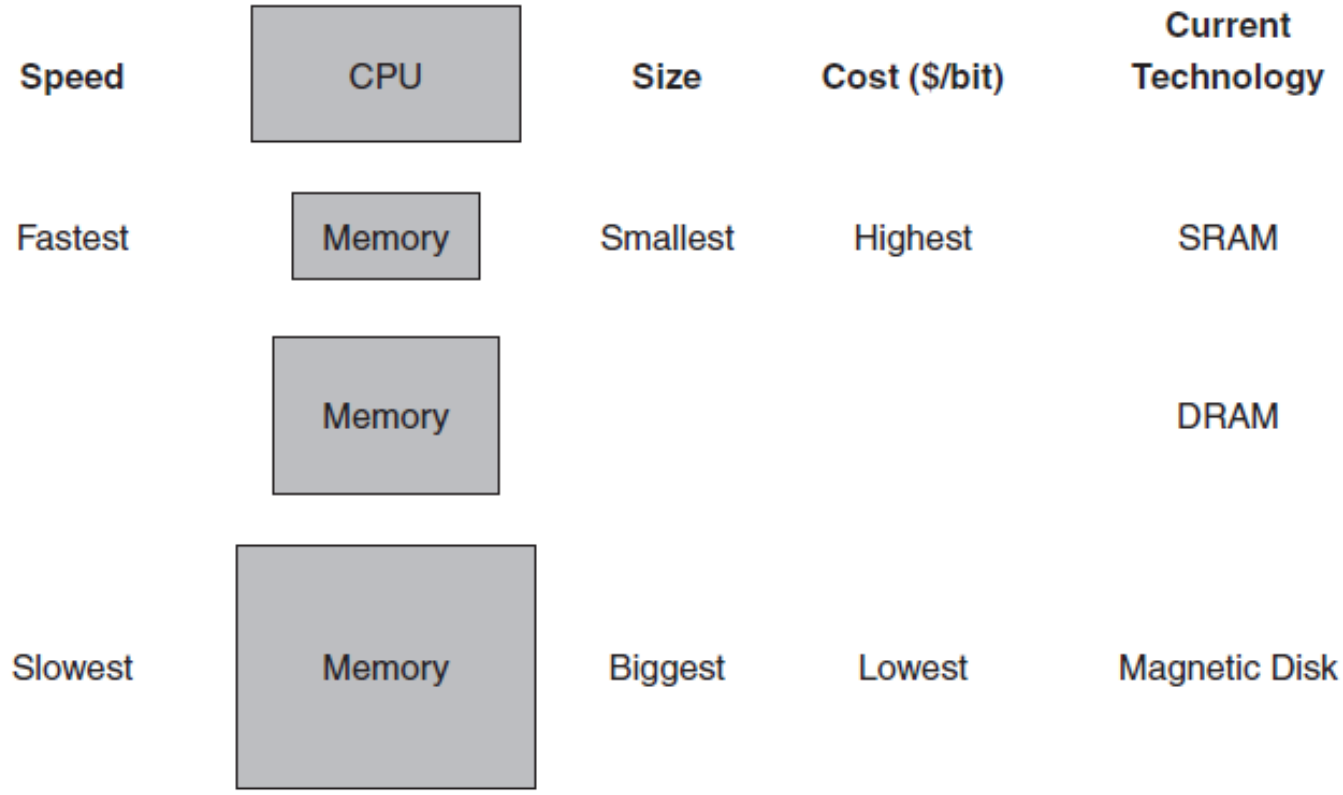
Unit 4

Memory Unit

Memory hierarchy

- Memory hierarchy: A structure that uses multiple levels of memories, as the distance from the CPU increases, the size of the memories and the access time both increase.



Introduction to Cache memory

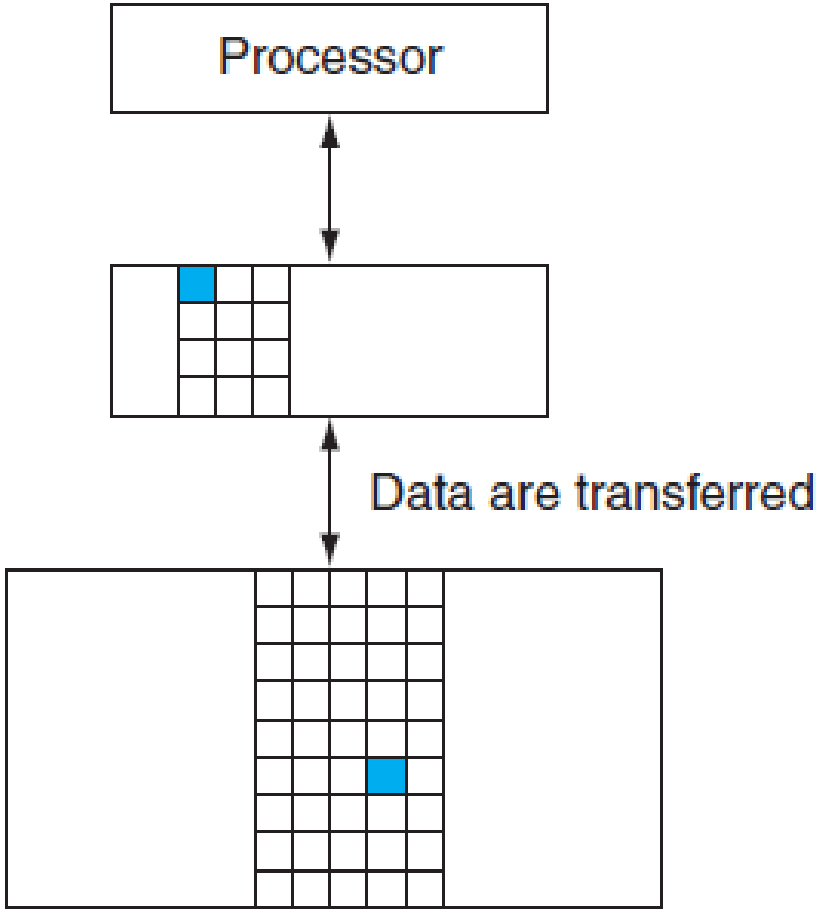
- **The principle of locality**: Programs access a relatively small portion of their address space at any instant of time.

