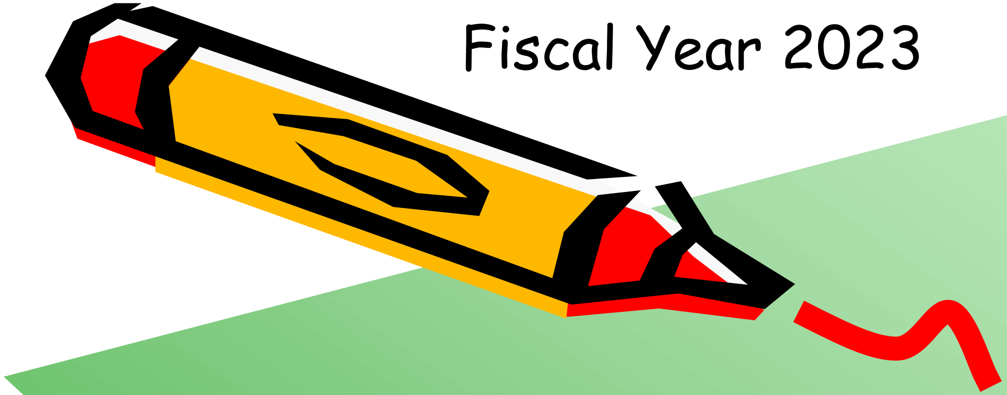
Fiscal Year 2023

Ver. 2024-01-26a



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

Advanced Computer Architecture

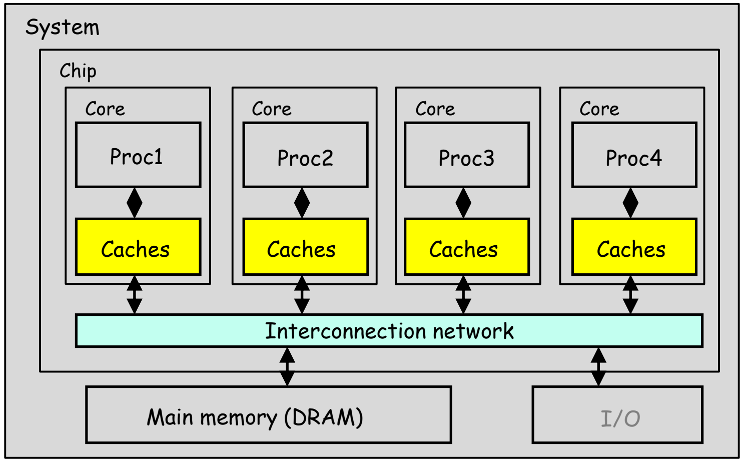
12. Thread Level Parallelism: Coherence and Synchronization

www.arch.cs.titech.ac.jp/lecture/ACA/
Room No.W834, Lecture (Face-to-face)
Mon 13:30-15:10, Thr 13:30-15:10

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

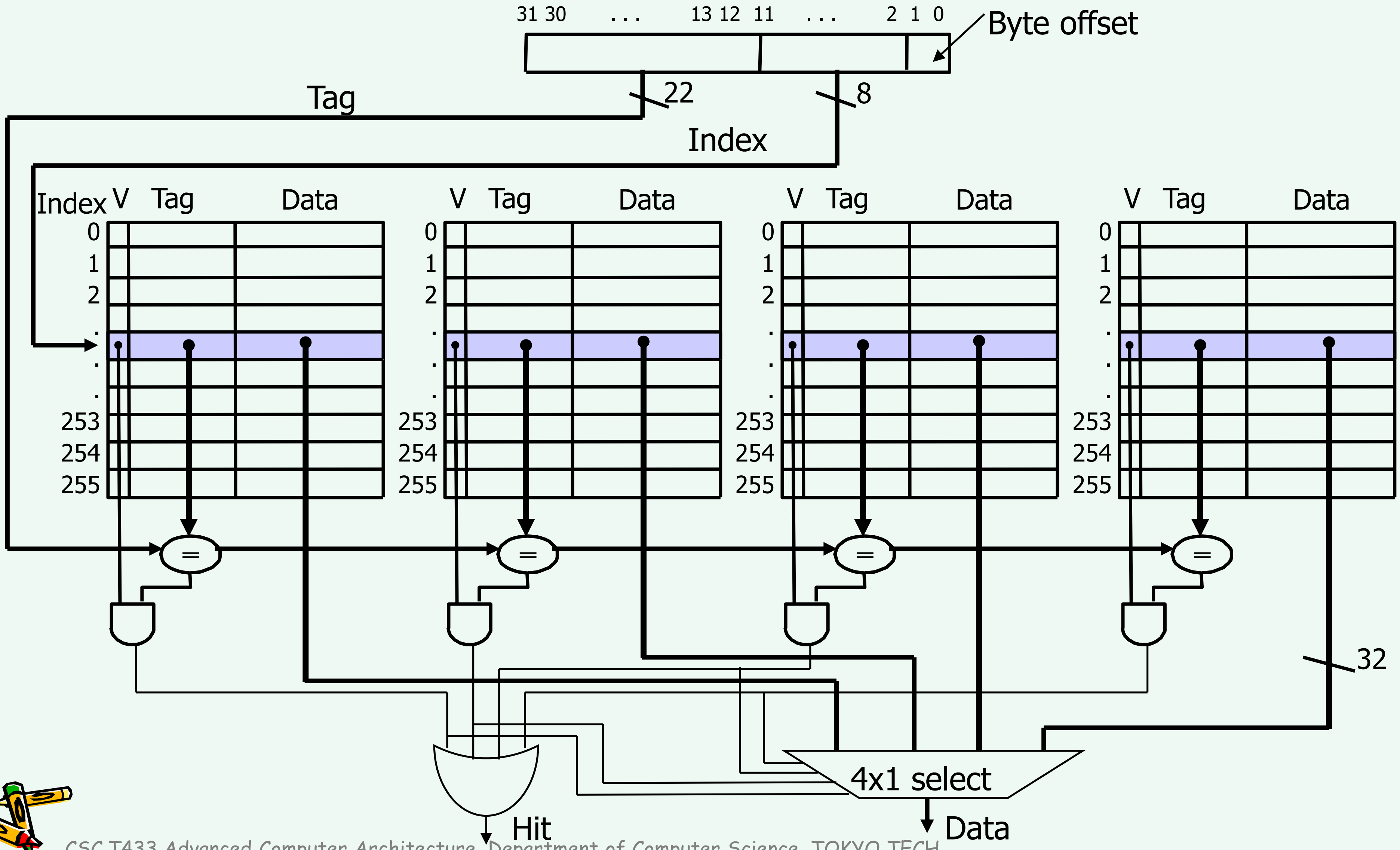
Key components of many-core processors

- 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



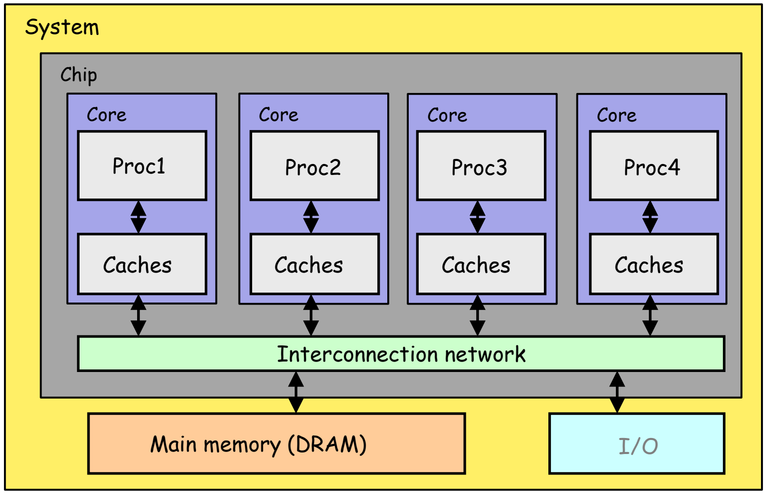
Four-Way Set Associative Cache

- One word/block, 2^8 = **256 sets** where each with four ways (each with one block)



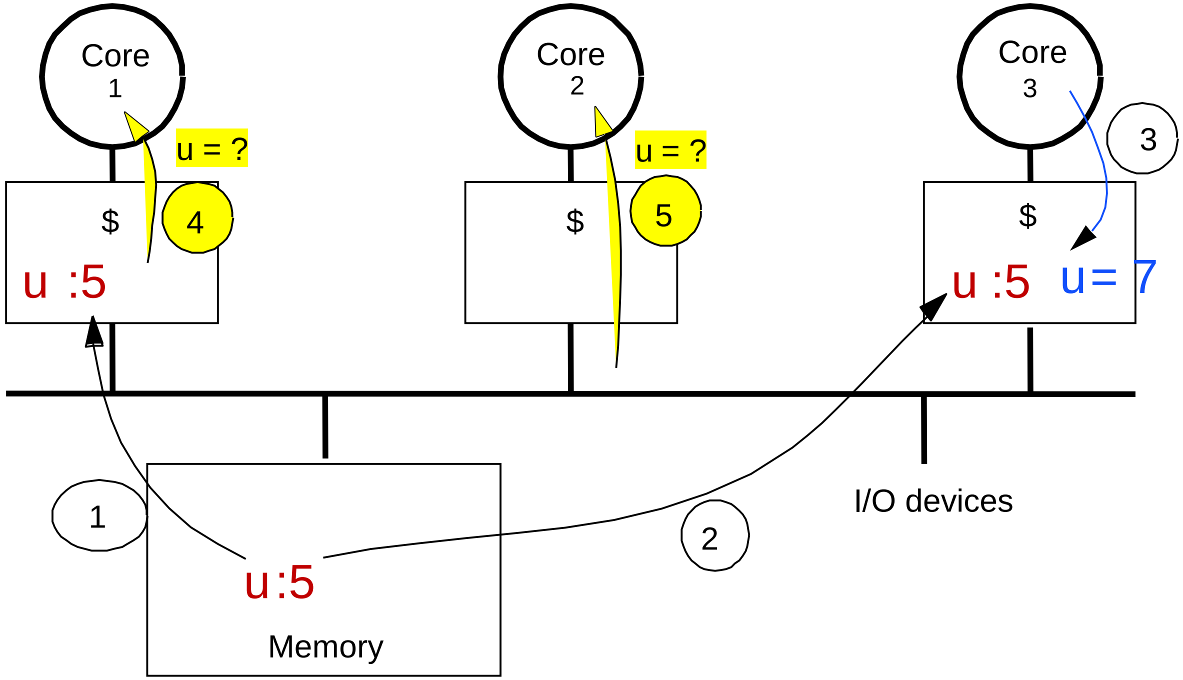
Cache writing policy

- Write-through
  - writing is done synchronously both to the cache and to the main memory. All stores update the main memory and memory bandwidth becomes a performance bottleneck.
- 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.
  - preferable for manycore.



