

Lecture 38
Input / Output

2004-11-29



Lecturer PSOE Dan Garcia
www.cs.berkeley.edu/~ddgarcia

\$100 "Green" PC

Runs Linux (DSL), operates on 12V DC power, (solar panel, car battery, or bicycle generator) & uses 10W, a fraction of today's PCs.



CS61C L38 I/O (1)

solarlite.org

Garcia, Fall 2004 © UCB

Review

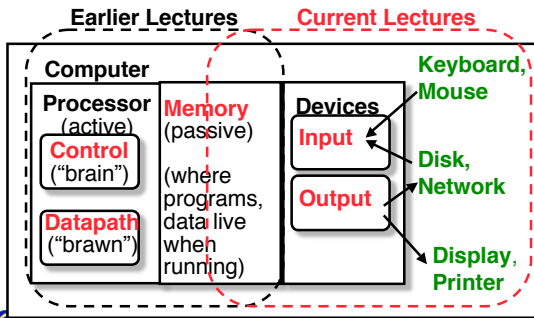
- Virtual memory to Physical Memory Translation too slow?
 - Add a cache of Virtual to Physical Address Translations, called a **TLB**
- Spatial Locality means Working Set of Pages is all that must be in memory for process to run fairly well
- Virtual Memory allows protected sharing of memory between processes with less swapping to disk



CS61C L38 I/O (2)

Garcia, Fall 2004 © UCB

Recall : 5 components of any Computer



CS61C L38 I/O (3)

Garcia, Fall 2004 © UCB

Motivation for Input/Output

- I/O is how humans interact with computers
- I/O gives computers long-term memory.
- 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)
- Computer without I/O like a car without wheels; great technology, but won't get you anywhere



CS61C L38 I/O (4)

Garcia, Fall 2004 © UCB

I/O Device Examples and Speeds

• I/O Speed: bytes transferred per second (from mouse to Gigabit LAN: 100-million-to-1)

Device	Behavior	Partner	Data Rate (KBytes/s)
Keyboard	Input	Human	0.01
Mouse	Input	Human	0.02
Voice output	Output	Human	5.00
Floppy disk	Storage	Machine	50.00
Laser Printer	Output	Human	100.00
Magnetic Disk	Storage	Machine	10,000.00
Wireless Network	I or O	Machine	10,000.00
Graphics Display	Output	Human	30,000.00
Wired LAN Network	I or O	Machine	1,000,000.00



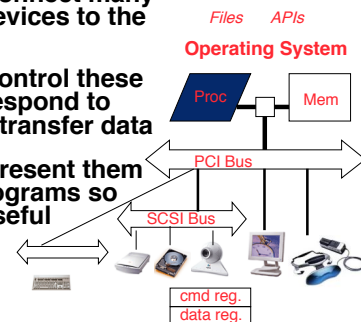
When discussing transfer rates, use 10^x

CS61C L38 I/O (5)

Garcia, Fall 2004 © UCB

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



CS61C L38 I/O (6)

Garcia, Fall 2004 © UCB

Instruction Set Architecture for I/O

- What must the processor do for I/O?
 - Input: reads a sequence of bytes
 - Output: writes a sequence of bytes
- Some processors have special input and output instructions
- Alternative model (used by MIPS):
 - 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)

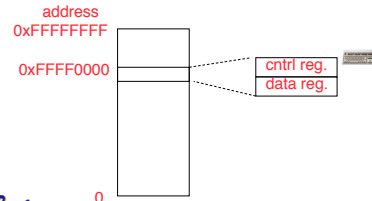


CS61C L39 I/O (7)

Garcia, Fall 2004 © UCB

Memory Mapped I/O

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



CS61C L39 I/O (8)

Garcia, Fall 2004 © UCB

Processor-I/O Speed Mismatch

- 1GHz microprocessor can execute 1 billion load or store instructions per second, or 4,000,000 KB/s data rate
 - I/O devices data rates range from 0.01 KB/s to 1,000,000 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 not be ready to accept data as fast as processor stores it

• What to do?



CS61C L39 I/O (9)

Garcia, Fall 2004 © UCB

Processor Checks Status before Acting

- Path to device generally has 2 registers:
 - **Control Register**, says it's OK to read/write (I/O ready) [think of a flagman on a road]
 - **Data Register**, contains data
- Processor reads from Control Register in loop, waiting for device to set **Ready** bit in Control reg (0 ⇒ 1) to say its OK
- Processor then loads from (input) or writes to (output) data register
 - Load from or Store into Data Register resets Ready bit (1 ⇒ 0) of Control Register



CS61C L39 I/O (10)

Garcia, Fall 2004 © UCB

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)	Ready (1E)
Receiver Data 0xffff0004	Unused (00...00)	Received Byte
Transmitter Control 0xffff0008	Unused (00...00)	Ready (1E)
Transmitter Data 0xffff000c	Unused	Transmitted Byte



CS61C L39 I/O (11)

Garcia, Fall 2004 © UCB

SPIM I/O

- Control register rightmost bit (0): Ready
 - Receiver: Ready==1 means character in Data Register not yet been read; 1 ⇒ 0 when data is read from Data Reg
 - Transmitter: Ready==1 means transmitter is ready to accept a new character; 0 ⇒ 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



CS61C L39 I/O (12)

Garcia, Fall 2004 © UCB

I/O Example

- Input: Read from keyboard into `$v0`

```
Waitloop:    lui    $t0, 0xffff #ffff0000
            lw     $t1, 0($t0) #control
            andi  $t1, $t1, 0x1
            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, 0x1
            beq  $t1, $zero, Waitloop
            sw     $a0, 12($t0) #data
```

- Processor waiting for I/O called “Polling”

- “Ready” bit from processor’s point of view!