- **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.
- **Example:** Library book.

For explanation follow the video https://nptel.ac.in/courses/106/106/106106134/

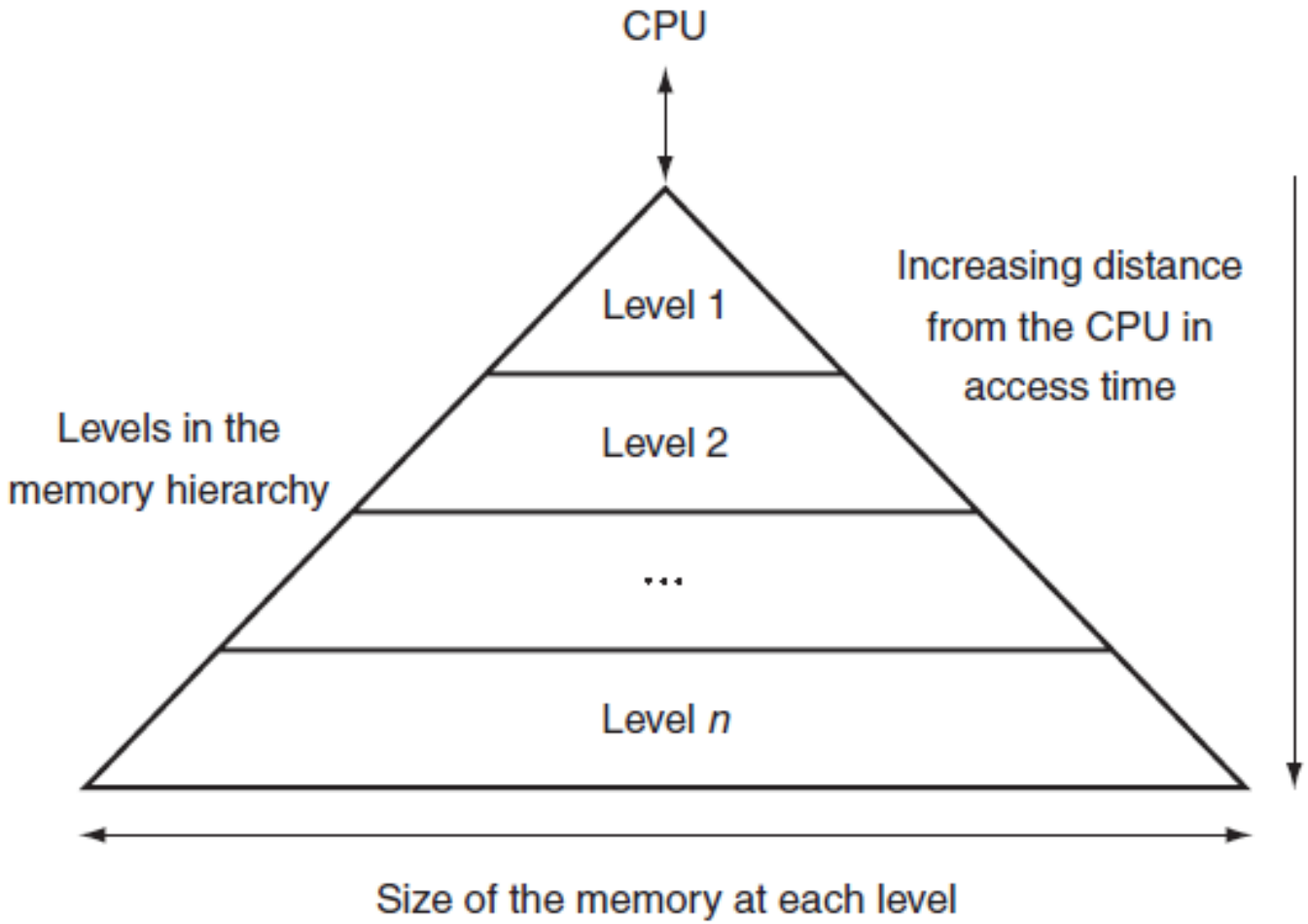
Part1 and part 2 of cache memory hierarchy.



Every pair of levels in the memory hierarchy can be thought of as having an upper and lower level

Terms

- **Block : The minimum unit of** information that can be either present or not present in the two-level hierarchy.
- **Hit rate: The fraction of memory** accesses found in a cache.
- **Miss rate: The fraction of** memory accesses not found in a level of the memory hierarchy.
- **Hit time: The time required to** access a level of the memory hierarchy, including the time needed to determine whether the access is a hit or a miss.
- **Miss penalty: The time** required to fetch a block into a level of the memory hierarchy from the lower level, including the time to access the block, transmit it from one level to the other, and insert it in the level that experienced the miss and then pass the block to the requestor.



Structure of a memory hierarchy: As the distance from the processor increases, so does the size.

