EEL 5764 Computer Architecture

Lecture 29: Multiprocessing

Sandip Ray
Department of Electrical and Computer Engineering
University of Florida

Announcements

- There will be no class next week
  - → Tue Nov 12 and Thu Nov 14
- I will put up recorded lectures for both these days during the week of Nov 18-22
- Expect your Midterm grades to show up any time now
  - → Please send any disputes by writing to Canvas within a week of posting and copy all teaching staff

	Single Data	Multiple Data
Single Instruction	**SISD**	**SIMD**
Multiple Instructions	**Not Exists**	**MIMD**

What are Multiprocessors?

- Tightly coupled processors
  - Controlled by a single OS
  - With shared memory space
  - Communication done in HW
- Clusters = processors connected by network
  - Comm. among different processors coordinated by OSs
- Support MIMD execution.
- Single Multicore chips, and systems with multiple chips
- Multithreading – thread executions interleaved on a single processor

Why Multiprocessors?

- Diminishing return from exploiting ILP with rising cost of power and chip area
- Easier to replicate cores
- Rise of web applications and cloud computing where natural parallelism found in web/data-intensive applications

Thread-Level Parallelism (TLP)

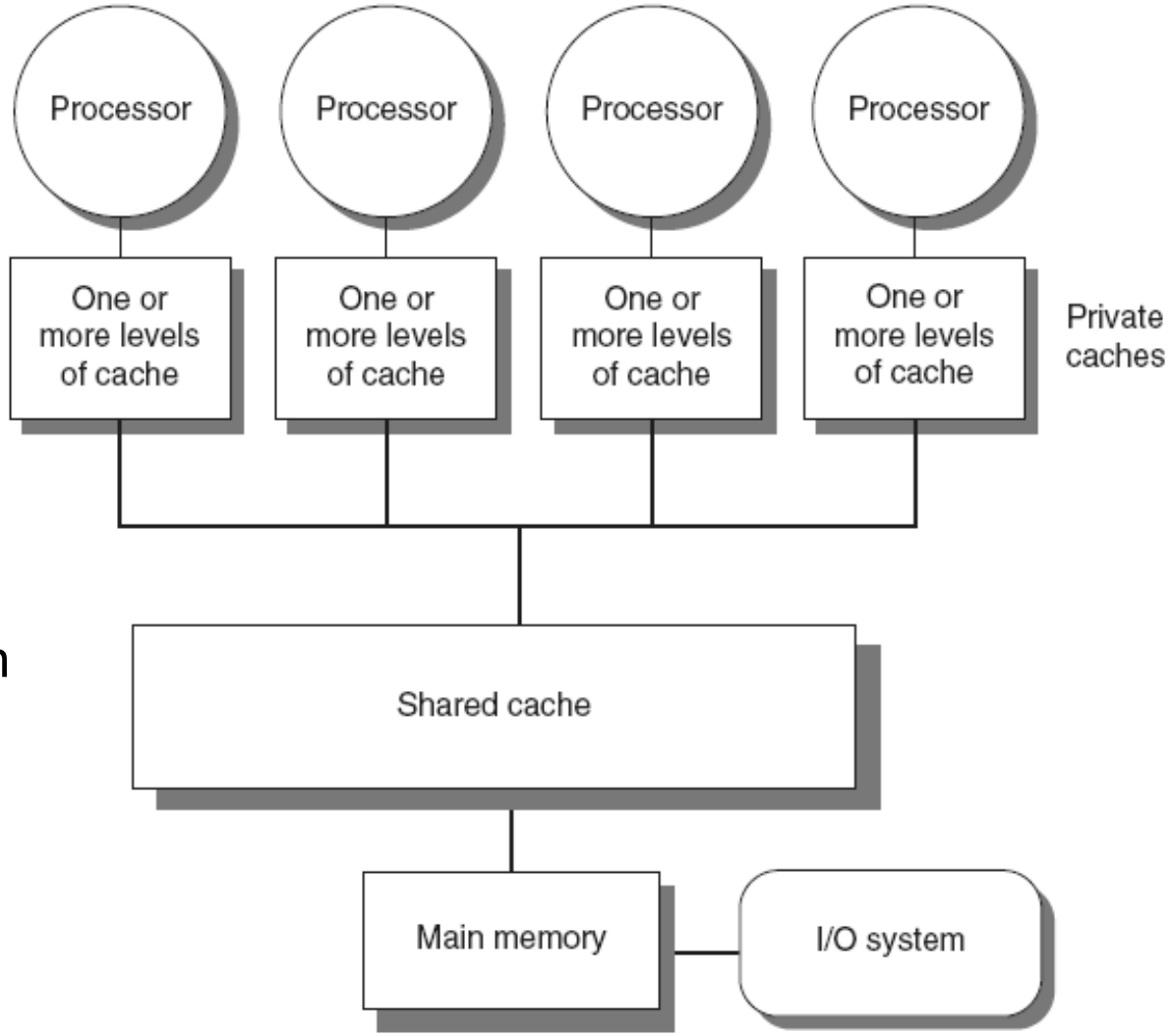
- Thread-Level parallelism
  - multiple running threads – multiple program counters
  - exploited by MIMD model
  - Targeted for tightly-coupled shared-memory multiprocessors