Cache coherence problem

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  - Processes accessing main memory may see stale (out-of-date) value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



Cache coherence problem

- Cores may see different values through their caches
  - assuming a write-back cache
  - after the value 0 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 core A	Cache contents for core B	Memory contents for location X
0				1
1	Core A reads X	1		1
2	Core B reads X	1	1	1
3	Core A stores 0 into X	0	**1**	**1**

Cache coherence and enforcing coherence

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

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

Snooping coherence protocols using bus network

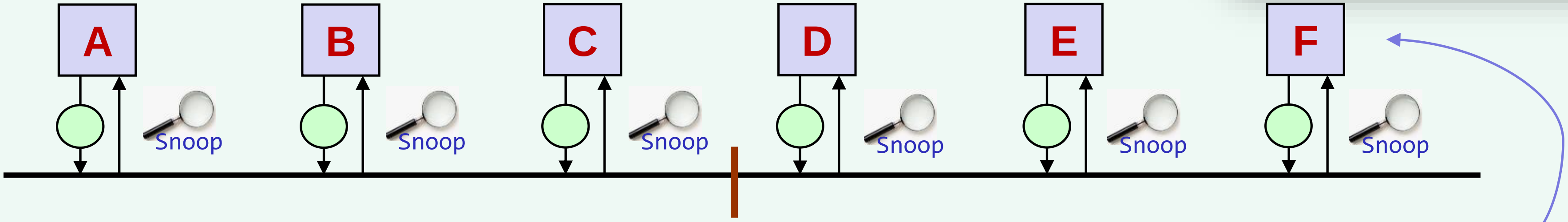
- Write invalidate
  - On write, invalidate all other copies by an invalidate broadcast
  - Use bus itself to serialize
    - Write cannot complete until bus access is obtained

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

- Write update
  - On write, update all copies

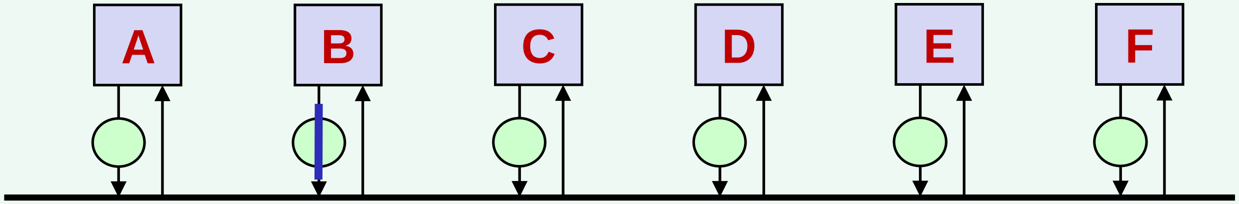
Bus Network

- 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



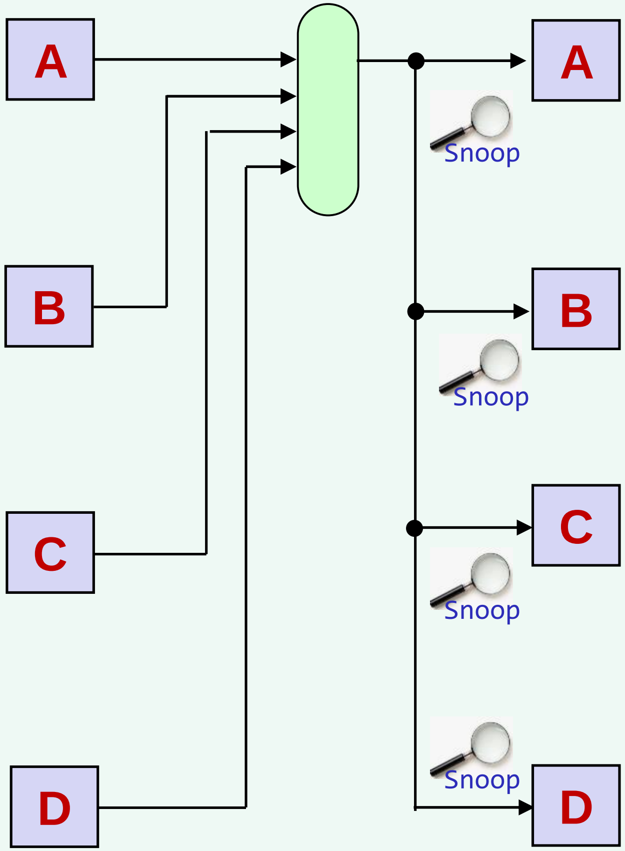
Core or processor node

The case where core B sends a packet to someone



Bus Network with multiplexer (mux)

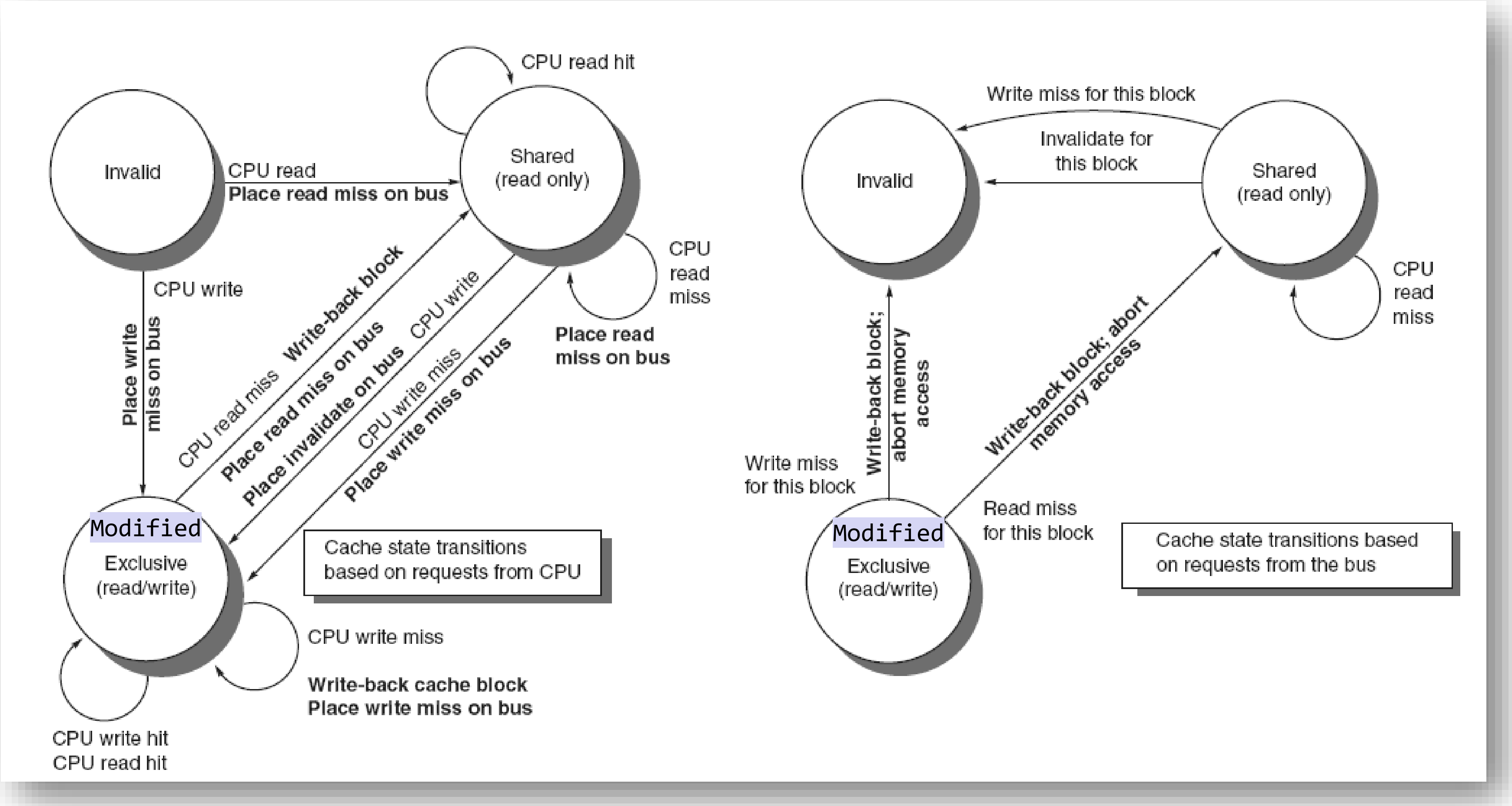
- one N-input multiplexer for N cores



The bus network organization of 4 cores using a 4-input mux.

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

Orchestration

- LOCK and UNLOCK around critical section
  - Lock provides exclusive access to the locked data.
  - Set of operations we want to execute atomically
- BARRIER ensures all reach here

```
float A[N+2], B[N+2]; /* these are in shared memory */
float diff=0.0;       /* variable  in shared memory */
int ncores = 2;
pthread_mutex_t m = PTHREAD_MUTEX_INITIALIZER;
pthread_barrier_t barrier;
void solve_pp (int pid) {
    int i, done = 0;                    /* private variables */
    int mymin = 1 + (pid * N/ncores);   /* private variable  */
    int mymax = mymin + N/ncores – 1;   /* private variable  */
    while (!done) {
        float mydiff = 0;
        for (i=mymin; i<=mymax; i++) {
            B[i] = 0.333 * (A[i-1] + A[i] + A[i+1]);
            mydiff = mydiff + fabsf(B[i] - A[i]);
        }
        pthread_mutex_lock(&m);
        diff = diff + mydiff;
        pthread_mutex_unlock(&m);

        pthread_barrier_wait(&barrier);
        if (diff <TOL) done = 1;
        pthread_barrier_wait(&barrier);
        if (pid==1) diff = 0.0;
        for (i=mymin; i<=mymax; i++) A[i] = B[i];
        pthread_barrier_wait(&barrier);
    }
}
```

These operations must be executed atomically

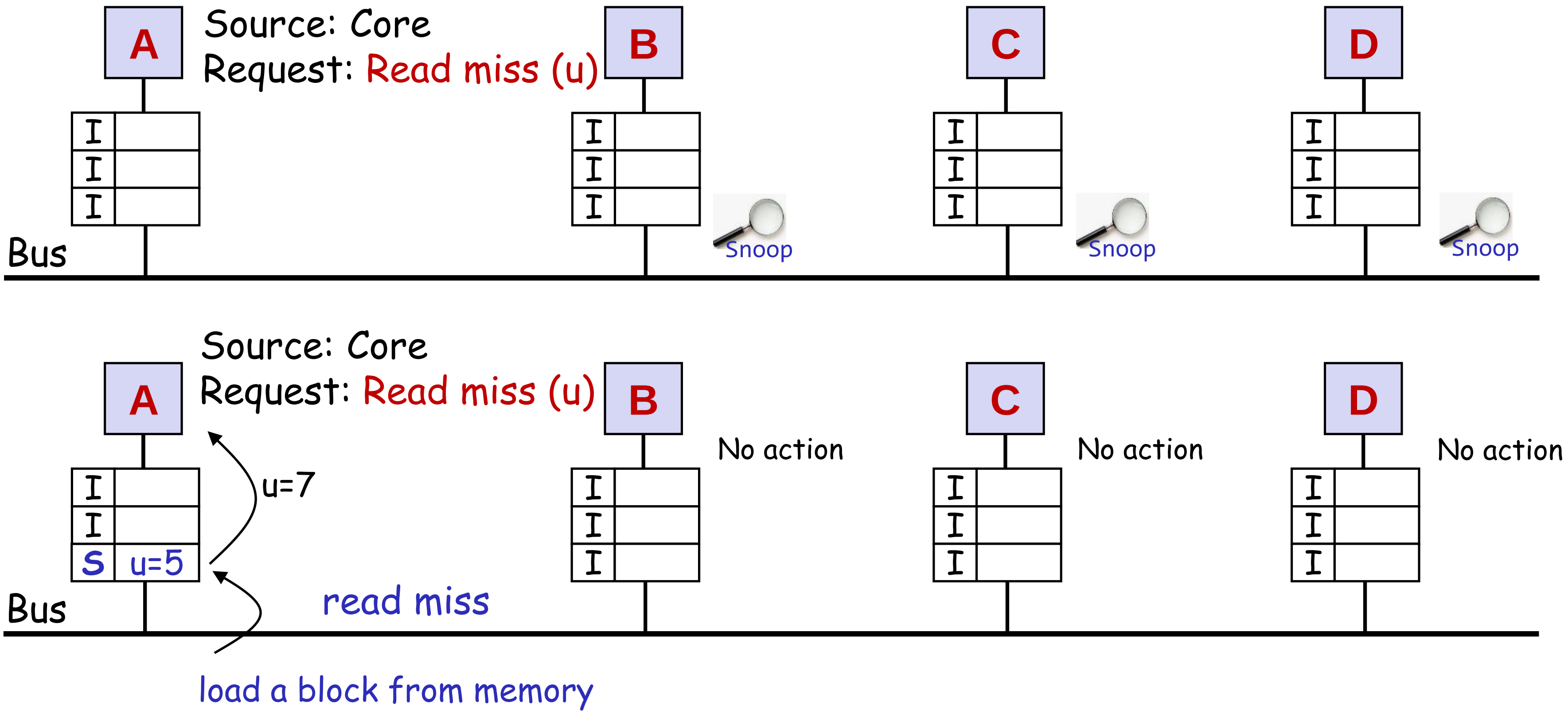
```
(1) load diff
(2) add
(3) store diff
```