Cache mapping techniques

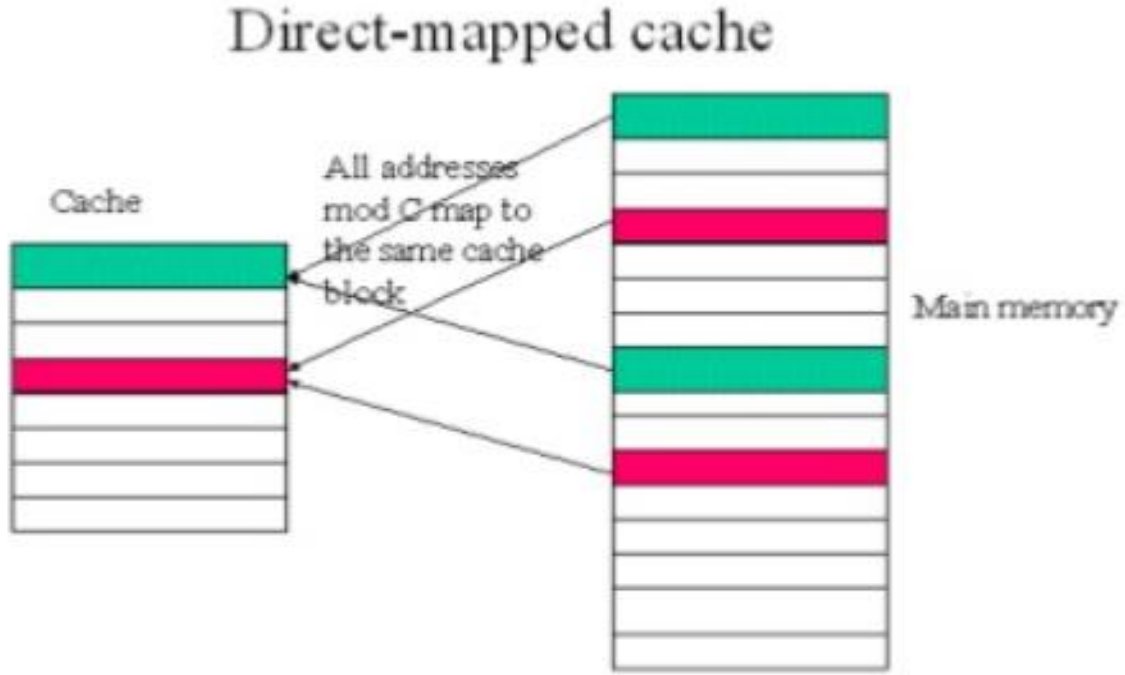
- Three mappings:
  - Direct Mapped Cache
  - Associative Cache
  - Set-associative Cache

For more explanation follow below link

http://www.infocobuild.com/education/audio-video-courses/computer-science/ComputerArchitecture-Mutyam-IIT-Madras/lecture-07.html

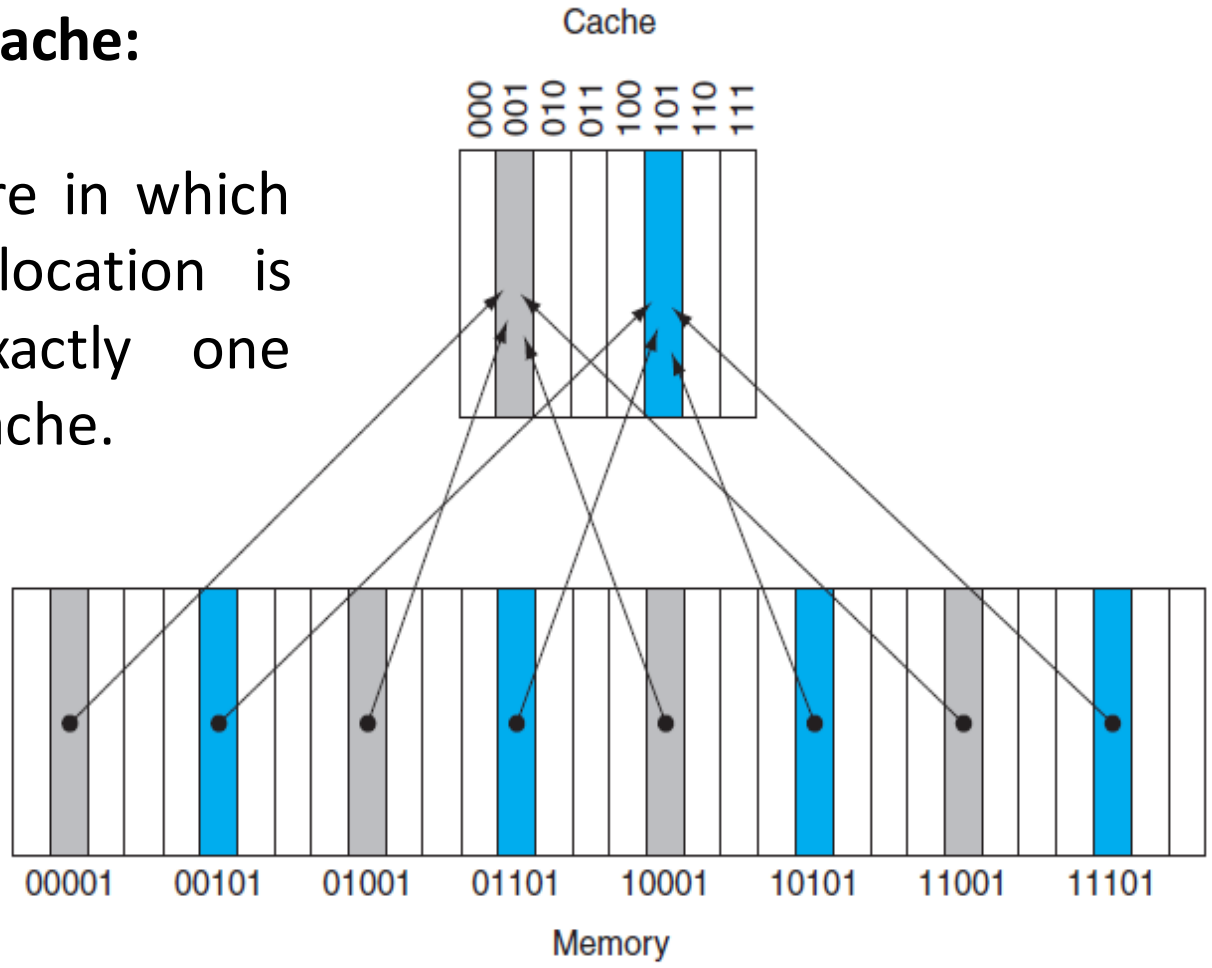
Different Cache Mapping Techniques

- **Direct Mapping:** Any main memory location can be loaded into one fixed location in cache.
  - **Advantage-** No search is required as there is only one location in cache for each main memory location.
  - **Disadvantage-** Hit ratio is poor as there is only one fixed location for each main memory location.



