Bus-Based Multiprocessors

Shared Memory Model:

\[ P_1 \rightarrow P_2 \rightarrow P_3 \rightarrow P_4 \rightarrow \ldots \]

Shared Memory

Shared Memory Implementation:

\[ \text{CPU core} \quad \text{CPU core} \]

\[ \text{Physical Memory (DRAMs)} \quad \text{Memory Controller} \]

\[ \text{Bridge} \quad \text{to I/O subsystem} \]
Cache Coherence Problem

(Assume writeback caches)

1. \( P_1 \) performs \( l d \ A \)
2. \( P_1 \) performs \( st \ A \)
   \[ \Rightarrow A \text{ is dirty in } P_1 \text{'s cache} \]
3. \( P_2 \) performs \( l d \ A \)

\[ \Rightarrow P_2 \text{ loads stale copy of } A \text{ from memory.} \]

(Assume write through caches)

1. \( P_1 \) performs \( l d \ A \)
2. \( P_2 \) performs \( l d \ A \)
3. \( P_1 \) performs \( st \ A \)
4. \( P_2 \) performs \( l d \ A \)

\[ \Rightarrow P_2 \text{ loads stale copy of } A \text{ from cache.} \]
Memory Consistency Model

⇒ When sequential consistency is broken.

Sequentially consistent order:

- \( P_1 \) writes \( A = 1 \)
- \( P_1 \) writes \( X = 1 \)
- \( P_2 \) reads \( X = 1 \)
- \( P_2 \) reads \( A = 1 \)
- \( P_2 \) writes \( B = A = 1 \)

⇒ Caches reorder writes.
- \( P_1 \) order: \( A = 1 \), \( X = 1 \)
- \( P_2 \) order: \( X = 1 \), \( A = 1 \)

Assume write back caches:

1. \( P_1 \) writes \( A = 1 \)
2. \( P_1 \) writes \( X = 1 \)
3. \( X \) is written back to memory
4. \( P_2 \) reads \( X = 1 \)
5. \( P_2 \) reads \( A = 0 \)
6. \( P_3 \) writes \( B = A = 0 \)
Bus-Based Write Invalidate