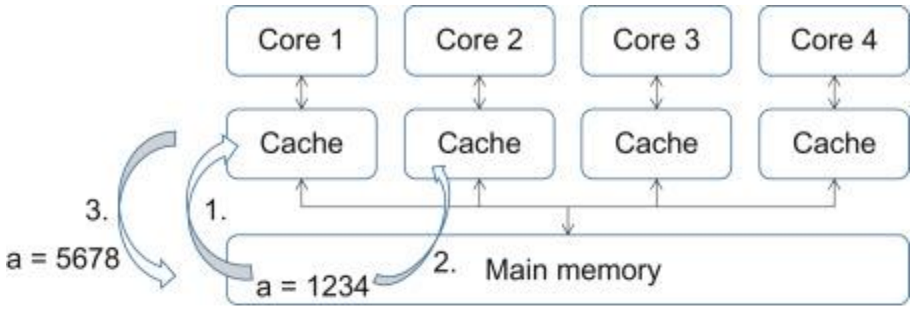
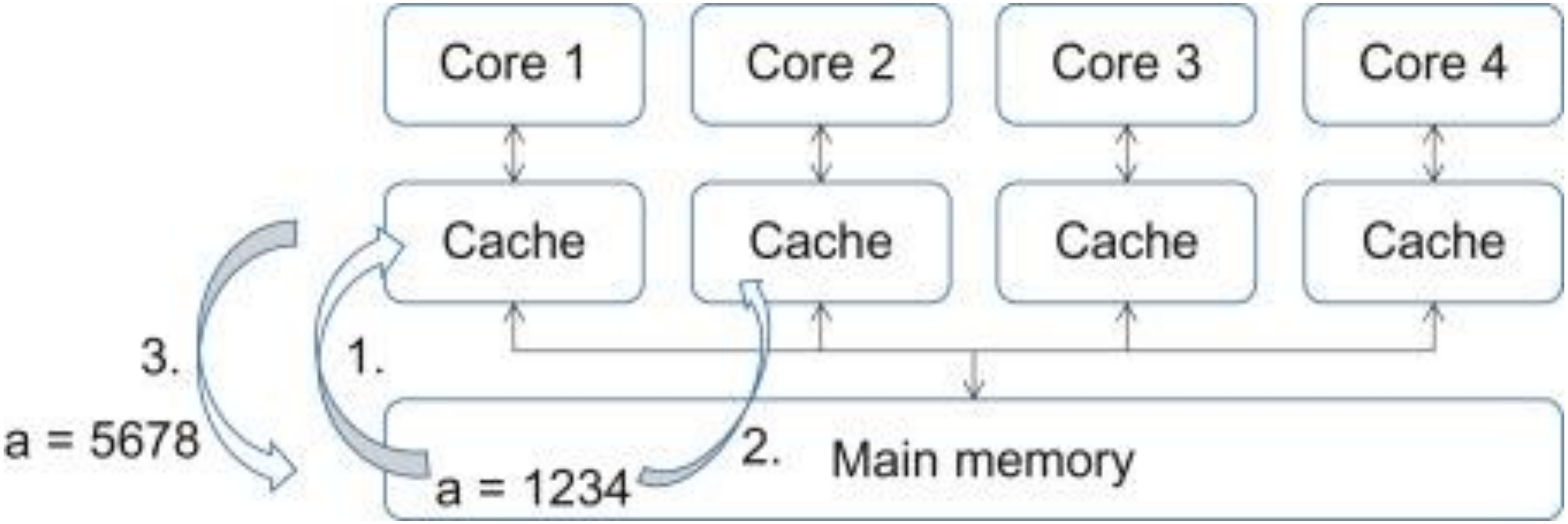
Cache coherency

Dr.Maheswari.R

Cache coherency

- Cache coherency is a situation where multiple processor cores share the same memory hierarchy, but have their own L1 data and instruction caches. Incorrect execution could occur if two or more copies of a given cache block exist, in two processors' caches, and one of these blocks is modified.



