Review: Coherence vs. Consistency

- Intuition says loads should return latest value
  - what is latest?
- Coherence concerns only one memory location
- Consistency concerns apparent ordering for all locations
- A Memory System is Coherent if
  - can serialize all operations to that location such that,
  - operations performed by any processor appear in program order
    » program order = order defined program text or assembly code
  - value returned by a read is value written by last store to that location
Review: Why Coherence $\neq$ Consistency

/* initial A = B = flag = 0 */

P1
A = 1;
B = 1;
flag = 1;

P2
while (flag == 0); /* spin */
print A;
print B;

Intuition says printed A = B = 1
Coherence doesn’t say anything, why?

Review: Sequential Consistency (SC)

sequential processors issue memory ops in program order

P1

P2

P3

switch randomly set after each memory op provides single sequential order among all operations

Memory
Review: Sufficient Conditions for SC

- Every processor issues memory ops in program order
- Processor must wait for store to complete before issuing next memory operation
- After load, issuing proc waits for load to complete, and store that produced value to complete before issuing next op
- Easily implemented with shared bus.

Relaxed Memory Models

- Motivation with Directory Protocols
  - Misses have longer latency (do cache hits to get to next miss)
  - Collecting acknowledgements can take even longer

- Recall SC has
  - Each processor generates at total order of its reads and writes
    (R→R, R→W, W→W, & R→W)
  - That are interleaved into a global total order

- (Most) Relaxed Models
- PC: Relax ordering from writes to (other proc’s) reads
- RC: Relax all read/write orderings (but add “fences”)
/* initial A = B = 0 */

\[
P1 \\
P2
\]

A = 1;
B = 1
r1 = B;
r2 = A;

Processor Consistent (PC) Models
Allow \( r1 = r2 = 0 \) (precluded by SC)
Examples: IBM 370, Sun TSO, & Intel IA-32

Why do this?
Allows FIFO write buffers
Does not astonish programmers (very much)

---

Write Buffers w/ Read Bypass

![Diagram of Write Buffers with Read Bypass](image)

P1
Flag 1 = 1
if (Flag 2 == 0)
critical section

P2
Flag 2 = 1
if (Flag 1 == 0)
critical section
Also Want “Causality”

/* initially all 0 */

\[
\begin{align*}
&\text{P1} & \text{P2} & \text{P3} \\
A &= 1; & \text{while (flag1==0)} & \text{while (flag2==0)} \\
\text{flag1} &= 1; & \text{flag2} &= 1; & r3 &= A;
\end{align*}
\]

All commercial models guarantee causality

Why Not Relax All Order?

/* initially all 0 */

\[
\begin{align*}
&\text{P1} & \text{P2} \\
A &= 1; & \text{while (flag == 0)} & /* spin */ \\
B &= 1; & r1 &= A; \\
\text{flag} &= 1; & r2 &= B;
\end{align*}
\]

Reorder of “A = 1”/“B = 1” or “r1 = A”/“r2 = B”
Via OOO processor, non-FIFO write buffers, delayed directory acknowledgements, etc.
But Must Order
“A = 1”/“B = 1” before “flag =1”
“flag != 0” before “r1 = A”/“r2 = B”
Order with “Synch” Operations

/* initially all 0 */

P1
A = 1;
B = 1;
SYNCH flag = 1;

while (SYNCH flag == 0);
r1 = A;
r2 = B;

Called “weak ordering” of “weak consistency” (WC)
Alternatively, relaxed consistency (RC) specializes

Acquires: force subsequent reads/writes after
Releases: force previous reads/writes before

Weak Ordering Example
Release Consistency Example

Review: Directory Example Sequential Consistency
Directory Example Release Consistency

(Please refer to the diagram for the detailed chart and its explanation.)

Commercial Models use “Fences”

/* initially all 0 */

\[
P1
\]

\[
P1
\]

A = 1; while (flag == 0);  
B = 1; FENCE;  
FENCE; r1 = A;  
flag = 1; r2 = B;

Examples: Compaq Alpha, IBM PowerPC, & Sun RMO

Can specialize fences (e.g., RMO)
The Programming Interface

- WO and RC require synchronized programs
- All synchronization operations must be labeled and visible to the hardware
  - easy if synchronization library used
  - must provide language support for arbitrary ld/st synchronization (event notification, e.g., flag)
- Program that is correct for TSO portable to WO & RC