- Types of parallelism
  - Tightly coupled – threads collaborating for a single task
  - Loosely coupled – multiple programs running independently

Exploiting TLP

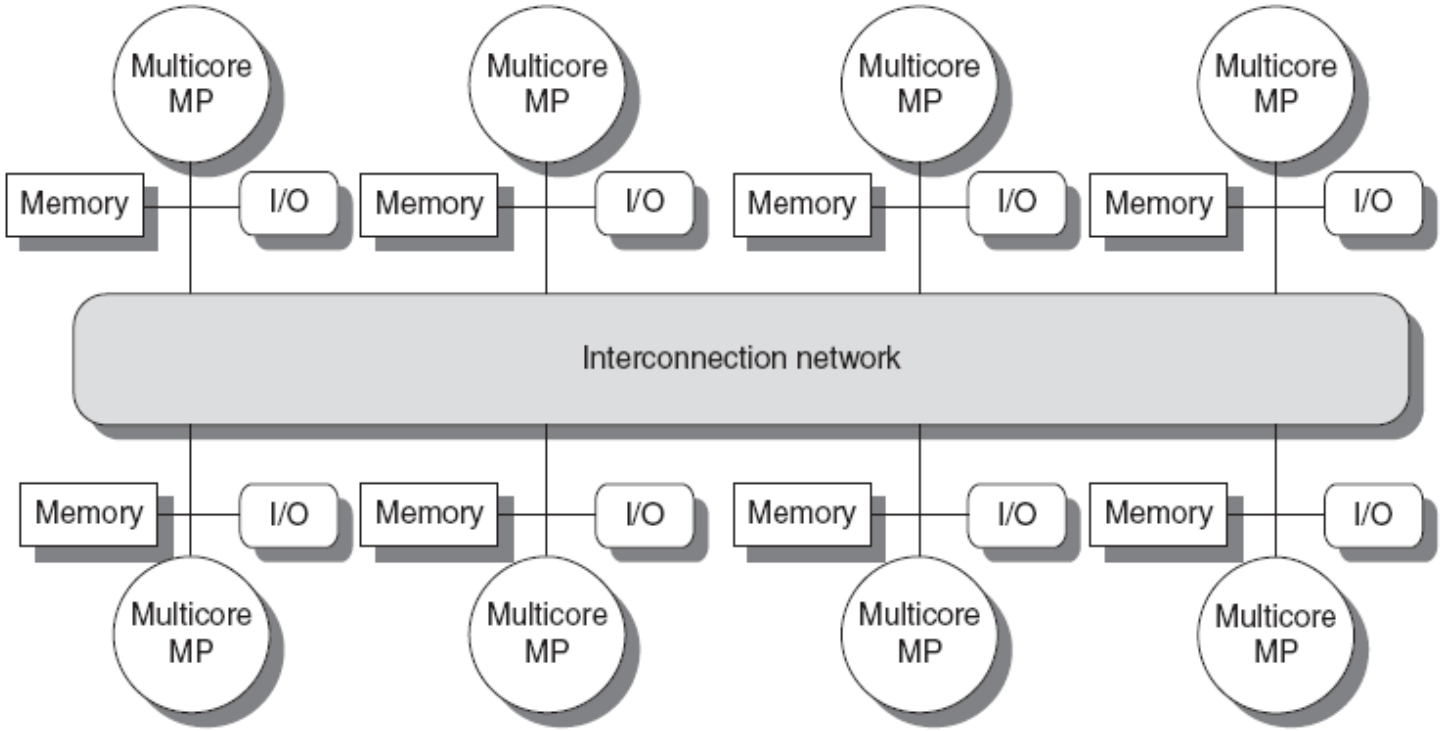
- Applications should contain abundant concurrency
  - For *n* processors, need *n*+ threads
- Identified by programmers or OS
- Amount of computation assigned to each thread = grain size
- Thread grain size must be large
  - To reduce overheads associated with thread execution