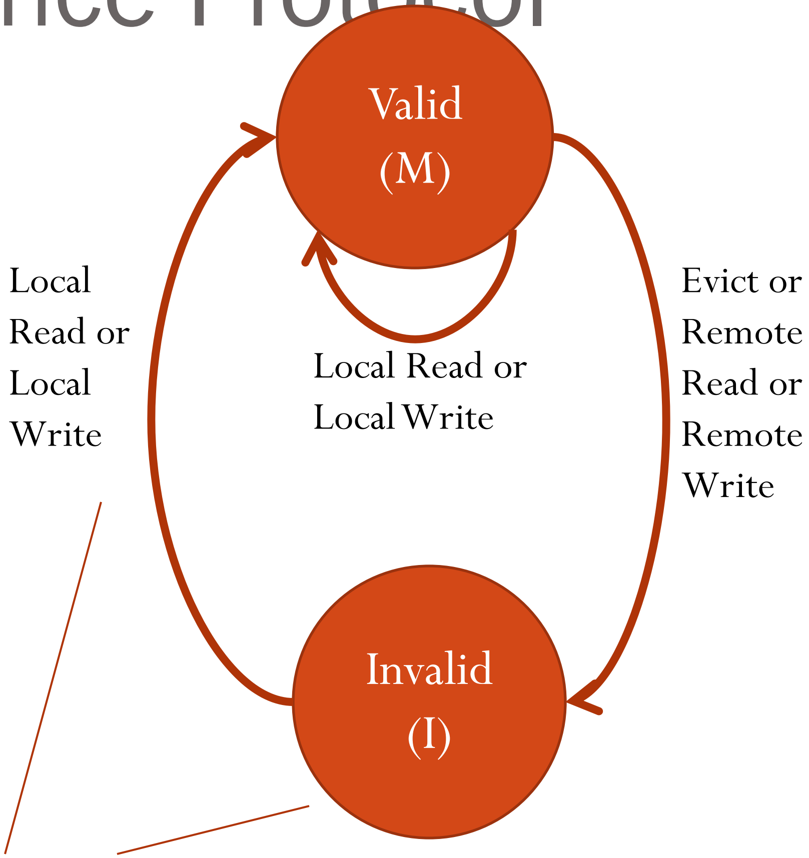

Core 1
Core 2
Core 3
Core 4
Cache
Cache
Cache
Cache
3.
1.
a = 5678
2.
Main memory
a = 1234

Snoopy Cache Coherence

- All requests **broadcast** on bus
- All processors and memory **snoop** and **respond**
- Cache blocks writeable at one processor or read-only at several
  - Single-writer protocol

Minimal Coherence Protocol

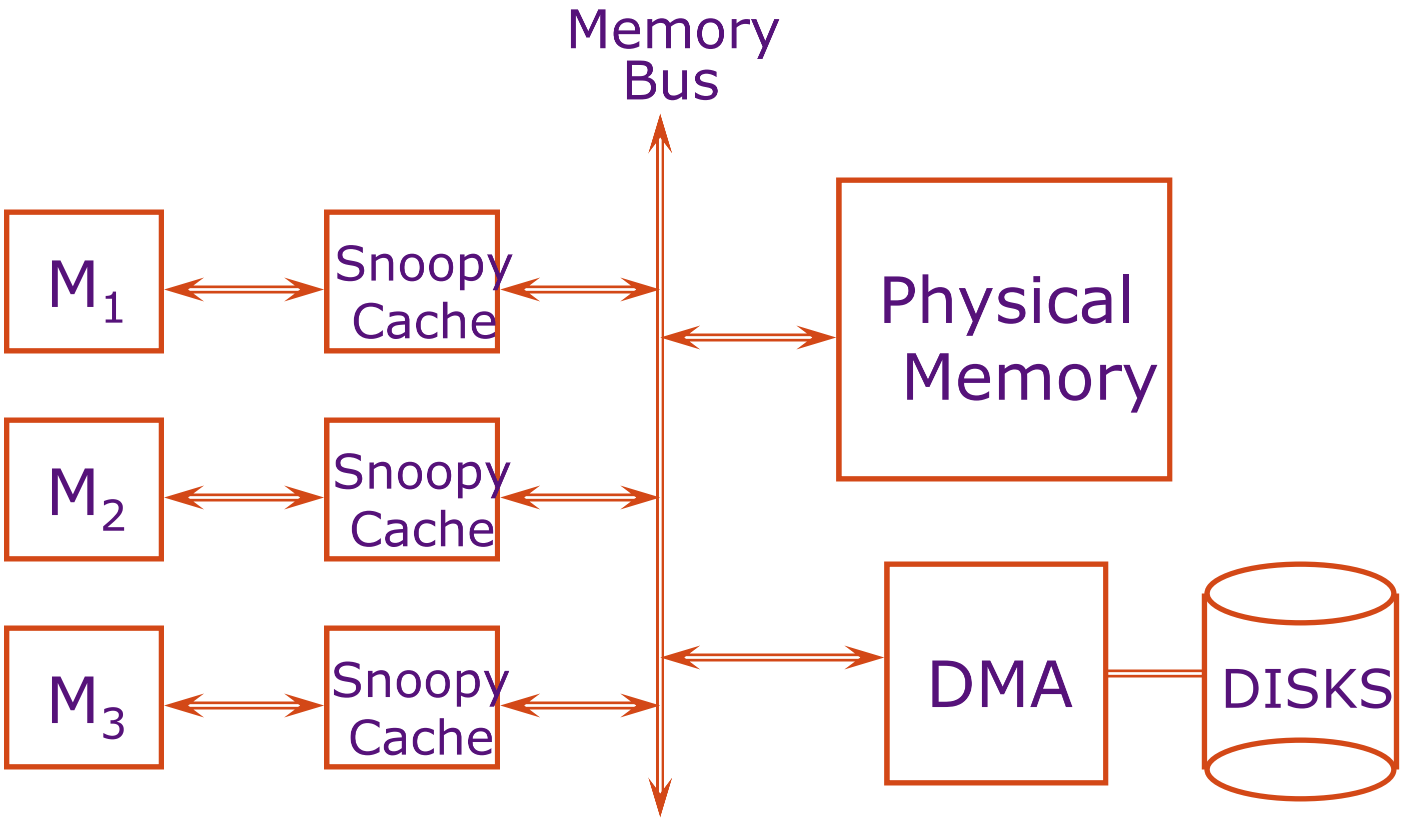
- Blocks are always private or exclusive
- State transitions:
  - Local read: I->M, fetch, invalidate other copies
  - Local write: I->M, fetch, invalidate other copies
  - Evict: M->I, write back data
  - Remote read: M->I, write back data
  - Remote write: M->I, write back data


