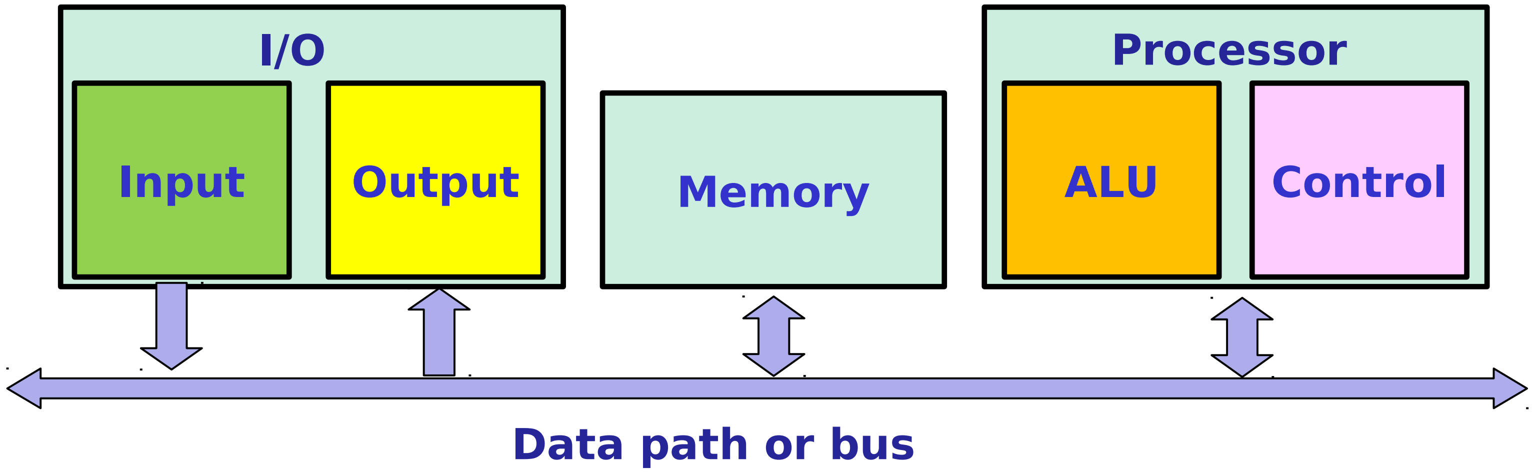
Input/Output (I/O) Subsystem



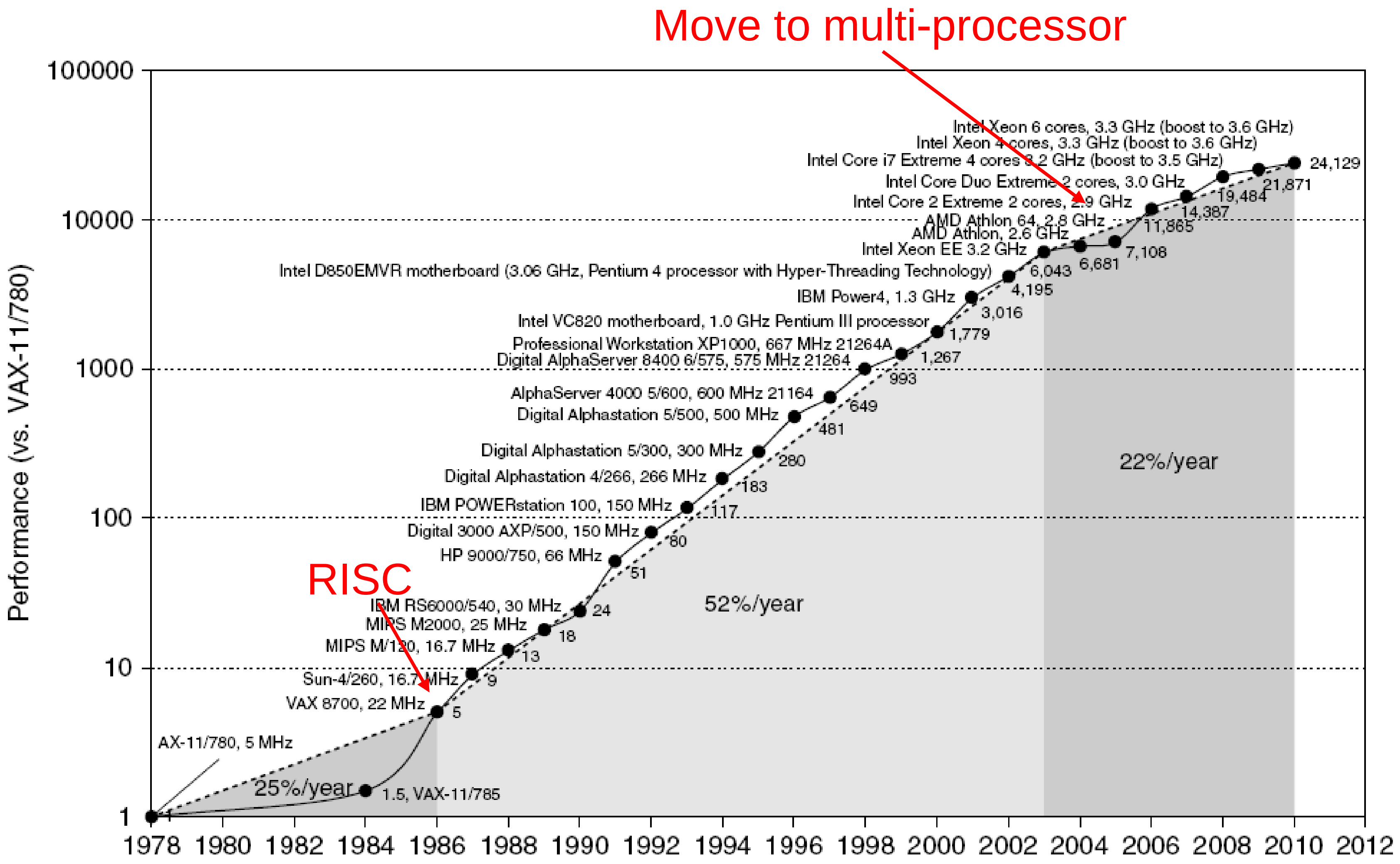
Computer Technology

Performance improvements:

- Improvements in semiconductor technology
  - Feature size, clock speed
- Improvements in computer architectures
  - Enabled by HLL compilers, UNIX
  - Lead to RISC architectures

- Together have enabled:
  - Lightweight computers
  - Productivity-based managed/interpreted programming languages

Single Processor Performance



Parallelism

Classes of parallelism in applications:

- Data-Level Parallelism (DLP)
- Task-Level Parallelism (TLP)

