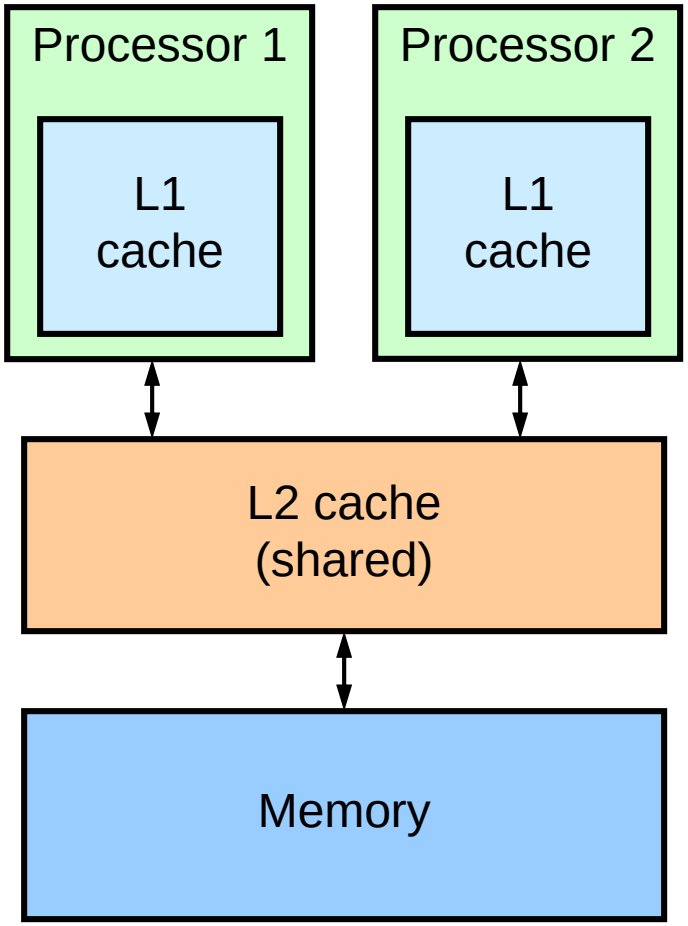
MESI Protocol

Multi-processor System

- **A memory system is *coherent* if**
  1. If P1 writes to address X,
     and later on P2 reads X,
     and there are no other writes to X in between
     ⇒ P2's read returns the value written by P1's write

  2. Writes to the same location are serialized:
     two writes to location X are seen in the same order by all processors

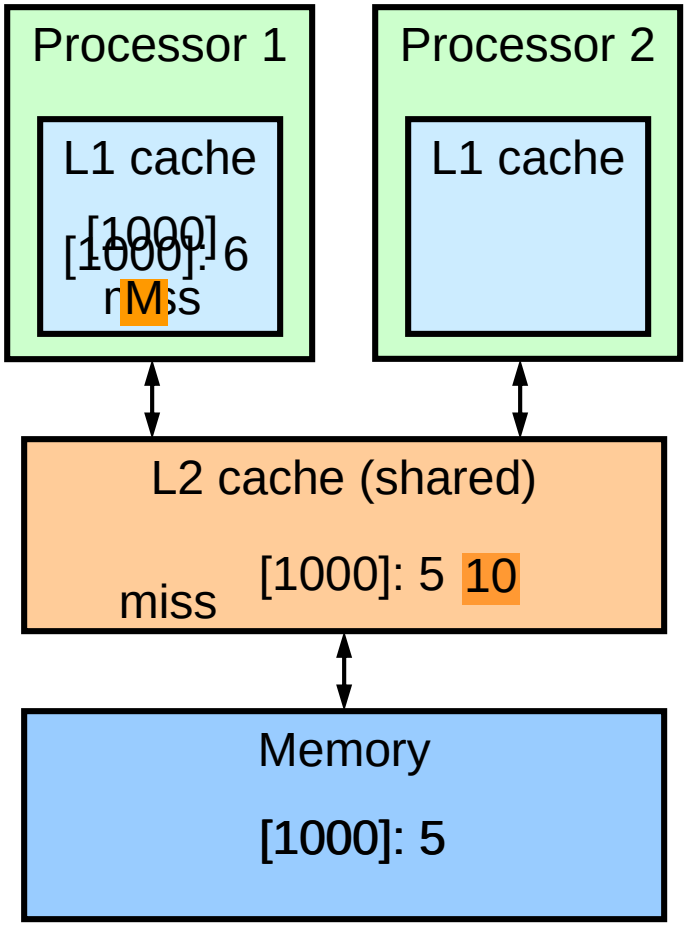


MESI Protocol

- **Each cache line can be in one of 4 states**
  - Invalid – Line's data is not valid
  - Shared – Line is valid and not dirty,
    copies may exist in other processors
  - Exclusive – Line is valid and not dirty,
    other processors do not have the line in their local caches
  - Modified – Line is valid and dirty,
    other processors do not have the line in their local caches

