MEMORY CONSISTENCY MODEL

Cache coherence ensures (memory consistency) that multiple processors see a consistent view of memory.

when a processor must see a value that has been updated by another processor, in what order must a processor observe the data writes of another processor.
(can be used for shared storage?)

what properties must be enforced among read and write to different locations by different processors.

```
P1: A = 0                         P2: B = 0
    ...                               ...
    A = 1                             B = 1
L1: if (B==0)                 L2: if (A==0)
```

↓ only 1 if can enter.

write must come before if condition.

→ sequential consistency.
most straightforward model for memory consistency is called sequential consistency.
(leslie lamport).

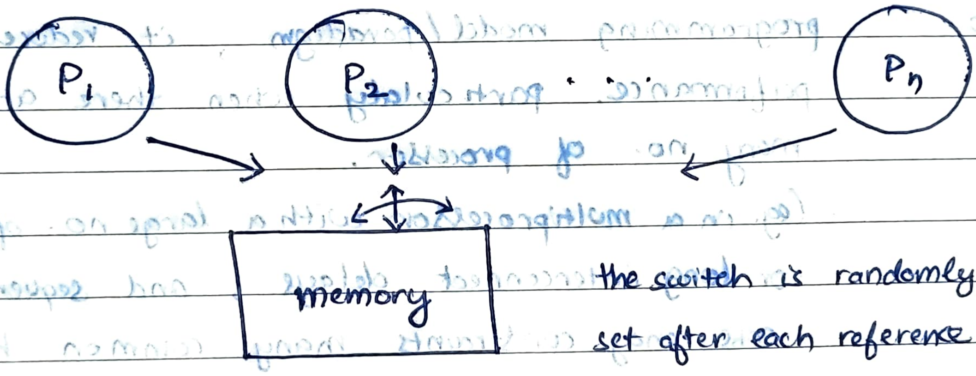
result of any execution be the same as though memory accesses executed by each processor kept in order, and accesses among different processors are arbitrarily interleaved.

SEQUENTIAL CONSISTENCY : LAMPORT

Sarita aldwell Karo Chorlo ieee comp. dec26.

sequential consistency formally defined by leslie lamport. a multiprocessor system is sequentially consistent if the result of any execution is the same if the operations of all the processors are executed in some sequential order, and operations of each individual processor, appear in the sequence in the ~~ordet~~ order specified by the program.

processerors issuing mem ref. as per program order.



the switch is randomly set after each reference

we have synchronization is needed, if we want to enforce atomicity across multiple memory operations from a processor.

sufficient conditions for preserving sequential consistency

① every process issues memory operations in program order.

②. after a write operation is issued, the issuing process waits for the write operation to complete before issuing its next operation.
(program order).

② ~~wr~~ after read operation is issued, the issuing process waits for the read to complete, and the write, whose value is being written by the read op. to complete, and for the write whose value is being returned by the read to complete before issuing next op.
(issues write atomicity)

although sequential consistency presents a simple programming model / paradigm, it reduces potential performance, particulaly when there are very many no. of processor.
(eg. in a multiprocessor with a large no. of processors, or long interconnect delays, and sequential consistency constraints many common hardware and compiler optimizations.)

we have to have the program order requirement and write atomicity.

to provide better performances, 2 approaches have been explored :-

① development of ambitious implementations that result sequential consistency, but use latency hiding techniques to reduce the penalty.

② development of less restrictive memory consistency models, but allow for faster hardware.

$\hookrightarrow$ relaxed memory models.

how they relax program order

we distinguish models based on whether they relax the ordedr from a write to a following read between 2 writes and finally from a read to a following read or write.

in all cases the relaxation only applies to operation pairs with different addresses.

with respect to the write atomicity requirement, we distinguish models based on whether they allow a read to return the value of another processors write, before all cache copies of the access location recieved the invalidation or update message generated by the write.

We have ① relaxed write-to-read program order, ② relaxed write-to-write program order ③ relaxed read-to-read and read-to-write program order, ① read other writes early. ② read own write only.

《writes and reads are 2 different addresses.》