CPEG 852 — Advanced Topics in Computing Systems
Memory Models
Current And Future Models

Pouya Fotouhi Stéphane Zuckerman

Computer Architecture & Parallel Systems Laboratory
Electrical & Computer Engineering Dept.
University of Delaware
140 Evans Hall Newark,DE 19716, United States
{pfotouhi,szuckerm}@udel.edu

October 13, 2015

Outline

Outline

Uniform vs. Non-Uniform Memory Consistency Models I

Overview

Uniform Memory Models

- No differentiation between memory operations, except for the `LOAD`/`STORE` distinction.
- Works well as long as everyone always reads everything from memory...
- Sequential Consistency (SC), Cache Consistency (Coherence), Pipelined RAM (P-RAM), and Processor Consistency (PC) are examples of uniform models

Non-Uniform Memory Models

- Differentiate between ordinary and synchronizing memory operations
- Ordinary operations:
  - From the viewpoint of the issuing processor, the operations it issued follow the program order, at least from a dependence point of view.
  - Other processors' ordinary operations can be seen in any order.
- Synchronization operations follow sequential consistency w.r.t. each other.
  - In other words: there is a total order between sync ops.
- Note: there may be authorized overlapping between ordinary and synchronizing operations.

Outline

Outline

The SPARC Processor Architecture

Origins

- Processor designed by Sun Microsystems (now Oracle)
- Started in the mid 80's
- RISC processor

SPARC Nowadays

- Used in the K-Computer: #4 in the Top500 supercomputer ranking
  - See `http://www.top500.org`
- 705,024 cores based on SPARC64 VIIIfx micro-architecture

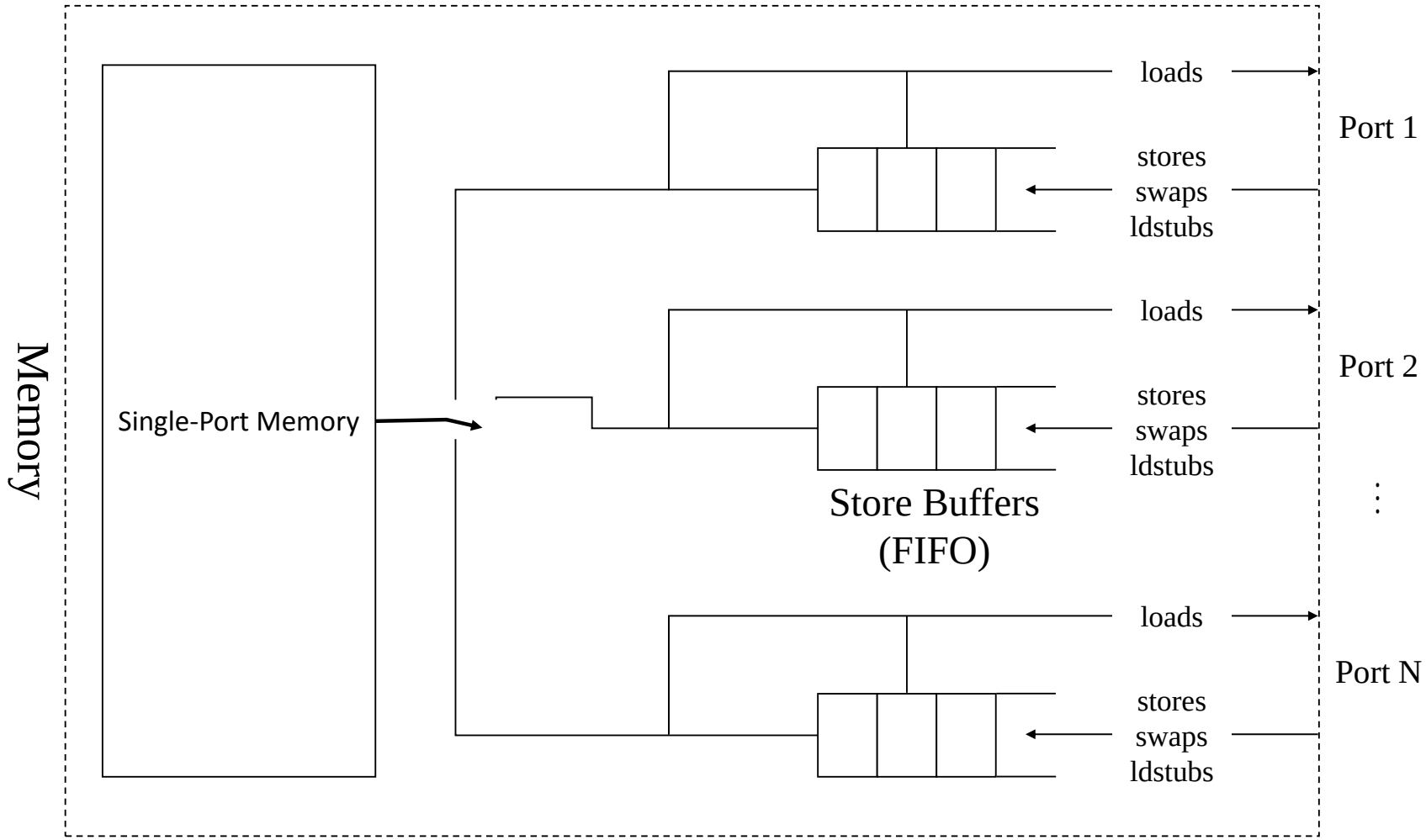
Total Store Ordering (TSO)

Used in SPARC v8, full compatibility in SPARC v9

- Operations: Store, FLUSH, atomic loads & stores
  - Appears to be executed serially in a single order called the *memory order*
  - The order of memory operations is identical to their issuing order (for a given processor)
- Sequentially consistent (SC) for Store, FLUSH, and atomic load/store operations

Total Store Ordering (TSO)

Model of Memory



Recap: P-RAM

- P-RAM is achieved if
  - All memory operations follow program order (w.r.t. process P)
  - All memory writes appear in some order to all processors of the system, but all memory writes issued by the same process P are in-order.

Recap: P-RAM

- P-RAM is achieved if
  - All memory operations follow program order (w.r.t. process P)
  - All memory writes appear in some order to all processors of the system, but all memory writes issued by the same process P are in-order.

Recap: TSO

- Total order on Store, FLUSH, and atomic load/store operations
  - Appear to be executed serially in a single order called the *memory order*
  - The order of memory operations is identical to their issuing order (for a given processor)

Recap: P-RAM

- P-RAM is achieved if
  - All memory operations follow program order (w.r.t. process P)
  - All memory writes appear in some order to all processors of the system, but all memory writes issued by the same process P are in-order.

Recap: TSO

- Total order on Store, FLUSH, and atomic load/store operations
  - Appear to be executed serially in a single order called the *memory order*
  - The order of memory operations is identical to their issuing order (for a given processor)

Recap: P-RAM

- P-RAM is achieved if
  - All memory operations follow program order (w.r.t. process P)
  - All memory writes appear in some order to all processors of the system, but all memory writes issued by the same process P are in-order.

Recap: TSO

- Total order on Store, FLUSH, and atomic load/store operations
  - Appear to be executed serially in a single order called the *memory order*
  - The order of memory operations is identical to their issuing order (for a given processor)

Total Store Ordering (TSO)
P-RAM vs.TSO

- TSO *may* look similar to P-RAM, but TSO enforces a *total order* on writes
- Not clear yet?! Let's see an example...

Total Store Ordering (TSO)

Example 0

Note:
Initial values are equal to zero unless stated otherwise.

Total Store Ordering (TSO)

Example 0

Thread 0	Thread 1	Thread 2	Thread 3
$x \leftarrow 1$	$\cdots \leftarrow x,\ x = 1$	$\cdots \leftarrow x,\ x = 1$	$\cdots \leftarrow x,\ x = 2$
	$x \leftarrow 2$	$\cdots \leftarrow x,\ x = 2$	$\cdots \leftarrow x,\ x = 1$

Table: Initially, $x = y = 0$. Can this program trace be P-RAM?

