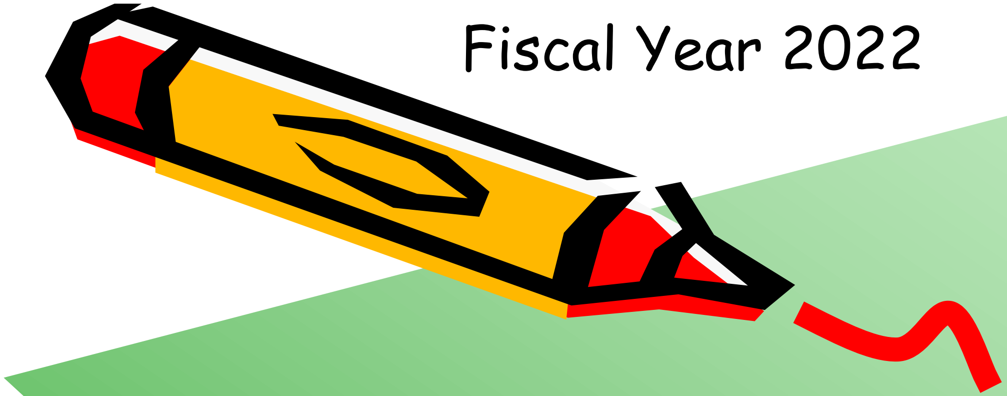


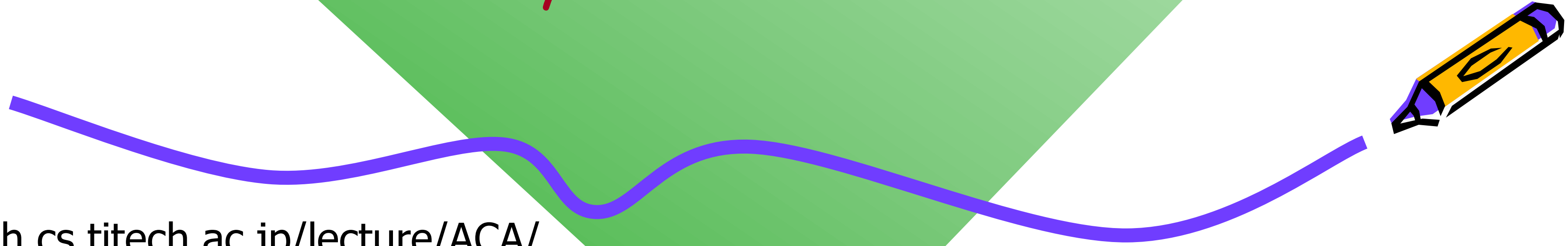
Fiscal Year 2022

Ver. 2023-01-25a

Course number: CSC.T433
School of Computing,
Graduate major in Computer Science

Advanced Computer Architecture

12. Thread Level Parallelism: Coherence and Synchronization

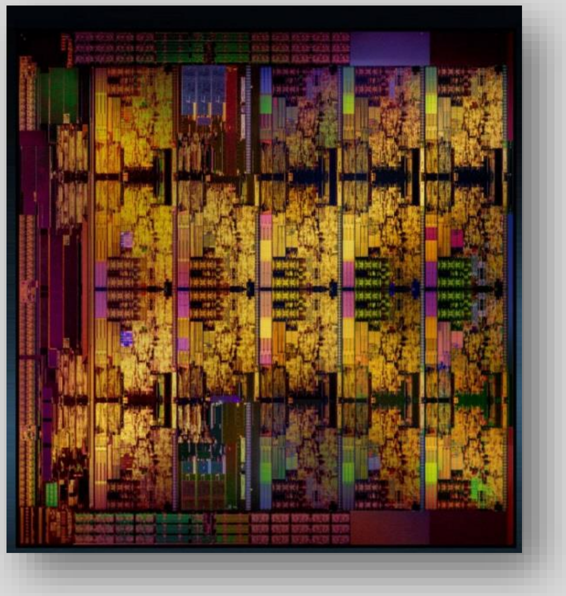


www.arch.cs.titech.ac.jp/lecture/ACA/
Room No.W831, HyFlex
Mon 13:45-15:25, Thr 13:45-15:25

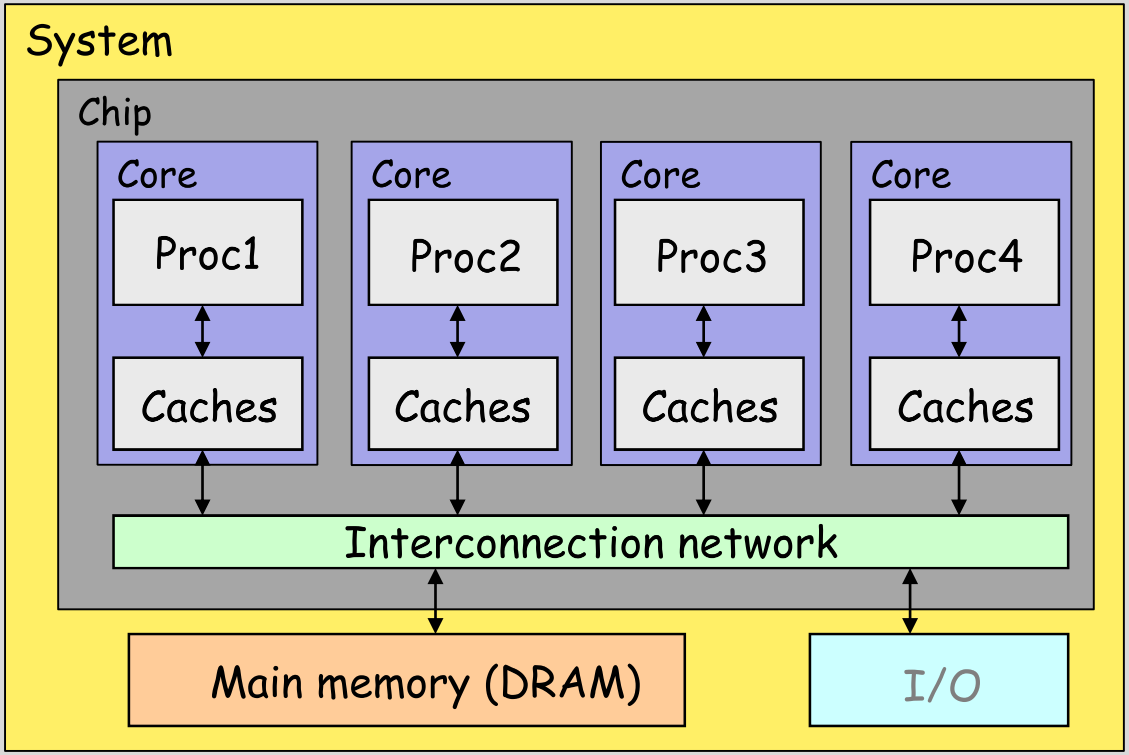
Kenji Kise, Department of Computer Science
kise _at_ c.titech.ac.jp

Shared memory many-core architecture

- The single-chip integrates many cores (conventional processors) and an interconnection network.
- All the processors can access the same address space of the main memory (shared memory) through an interconnection network.
- The shared memory or shared address space (SAS) is used as a means for communication between the processors.



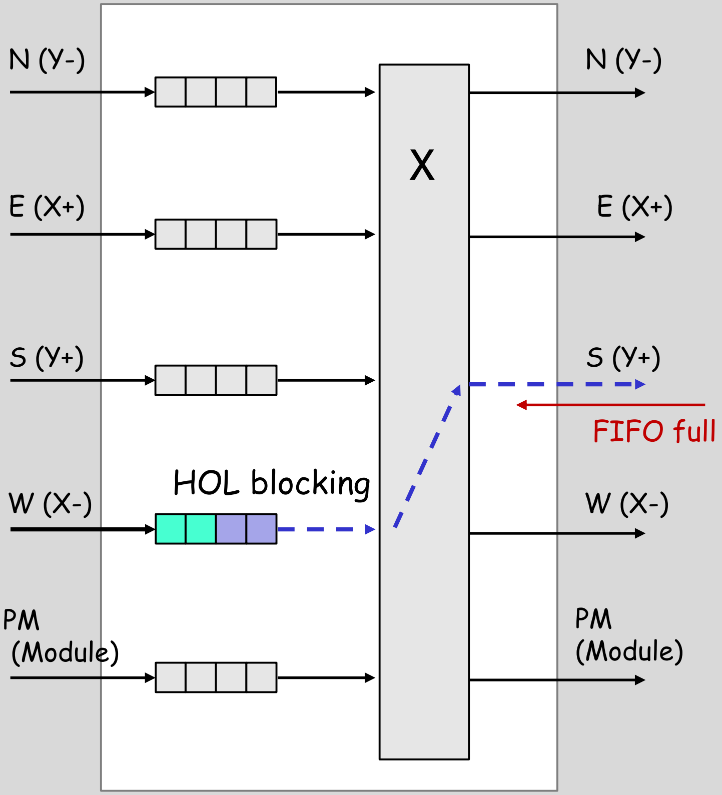
Intel Skylake-X, Core i9-7980XE, 2017



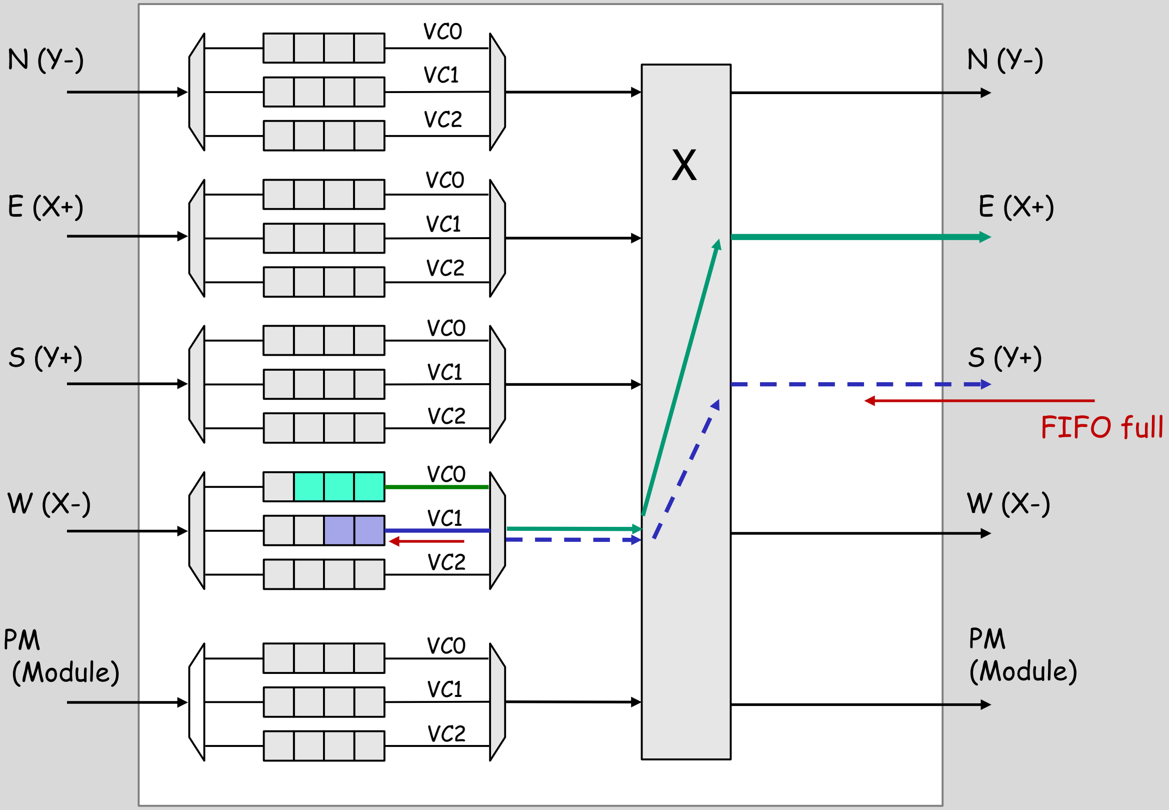
Datapath of Virtual Channel (VC) NoC router