Direct-mapped cache:

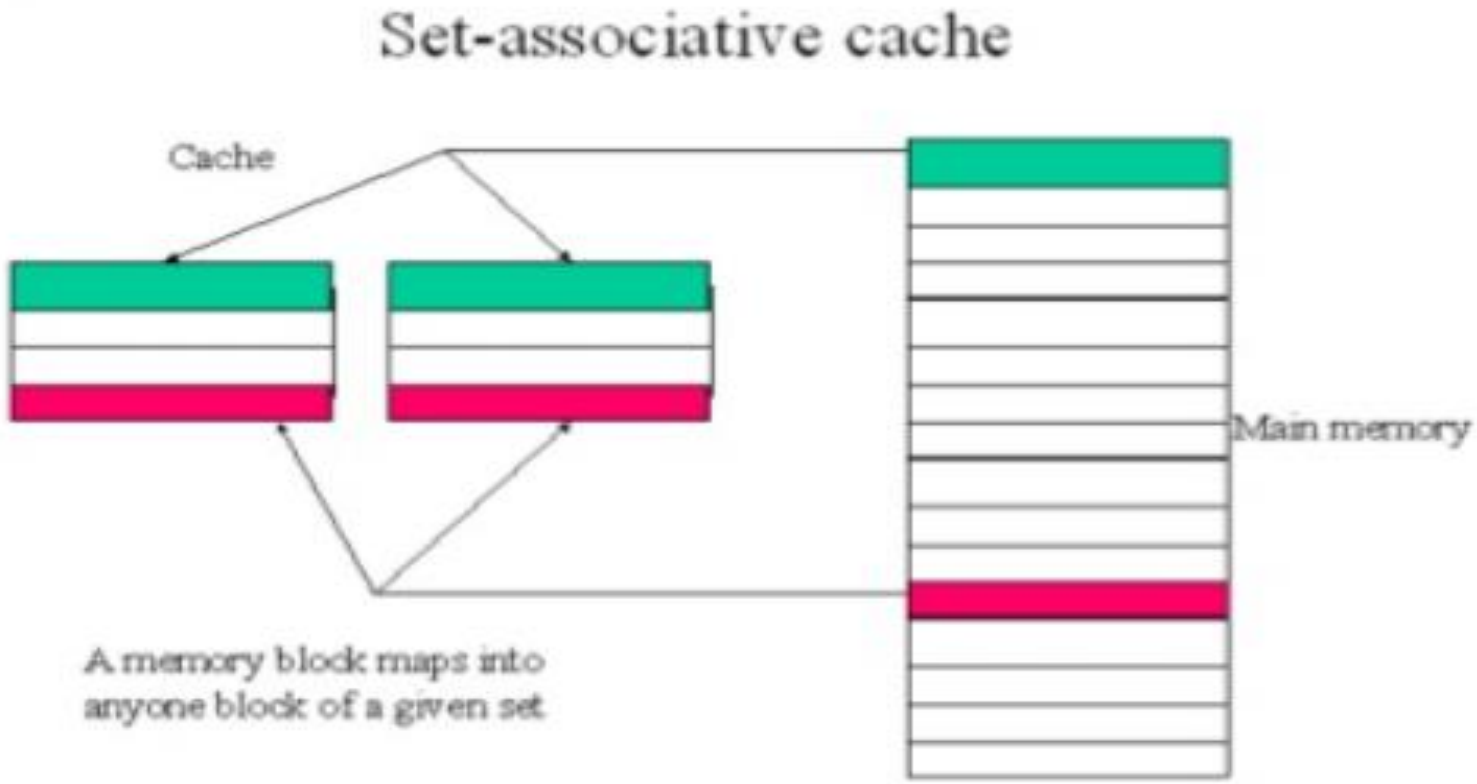
A cache structure in which each memory location is mapped to exactly one location in the cache.



A direct-mapped caches use the mapping

(Block address) modulo (Number of cache blocks in the cache)

- **Set Associative Mapping:** Any main memory location can be loaded to at least two or more memory location in cache.
  - **Advantage-** Hit ratio is more compared to direct mapped cache and amount of energy consumption is less as compared to fully associative cache because limited no tag patterns needs to be searched.



Block placement as a variation on set associativity

One-way set associative (direct mapped)

Block	Tag	Data
0		
1		
2		
3		
4		
5		
6		
7		

Two-way set associative

Set	Tag	Data	Tag	Data
0				
1				
2				
3				

Four-way set associative

Set	Tag	Data	Tag	Data	Tag	Data	Tag	Data
0								
1								

Eight-way set associative (fully associative)

Tag	Data	Tag	Data	Tag	Data	Tag	Data	Tag	Data	Tag	Data	Tag	Data	Tag	Data

Misses and Associativity in Caches

Example:

Assume a two way set associative cache with four one-word blocks. Find the number of misses for each cache organization given the following sequence of block addresses: 0, 8, 0, 6, and 8.

Solution

Set Associative Cache:

Block address	Cache set
0	(0 modulo 2) = 0
6	(6 modulo 2) = 0
8	(8 modulo 2) = 0

Address of memory block accessed	Hit or miss	Contents of cache blocks after reference			
		Set 0	Set 0	Set 1	Set 1
0	miss	Memory[0]			
8	miss	Memory[0]	Memory[8]		
0	hit	Memory[0]	Memory[8]		
6	miss	Memory[0]	Memory[6]		
8	miss	Memory[8]	Memory[6]		

Solution

Direct Mapped Cache:

Block address	Cache block
0	(0 modulo 4) = 0
6	(6 modulo 4) = 2
8	(8 modulo 4) = 0

Address of memory block accessed	Hit or miss	Contents of cache blocks after reference			
		0	1	2	3
0	miss	Memory[0]			
8	miss	Memory[8]			
0	miss	Memory[0]			
6	miss	Memory[0]		Memory[6]	
8	miss	Memory[8]		Memory[6]	

Fully associative cache

- Fully associative cache : To find a given block in a fully associative cache, all the entries in the cache must be searched because a block can be placed in any one block.