Classes of architectural parallelism:

- Instruction-Level Parallelism (ILP)
- Vector architectures/Graphic Processor Units (GPUs)
- Thread-Level Parallelism
- Request-Level Parallelism

Flynn’s Taxonomy

Single instruction stream, single data stream (SISD)

Single instruction stream, multiple data streams (SIMD)

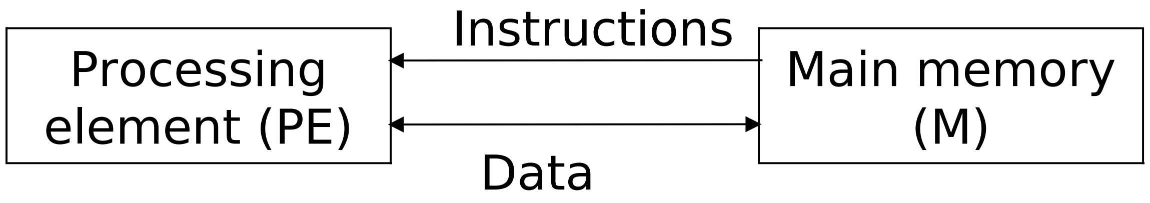
- Vector architectures
- Multimedia extensions
- Graphics processor units

Multiple instruction streams, single data stream (MISD)

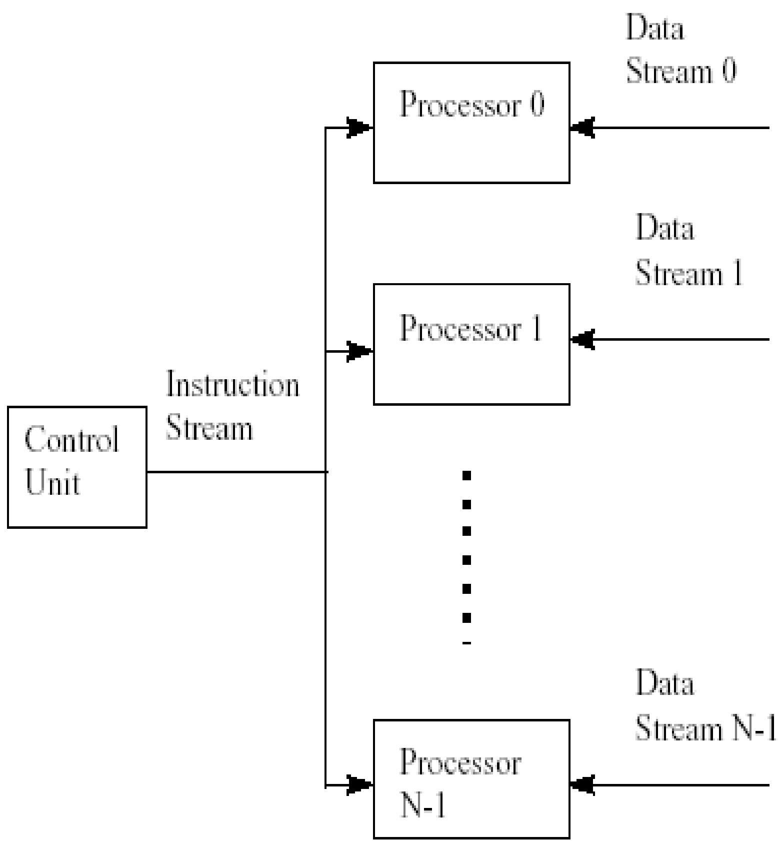
Multiple instruction streams, multiple data streams (MIMD)