After all cores update the diff, *if statement* must be executed.

```
if (diff <TOL) done = 1;
```

Cache miss and the addressed block is invalid

- Core A
  - Source: Core
  - State: Invalid
  - Request: Read miss (u)
  - Function: Place read miss on bus



Source: Core
Request: Read miss (u)

Source: Core
Request: Read miss (u)

No action

No action

No action

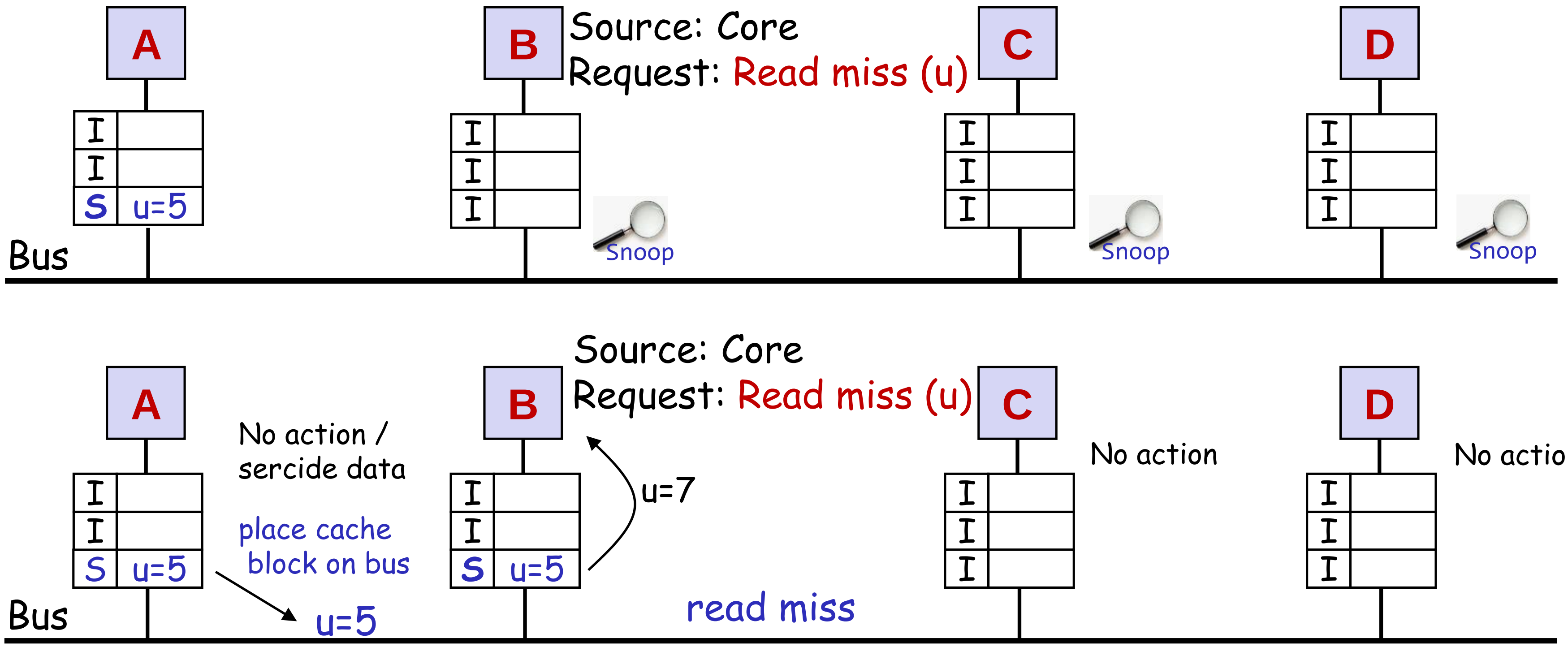
u=7

read miss

load a block from memory

Cache miss and the addressed block is invalid

- Core B
  - Source: Core
  - State: Invalid
  - Request: Read miss (u)
  - Function: Place read miss on bus



Source: Core
Request: Read miss (u)

Source: Core
Request: Read miss (u)

No action / sercide data

place cache block on bus

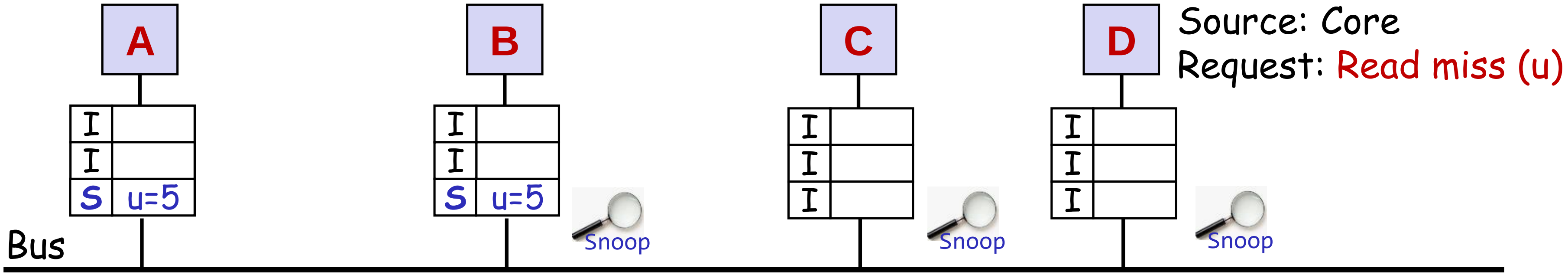
No action

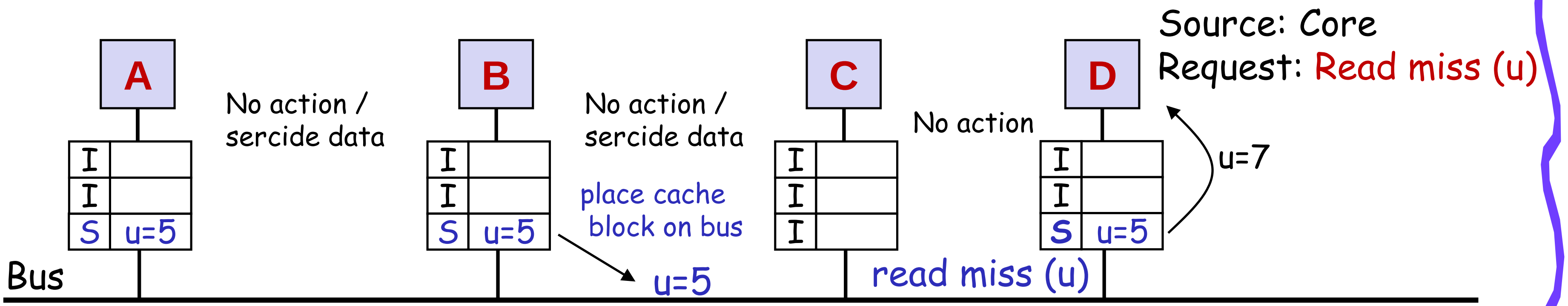
No action

load a block from memory or allow shared cache to service data

Cache miss and the addressed block is invalid

- Core D
  - Source: Core
  - State: Invalid
  - Request: Read miss (u)
  - Function: Place read miss on bus


Source: Core
Request: Read miss (u)


Source: Core
Request: Read miss (u)

No action / sercide data

No action / sercide data

place cache block on bus

No action

u=7

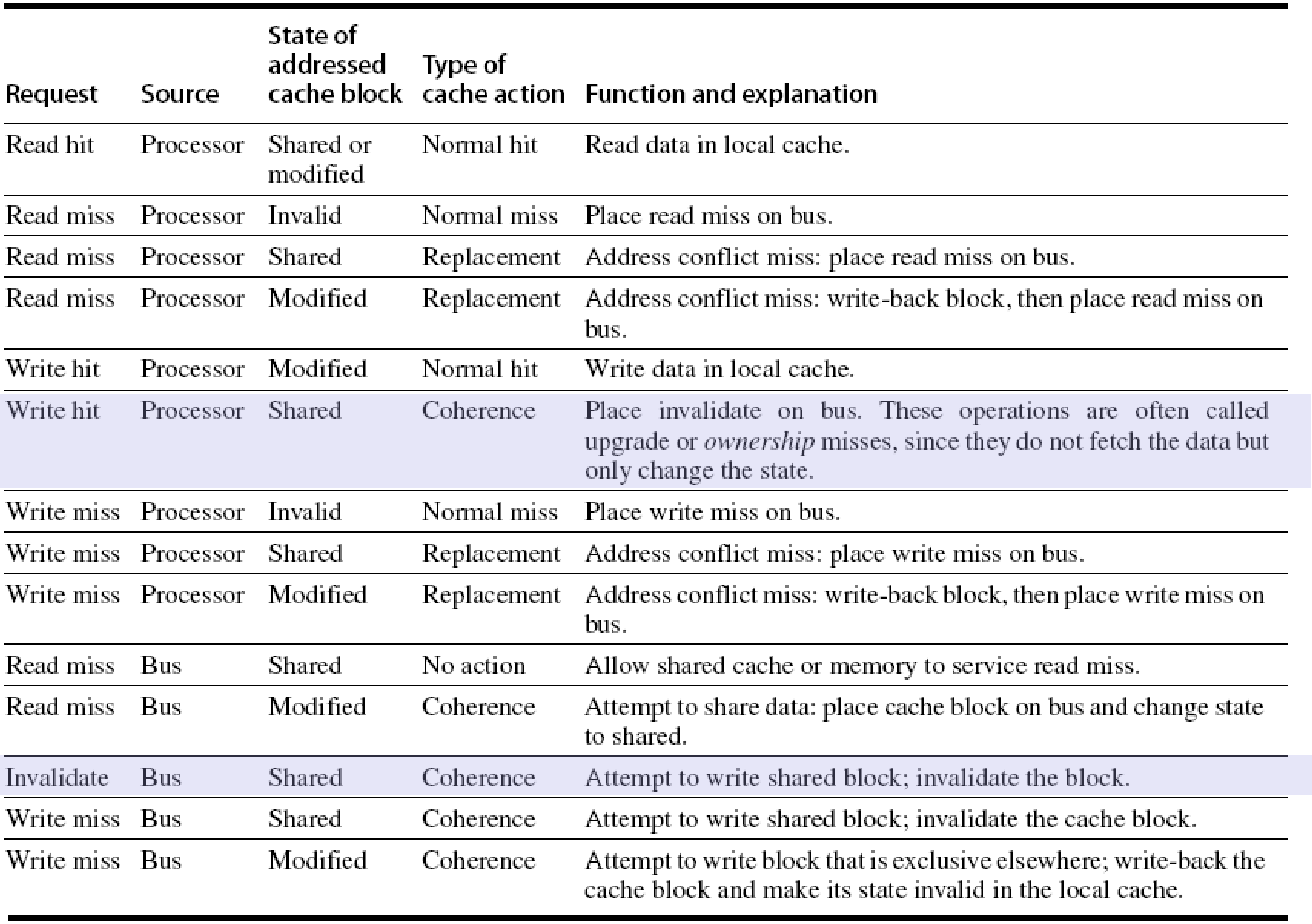
u=5

read miss (u)

load a block from memory or allow shared cache to service data

Snooping coherence protocols using bus network

- The coherence mechanism of a private cache (using word processor for core).

