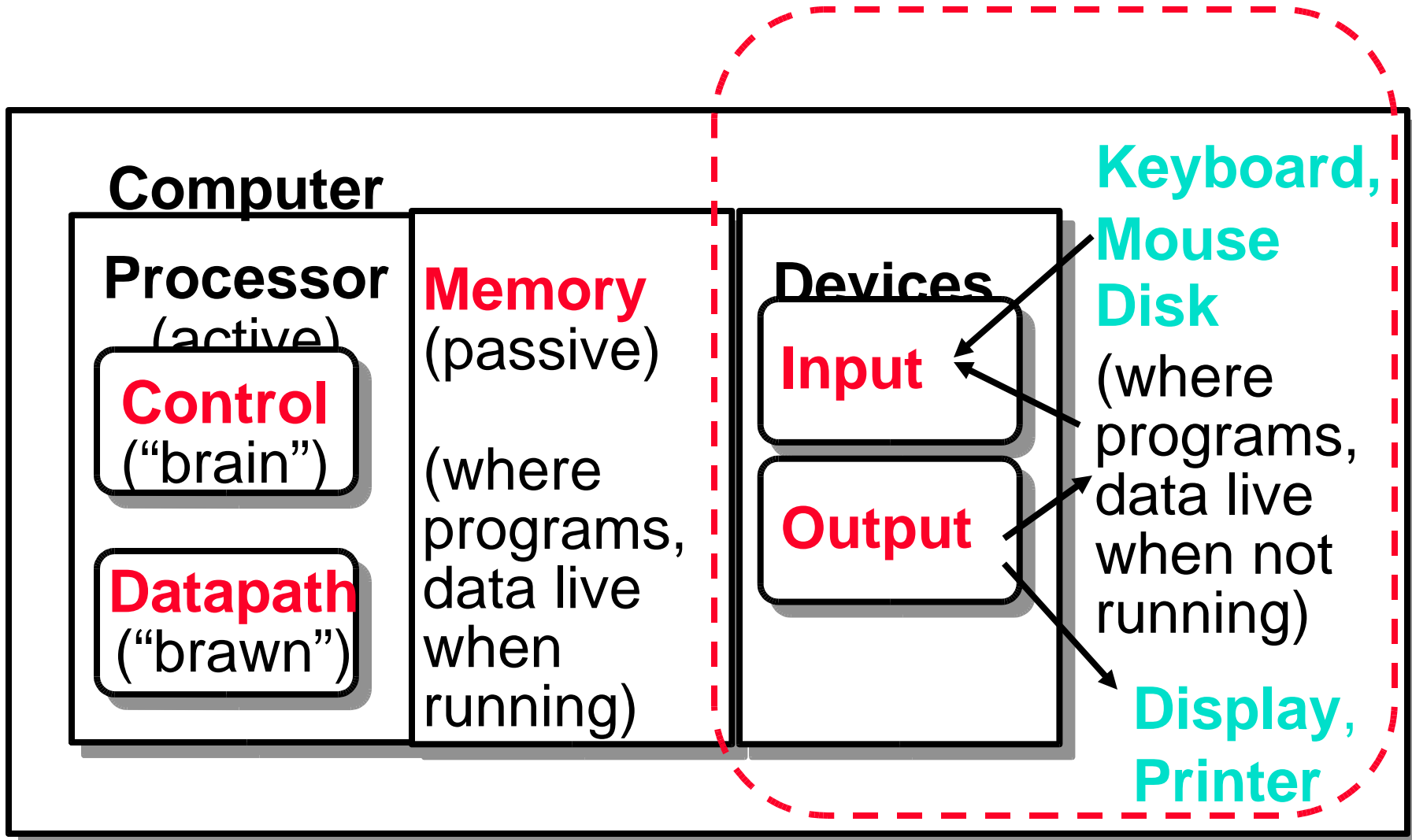

Input/Output: Polling and Interrupts

**Based on slides by
David Patterson**

Outline

- **I/O Background**
- **Polling**
- **Interrupts**

Anatomy: 5 components of any Computer



Motivation for Input/Output

◦ I/O is how humans interact with computers

◦ I/O lets computers do amazing things:

- Read pressure of synthetic hand and control synthetic arm and hand of fireman
- Control propellers, fins, communicate in BOB (Breathable Observable Bubble)
- Read bar codes of items in refrigerator



◦ Computer without I/O like a car without wheels; great technology, but won't get you anywhere

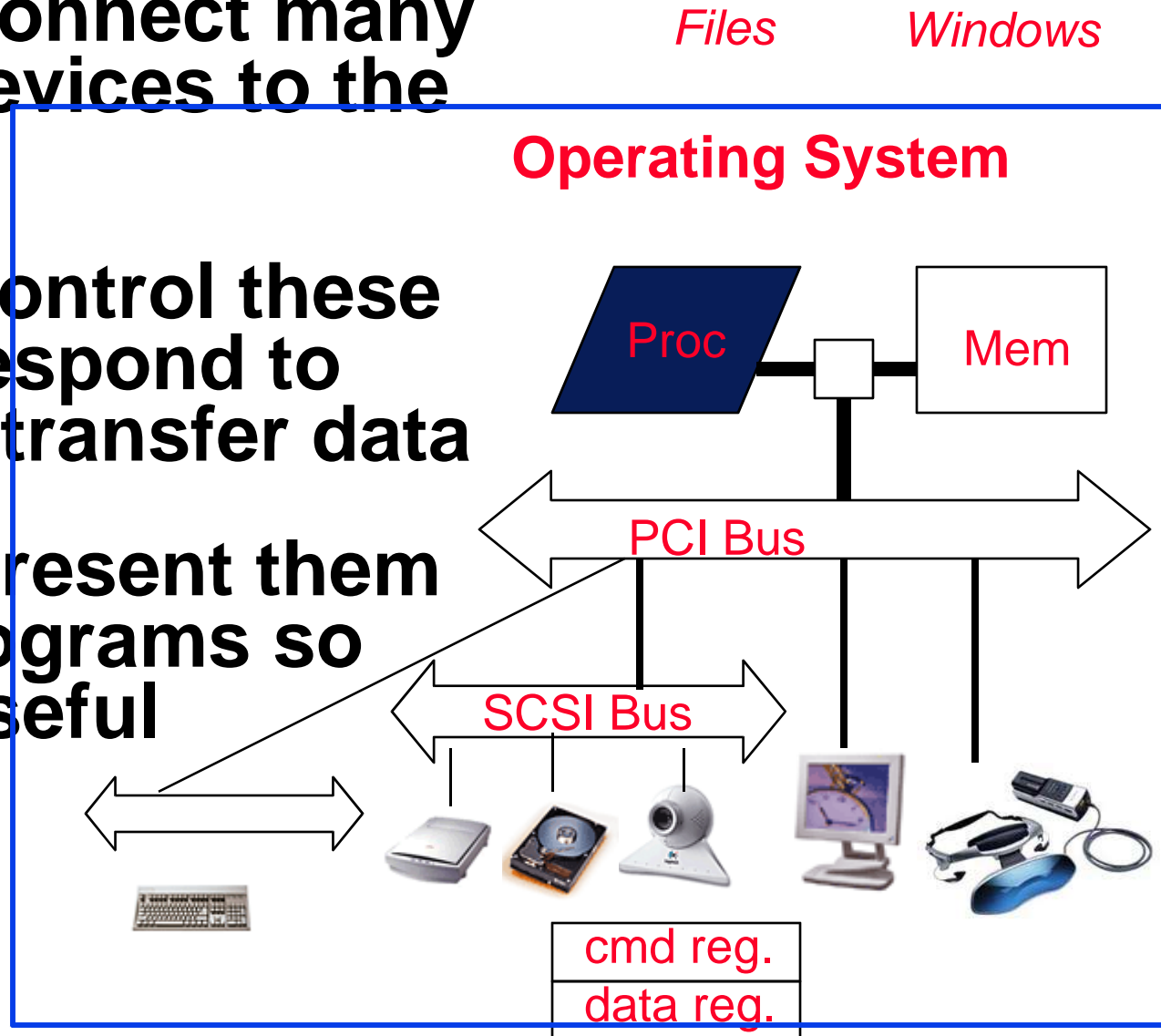
I/O Device Examples and Speeds

° I/O Speed: bytes transferred per second
(from mouse to display: million-to-1)

° Device	Behavior	Partner	Data Rate (Kbytes/sec)
Keyboard	Input	Human	0.01
Mouse	Input	Human	0.02
Line Printer	Output	Human	1.00
Floppy disk	Storage	Machine	50.00
Laser Printer	Output	Human	100.00
Magnetic Disk	Storage	Machine	10,000.00
Network-LAN	I or O	Machine	10,000.00
Graphics Display	Output	Human	30,000.00

What do we need to make I/O work?

