A System-Level Specification Framework for I/O Architectures*

Mark D. Hill, Anne E. Condon, Manoj Plakal, Daniel J. Sorin

Computer Sciences Department
University of Wisconsin-Madison
{markhill,condon,plakal,sorin}@cs.wisc.edu

*. This work is supported in part by the National Science Foundation with grants MIP-9225097, MIPS-9625558, CCR 9257241, and CDA-9623632, a Wisconsin Romnes Fellowship, and donations from Sun Microsystems and Intel Corporation.

Motivation & Problem

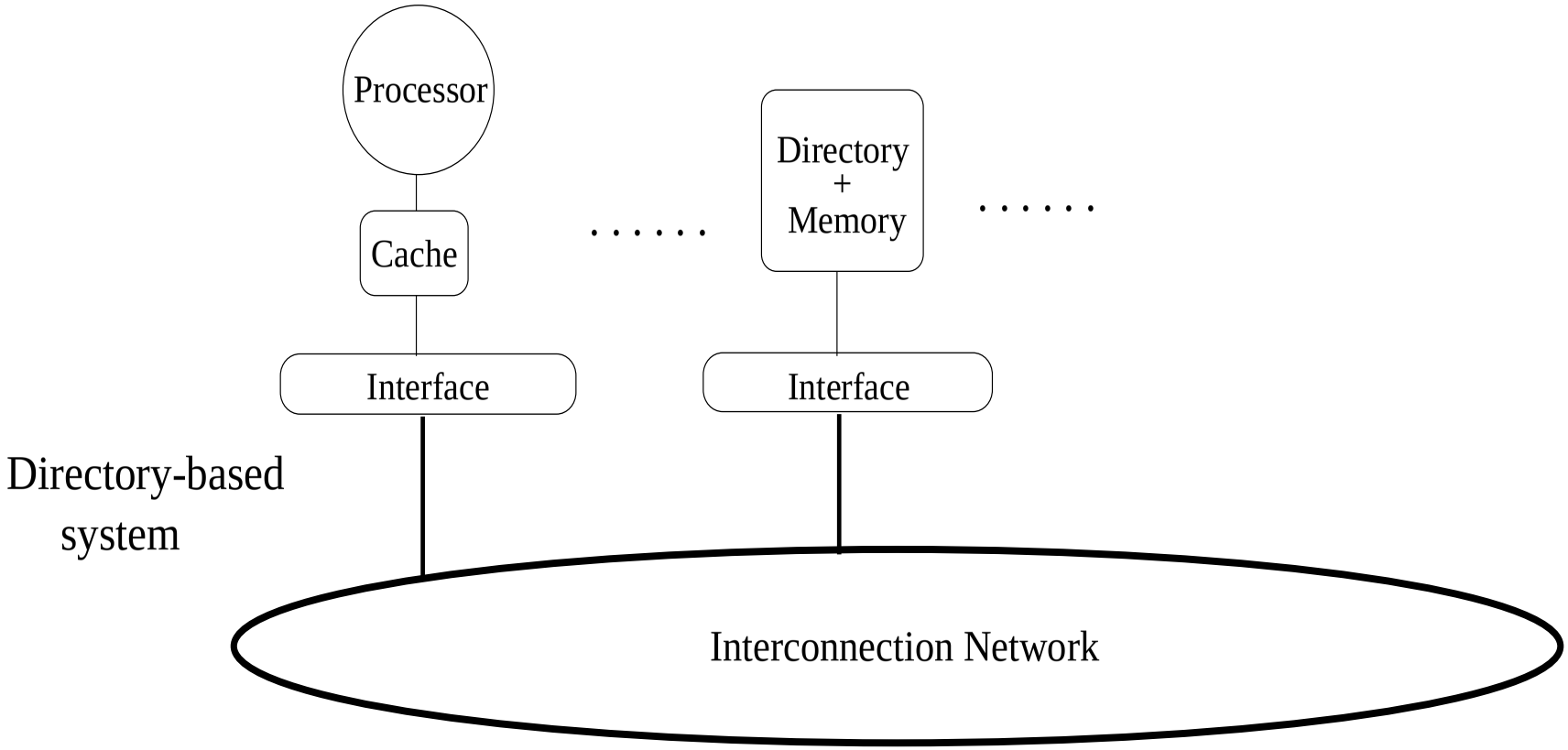
- Memory systems for shared memory multiprocessors
  - Caches, interconnect, directories, etc.
  - Memory consistency model allows end-to-end reasoning
- But what about Input/Output (I/O) Systems?
  - Memory system, I/O bus, disks, network interface, etc.
  - No model for end-to-end reasoning
- We propose Wisconsin I/O Ordering (WIO)
  - Extends end-to-end reasoning of memory consistency models
  - Parameterizable framework for system-wide ordering
  - Contract between system implementer and programmer