- To mitigate head-of-line (HOL) blocking, virtual channels are used

Flit	Route info	VC	Payload



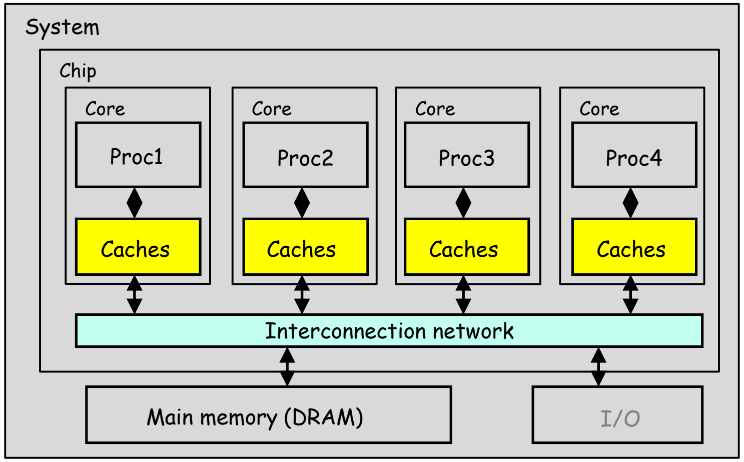
Simple NoC router



VC NoC router

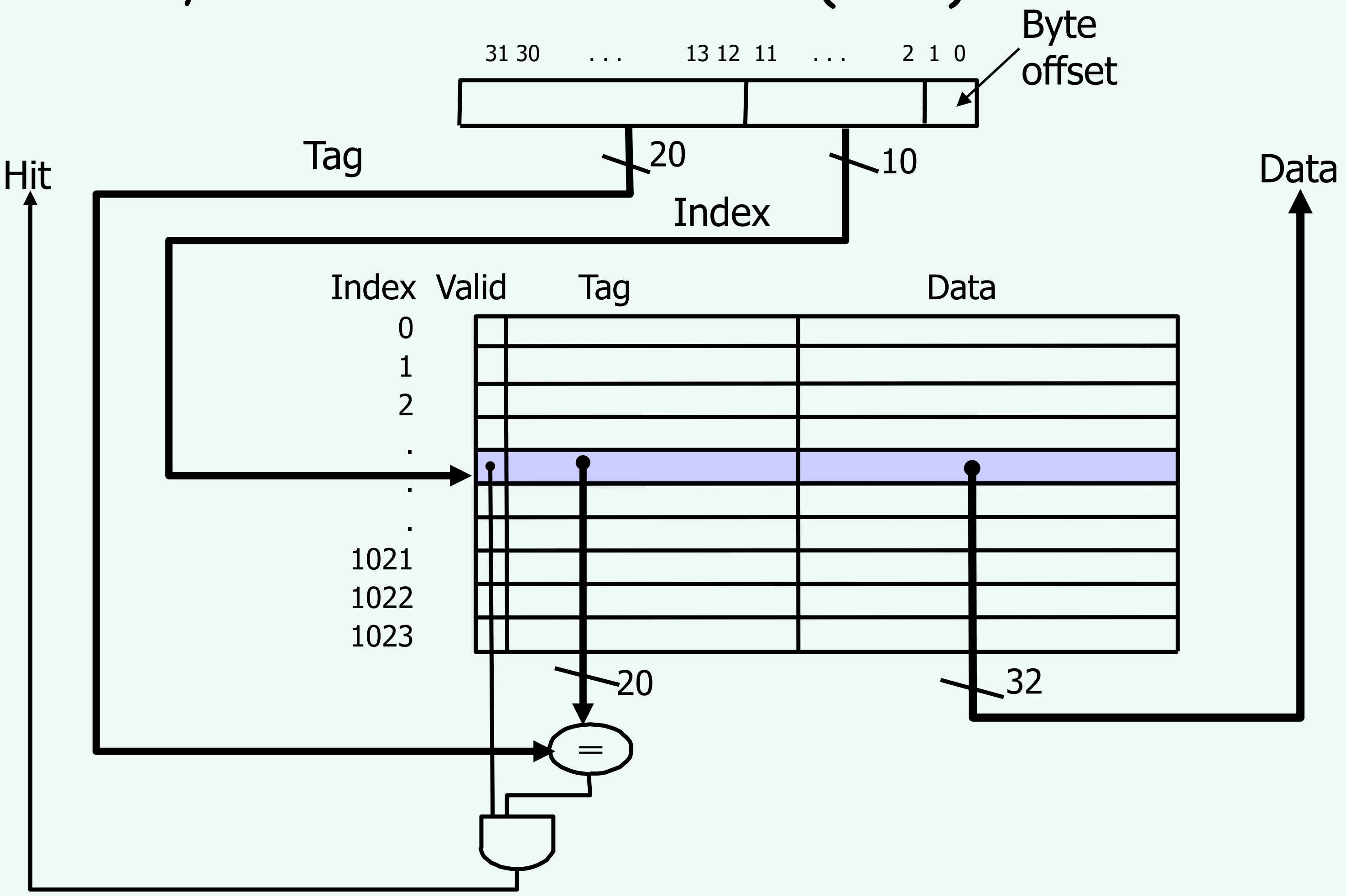
Key components of many-core processors

- Interconnection network
  - connecting many modules on a chip achieving high throughput and low latency
- Main memory and caches
  - Caches are used to reduce latency and to lower network traffic
  - A parallel program has private data and shared data
  - New issues are cache coherence and memory consistency
- Core
  - High-performance superscalar processor providing a hardware mechanism to support thread synchronization



MIPS Direct Mapped Cache Example

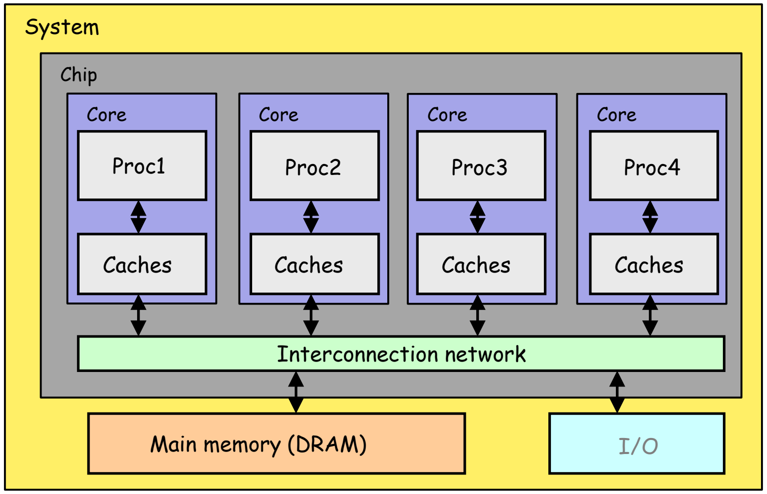
- One word/block, cache size = 1K words (4KB)



What kind of locality are we taking advantage of?

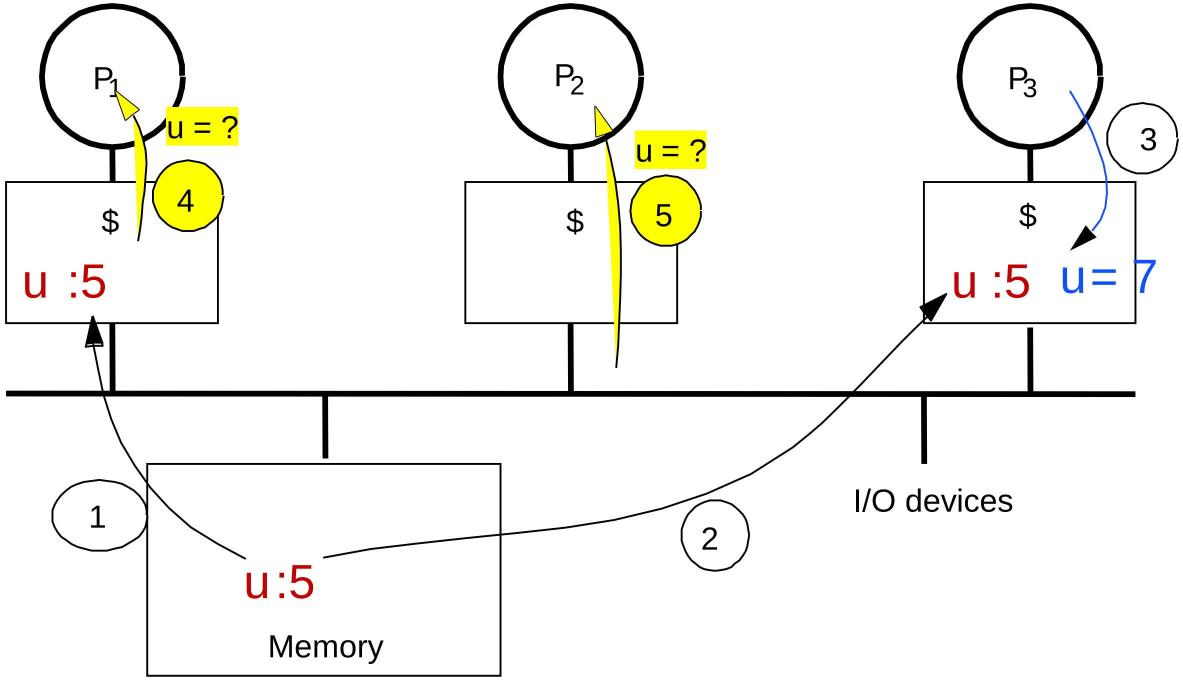
Cache writing policy

- Write-through
  - writing is done synchronously both to the cache and to the main memory. All stores update the main memory.
- Write-back
  - initially, writing is done only to the cache. The write to the main memory is postponed until the modified content is about to be replaced by another cache block.
  - reduces the required network and memory bandwidth.
- Which policy is better for many-core?



