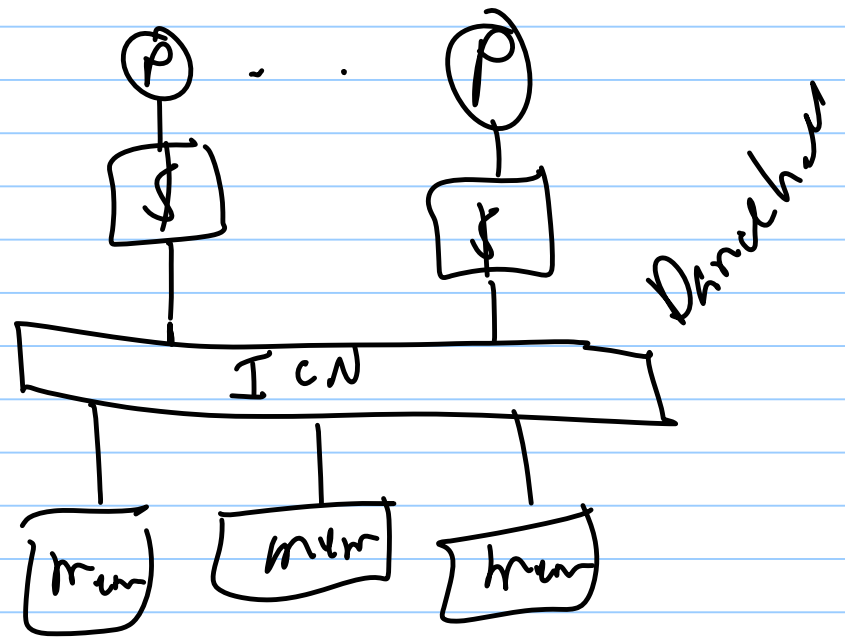
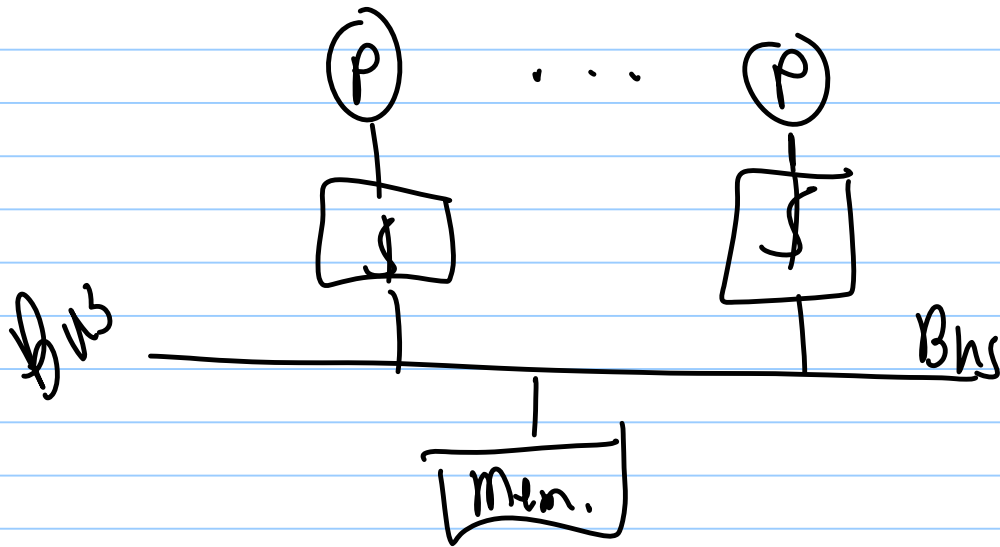
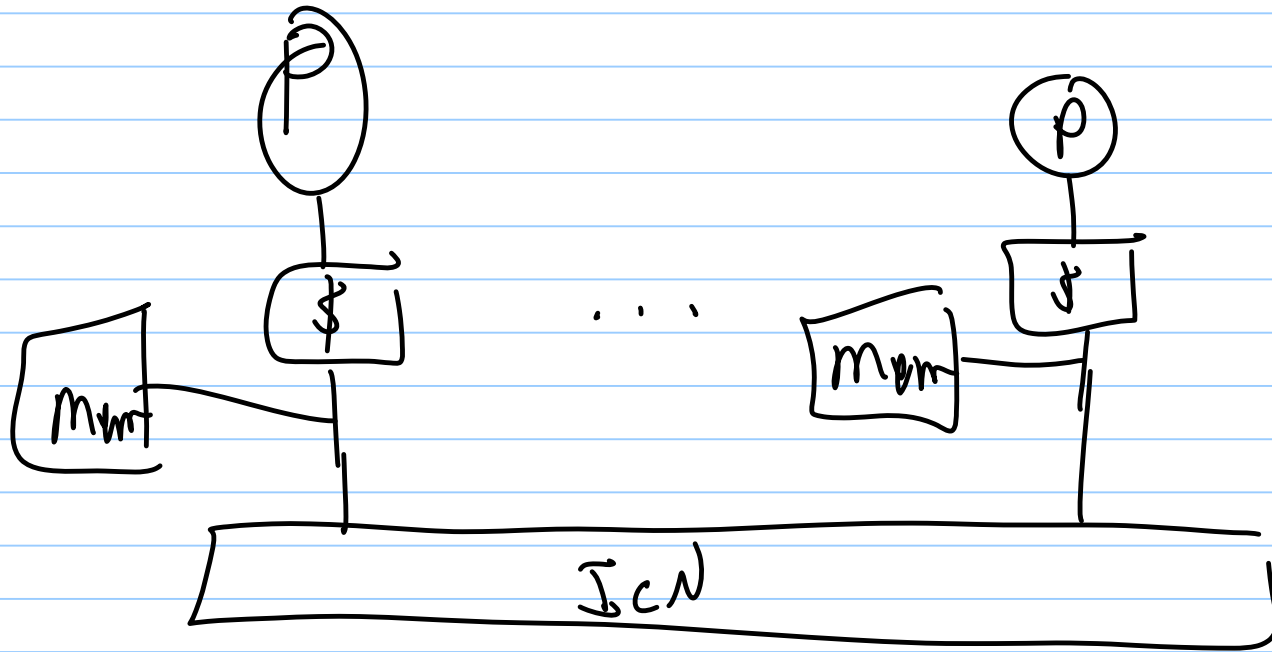