Valid
(M)
Local Read or Local Write
Local Read or Local Write
Evict or Remote Read or Remote Write
Invalid
(I)

Cache

Tag	State	Data
A	M	…
B	I	…

Shared Memory Multiprocessor



Use snoopy mechanism to keep all processors' view of memory coherent

MSI Protocol:

- This is a basic cache coherence protocol used in multiprocessor system. The letters of protocol name identify possible states in which a cache can be.

MSI

- **Modified –**

  The block has been modified n cache, i.e., the data in the cache **is inconsistent** with the backing store (memory). So, a cache with a block in "M" state has responsibility to write the block to backing store when it is evicted.
- **Shared –**

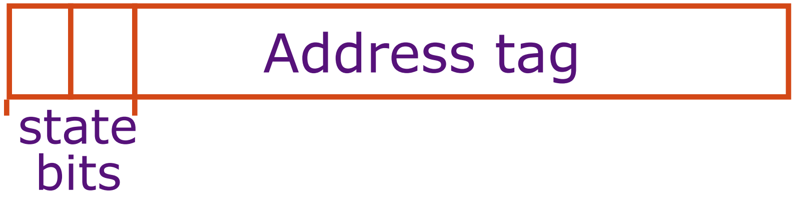
  This block is not modified and is present in **atleast one cache**. The cache can evict the data without writing it to backing store.
- **Invalid –**

  This block is **invalid** and must be fetched from **memory** or from another **cache** if it is to be stored in this cache.