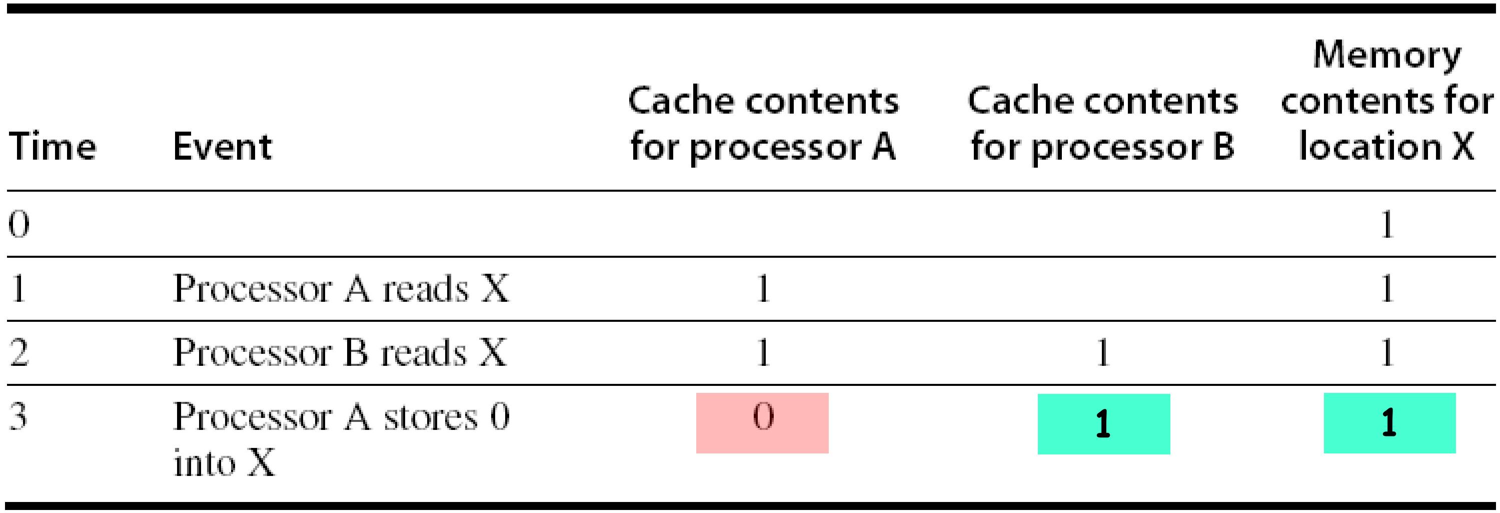
Cache coherence problem

- Processors (cores) see different values for shared data u after event 3
- With write-back caches, value written back to memory depends on which cache flushes or writes back value when
    - Processes accessing main memory may see stale (out-of-date) value
- Unacceptable for programming, and its frequent!



Cache coherence problem

- Processors may see different values through their caches
  - assuming a write-back cache
  - after the value of X has been written by A, A's cache contains the new value, but B's cache and the main memory do not



Time	Event	Cache contents for processor A	Cache contents for processor B	Memory contents for location X
0				1
1	Processor A reads X	1		1
2	Processor B reads X	1	1	1
3	Processor A stores 0 into X	0	**1**	**1**

this slide is to be used as a whiteboard

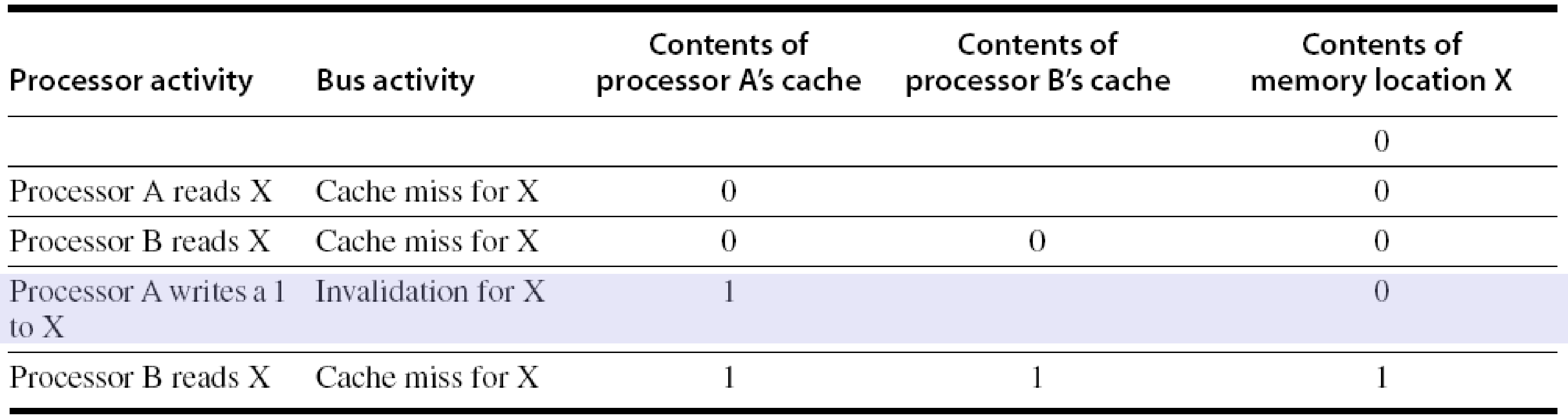
Cache coherence and enforcing coherence

- Cache coherence
  - All reads by any processor must return the most recently written value
  - Writes to the same location by any two processors are seen in the same order by all processors

- Cache coherence protocols
  - Snooping (write invalidate / write update)
    - Each core tracks sharing status of each block
  - Directory based
    - Sharing status of each block kept in one location

Snooping coherence protocols using bus network

- Write invalidate
  - On write, invalidate all other copies by an invalidate broadcast
  - Use bus itself to serialize
    - Write cannot complete until bus access is obtained



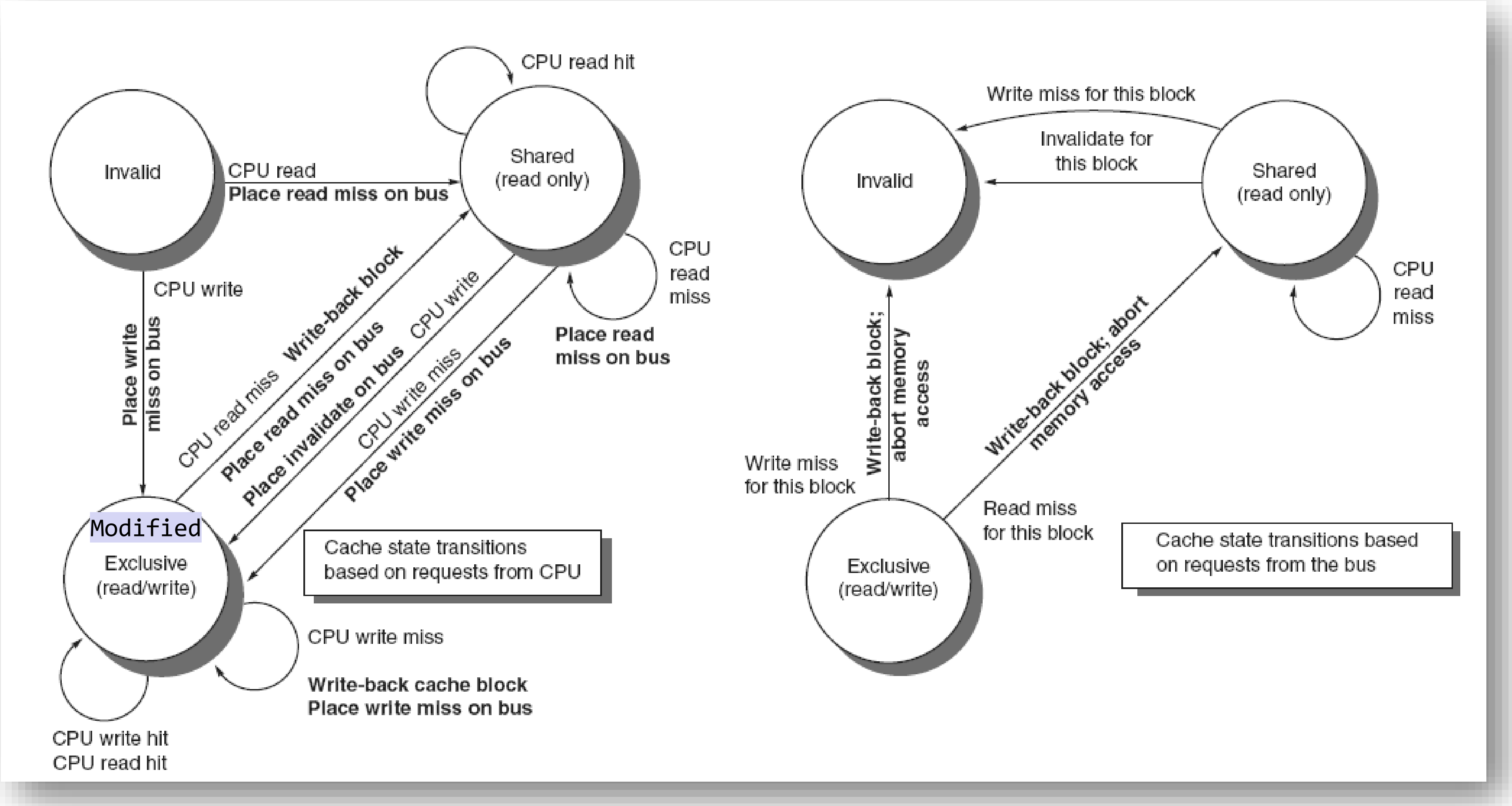
Processor activity	Bus activity	Contents of processor A's cache	Contents of processor B's cache	Contents of memory location X
				0
Processor A reads X	Cache miss for X	0		0
Processor B reads X	Cache miss for X	0	0	0
Processor A writes a 1 to X	Invalidation for X	1		0
Processor B reads X	Cache miss for X	1	1	1