Cache Coherence

memory system of mp





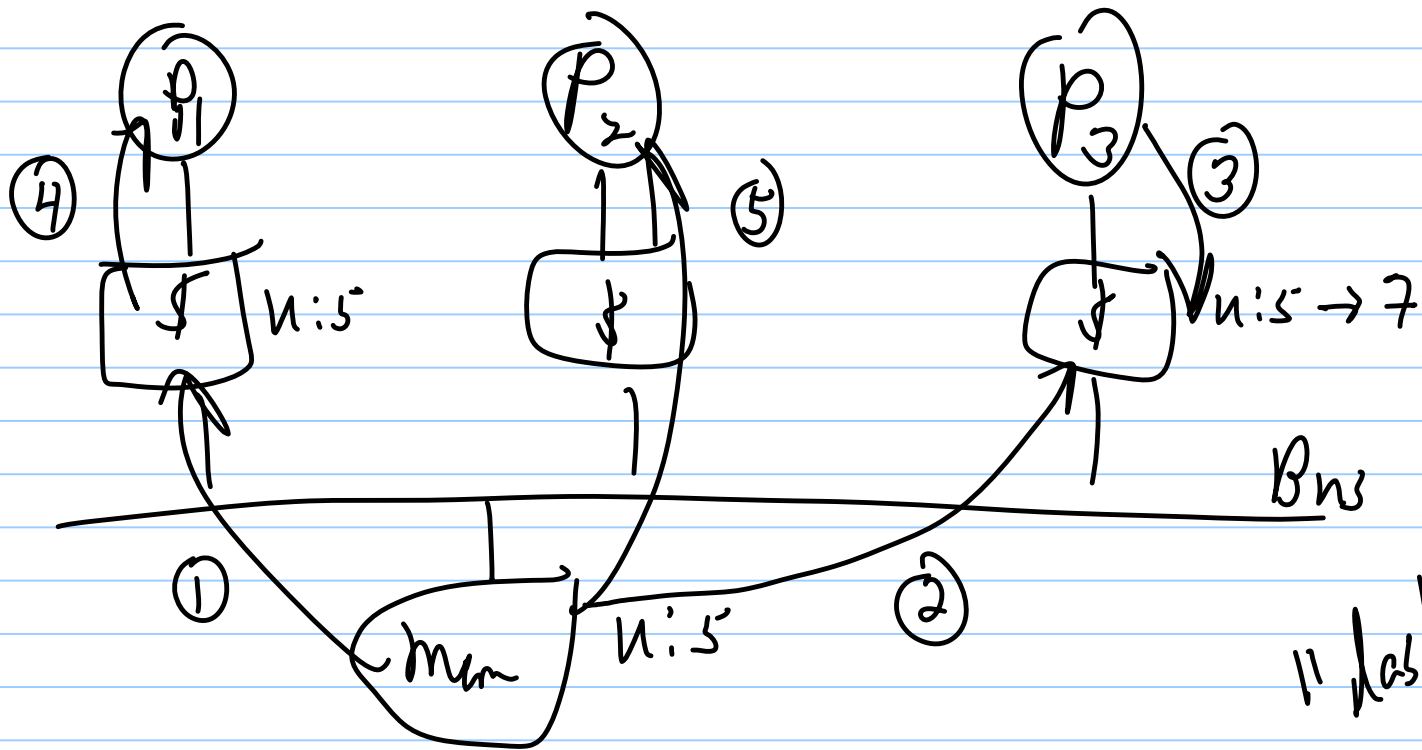
dist. mem.

