**National Institute of Technology Karnataka Surathkal**
**Department of Information Technology**

# IT 301 Parallel Computing

## Memory Models and Cache Coherence

**Dr. Geetha V**

*Assistant Professor*

*Dept of Information Technology*

*NITK Surathkal*

# Course Outline

**Course Plan: Theory:**

**Part A: Parallel Computer Architectures**

Week 1,2,3: ***Introduction to Parallel Computer Architecture:*** Parallel Computing, Parallel architecture, bit level, instruction level , data level and task level parallelism. Instruction level parelllelism: pipelining(Data and control instructions), scalar and superscalar processors, vector processors. Parallel computers and computation.

Week 4,5: Memory Models: UMA, NUMA and COMA. Flynns classification, Cache coherence,

Week 6,7: Amdahl's Law. Performance evaluation, Designing parallel algorithms : Divide and conquer, Load balancing, Pipelining.

Week 8 -11: ***Parallel Programming techniques like Task Parallelism using TBB, TL2, Cilk++ etc. and software transactional memory techniques.***

# Course Outline

**Part B: OpenMP/MPI/CUDA**

Week 1,2,3 : ***Shared Memory Programing Techniques:*** Introduction to OpenMP : Directives: parallel, for, sections, task, single, critical, barrier, taskwait, atomic. Clauses: private, shared, firstprivate, lastprivate, reduction, nowait, ordered, schedule, collapse, num_threads, shared, if().

Week 4,5: ***Distributed Memory programming Techniques:*** MPI: Blocking, Non-blocking.

Week 6,7 : CUDA : OpenCL, Execution models, GPU memory, GPU libraries.

Week 10,11,: ***Introduction to accelerator programming using CUDA/OpenCL and Xeon-phi. Concepts of Heterogeneous programming techniques.***

**Practical:**

Implementation of parallel programs using OpenMP/MPI/CUDA.

**Assignment:** Performance evaluation of parallel algorithms (in group of 2 or 3 members)

# Index

1. Memory models
2. Cache Coherence
    1. Cache coherence problem
    2. Snooping Bus Protocols: MSI.MESI

# 2.Memory Models

## Parallel Computers Architectural Model/ Physical Model

Distinguished by having-

**1. Shared Common Memory:**

Three Shared-Memory Multiprocessor Models are:

i. UMA (Uniform-Memory Access)

ii. NUMA (Non-Uniform-Memory Access)

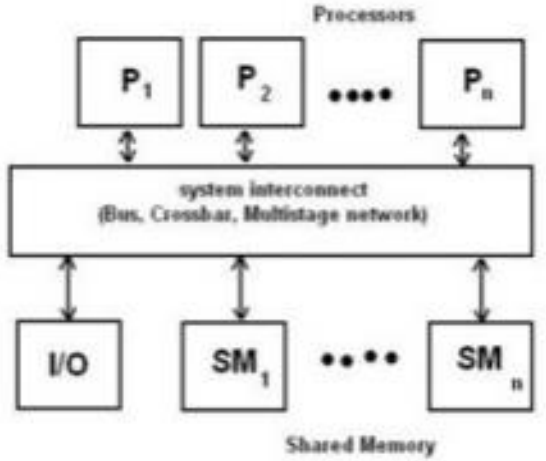
iii. COMA (Cache-Only Memory Architecture)

**2. Unshared Distributed Memory**

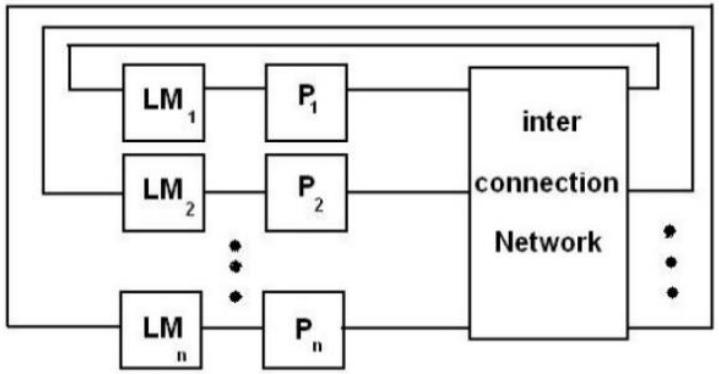
i. CC-NUMA (Cache-Coherent -NUMA)

# Memory Models: UMA, NUMA and COMA

Uniform Memory Access



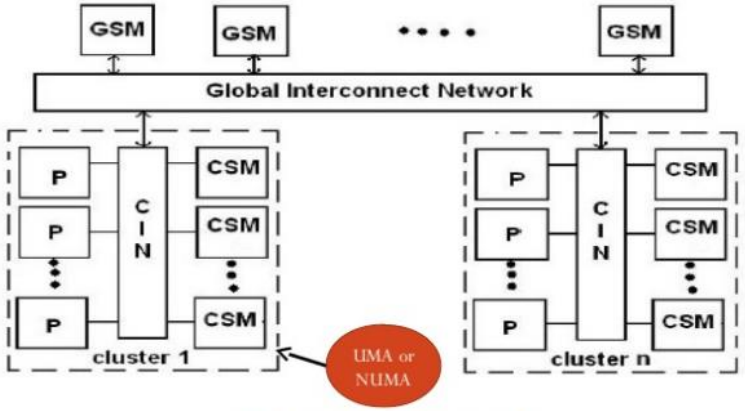

Non Uniform Memory Access




(a) Shared local memories

LM – Local Memory
P - Local Processor

Non Uniform Memory Access

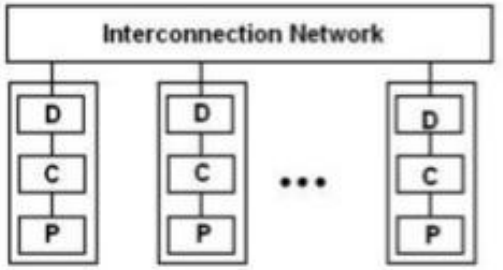



(b) A hierarchical cluster model

(Access of Remote Memory)

