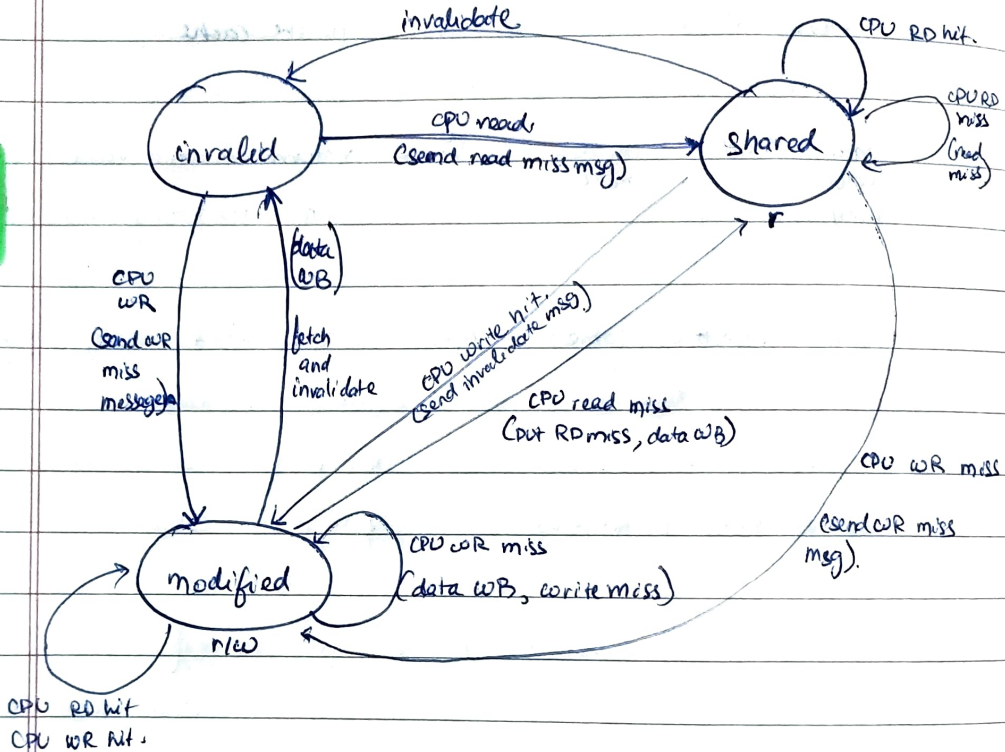
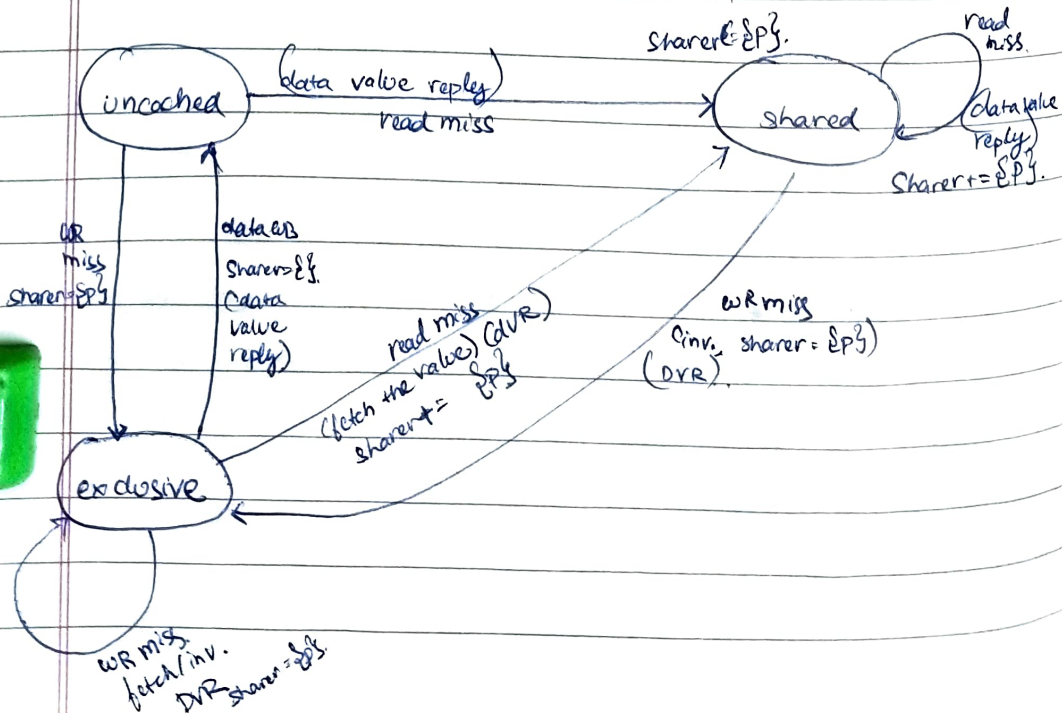


DIRECTORY BASED CACHE COHERENCE

STATE TRANSITION DIAGRAM FOR AN INDIVIDUAL CACHE BLOCK IN A DIRECTORY BASED SYSTEM.



STATE TRANSITION DIAGRAM FOR DIRECTORY.



## SYNCHRONIZATION THE BASICS

synchronization mechanisms are typically built with user-level software, but we have got hardware supplied instructions.

we have

- low contention situations
- high contention situations.

in high contention situations, performance may become a bottleneck. we have a set of basic hardware primitives.

mechanisms

- atomic exchange.

contents of mem loc. and reg. are exchanged without any interruption.

- test and set instructions (PDP-11)

- fetch and increment.

- a pair of instructions

(load result (load locked))

(store conditional)

→ 0 + of 2 inst. exec. atomically

ex try: mov x3, x4 ; move exchange value  
 lr x3, x1 ; load result word  
 sc x3, 0(x1) ; store conditional  
 bnez x3, try ; branch if store fails.  
 mov x4, x2

ex try: lr x2, x1 ; load result 0(x1)  
 addi x3, x2, 1 ; increment  
 sc x3, 0(x1) ; store conditional  
 bnez x3, try ; branch if store fails.

Only register-register instructions can safely be permitted

ex addi x2, R0, #1  
 lockit: EXCH x2, 0(x1) ; atomic exchange  
 bnez x2, lockit

ex lockit: ld x2, 0(x1) ; load of lock  
 bnez x2, lockit ; not available, spin  
 addi x2, R0, #1 ; load locked value  
 EXCH x2, 0(x1) ; swap  
 bnez x2, lockit ; branch if lock was not 0.



MEMORY CONSISTENCY MODEL

ex lockit:  $ld^r$   $x2, 0(x1)$  ; load reserved.

if (  $bnez$   $x2, lockit$  ) ; not available - spin.

addi  $x2, R0, #1$  ; locked value increment

sc  $x2, 0(x1)$  ; store

bnez  $x2, lockit$  ; branch if store fails.

...

...

(compared to lockit)

lockit must be enforced around read and

write to different locations of different processors.

$B = 0$

$A = 0$

...

...

$B = 1$

$A = 1$

if (  $B = 0$  ) ;

if (  $A = 0$  ) ;

write must occur before if condition.

→ sequential consistency

most straightforward model for memory consistency

is called sequential consistency.

(Lamport).

order of execution of the same or through

memory accessed by each processor kept

in order and accessed around different processors

on different processors.