Cache Coherence

- ***Mostly important when we study multiprocessors***
  - ***Cache Coherence (by hardware; what is/isn't guaranteed)***
    - ***Same address in two caches MUST contain the same value***
    - ***No guarantees as to what the value is***
  - ***Memory Consistency (by software, needs cache coherence)***
    - ***If you care what the value is***
    - ***Needs cache coherence to make it work***
- ***Even in a uniprocessor***
  - ***IF we have intelligent controller***
  - ***e.g., DMA loads memory, independent of the processor***
    - ***Processor sees one thing in cache, Memory sees something else***

Cache Coherence

- ***Definition:***

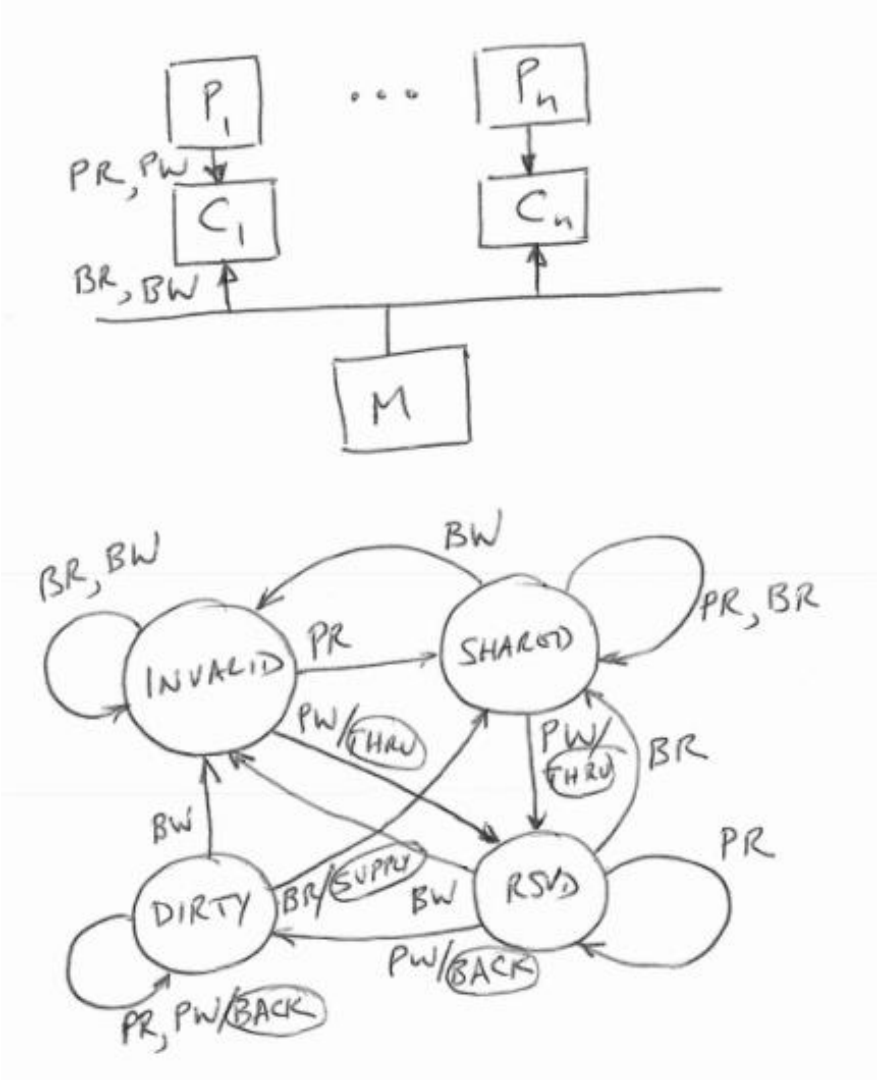
  If a cache line is present in more than one cache, it has the same values in all caches