Symmetric multiprocessors (SMP)



- Small number of cores (<= 8 cores)
- Share single memory with uniform memory latency
- Also called UMA MP.

Distributed Shared Memory (DSM)



- Memory distributed among processors
- Non-uniform memory access/latency (NUMA)
- Processors connected via direct (switched) and non-direct (multi-hop) interconnection networks
- Long latency to access remote memory.

Centralized Shared-Memory Architecture

Caching

- Reduce latency of main memory and remote access
  - Also reduce contentions among memory accesses from different processors
- There are private & shared caches
- Private data: used by a single processor
- Shared data: used by multiple processors
  - Replicated in multiple private caches

Cache Coherence Problem

- Processors may see different values through their private caches

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	0

Cache Coherence

- Coherence – what values returned for reads
  - A read by a processor A to a location X that follows a write to X by A returns the value written by A if no other processors write in between
  - A read by processor A to location X after a write to X by processor B returns the written value if the read and write are sufficiently separated, and no other writes occur in between
  - Writes to the same location are serialized

- Consistency – when a written value seen by a read
  - Concerns reads & writes to different memory locations from multiple processors

Enforcing Coherence

- Coherent caches provide:
  - *Migration*: movement of data – reduce latency
  - *Replication*: multiple copies of data – reduce latency & memory bandwidth demand

- Cache coherence protocols
  - Snooping
    - Every cache tracks sharing status of each cache block
    - Mem requests are broadcast on a bus to all caches
    - Writes serialized naturally
  - Directory based
    - Sharing status of each cache block kept in one location

