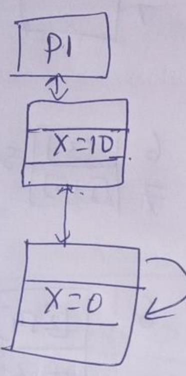
1) Cache coherence:- It defines the behavior of reads and writes to the same memory location. It is one of the issues in multiprocessor systems. Two processors can see different values in their individual cache for the same variable.

A memory system is coherent if

1) 

initially x=0
P1 writes X=10
P1 reads X=10

later x=10 in m.m

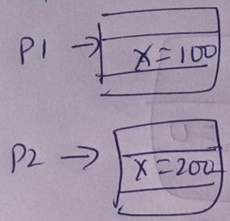
2)

Event	P1	P2	memory contents
P1 reads x	1		1
P2 read x	1	1	
P1 writes x	0	1	0

* P1 writes
  P2 reads (P2 should read updated value)

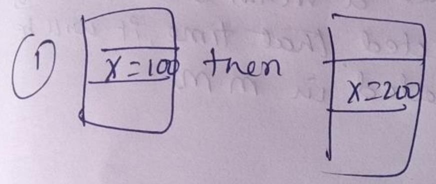
3) P1 ⟶ writes   P2 ⟶ write

i.e X = 10



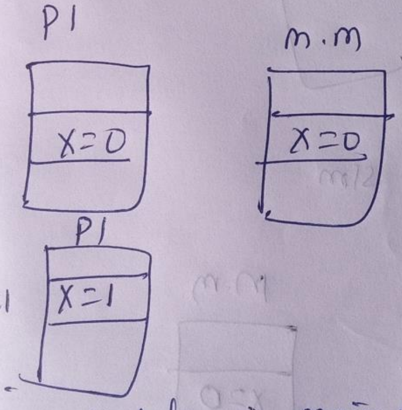
X=100
X=200 both shold be seen in other processor's cache orderwise

P3 → should see in its cache



not as 200 & then 100

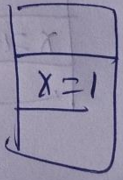
write through cache



P1 reads X at time t0

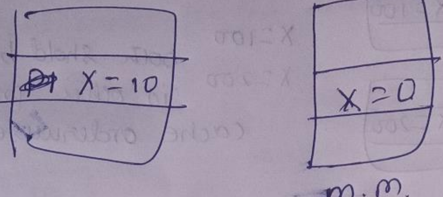
At t1

immediately in main memory also is updated.

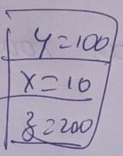


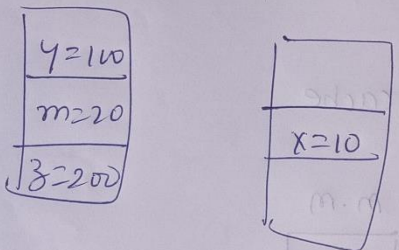
| X = 0 | $\overset{P1}{\text{read}}$ from m.m |

P1 changes at to X=10 in m.m X = 0

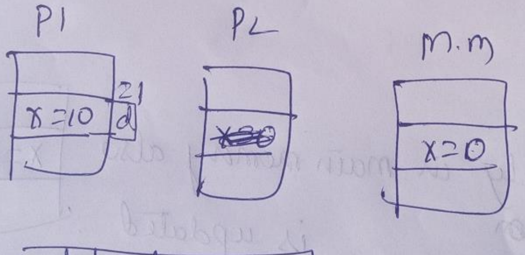


When the cache block is full & new data to be loaded & victim block x=10 is selected that time, it will be updated in m.m



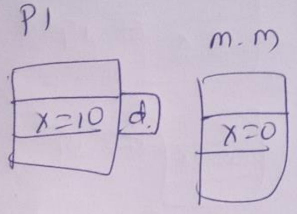


in multiprocessor s/m

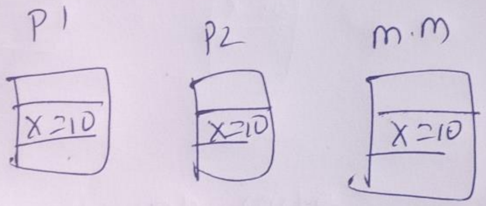


| d | v | data | cache

d=1 means it is modified so in m.m ~~d~~ x value ~~is~~ 0 is aborted / invalidated if P2 wants to read the data, that time updated copy of P1 cache i.e $x=10$ is stored in m.m



P2 has requested x, ~~so~~ and dirty bit of ~~P1~~ $x=10$ is 1 so $x=10$ is copied to m.m


P1
P2
m.m
x=10
x=10
x=10