- To make the search practical, it is done in parallel with a comparator associated with each cache entry.

Solution (consider previous problem with fully associative

Fully Associative Cache:

Address of memory block accessed	Hit or miss	Contents of cache blocks after reference			
		Block 0	Block 1	Block 2	Block 3
0	miss	Memory[0]			
8	miss	Memory[0]	Memory[8]		
0	hit	Memory[0]	Memory[8]		
6	miss	Memory[0]	Memory[8]	Memory[6]	
8	hit	Memory[0]	Memory[8]	Memory[6]	

Multiprocessor architecture

- Multiprocessors: defined as computers consisting of tightly coupled processors whose coordination and usage are typically controlled by a single operating system and that share memory through a shared address space.
- The multiprocessors typically range in size from a dual processor to dozens of processors and communicate and coordinate through the sharing of memory.
- Sharing through memory implies a shared address space, it does not necessarily mean there is a single physical memory.
- It can be both single-chip systems with multiple cores, known as multicore, and computers consisting of multiple chips, each of which may be a multicore design.

Existing shared-memory multiprocessors fall into two classes, depending on the number of processors involved, which in turn dictates a memory organization and interconnect strategy.

The first group, called symmetric (shared-memory) multiprocessors (SMPs), or centralized shared-memory multiprocessors, features small numbers of cores, typically eight or fewer.

The processors to share a single centralized memory that all processors have equal access to, hence the term symmetric.

In multicore chips, the memory is effectively shared in a centralized fashion among the cores, and all existing multicores are SMPs.

SMP architectures are also sometimes called uniform memory access (UMA) multiprocessors, since all processors have a uniform latency even if the memory is organized into multiple banks.

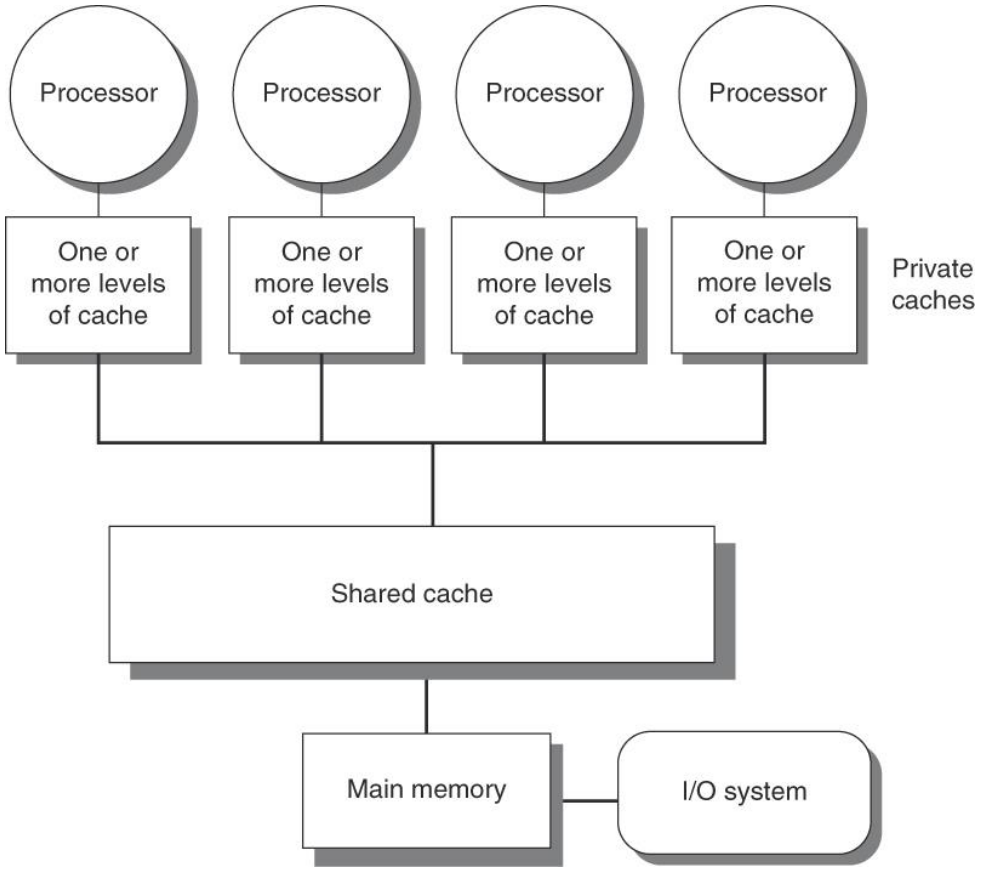


Figure 5.1 Basic structure of a centralized shared-memory multiprocessor based on a multicore chip. Multiple processor–cache subsystems share the same physical memory, typically with one level of shared cache, and one or more levels of private per-core cache. The key architectural property is the uniform access time to all of the memory from all of the processors. In a multichip version the shared cache would be omitted and the bus or interconnection network connecting the processors to memory would run between chips as opposed to within a single chip.

- The second group consists of multiprocessors with physically distributed memory, called distributed shared memory (DSM).

- To support larger processor counts, memory must be distributed among the processors rather than centralized to support the bandwidth excessively long access latency.

- Distributing the memory among the nodes both increases the bandwidth and reduces the latency to local memory.

- A DSM multiprocessor is also called a NUMA (nonuniform memory access), since the access time depends on the location of a data word in memory.