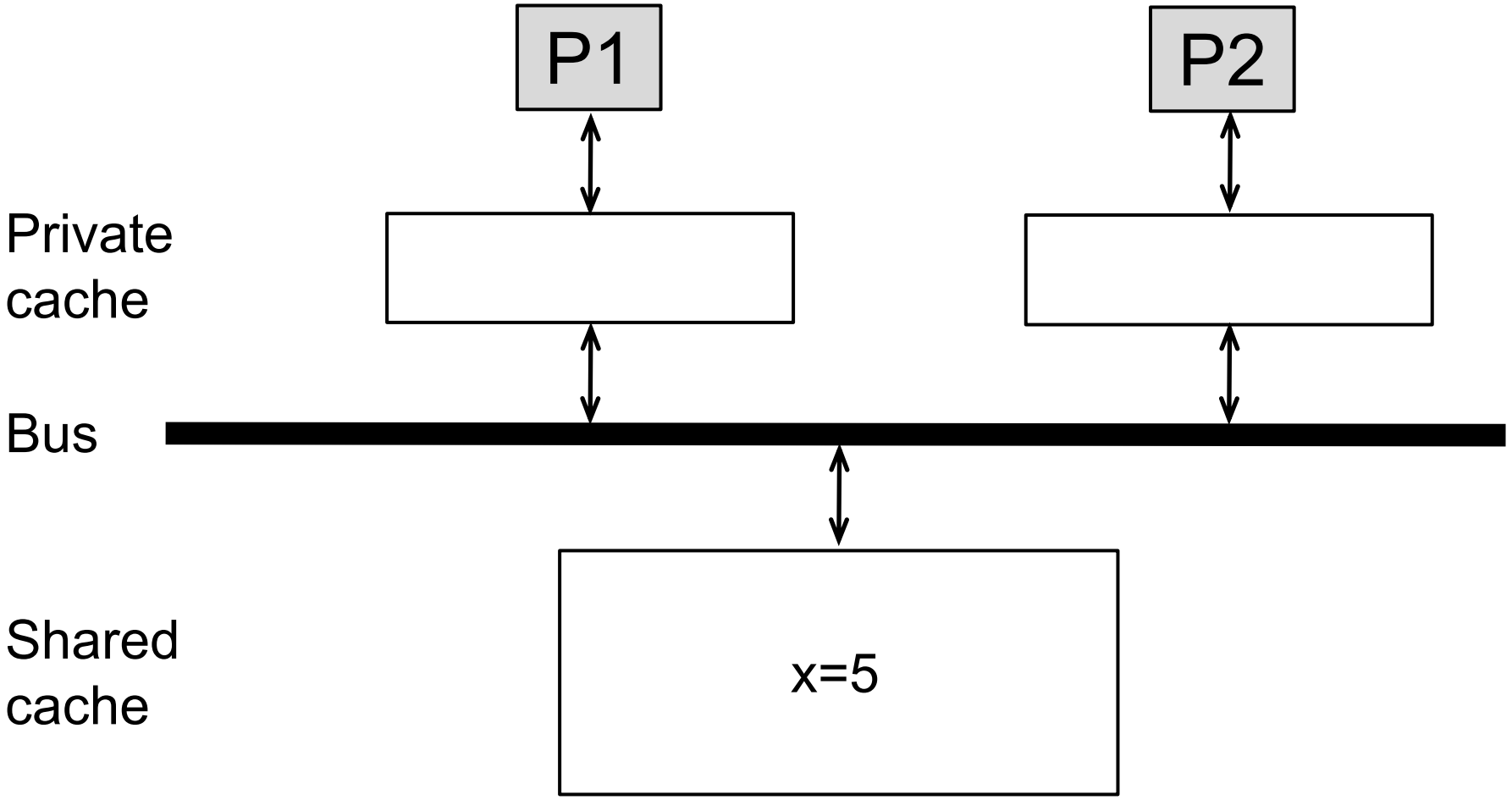
Snoopy Coherence Protocols

- Write invalidate
  - On write, invalidate all other copies
  - Use bus itself to serialize
    - Write cannot complete until bus access is obtained
- Write update
  - On write, update all copies
- **Invariant**: blocks in cache are always coherent