- Tightly-coupled MIMD
- Loosely-coupled MIMD

SISD



SIMD



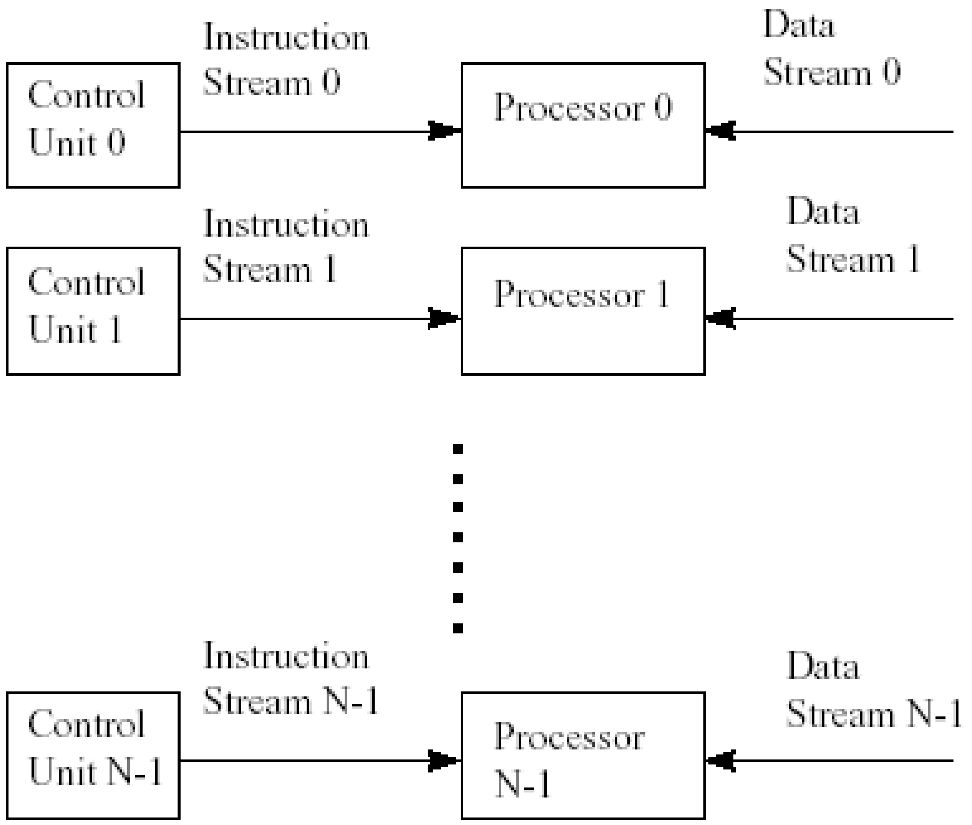
SISD system architecture of [Fly66]

Image processing application

- Large number of PEs
- Minimum complexity PEs
- Programming language is a simple extension of a sequential language

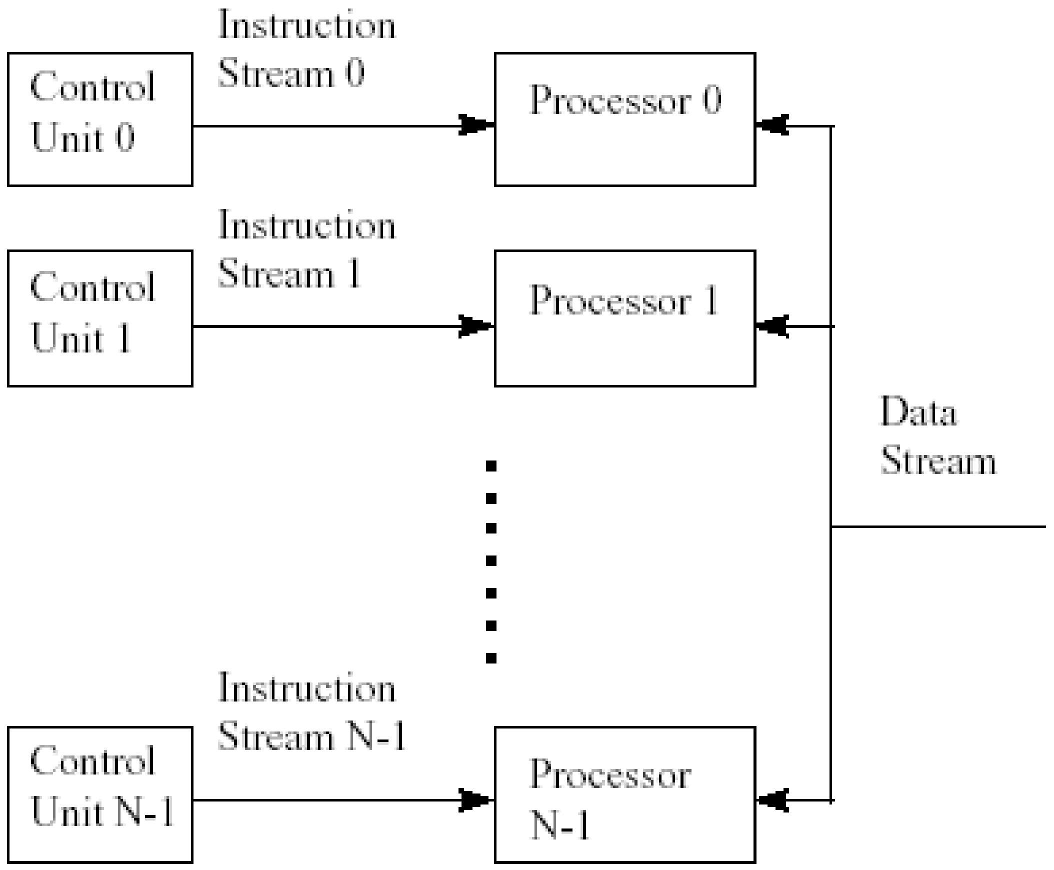
- Each PE is of higher complexity and it is usually built with commercial devices
- Each PE has local memory

MIMD



MIMD system architecture of [Fly66]