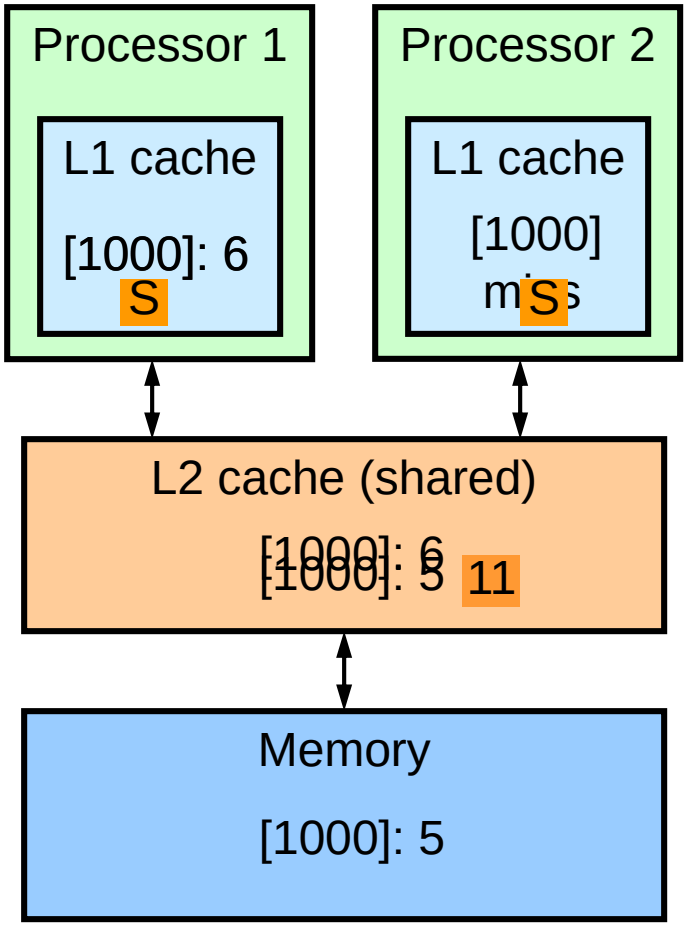
Multi-processor System: Example

- **P1 reads 1000**
- **P1 writes 1000**



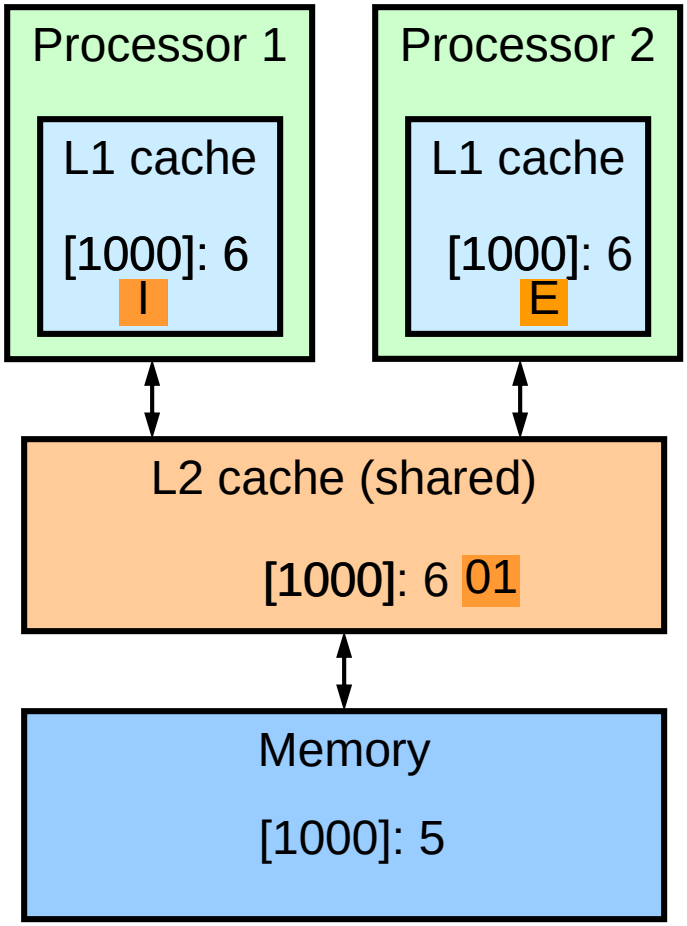
Multi-processor System: Example

- **P1 reads 1000**
- **P1 writes 1000**
- **P2 reads 1000**
- **L2 snoops 1000**
- **P1 writes back 1000**
- **P2 gets 1000**



Multi-processor System: Example

- P1 reads 1000
- P1 writes 1000
- P2 reads 1000
- L2 snoops 1000
- P1 writes back 1000
- P2 gets 1000
- P2 requests for ownership with write intent



Core Valid Bits and Inclusion

- **L2 keeps track of the presence of each line in each of the Core's L1 caches**
  - Determine if it needs to send a snoop to a processor
  - Determine in what state to provide a requested line (S,E)
  - Maintain Core Valid Bits (CVB) per cache line
  - ⇒ Need to guarantee that the L1 caches in each Core are inclusive of the L2 cache
- **When L2 evicts a line**
  - L2 sends a snoop invalidate to all processors that have it
  - If the line is modified in the L1 cache of one of the processors (in which case it exist only in that processor)
    - The processor responds by sending the updated value to L2
    - When the line is evicted from L2, the updated value gets written to memory

MESI Protocol States

State	Valid	Modified	Copies may exist in other processors
Invalid	No	N.A.	N.A
Shared	Yes	No	Yes
Exclusive	Yes	No	No
Modified	Yes	Yes	No

- **A modified line must be exclusive**
  - Otherwise, another processor which has the line will be using stale data
  - Therefore, before modifying a line, a processor must request ownership of the line

MESI Protocol Example

- A four-processor shared-memory system implements MESI protocol
- For the following sequence of memory references, show the state of the line containing the variable X in each processor's cache after each reference is resolved