P – Processor
CSM – Cluster Shared Memory
CIN – Cluster Interconnection Network
GSM – Global Shared Memory

Cache Only Memory Access



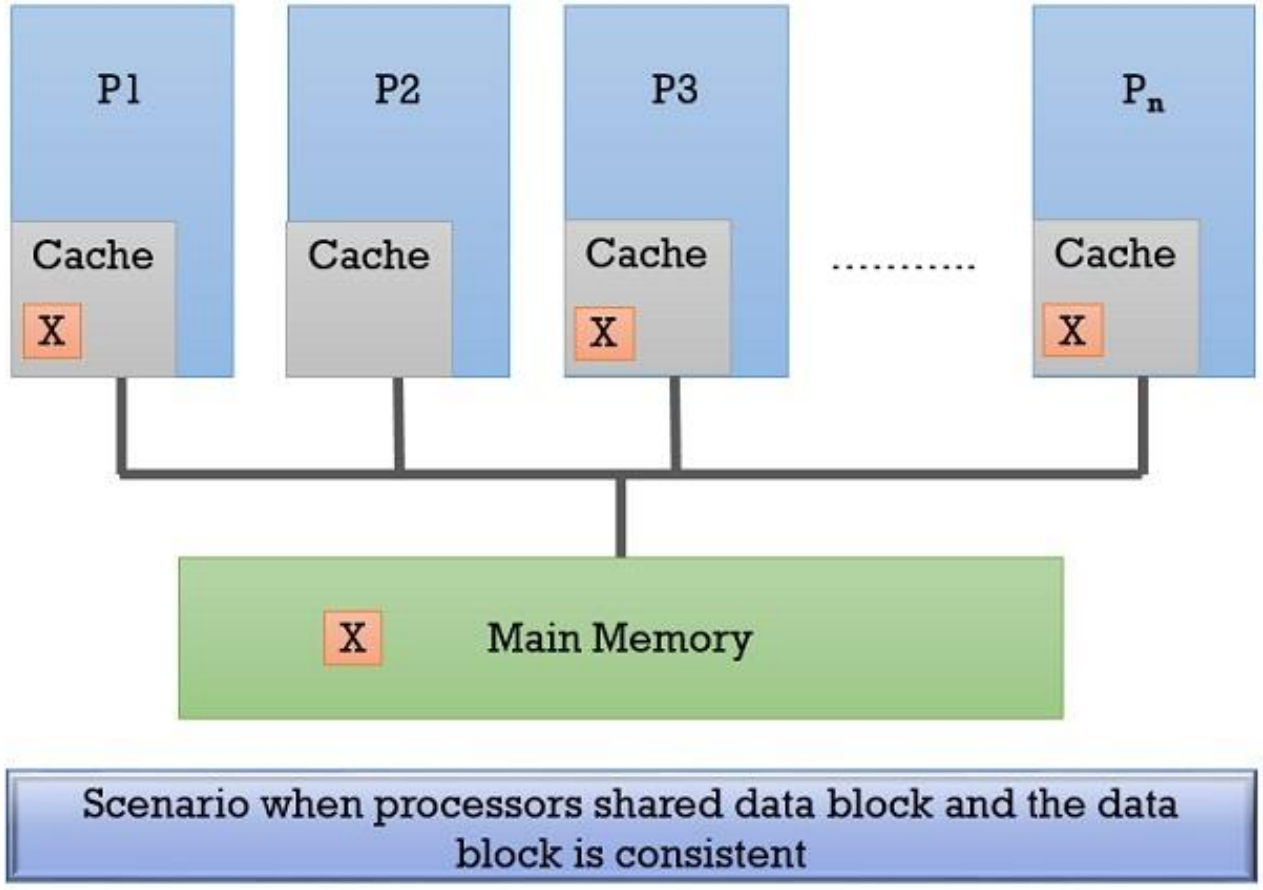

C – Cache
P – Processor
D - Directories

• Distributed Main Memory converted to Cache

•Cache form Global Address Space

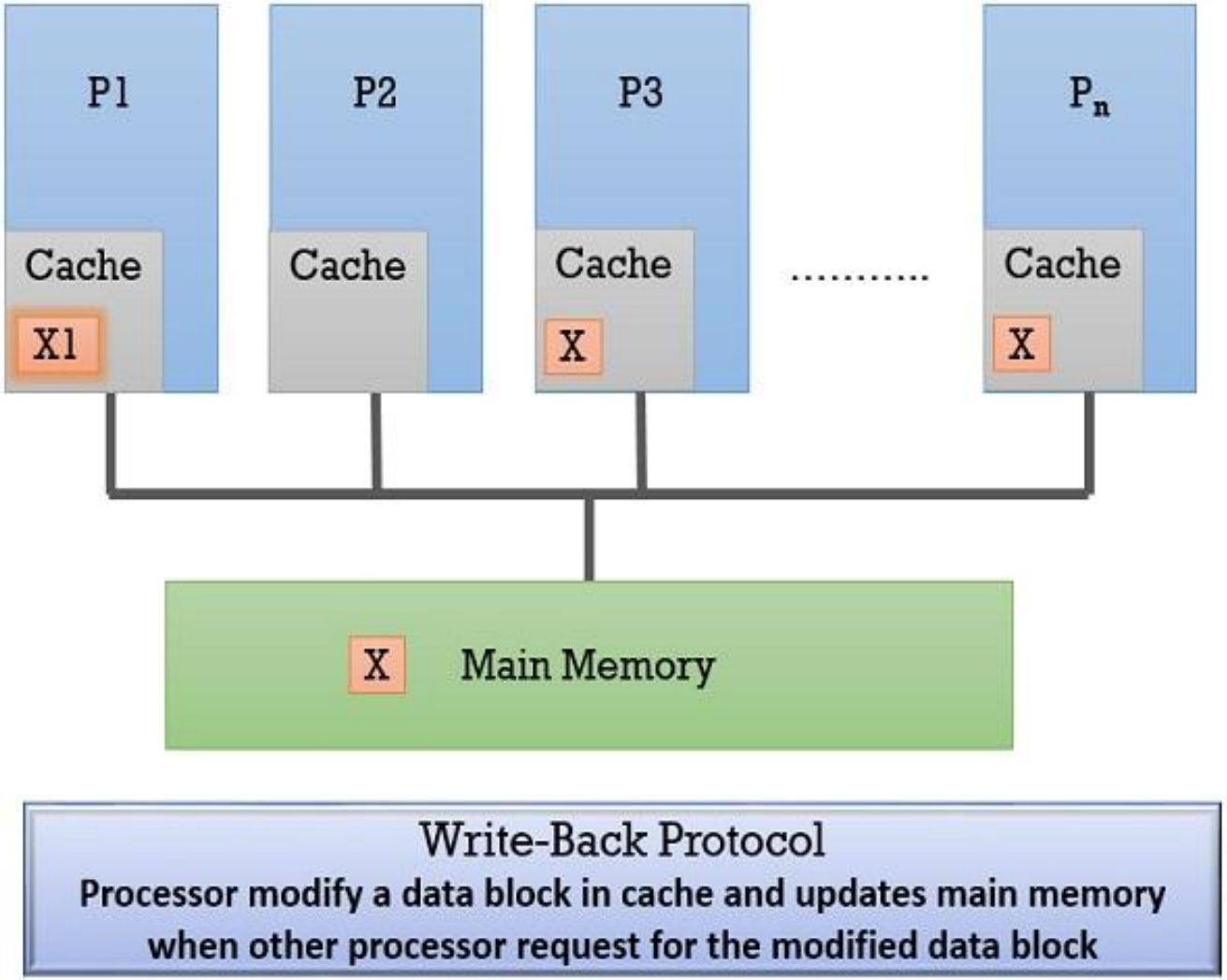
•Remote Cache access by – Distributed cache Directories

# Shared Memory Model



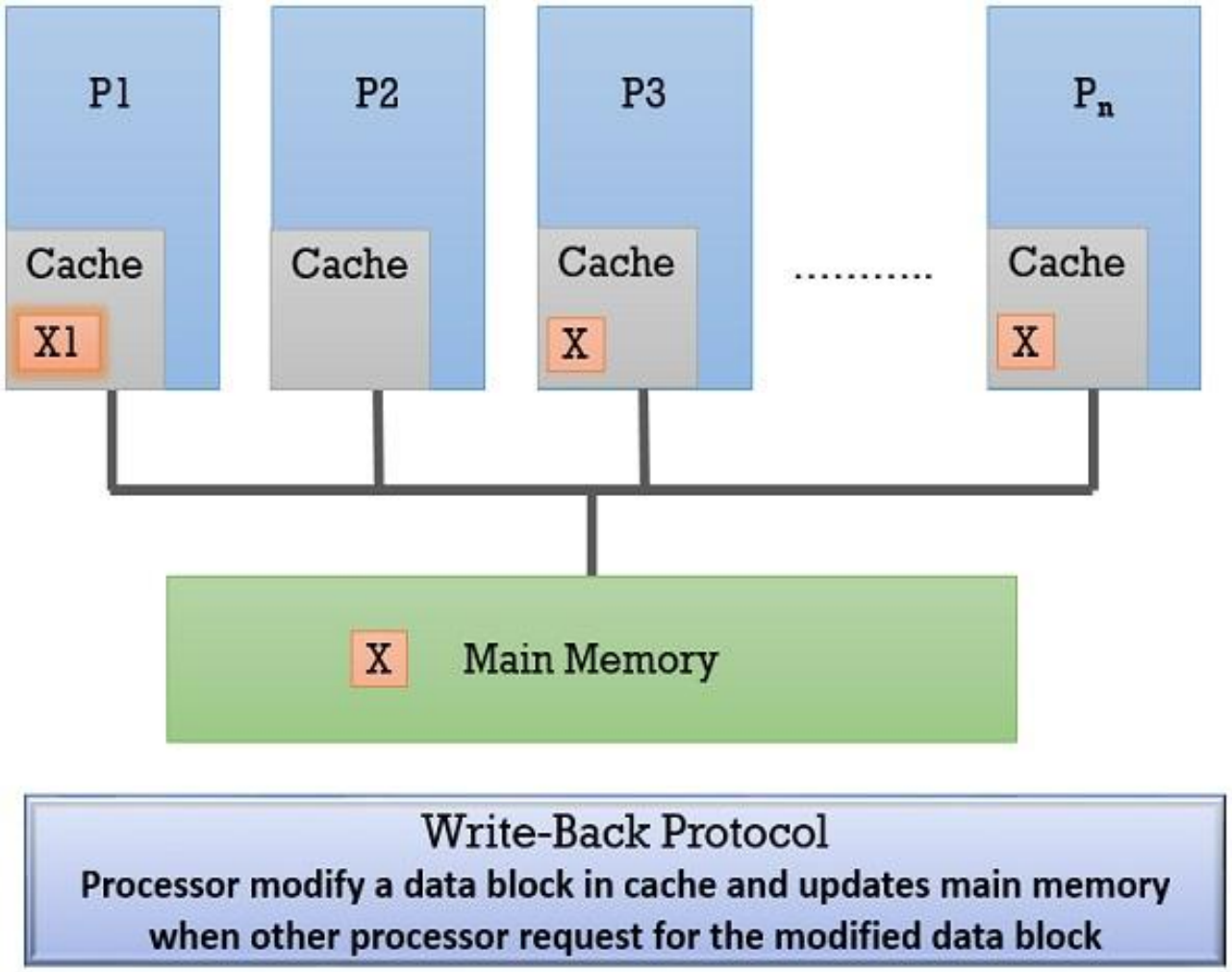

- Main memory is shared among all the processors.
- Accessing main memory every time is time consuming
- Better to have private cache for all the processors
- Shared memory consistency must be maintained.
- When the data block is shared among all other processors, the change in one cache must be reflected in all other processor cache. Then, the system can be considered as memory consistency is maintained.