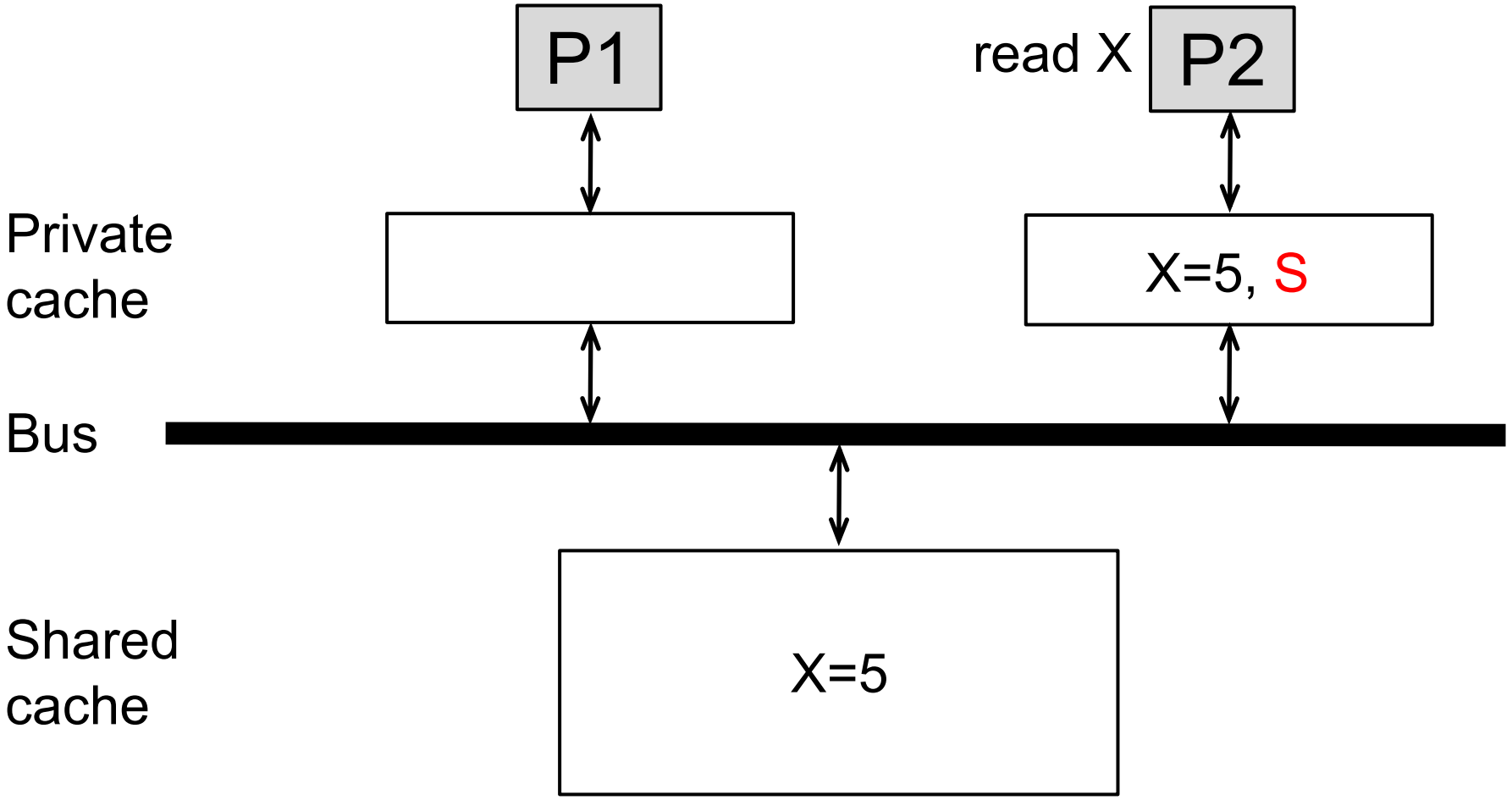
Snoopy Coherence Protocols

- Each cache block is in one of following states
  - Invalid (I)
  - Shared (S)
  - Modified (M) – implies exclusion or not shared
- Locating an item when a read miss occurs
  - In write-back cache, the updated value must be sent to the requesting processor
- Cache lines marked as shared or exclusive/modified
  - Only writes to shared lines need an invalidate broadcast
    - After this, the line is marked as exclusive