Total Store Ordering (TSO)

Example 0

Thread 0	Thread 1	Thread 2	Thread 3
$x \leftarrow 1$	$\cdots \leftarrow x, x = 1$	$\cdots \leftarrow x, x = 1$	$\cdots \leftarrow x, x = 2$
	$x \leftarrow 2$	$\cdots \leftarrow x, x = 2$	$\cdots \leftarrow x, x = 1$

Table: Initially, $x = y = 0$. Can this program trace be P-RAM?

Yes!

T2 $\cdots, x \leftarrow 1, x \leftarrow 2$

T3 $\cdots, x \leftarrow 2, x \leftarrow 1$

Total Store Ordering (TSO)

Example 0

Thread 0	Thread 1	Thread 2	Thread 3
$x \leftarrow 1$	$\cdots \leftarrow x,\ x = 1$	$\cdots \leftarrow x,\ x = 1$	$\cdots \leftarrow x,\ x = 2$
	$x \leftarrow 2$	$\cdots \leftarrow x,\ x = 2$	$\cdots \leftarrow x,\ x = 1$

Table: Initially, $x = y = 0$. Can this program trace be TSO?

Total Store Ordering (TSO)

Example 0

Thread 0	Thread 1	Thread 2	Thread 3
$x \leftarrow 1$	$\cdots \leftarrow x,\ x = 1$	$\cdots \leftarrow x,\ x = 1$	$\cdots \leftarrow x,\ x = 2$
	$x \leftarrow 2$	$\cdots \leftarrow x,\ x = 2$	$\cdots \leftarrow x,\ x = 1$

Table: Initially, $x = y = 0$. Can this program trace be TSO?

No!

There is no *total order* on stores to be found.

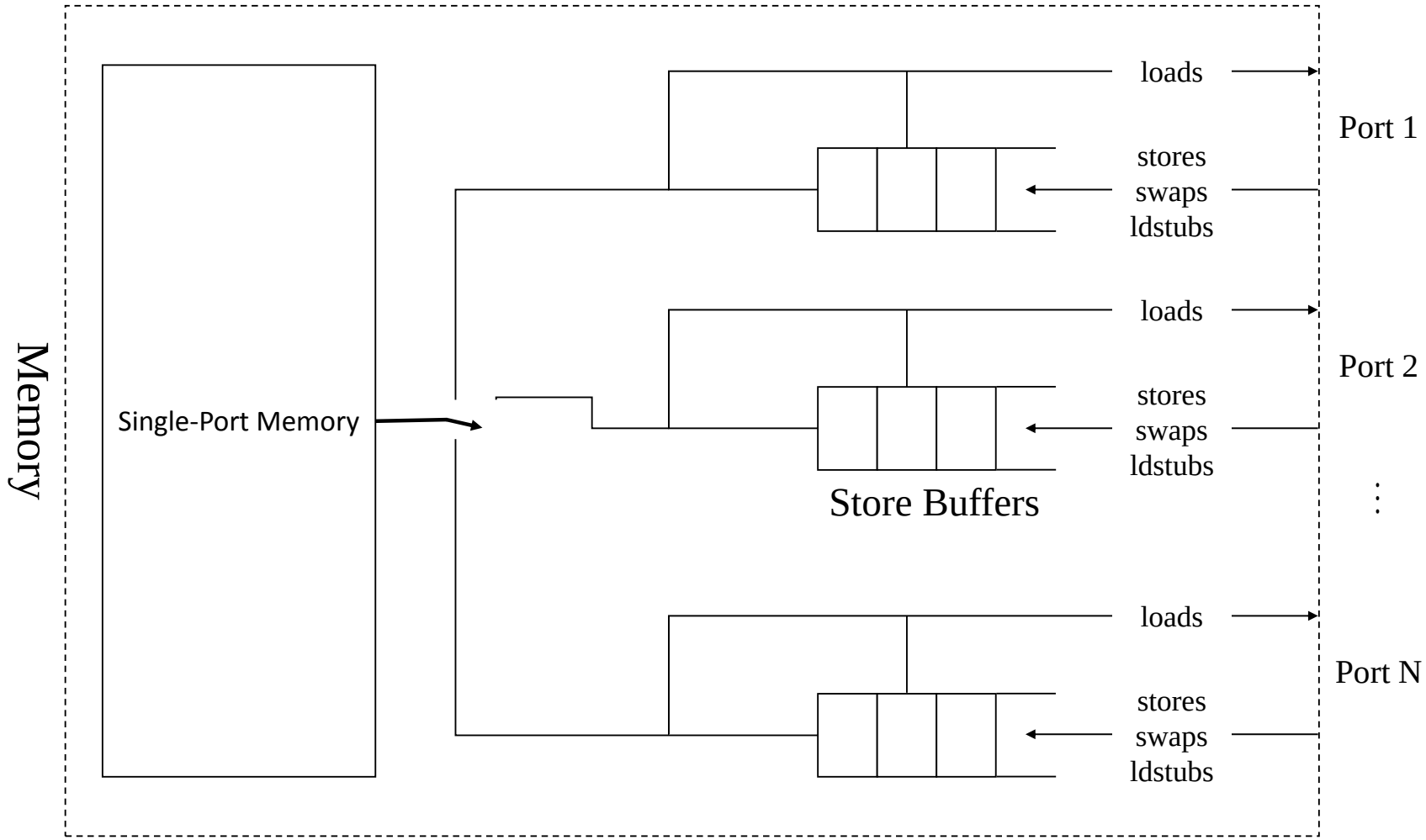
Partial Store Ordering (PSO)

Used in SPARC v8, full compatibility in SPARC v9

- Operations: Store, FLUSH, atomic loads & stores
  - Appears to be executed serially in a single order called the *memory order*
  - The order of memory operations is not necessarily identical to their issuing order (for a given processor)
- Sequentially consistent (SC) for Store, FLUSH, and atomic load/store operations

Partial Store Ordering (PSO)

Model of Memory



Example 1

Total Store Ordering (TSO)

Thread 0	Thread 1	Thread 2
$x \leftarrow 1$	$r_0 \leftarrow y$	$r_1 \leftarrow z$
$y \leftarrow 1$	$z \leftarrow r_0$	$r_2 \leftarrow x$

Table: Initially, $x = y = 0$. Can this program trace yield $r_1 = 1 \wedge r_2 = 0$ under TSO?

Example 1

Total Store Ordering (TSO)

Thread 0	Thread 1	Thread 2
$x \leftarrow 1$	$r_0 \leftarrow y$	$r_1 \leftarrow z$
$y \leftarrow 1$	$z \leftarrow r_0$	$r_2 \leftarrow x$

Table: Initially, $x = y = 0$. Can this program trace yield $r_1 = 1 \wedge r_2 = 0$ under TSO?

No!

There is no *total order* on stores to be found.

Example 1

Partial Store Ordering (PSO)

Thread 0	Thread 1	Thread 2
$x \leftarrow 1$	$r_0 \leftarrow y$	$r_1 \leftarrow z$
$y \leftarrow 1$	$z \leftarrow r_0$	$r_2 \leftarrow x$

Table: Initially, $x = y = 0$. Can this program trace yield $r_1 = 1 \wedge r_2 = 0$ under PSO?

Example 1

Partial Store Ordering (PSO)

Thread 0	Thread 1	Thread 2
$x \leftarrow 1$	$r_0 \leftarrow y$	$r_1 \leftarrow z$
$y \leftarrow 1$	$z \leftarrow r_0$	$r_2 \leftarrow x$

Table: Initially, $x = y = 0$. Can this program trace yield $r_1 = 1 \wedge r_2 = 0$ under PSO?

Yes!

$$y \leftarrow 1 \quad \Rightarrow \quad r_0 \leftarrow y \quad \Rightarrow \quad z \leftarrow r_0 \quad \Rightarrow \quad r_1 \leftarrow z \quad \Rightarrow \quad r_2 \leftarrow x \quad \Rightarrow \quad x \leftarrow 1$$

Example 1

Partial Store Ordering (PSO)

Thread 0	Thread 1	Thread 2
$x \leftarrow 1$	$r_0 \leftarrow y$	$r_1 \leftarrow z$
$y \leftarrow 1$	$z \leftarrow r_0$	$r_2 \leftarrow x$

Table: Initially, $x = y = 0$. Can this program trace yield $r_1 = 1 \wedge r_2 = 0$ under PSO?