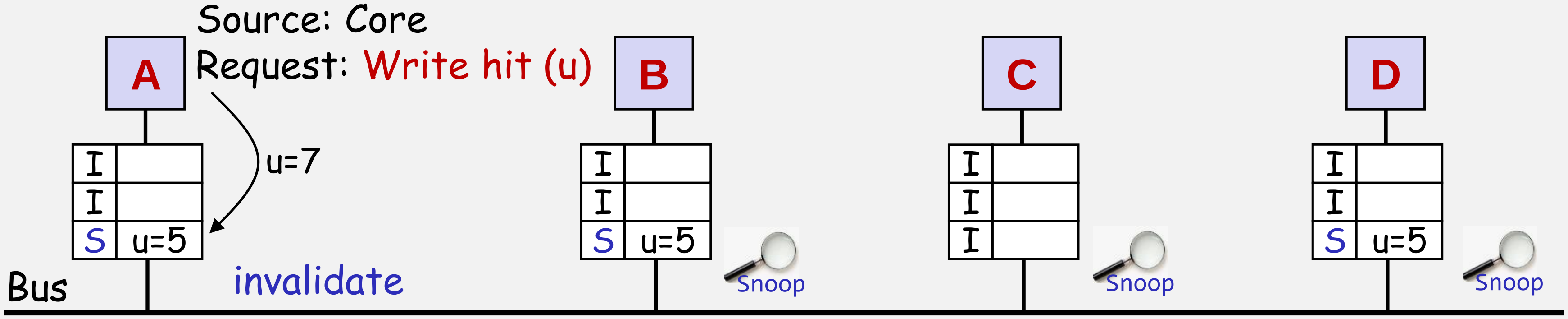


	Request	Source	State of addressed cache block	Type of cache action	Function and explanation
	Read hit	Processor	Shared or modified	Normal hit	Read data in local cache.
	Read miss	Processor	Invalid	Normal miss	Place read miss on bus.
	Read miss	Processor	Shared	Replacement	Address conflict miss: place 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.
Coh1	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.
Coh2	Read miss	Bus	Modified	Coherence	Attempt to share data: place cache block on bus and change state to shared.
Coh3	Invalidate	Bus	Shared	Coherence	Attempt to write shared block; invalidate the block.
Coh4	Write miss	Bus	Shared	Coherence	Attempt to write shared block; invalidate the cache block.
Coh5	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.

Exercise 1

- Coh1 (Core A)
  - Source: Core
  - State: Shared
  - Request: Write hit (u)
  - Function: Place invalidate on bus

- Coh3 (Core B, D)
  - Source: Bus
  - State: Shared
  - Request: Invalidate
  - Function: attempt to write shared block; invalidate the block



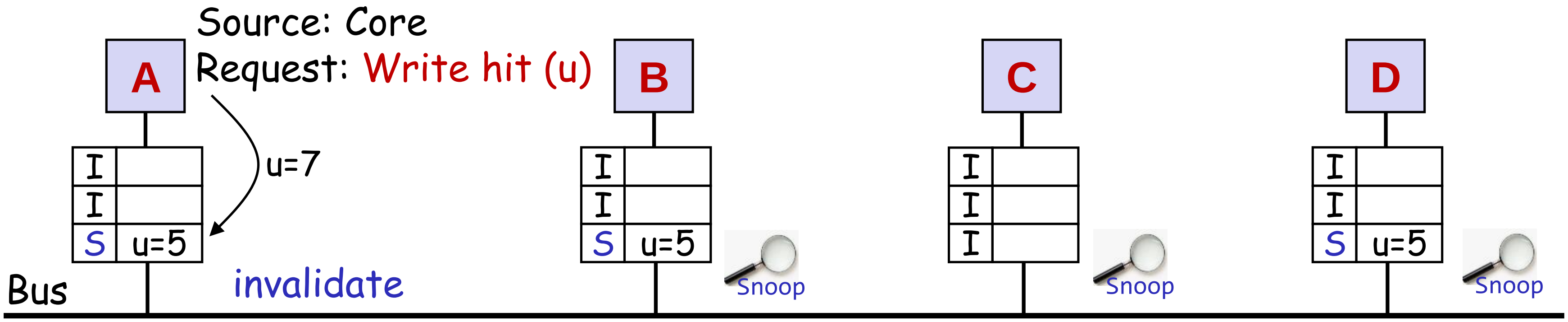
Source: Core
Request: Write hit (u)

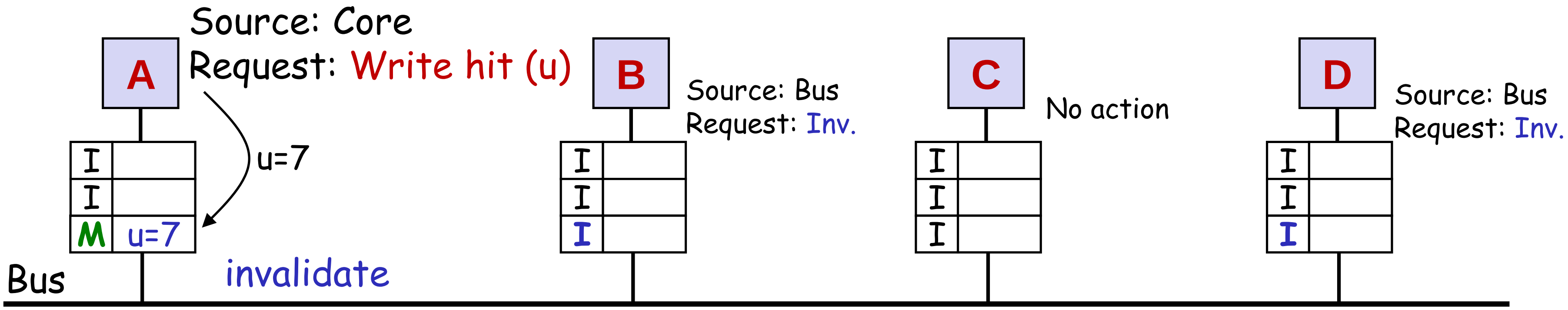
Draw the behavior of this request

Coherence 1 (Coh1) and Coherence3 (Coh3)

- Coh1 (Core A)
  - Source: Core
  - State: Shared
  - Request: Write hit (u)
  - Function: Place invalidate on bus

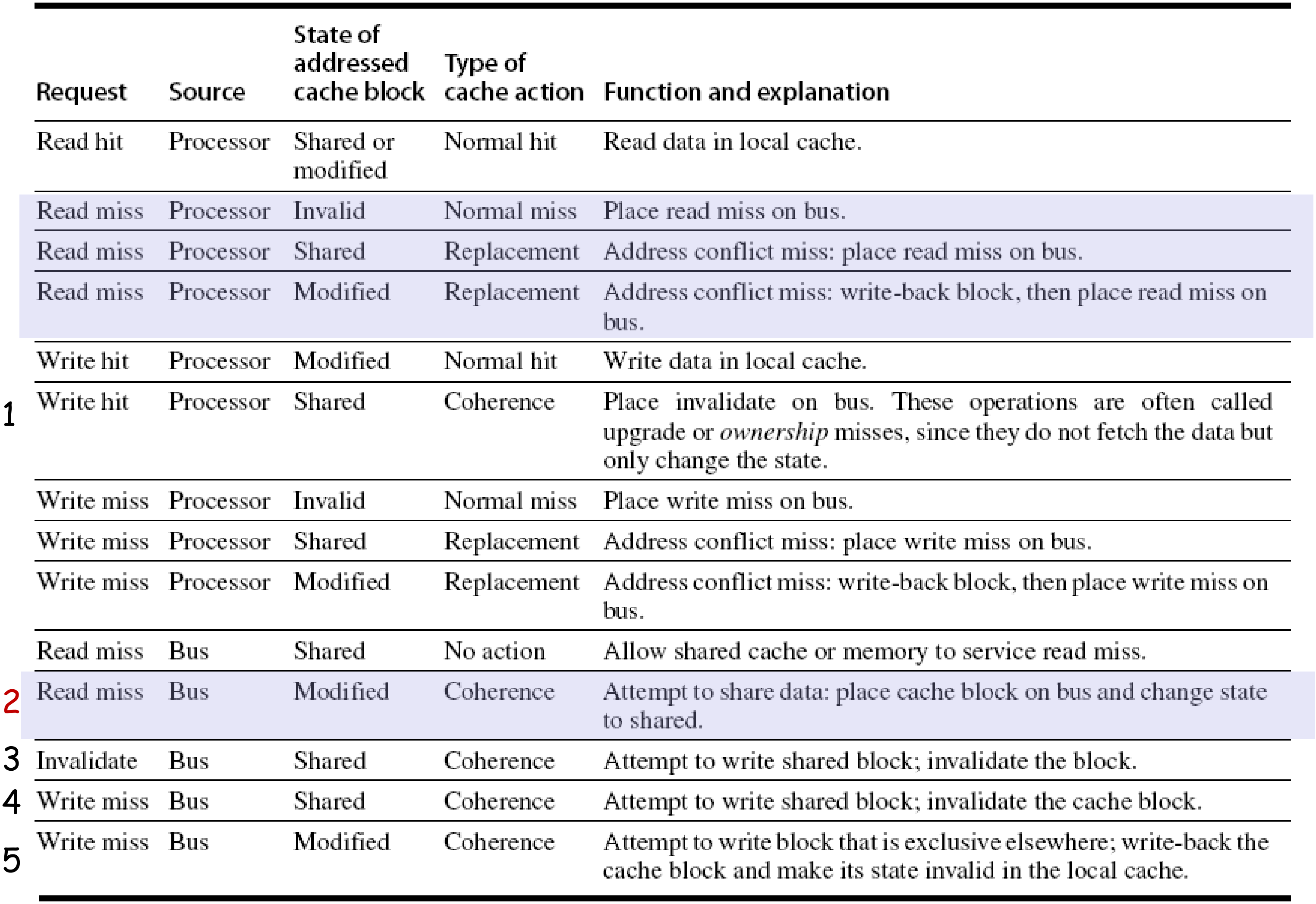
- Coh3 (Core B, D)
  - Source: Bus
  - State: Shared
  - Request: Invalidate
  - Function: attempt to write shared block; invalidate the block





Snooping coherence protocols using bus network

- The coherence mechanism of a private cache (using word processor for core).