MISD



MISD system architecture of [Fly66]

Applications:
- Classification
- Robot vision

Cache coherence

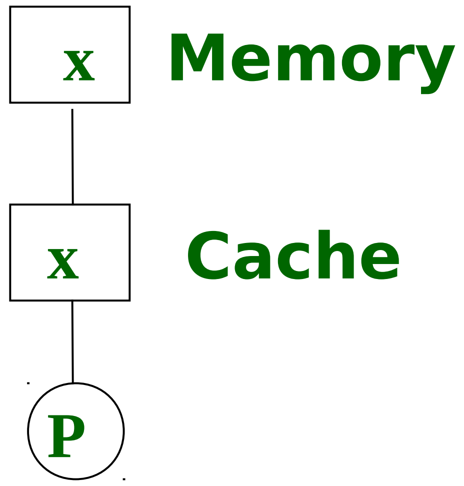
Single Processor caching

Hit: data in the cache

Miss: data is not in the cache

Hit rate: h

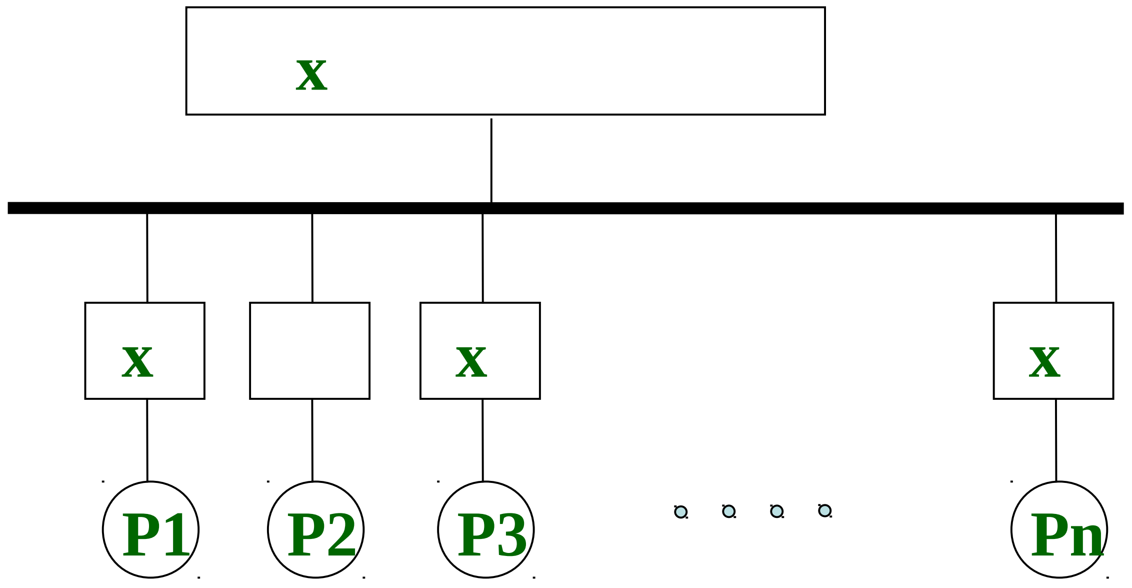
Miss rate: m = (1-h)



Cache Coherence Policies

- **Writing to Cache in 1 processor case**
  - **Write Through**
  - **Write Back**