Outline

- Memory Consistency Models
- I/O Systems
- Wisconsin I/O Ordering for Sequentially Consistent Memory
- WIO for Other Underlying Memory Models
- Conclusions

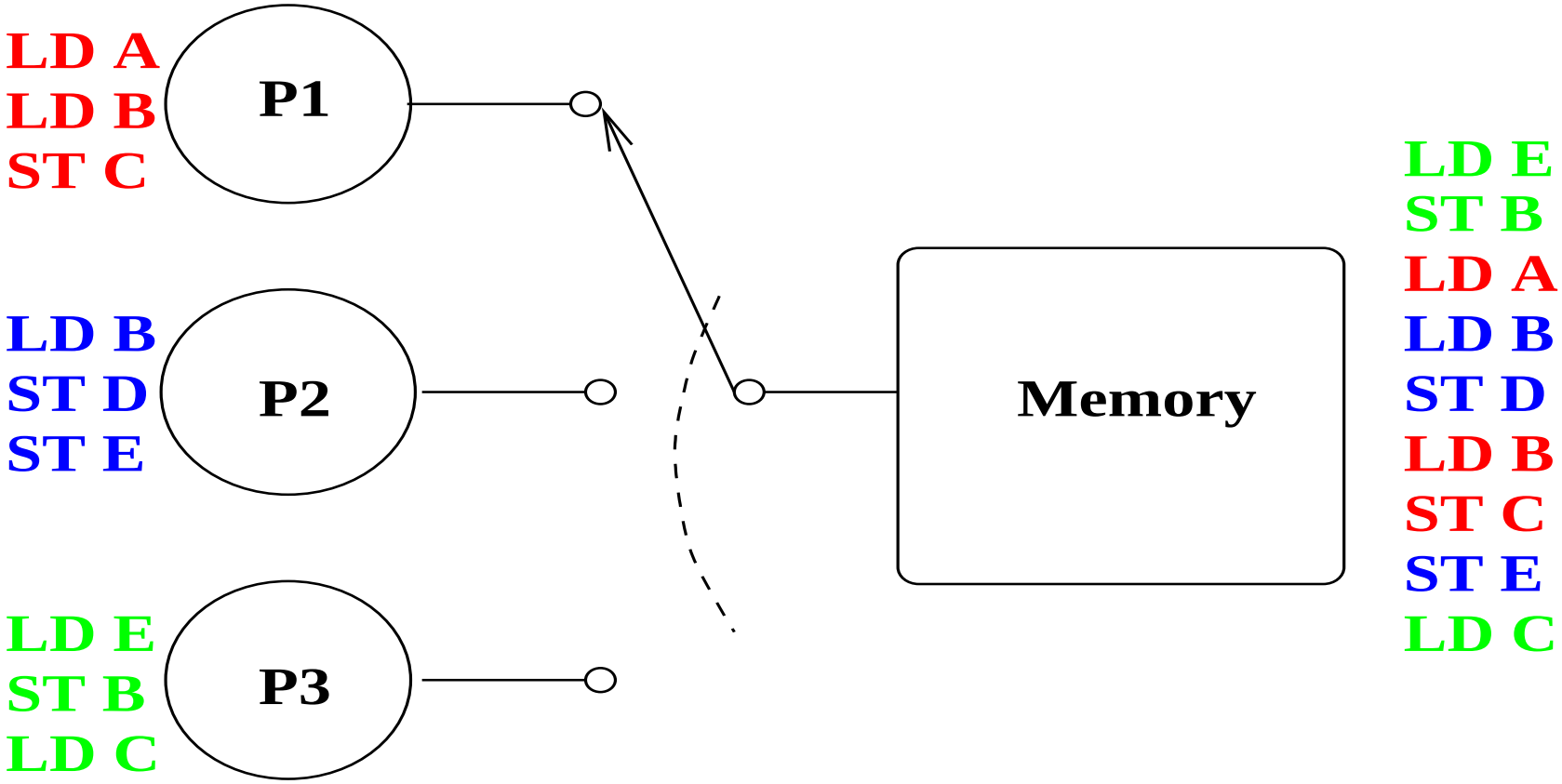
Systems without I/O

