University Of Diyala
College Of Engineering
Computer Engineering Department

COMPUTER ARCHITECTURE II

PART 6: CACHE COHERENCE IN MULTIPROCESSOR SYSTEMS

Asst. Prof. Ahmed Salah Hameed

Second stage

2022-2023

From previous lectures

Cache Memory

- Very high-speed memory used to speed up and synchronizing with high-speed CPU.
- The cache is the fastest memory type that works as a buffer between RAM and the CPU.
- The cache is a smaller and faster memory which stores copies of the data from frequently used main memory locations.

Multiprocessors

- Tightly coupled or Shared- Memory System
- Loosely Coupled: or Distributed-Memory System

Cache Coherence

- The primary advantage of cache is its ability to reduce the average access time in uniprocessors.

Read procedures

- When the processor finds a word in cache during a read operation, the main memory is not involved in the transfer.
- When the word is not available, one of the block mapping procedures with memory is used.

Write procedures

- ***write-through policy*** :both cache and main memory are updated with every write operation.
- ***write-back policy***, only the cache is updated and the location is marked so that it can be copied later into main memory.

- What is **cache coherence** ?
- Cache coherence problems exist in multiprocessors with private caches because of the need to share writable data. Read-only data can safely be replicated without cache coherence enforcement mechanisms.

Cache Coherence

- The problem that processors may see different values through their 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

Solutions to Cache Coherence Problem

1) Software Level solutions (Compiler-based cache coherence)

- Perform an analysis on the code to determine which data items may become unsafe for caching, and they mark those items accordingly. So, there are some more cacheable items, and the operating system or hardware does not cache those items.

- Prevent any shared data variables from being cached.

- More efficient approaches analyze the code to determine safe periods for shared variables. The compiler then inserts instructions into the generated code to enforce cache coherence during the critical periods.

Solutions to Cache Coherence Problem

2) Hardware Level solutions

- Hardware solution provide dynamic recognition at run time of potential inconsistency conditions. Because the problem is only dealt with when it actually arises, there is more effective use of caches, leading to improved performances over a software approach.

- Cache coherence protocols
- **Snoopy Protocols**

  Sharing status of each block kept in one location
- **Directory protocols**

  Each core tracks sharing status of each block

Snoopy Coherence Protocols

1- Write invalidate protocol (Used in modern microprocessors)

- On write, invalidate all other copies

2- Write update/write broadcast protocol

- On write, update all copies
- It consumes considerably more bandwidth.

Write invalidate (write through)

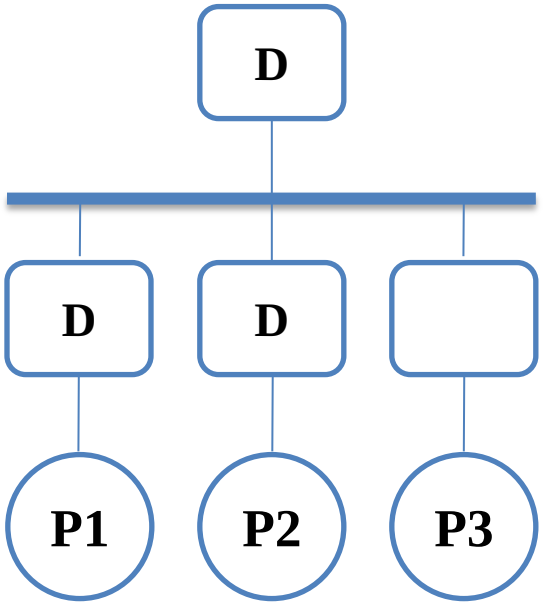
Write invalidate (write back)

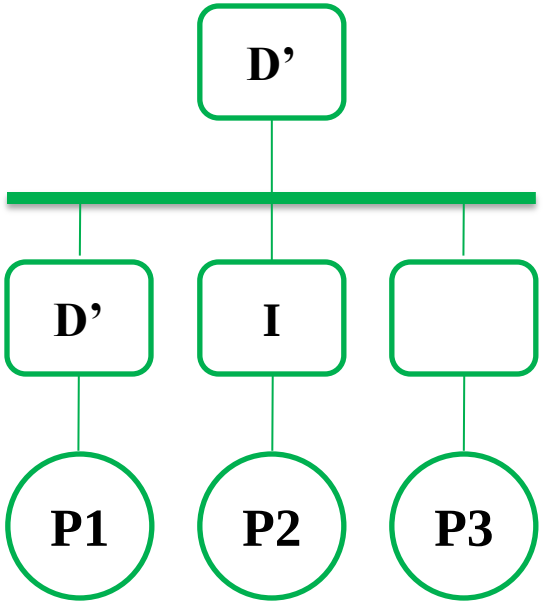
Write update (write through)