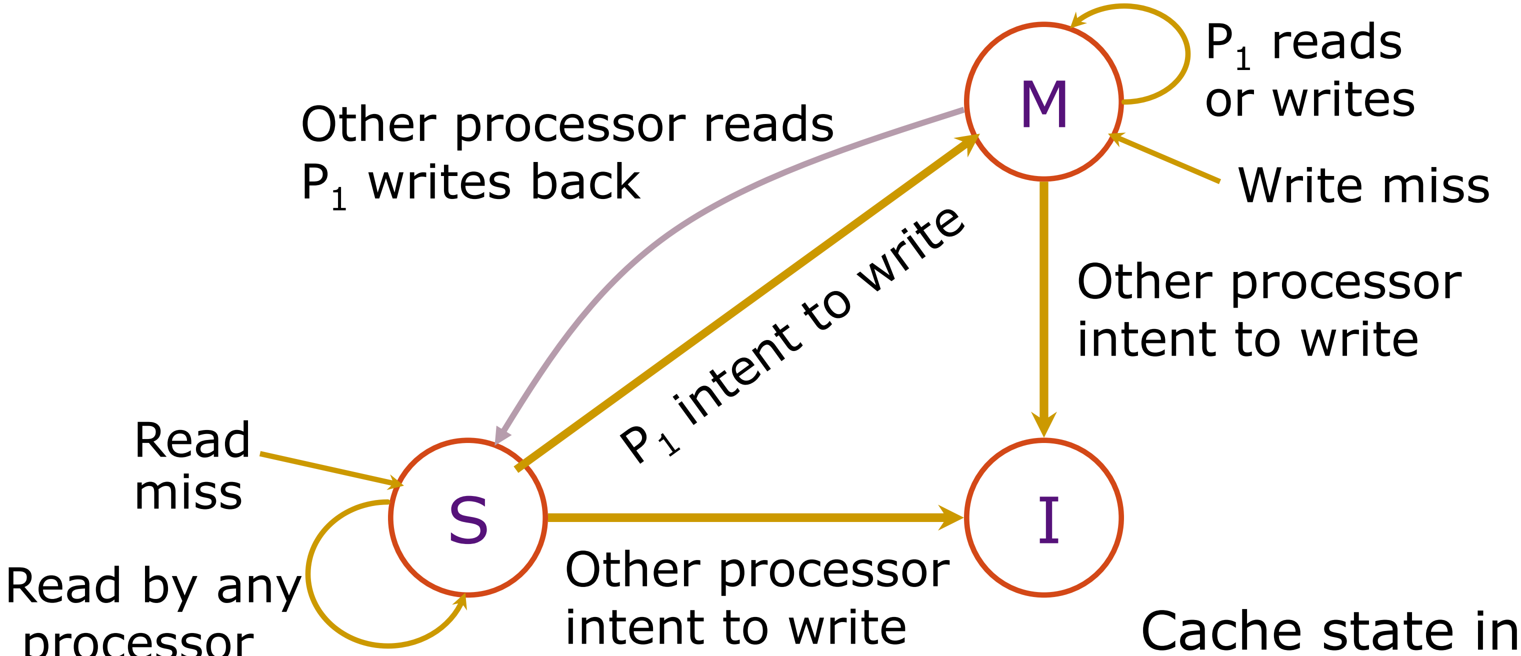
Cache State Transition Diagram

The MSI protocol

Each cache line has a tag

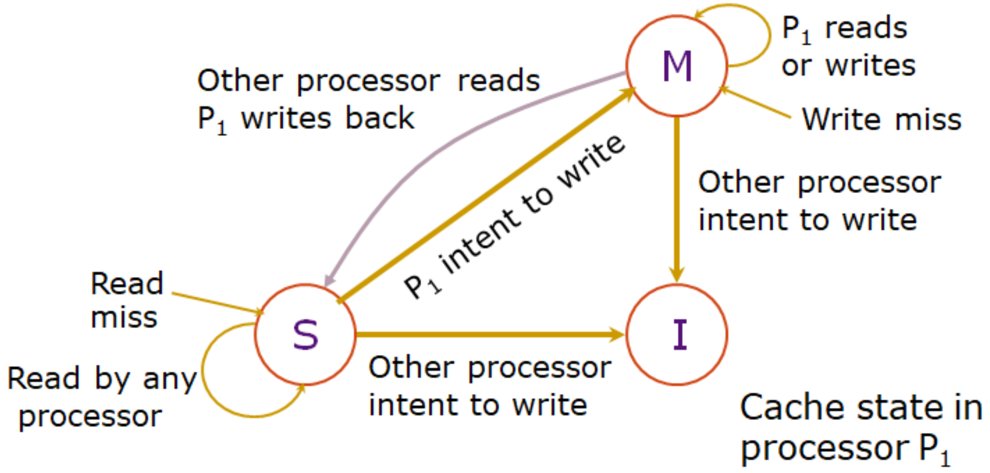


M: Modified
S: Shared
I: Invalid


M
S
I
P_1 reads or writes
Write miss
Other processor reads P_1 writes back
P_1 intent to write
Other processor intent to write
Read miss
Read by any processor
Other processor intent to write

Cache state in processor P_1

MSI Protocol


M
S
I
P₁ reads or writes
Write miss
Other processor reads P₁ writes back
P₁ intent to write
Other processor intent to write
Read miss
Read by any processor
Other processor intent to write