- A way to connect many types of devices to the Proc-Mem
- A way to control these devices, respond to them, and transfer data
- A way to present them to user programs so they are useful



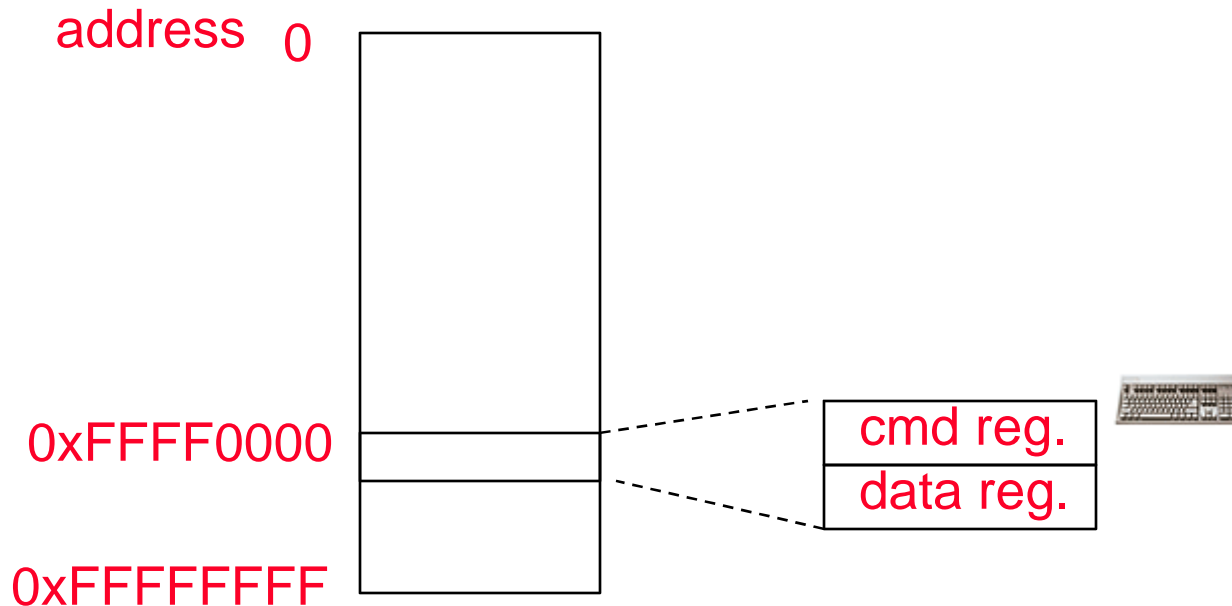
Instruction Set Architecture for

I/O

- Some machines have special input and output instructions
- Alternative model (used by MIPS):
 - Input: ~ reads a sequence of bytes
 - Output: ~ writes a sequence of bytes
- Memory also a sequence of bytes, so use loads for input, stores for output
 - Called “Memory Mapped Input/Output”
 - A portion of the address space dedicated to communication paths to Input or Output devices (no memory there)

Memory Mapped I/O

- Certain addresses are not regular memory
- Instead, they correspond to registers in I/O devices



Processor-I/O Speed

Mismatch

- **500 MHz microprocessor can execute 500 million load or store instructions per second, or 2,000,000 KB/s data rate**
 - **I/O devices from 0.01 KB/s to 30,000 KB/s**
- **Input: device may not be ready to send data as fast as the processor loads it**
 - **Also, might be waiting for human to act**
- **Output: device may not be ready to accept data as fast as processor stores it**
- **What to do?**

