Symmetric Shared-Memory Architectures

- From multiple boards on a shared bus to multiple processors inside a single chip
- Caches both
  - **Private data** that is used by a single processor
  - **Shared data** that is used by multiple processors
- Caching shared data reduces
  - latency to shared data,
  - memory bandwidth for shared data, and
  - interconnect bandwidth

⇒ but **cache coherence problem**

Example Cache **Coherence Problem**

- Processes see different values for `u` after `P_3`'s write
- With write back caches, value written back to memory depends on happenstance of which cache flushes or writes back value when
  - Processes accessing main memory may see **stale value**
  - Unacceptable for programming, and it happens frequently!
Example

P1

/*Assume initial value of A and flag is 0*/
A = 1; while (flag == 0); /*spin idly*/
flag = 1; print A;

P2

• Intuition not guaranteed by coherence
• Expect memory to respect order between accesses to different locations issued by a given process
  – to preserve orders among accesses to same location by different processes
• Coherence is not enough!
  – pertains only to single location

Intuitive Memory Model

• Reading an address should return the last value written to that address
  – Easy in uniprocessors, except for I/O
• Too vague and simplistic; 2 issues
  1. Coherence defines values returned by a read
  2. Consistency determines when a written value will be returned by a read
• Coherence defines behavior to same location, Consistency defines behavior to other locations