state: invalid
Write back (write-allocate) cache

Example

	P1			P2			Bus				Memory	
Step	State	Addr	Value	State	Addr	Value	Msg	Proc	Addr	Value	Addr	Value
P1 write 10 to A1	Excl.	A1	10				WrMs	P1	A1	--		
P1 read A1	Excl.	A1	10									
P2 read A1	Shrd.	A1	10	Shrd.	A1	10	RdMs WrBk	P2 P1	A1 A1	-- 10	A1	10
P2 write 20 to A1	Invd.	A1	--	Excl.	A1	20	Invd.	P2	A1	--		
P2 write 40 to A2							WrBk WrMs.	P2 P2	A1 A2	20 --	A1	20

Assumes A1 and A2 map to same cache index (conflict),
initial block state: invalid
Write back (write-allocate) cache

Example

	P1			P2			Bus				Memory	
Step	State	Addr	Value	State	Addr	Value	Msg	Proc	Addr	Value	Addr	Value
P1 write 10 to A1	Excl.	A1	10				WrMs	P1	A1	--		
P1 read A1	Excl.	A1	10									
P2 read A1	Shrd.	A1	10	Shrd.	A1	10	RdMs WrBk	P2 P1	A1 A1	-- 10	A1	10
P2 write 20 to A1	Invd.	A1	--	Excl.	A1	20	Invd.	P2	A1	--		
P2 write 40 to A2				Excl.	A2	40	WrBk WrMs.	P2 P2	A1 A2	20 --	A1	20

Assumes A1 and A2 map to same cache index (conflict),
initial block state: invalid
Write back (write-allocate) cache

Discussion

- What will happen for the three scenarios of cache accesses?
  - Hits
    - Shared + write hit ➔ invalidate + transit to Exclusive
    - Otherwise no state change; no bus traffic
  - Misses without replacement:
    - Invalid to other states
    - Misses broadcasted on bus
  - Misses with replacement:
    - If dirty ➔ write back data; Exclusive transit to other state
    - Misses broadcasted on bus
- Same kinds of misses (compulsory, conflict, capacity) as before are possible

Summary

- Goal: higher performance by using multiple processors
  - SIMD, MIMD
  - Shared memory multiprocessor, message-passing multiprocessor
- Cache coherence
- Parallel processing
  - Developing parallel software (Ch 6.5, 6.7)
  - Challenges (Ch6.2)

Course Evaluation

- Which option below do you think would be more appropriate for CS465?
  - Coding assignment in assembly with straightforward algorithms and basic data structures (like what we had this semester); or
  - More challenging coding assignment in C on architecture related tasks (e.g. a cache simulator)