Write update (write back)

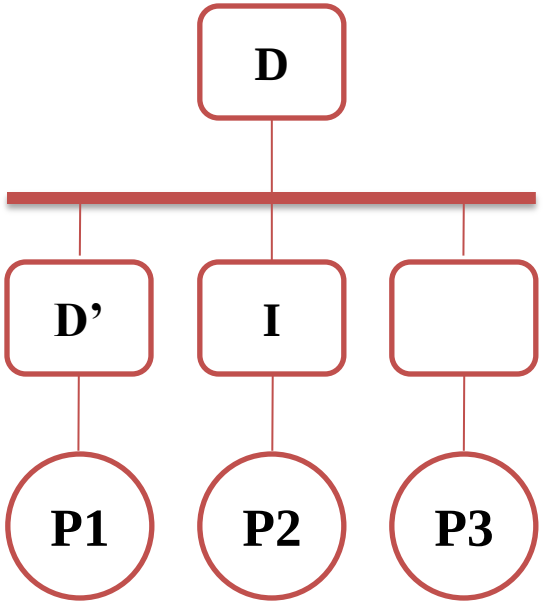
Write invalidate protocol

P1 update D to D'





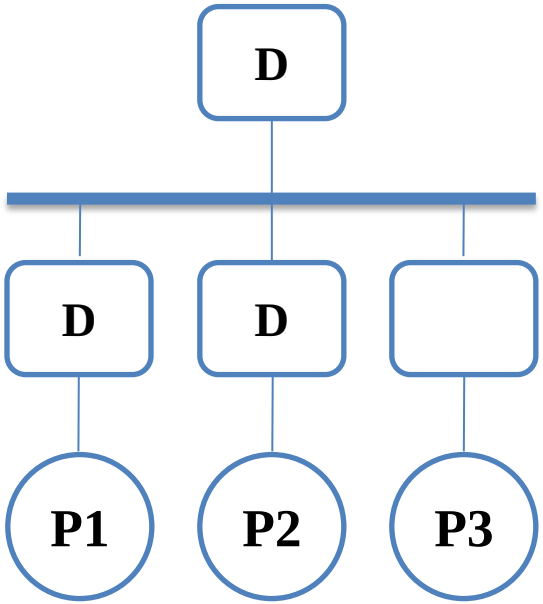
Write invalidate (write through)

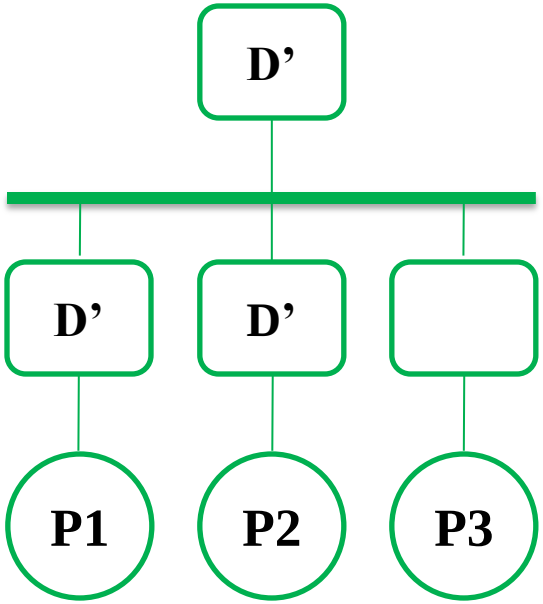


Write invalidate (write back)

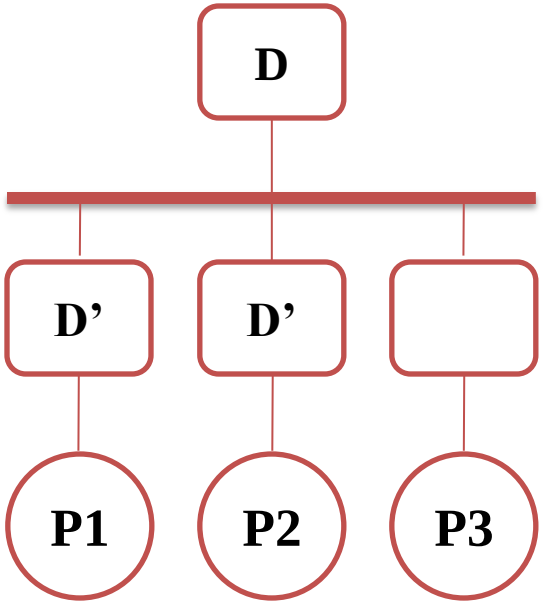
Write update/write broadcast protocol

P1 update D to D'