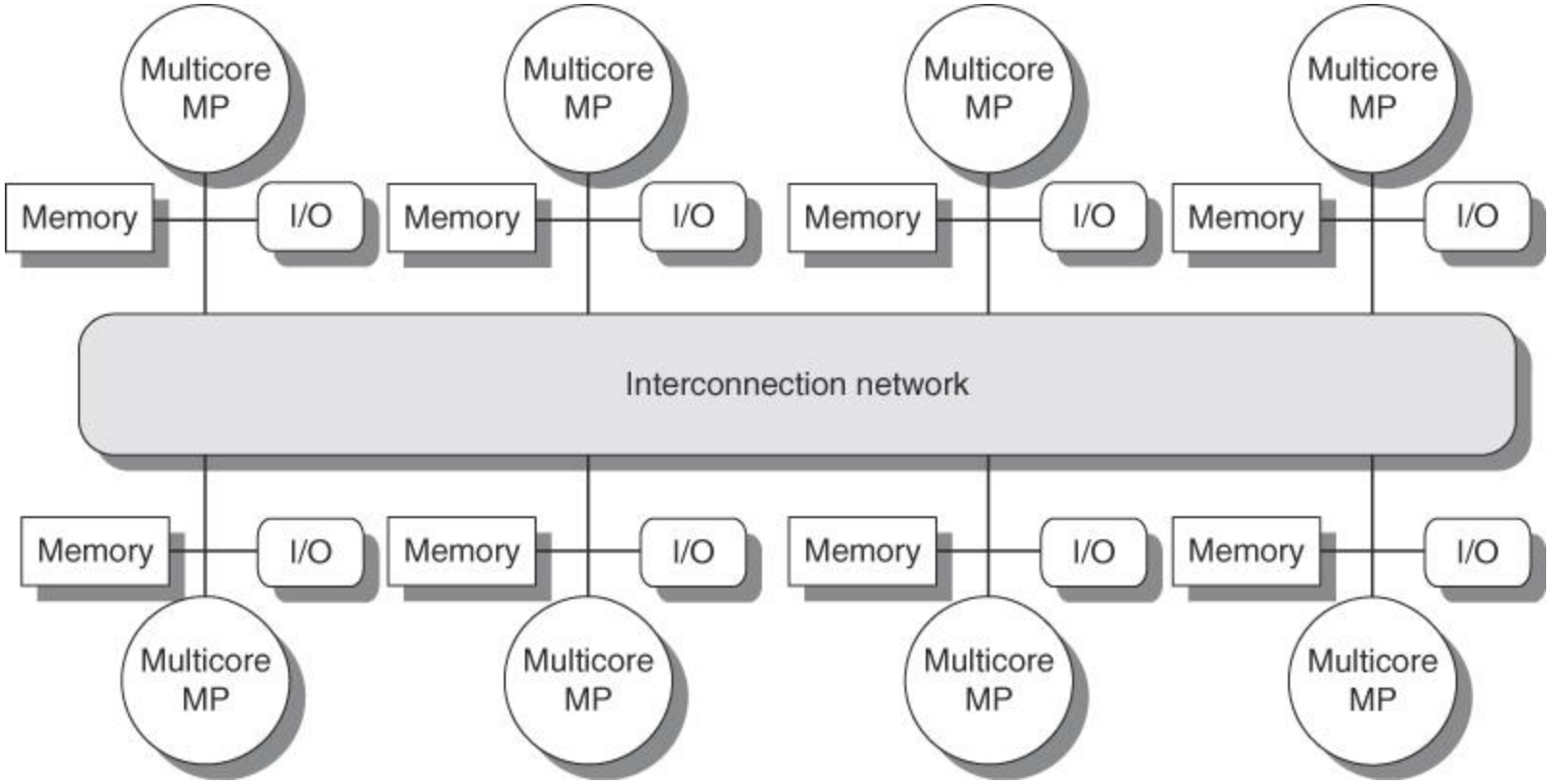


Figure 5.2 The basic architecture of a distributed-memory multiprocessor in 2011 typically consists of a multicore multiprocessor chip with memory and possibly I/O attached and an interface to an interconnection network that connects all the nodes. Each processor core shares the entire memory, although the access time to the lock memory attached to the core's chip will be much faster than the access time to remote memories.

Centralized Shared-Memory Architectures

- In centralized approach, recent microprocessors directly connect memory to a single chip, which is sometimes called a backside or memory bus.
- Accessing a chip's local memory whether for an I/O operation or for an access from another chip requires going through the chip that "owns" that memory.
- Thus, access to memory is asymmetric: faster to the local memory and slower to the remote memory.

- Symmetric shared-memory machines usually support the caching of both shared and private data.

- Private data are used by a single processor, while shared data are used by multiple processors, essentially providing communication among the processors through reads and writes of the shared data.

- When a private item is cached, its location is migrated to the cache, reducing the average access time as well as the memory bandwidth required. Since no other processor uses the data, the program behavior is identical to that in a uniprocessor.

- When shared data are cached, the shared value may be replicated in multiple caches.

Caching of shared data, however, introduces a new problem: cache coherence.

cache coherence:

A memory system is coherent if any read of a data item returns the most recently written value of that data item.

This simple definition contains two different aspects of memory system behavior, both of which are critical to writing correct shared-memory programs.

The first aspect, called coherence, defines what values can be returned by a read.

The second aspect, called consistency, determines when a written value will be returned by a read. Let’s look at coherence first.

A memory system is coherent if

1. A read by processor P to location X that follows a write by P to X, with no writes of X by another processor occurring between the write and the read by P, always returns the value written by P.

2. A read by a processor to location X that follows a write by another processor to X returns the written value if the read and write are sufficiently separated in time and no other writes to X occur between the two accesses.

3. Writes to the same location are serialized; that is, two writes to the same location by any two processors are seen in the same order by all processors. For example, if the values 1 and then 2 are written to a location, processors can never read the value of the location as 2 and then later read it as 1.

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

The cache coherence problem for a single memory location (X), read and written by two processors (A and B). We initially assume that neither cache contains the variable and that X has the value 1.

We also assume a write-through cache; a writeback cache adds some additional but similar complications. After the value of X has been written by A, A's cache and the memory both contain the new value, but B's cache does not, and if B reads the value of X it will receive 1!