Processor Checks Status before Acting

- Path to device generally has 2 registers:
 - 1 register says it's OK to read/write (I/O ready), often called Control Register
 - 1 register that contains data, often called Data Register
- Processor reads from Control Register in loop, waiting for device to set Ready bit in Control reg to say its OK ($0 \Rightarrow 1$)
- Processor then loads from (input) or writes to (output) data register
 - Load from device/Store into Data Register resets Ready bit ($1 \Rightarrow 0$) of Control Register

SPIM I/O Simulation

- ° SPIM simulates 1 I/O device: memory-mapped terminal (keyboard + display)
 - Read from keyboard (**receiver**); 2 device regs
 - Writes to terminal (**transmitter**); 2 device regs

Receiver Control 0xffff0000	Unused (00...00)	(I.E.)	Ready
Receiver Data 0xffff0004	Unused (00...00)	Received Byte	
Transmitter Control 0xffff0008	Unused (00...00)	(I.E.)	Ready
Transmitter Data 0xffff000c	Unused	Transmitted Byte	

SPIM I/O

- **Control register rightmost bit (0): Ready**
 - **Receiver: Ready==1 means character in Data Register not yet been read;
1 \Rightarrow 0 when data is read from Data Reg**
 - **Transmitter: Ready==1 means transmitter is ready to accept a new character;
0 \Rightarrow Transmitter still busy writing last char**
 - I.E. bit discussed later
- **Data register rightmost byte has data**
 - **Receiver: last char from keyboard; rest = 0**
 - **Transmitter: when write rightmost byte, writes char to display**

I/O Example

◦ Input: Read from keyboard into \$v0

```
Waitloop:    lui    $t0, 0xffff #ffff0000
             lw     $t1, 0($t0) #control
             andi   $t1, $t1, 0x0001
             beq    $t1, $zero, Waitloop
             lw     $v0, 4($t0) #data
```

◦ Output: Write to display from \$a0

```
Waitloop:    lui    $t0, 0xffff #ffff0000
             lw     $t1, 8($t0) #control
             andi   $t1, $t1, 0x0001
             beq    $t1, $zero, Waitloop
             sw    $a0, 12($t0) #data
```

◦ Processor waiting for I/O called “Polling”

“What’s This Stuff Good For?”



Remote Diagnosis:
“NeoRest ExII,” a high-tech toilet features microprocessor-controlled seat warmers, automatic lid openers, air deodorizers, water sprays and blow-dryers that do away with the need for toilet tissue. About 25 percent of new homes in Japan have a “washlet,” as these toilets are called. Toto's engineers are now working on a model that analyzes urine to determine blood-sugar levels in diabetics and then automatically sends a daily report, by modem, to the user's physician.

One Digital Day, 1998

www.intel.com/onedigitalday

Cost of Polling?

- **Assume for a processor with a 500-MHz clock it takes 400 clock cycles for a polling operation (call polling routine, accessing the device, and returning). Determine % of processor time for polling**
 - **Mouse: polled 30 times/sec so as not to miss user movement**
 - **Floppy disk: transfers data in 2-byte units and has a data rate of 50 KB/second. No data transfer can be missed.**
 - **Hard disk: transfers data in 16-byte chunks and can transfer at 8 MB/second. Again, no transfer can be missed.**

% Processor time to poll mouse, floppy

◦ **Mouse Polling Clocks/sec**

$$= 30 * 400 = 12000 \text{ clocks/sec}$$

◦ **% Processor for polling:**

$$12 * 10^3 / 500 * 10^6 = 0.002\%$$

⇒ **Polling mouse little impact on processor**

◦ **Times Polling Floppy/sec**

$$= 50 \text{ KB/s} / 2\text{B} = 25\text{K polls/sec}$$

◦ **Floppy Polling Clocks/sec**

$$= 25\text{K} * 400 = 10,000,000 \text{ clocks/sec}$$

◦ **% Processor for polling:**

$$10 * 10^6 / 500 * 10^6 = 2\%$$

⇒ **OK if not too many I/O devices**

% Processor time to hard disk

◦ **Times Polling Disk/sec**

$$= 8 \text{ MB/s} / 16\text{B} = 500\text{K polls/sec}$$

◦ **Disk Polling Clocks/sec**

$$= 500\text{K} * 400 = 200,000,000 \text{ clocks/sec}$$

◦ **% Processor for polling:**

$$200 * 10^6 / 500 * 10^6 = 40\%$$

⇒ **Unacceptable**

What is the alternative to polling?