Write update (write through)



Write update (write back)

Directory protocols

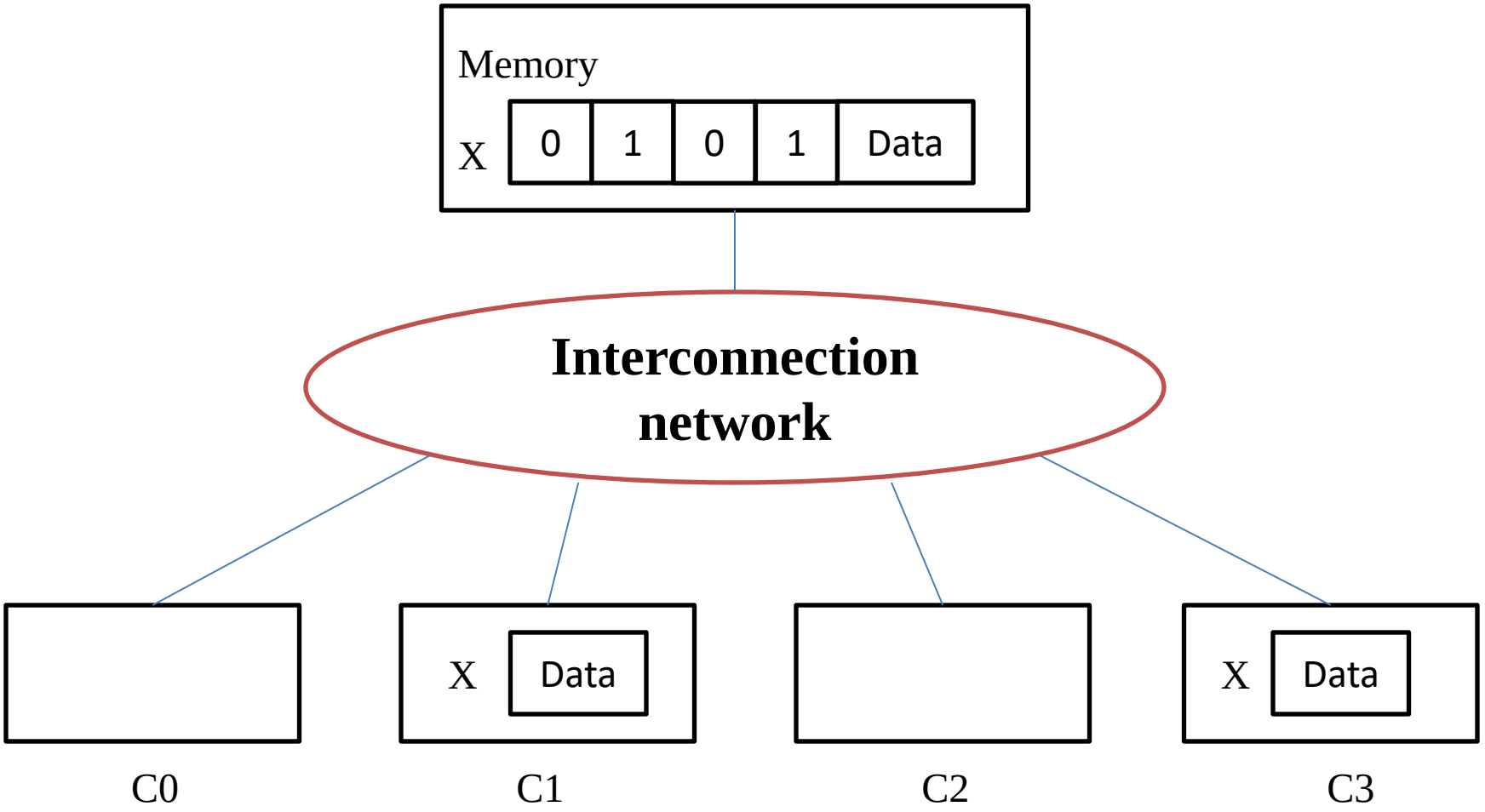
- Centralized controller is used as part of the main memory controller, and a directory that is stored in main memory.
- The directory contains global state information about the contents of the various local caches.
- When an individual cache controller makes a request, the centralized controller checks and issues necessary commands for data transfer between memory and caches or between caches themselves.

Directory protocols

❑ Full-map Directories

- Each directory entry contains N pointers, where N is the number of processors.
- There could be N cached copies of a particular block shared by all processors.
- For every memory block, an N bit vector is maintained, where N equals the number of processors in the shared memory system. Each bit in the vector corresponds to one processor.