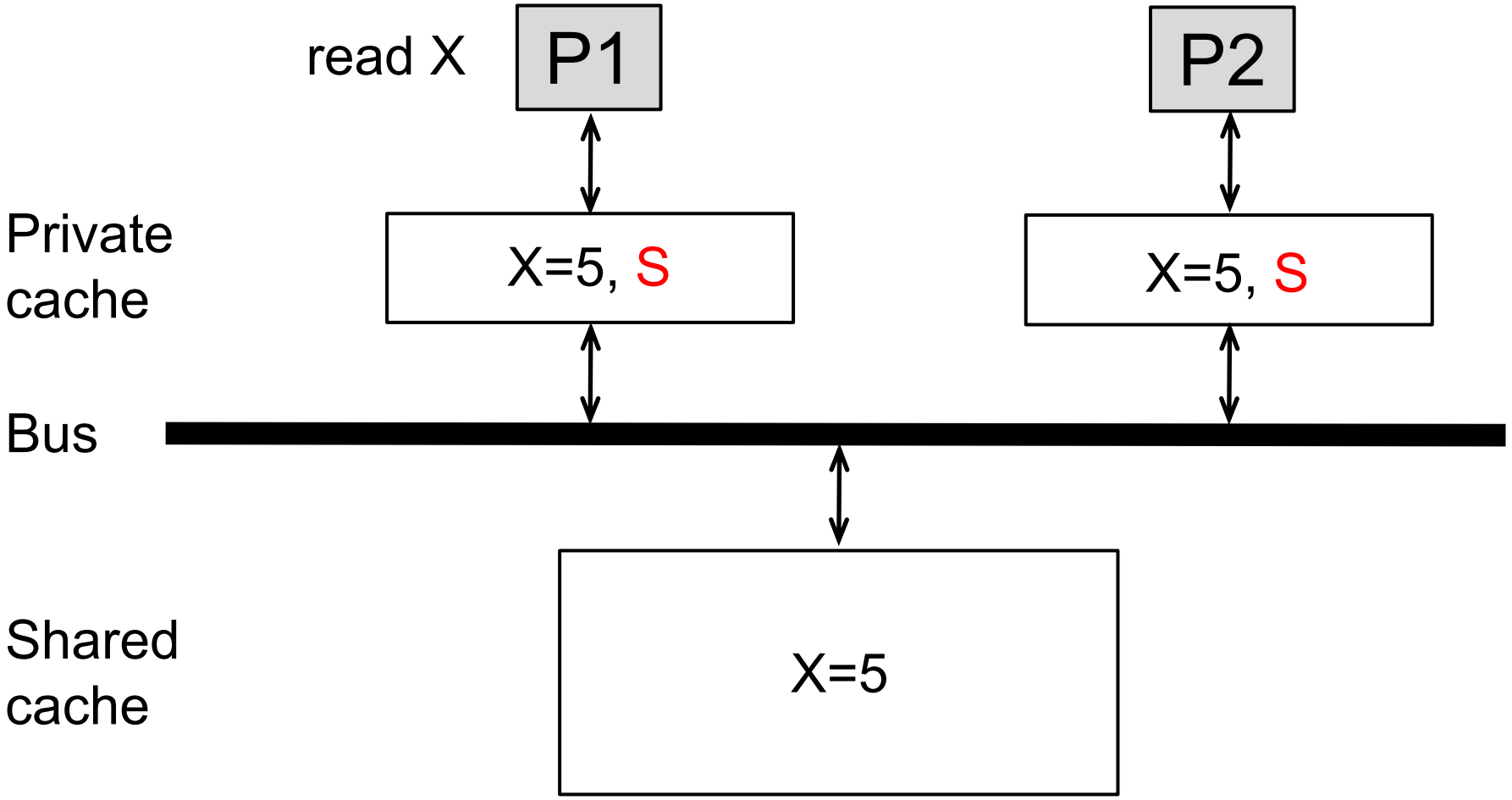
Snoopy Coherence Protocols



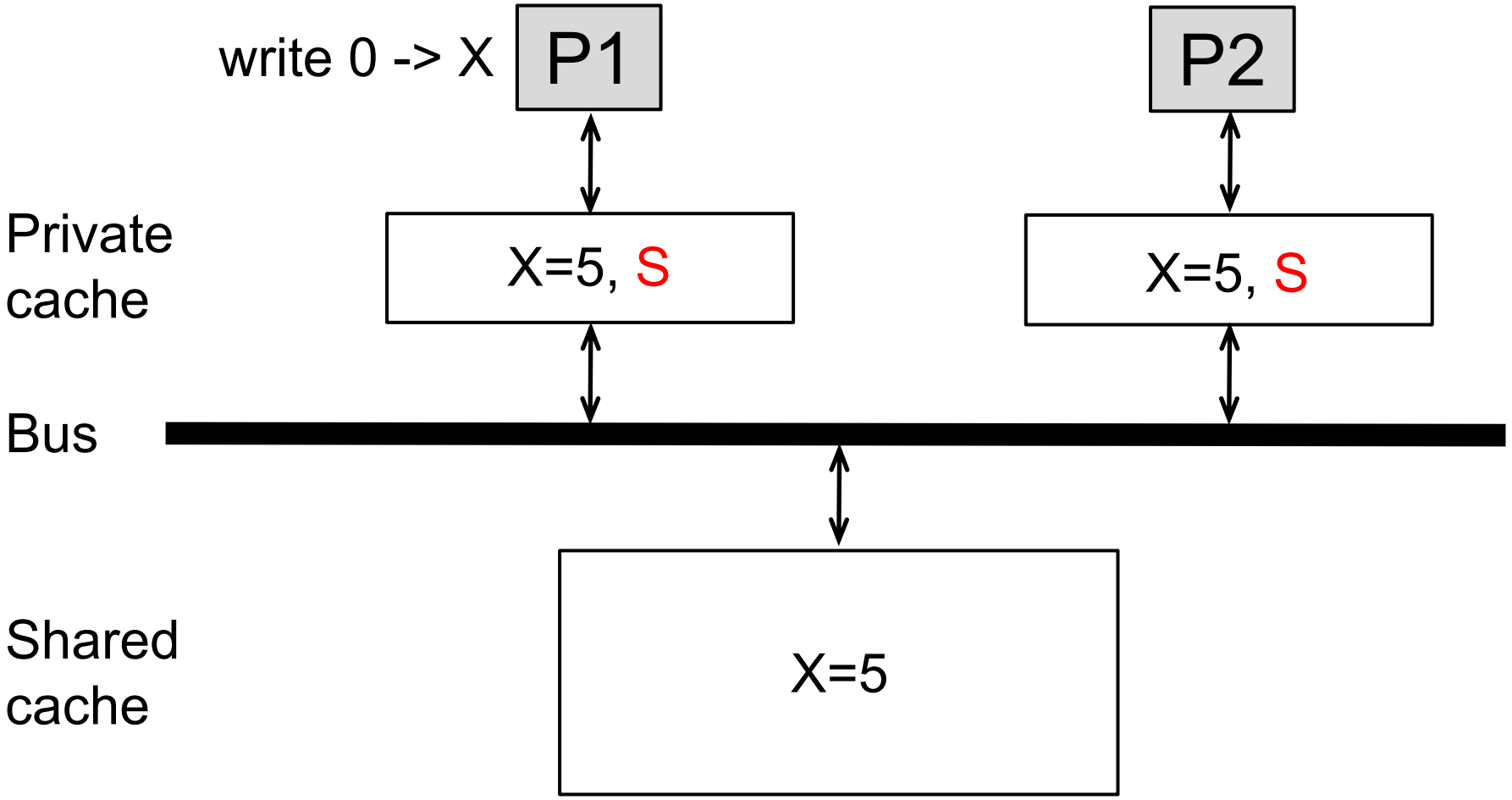