- Write update
  - On write, update all copies

this slide is to be used as a whiteboard

Snooping coherence protocols using bus network

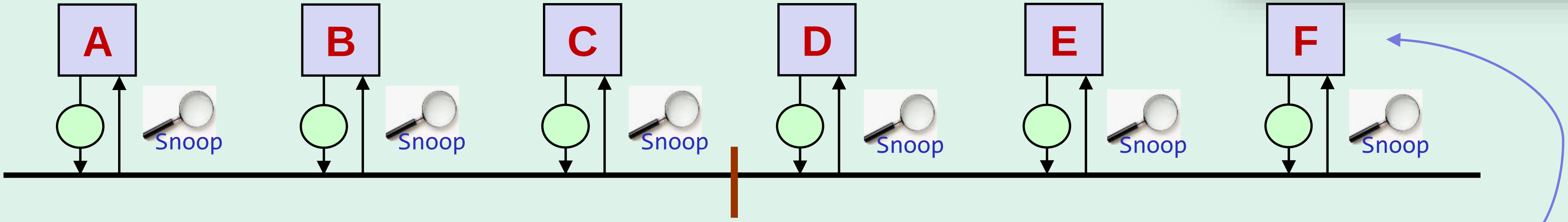
- A write invalidate, cache coherence protocol for a private write-back cache showing the states and state transitions for each block in the cache



MSI (Modified, Shared, Invalid) protocol

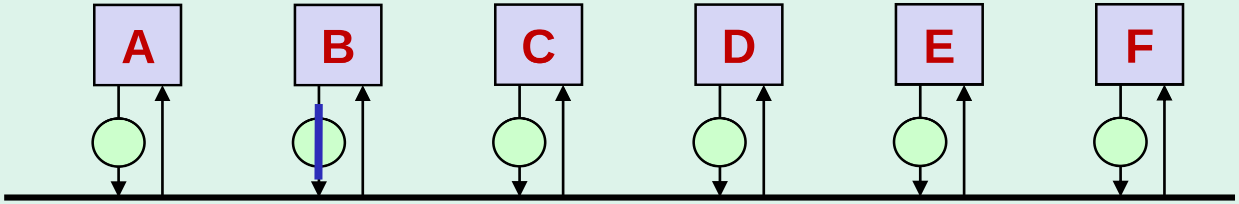
Bus Network

- N cores (□), N switch (○), 1 link (the bus)
- Only 1 simultaneous transfer at a time
  - NB (best case) = link (bus) bandwidth x 1
  - BB (worst case) = link (bus) bandwidth x 1
- All processors can snoop the bus



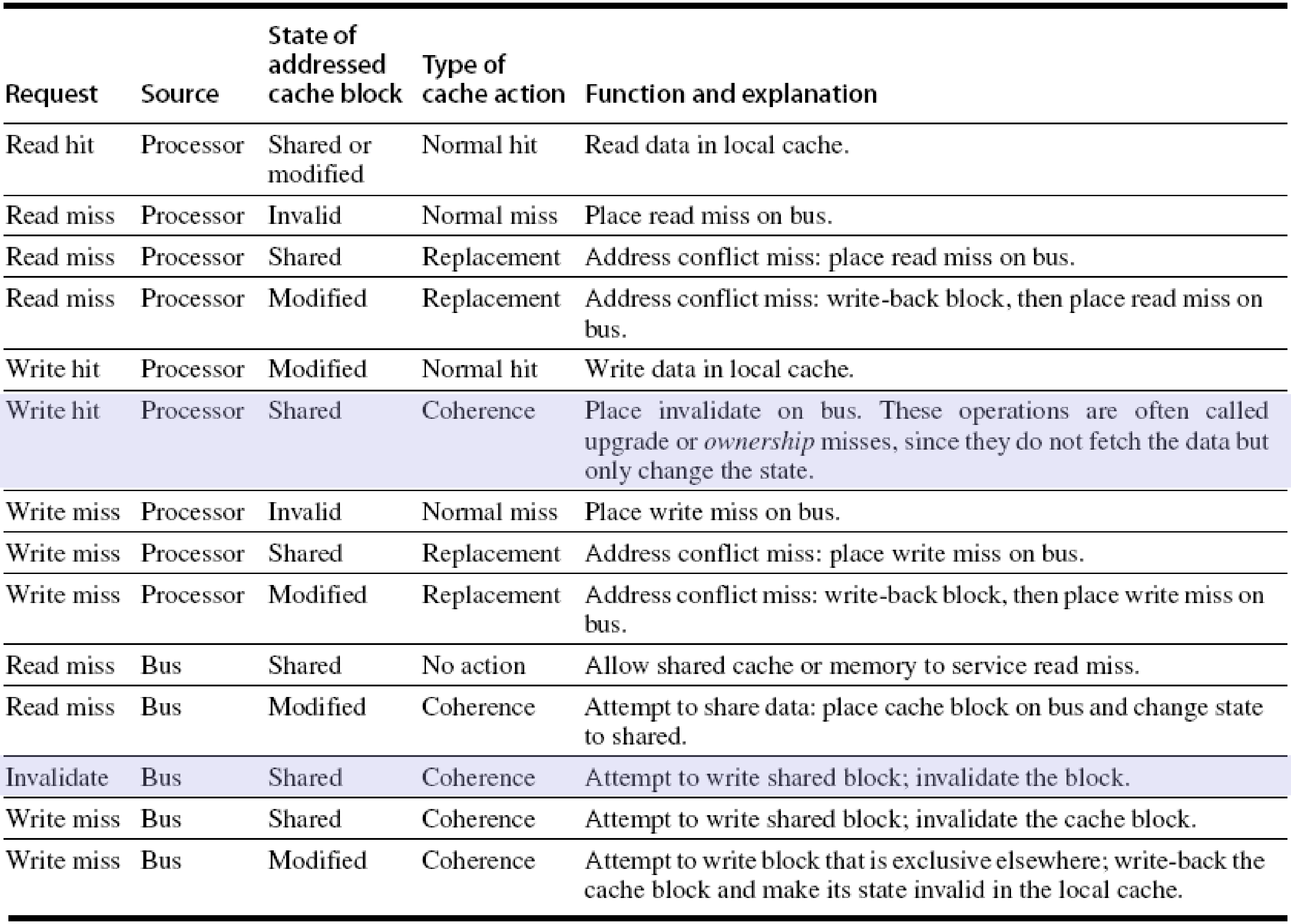
Core or processor node

The case where core B sends a packet to someone



Snooping coherence protocols using bus network

- The coherence mechanism of a private cache

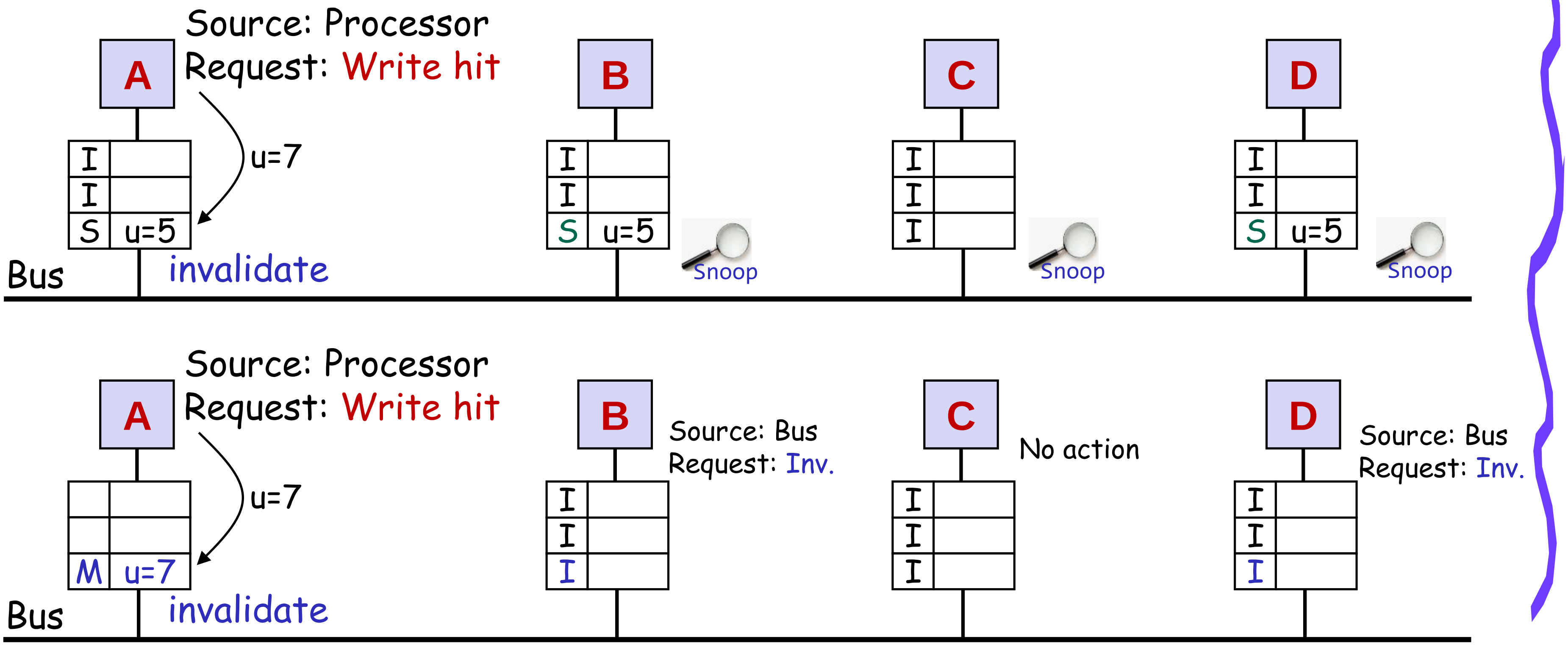


	Request	Source	State of addressed cache block	Type of cache action	Function and explanation
	Read hit	Processor	Shared or modified	Normal hit	Read data in local cache.
	Read miss	Processor	Invalid	Normal miss	Place read miss on bus.
	Read miss	Processor	Shared	Replacement	Address conflict miss: place read miss on bus.
	Read miss	Processor	Modified	Replacement	Address conflict miss: write-back block, then place read miss on bus.
	Write hit	Processor	Modified	Normal hit	Write data in local cache.
C1	Write hit	Processor	Shared	Coherence	Place invalidate on bus. These operations are often called upgrade or *ownership* misses, since they do not fetch the data but only change the state.
	Write miss	Processor	Invalid	Normal miss	Place write miss on bus.
	Write miss	Processor	Shared	Replacement	Address conflict miss: place write miss on bus.
	Write miss	Processor	Modified	Replacement	Address conflict miss: write-back block, then place write miss on bus.
	Read miss	Bus	Shared	No action	Allow shared cache or memory to service read miss.
C2	Read miss	Bus	Modified	Coherence	Attempt to share data: place cache block on bus and change state to shared.
C3	Invalidate	Bus	Shared	Coherence	Attempt to write shared block; invalidate the block.
C4	Write miss	Bus	Shared	Coherence	Attempt to write shared block; invalidate the cache block.
C5	Write miss	Bus	Modified	Coherence	Attempt to write block that is exclusive elsewhere; write-back the cache block and make its state invalid in the local cache.