Yes!

$$\underbrace{y \leftarrow 1 \quad \Rightarrow \quad r_0 \leftarrow y \quad \Rightarrow \quad z \leftarrow r_0 \quad \Rightarrow \quad r_1 \leftarrow z \quad \Rightarrow \quad r_2 \leftarrow x \quad \Rightarrow \quad x \leftarrow 1}_{r_0 = 1}$$

Example 1

Partial Store Ordering (PSO)

Thread 0	Thread 1	Thread 2
$x \leftarrow 1$	$r_0 \leftarrow y$	$r_1 \leftarrow z$
$y \leftarrow 1$	$z \leftarrow r_0$	$r_2 \leftarrow x$

Table: Initially, $x = y = 0$. Can this program trace yield $r_1 = 1 \wedge r_2 = 0$ under PSO?

Yes!

$$y \leftarrow 1 \quad \Rightarrow \quad r_0 \leftarrow y \quad \Rightarrow \quad z \leftarrow r_0 \quad \Rightarrow \quad r_1 \leftarrow z \quad \Rightarrow \quad r_2 \leftarrow x \quad \Rightarrow \quad x \leftarrow 1$$

$r_0 = 1$ $\qquad$ $r_1 = 1$

Example 1

Partial Store Ordering (PSO)

Thread 0	Thread 1	Thread 2
$x \leftarrow 1$	$r_0 \leftarrow y$	$r_1 \leftarrow z$
$y \leftarrow 1$	$z \leftarrow r_0$	$r_2 \leftarrow x$

Table: Initially, $x = y = 0$. Can this program trace yield $r_1 = 1 \wedge r_2 = 0$ under PSO?

Yes!

$$y \leftarrow 1 \quad \Rightarrow \quad r_0 \leftarrow y \quad \Rightarrow \quad z \leftarrow r_0 \quad \Rightarrow \quad r_1 \leftarrow z \quad \Rightarrow \quad r_2 \leftarrow x \quad \Rightarrow \quad x \leftarrow 1$$

$$r_0 = 1 \qquad r_1 = 1 \qquad r_2 = 0$$

Example 2

Total Store Ordering (TSO)

Thread 0	Thread 1
$x \leftarrow 1$	$r_1 \leftarrow y$
$r_0 \leftarrow x$	$r_2 \leftarrow x$
$y \leftarrow r_0$	

Table: Initially, $x = y = 0$. Can this program trace yield $r_1 = 1 \wedge r_2 = 0$ under TSO?

Example 2

Total Store Ordering (TSO)

Thread 0	**Thread 1**
$x \leftarrow 1$	$r_1 \leftarrow y$
$r_0 \leftarrow x$	$r_2 \leftarrow x$
$y \leftarrow r_0$	

Table: Initially, $x = y = 0$. Can this program trace yield $r_1 = 1 \wedge r_2 = 0$ under TSO?

No!

There is no *total order* on stores to be found.

Example 2

Partial Store Ordering (PSO)

Thread 0	Thread 1
$x \leftarrow 1$	$r_1 \leftarrow y$
$r_0 \leftarrow x$	$r_2 \leftarrow x$
$y \leftarrow r_0$	

Table: Initially, $x = y = 0$. Can this program trace yield $r_1 = 1 \wedge r_2 = 0$ under PSO?

Example 2

Partial Store Ordering (PSO)

Thread 0	Thread 1
$x \leftarrow 1$	$r_1 \leftarrow y$
$r_0 \leftarrow x$	$r_2 \leftarrow x$
$y \leftarrow r_0$	

Table: Initially, $x = y = 0$. Can this program trace yield $r_1 = 1 \wedge r_2 = 0$ under PSO?

Yes!

$$r_0 \leftarrow x \quad \Rightarrow \quad y \leftarrow r_0 \quad \Rightarrow \quad r_1 \leftarrow y \quad \Rightarrow \quad r_2 \leftarrow x \quad \Rightarrow \quad x \leftarrow 1$$

Example 2

Partial Store Ordering (PSO)

Thread 0	Thread 1
$x \leftarrow 1$	$r_1 \leftarrow y$
$r_0 \leftarrow x$	$r_2 \leftarrow x$
$y \leftarrow r_0$	

Table: Initially, $x = y = 0$. Can this program trace yield $r_1 = 1 \wedge r_2 = 0$ under PSO?

Yes!

$$r_0 \leftarrow x \quad \Rightarrow \quad y \leftarrow r_0 \quad \Rightarrow \quad r_1 \leftarrow y \quad \Rightarrow \quad r_2 \leftarrow x \quad \Rightarrow \quad x \leftarrow 1$$

$r_0 = 1$

Example 2

Partial Store Ordering (PSO)

Thread 0	Thread 1
$x \leftarrow 1$	$r_1 \leftarrow y$
$r_0 \leftarrow x$	$r_2 \leftarrow x$
$y \leftarrow r_0$	

Table: Initially, $x = y = 0$. Can this program trace yield $r_1 = 1 \wedge r_2 = 0$ under PSO?

Yes!

$$r_0 \leftarrow x \quad \Rightarrow \quad y \leftarrow r_0 \quad \Rightarrow \quad r_1 \leftarrow y \quad \Rightarrow \quad r_2 \leftarrow x \quad \Rightarrow \quad x \leftarrow 1$$

$r_0 = 1$ $\qquad$ $r_1 = 1$

Example 2

Partial Store Ordering (PSO)

Thread 0	Thread 1
$x \leftarrow 1$	$r_1 \leftarrow y$
$r_0 \leftarrow x$	$r_2 \leftarrow x$
$y \leftarrow r_0$	

Table: Initially, $x = y = 0$. Can this program trace yield $r_1 = 1 \wedge r_2 = 0$ under PSO?

Yes!

$$\underbrace{r_0 \leftarrow x}_{r_0 = 1} \Rightarrow y \leftarrow r_0 \Rightarrow \underbrace{r_1 \leftarrow y}_{r_1 = 1} \Rightarrow \underbrace{r_2 \leftarrow x}_{r_2 = 0} \Rightarrow x \leftarrow 1$$

Relaxed Memory Ordering (RMO)

Used in SPARC v9

A system complies with RMO if:

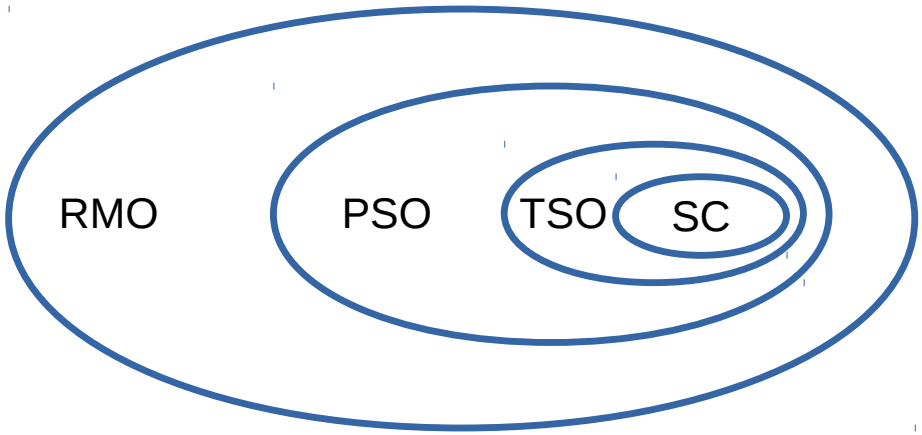
Conditions to Follow RMO

- A processor's memory operations follow *self-consistency*
- No specific ordering for load-stores otherwise
- To enforce a specific order, the user/programmer must use `MEMBAR` operations
  - `MEMBAR StoreLoad` (available in TSO)
  - `MEMBAR StoreStore` (implied in TSO, available in PSO)
  - `MEMBAR LoadStore` (implied in TSO and PSO)
  - `MEMBAR LoadLoad` (implied in TSO and PSO)

Definition: Self-Consistency

A self-consistent execution trace is one that generates precisely the same results as those produced by a program order execution trace.