- Memory consistency models specify behavior of loads/stores



How We'd Like to Reason About Systems

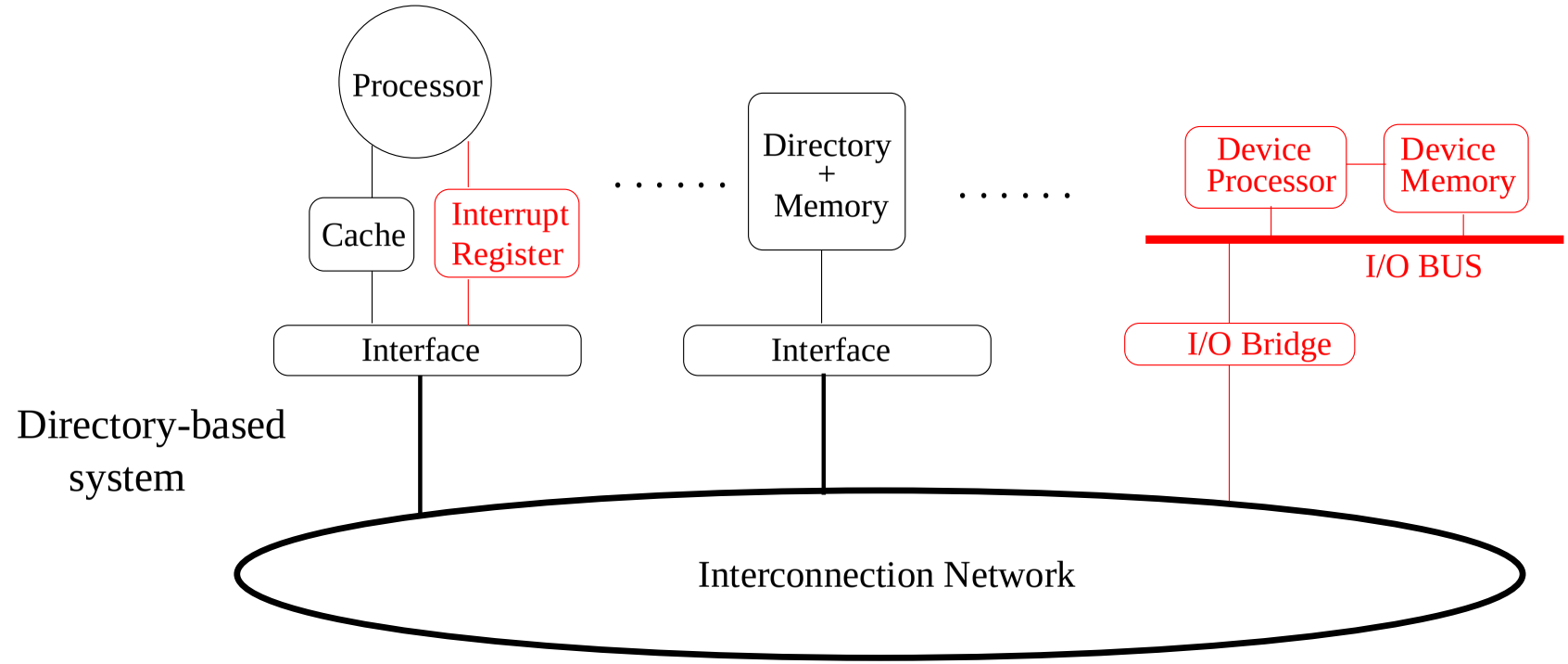
- We'd like a simple, intuitive consistency model like sequential consistency (SC)