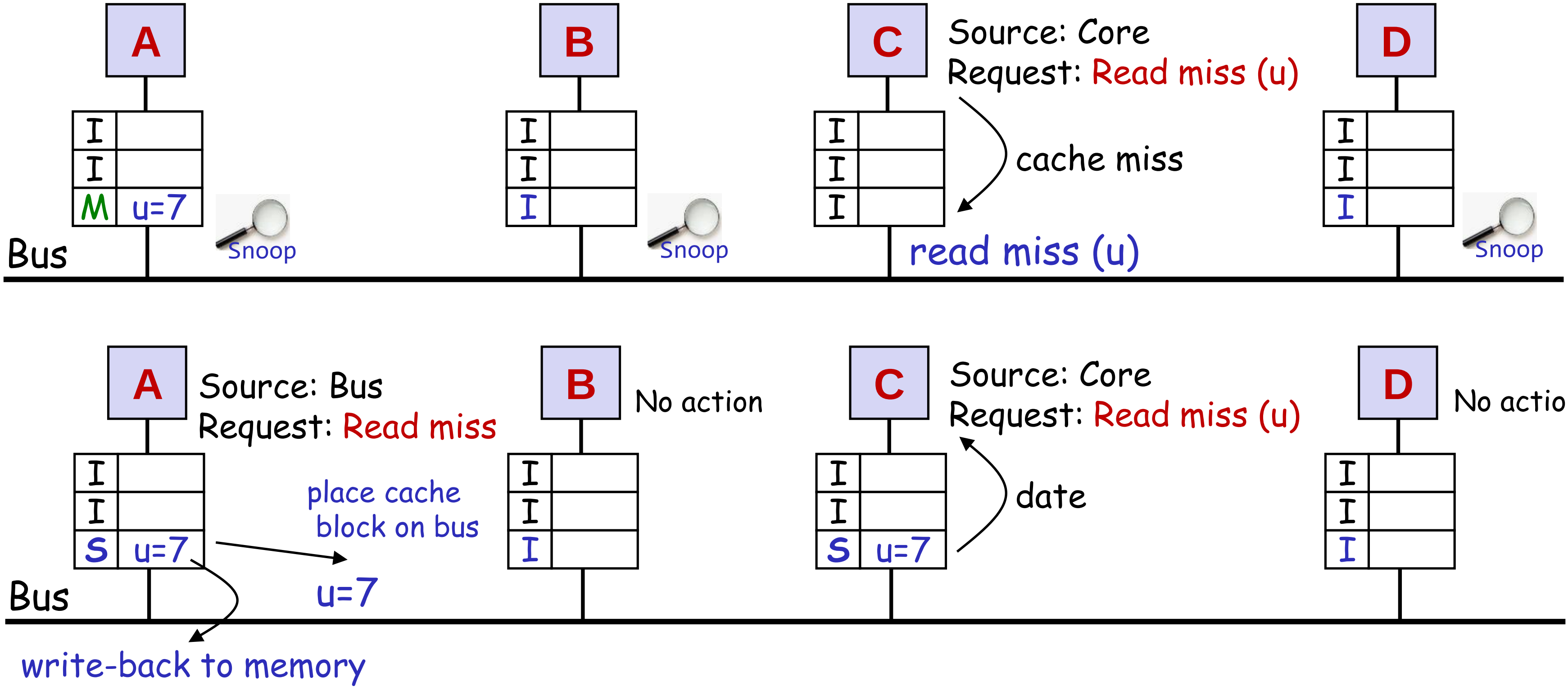


	Request	Source	State of addressed cache block	Type of cache action	Function and explanation
	Read hit	Processor	Shared or modified	Normal hit	Read data in local cache.
	Read miss	Processor	Invalid	Normal miss	Place read miss on bus.
	Read miss	Processor	Shared	Replacement	Address conflict miss: place 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.
Coh1	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.
Coh2	Read miss	Bus	Modified	Coherence	Attempt to share data: place cache block on bus and change state to shared.
Coh3	Invalidate	Bus	Shared	Coherence	Attempt to write shared block; invalidate the block.
Coh4	Write miss	Bus	Shared	Coherence	Attempt to write shared block; invalidate the cache block.
Coh5	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 (Coh2)

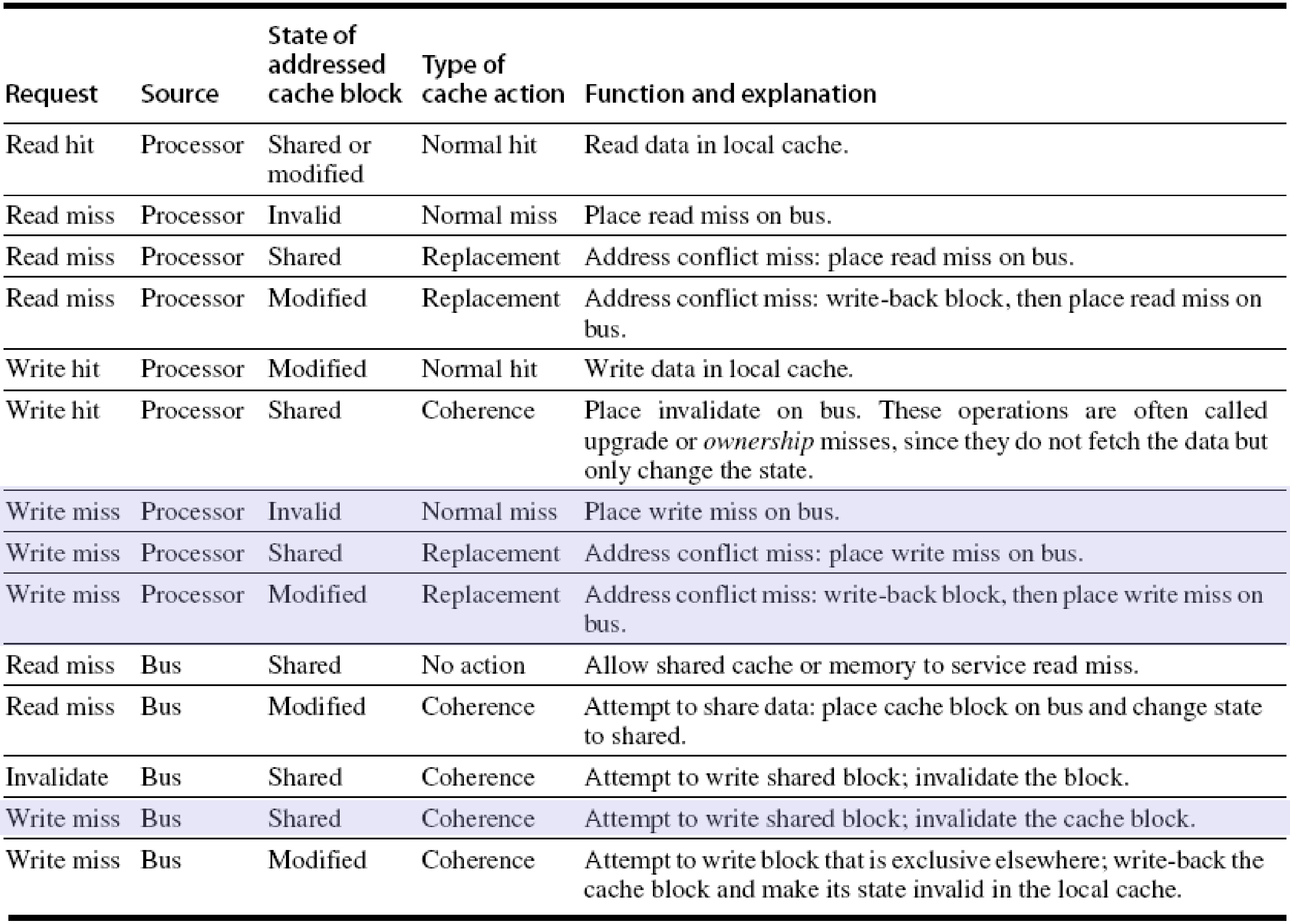
- Core C
  - Source: Core
  - State: Invalid
  - Request: Read miss (u)
  - Function: Place read miss on bus

- Coh2 (Core A)
  - Source: Bus
  - State: Modified
  - Request: Read miss (u)
  - 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

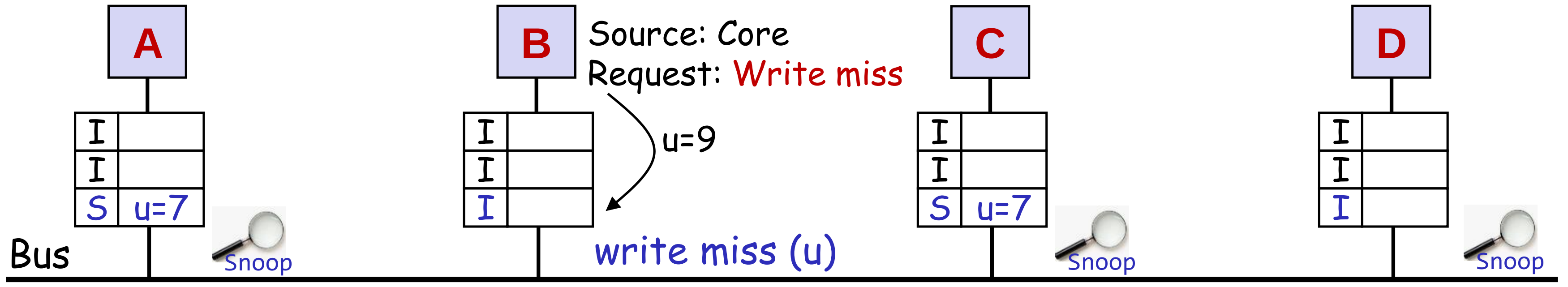


	Request	Source	State of addressed cache block	Type of cache action	Function and explanation
	Read hit	Processor	Shared or modified	Normal hit	Read data in local cache.
	Read miss	Processor	Invalid	Normal miss	Place read miss on bus.
	Read miss	Processor	Shared	Replacement	Address conflict miss: place 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.
Coh1	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.
Coh2	Read miss	Bus	Modified	Coherence	Attempt to share data: place cache block on bus and change state to shared.
Coh3	Invalidate	Bus	Shared	Coherence	Attempt to write shared block; invalidate the block.
Coh4	Write miss	Bus	Shared	Coherence	Attempt to write shared block; invalidate the cache block.
Coh5	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 (Coh4)

- Core B
  - Source: Core
  - State: Invalid
  - Request: Write miss (u)
  - Function: Place write miss on bus

- Coh4 (Core A, C)
  - Source: Bus
  - State: Shared
  - Request: Write miss (u)
  - Function: attempt to write shared block; invalidate the cache block



A

I	
I	
S	u=7

Snoop

B

Source: Core
Request: Write miss

u=9

I	
I	
I	

write miss (u)