TSO, PSO, and RMO



Example 3

Thread 0	Thread 1	Thread 2
$x \leftarrow 1$	$r_0 \leftarrow x$	$r_2 \leftarrow y$
$y \leftarrow 1$	$r_1 \leftarrow y$	$r_3 \leftarrow x$

Table: Initially, $x = y = 0$. Can this program trace yield $r_0 = 1 \wedge r_1 = 1 \wedge r_2 = 1 \wedge r_3 = 1$ under TSO? PSO? RMO?

Example 3

Thread 0	Thread 1	Thread 2
$x \leftarrow 1$	$r_0 \leftarrow x$	$r_2 \leftarrow y$
$y \leftarrow 1$	$r_1 \leftarrow y$	$r_3 \leftarrow x$

Table: Initially, $x = y = 0$. Can this program trace yield $r_0 = 1 \wedge r_1 = 1 \wedge r_2 = 1 \wedge r_3 = 1$ under TSO? PSO? RMO?

TSO:Yes!

Example 3

Thread 0	Thread 1	Thread 2
$x \leftarrow 1$	$r_0 \leftarrow x$	$r_2 \leftarrow y$
$y \leftarrow 1$	$r_1 \leftarrow y$	$r_3 \leftarrow x$

Table: Initially, $x = y = 0$. Can this program trace yield $r_0 = 1 \wedge r_1 = 1 \wedge r_2 = 1 \wedge r_3 = 1$ under TSO? PSO? RMO?

TSO:Yes!

PSO:Yes!

Example 3

Thread 0	Thread 1	Thread 2
$x \leftarrow 1$	$r_0 \leftarrow x$	$r_2 \leftarrow y$
$y \leftarrow 1$	$r_1 \leftarrow y$	$r_3 \leftarrow x$

Table: Initially, $x = y = 0$. Can this program trace yield $r_0 = 1 \wedge r_1 = 1 \wedge r_2 = 1 \wedge r_3 = 1$ under TSO? PSO? RMO?

TSO:Yes! PSO:Yes! RMO:Yes!

$$x \leftarrow 1 \quad \Rightarrow \quad y \leftarrow 1 \quad \Rightarrow \quad r_0 \leftarrow x \quad \Rightarrow \quad r_1 \leftarrow y \quad \Rightarrow \quad r_2 \leftarrow y \quad \Rightarrow \quad r_3 \leftarrow x$$

$r_0 = 1$

Example 3

Thread 0	Thread 1	Thread 2
$x \leftarrow 1$	$r_0 \leftarrow x$	$r_2 \leftarrow y$
$y \leftarrow 1$	$r_1 \leftarrow y$	$r_3 \leftarrow x$

Table: Initially, $x = y = 0$. Can this program trace yield $r_0 = 1 \wedge r_1 = 1 \wedge r_2 = 1 \wedge r_3 = 1$ under TSO? PSO? RMO?

TSO:Yes! PSO:Yes! RMO:Yes!

$$x \leftarrow 1 \Rightarrow y \leftarrow 1 \Rightarrow r_0 \leftarrow x \Rightarrow r_1 \leftarrow y \Rightarrow r_2 \leftarrow y \Rightarrow r_3 \leftarrow x$$

$r_0 = 1$ $r_1 = 1$

Example 3

Thread 0	Thread 1	Thread 2
$x \leftarrow 1$	$r_0 \leftarrow x$	$r_2 \leftarrow y$
$y \leftarrow 1$	$r_1 \leftarrow y$	$r_3 \leftarrow x$

Table: Initially, $x = y = 0$. Can this program trace yield $r_0 = 1 \wedge r_1 = 1 \wedge r_2 = 1 \wedge r_3 = 1$ under TSO? PSO? RMO?

TSO:Yes! PSO:Yes! RMO:Yes!

$$x \leftarrow 1 \Rightarrow y \leftarrow 1 \Rightarrow r_0 \leftarrow x \Rightarrow r_1 \leftarrow y \Rightarrow r_2 \leftarrow y \Rightarrow r_3 \leftarrow x$$

$r_0 = 1$ $r_1 = 1$ $r_2 = 1$

Example 3

Thread 0	Thread 1	Thread 2
$x \leftarrow 1$	$r_0 \leftarrow x$	$r_2 \leftarrow y$
$y \leftarrow 1$	$r_1 \leftarrow y$	$r_3 \leftarrow x$

Table: Initially, $x = y = 0$. Can this program trace yield $r_0 = 1 \wedge r_1 = 1 \wedge r_2 = 1 \wedge r_3 = 1$ under TSO? PSO? RMO?

TSO:Yes! PSO:Yes! RMO:Yes!

$$x \leftarrow 1 \Rightarrow y \leftarrow 1 \Rightarrow r_0 \leftarrow x \Rightarrow r_1 \leftarrow y \Rightarrow r_2 \leftarrow y \Rightarrow r_3 \leftarrow x$$

$r_0 = 1$ $r_1 = 1$ $r_2 = 1$ $r_3 = 1$

Example 3

Thread 0	Thread 1	Thread 2
$x \leftarrow 1$	$r_0 \leftarrow x$	$r_2 \leftarrow y$
$y \leftarrow 1$	$r_1 \leftarrow y$	$r_3 \leftarrow x$

Table: Initially, $x = y = 0$. Can this program trace yield $r_0 = 1 \wedge r_1 = 1 \wedge r_2 = 1 \wedge r_3 = 0$ under TSO? PSO? RMO?

Example 3

Thread 0	Thread 1	Thread 2
$x \leftarrow 1$	$r_0 \leftarrow x$	$r_2 \leftarrow y$
$y \leftarrow 1$	$r_1 \leftarrow y$	$r_3 \leftarrow x$

Table: Initially, $x = y = 0$. Can this program trace yield $r_0 = 1 \wedge r_1 = 1 \wedge r_2 = 1 \wedge r_3 = 0$ under TSO? PSO? RMO?

TSO:No!

Example 3

Thread 0	Thread 1	Thread 2
$x \leftarrow 1$	$r_0 \leftarrow x$	$r_2 \leftarrow y$
$y \leftarrow 1$	$r_1 \leftarrow y$	$r_3 \leftarrow x$

Table: Initially, $x = y = 0$. Can this program trace yield $r_0 = 1 \wedge r_1 = 1 \wedge r_2 = 1 \wedge r_3 = 0$ under TSO? PSO? RMO?

TSO:No!

PSO:Yes!

Example 3

Thread 0	Thread 1	Thread 2
$x \leftarrow 1$	$r_0 \leftarrow x$	$r_2 \leftarrow y$
$y \leftarrow 1$	$r_1 \leftarrow y$	$r_3 \leftarrow x$

Table: Initially, $x = y = 0$. Can this program trace yield $r_0 = 1 \wedge r_1 = 1 \wedge r_2 = 1 \wedge r_3 = 0$ under TSO? PSO? RMO?

TSO:No! PSO:Yes! RMO:Yes!

$y \leftarrow 1 \quad \Rightarrow \quad r_2 \leftarrow y \quad \Rightarrow \quad r_3 \leftarrow x \quad \Rightarrow \quad x \leftarrow 1 \quad \Rightarrow \quad r_0 \leftarrow x \quad \Rightarrow \quad r_1 \leftarrow y$

Example 3

Thread 0	Thread 1	Thread 2
$x \leftarrow 1$	$r_0 \leftarrow x$	$r_2 \leftarrow y$
$y \leftarrow 1$	$r_1 \leftarrow y$	$r_3 \leftarrow x$

Table: Initially, $x = y = 0$. Can this program trace yield $r_0 = 1 \wedge r_1 = 1 \wedge r_2 = 1 \wedge r_3 = 0$ under TSO? PSO? RMO?

