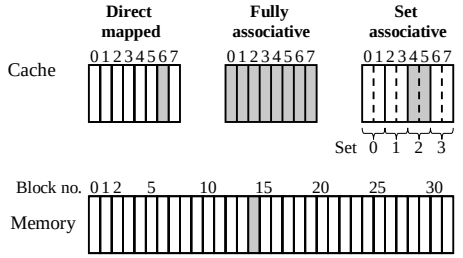
Parallel Systems
- Outline for lecture 3 -

- Caches (a quick review)
- Shared memory multiprocessors
  - Memory hierarchies
  - Cache coherence
  - Snooping protocols
    - » Invalidation protocols (MSI, MESI)
    - » Update protocol (Dragon)
  - Protocol tradeoffs

Caches

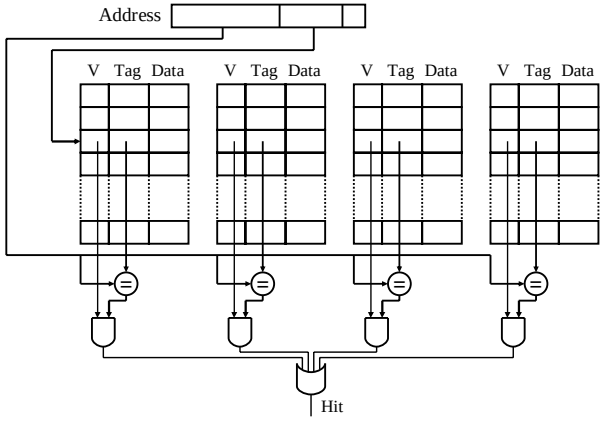
- Caches memories are small, fast buffers that are used to temporarily hold
  - Recently used information
  - Information that might be needed in the near future
- Principle of locality
  - Programs access a relatively small portion of their address space at any instant of time
  - Temporal locality (locality in time): If an item is referenced, it will tend to be referenced again soon.
  - Spatial locality (locality in space): If an item is referenced, items whose addresses are close by will tend to be referenced soon
- Q1: Where can a block be placed in a cache?
- Q2: How is a block found?
- Q3: What block is replaced on a miss?
- Q4: How are writes handled?

Caches
- Block placement -



Direct mapped: (block address) MOD (number of blocks in the cache)
Fully associative: Blocks can go anywhere
Set associative: (block address) MOD (number of sets in the cache)

Caches
- Finding blocks -

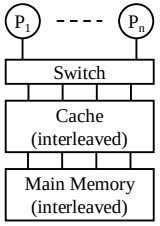


Caches
- Block replacement and write policies -

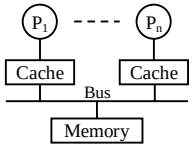
- Block replacement
  - Direct mapped caches: Block can only go in one place
  - Set and fully associative caches: Must choose block to replace
    - » Random
    - » Least-recently used
- Write policies
  - Write through (store through)
    - » Data is written to both the cache and to the memory
  - Write back (copy back)
    - » Data is only written to the cache. The modified data is written back to memory only when it is replaced.
- Further reading
  - Computer Architecture – A Quantitative Approach (Hennessy, Patterson)
  - Computer Organization and Design (Patterson, Hennessy)

Memory Hierarchies

- Shared cache
  - Can be used to connect a small number of processors (2-8)
  - Interconnect between the processors and the shared cache is a critical path
  - Mid-80s: connecting a few processors on a board
  - Today: one strategy for multiprocessors-on-a-chip

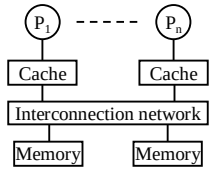


- Bus-based shared memory
  - Widely used for small to medium scale (20-30 processors)
  - Scaling limit comes from the limited bandwidth on the bus

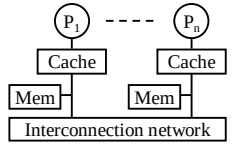


Memory Hierarchies [cont.]

- Dancehall approach
  - Interconnect is a scalable point-to-point network rather than a bus
  - Memory is divided into many logical modules
  - All main memory is uniformly far away from all processors (UMA)
  - Drawback: leads to several “hops” in the interconnect



- Distributed memory
  - Not symmetric
  - Each node has its own local portion of the global main memory
  - Most attractive approach for scalable multiprocessors (>100 processors)



Memory Hierarchies [cont.]