Full-map Directories

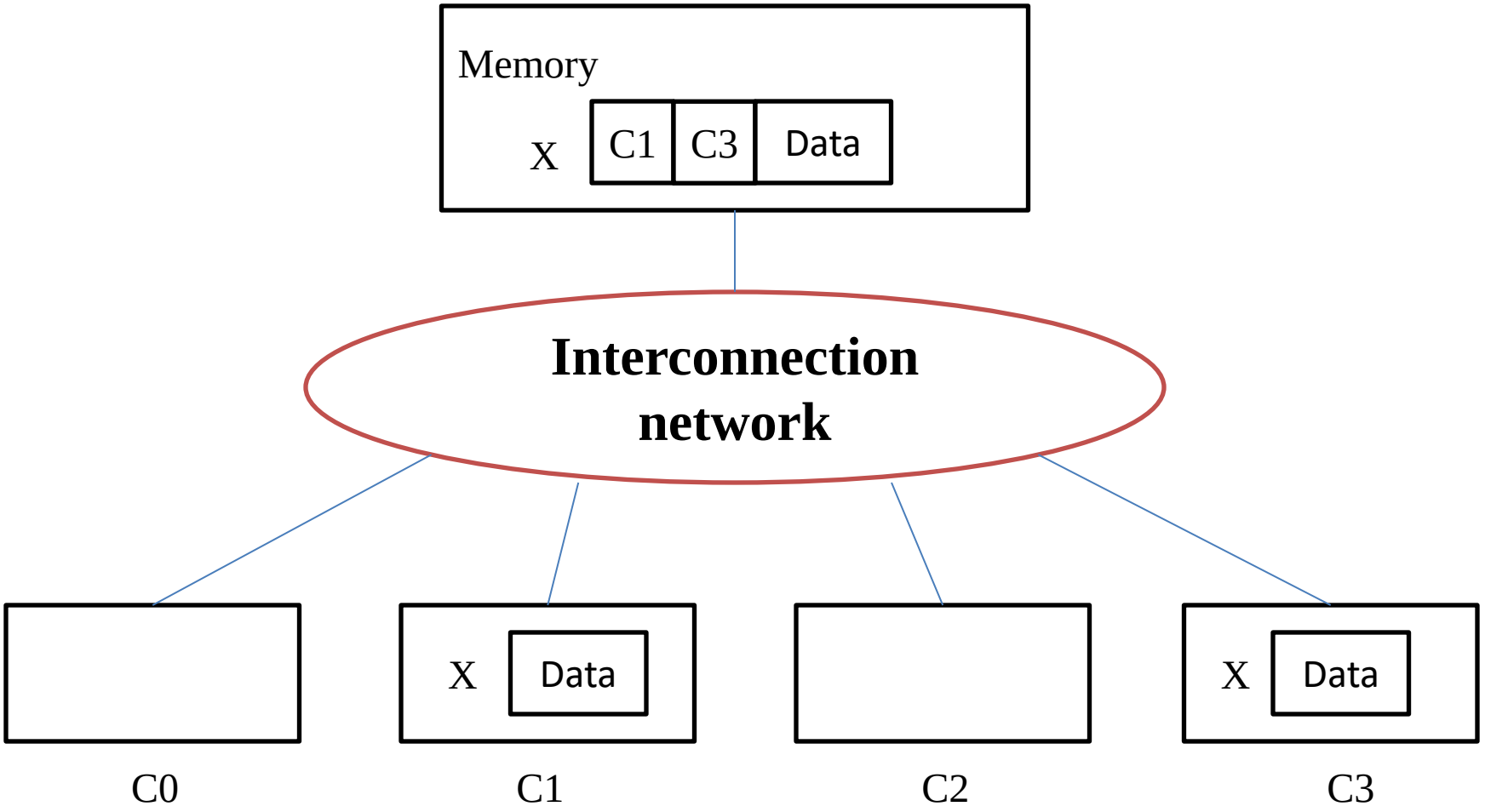


Directory protocols

❑ Limited Directories

- Fixed number of pointers per directory entry regardless of the number of processors.
- Restricting the number of simultaneously cached copies of any block should solve the directory size problem that might exist in full-map directories.

Limited Directories



Directory protocols

❑ Chained Directories

- Chained directories emulate full-map by distributing the directory among the caches.
- Solving the directory size problem without restricting the number of shared block copies.
- Chained directories keep track of shared copies of a particular block by maintaining a chain of directory pointers.

Chained Directories