TSO:No! PSO:Yes! RMO:Yes!

$y \leftarrow 1 \quad \Rightarrow \quad r_2 \leftarrow y \quad \Rightarrow \quad r_3 \leftarrow x \quad \Rightarrow \quad x \leftarrow 1 \quad \Rightarrow \quad r_0 \leftarrow x \quad \Rightarrow \quad r_1 \leftarrow y$

$r_2 = 1$

Example 3

Thread 0	Thread 1	Thread 2
$x \leftarrow 1$	$r_0 \leftarrow x$	$r_2 \leftarrow y$
$y \leftarrow 1$	$r_1 \leftarrow y$	$r_3 \leftarrow x$

Table: Initially, $x = y = 0$. Can this program trace yield $r_0 = 1 \wedge r_1 = 1 \wedge r_2 = 1 \wedge r_3 = 0$ under TSO? PSO? RMO?

TSO:No! PSO:Yes! RMO:Yes!

$$y \leftarrow 1 \Rightarrow r_2 \leftarrow y \Rightarrow r_3 \leftarrow x \Rightarrow x \leftarrow 1 \Rightarrow r_0 \leftarrow x \Rightarrow r_1 \leftarrow y$$

$r_2 = 1$ $r_3 = 0$

Example 3

Thread 0	Thread 1	Thread 2
$x \leftarrow 1$	$r_0 \leftarrow x$	$r_2 \leftarrow y$
$y \leftarrow 1$	$r_1 \leftarrow y$	$r_3 \leftarrow x$

Table: Initially, $x = y = 0$. Can this program trace yield $r_0 = 1 \wedge r_1 = 1 \wedge r_2 = 1 \wedge r_3 = 0$ under TSO? PSO? RMO?

TSO:No! PSO:Yes! RMO:Yes!

$$y \leftarrow 1 \Rightarrow r_2 \leftarrow y \Rightarrow r_3 \leftarrow x \Rightarrow x \leftarrow 1 \Rightarrow r_0 \leftarrow x \Rightarrow r_1 \leftarrow y$$

$r_2 = 1$ $r_3 = 0$ $r_0 = 1$

Example 3

Thread 0	Thread 1	Thread 2
$x \leftarrow 1$	$r_0 \leftarrow x$	$r_2 \leftarrow y$
$y \leftarrow 1$	$r_1 \leftarrow y$	$r_3 \leftarrow x$

Table: Initially, $x = y = 0$. Can this program trace yield $r_0 = 1 \wedge r_1 = 1 \wedge r_2 = 1 \wedge r_3 = 0$ under TSO? PSO? RMO?

TSO:No! PSO:Yes! RMO:Yes!

$$y \leftarrow 1 \Rightarrow r_2 \leftarrow y \Rightarrow r_3 \leftarrow x \Rightarrow x \leftarrow 1 \Rightarrow r_0 \leftarrow x \Rightarrow r_1 \leftarrow y$$

$r_2 = 1$ $r_3 = 0$ $r_0 = 1$ $r_1 = 1$

Example 3

Thread 0	Thread 1	Thread 2
$x \leftarrow 1$	$r_0 \leftarrow x$	$r_2 \leftarrow y$
$y \leftarrow 1$	$r_1 \leftarrow y$	$r_3 \leftarrow x$

Table: Initially, $x = y = 0$. Can this program trace yield $r_0 = 1 \wedge r_1 = 0 \wedge r_2 = 1 \wedge r_3 = 0$ under TSO? PSO? RMO?

Example 3

Thread 0	Thread 1	Thread 2
$x \leftarrow 1$	$r_0 \leftarrow x$	$r_2 \leftarrow y$
$y \leftarrow 1$	$r_1 \leftarrow y$	$r_3 \leftarrow x$

Table: Initially, $x = y = 0$. Can this program trace yield $r_0 = 1 \wedge r_1 = 0 \wedge r_2 = 1 \wedge r_3 = 0$ under TSO? PSO? RMO?

TSO:No!

Example 3

Thread 0	Thread 1	Thread 2
$x \leftarrow 1$	$r_0 \leftarrow x$	$r_2 \leftarrow y$
$y \leftarrow 1$	$r_1 \leftarrow y$	$r_3 \leftarrow x$

Table: Initially, $x = y = 0$. Can this program trace yield $r_0 = 1 \wedge r_1 = 0 \wedge r_2 = 1 \wedge r_3 = 0$ under TSO? PSO? RMO?

TSO:No!

PSO:No!

Example 3

Thread 0	Thread 1	Thread 2
$x \leftarrow 1$	$r_0 \leftarrow x$	$r_2 \leftarrow y$
$y \leftarrow 1$	$r_1 \leftarrow y$	$r_3 \leftarrow x$

Table: Initially, $x = y = 0$. Can this program trace yield $r_0 = 1 \wedge r_1 = 0 \wedge r_2 = 1 \wedge r_3 = 0$ under TSO? PSO? RMO?

TSO:No! PSO:No! RMO:Yes!

$r_3 \leftarrow x \quad \Rightarrow \quad r_1 \leftarrow y \quad \Rightarrow \quad x \leftarrow 1 \quad \Rightarrow \quad r_0 \leftarrow x \quad \Rightarrow \quad y \leftarrow 1 \quad \Rightarrow \quad r_2 \leftarrow y$

Example 3

Thread 0	Thread 1	Thread 2
$x \leftarrow 1$	$r_0 \leftarrow x$	$r_2 \leftarrow y$
$y \leftarrow 1$	$r_1 \leftarrow y$	$r_3 \leftarrow x$

Table: Initially, $x = y = 0$. Can this program trace yield $r_0 = 1 \wedge r_1 = 0 \wedge r_2 = 1 \wedge r_3 = 0$ under TSO? PSO? RMO?

TSO:No! PSO:No! RMO:Yes!

$r_3 \leftarrow x \quad \Rightarrow \quad r_1 \leftarrow y \quad \Rightarrow \quad x \leftarrow 1 \quad \Rightarrow \quad r_0 \leftarrow x \quad \Rightarrow \quad y \leftarrow 1 \quad \Rightarrow \quad r_2 \leftarrow y$

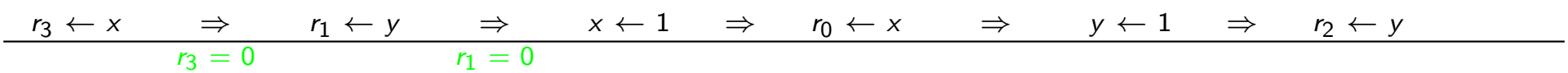
$r_3 = 0$

Example 3

Thread 0	Thread 1	Thread 2
$x \leftarrow 1$	$r_0 \leftarrow x$	$r_2 \leftarrow y$
$y \leftarrow 1$	$r_1 \leftarrow y$	$r_3 \leftarrow x$

Table: Initially, $x = y = 0$. Can this program trace yield $r_0 = 1 \wedge r_1 = 0 \wedge r_2 = 1 \wedge r_3 = 0$ under TSO? PSO? RMO?

TSO:No! PSO:No! RMO:Yes!

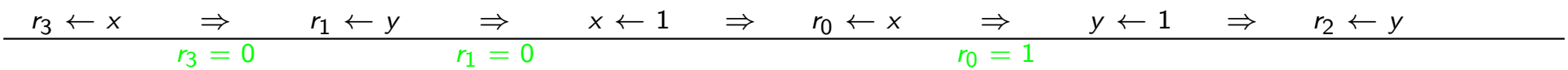


Example 3

Thread 0	Thread 1	Thread 2
$x \leftarrow 1$	$r_0 \leftarrow x$	$r_2 \leftarrow y$
$y \leftarrow 1$	$r_1 \leftarrow y$	$r_3 \leftarrow x$

Table: Initially, $x = y = 0$. Can this program trace yield $r_0 = 1 \wedge r_1 = 0 \wedge r_2 = 1 \wedge r_3 = 0$ under TSO? PSO? RMO?