- Caches reduces the bandwidth demand placed on the shared interconnect
  - Use of several private caches rises a challenge of cache coherence
- Approaches
  - Interconnect is visible to all processors => snooping technique
  - Decentralized interconnect => directory based technique
- Fundamental property of memories
  - A set of locations that hold values
  - A read should return the latest value written to that location

Cache Coherence

- Some definitions
  - *Memory operations*: Single read, write, or read-modify-write access to a memory location
  - A memory operation *issues* when it leaves the processor and is presented to the memory system
  - A multiprocessor memory system is coherent if
    - » Operations issued by any process occur in the order in which they were issued to the memory system by that process
    - » The value returned by each read operation is the value written by the last write to that location in the serial order
- Two properties follows by the definitions
  - *Write propagation*: writes become visible to other processes
  - *Write serialization*: all writes to a location are seen in the same order by all processes

Snooping Protocols
- Basic definitions -

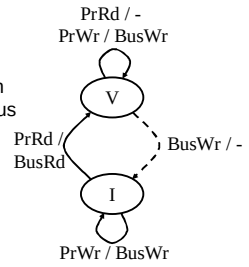
- Coherence is maintained by having all cache controllers “snoop” on the bus and monitor the transactions
- Key properties of a bus that supports coherence
  - All transaction that appear on the bus are visible to all cache controllers
  - All transactions are visible in the same order
- Simplest approach
  - Single level write through caches
    - » This was the approach used in the first commercial bus-based SMP’s
- Coherence protocols
  - Invalidation-based
  - Update-based

Snooping Protocols
- Basic definitions [cont.]-

- Snoopy protocols ties together *bus transactions* and the *state transition diagram* associated with a cache block
- Bus transactions
  - arbitration, command (read&write), and data transfer
- State transition diagram
  - Each cache block has a state associated with it, along with the tag and data (e.g., valid or invalid)
  - State changes is the same for all blocks and all caches, but the current state of a block in different caches may be different
  - Two inputs to cache controller
    - » Memory requests issued by the processor
    - » The bus snooper informs about bus transactions

Snooping Protocols
- A two-state write-through invalidation protocol -

- Notation “A / B” means
  - If transaction A is observed, then transaction B is generated
- Write-through
  - Writes are serialized by the order in which they appear on the shared bus
  - Drawback: Every store consumes bandwidth on the shared bus
    => Poor scalability

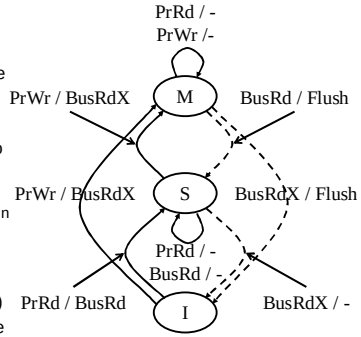


Snooping Protocols
- A three-state write-back invalidation protocol -

- Write-back caches
  - Processors can write in their local caches without any bus transactions
  - Actions on a write miss
    - » Read block from memory
    - » Update block
    - » Retain block in modified (dirty) state so it can be written back to memory upon replacement
  - *Owner*: The cache must supply the data upon a request for that block
  - *Exclusive copy*: This is the only cache with a valid copy of the block
  - States used: MSI
    - » Modified (M)
      - Only this cache has a valid copy of the block.
      - Main memory may or may not have a valid copy
    - » Shared (S)
      - Block is present in an unmodified state. Other caches may also have an copy
    - » Invalid (I)

Snooping Protocols
- A three-state write-back invalidation protocol [cont.] -

- Transactions
  - Bus Read (*BusRd*)
    - » Generated by a PrRd that misses in the cache
  - Bus Read Exclusive (*BusRdX*)
    - » Generated by a PrWr to a block that is
      - Not in the cache, or
      - In the cache, but not in the modified state
    - » All other caches are invalidated
  - Bus Write Back (*BusWB*)
    - » Generated by the cache controller on a write back



Snooping Protocols
- A three-state write-back invalidation protocol [cont.] -

Example

Processor action	State in P1	State in P2	State in P3	Bus action	Data supplied by
P1 reads U	S	-	-	BusRd	Memory
P3 reads U	S	-	S	BusRd	Memory
P3 writes U	I	-	M	BusRdX	Memory
P1 reads U	S	-	S	BusRd	P3 cache
P2 reads U	S	S	S	BusRd	Memory

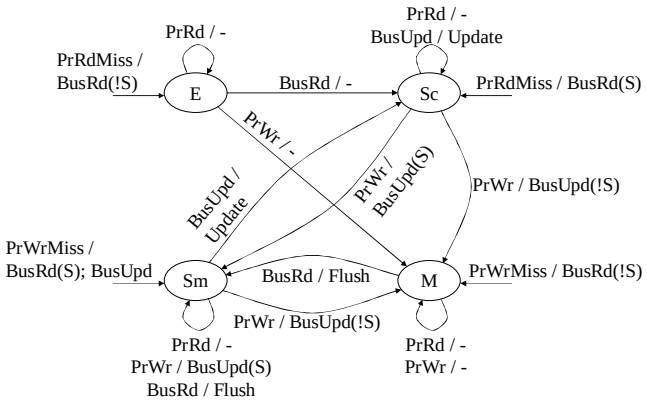
Snooping Protocols
- A four-state write-back invalidation protocol -