# Shared Memory Model




- Cache Coherent Problem:
- Caching of shared data, however, introduces **Cache Coherence problem.**
- **Shared data can have different data in different cache.**
- This must be handled properly

# Shared Memory Model




- Cache Coherent: A memory system is coherent if any read of a data items return the most recently written value of that data item .

# Shared Memory Model

- A memory system is consistent if the following hold good:

1. A read by a processor P to a 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. **: Maintains program order**

**Assume X=5;**

**N=Read X; Write X, 10; N=Read X : Here, P must get value 10 , 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. **: Coherent rule of memory**

**Assume X=5;**

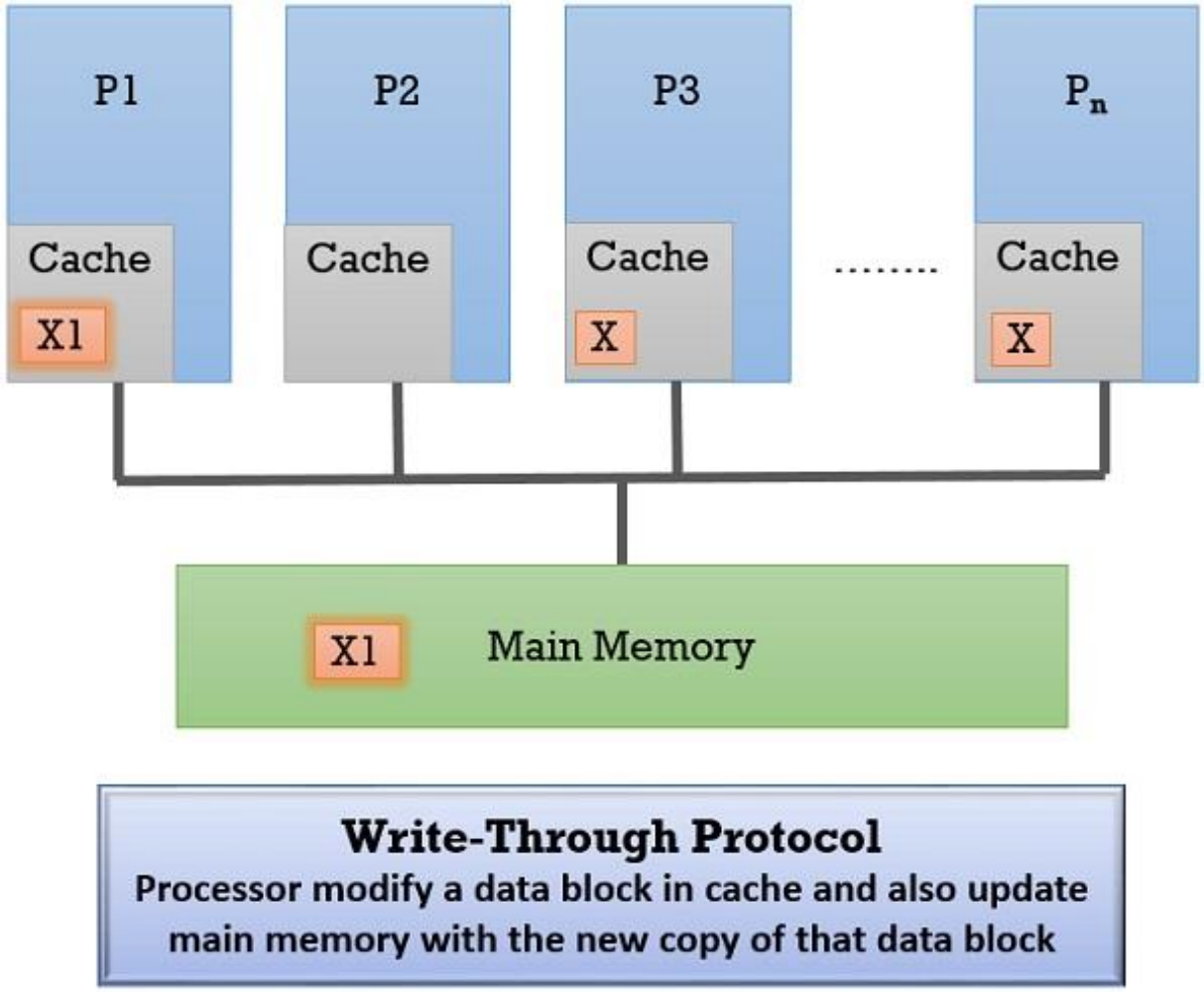
**N=Read X by P1; Write X, 10 by P2; N=Read X by P1 : Here, must get value 10 , written by P2**

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. This ensures that we do not see the older value after the newer value. **: Serialization of memory operations**