TSO:No! PSO:No! RMO:Yes!

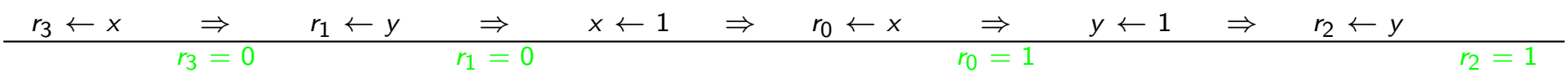


Example 3

Thread 0	Thread 1	Thread 2
$x \leftarrow 1$	$r_0 \leftarrow x$	$r_2 \leftarrow y$
$y \leftarrow 1$	$r_1 \leftarrow y$	$r_3 \leftarrow x$

Table: Initially, $x = y = 0$. Can this program trace yield $r_0 = 1 \wedge r_1 = 0 \wedge r_2 = 1 \wedge r_3 = 0$ under TSO? PSO? RMO?

TSO:No! PSO:No! RMO:Yes!



Outline

The x86 Family of Processors

- Invented by Intel
- Complex Instruction Set Computer (CISC)
  - Now, more like "CRISC:" Instruction set is CISC but...
  - Front-end: CISC; after decoding (rest of pipeline): RISC/VLIW μ-architecture
- ISA cloned and extended by multiple vendors: AMD, Cyrix, VIA, *etc.*
- Most popular processor family for mainstream computing (and very popular for HPC)

A Programmer's Model

- Clarifies allowable behavior
- Does not precisely describes the internal structure of the micro-processor
  - From generation to generation, and from vendor to vendor, the *exact* memory model varies too much

Two Ways to Define the Model

- An abstract machine
- An axiomatic model

x86-TSO Memory Model

Abstract Machine

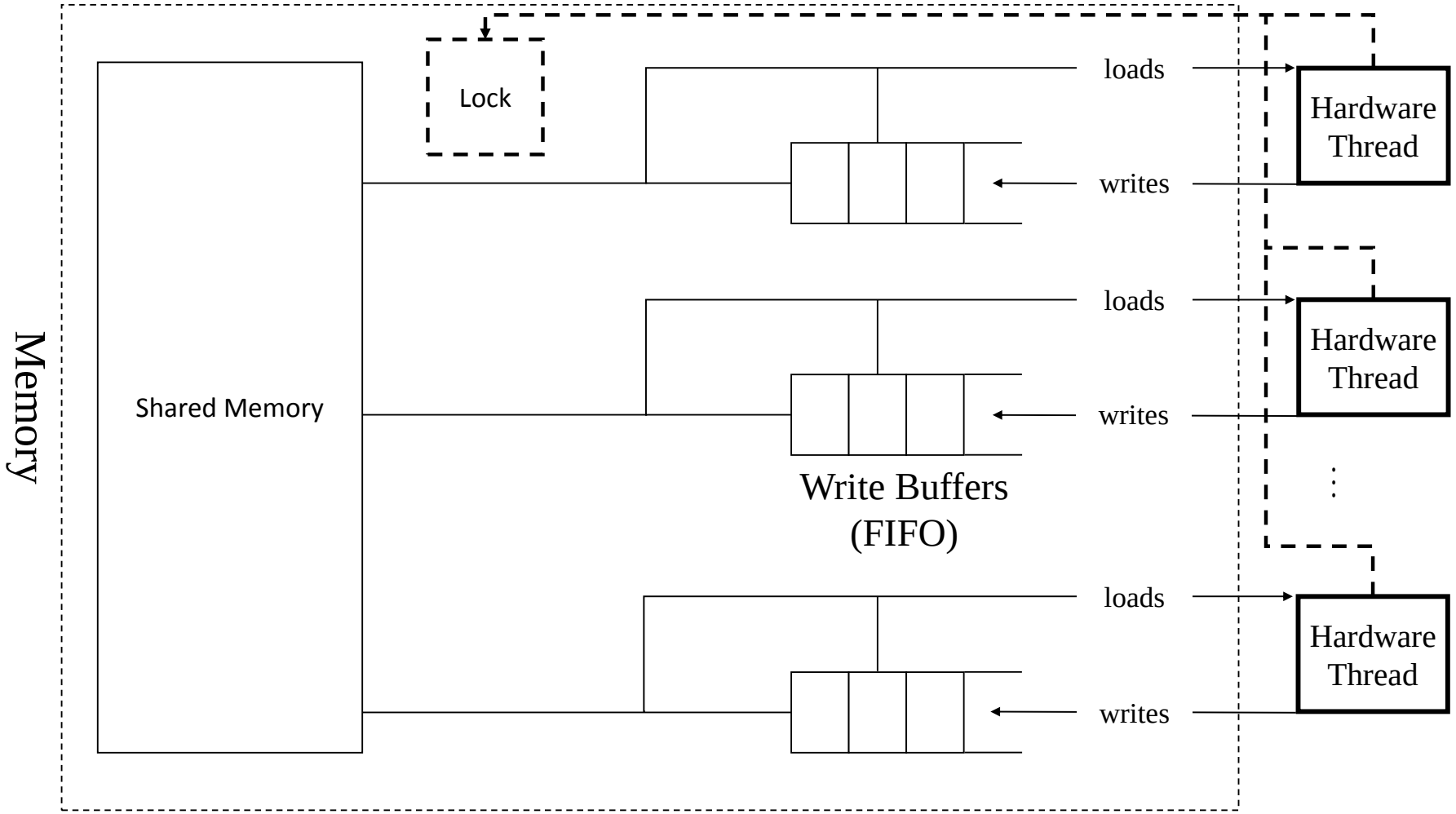


Figure: ¡+caption text+¡

- Store buffers
  - FIFO
  - Reading memory location X returns the most recent buffered write to X
    - If there is no most recent X, read from shared memory
- `MFENCE`
  - Flushes the store buffer of the thread which issued `MFENCE`
- LOCK'd instructions
  - Obtain lock on memory location
  - Flushes its store buffer

Example 4
x86-TSO

Thread 0	Thread 1
$x \leftarrow 1$	$r_0 \leftarrow y$
$y \leftarrow 1$	$r_1 \leftarrow x$

Table: Initially, $x = y = 0$. Can we have $r_0 = 1 \wedge r_1 = 0$ under x86-TSO?

Example 4
x86-TSO

Thread 0	Thread 1
$x \leftarrow 1$	$r_0 \leftarrow y$
$y \leftarrow 1$	$r_1 \leftarrow x$

Table: Initially, $x = y = 0$. Can we have $r_0 = 1 \wedge r_1 = 0$ under x86-TSO?

No!

Stores cannot be reordered with other stores

Example 5
x86-TSO

Thread 0	Thread 1
$r_0 \leftarrow y$	$r_1 \leftarrow x$
$x \leftarrow 1$	$y \leftarrow 1$

Table: Initially, $x = y = 0$. Can we have $r_0 = 1 \wedge r_1 = 1$ under x86-TSO?

Example 5
x86-TSO

Thread 0	Thread 1
$r_0 \leftarrow y$	$r_1 \leftarrow x$
$x \leftarrow 1$	$y \leftarrow 1$

Table: Initially, $x = y = 0$. Can we have $r_0 = 1 \wedge r_1 = 1$ under x86-TSO?

No!

Stores cannot be reordered with older loads

Example 6
x86-TSO

Thread 0	Thread 1
$x \leftarrow 1$	$y \leftarrow 1$
$r_0 \leftarrow y$	$r_1 \leftarrow x$

Table: Initially, $x = y = 0$. Can we have $r_0 = 0 \wedge r_1 = 0$ under x86-TSO?

Example 6
x86-TSO

Thread 0	Thread 1
$x \leftarrow 1$	$y \leftarrow 1$
$r_0 \leftarrow y$	$r_1 \leftarrow x$

Table: Initially, $x = y = 0$. Can we have $r_0 = 0 \wedge r_1 = 0$ under x86-TSO?

Yes!