Memory Consistency Model

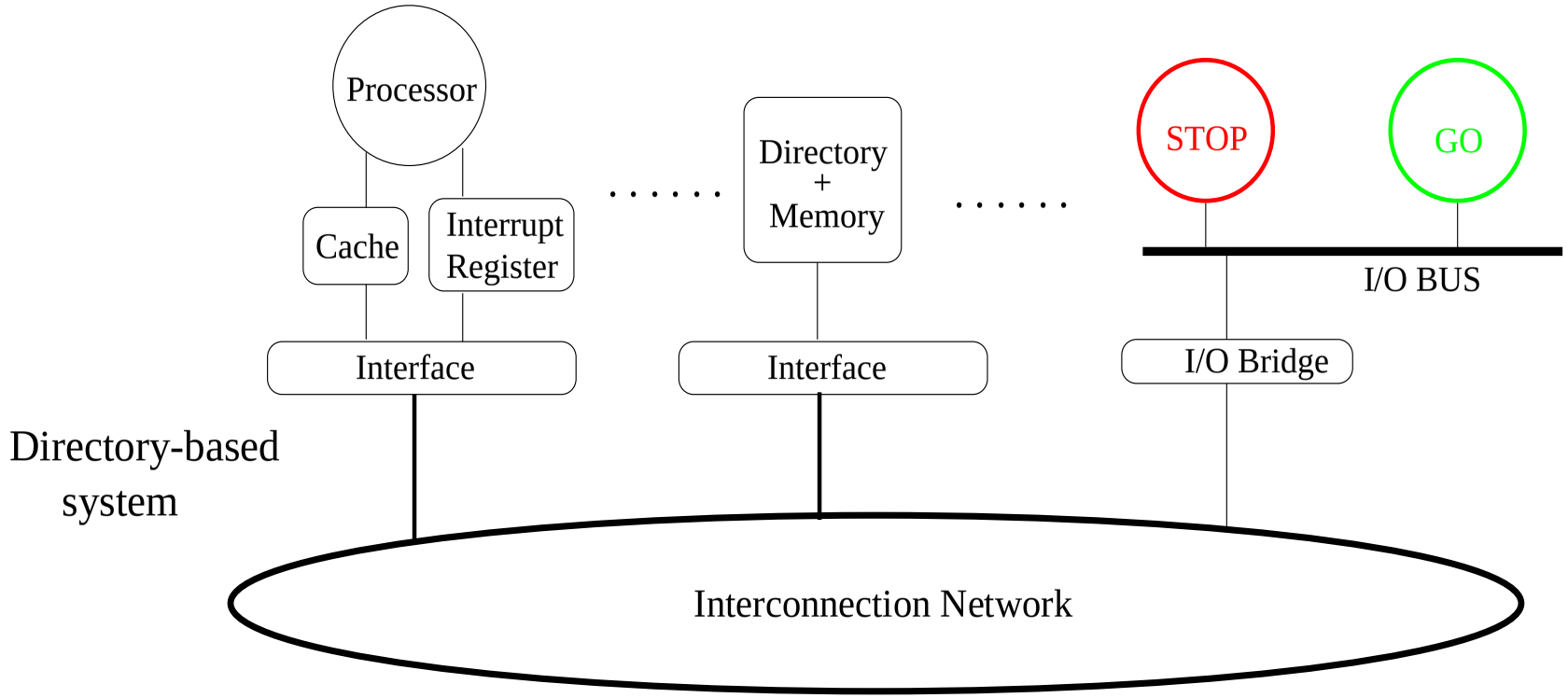
- Provides interface between the architecture and the programmer
- Specifies ordering between loads and stores to regular (cacheable) memory
- Example: Sequential Consistency (SC)
- For each execution
  - There exists a total order
  - That respects the program order of each processor
  - Loads return the value of the last store in that order