- **Wasteful to have processor spend most of its time “spin-waiting” for I/O to be ready**
- **Wish we could have an unplanned procedure call that would be invoked only when I/O device is ready**
- **Solution: use exception mechanism to help I/O. Interrupt program when I/O ready, return when done with data transfer**

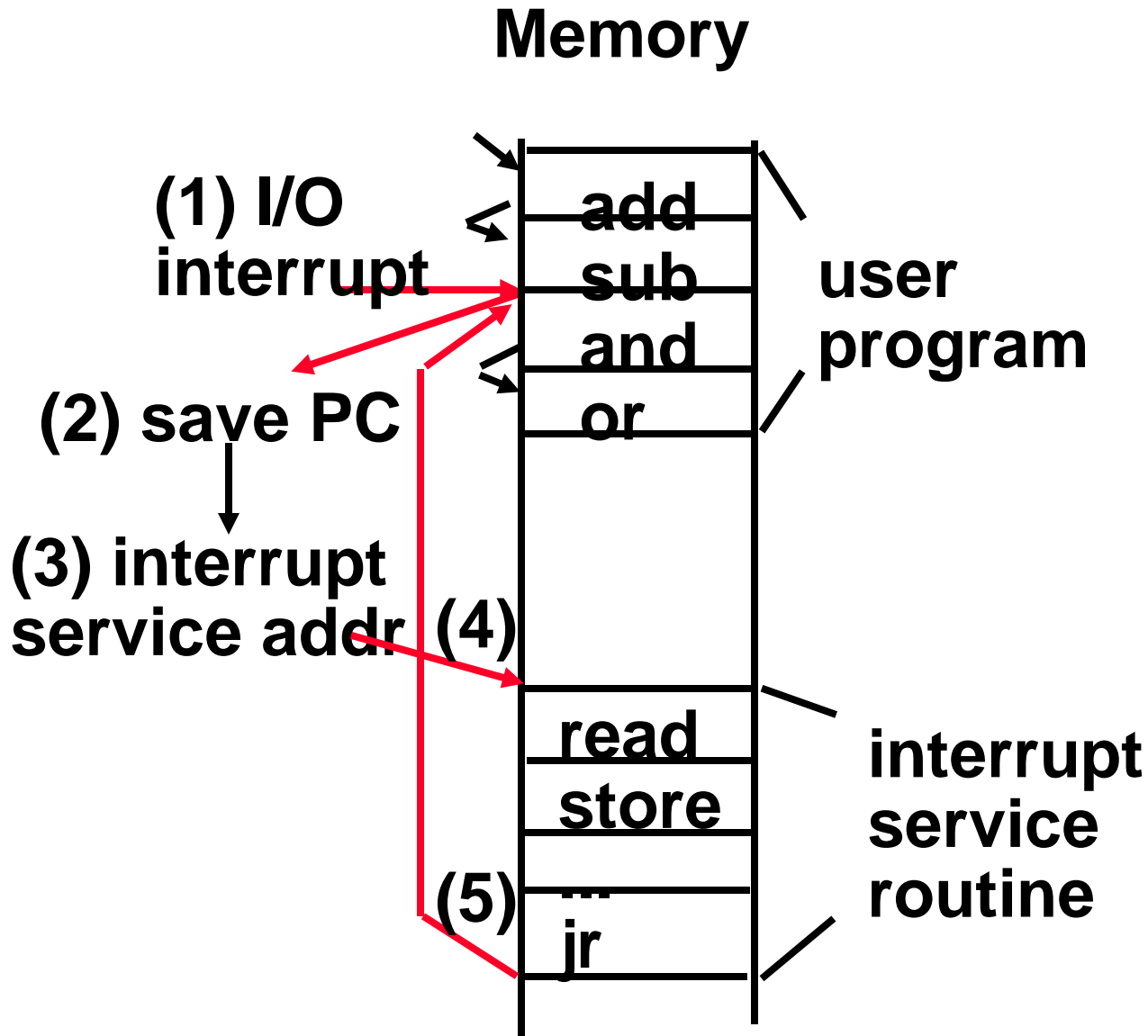
I/O Interrupt

- **An I/O interrupt is like an overflow exceptions except:**
 - **An I/O interrupt is “asynchronous”**
 - **More information needs to be conveyed**
- **An I/O interrupt is asynchronous with respect to instruction execution:**
 - **I/O interrupt is not associated with any instruction, but it can happen in the middle of any given instruction**
 - **I/O interrupt does not prevent any instruction from completion**