Snoopy Coherence Protocols



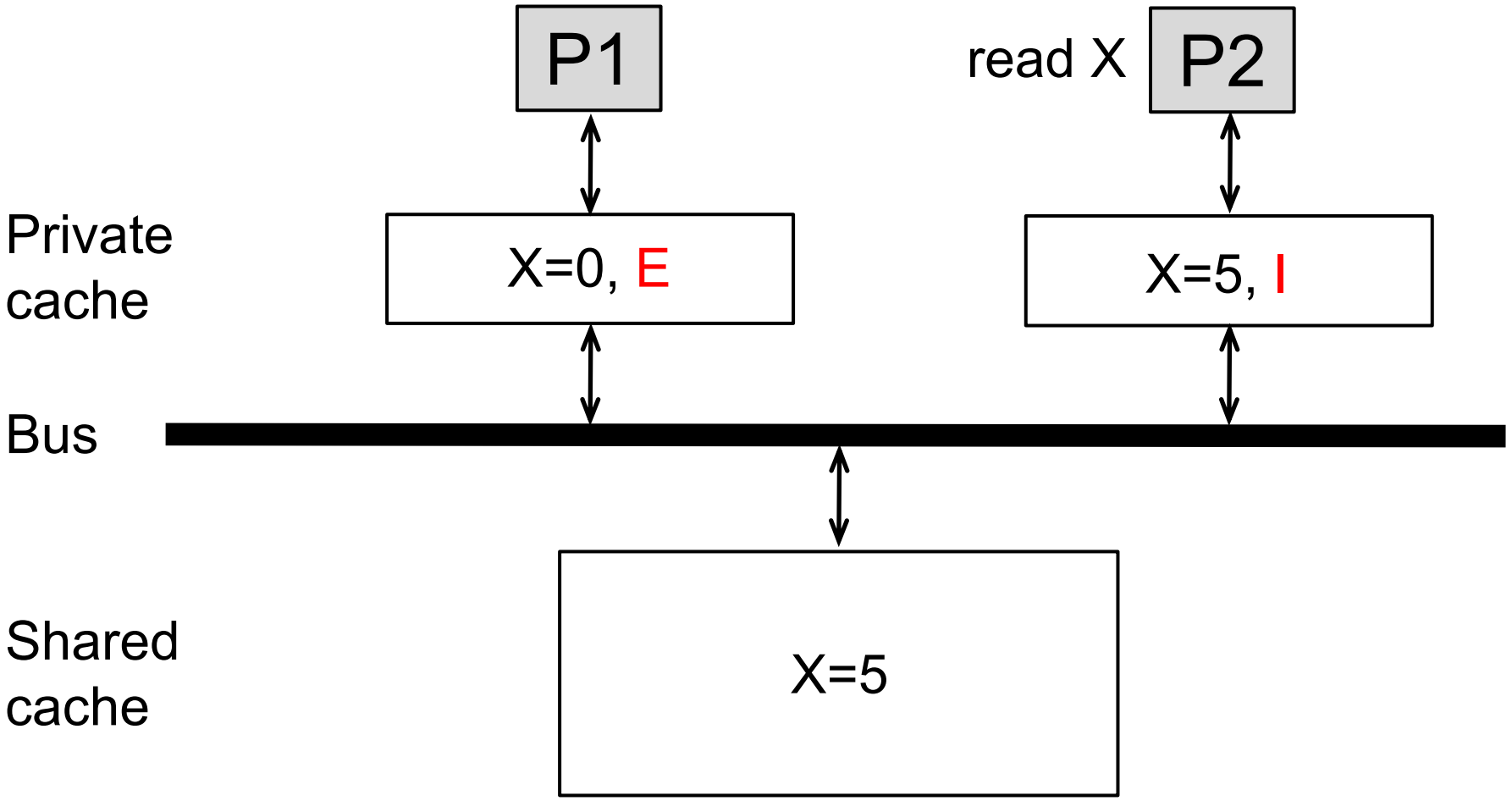
Snoopy Coherence Protocols



Snoopy Coherence Protocols



Snoopy Coherence Protocols



where does P2 find data for X?

Snoopy Coherence Protocols