Components of I/O Systems



- I/O Bridge and I/O Bus
- Devices: modeled as memory/processor pairs

The Problem

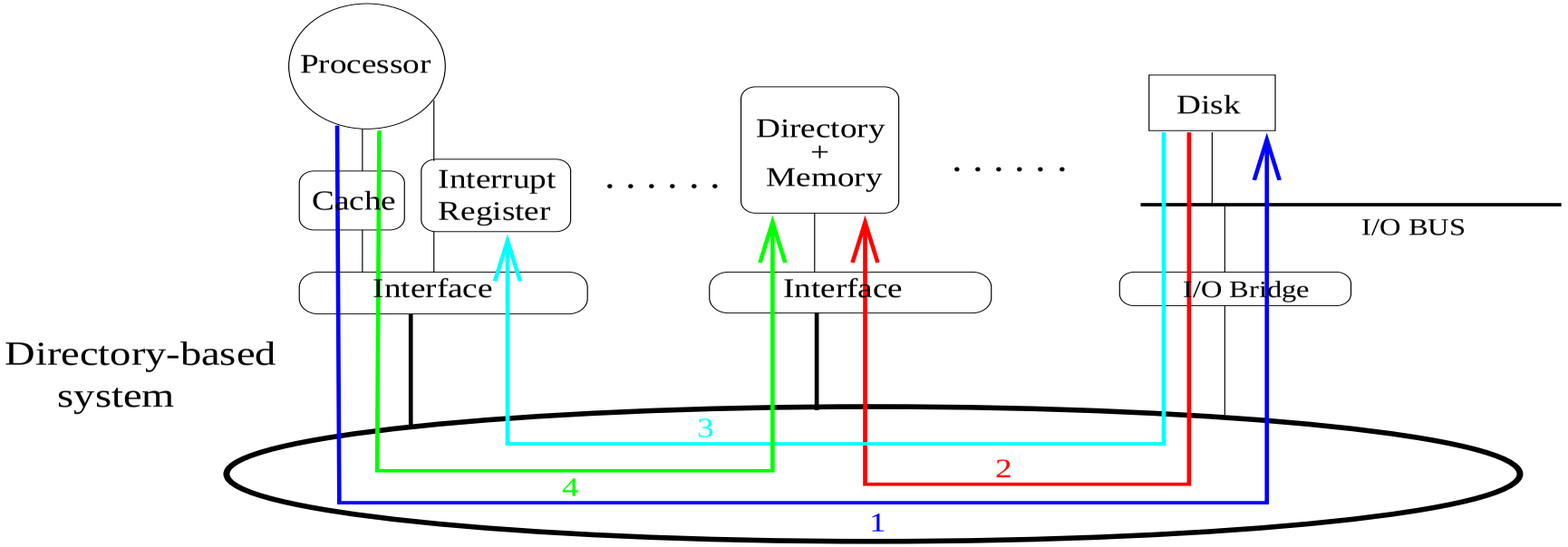


- turn green light on
- turn green light off
- turn red light on

} Need to preserve order here

The Problem

- Example: reading from a disk:



- Must keep these operations in order or else processor could get stale data

- Goal:
  - Create framework for specifying ordering among *all* operations

System Operations

operation	description
Load I/O	Read from device register mapped into memory space
Store I/O	Write to device register mapped into memory space
Load Block	Device reads a block from regular memory
Store Block	Device writes a block to regular memory
Interrupt	Device interrupts processor
Load	Read from regular memory
Store	Write to regular memory

Outline

- ~~Memory Consistency Models~~
- ~~I/O Systems~~
- Wisconsin I/O Ordering for Sequentially Consistent Memory
  - Local Ordering
  - System-wide Ordering
- Other Underlying Memory Models
- Conclusions

Wisconsin I/O Ordering

- Provides end-to-end reasoning about system-wide ordering

Feature	**Sequential Consistency**	**Wisconsin I/O**
global order of operations	respects local orders	respects local orders
local order	program order at processors	relaxed program order at processors and devices
operations	loads and stores	all system operations
value of read	value of most recent write to same address	value of most recent write to same address

A Processor's Local Ordering (without I/O)

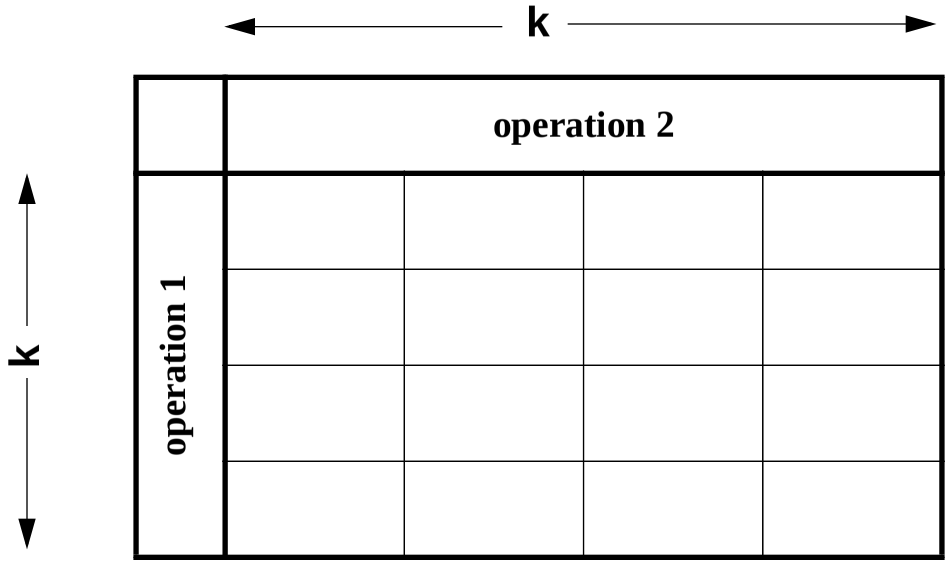
- Sequentially Consistent (SC) memory model
- Local order at a processor observes program order