C

I	
I	
S	u=7

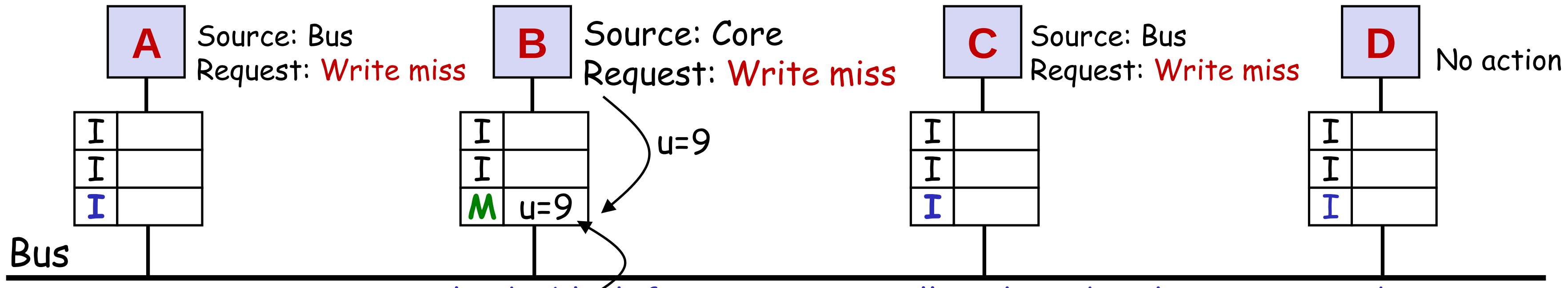
Snoop

D

I	
I	
I	

Snoop

Bus



A

Source: Bus
Request: Write miss

I	
I	
I	

B

Source: Core
Request: Write miss

u=9

I	
I	
M	u=9

C

Source: Bus
Request: Write miss

I	
I	
I	

D

No action

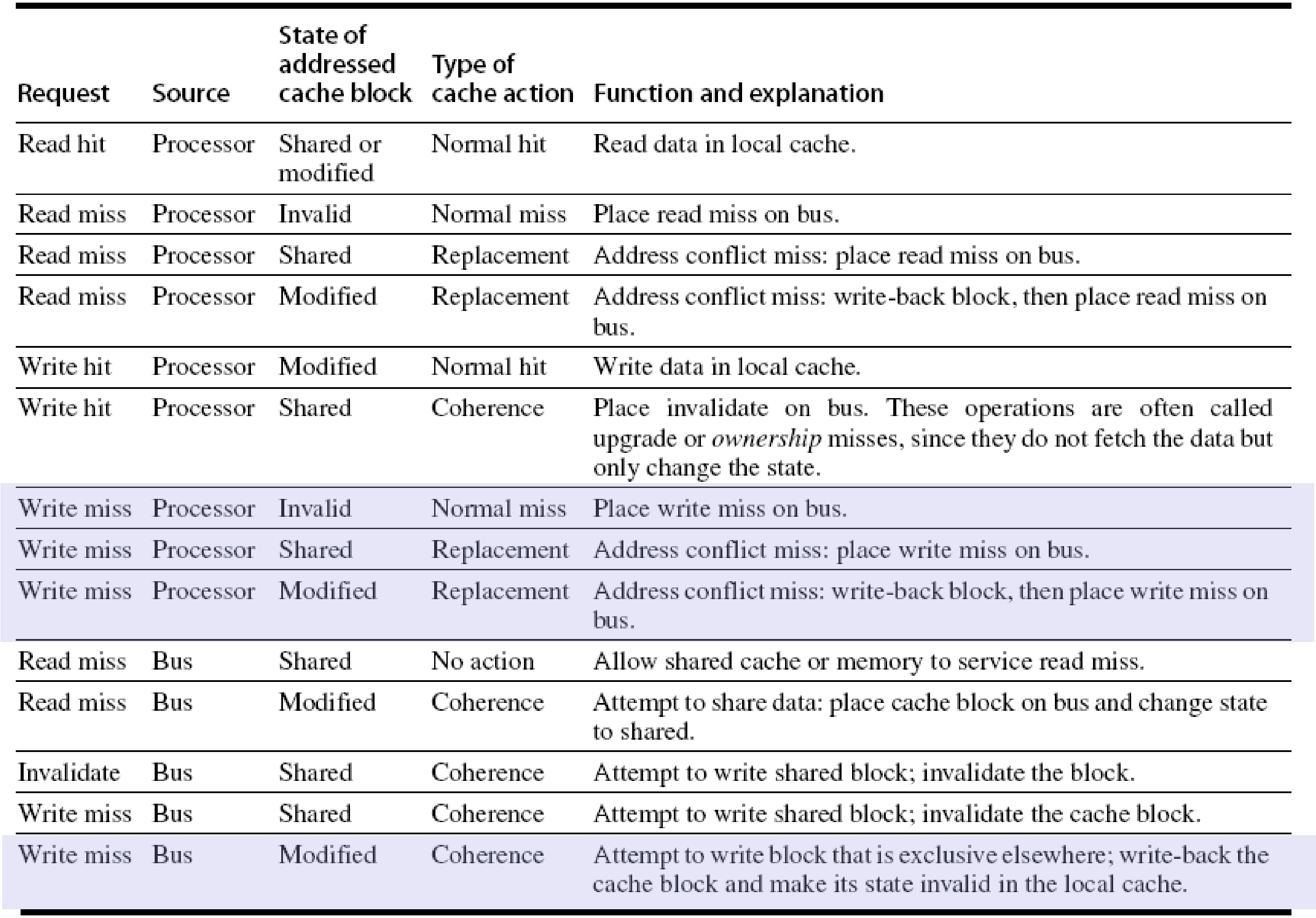
I	
I	
I	

Bus

load a block from memory or allow shared cache to service data

Snooping coherence protocols using bus network

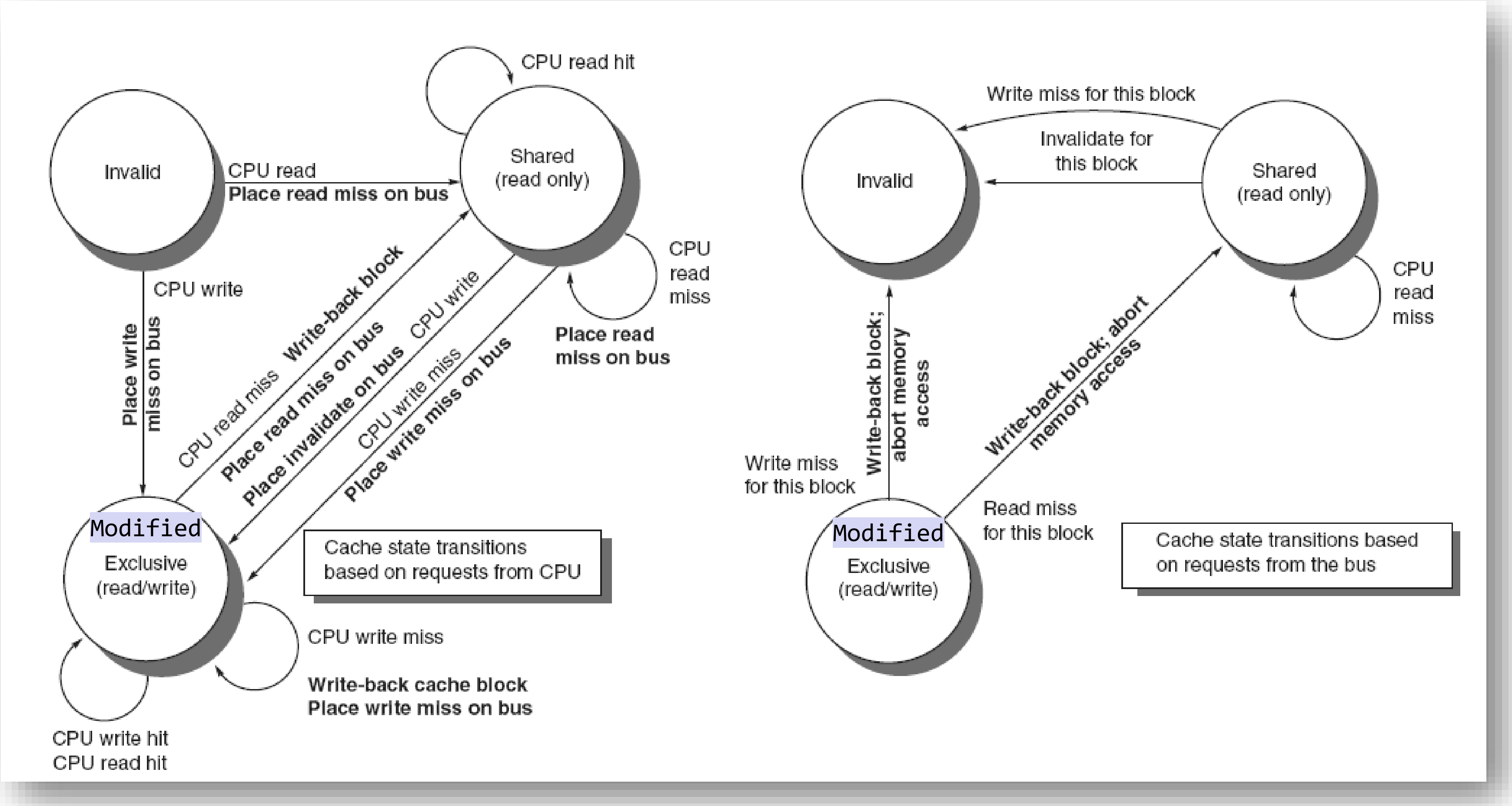
- The coherence mechanism of a private cache



	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.
Coh1	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.
Coh2	Read miss	Bus	Modified	Coherence	Attempt to share data: place cache block on bus and change state to shared.
Coh3	Invalidate	Bus	Shared	Coherence	Attempt to write shared block; invalidate the block.
Coh4	Write miss	Bus	Shared	Coherence	Attempt to write shared block; invalidate the cache block.
Coh5	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



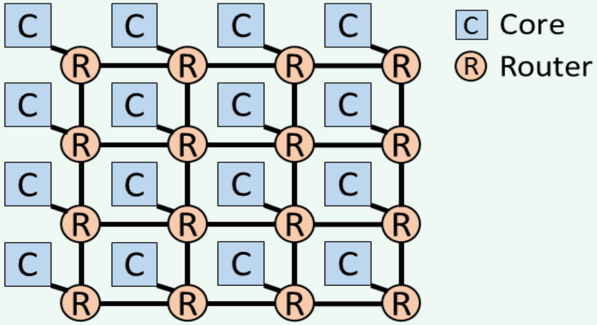
MSI (Modified, Shared, Invalid) protocol