**Assume X=5;**

**Write X, 10 by P1; Write X, 15 by P2; N=Read X by P1 : Here, must get value 15 , written by P2**

# Cache Coherence Protocols

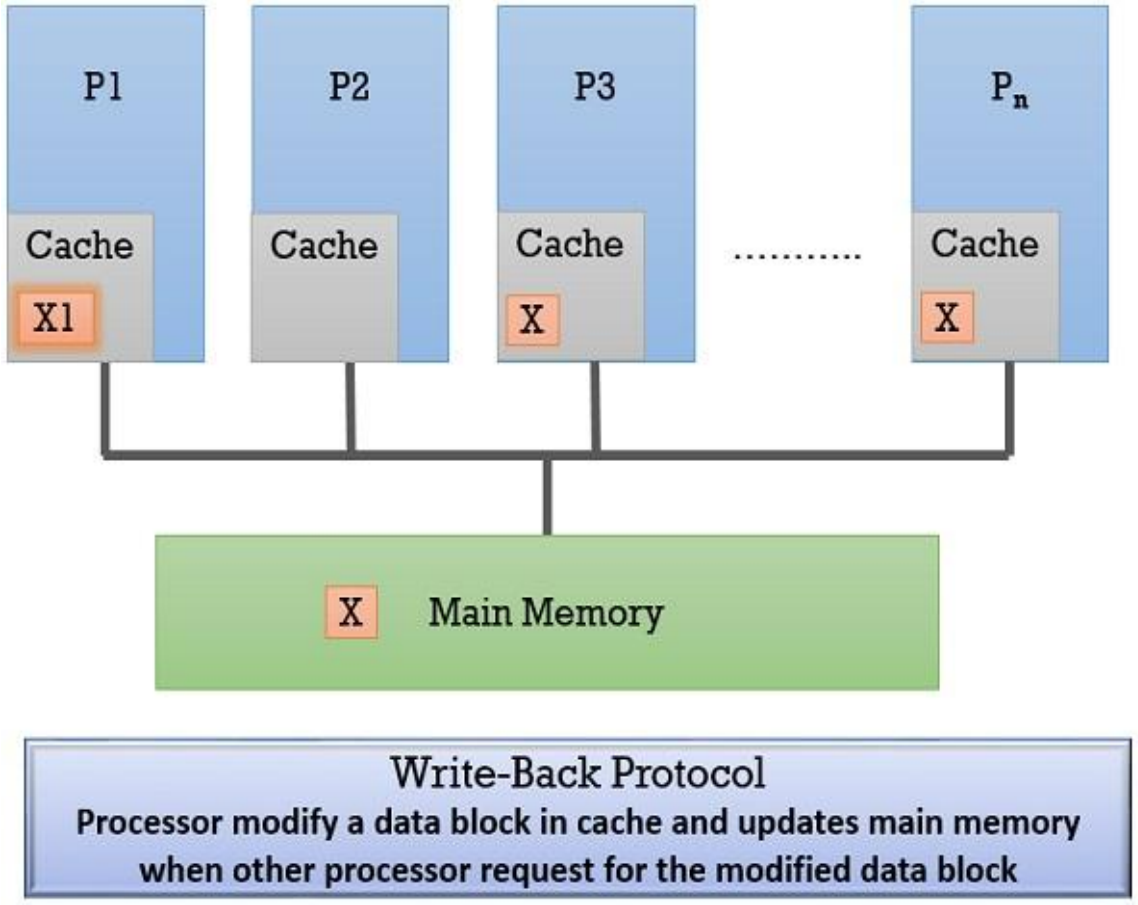



- **Memory Write Operations**

## 1. Write Through:

- When a processor modifies a data block in its cache, it immediately updates the main memory with the new copy of the same data block.
- Main memory has consistent data.
- Cache coherence : Either through update or invalidate