Cache Coherence



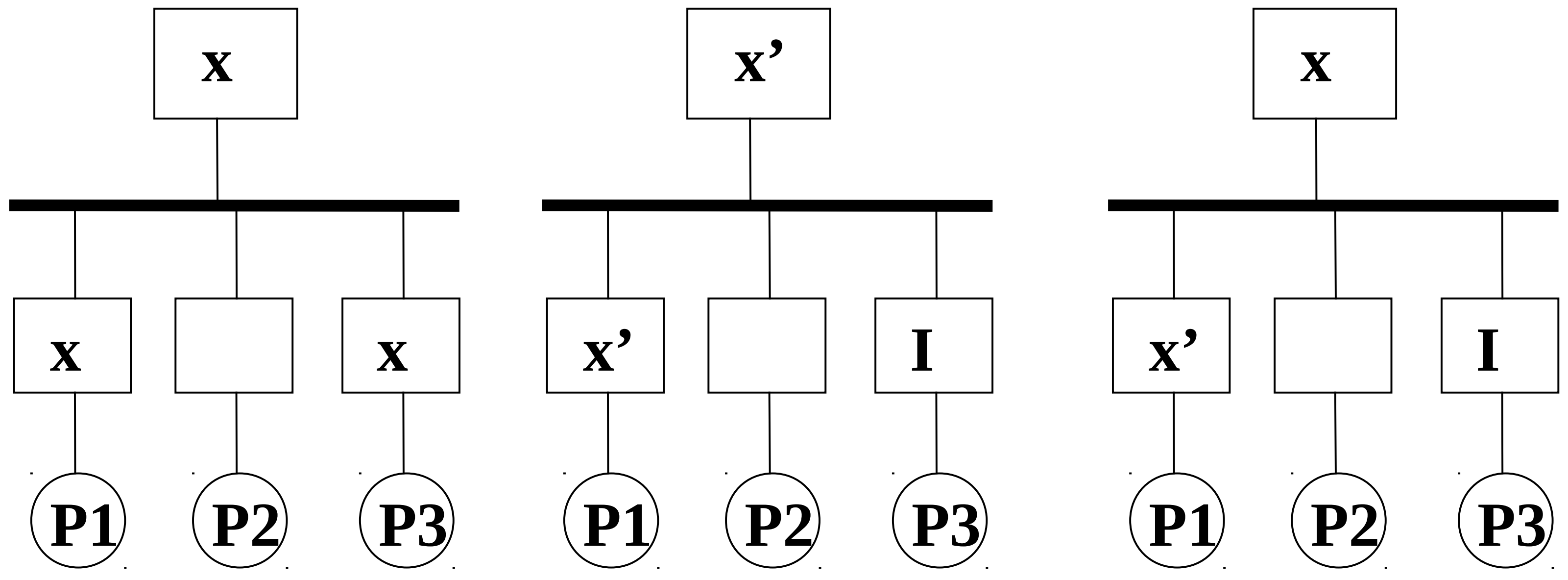
-Multiple copies of x

-What if P1 updates x?

Cache Coherence Policies

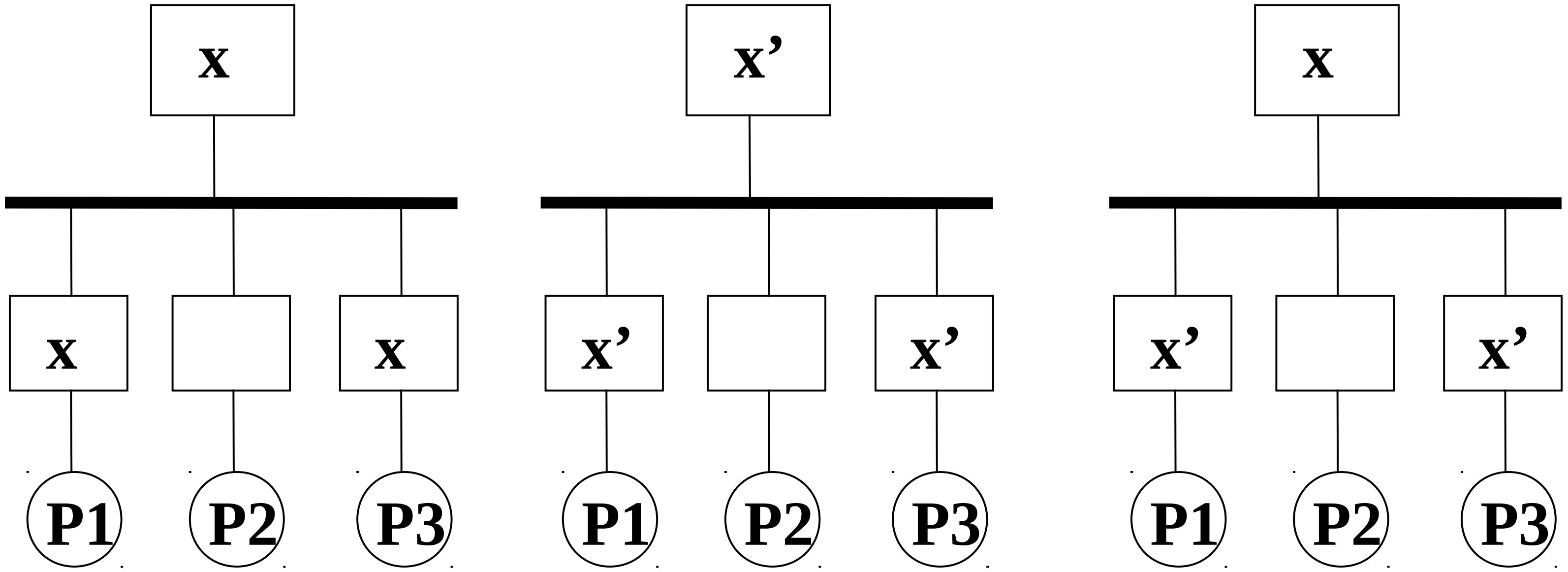
- **Writing to Cache in n processor case**
  - **Write Invalidate - Write Back**
  - **Write Invalidate - Write Through**
  - **Write Update - Write Through**
  - **Write Update - Write Back**