		Operation 2	
		LD	ST
Operation 1	LD	A	A
	ST	A	A

A = always ordered

An Ordering Framework

- A processor or device can issue k types of operations.
- For each processor and each device, specify a k X k table of local ordering rules, called the *partial program order.*



- Partial program order is a relaxation of program order.

A Processor's Local Ordering with I/O

- Partial Program Order at a Processor

		Operation 2			
		LD	ST	LDio	STio
Operation 1	LD	A	A	A	A
	ST	A	A	A	A
	LDio	A	A	D	D
	STio	—	—	D	D

A = always ordered

D = ordered if both operations are to the same device

— = only ordered if both operations are to the same address

A Processor's Local Ordering with I/O

- Partial Program Order at a Processor

		Operation 2			
		LD	ST	LDio	STio
Operation 1	LD	A	A	A	A
	ST	A	A	A	A
	LDio	A	A	D	D
	STio	—	—	D	D

- Quiz: Create order from a STio(green) to a STio(red).

STio(green) $\rightarrow$ $\rightarrow$ STio(red)

LDio (green) $\rightarrow$ LD

A Device's Local Ordering with I/O

- Partial program order at a device processor

		Operation 2				
		LDio	STio	INT	LDblk	STblk
Operation 1	LDio	A	A	A	A	A
	STio	A	A	A	A	A
	INT	—	—	D	—	—
	LDblk	—	—	A	—	—
	STblk	—	—	A	—	—

A = always ordered

D = ordered if both operations are to the same device

— = only ordered if both operations are to the same address

Wisconsin I/O Ordering (WIO)

- Specifies system-wide ordering rules based on partial program orders at processors and devices
- WIO is a framework, not a specific set of rules
- A total order of all operations in a system obeys WIO if:

  (1) The total order respects all partial program orders, and

  (2) The value of every Read Operation is the same as the value of the most recent Write Operation to the same address in this order.

- WIO is parameterized by an n-tuple of partial program ordering tables.

Other Memory Models

- So far, we've just looked at systems with SC memory.
- How does this apply to more relaxed memory models?
  - Example: Compaq Alpha

		Operation 2				
		LD	ST	MB	LDio	STio
Operation 1	LD	—	—	A	A	A
	ST	—	—	A	A	A
	MB	A	A	A	A	A
	LDio	A	A	A	D	D
	STio	—	—	A	D	D

A = always ordered

D = ordered if both operations are to the same device

— = only ordered if both operations are to the same address

What Isn't in This Talk

- Tables for other memory models (TSO, IA-32)
- Proof that sample system obeys WIO
  - Uses technique of Lamport Clocks [SPAA '98] to create total order of operations
  - Shows that the system can only generate orders that obey WIO

Conclusions

- WIO allows for end-to-end reasoning about systems with I/O

Feature	Sequential Consistency	Wisconsin I/O
global order of operations	respects local orders	respects local orders
local order	program order at processors	relaxed program order at processors and devices
operations	loads and stores	all system operations
value of read	value of most recent write to same address	value of most recent write to same address

- ftp://ftp.cs.wisc.edu/wwt/tr1398_io.ps
- http://www.cs.wisc.edu/multifacet/

Intel IA-64 Consistency

- Cacheable memory only - similar to RCpc

		Operation 2				
		LD	ST	MB	Acquire	Release
Operation 1	LD	—	—	A	A	A
	ST	—	—	A	A	A
	MB	A	A	A	A	A
	Acquire	A	A	A	A	A
	Release	—	—	A	—	A

A = always ordered

— = only ordered if both operations are to the same address