Snooping coherence protocols using bus network

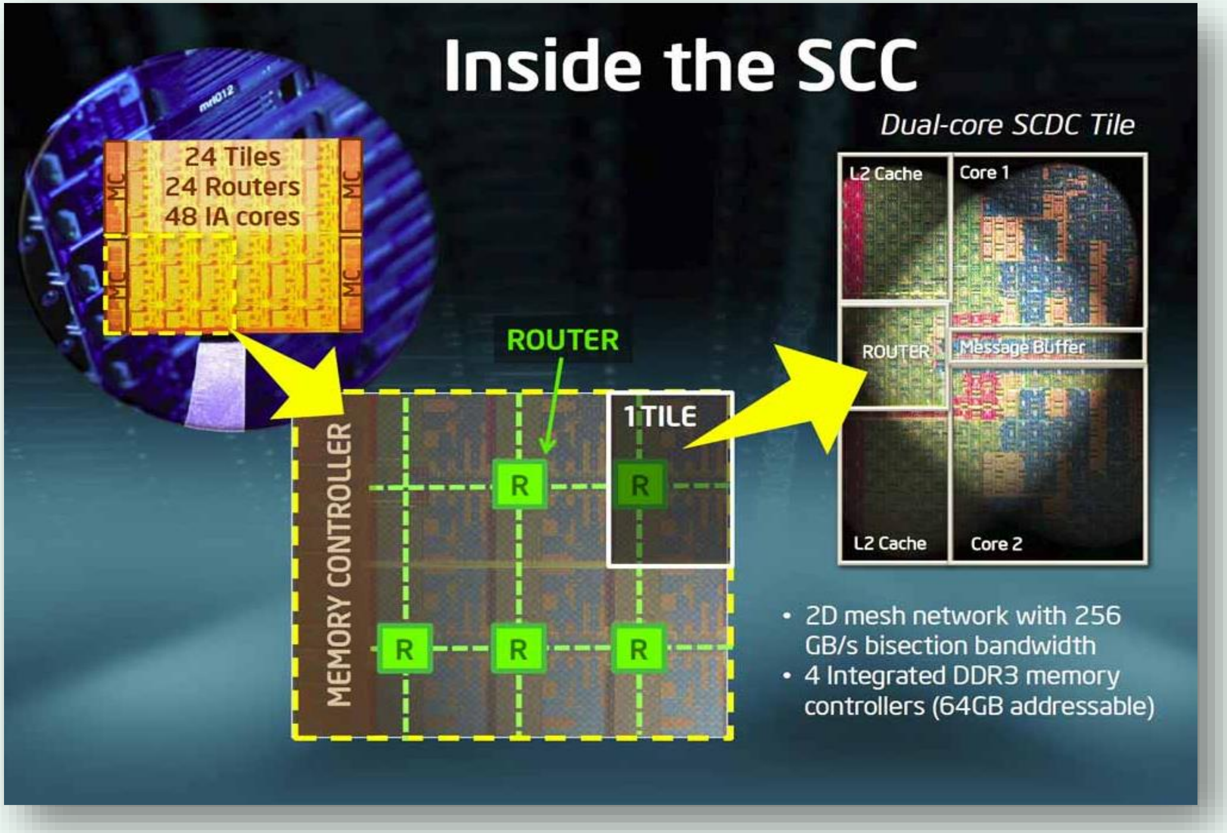
- The basic coherence protocol
  - MSI (Modified, Shared, Invalid) protocol
- Extensions
  - MESI (Modified, Exclusive, Shared, Invalid) protocol
  - MOESI (MESI + Owned) protocol

Intel Single-Chip Cloud Computer (2009)

- To research multi-core processors and parallel processing.



A many-core architecture with 2D Mesh NoC



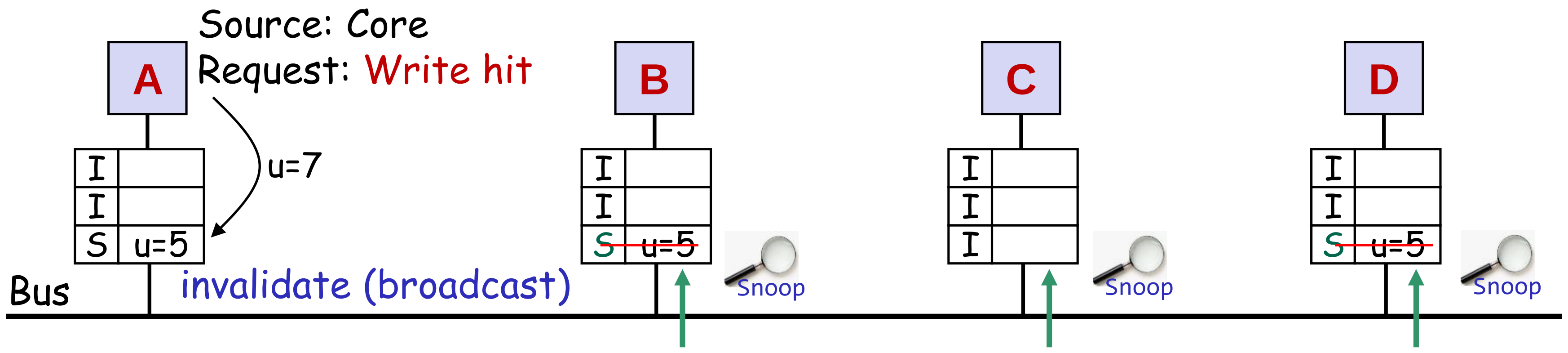
- 2D mesh network with 256 GB/s bisection bandwidth
- 4 Integrated DDR3 memory controllers (64GB addressable)

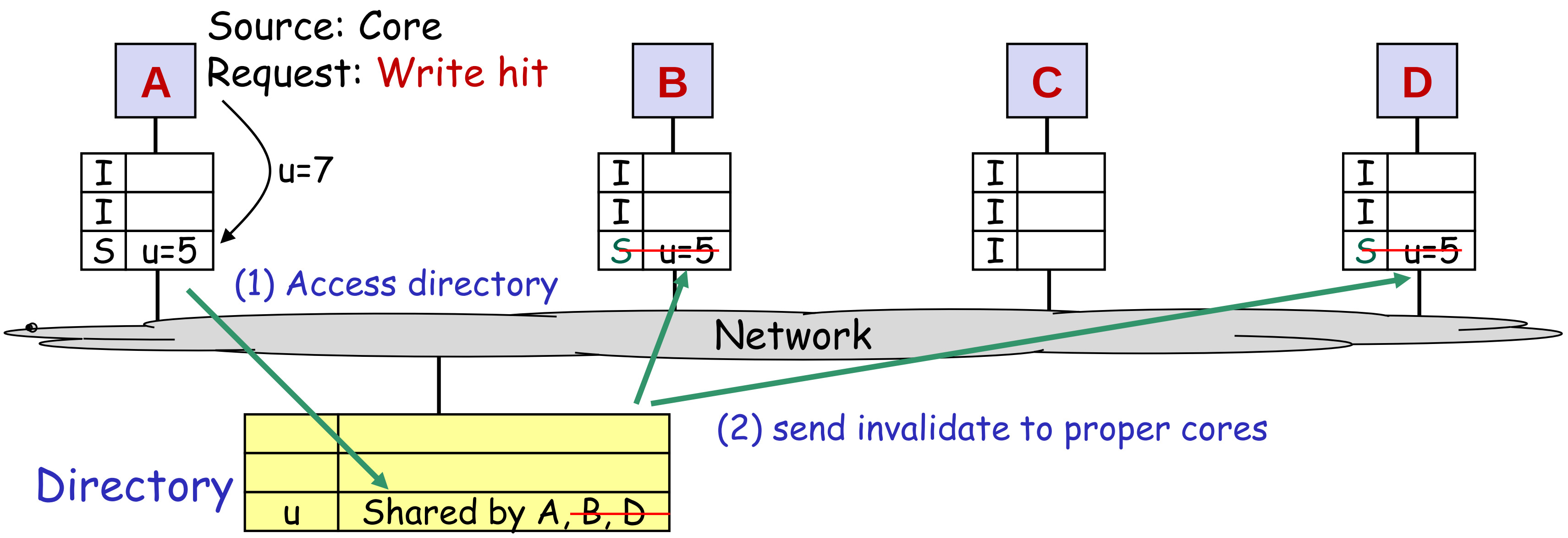
Intel Single-Chip Cloud Computer (48 Core)

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.

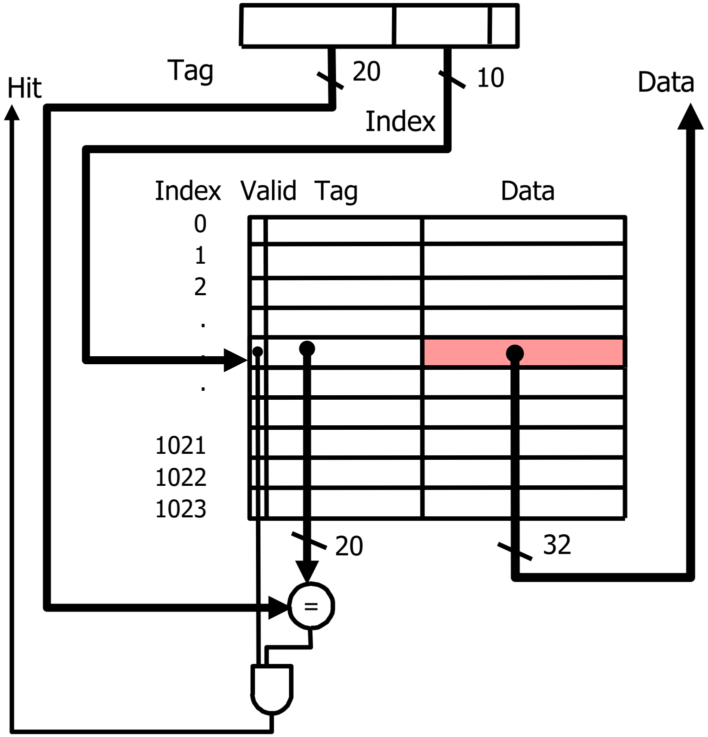
Snooping coherence protocol and one with directory



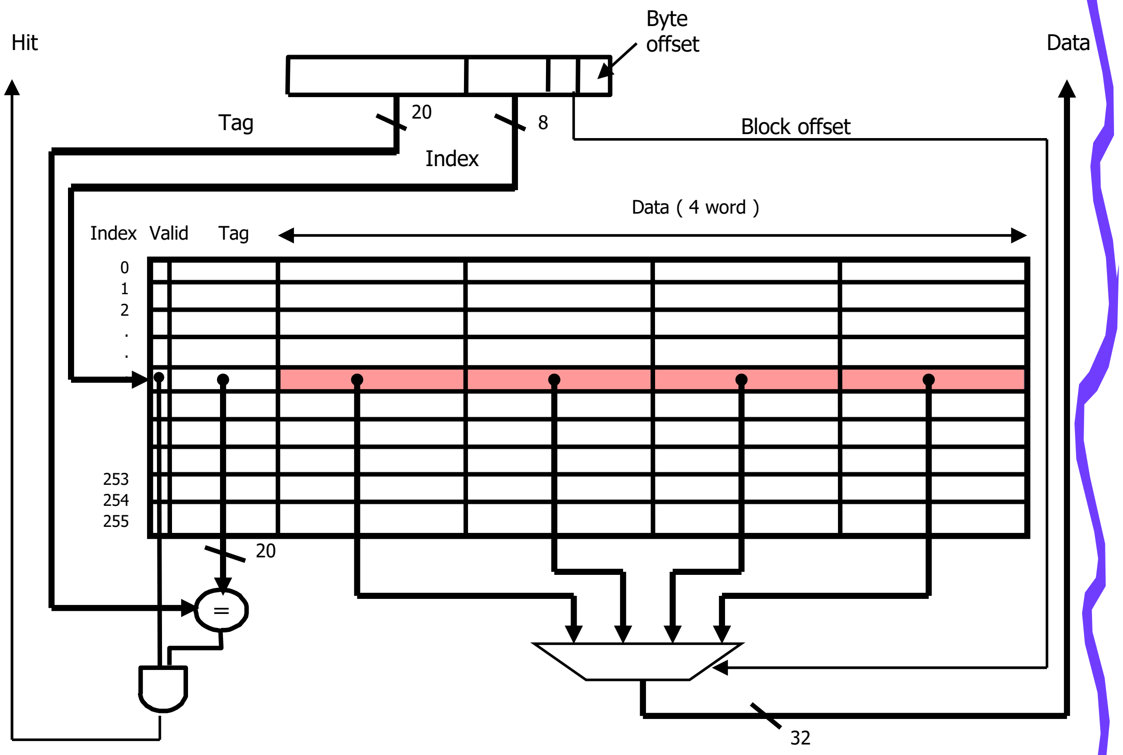


Two caches of different block sizes

- Temporal Locality (Locality in Time):
  - Keep most recently accessed data items closer to the processor
- Spatial Locality (Locality in Space)
  - Move blocks consisting of contiguous words to the upper levels



cache line of one word

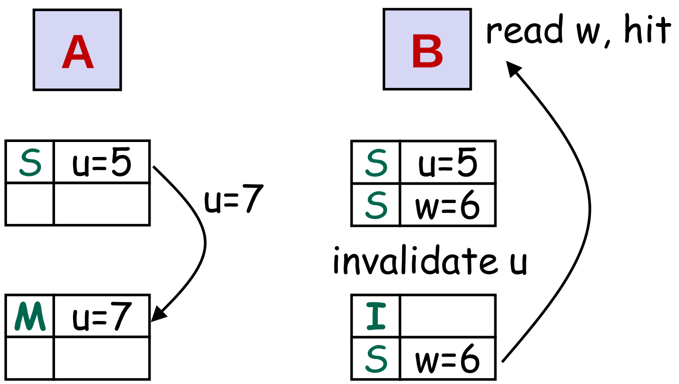


cache line of four words (multiword block)