Coherence 1 (C1) and Coherence3 (C3)

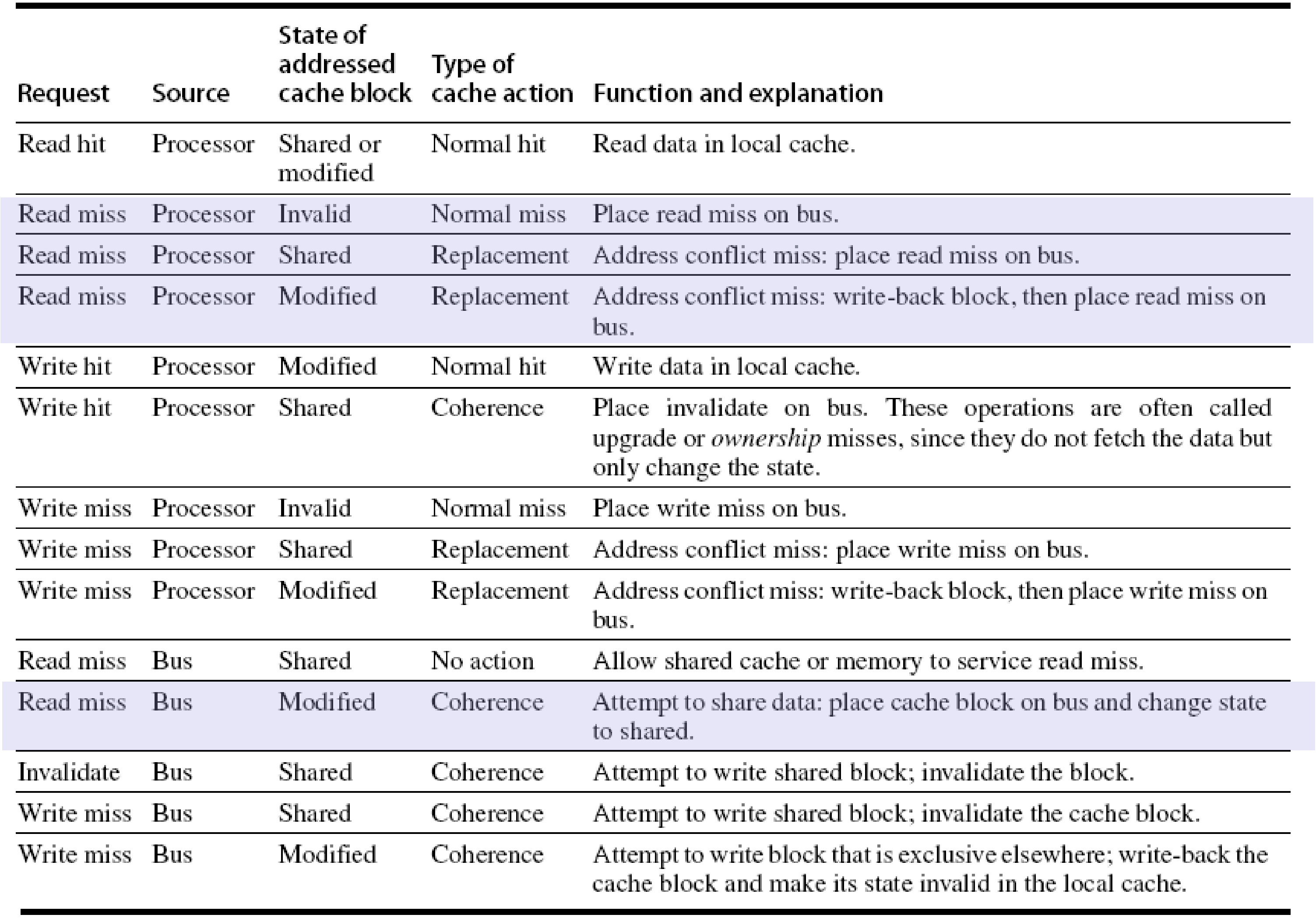
- C1 (Core A)
  - Source: Processor
  - State: Shared
  - Request: Write hit
  - Function: Place invalidate on bus

- C3 (Core B, D)
  - Source: Bus
  - State: Shared
  - Request: Invalidate
  - Function: attempt to write shared block; invalidate the block



Snooping coherence protocols using bus network

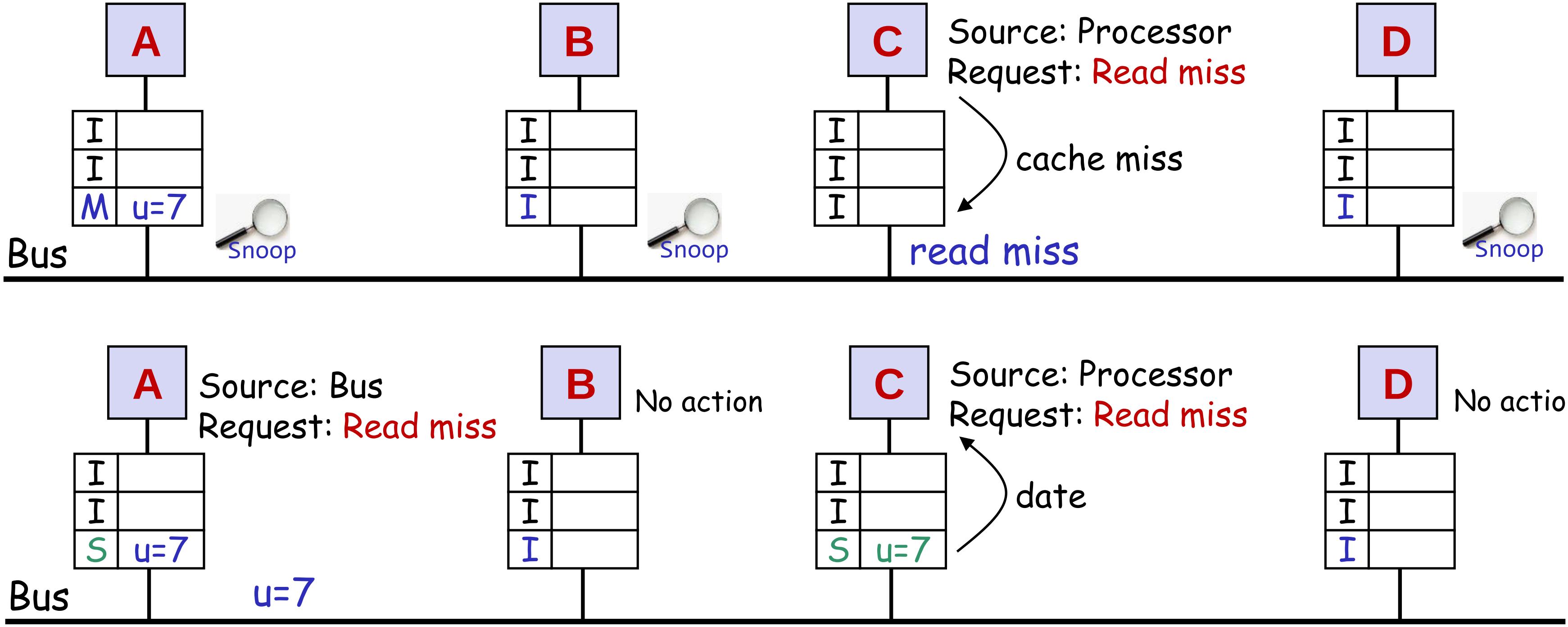
- The coherence mechanism of a private cache



	Request	Source	State of addressed cache block	Type of cache action	Function and explanation
	Read hit	Processor	Shared or modified	Normal hit	Read data in local cache.
	Read miss	Processor	Invalid	Normal miss	Place read miss on bus.
	Read miss	Processor	Shared	Replacement	Address conflict miss: place read miss on bus.
	Read miss	Processor	Modified	Replacement	Address conflict miss: write-back block, then place read miss on bus.
	Write hit	Processor	Modified	Normal hit	Write data in local cache.
C1	Write hit	Processor	Shared	Coherence	Place invalidate on bus. These operations are often called upgrade or *ownership* misses, since they do not fetch the data but only change the state.
	Write miss	Processor	Invalid	Normal miss	Place write miss on bus.
	Write miss	Processor	Shared	Replacement	Address conflict miss: place write miss on bus.
	Write miss	Processor	Modified	Replacement	Address conflict miss: write-back block, then place write miss on bus.
	Read miss	Bus	Shared	No action	Allow shared cache or memory to service read miss.
C2	Read miss	Bus	Modified	Coherence	Attempt to share data: place cache block on bus and change state to shared.
C3	Invalidate	Bus	Shared	Coherence	Attempt to write shared block; invalidate the block.
C4	Write miss	Bus	Shared	Coherence	Attempt to write shared block; invalidate the cache block.
C5	Write miss	Bus	Modified	Coherence	Attempt to write block that is exclusive elsewhere; write-back the cache block and make its state invalid in the local cache.