Cache state in processor P_1

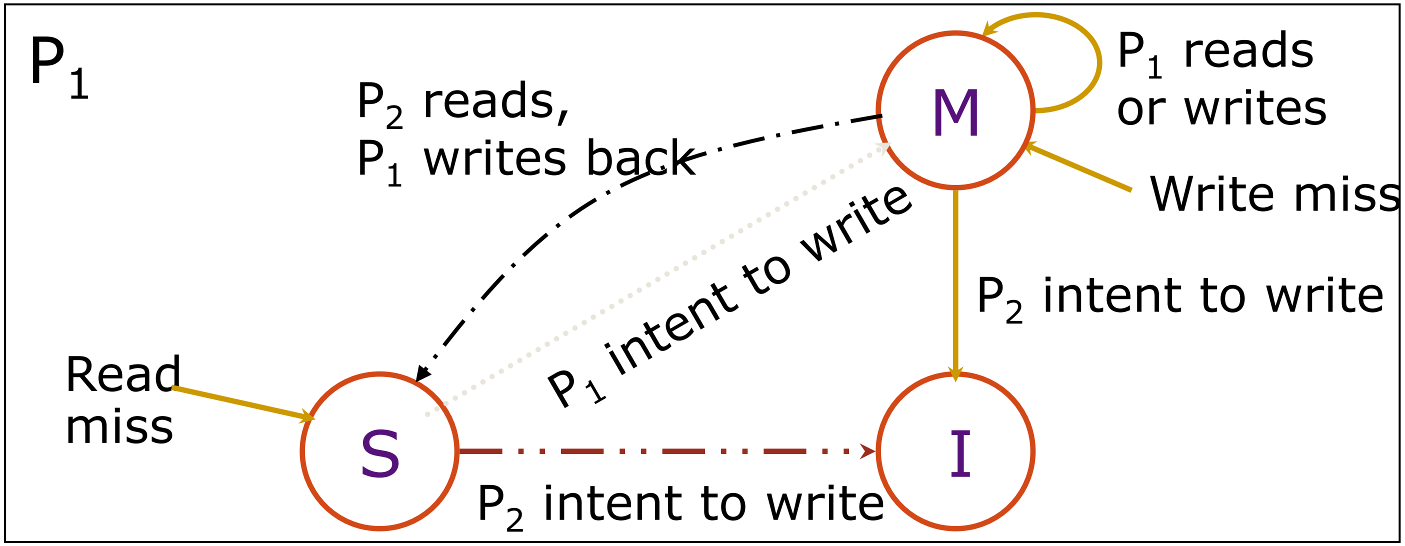
	Action and Next State						
Current State	***Processor Read***	***Processor Write***	***Eviction***		***Cache Read***	***Cache Read&M***	***Cache Upgrade***
I	***Cache Read*** **Acquire Copy** **→ S**	***Cache Read&M*** **Acquire Copy** **→ M**			**No Action** **→ I**	**No Action** **→ I**	**No Action** **→ I**
S	**No Action** **→ S**	***Cache Upgrade*** **→ M**	**No Action** **→ I**		**No Action** **→ S**	***Invalidate Frame*** **→ I**	***Invalidate Frame*** **→ I**

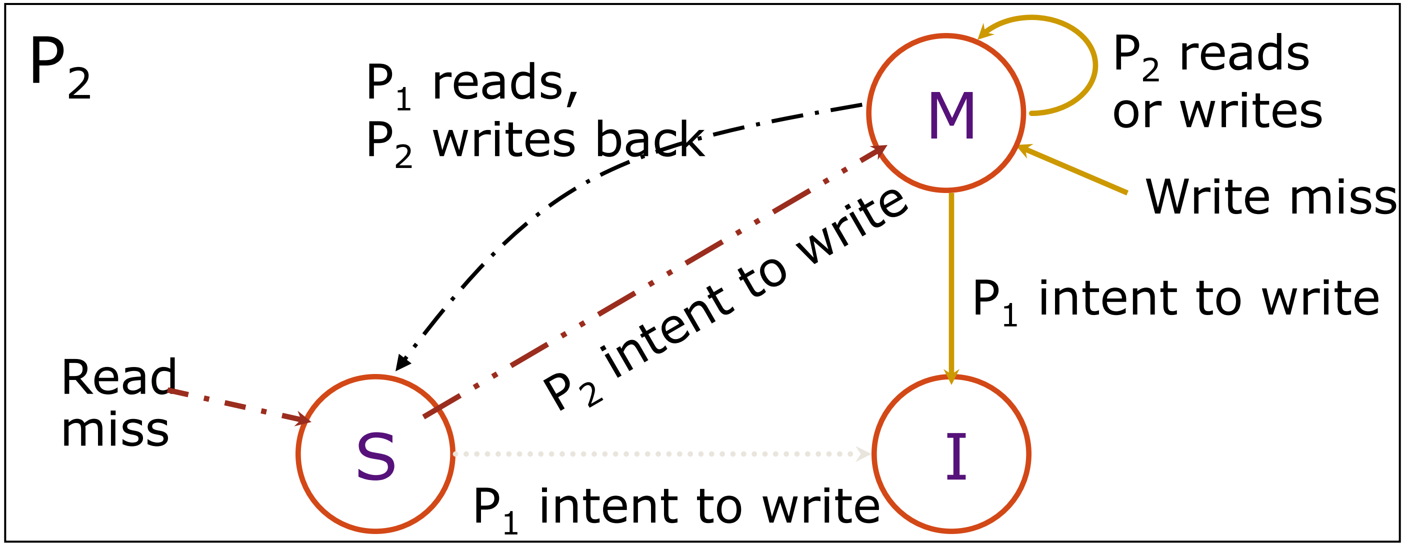
M	**No Action**
	→ M
		No Action
		→ M
			Cache Write back
			→ I
					Memory inhibit; Supply data;
					→ S
						Invalidate Frame;
						Memory inhibit; Supply data;
						→ I	