Write-invalidate



Before   Write Through   Write back

Write-Update

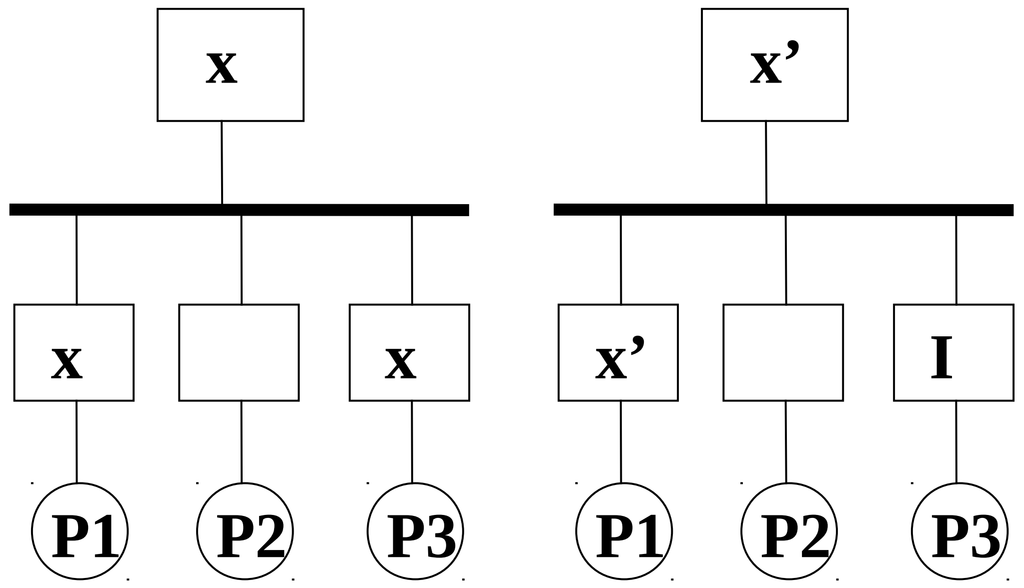


Before **Write Through** **Write back**

Snooping Protocols

Snooping protocols are based on watching bus activities and carry out the appropriate coherency commands when necessary. Global memory is moved in blocks, and each block has a state associated with it, which determines what happens to the entire contents of the block. The state of a block might change as a result of the operations Read-Miss, Read-Hit, Write-Miss, and Write-Hit.

Write-invalidate



Before **Write Through**

Multiple processors can read block copies from main memory safely until one processor updates its copy. At this time, all cache copies are invalidated and the memory is updated to remain consistent.

Write Through- Write Invalidate (cont.)

State	Description
Valid [VALID]	The copy is consistent with global memory
Invalid [INV]	The copy is inconsistent

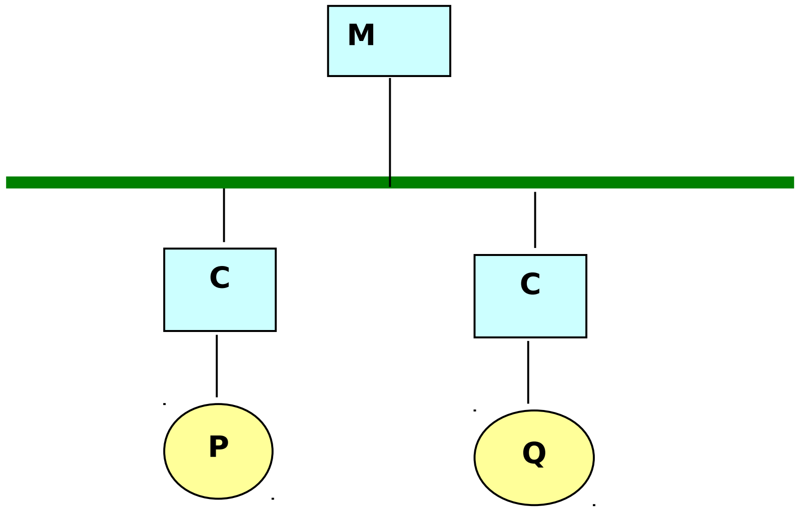
Write Through- Write Invalidate (cont.)

Event	Actions
Read Hit	Use the local copy from the cache.
Read Miss	Fetch a copy from global memory. Set the state of this copy to Valid.
Write Hit	Perform the write locally. Broadcast an Invalid command to all caches. Update the global memory.
Write Miss	Get a copy from global memory. Broadcast an invalid command to all caches. Update the global memory. Update the local copy and set its state to Valid.
Replace	Since memory is always consistent, no write back is needed when a block is replaced.

Example 1

X = 5

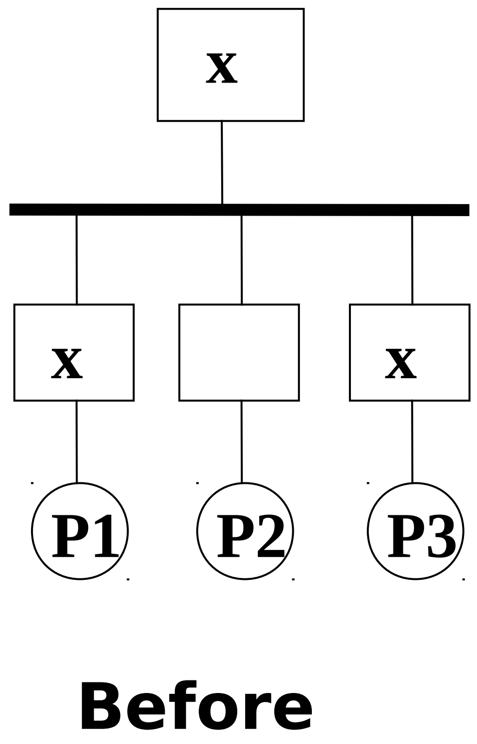
1. P reads X
2. Q reads X
3. Q updates X, X=10
4. Q reads X
5. Q updates X, X=15
6. P updates X, X=20
7. Q reads X

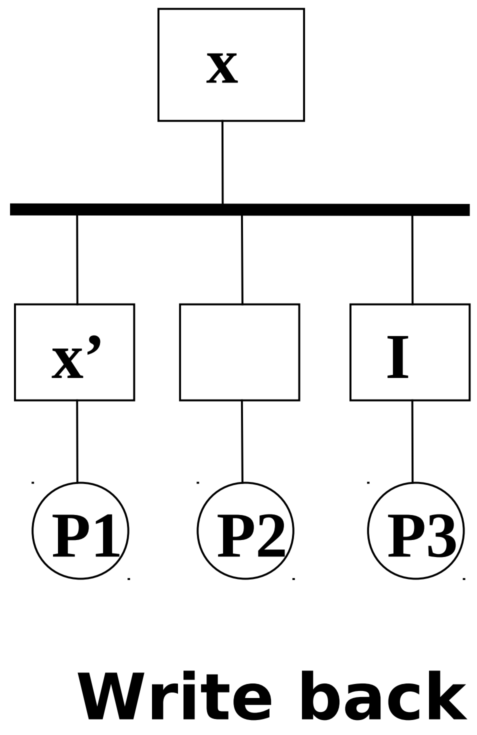


Write through write invalidate

		Memory	P's	C	Q's	C
	Event	X	X	State	X	State
0	Original value	5				
1	P reads X (Read Miss)	5	5	V		

Write-invalidate





A valid block can be owned by memory and shared in multiple caches that can contain only the shared copies of the block. Multiple processors can safely read these blocks from their caches until one processor updates its copy. At this time, the writer becomes the only owner of the valid block and all other copies are invalidated.