- ***Why is it important?***

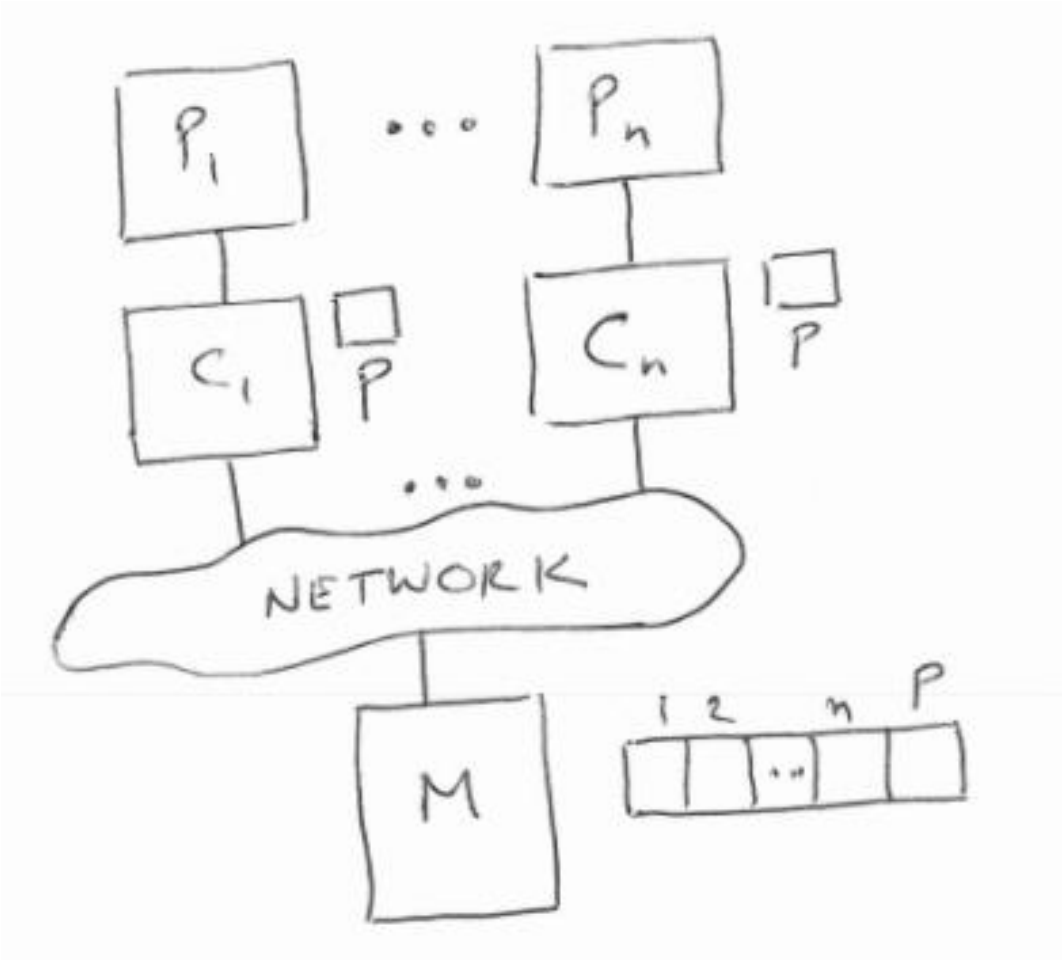
- ***Snoopy schemes***
  - ***All caches snoop the common bus***

- ***Directory schemes***
  - ***Each cache line has a directory entry in memory***

A Snoopy Scheme



Directory Scheme (p=n)



Sequential Consistency

- ***Definition: Memory sees loads/stores in program order.***
- ***Why it is important: It guarantees mutual exclusion.***

Two Processors and a critical section

Processor 1	*Processor 2*
L1=0	*L2=0*
A: L1 = 1	*X: L2=1*
B: If L2 =0	*Y: If L1=0*
{critical section}	*{critical section}*
C: L1=0	*Z: L2=0*

What can happen?