Two Processor Example

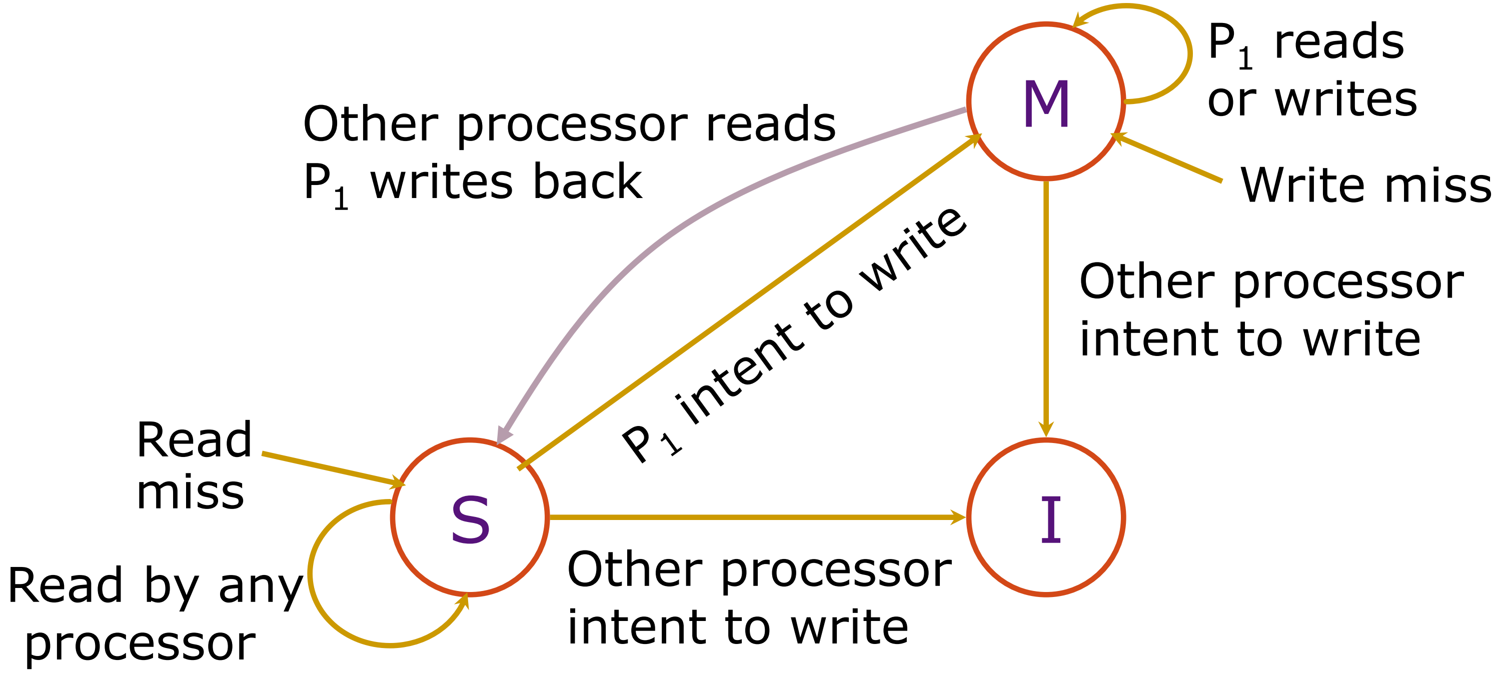
(Reading and writing the same cache line)

P_1 reads
P_1 writes
P_2 reads
P_2 writes
P_1 reads
P_1 writes
P_2 writes
P_1 writes





Observation



M
P1 reads or writes
Write miss
Other processor reads P1 writes back
P1 intent to write
Other processor intent to write
Read miss
S
Read by any processor
Other processor intent to write
I

- If a line is in the M state then no other cache can have a copy of the line!
  - Memory stays coherent, multiple differing copies cannot exist

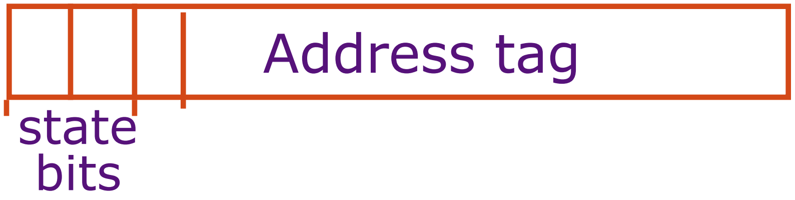
MESI Protocol

- Variation used in many Intel processors
- 4-State Protocol
  - **M**odified: <1,0,0…0>
  - **E**xclusive: <1,0,0,…,1>
  - **S**hared: <1,X,X,…,1>
  - **I**nvalid: <0,X,X,…X>
- Bus/Processor Actions
  - Same as MSI
- Adds *shared* signal to indicate if other caches have a copy