- Each processors start out with the line containing X invalid in their cache

	P0's state	**P1's state**	**P2's state**	**P3's state**
Initial State	*I*	*I*	*I*	*I*
P0 reads X	*E*	*I*	*I*	*I*
P1 reads X	*S*	*S*	*I*	*I*
P2 reads X	*S*	*S*	*S*	*I*
P3 writes X	*I*	*I*	*I*	*M*
P0 reads X	*S*	*I*	*I*	*S*

CVBs
0000
1000
1100
1110
0001
1001

MESI Question 2

- **Dual Core processor**
- **MESI**
- **Each core has an L1 cache – L2 cache is shared**
- **L1 can send the following packets to L2**
  - Read Address (A)
    - In case of L1 miss on address A
  - RFO (A)
  - Data (A)
- **L2 can send the following packets**
  - Read Address (A) to memory
  - Data (A) to L1 including MESI state
  - RFO (A) – after a RFO from L1
  - Snoop (A) to L1
- **Memory can send the following packets**
  - Data (A) to L2

MESI question 2

- **Messages times:**
  - between L1 and L2: 10ns
  - Between L2 and memory: 100ns
- **Caches are empty upon reset**
- **Series of requests to address A**

	P1's L1 state	P2's L1 state	P3's L1 state	L2 CVBs P1	P2	P3	Time	Total message time
Initial State	I	I	I	0	0	0		
P2 reads A								
P2 writes A								
P1 reads A								
P2 reads A								
P3 reads A								
P3 writes A								

Read For Ownership (RFO)

RFO Request:

- A signal from private to shared cache (i.e. L1->L2) requesting cache line exclusivity for write intent
  - MLC/LLC invalidates cache line in other L1s
  - MLC/LLC responds to L1 that RFO has been granted
  - L1 can now modify cache line

Global Observation (GO)

Assume L1 (private) requests line from L2 (shared)

<u>Global Observation (GO):</u>

Before sending the actual data to L1, L2 responds to L1 that line is observed to be in it by all other processors