Reading a doc. should return latest
value written by any process

In uniprocessor easy except for I/O

infrequent
⇒ software mech.
OK

or use uncached
memory



events (4) + (5) are not what we want

"last"
is ill defined

Defs

memory operation : single read (load), write (store), read-modify-write access to a memory location

execute atomically

issue : a mem. op. issues when it leaves the processor's environment and is presented to mem. subsystem

perform : - a write perf. wrt the proc. when a sub. read by the proc. returns the value of that write
- a read perf. wrt the proc. when sub. writes cannot affect the value returned by the read

In multiprocessor replace "the" proc. w/ "a" proc.

w/o caches, memory imposes a total order (serial order) on operations to a location

- ops to that loc. are in prog. order from given processor.
- order of ops from diff processors is some interleaving that preserves indiv. prog. order

results of program : values returned by its read ops

A memory system is coherent if the results of any execution are such that @ each location it is possible to construct hypothetical serial order of all operations to the location in which:

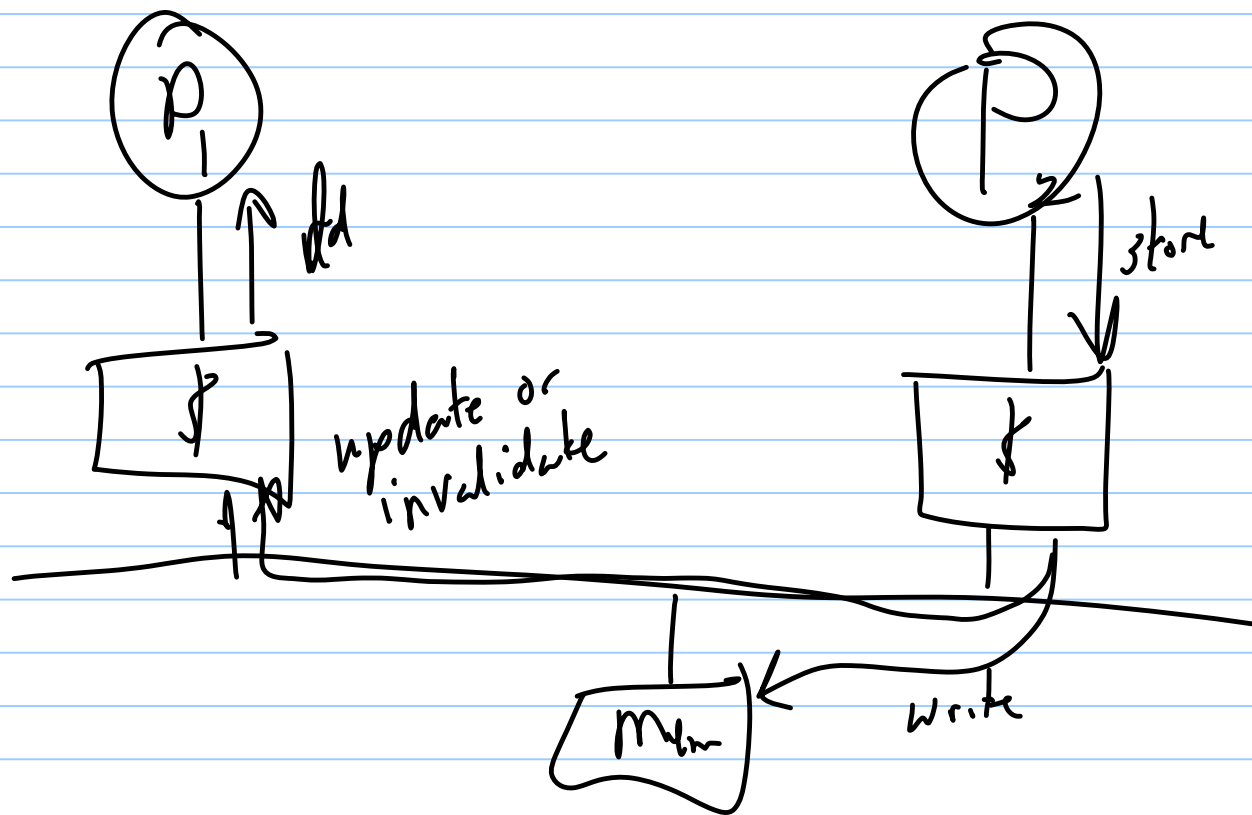
1. ops issued by any process occur in the order issued by that process
2. the value returned by a read is the value written by the last write to that loc. in the serial order