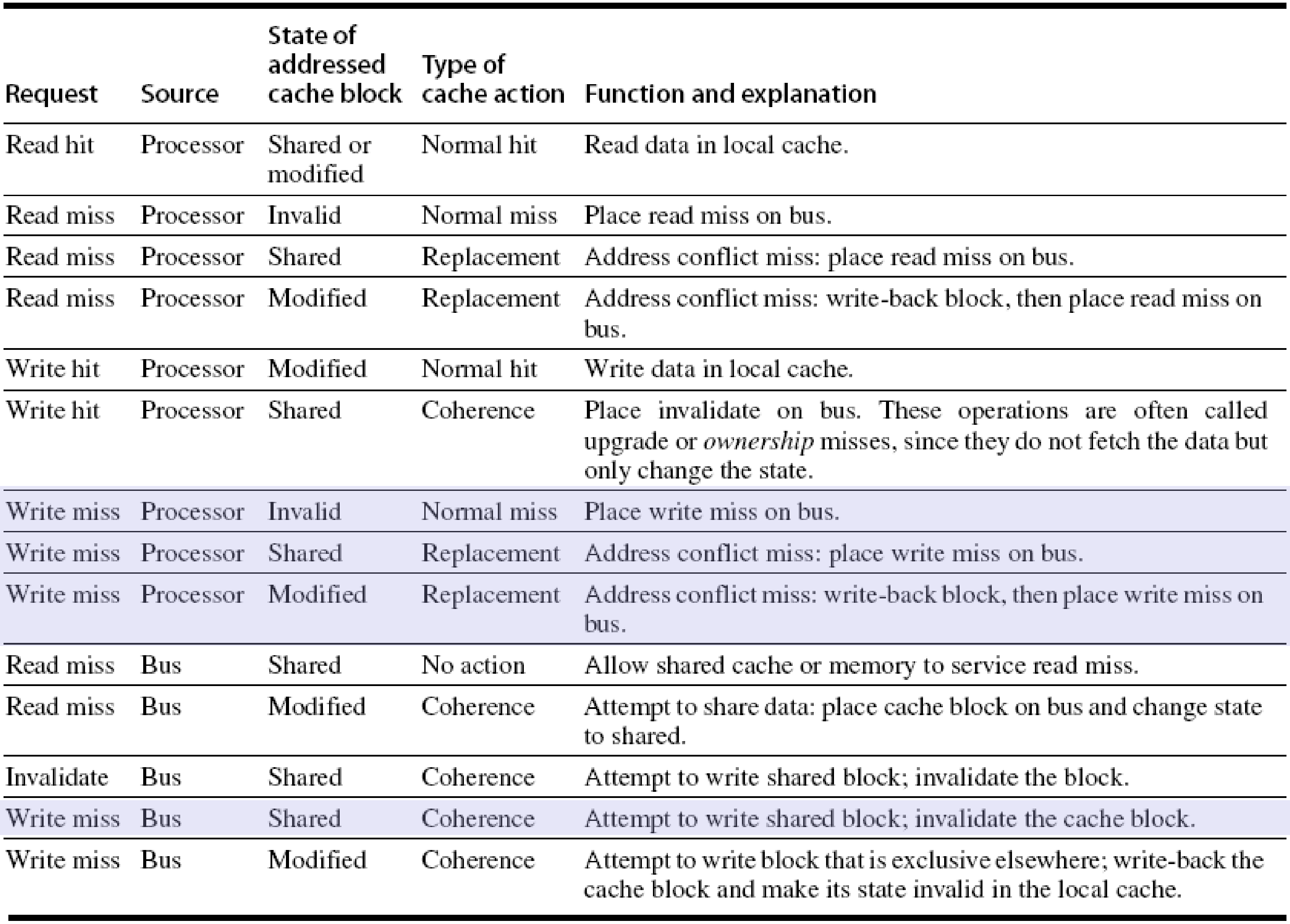
Coherence 2 (C2)

- Core C
  - Source: Processor
  - State: Invalid
  - Request: Read miss
  - Function: Place read miss on bus
- C2 (Core A)
  - Source: Bus
  - State: Modified
  - Request: Read miss
  - Function: attempt to shared data; place cache block on bus and change state to shared



Snooping coherence protocols using bus network

- The coherence mechanism of a private cache

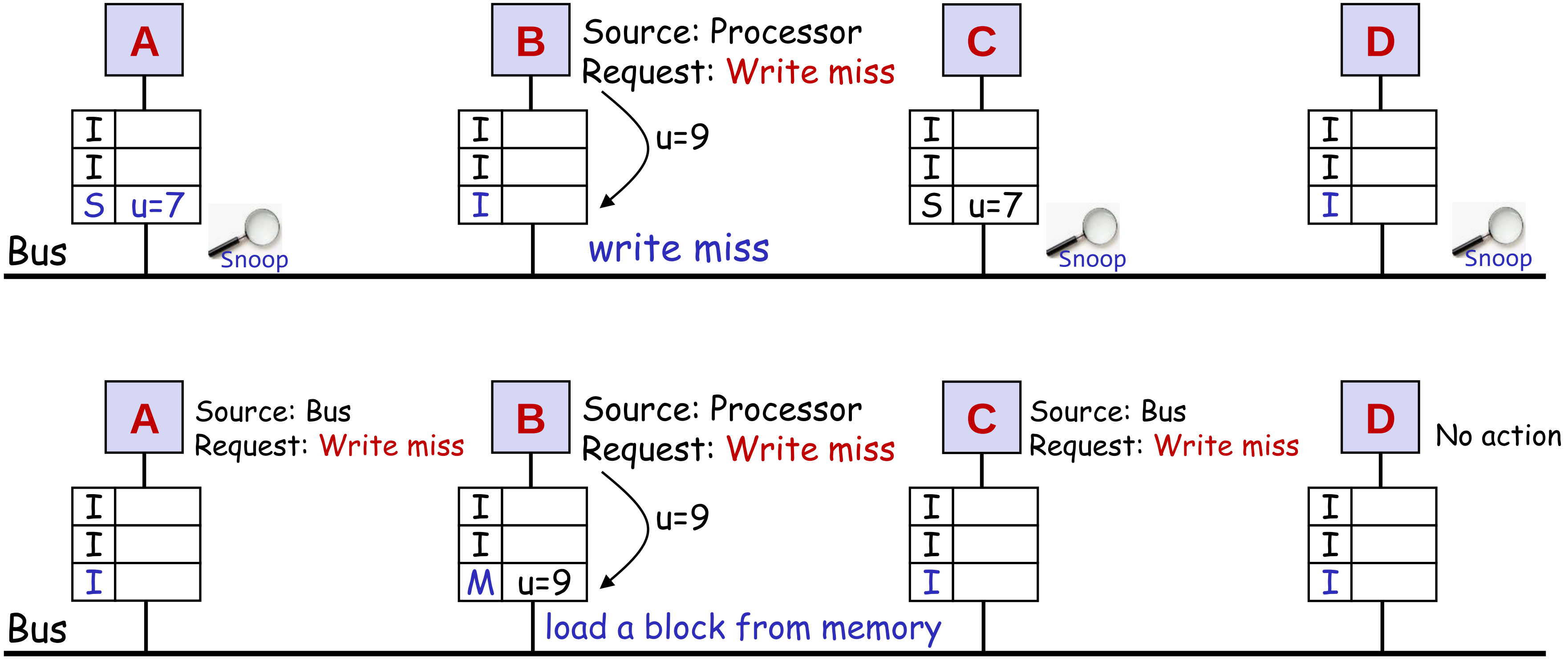


	Request	Source	State of addressed cache block	Type of cache action	Function and explanation
	Read hit	Processor	Shared or modified	Normal hit	Read data in local cache.
	Read miss	Processor	Invalid	Normal miss	Place read miss on bus.
	Read miss	Processor	Shared	Replacement	Address conflict miss: place read miss on bus.
	Read miss	Processor	Modified	Replacement	Address conflict miss: write-back block, then place read miss on bus.
	Write hit	Processor	Modified	Normal hit	Write data in local cache.
C1	Write hit	Processor	Shared	Coherence	Place invalidate on bus. These operations are often called upgrade or *ownership* misses, since they do not fetch the data but only change the state.
	Write miss	Processor	Invalid	Normal miss	Place write miss on bus.
	Write miss	Processor	Shared	Replacement	Address conflict miss: place write miss on bus.
	Write miss	Processor	Modified	Replacement	Address conflict miss: write-back block, then place write miss on bus.
	Read miss	Bus	Shared	No action	Allow shared cache or memory to service read miss.
C2	Read miss	Bus	Modified	Coherence	Attempt to share data: place cache block on bus and change state to shared.
C3	Invalidate	Bus	Shared	Coherence	Attempt to write shared block; invalidate the block.
C4	Write miss	Bus	Shared	Coherence	Attempt to write shared block; invalidate the cache block.
C5	Write miss	Bus	Modified	Coherence	Attempt to write block that is exclusive elsewhere; write-back the cache block and make its state invalid in the local cache.

Coherence 4 (C4)

- Core B
  - Source: Processor
  - State: Invalid
  - Request: Write miss
  - Function: Place write miss on bus

- C4 (Core A, C)
  - Source: Bus
  - State: Shared
  - Request: Write miss
  - Function: attempt to write shared block; invalidate the cache block