# Cache Coherence Protocols

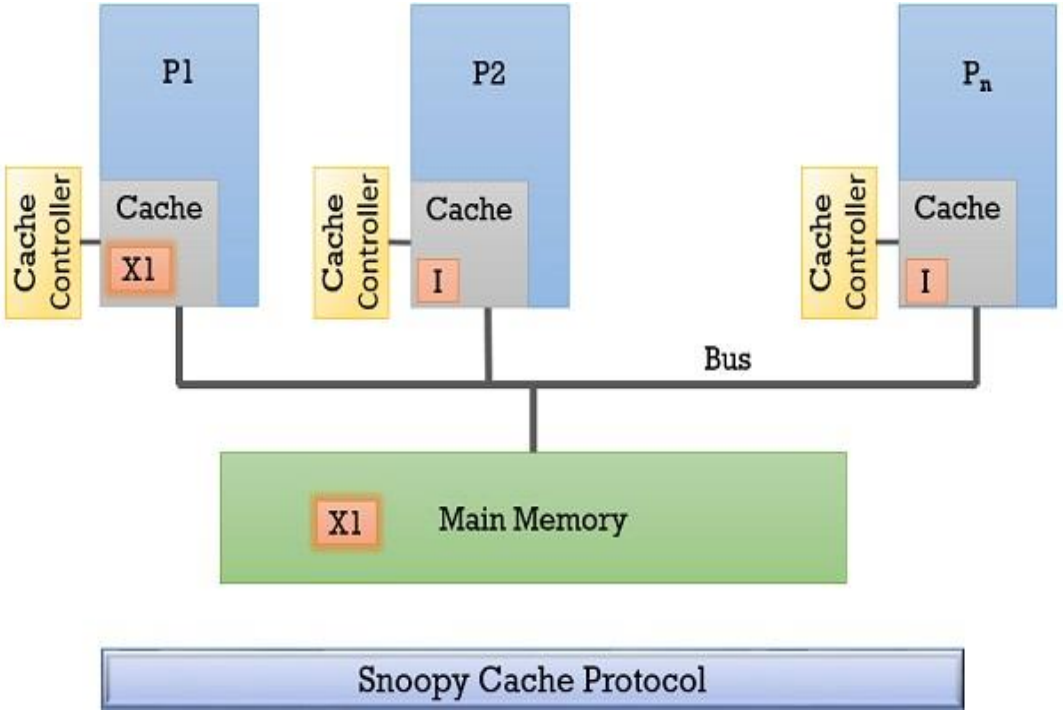



- **Memory Write Operations**

**1. Write Back:**

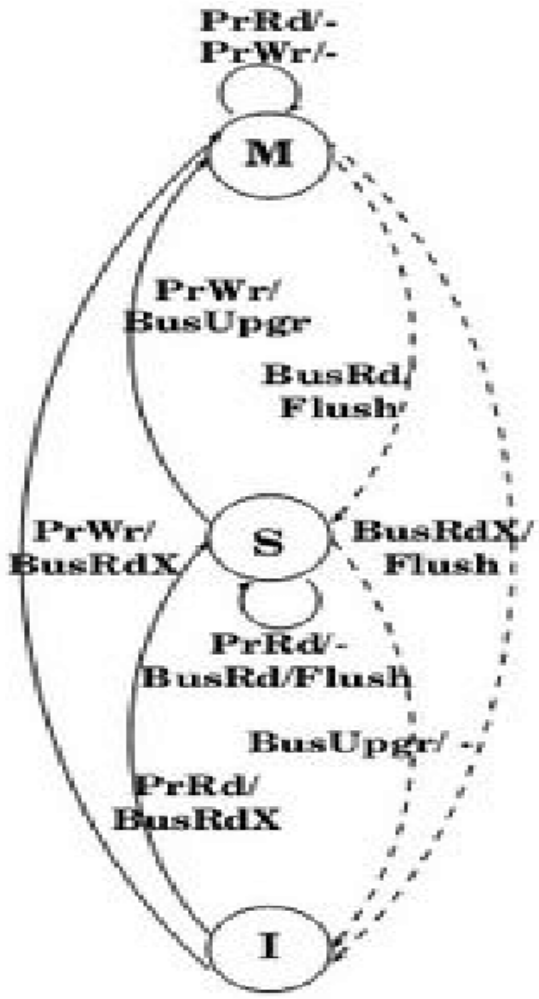
- When a processor modifies a data block in its cache, it **does not** immediately update the main memory with the new copy of the same data block. Write back durig cache replacement or request from other processors.
- That processor has the data exclusively
- **Cache coherence : invalidate all other cache entries and update when there is request.**

# Cache Coherence Protocols : Snoopy Bus Protocol

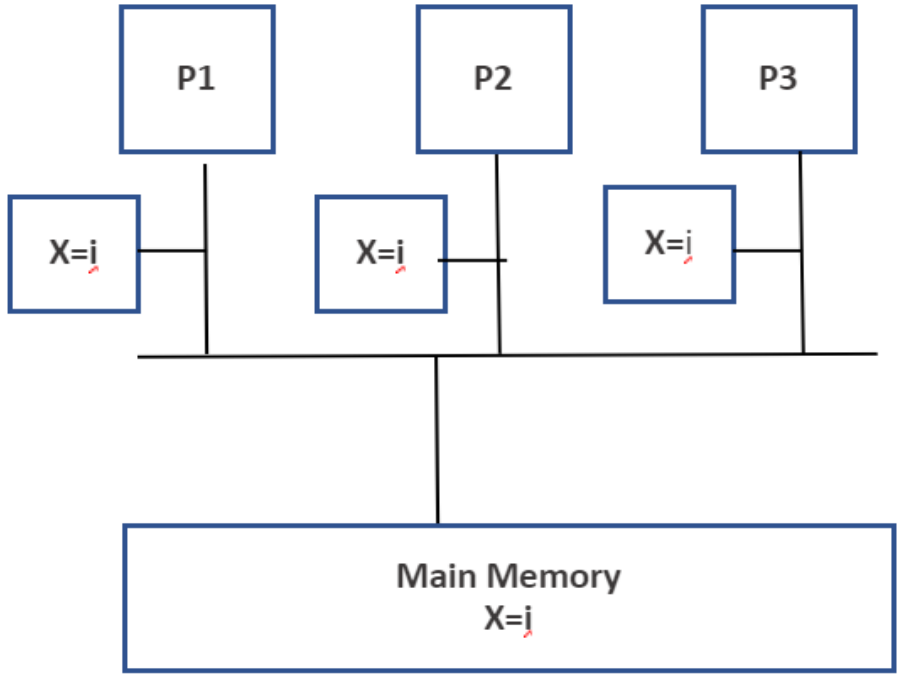



- **Snoopy Bus Protocol**
- Main memory is shared among processors via single bus.
- Cache controller performs snoop operation on all transaction over bus.
- Snoop bus protocol is hardware solution.
- It is used for small multiprocessor environment

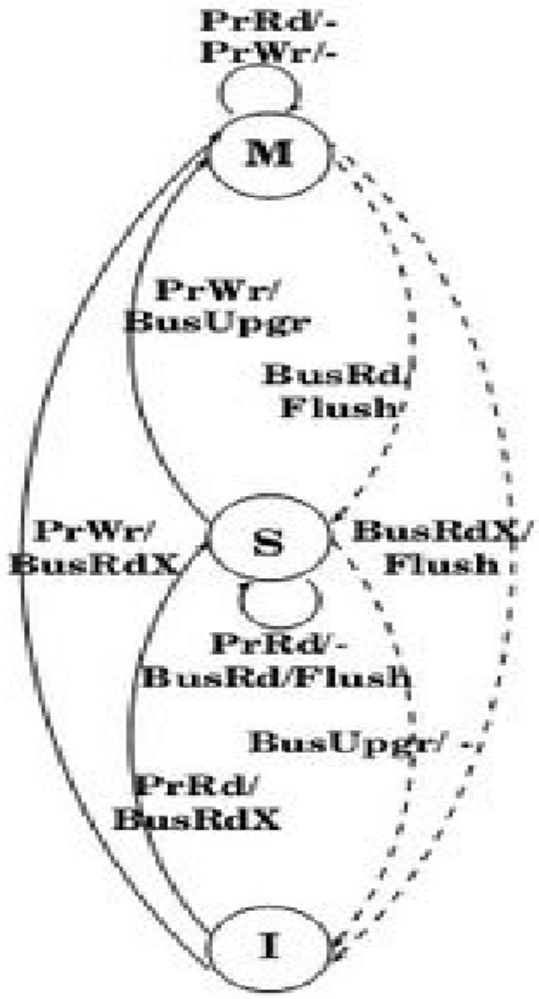
# Snoopy Bus Protocol MSI:Modified-Shared-Invalid




- **Snoopy Bus Protocol**
- Invalid State: Initially all the processor cache elements are invalid.