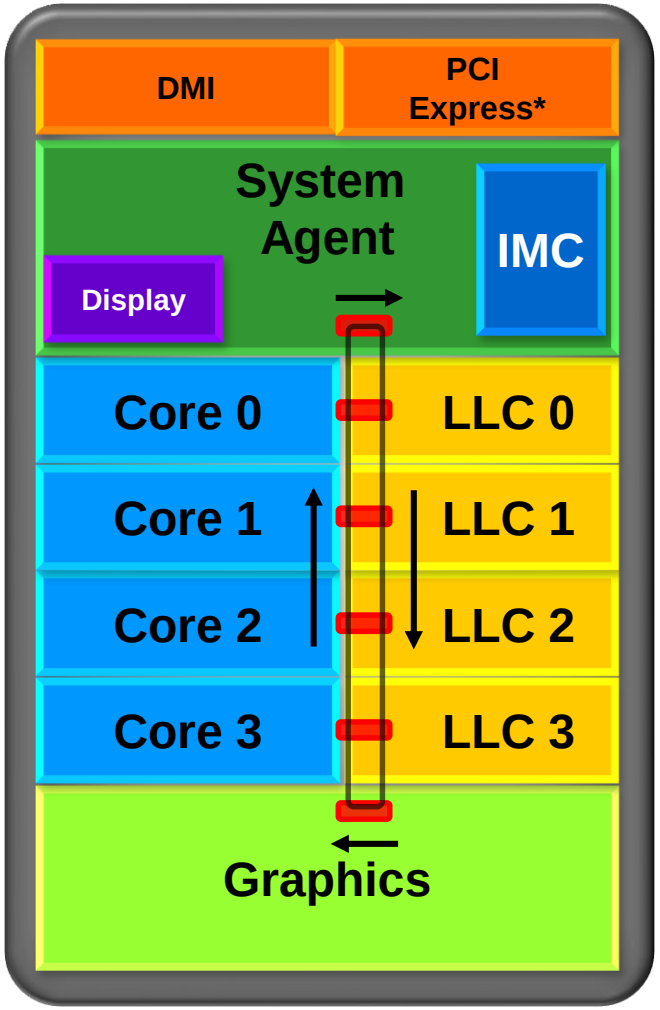
The Global Observation carries the MESI state E/S

- **So each L1→L2 line request is answered in 2 steps:**
  - GO
  - Fill (actual data)

Ring Interconnect

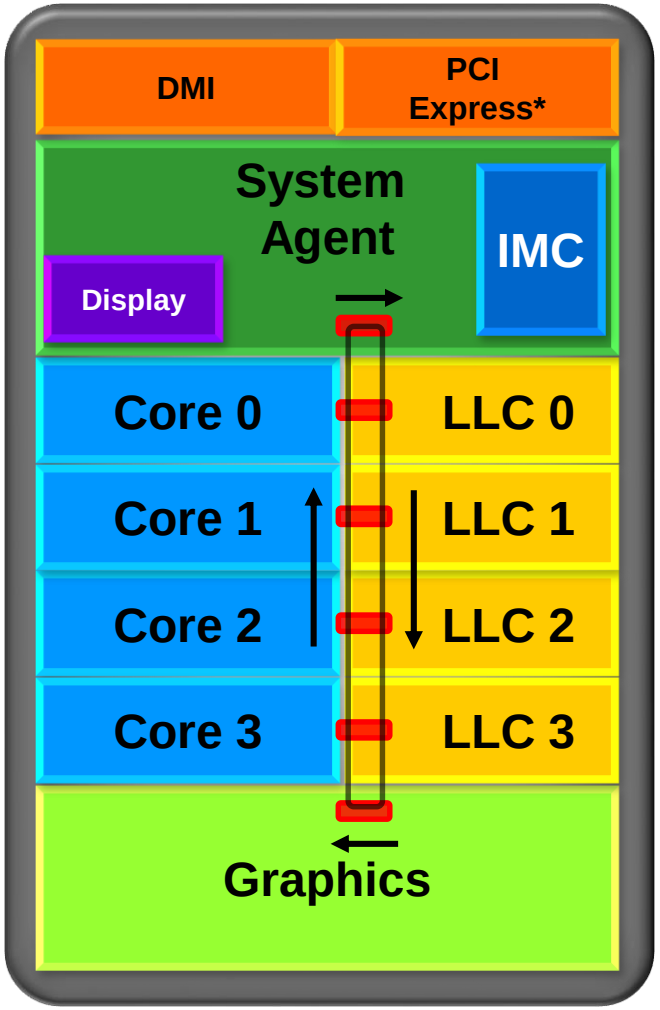
Ring

- **2 x 4 rings**
  - Req / Data / Ack / Snoop
  - Packets always use the shortest path
  - Static Even/Odd polarity per station
  - Each ring switches polarity on each cycle