A

I	
I	
S	u=7

Bus

Snoop

B

Source: Processor
Request: Write miss

u=9

I	
I	
I	

write miss

C

I	
I	
S	u=7

Snoop

D

I	
I	
I	

Snoop

A

Source: Bus
Request: Write miss

I	
I	
I	

Bus

B

Source: Processor
Request: Write miss

u=9

I	
I	
M	u=9

load a block from memory

C

Source: Bus
Request: Write miss

I	
I	
I	

D

No action

I	
I	
I	

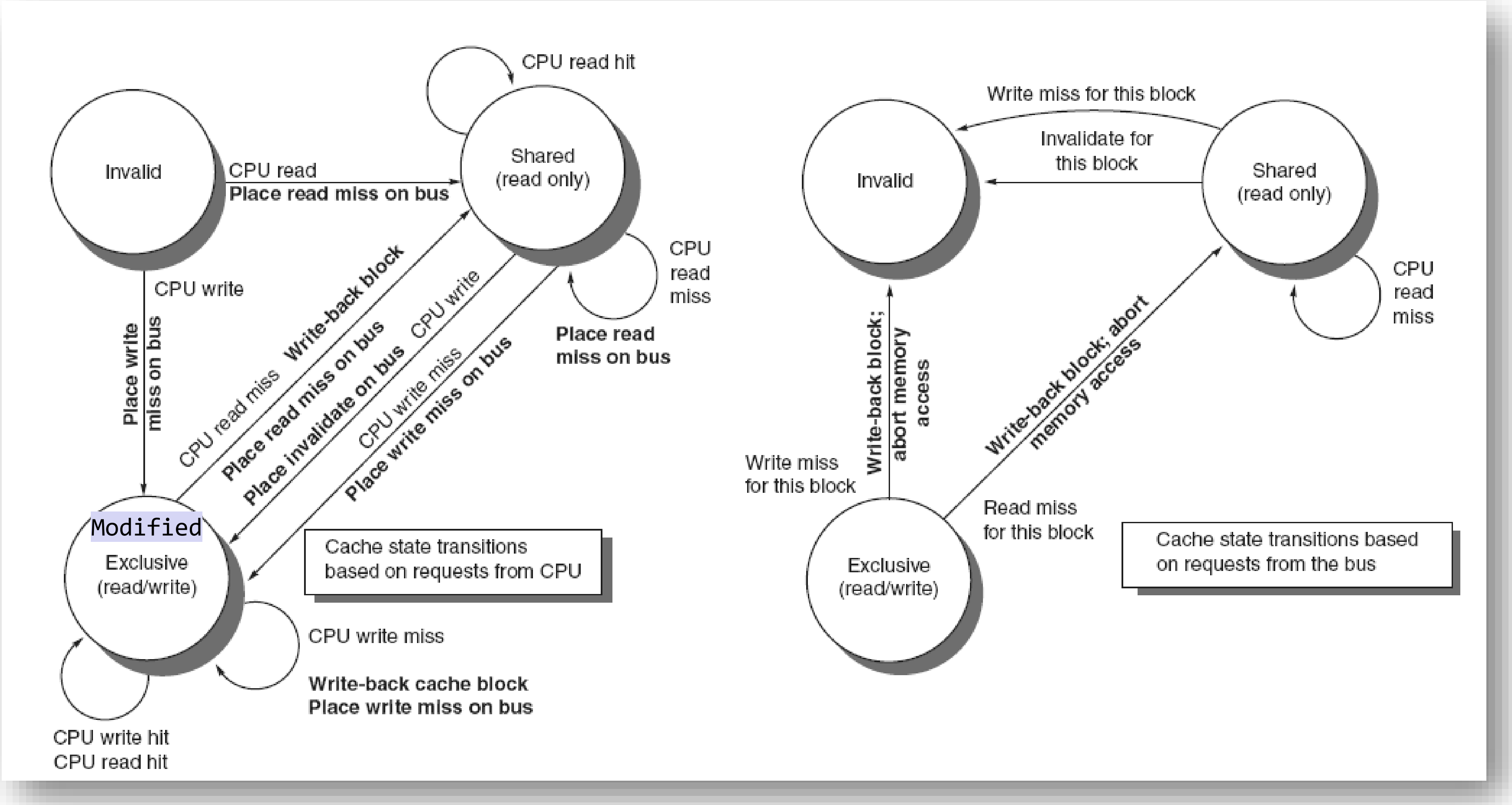
Snooping coherence protocols using bus network

- The coherence mechanism of a private cache

	Request	Source	State of addressed cache block	Type of cache action	Function and explanation
	Read hit	Processor	Shared or modified	Normal hit	Read data in local cache.
	Read miss	Processor	Invalid	Normal miss	Place read miss on bus.
	Read miss	Processor	Shared	Replacement	Address conflict miss: place read miss on bus.
	Read miss	Processor	Modified	Replacement	Address conflict miss: write-back block, then place read miss on bus.
	Write hit	Processor	Modified	Normal hit	Write data in local cache.
C1	Write hit	Processor	Shared	Coherence	Place invalidate on bus. These operations are often called upgrade or *ownership* misses, since they do not fetch the data but only change the state.
	Write miss	Processor	Invalid	Normal miss	Place write miss on bus.
	Write miss	Processor	Shared	Replacement	Address conflict miss: place write miss on bus.
	Write miss	Processor	Modified	Replacement	Address conflict miss: write-back block, then place write miss on bus.
	Read miss	Bus	Shared	No action	Allow shared cache or memory to service read miss.
C2	Read miss	Bus	Modified	Coherence	Attempt to share data: place cache block on bus and change state to shared.
C3	Invalidate	Bus	Shared	Coherence	Attempt to write shared block; invalidate the block.
C4	Write miss	Bus	Shared	Coherence	Attempt to write shared block; invalidate the cache block.
C5	Write miss	Bus	Modified	Coherence	Attempt to write block that is exclusive elsewhere; write-back the cache block and make its state invalid in the local cache.

Snooping coherence protocols using bus network

- *A write invalidate, cache coherence protocol for a private write-back cache showing the states and state transitions for each block in the cache*



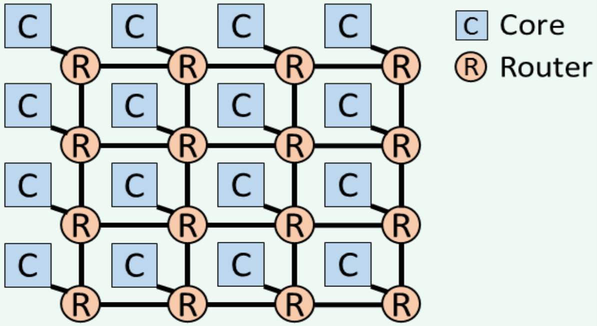
MSI (Modified, Shared, Invalid) protocol

Snooping coherence protocols using bus network

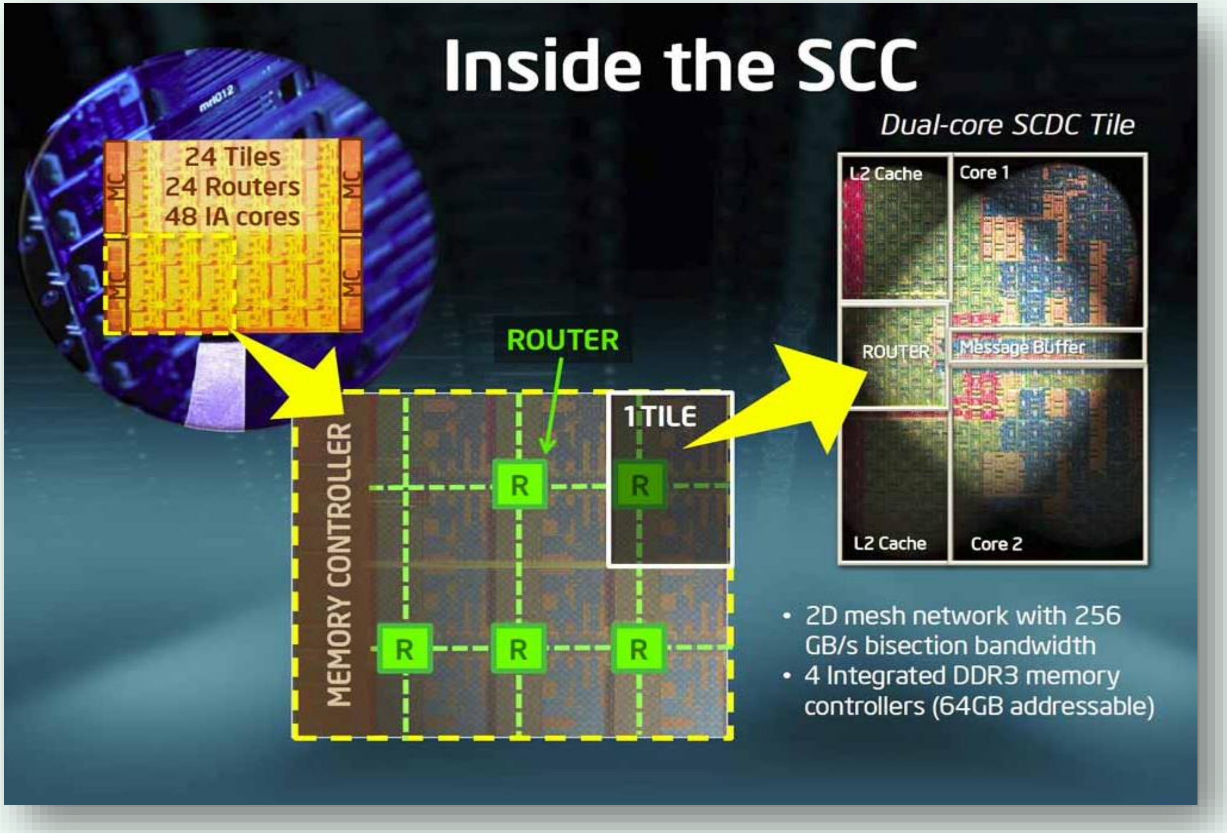
- The basic coherence protocol
  - MSI (Modified, Shared, Invalid) protocol
- Extensions
  - MESI (Modified, Exclusive, Shared, Invalid) protocol
  - MOESI (MESI + Owned) protocol

Intel Single-Chip Cloud Computer (2009)

- To research multi-core processors and parallel processing.



A many-core architecture with 2D Mesh NoC



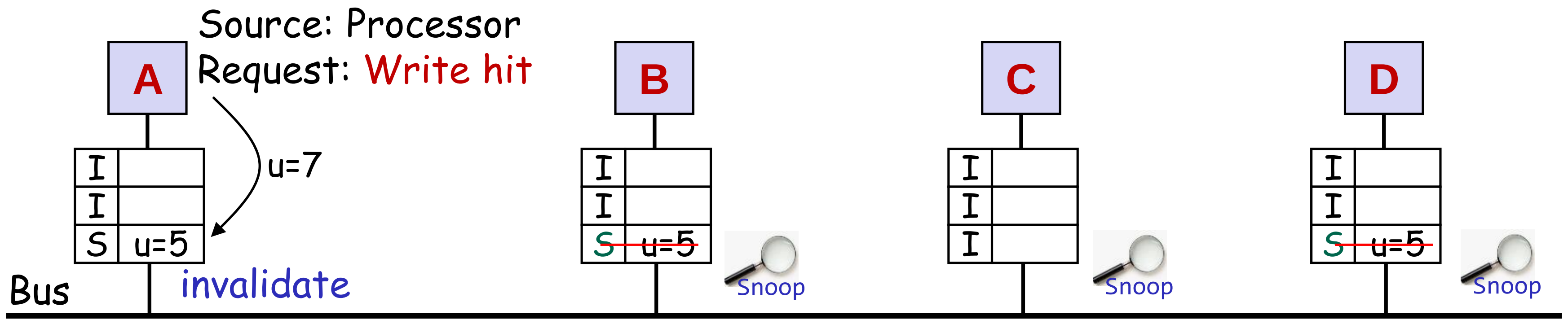
- 2D mesh network with 256 GB/s bisection bandwidth
- 4 Integrated DDR3 memory controllers (64GB addressable)