Write-invalidate

In the write-back protocol, multiple copies of a cache block may exist if different processors have loaded (read) the block into their caches.

If some processor wants to change this block, it must first become an exclusive owner of this block.

When the ownership is granted to this processor by the memory module that is the home location of the block. All other copies, including the one in the memory module, are invalidated.

Now the owner of the block may change the contents of the memory.

When another processor wishes to read this block, the data are sent to this processor by the current owner.

The data are also sent to the home memory module, which requires ownership and updates the block to contain the latest value.

Write Back- Write Invalidate

State	Description
Shared (Read-Only) [RO]	**Data is valid and can be read safely. Multiple copies can be in this state**
Exclusive (Read-Write) [RW]	**Only one valid cache copy exists and can be read from and written to safely. Copies in other caches are invalid**
Invalid [INV]	**The copy is inconsistent**

Write Back- Write Invalidate

State	Description
Shared (Read-Only) [RO]	**Data is valid and can be read safely. Multiple copies can be in this state**
Exclusive (Read-Write) [RW]	**Only one valid cache copy exists and can be read from and written to safely. Copies in other caches are invalid**
Invalid [INV]	**The copy is inconsistent**

Ownership

Event	Action
Read Hit	Use the local copy from the cache.
Read Miss:	If no Exclusive (Read-Write) copy exists, then supply a copy from global memory. Set the state of this copy to Shared (Read-Only). If an Exclusive (Read-Write) copy exists, make a copy from the cache that set the state to Exclusive (Read-Write), update the global memory and local cache with the copy. Set the state to Shared (Read-Only) in both caches.

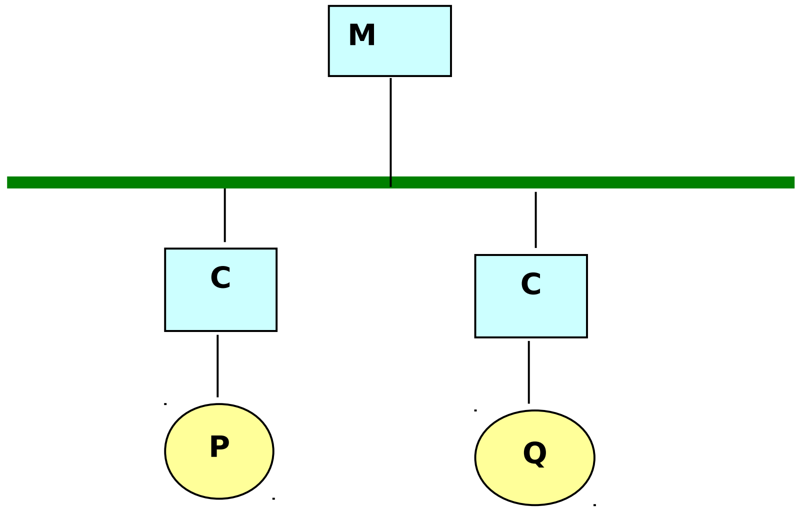
Ownership (cont.)

Write Hit	If the copy is Exclusive (Read-Write), perform the write locally. If the state is Shared (Read-Only), then broadcast an Invalid to all caches. Set the state to Exclusive (Read-Write).
Write Miss	Get a copy from either a cache with an Exclusive (Read-Write) copy, or from global memory itself. Broadcast an Invalid command to all caches. Update the local copy and set its state to Exclusive (Read-Write).
Block Replacement	If a copy is in an Exclusive (Read-Write) state, it has to be written back to main memory if the block is being replaced. If the copy is in Invalid or Shared (Read-Only) states, no write back is needed when a block is replaced.

Example 1

X = 5

1. P reads X
2. Q reads X
3. Q updates X, X=10
4. Q reads X
5. P reads X
6. Q updates X, X=15
7. P updates X, X=20
8. Q reads X