Loads may be reordered with older stores

$r_0 \leftarrow y \Rightarrow r_1 \leftarrow x \Rightarrow x \leftarrow 1 \Rightarrow y \leftarrow 1$

Example 7
x86-TSO

Thread 0
$x \leftarrow 1$
$r_0 \leftarrow x$

Table: Initially, $x = y = 0$. Can we have $r_0 = 0$ under x86-TSO?

Example 7
x86-TSO

Thread 0
$x \leftarrow 1$
$r_0 \leftarrow x$

Table: Initially, $x = y = 0$. Can we have $r_0 = 0$ under x86-TSO?

No!

Loads cannot be reordered with older stores to the same location

Example 8
x86-TSO

Thread 0	Thread 1	Thread 2
$x \leftarrow 1$	$r_0 \leftarrow x$	$r_1 \leftarrow y$
	$y \leftarrow 1$	$r_2 \leftarrow x$

Table: Initially, $x = y = 0$. Can we have $r_0 = 1 \wedge r_1 = 1 \wedge r_2 = 0$ under x86-TSO?

Example 8
x86-TSO

Thread 0	Thread 1	Thread 2
$x \leftarrow 1$	$r_0 \leftarrow x$	$r_1 \leftarrow y$
	$y \leftarrow 1$	$r_2 \leftarrow x$

Table: Initially, $x = y = 0$. Can we have $r_0 = 1 \wedge r_1 = 1 \wedge r_2 = 0$ under x86-TSO?

No!

Stores are transitively visible.

Example 9
x86-TSO

Thread 1	**Thread 2**
$\texttt{XCHG}(x) \leftarrow r_0$	$\texttt{XCHG}(y) \leftarrow r_2$
$r_1 \leftarrow x$	$r_3 \leftarrow y$

Table: Initially, $x = y = 0$; $r_1 = r_2 = 1$. Can we have $r_1 = 0 \wedge r_3 = 0$ under x86-TSO?

Example 9
x86-TSO

Thread 1	**Thread 2**
$\texttt{XCHG}(x) \leftarrow r_0$	$\texttt{XCHG}(y) \leftarrow r_2$
$r_1 \leftarrow x$	$r_3 \leftarrow y$

Table: Initially, $x = y = 0$; $r_1 = r_2 = 1$. Can we have $r_1 = 0 \wedge r_3 = 0$ under x86-TSO?

No!

Loads cannot be reordered with locks.

Outline

The Future of Parallel Computing Systems I

- Future many-core systems will probably feature network-on-chip (NoC) technologies
- NoC can provide static or dynamic routes
  - Static routes require very small headers and low-overhead if a given NoC router is not contended
  - Dynamic routes allow a NoC to adapt to congested routers at runtime, thus handling more gracefully heavy loads.
- Future many-core chips: hundreds or even thousands of cores on a single general-purpose chip
  - GPUs are not quite general purpose, but already propose thousands of threads (streams) on a board
  - Intel's Xeon Phi features 61 cores (244 hyperthreads) which leverage modified a Pentium (P54C) micro-architecture
- Other available chips:
  - Tilera's Tile processors (32, 36, 64 cores, and soon 100 cores on a chip),

The Future of Parallel Computing Systems II

- Kalaray's MPPA-256: "heavy" cores run regular operating systems and handle I/Os; "lightweight" cores perform the computation, using a dataflow approach to know where to write the results in small on-chip memories.
- Adapteva's Epiphany III (16 cores) and IV (64 cores) for low-power embedded computing: each core has a 32KB scratchpad, accessible from other cores.

- What will the memory model of future many-core chips look like? How can we ensure it is scalable?

A Few Open Questions About Memory Models

Q0 The "regular" consistency question: what happens when two memory operations reach the same memory location, and at least one is a write?

- In other words, what rules govern data races?

Q1 Should the hardware allow more than one path from a given processor to reach a given memory location?

- In other words:

Q2 If we assume it is possible that Q1's answer is yes, if a processor P issues two memory operations to the same memory location X, can these operations be performed out-of-order?

Q3 Can we get rid of the coherence assumption?

- Most "hardware-oriented" memory consistency models assume coherence (*i.e.*, all accesses to a given memory location X are serialized)

Q4 Should a memory consistency model preserve the notion of causality?

A Few Open Questions About Memory Models

Q0 The "regular" consistency question: what happens when two memory operations reach the same memory location, and at least one is a write?

- In other words, what rules govern data races?

Q1 Should the hardware allow more than one path from a given processor to reach a given memory location?

- In other words:

Q2 If we assume it is possible that Q1's answer is yes, if a processor P issues two memory operations to the same memory location X, can these operations be performed out-of-order?

Q3 Can we get rid of the coherence assumption?

- Most "hardware-oriented" memory consistency models assume coherence (*i.e.*, all accesses to a given memory location X are serialized)

Q4 Should a memory consistency model preserve the notion of causality?

- Honestly, if we throw causality out of the window, I don't really see how we can get correct programs...

Location Consistency (LC): Philosophy
(GaoSar95; GaoSar00)

- If you wish to share data and make sure their value are consistent across threads/processors, you should explicitly say so
- Every other value that was produced by at least one processor/thread is fair game in all the other cases.

Location Consistency (LC): Principles I
(GaoSar95; GaoSar00)

Location Consistency (LC) is a rather different way to look at memory compared to other memory models.

Definition: Partially Ordered Multi-Sets (POMSETs)

- { 1, 5, 3 } is a *set*
- { 1, 5, 3, 5 } is a *multi-set* (values can be repeated)
- { {1, 5} {3, 5}, {1} } is a *partially ordered multi-set*

Location Consistency (LC): Principles II
(GaoSar95; GaoSar00)

UNIVERSITY OF DELAWARE

LC: an Informal Description

We consider a memory location X for the remainder of these explanations.

- Each time X is written using an ordinary operation, the new value is added to its associated POMSET
- If a X is consistently accessed using acquire-release pairs, its POMSET always remains a singleton (*i.e.*, a single value)
- What happens if a programmer mixes up ordinary and synchronized accesses to X?
  - The processor using the acquire-release pair will reduce its own set of values w.r.t. X, but
  - The other writes to X from other processors are still within its POMSET: when reading the next value from X, the "dirty" values may be returned at any time instead of the "clean" one.
- LC proposes various synchronizing operations which help processors to reduce the POMSET of X to a singleton.
  - Most important ones: acquire, release, sync.

With No Synchronization

Thread 0 [T0]	Thread 1 [T1]	Thread 2 [T2]
$x \leftarrow 0$		
$y \leftarrow 0$	$\cdots \leftarrow y$	
	$x \leftarrow 1$	$\cdots \leftarrow x$
		$\cdots \leftarrow x$

Table: Initially, $x = y = 0$. What are the possible values in LC?

UNIVERSITY OF DELAWARE

With No Synchronization

Thread 0 [T0]	Thread 1 [T1]	Thread 2 [T2]
$x \leftarrow 0$		
$y \leftarrow 0$	$\cdots \leftarrow y$	
	$x \leftarrow 1$	$\cdots \leftarrow x$
		$\cdots \leftarrow x$

Table: Initially, $x = y = 0$. What are the possible values in LC?

T0 $x = \{\{0\}, \{1\}\}$; $y = \{\{0\}, \{0\}\}$

T1 $x = \{\{0\}, \{1\}\}$; $y = \{\{0\}, \{0\}\}$

T2 $x = \{\{0\}, \{1\}\}$; $y = \{\{0\}, \{0\}\}$

First Case

Thread 0 [T0]	Thread 1 [T1]	Thread 2 [T2]
$x \leftarrow 0$		
acq(y); $y \leftarrow 0$ rel(y);	acq(y); $\cdots \leftarrow y$ rel(y);	
	$x \leftarrow 1$	$\cdots \leftarrow x$
		$\cdots \leftarrow x$

Table: Initially, $x = y = 0$. What are the possible values in LC?