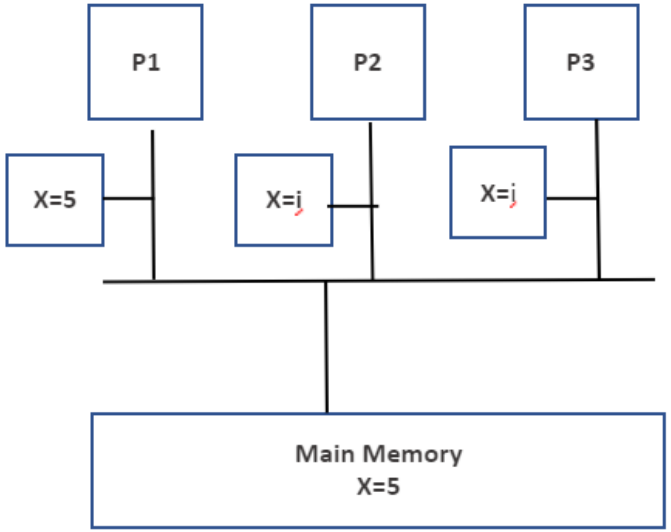


# Snoopy Bus Protocol MSI:Modified-Shared-Invalid

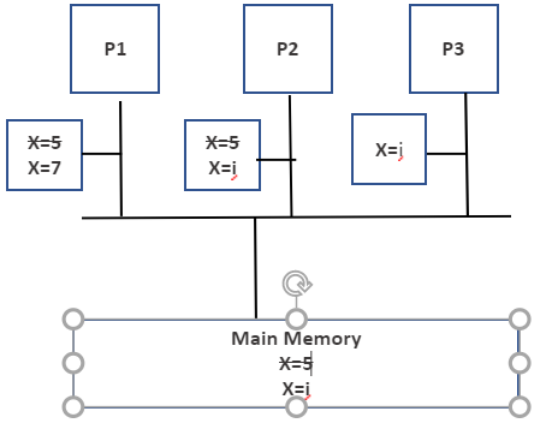



- **Snoopy Bus Protocol**
- Shared State: Initially all the processor cache elements are invalid.

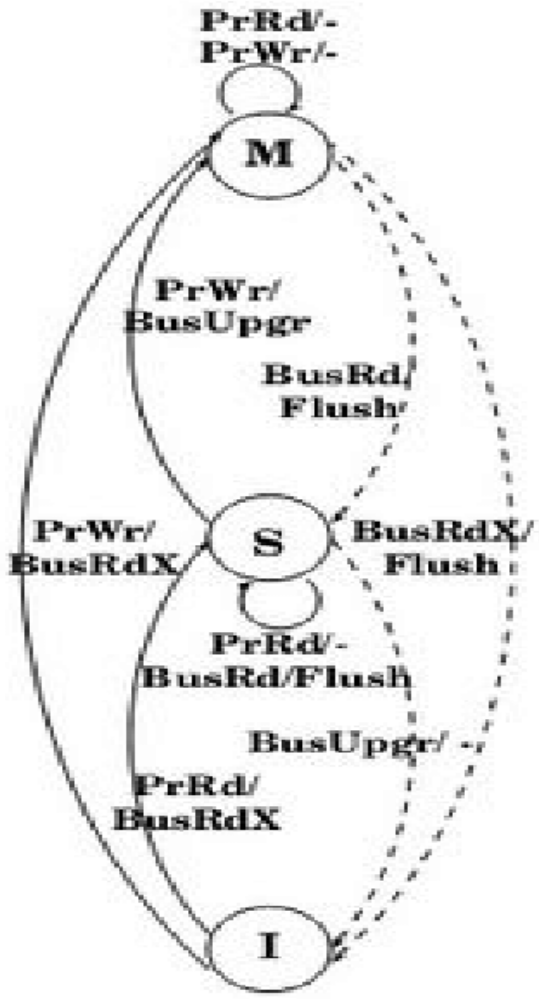
I-S: PrRd/BusRdx: P1 sending read




S-M: PrWr/BusUpgr : P1 in shared state, send PrWr

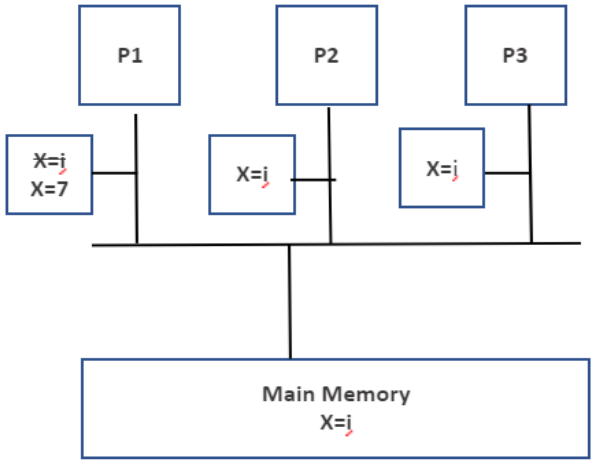


# Snoopy Bus Protocol MSI:Modified-Shared-Invalid

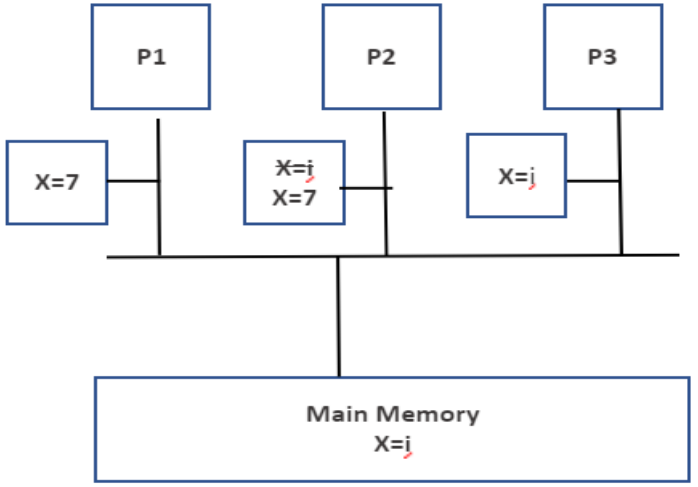



- **Snoopy Bus Protocol**
- Modified State: Initially all the processor cache elements are invalid.