MESI: An Enhanced MSI protocol

increased performance for private data

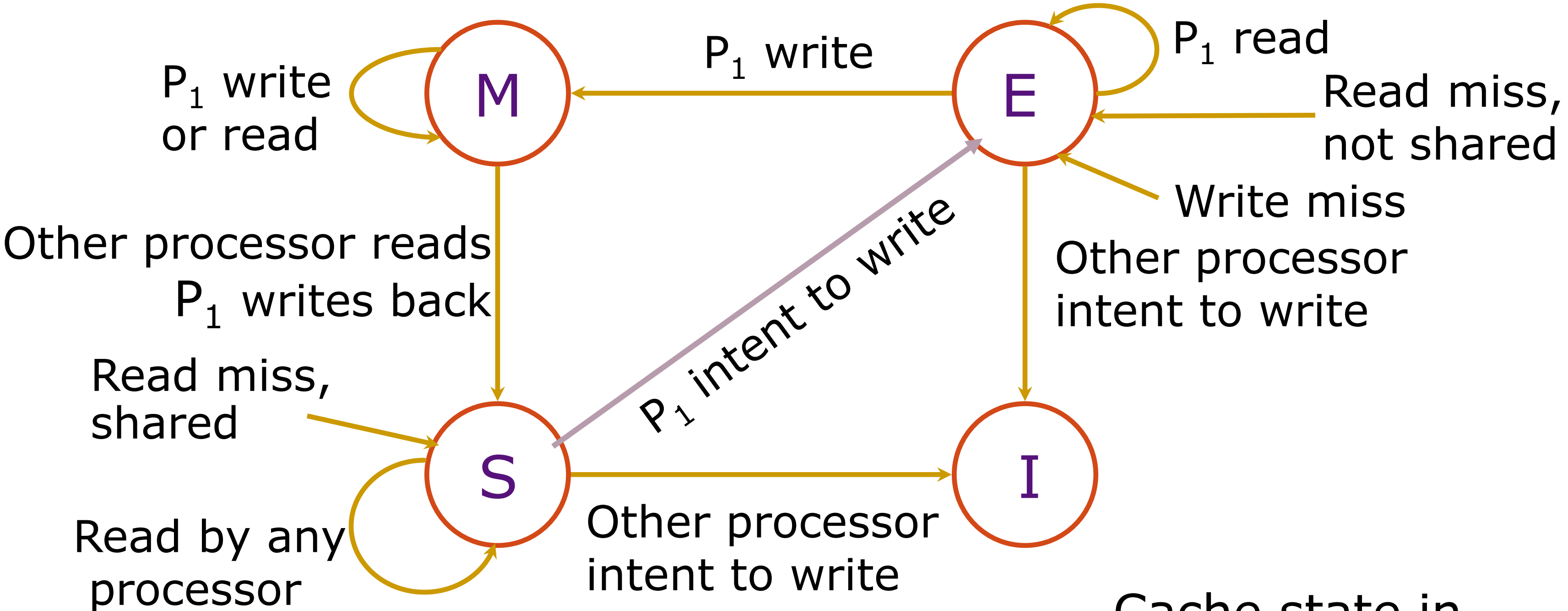
Each cache line has a tag



Address tag

state bits

M: Modified Exclusive
E: Exclusive, unmodified
S: Shared
I: Invalid



M

E

S

I

P_1 write or read

P_1 write

P_1 read

Read miss, not shared

Write miss

Other processor intent to write

Other processor reads
P_1 writes back

P_1 intent to write

Read miss, shared

Read by any processor

Other processor intent to write

Cache state in processor P_1

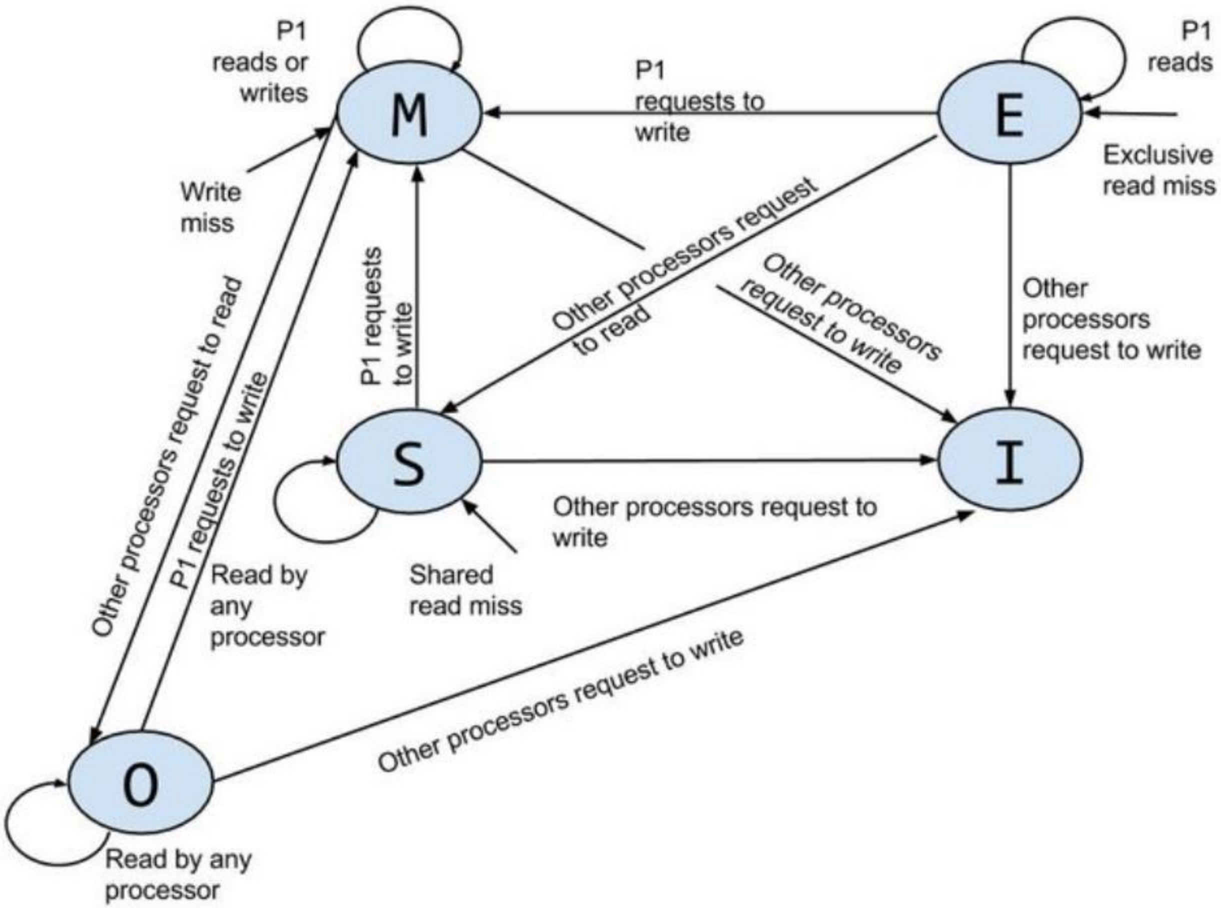
MESI Protocol

	Action and Next State						
Current State	***Processor Read***	***Processor Write***	***Eviction***		***Cache Read***	***Cache Read&M***	***Cache Upgrade***
I	*Cache Read* If no sharers: → E If sharers: → S	*Cache Read&M* → M			No Action → I	No Action → I	No Action → I
S	No Action → S	*Cache Upgrade* → M	No Action → I		Respond Shared: → S	No Action → I	No Action → I
E	No Action → E	No Action → M	No Action → I		Respond Shared; → S	No Action → I	

M	No Action
	→ M
		No Action
		→ M
			Cache Write-back
			→ I
					Respond dirty;
					Write back data;
					→ S
						Respond dirty;
						Write back data;
						→ I	

MOESI Protocol

- Used in AMD Opteron
- 5-State Protocol
  - **M**odified: <1,0,0…0>
  - **E**xclusive: <1,0,0,…,1>
  - **S**hared: <1,X,X,…,1>
  - **I**nvalid: <0,X,X,…X>
  - **O**wned: <1,X,X,X,0> ; only one owner, memory not up to date