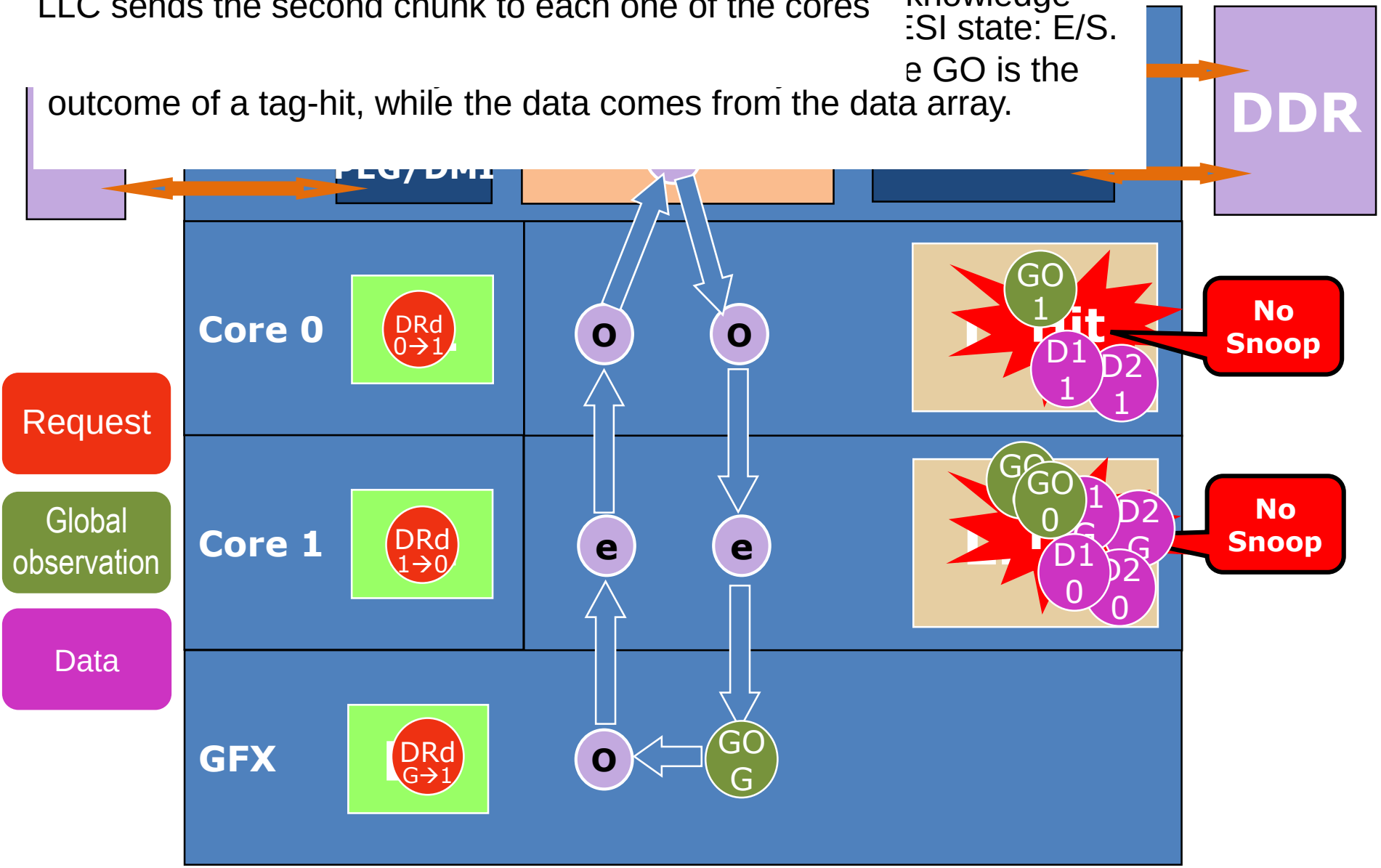
Ring

- **In our example:**
  - 64B cache line (L1/L2/LLC)
  - 32B/cycle bus BW
  - → Cache line data transfers from LLC to L1 in two strokes



LLC sends the second chunk to each one of the cores

knowledge
ESI state: E/S.
e GO is the
outcome of a tag-hit, while the data comes from the data array.



Cycle 15: up ring E, down ring O