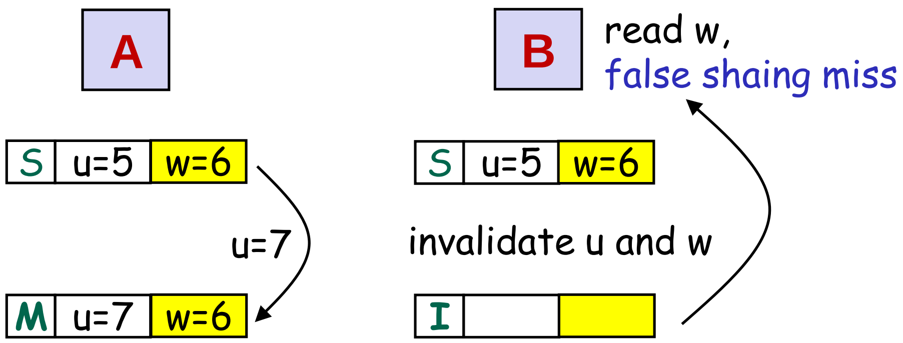
Coherence influences the cache miss rate

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

Senario: A reads u -> B reads u -> B reads w -> A writes u -> B reads w



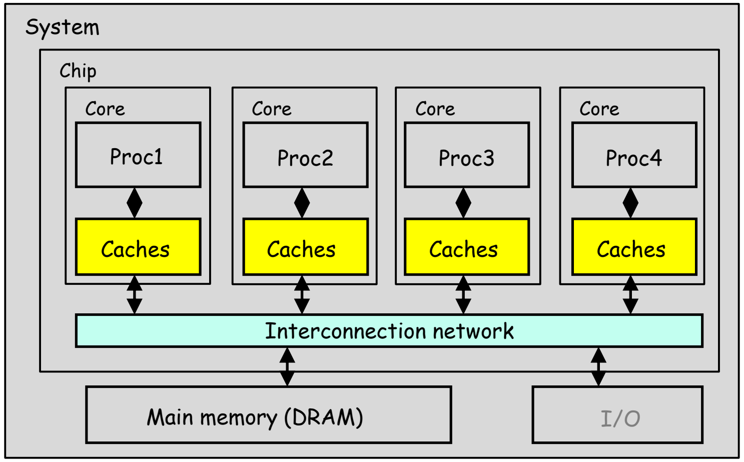
cache line of one word



cache line of two words (multiword block)
u and w are in the same cache block

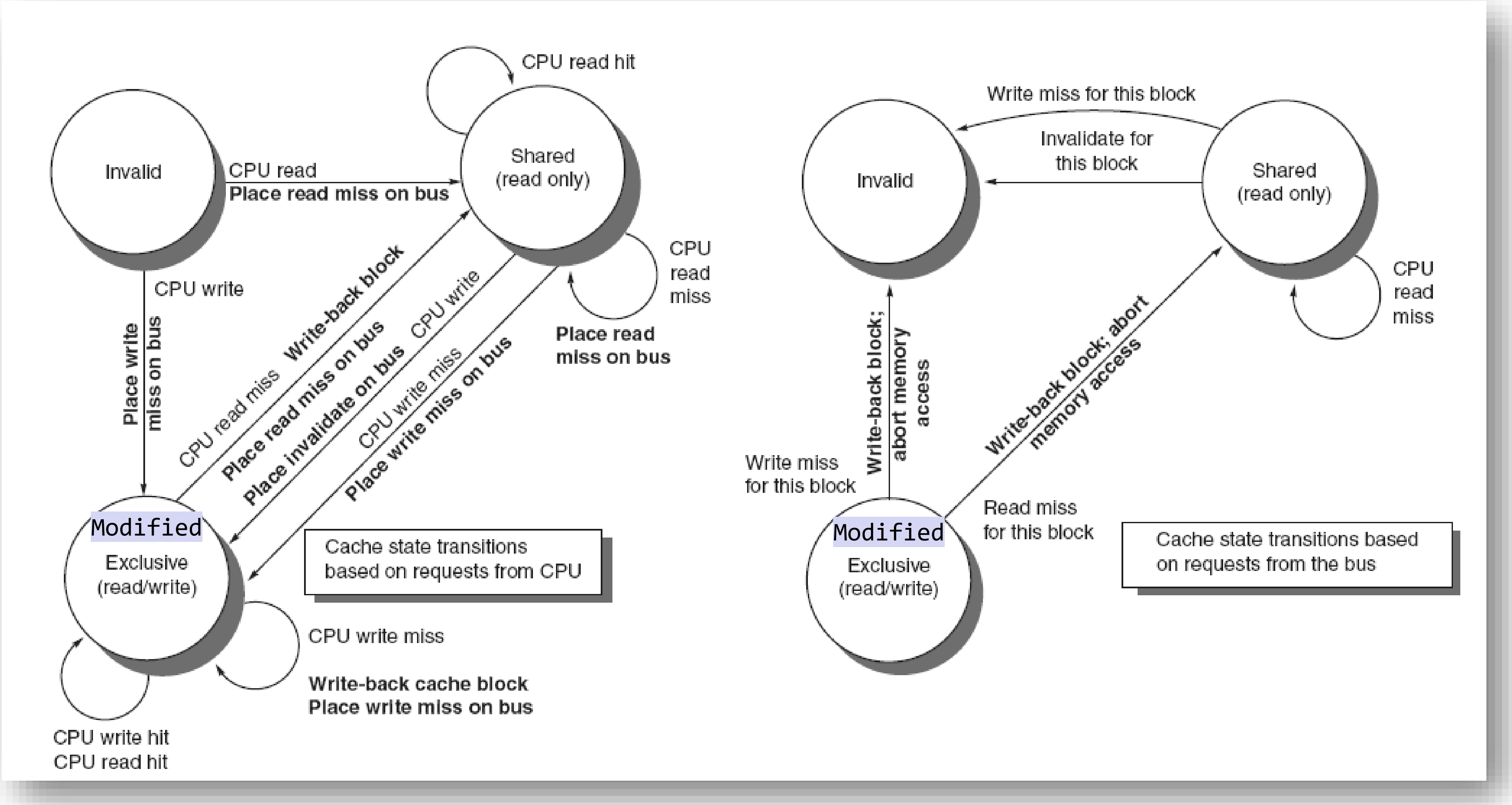
Key components of many-core processors

- 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



Snooping coherence protocols using bus network

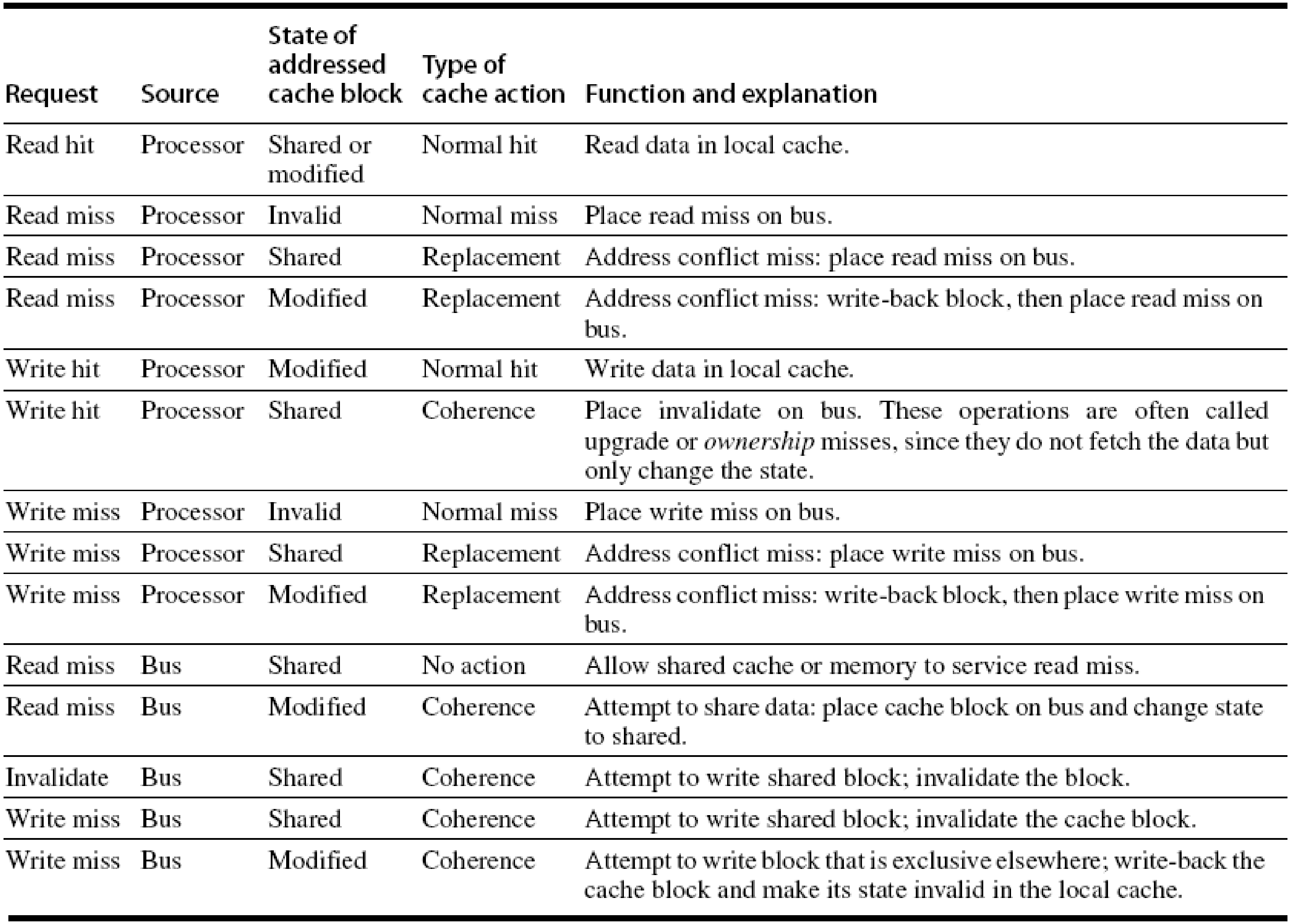
- A write invalidate, cache coherence protocol for a private write-back cache showing the states and state transitions for each block in the cache



MSI (Modified, Shared, Invalid) protocol

Snooping coherence protocols using bus network

- The coherence mechanism of a private cache (using word processor for core).



	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.
Coh1	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.
Coh2	Read miss	Bus	Modified	Coherence	Attempt to share data: place cache block on bus and change state to shared.
Coh3	Invalidate	Bus	Shared	Coherence	Attempt to write shared block; invalidate the block.
Coh4	Write miss	Bus	Shared	Coherence	Attempt to write shared block; invalidate the cache block.
Coh5	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.