First Case

Thread 0 [T0]	Thread 1 [T1]	Thread 2 [T2]
$x \leftarrow 0$		
acq(y); $y \leftarrow 0$ rel(y);	acq(y); $\cdots \leftarrow y$ rel(y);	
	$x \leftarrow 1$	$\cdots \leftarrow x$
		$\cdots \leftarrow x$

Table: Initially, $x = y = 0$. What are the possible values in LC?

T0 $x = \{\{0\}, \{1\}\}$; $y = \{\{0\}\}$

T1 $x = \{\{0\}, \{1\}\}$; $y = \{\{0\}\}$

T2 $x = \{\{0\}, \{1\}\}$; $y = \{\{0\}\}$

Second Case

Thread 0 [T0]	Thread 1 [T1]	Thread 2 [T2]
acq(x); $x \leftarrow 0$ rel(x);		
acq(y); $y \leftarrow 0$ rel(y);	acq(y); $\cdots \leftarrow y$ rel(y);	
	acq(x); $x \leftarrow 1$ rel(x);	acq(x); $\cdots \leftarrow x$ rel(x);
		acq(x); $\cdots \leftarrow x$ rel(x);

Table: Initially, $x = y = 0$. What are the possible values in LC?

Second Case

Thread 0 [T0]	Thread 1 [T1]	Thread 2 [T2]
acq(x); $x \leftarrow 0$ rel(x);		
acq(y); $y \leftarrow 0$ rel(y);	acq(y); $\cdots \leftarrow y$ rel(y);	
	acq(x); $x \leftarrow 1$ rel(x);	acq(x); $\cdots \leftarrow x$ rel(x);
		acq(x); $\cdots \leftarrow x$ rel(x);

Table: Initially, $x = y = 0$. What are the possible values in LC?

T0 $x = \{\{0\}$ or $\{\{1\}\}$; $y = \{\{0\}\}$

T1 $x = \{\{0\}$ or $\{\{1\}\}$; $y = \{\{0\}\}$

T2 $x = \{\{0\}$ or $\{\{1\}\}$; $y = \{\{0\}\}$

Second Case – Variant

Thread 0 [T0]	Thread 1 [T1]	Thread 2 [T2]
acq(x,y); $x \leftarrow 0$		
$y \leftarrow 0$ rel(x,y);	acq(x,y); $\cdots \leftarrow y$	
	$x \leftarrow 1$ rel(x,y);	acq(x); $\cdots \leftarrow x$ rel(x);
		acq(x); $\cdots \leftarrow x$ rel(x);

Table: Initially, $x = y = 0$. What are the possible values in LC?

Second Case – Variant

Thread 0 [T0]	Thread 1 [T1]	Thread 2 [T2]
acq(x,y); $x \leftarrow 0$		
$y \leftarrow 0$ rel(x,y);	acq(x,y); $\cdots \leftarrow y$	
	$x \leftarrow 1$ rel(x,y);	acq(x); $\cdots \leftarrow x$ rel(x);
		acq(x); $\cdots \leftarrow x$ rel(x);

Table: Initially, $x = y = 0$. What are the possible values in LC?

T0 $x = \{\{0\}$ or $\{\{1\}\}$; $y = \{\{0\}\}$

T1 $x = \{\{0\}$ or $\{\{1\}\}$; $y = \{\{0\}\}$

T2 $x = \{\{0\}$ or $\{\{1\}\}$; $y = \{\{0\}\}$

Second Case – Variant

Thread 0 [T0]	Thread 1 [T1]	Thread 2 [T2]
$x \leftarrow 0$		
$y \leftarrow 0$	$\cdots \leftarrow y$	
$sync_{T0,T1}(x, y)$	$sync_{T0,T1}(x, y)$	
	$x \leftarrow 1$	$\cdots \leftarrow x$
		$\cdots \leftarrow x$

Table: Initially, $x = y = 0$. What are the possible values in LC?

Second Case – Variant

Thread 0 [T0]	Thread 1 [T1]	Thread 2 [T2]
$x \leftarrow 0$		
$y \leftarrow 0$	$\cdots \leftarrow y$	
$sync_{T0,T1}(x, y)$	$sync_{T0,T1}(x, y)$	
	$x \leftarrow 1$	$\cdots \leftarrow x$
		$\cdots \leftarrow x$

Table: Initially, $x = y = 0$. What are the possible values in LC?

T0 $x = \{\{0\}\}; y = \{\{0\}\}$

T1 $x = \{\{1\}\}; y = \{\{0\}\}$

T2 $x = \{\{0\}, \{1\}\}; y = \{\{0\}\}$

Example 10
SPARC (TSO, PSO, RMO)

Thread 1	Thread 2
$x \leftarrow 1$	$y \leftarrow 1$
$r_0 \leftarrow y$	$r_1 \leftarrow x$

Table: Initially, $x = y = 0$. Can we have $r_0 = 0 \wedge r_1 = 0$ under TSO? PSO? RMO?

Example 10
SPARC (TSO, PSO, RMO)

Thread 1	**Thread 2**
$x \leftarrow 1$	$y \leftarrow 1$
$r_0 \leftarrow y$	$r_1 \leftarrow x$

Table: Initially, $x = y = 0$. Can we have $r_0 = 0 \wedge r_1 = 0$ under TSO? PSO? RMO?

TSO: Yes!

Example 10
SPARC (TSO, PSO, RMO)

Thread 1	**Thread 2**
$x \leftarrow 1$	$y \leftarrow 1$
$r_0 \leftarrow y$	$r_1 \leftarrow x$

Table: Initially, $x = y = 0$. Can we have $r_0 = 0 \wedge r_1 = 0$ under TSO? PSO? RMO?

TSO: Yes!

PSO: Yes!

Example 10
SPARC (TSO, PSO, RMO)

Thread 1	Thread 2
$x \leftarrow 1$	$y \leftarrow 1$
$r_0 \leftarrow y$	$r_1 \leftarrow x$

Table: Initially, $x = y = 0$. Can we have $r_0 = 0 \wedge r_1 = 0$ under TSO? PSO? RMO?

TSO: Yes! PSO: Yes! RMO: Yes!

Loads (reads) can be reordered with the stores

Example 10
SPARC (TSO, PSO, RMO)

Thread 1	**Thread 2**
$x \leftarrow 1$	$y \leftarrow 1$
MEMBAR #StoreLoad	MEMBAR #StoreLoad
$r_0 \leftarrow y$	$r_1 \leftarrow x$

Table: Initially, $x = y = 0$. Can we have $r_0 = 0 \wedge r_1 = 0$ under TSO? PSO? RMO?

TSO: Yes! PSO: Yes! RMO: Yes!

Loads (reads) can be reordered with the stores . . . Unless we insert a MEMBAR #StoreLoad operation.

Example 11
x86-TSO

Thread 1	Thread 2
$x \leftarrow 1$	$y \leftarrow 1$
$r_0 \leftarrow y$	$r_1 \leftarrow x$

Table: Initially, $x = y = 0$. Can we have $r_0 = 0 \wedge r_1 = 0$ under x86-TSO?

Example 11
x86-TSO

Thread 1	Thread 2
$x \leftarrow 1$	$y \leftarrow 1$
$r_0 \leftarrow y$	$r_1 \leftarrow x$

Table: Initially, $x = y = 0$. Can we have $r_0 = 0 \wedge r_1 = 0$ under x86-TSO?

TSO: Yes!

Loads (reads) can be reordered with the stores

Example 11
x86-TSO

Thread 1	Thread 2
$x \leftarrow 1$	$y \leftarrow 1$
MFENCE	MFENCE
$r_0 \leftarrow y$	$r_1 \leftarrow x$

Table: Initially, $x = y = 0$. Can we have $r_0 = 0 \wedge r_1 = 0$ under x86-TSO?

TSO: Yes!

Loads (reads) can be reordered with the stores . . . Unless we insert a MFENCE operation.

References I

- AdveGha96
- BoehmAdv08
- ChenEtAl10
- GaoSar95
- GaoSar97
- GaoSar00
- HuttoAhaA90
- Mosberger93
- MansonPugAdv05
- SewellEtAl10
- MANSparc92
- MANSparc94