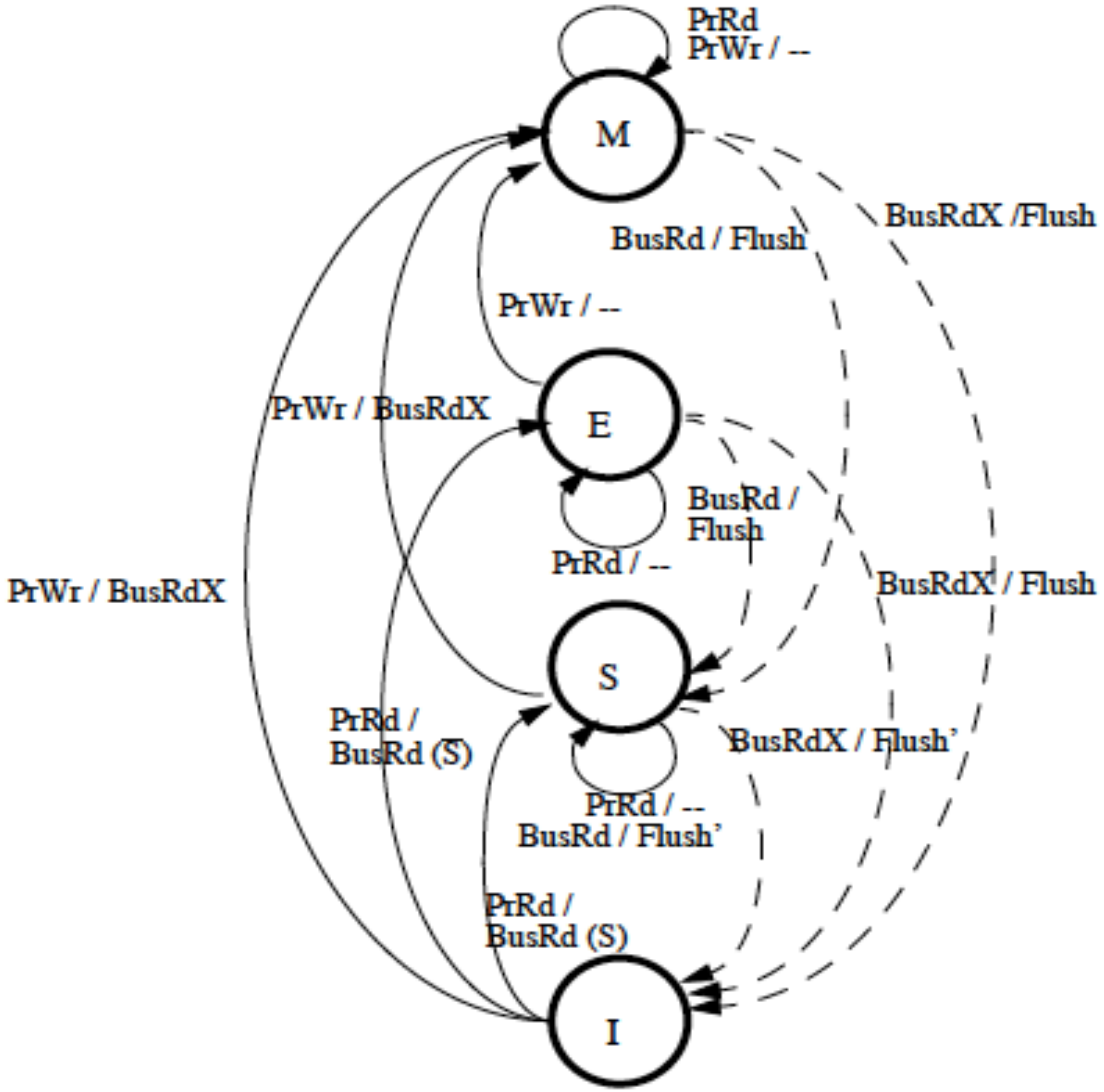
I-M: PrWr/BusRdx: P1 sending Write




M-S: BusRd: P2 sending PrRd



# Snoopy Bus Protocol MESI:Modified-Exclusive-Shared-Invalid




- **Snoopy Bus Protocol: MESI**
- Invalid State: The data items are invalid initially.
- Shared state: Valid copy of data in many cache
- Exclusive: The processor wants to perform write operation, and it has exclusive copy. **Invalid to Excusive**
- Modified State: Data block is modified, and the processor has modified data. Shared to Modified.

# Summary

1. Memory models
2. Cache Coherence
    1. Cache coherence problem
    2. Snooping Bus Protocols: MSI.MESI

# Reference

**Text Books and/or Reference Books:**


1. Professional CUDA C Programming – John Cheng, Max Grossman, Ty McKercher, 2014
2. B.Wilkinson, M.Allen, ”Parallel Programming: Techniques and Applications Using Networked Workstations and Parallel Computers”, Pearson Education, 1999
3. I.Foster, ”Designing and building parallel programs”, 2003
4. Parallel Programming in C using OpenMP and MPI – Micheal J Quinn, 2004
5. Introduction to Parallel Programming – Peter S Pacheco, Morgan Kaufmann Publishers, 2011
6. Advanced Computer Architectures: A design approach, Dezso Sima, Terence Fountain, Peter Kacsuk, 2002
7. Parallel Computer Architecture : A hardware/Software Approach, David E Culler, Jaswinder Pal Singh Anoop Gupta, 2011
8. Introduction to Parallel Computing, Ananth Grama, Anshul Gupta, George Karypis, Vipin Kumar, Pearson, 2011

# Thank You