Order of A,B,C,X,Y,Z obeying seq. consistency (Shown are the ten starting with A, There are also ten starting with X)

A	A	A	A	A	A	A	A	A	A
B	B	B	B	X	X	X	X	X	X
C	X	X	X	B	B	B	Y	Y	Y
X	C	Y	Y	C	Y	Y	B	B	Z
Y	Y	C	Z	Y	C	Z	C	Z	B
Z	Z	Z	C	Z	Z	C	Z	C	C

Which processors get exclusive access to the critical section under each order?

1,2	1,2	1	1	2	None	None	None	None	1

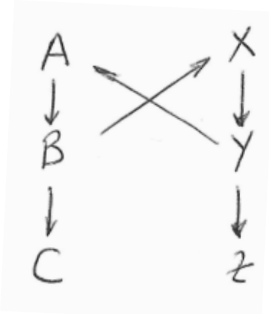
Note: 1,2 means first 1 gets exclusive access, and when done, 2 gets access.

Critical Sections require Mutual Exclusion

- ***A critical section: a region of memory containing data wherein only one thread can be allowed access at a time.***
- ***Why is this important?***
  - ***P1 program wants to execute LD R1,A followed by ADD A,A,R1***
  - ***Between the two instructions, suppose P2 executes ST R2,A***
  - ***The bad result: A contains A + R2 instead of A + A.***
- ***Mutual Exclusion Property***
  - ***Only one processor at a time can access the critical section***
- ***What happens if B occurs before X, and Y occurs before A?***
  - ***If B occurs before X, Process 1 can access the memory***
  - ***If Y occurs before A, Process 2 can access the memory***
  - ***No mutual exclusion!***

Sequential Consistency Guarantees Mutual Exclusion

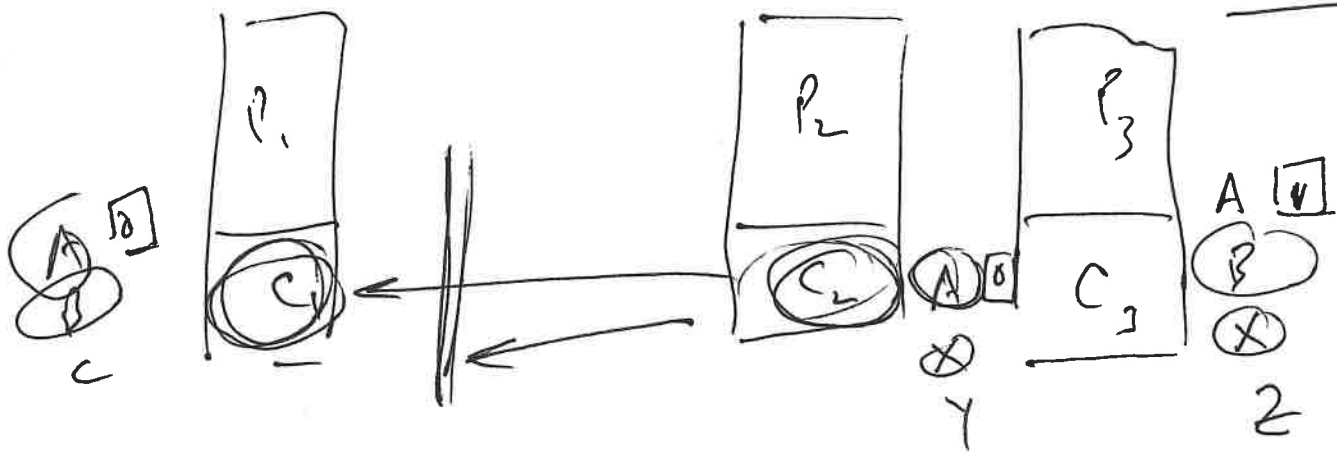
- ***Sequential consistency means all memory accesses reach the memory system in program order. That is, A occurs before B, B before C, and X before Y, Y before Z.***
- ***Both processors accessing shared memory concurrently means B must occur before X and Y must occur before A.***
- ***That is, sequential consistency AND concurrent access only if A before B before X before Y before A. Impossible!***

