CS 152 Computer Architecture and Engineering

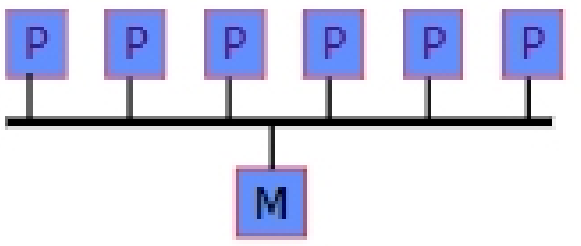
Lecture 20: Snoopy Caches

Krste Asanovic
Electrical Engineering and Computer Sciences
University of California, Berkeley

http://www.eecs.berkeley.edu/~krste
http://inst.cs.berkeley.edu/~cs152

Recap: Sequential Consistency

A Memory Model



" A system is *sequentially consistent* if the result of any execution is the same as if the operations of all the processors were executed in some sequential order, and the operations of each individual processor appear in the order specified by the program"

Leslie Lamport

Sequential Consistency =
arbitrary *order-preserving interleaving*
of memory references of sequential programs

Recap: Sequential Consistency

Sequential consistency imposes more memory ordering constraints than those imposed by uniprocessor program dependencies (⟶)

What are these in our example ?

T1:
- Store (X), 1 *(X = 1)*
- Store (Y), 11 *(Y = 11)*

T2:
- Load R_1, (Y)
- Store (Y'), R_1 *(Y'= Y)*
- Load R_2, (X)
- Store (X'), R_2 *(X'= X)*

⟶ additional SC requirements

Recap: Mutual Exclusion and Locks

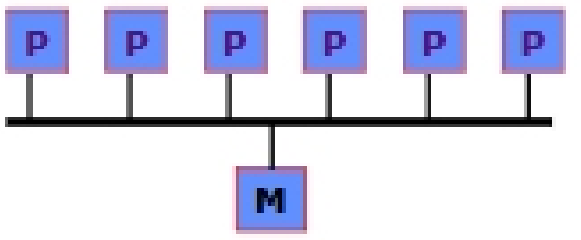
Want to guarantee only one process is active in a critical section

- Blocking atomic read-modify-write instructions
  e.g., Test&Set, Fetch&Add, Swap

  vs
- Non-blocking atomic read-modify-write instructions
  e.g., Compare&Swap, Load-reserve/Store-conditional

  vs
- Protocols based on ordinary Loads and Stores

Issues in Implementing Sequential Consistency



Implementation of SC is complicated by two issues

- *Out-of-order execution capability*

Load(a); Load(b)	*yes*
Load(a); Store(b)	*yes if* a ≠ b
Store(a); Load(b)	*yes if* a ≠ b
Store(a); Store(b)	*yes if* a ≠ b

- *Caches*

  Caches can prevent the effect of a store from being seen by other processors

SC complications motivates architects to consider *weak* or *relaxed* memory models

Memory Fences

Instructions to sequentialize memory accesses

Processors with *relaxed or weak memory models* (i.e., permit Loads and Stores to different addresses to be reordered) need to provide *memory fence* instructions to force the serialization of memory accesses

Examples of processors with relaxed memory models:

Sparc V8 (TSO,PSO): Membar

Sparc V9 (RMO):

Membar #LoadLoad, Membar #LoadStore

Membar #StoreLoad, Membar #StoreStore

PowerPC (WO): Sync, EIEIO

Memory fences are expensive operations, however, one pays the cost of serialization only when it is required