Basic Schemes for Enforcing Coherence

- Snooping protocol
- Directory based protocol

Snooping protocol(Centralized memory)

Also called as write invalidate protocol because it invalidates other copies on a write. It is by far the most common protocol.

Exclusive access ensures that no other readable or writable copies of an item exist when the write occurs: All other cached copies of the item are invalidated.

Figure shows an example of an invalidation protocol with write-back caches in action.

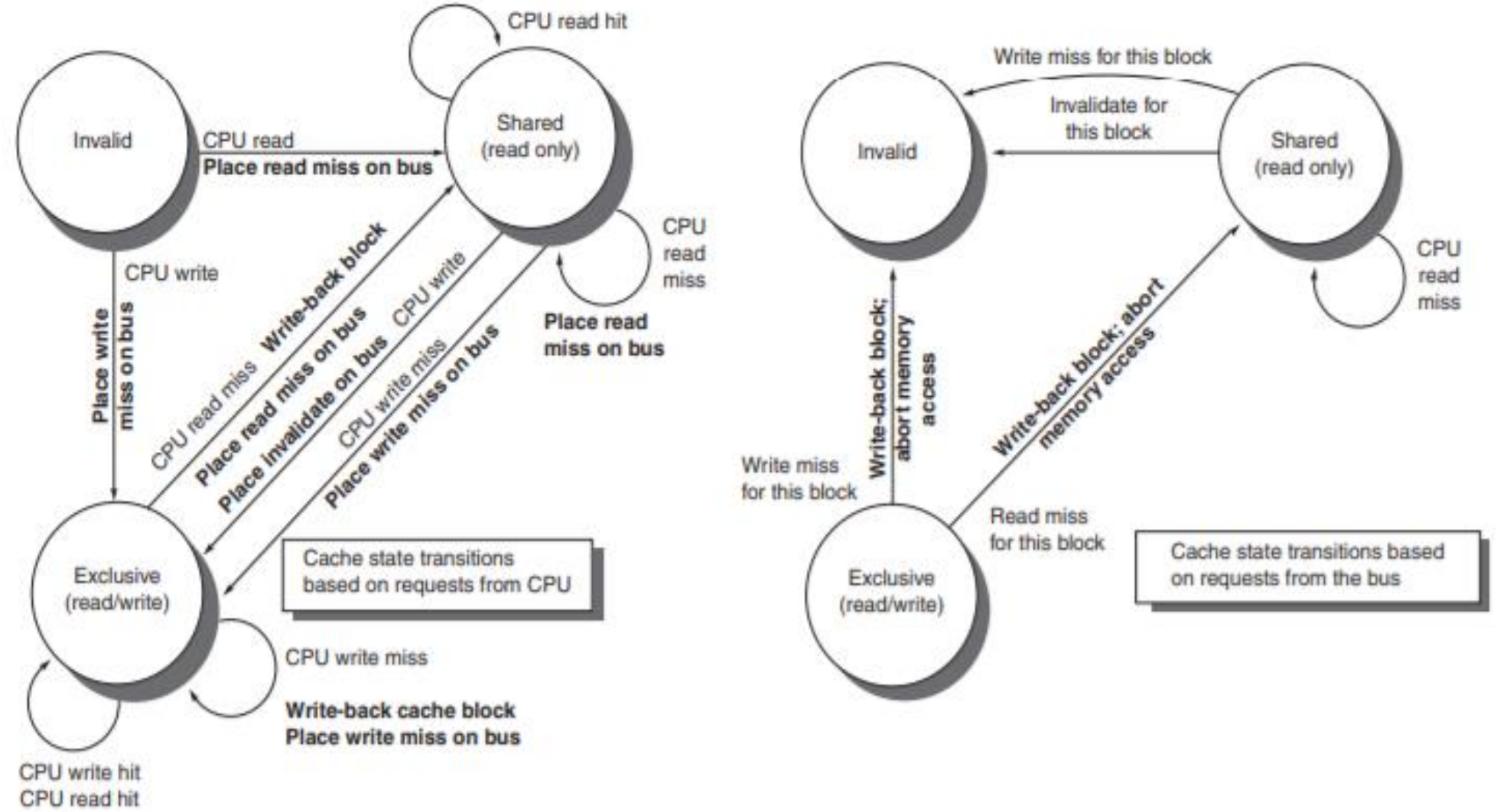
For example, consider a write followed by a read by another processor: Since the write requires exclusive access, any copy held by the reading processor must be invalidated (hence, the protocol name).

Thus, when the read occurs, it misses in the cache and is forced to fetch a new copy of the data. For a write, we require that the writing processor have exclusive access, preventing any other processor from being able to write simultaneously.

- If two processors do attempt to write the same data simultaneously, one of them wins the race ,causing the other processor's copy to be invalidated.
- For the other processor to complete its write, it must obtain a new copy of the data, which must now contain the updated value. Therefore, this protocol enforces write serialization.

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

Snooping protocol



Please refer the following link for the explanation

https://freevideolectures.com/course/4974/nptel-computer-architecture-course-sponsored-aricent/30

Directory based protocol(Distributed memory)

There are two primary operations that a directory protocol must implement:

- Handling a read miss
- Handling a write to a shared, clean cache block.

To implement these operations, a directory must track the state of each cache block. In a simple protocol, these states could be the following:

■ Shared—One or more nodes have the block cached, and the value in memory is up to date (as well as in all the caches).

■ Uncached—No node has a copy of the cache block.

■ Modified—Exactly one node has a copy of the cache block, and it has written the block, so the memory copy is out of date. The processor is called the owner of the block.