Definitions for Clarification

- **Exception**: signal marking that something “out of the ordinary” has happened and needs to be handled
- **Interrupt**: asynchronous exception
- **Trap**: synchronous exception
Note: These are different from the book’s definitions.

Interrupt Driven Data Transfer



Instruction Set Support for I/O Interrupt

- **Save the PC for return**
 - **But where?**
- **Where go when interrupt occurs?**
 - **MIPS defines location: 0x80000080**
- **Determine cause of interrupt?**
 - **MIPS has Cause Register, 4-bit field (bits 5 to 2) gives cause of exception**

Instruction Set Support for I/O Interrupt

- Portion of MIPS architecture for interrupts called “coprocessor 0”
- Coprocessor 0 Instructions
 - Data transfer: lwc0, swc0
 - Move: mfc0, mtc0
- Coprocessor 0 Registers:

<i>name</i>	<i>number</i>	<i>usage</i>
BadVAddr	\$8	Address of Int Status...
....	\$12	Interrupt enable
Cause	\$13	Exception type
EPC	\$14	Return address

SPIM I/O Simulation: Interrupt Driven I/O

- I.E. stands for Interrupt Enable
- Set Interrupt Enable bit to 1 have interrupt occur whenever Ready bit is set

Receiver Control 0xffff0000	Unused (00...00)	(I.E.)	Ready
Receiver Data 0xffff0004	Unused (00...00)	Received Byte	
Transmitter Control 0xffff0008	Unused (00...00)	(I.E.)	Ready
Transmitter Data 0xffff000c	Unused	Transmitted Byte	

Benefit of Interrupt-Driven I/O

- 500 clock cycle overhead for each transfer, including interrupt. Find the % of processor consumed if the hard disk is only active 5% of the time.
- Interrupt rate = polling rate
 - Disk Interrupts/sec = 8 MB/s /16B
= 500K interrupts/sec
 - Disk Polling Clocks/sec = 500K * 500
= 250,000,000 clocks/sec
 - % Processor for during transfer:
 $250 * 10^6 / 500 * 10^6 = 50\%$
- Disk active 5% \Rightarrow 5% * 50% \Rightarrow 2.5% busy₂₅

Questions Raised about Interrupts

- **Which I/O device caused exception?**
 - Needs to convey the identity of the device generating the interrupt
- **Can avoid interrupts during the interrupt routine?**
 - What if more important interrupt occurs while servicing this interrupt?
 - Allow interrupt routine to be entered again?
- **Who keeps track of status of all the devices, handle errors, know where to put/supply the I/O data?**

4 Responsibilities leading to OS

- **The I/O system is shared by multiple programs using the processor**
- **Low-level control of I/O device is complex because requires managing a set of concurrent events and because requirements for correct device control are often very detailed**
- **I/O systems often use interrupts to communicate information about I/O operations**
- **Would like I/O services for all user programs under safe control**

4 Functions OS must provide

- **OS guarantees that user's program accesses only the portions of I/O device to which user has rights (e.g., file access)**
- **OS provides abstractions for accessing devices by supplying routines that handle low-level device operations**
- **OS handles the exceptions generated by I/O devices (and arithmetic exceptions generated by a program)**
- **OS tries to provide equitable access to the shared I/O resources, as well as schedule accesses in order to enhance system performance**

Things to Remember

- **I/O gives computers their 5 senses**
- **I/O speed range is million to one**
- **Processor speed means must synchronize with I/O devices before use**
- **Polling works, but expensive**
 - **processor repeatedly queries devices**
- **Interrupts works, more complex**
 - **devices causes an exception, causing OS to run and deal with the device**
- **I/O control leads to Operating Systems**