- Four states are used (MESI) - *Illinois protocol*
  - Modified (M)
  - Exclusive (E)
    - » Only this cache has a copy of the block and it is not modified
  - Shared (S)
  - Invalid (I)
- A new signal must be available on the interconnect
  - Shared: Determine (on BusRd) if other caches holds a copy of this block
- Problem with three-state (MSI) protocol:
  - Two bus transactions are generated when the processor reads in and modifies a data item (even though there are never any sharers)
- **Exercise**: Write a state transition diagram for the MESI protocol.

Snooping Protocols
- A four-state write-back update protocol -

- Four states
  - Exclusive-clean (E)
    - » Same meaning as in MESI
  - Shared-clean (Sc)
    - » Several caches may have a copy of this block
  - Shared-modified (Sm)
    - » Several caches may have a copy of this block, and it is this cache's responsibility to update the main memory when the block is replaced from the cache
  - Modified (M)
    - » Same meaning as in MESI
- Transactions
  - No invalid state => two more request types: PrRdMiss, PrWrMiss
  - New transaction - *BusUpd*: Broadcast the updated value on the bus so that all other caches can update themselves

Snooping Protocols
- A four-state write-back update protocol [cont.] -



Snooping Protocols
- A four-state write-back update protocol [cont.] -

Example

Processor action	State in P1	State in P2	State in P3	Bus action	Data supplied by
P1 reads U	E	-	-	BusRd	Memory
P3 reads U	Sc	-	Sc	BusRd	Memory
P3 writes U	Sc	-	Sm	BusUpd	P3 cache
P1 reads U	Sc	-	Sm	-	-
P2 reads U	Sc	Sc	Sm	BusRd	P3 cache

Protocol Tradeoffs

- The coherence protocol is a crucial design issue for a multiprocessor
  - Protocol class (invalidation or update)
  - Protocol states and actions
  - Protocol decisions interact with other design issues (e.g., latency and bandwidth demand on the interconnect)
- Goals:
  - Meet a cost-performance target
  - Have a well balanced system (no bottlenecks)
- Use simulation results to evaluate effect of protocol choices

Protocol Tradeoffs
- State transitions per 1000 data references -

Barnes-Hut		To				
		NP	I	E	S	M
From	NP	0	0	0,0011	0.0362	0,0035
	I	0,0201	0	0,0001	0,1856	0,0010
	E	0	0	0,0153	0,0002	0,0010
	S	0,0029	0,2130	0	97,1712	0,1253
	M	0,0013	0,0010	0	0,1277	902,782

Raytrace		NP	I	E	S	M
From	NP	0	0	1,3358	0,1549	0,0026
	I	0,0242	0	0	0,3403	0
	E	0,8664	0	29,02	0,3639	0,0175
	S	1,1181	0,3750	0	310,95	0,2898
	M	0,0559	0,0001	0	0,2970	661,01

Protocol Tradeoffs
- Cache misses -

- Cache misses in uniprocessor context
  - *Compulsory* (cold start) misses
    - » First reference to a memory block by a processor
  - *Capacity* misses:
    - » All blocks that are referenced by a processor during the execution of a program do not fit in the cache
  - *Conflict* (collision) misses:
    - » Occurs when the collection of blocks referenced by a program maps to a single cache set does not fit in the set
  - Tradeoff examples
    - » Conflict misses can be reduced by *reducing* the block size
    - » Cold start misses can be reduced by *increasing* the block size

Protocol Tradeoffs
- Cache misses [cont.] -

- Coherence misses:
  - Occurs when blocks of data are shared among multiple caches
  - *True sharing*: A data word produced (written) by one processor is used (read or written) by another.
    - » The miss truly communicates newly defined data values
  - *False sharing*: Independent data words accessed by different processors happen to be placed in the same memory (cache) block
    - » False sharing is an example of artifactual communication
    - » Increases with larger block size

Summary

- Memory hierarchies and cache coherence problem
- Cache coherence through *snooping protocols*
  - Invalidation-based protocols
    - » Simple protocol for write-through caches
    - » MSI & MESI for write-back caches
  - Update-based protocol
- Coherence protocol tradeoffs
  - Frequency analysis of state transitions
  - Tradeoffs in cache block sizes