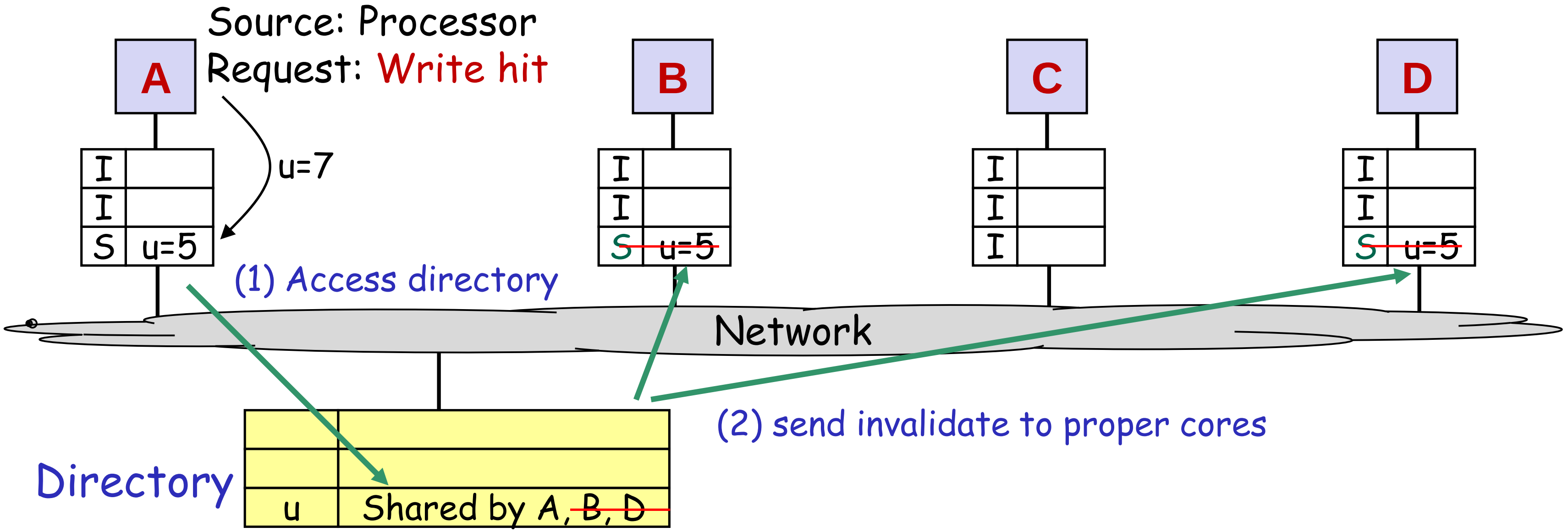
Intel Single-Chip Cloud Computer (48 Core)

Directory protocols

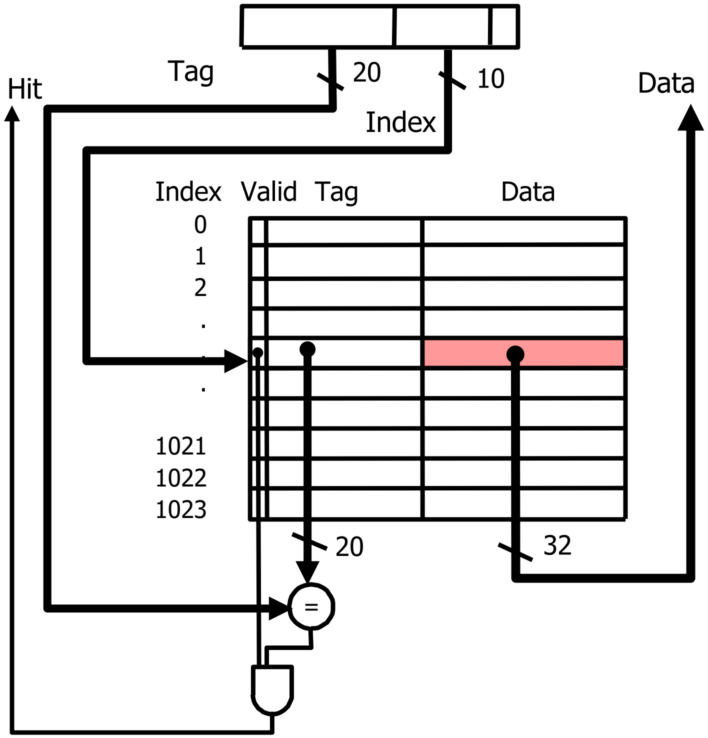
- Snooping coherence protocols are based on the use of bus network.
  What are the protocols for mesh topology NoC?
- Directory protocols
  - A logically-central directory keeps track of where the copies of each cache block reside. Caches consult this directory to ensure coherence.

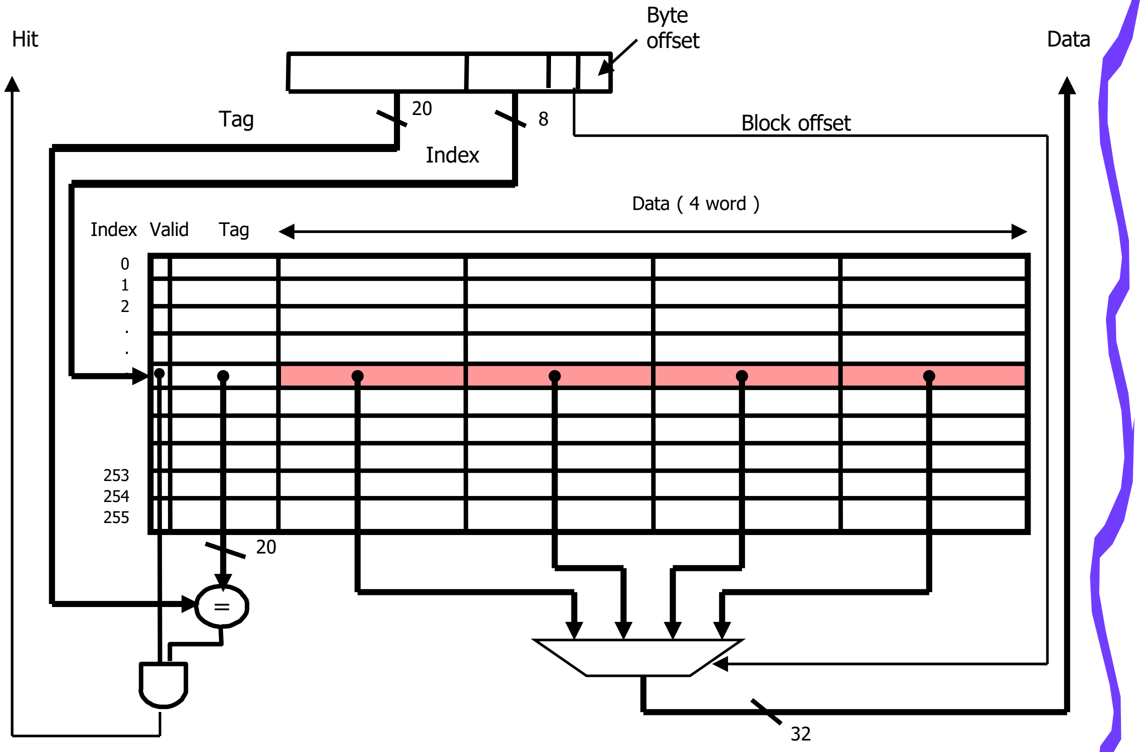
Snooping coherence protocol and one with directory





Two caches of different block sizes



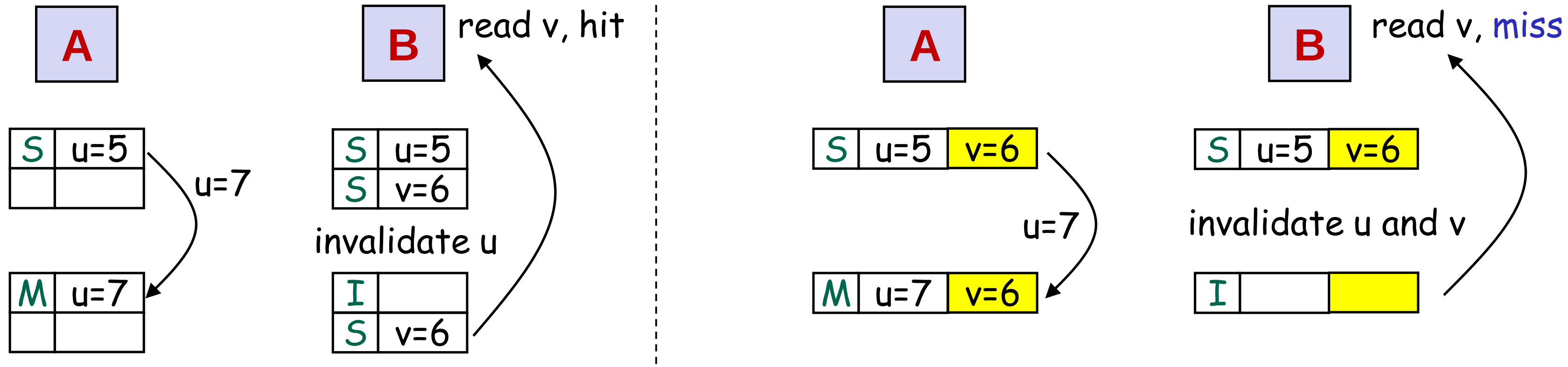


One word/block

Four words/block

Coherence influences the cache miss rate

- Coherence misses
  - True sharing misses
    - Write to shared block (transmission of invalidation)
    - Read
  - False sharing misses

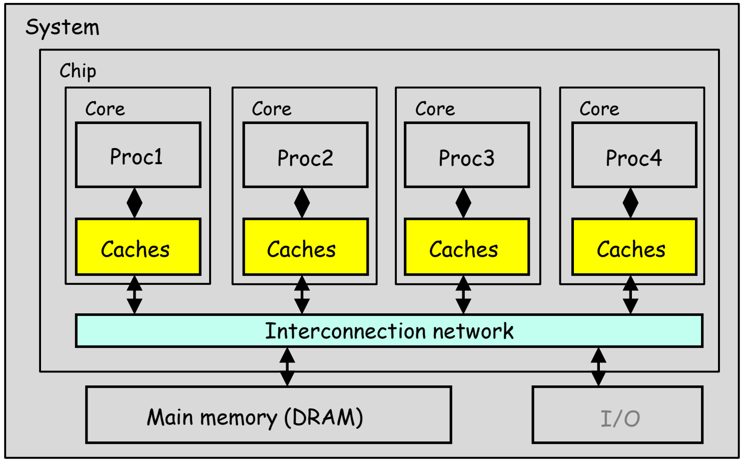


cache block of one word

cache block of two words

Key components of many-core processors

- Interconnection network
  - connecting many modules on a chip achieving high throughput and low latency
- Main memory and caches
  - Caches are used to reduce latency and to lower network traffic
  - A parallel program has private data and shared data
  - New issues are cache coherence and memory consistency
- Core
  - High-performance superscalar processor providing a hardware mechanism to support thread synchronization



this slide is to be used as a whiteboard