- In addition to tracking the state of each potentially shared memory block, we must track which nodes have copies of that block, since those copies will need to be invalidated on a write.
- The simplest way to do this is to keep a bit vector for each memory block. When the block is shared, each bit of the vector indicates whether the corresponding processor chip (which is likely a multicore) has a copy of that block.
- We can also use the bit vector to keep track of the owner of the block when the block is in the exclusive state. For efficiency reasons, we also track the state of each cache block at the individual caches.
- The states and transitions for the state machine at each cache are identical to what we used for the snooping cache, although the actions on a transition are slightly different.

Example

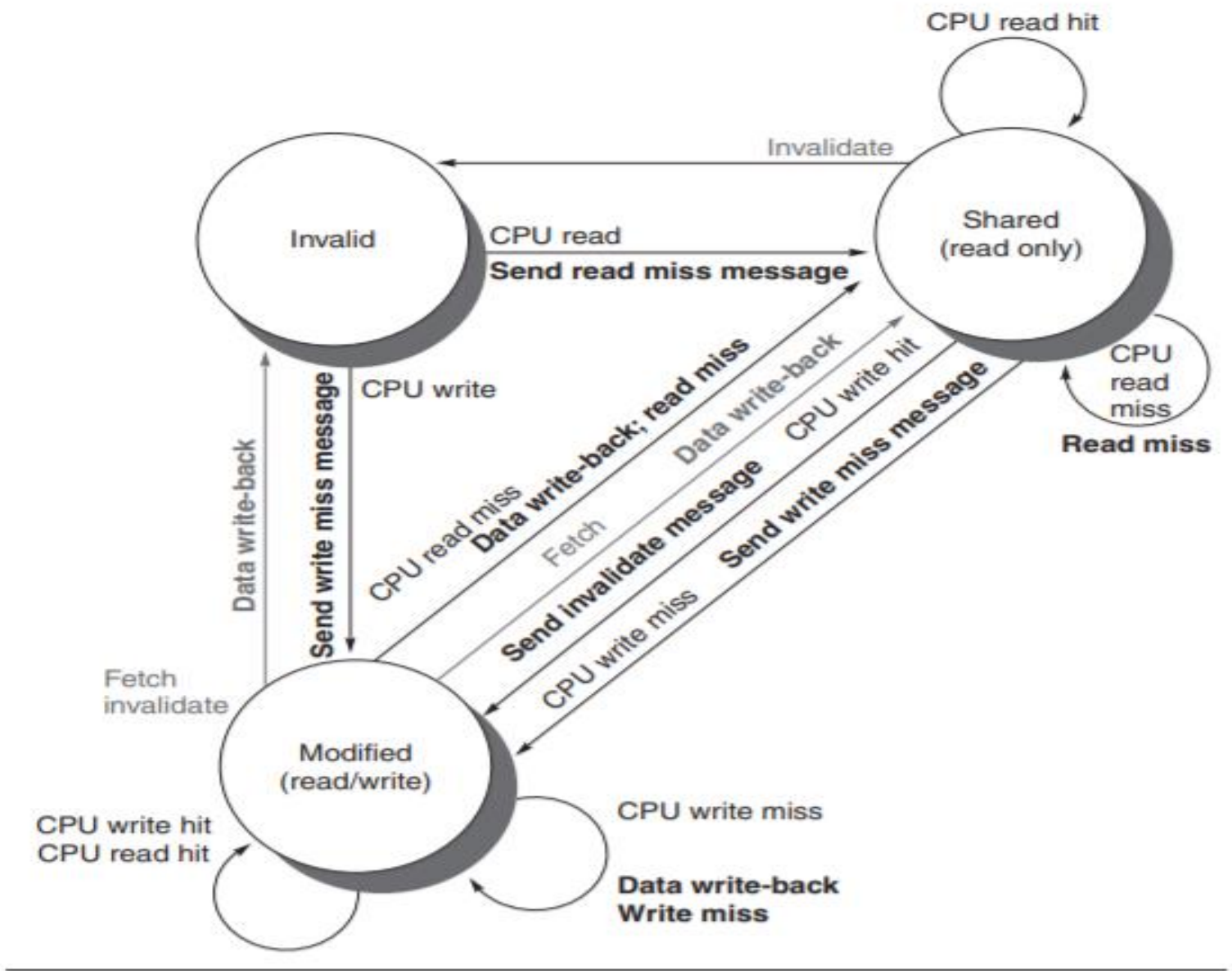
- For the table shows the types of messages sent among nodes. The local node is the node where a request originates. The home node is the node where the memory location and the directory entry of an address reside.
- The physical address space is statically distributed, so the node that contains the memory and directory for a given physical address is known. For example, the high-order bits may provide the node number, while the low-order bits provide the offset within the memory on that node.

- The local node may also be the home node. The directory must be accessed when the home node is the local node, since copies may exist in yet a third node, called a remote node.

- The possible messages sent among nodes to maintain coherence, along with the source and destination node, the contents (where P = requesting node number, A = requested address, and D = data contents), and the function of the message.

Message type	Source	Destination	Message contents	Function of this message
Read miss	Local cache	Home directory	P, A	Node P has a read miss at address A; request data and make P a read sharer.
Write miss	Local cache	Home directory	P, A	Node P has a write miss at address A; request data and make P the exclusive owner.
Invalidate	Local cache	Home directory	A	Request to send invalidates to all remote caches that are caching the block at address A.
Invalidate	Home directory	Remote cache	A	Invalidate a shared copy of data at address A.
Fetch	Home directory	Remote cache	A	Fetch the block at address A and send it to its home directory; change the state of A in the remote cache to shared.
Fetch/invalidate	Home directory	Remote cache	A	Fetch the block at address A and send it to its home directory; invalidate the block in the cache.
Data value reply	Home directory	Local cache	D	Return a data value from the home memory.
Data write-back	Remote cache	Home directory	A, D	Write-back a data value for address A.

- An Example Directory Protocol



For the explanation follow the below link.

https://www.youtube.com/watch?v=w5wJs9cFYI8

Prescribed textbook

Chapter 5(from Centralized approach onwards)
Computer Architecture A Quantitative Approach Fifth Edition by
John L. Hennessy and David A. Patterson