Use bus to build a mechanism for cache coherence

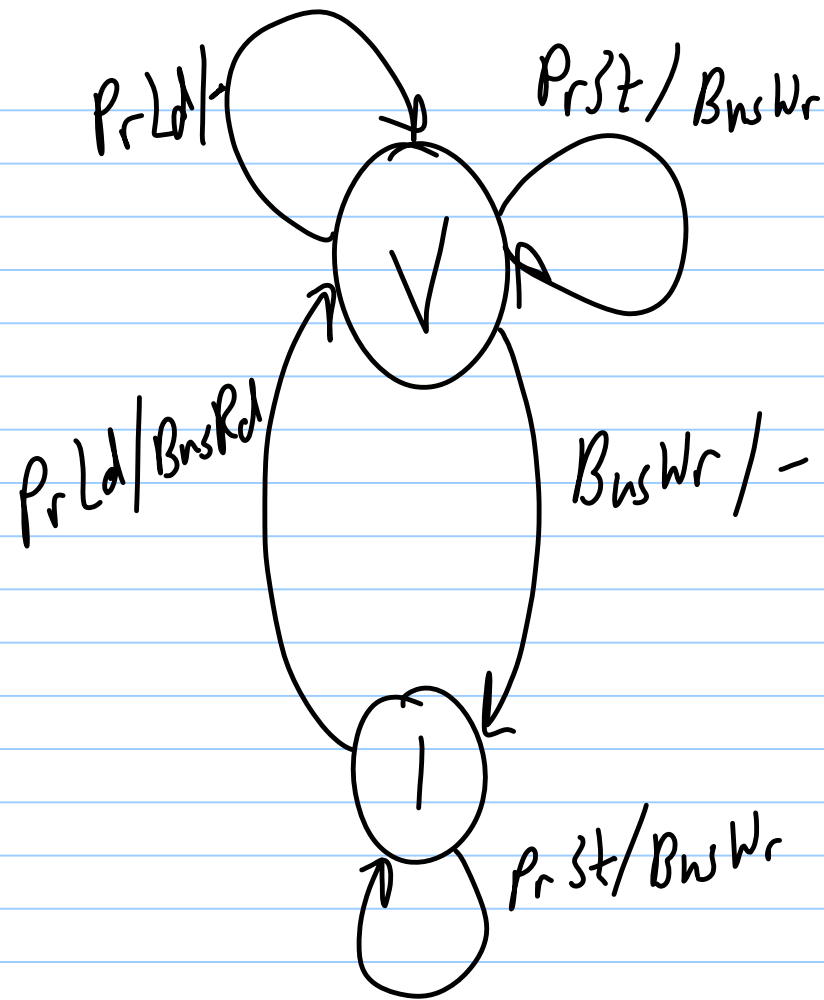
- assume bus operations (transactions) atomic

- assume cache management is via
finite state machine

Snooping-based cache coherence



issues from above
transactions from
below



write-through
no-allocate
invalidate