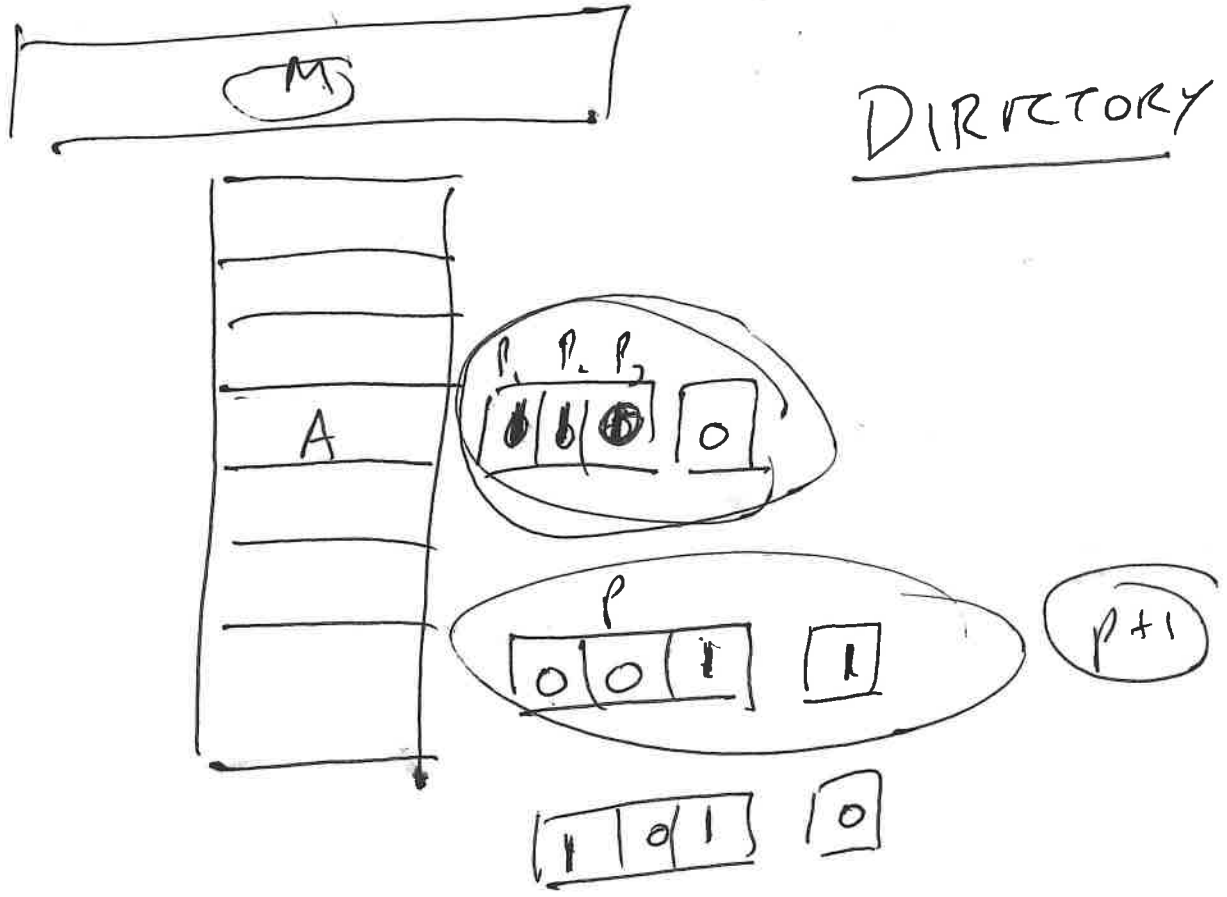


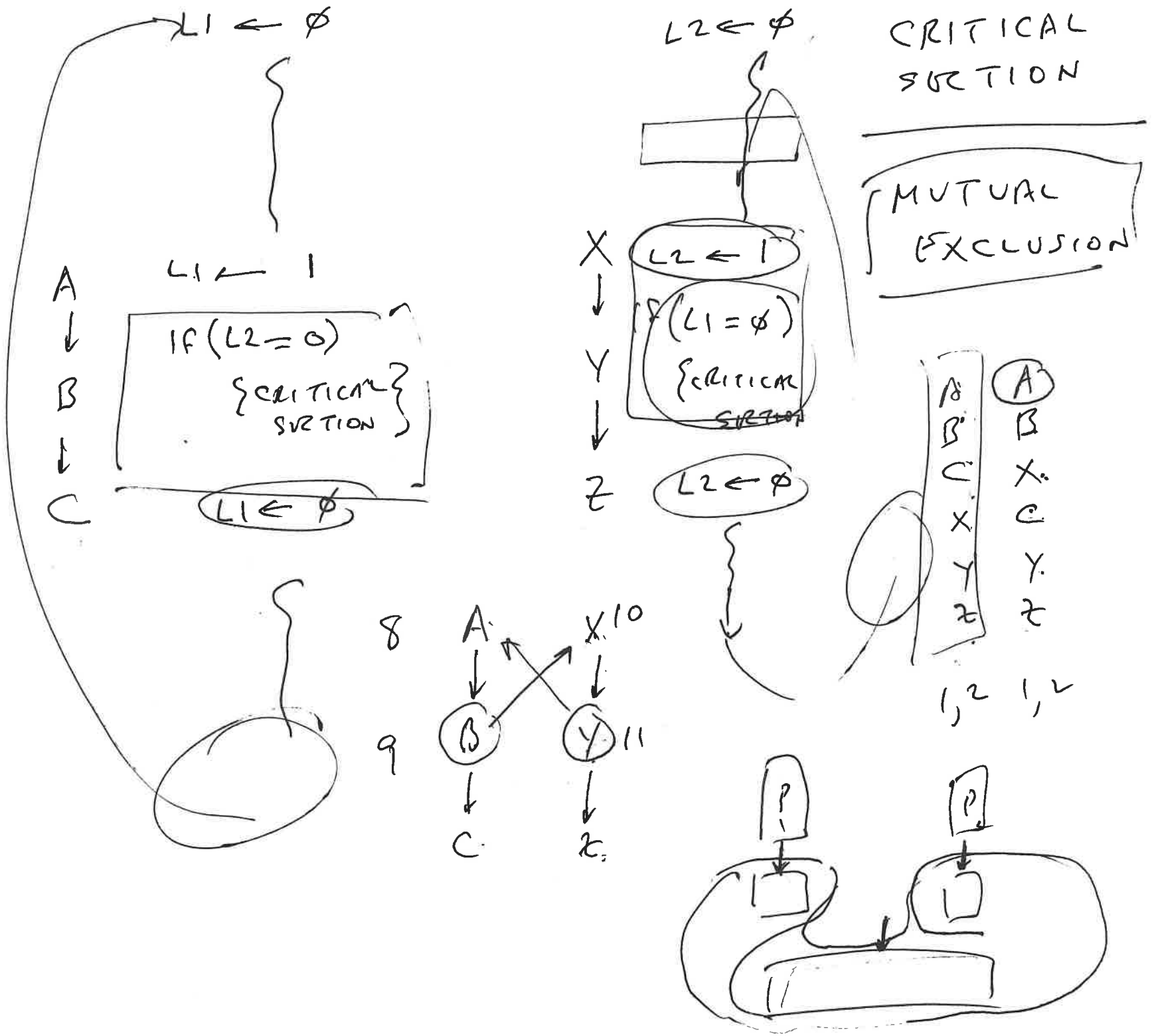
- ***Therefore, having sequential consistency guarantees no concurrent access, i.e., mutual exclusion.***

UPDATE

INVALIDATE






L1 ← ∅
L2 ← ∅
CRITICAL SECTION
MUTUAL EXCLUSION
L1 ← 1
A
B
C
IF (L2 = 0)
{CRITICAL SECTION}
L1 ← ∅
X
Y
Z
L2 ← 1
IF (L1 = ∅)
{CRITICAL SECTION}
L2 ← ∅
A
B
C
X
Y
Z
1,2
1,2
8
9
A
B
C
X 10
Y 11
Z
P1
P2
1
1