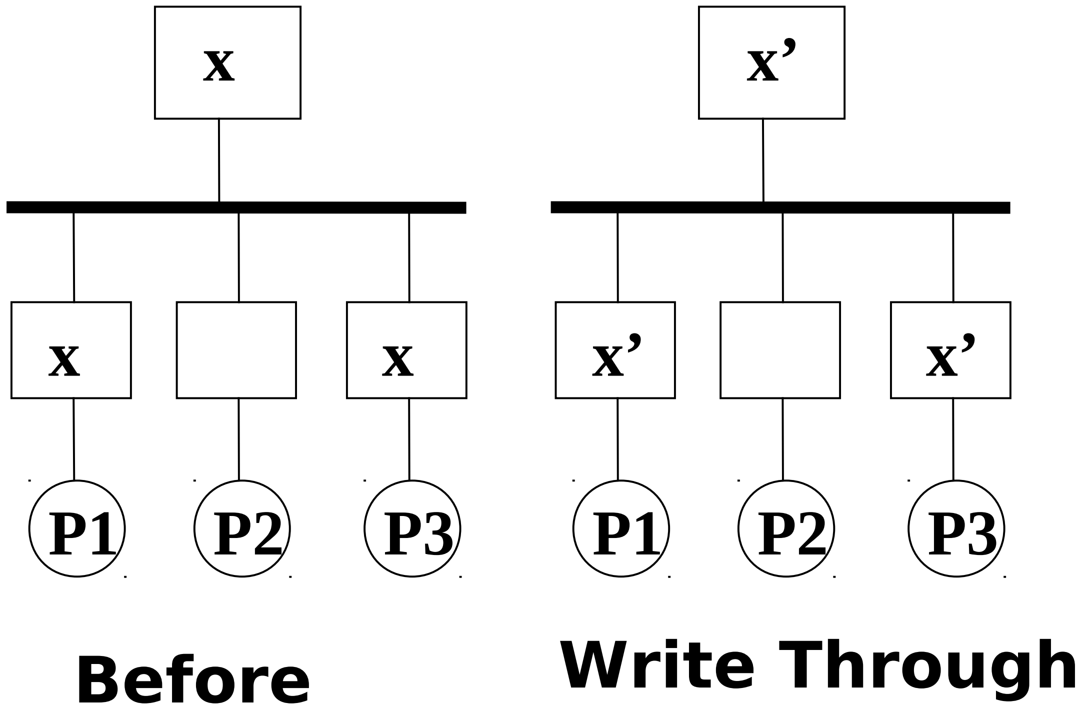
Complete the table

		Memory	P's	Cache	Q's	Cache
	Event	X	X	State	X	State
0	Original value	5				
1	P reads X (Read Miss)	5	5	RO		

Write Once This write-invalidate protocol, which was proposed by Goodman in 1983 uses a combination of write-through and write-back.

Write-through is used the very first time a block is written. Subsequent writes are performed using write back.

Write-Update



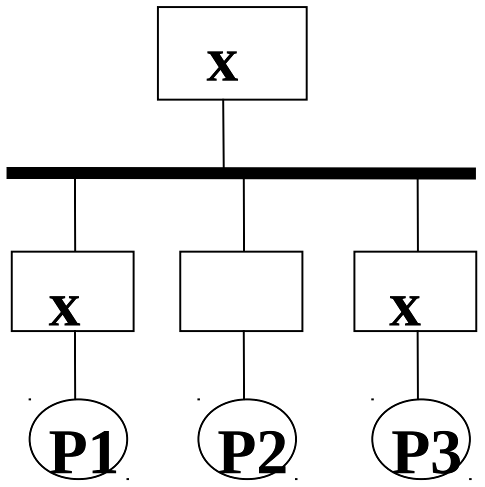
Write update and partial through:

- one cache is written to memory at the same time it is broadcast to other caches sharing the updated block.
- These caches snoop on the bus and perform updates to their local copies.
- There is also a special bus line, which is asserted to indicate that at least one other cache is sharing the block.

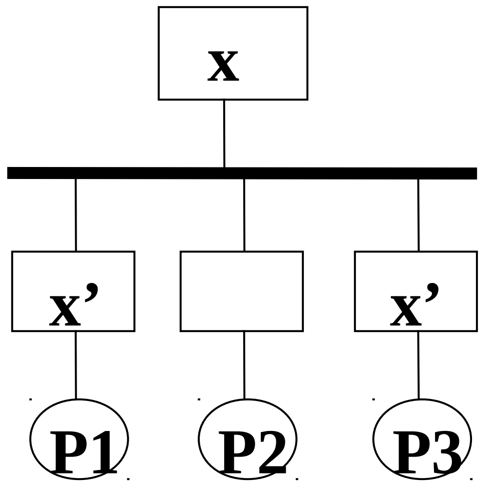
Write update and partial write through

State	Description
Valid Exclusive [VAL-X]	This is the only cache copy and is consistent with global memory
Shared [SHARE]	There are multiple caches copies shared. All copies are consistent with memory
Dirty [DIRTY]	This copy is not shared by other caches and has been updated. It is not consistent with global memory. (Copy ownership)

Write-Update Write Back



Before



Write back

This protocol is similar to the, **Write update: write partial through**, one except that instead of writing through to the memory whenever a shared block is updated, memory updates are done only when the block is being replaced.

Write Update Write Back

State	Description
Valid Exclusive [VAL-X]	This is the only cache copy and is consistent with global memory
Shared Clean [SH-CLN]	There are multiple caches copies shared.
Shared Dirty [SH-DRT]	There are multiple shared caches copies. This is the last one being updated. (Ownership)
Dirty [DIRTY]	This copy is not shared by other caches and has been updated. It is not consistent with global memory. (Ownership)

Directory Based Protocols

- Due to the nature of some interconnection networks and the size of the shared memory system, updating or invalidating caches using snoopy protocols might become unpractical.
- Cache coherence protocols that somehow store information on where copies of blocks reside are called directory schemes.
  - A directory is a data structure that maintains information on the processors that share a memory block and on its state. The information maintained in the directory could be either centralized or distributed.
  - **Centralized vs. Distributed**

A Central directory maintains information about all blocks in a central data structure. (Bottleneck, large search time!)

The same information can be handled in a distributed fashion by allowing each memory module to maintain a separate directory.