- Owner can supply data, so memory does not have to supply
  - Avoids lengthy memory access


M
E
S
I
O
P1 reads or writes
Write miss
P1 requests to write
P1 reads
Exclusive read miss
Other processors request to read
Other processors request to write
Other processors request to write
Other processors request to read
P1 requests to write
P1 requests to write
Read by any processor
Shared read miss
Other processors request to write
Other processors request to write
Read by any processor

MOESI Protocol

	Action and Next State						
Current State	***Processor Read***	***Processor Write***	***Eviction***		***Cache Read***	***Cache Read&M***	***Cache Upgrade***
I	***Cache Read*** **If no sharers:** **→ E** **If sharers:** **→ S**	***Cache Read&M*** **→ M**			**No Action** **→ I**	**No Action** **→ I**	**No Action** **→ I**
S	**No Action** **→ S**	***Cache Upgrade*** **→ M**	**No Action** **→ I**		**Respond shared;** **→ S**	**No Action** **→ I**	**No Action** **→ I**

E	**No Action**
	→ E
		No Action
		→ M
			No Action
			→ I
					Respond shared;
					Supply data;
					→ S
						Respond shared;
						Supply data;
						→ I	

O	**No Action**
	→ O
		Cache Upgrade
		→ M
			Cache Write-back
			→ I
					Respond shared;
					Supply data;
					→ O
						Respond shared;
						Supply data;
						→ I	

M	**No Action**
	→ M
		No Action
		→ M
			Cache Write-back
			→ I
					Respond shared;
					Supply data;
					→ O
						Respond shared;
						Supply data;
						→ I	