CS61C L39 I/O (13)

Garcia, Fall 2004 © UCB

Cost of Polling?

- Assume for a processor with a 1GHz 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 16 MB/second. Again, no transfer can be missed.



CS61C L39 I/O (15)

Garcia, Fall 2004 © UCB

% Processor time to poll [p. 677 in book]

Mouse Polling, Clocks/sec

$$= 30 \text{ [polls/s]} * 400 \text{ [clocks/poll]} = 12\text{K [clocks/s]}$$

- % Processor for polling:

$$12 * 10^3 \text{ [clocks/s]} / 1 * 10^9 \text{ [clocks/s]} = 0.0012\%$$

⇒ Polling mouse little impact on processor

Frequency of Polling Floppy

$$= 50 \text{ [KB/s]} / 2 \text{ [B/poll]} = 25\text{K [polls/s]}$$

- Floppy Polling, Clocks/sec

$$= 25\text{K [polls/s]} * 400 \text{ [clocks/poll]} = 10\text{M [clocks/s]}$$

- % Processor for polling:

$$10 * 10^6 \text{ [clocks/s]} / 1 * 10^9 \text{ [clocks/s]} = 1\%$$

⇒ OK if not too many I/O devices



CS61C L39 I/O (16)

Garcia, Fall 2004 © UCB

% Processor time to poll hard disk

Frequency of Polling Disk

$$= 16 \text{ [MB/s]} / 16 \text{ [B]} = 1\text{M [polls/s]}$$

- Disk Polling, Clocks/sec

$$= 1\text{M [polls/s]} * 400 \text{ [clocks/poll]}$$

$$= 400\text{M [clocks/s]}$$

- % Processor for polling:

$$400 * 10^6 \text{ [clocks/s]} / 1 * 10^9 \text{ [clocks/s]} = 40\%$$

⇒ Unacceptable



CS61C L39 I/O (17)

Garcia, Fall 2004 © UCB

What is the alternative to polling?

- Wasteful to have processor spend most of its time “spin-waiting” for I/O to be ready
- Would like 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



CS61C L39 I/O (18)

Garcia, Fall 2004 © UCB

I/O Interrupt

- An I/O interrupt is like 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



CS61C L39 I/O (19)

Garcia, Fall 2004 © UCB

Definitions for Clarification

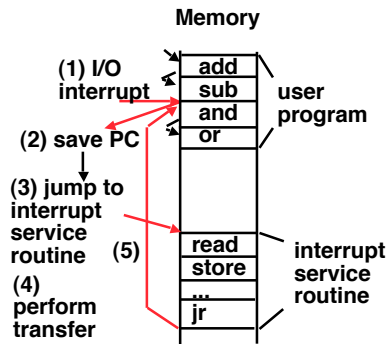
- **Exception:** signal marking that something “out of the ordinary” has happened and needs to be handled
- **Interrupt:** asynchronous exception
- **Trap:** synchronous exception
- **Note:** Many systems folks say “interrupt” to mean what we mean when we say “exception”.



CS61C L39 I/O (28)

Garcia, Fall 2004 © UCB

Interrupt Driven Data Transfer



CS61C L39 I/O (21)

Garcia, Fall 2004 © UCB

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



CS61C L39 I/O (22)

Garcia, Fall 2004 © UCB

Benefit of Interrupt-Driven I/O

- Find the % of processor consumed if the hard disk is only active 5% of the time. Assuming 500 clock cycle overhead for each transfer, including interrupt:
 - Disk Interrupts/s = 16 MB/s / 16B/interrupt = 1M interrupts/s
 - Disk Interrupts, clocks/s = 1M interrupts/s * 500 clocks/interrupt = 500,000,000 clocks/s
 - % Processor for during transfer: $500 \times 10^6 / 1 \times 10^9 = 50\%$
- Disk active 5% \Rightarrow 5% * 50% \Rightarrow 2.5% busy



CS61C L39 I/O (23)

Garcia, Fall 2004 © UCB

Peer Instruction

- A faster CPU will result in faster I/O.
- Hardware designers handle mouse input with interrupts since it is better than polling in almost all cases.
- Low-level I/O is actually quite simple, as it's really only reading and writing bytes.

	ABC
1:	FFF
2:	FFT
3:	FTF
4:	FTT
5:	TFF
6:	TFT
7:	FTT
8:	TTT



CS61C L39 I/O (26)

Garcia, Fall 2004 © UCB

“And in conclusion...”

- I/O gives computers their 5 senses
- I/O speed range is 100-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**



CS61C L39 I/O (26)

Garcia, Fall 2004 © UCB