With the rapid deployment of “Smart Grid” technology, such as the smart meters PG&E is currently deploying throughout all of Berkeley, questions are being raised as to the safety and security and such devices.
Goals for today:

• Today we’re going to be giving you a VERY high level overview of system input/output. The goal is not to make you experts in any one area, but simply to give you brief exposure in a number of areas.

• We’ll also start talking about system performance, which we’ll finish up tomorrow.

• We’re going to start with:

  I/O Basics
Recall: 5 components of any Computer

- Processor (active)
- Control ("brain")
- Datapath ("brawn")
- Memory (passive)
- Devices
  - Input
  - Output

Earlier Lectures

Current Lecture

- Keyboard, Mouse
- Disk, Network
- Display, Printer
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
Memory Mapped I/O

• Certain addresses are not regular memory

• Instead, they correspond to registers in I/O devices

address
0xFFFFFFFF

0xFFFF0000

cntrl reg.
data reg.

0
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

• This is called “Polling”
What is the alternative to polling?

• Wasteful to have processor spend most of its time “spin-waiting” for I/O to be ready if events are infrequent
  
  • 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 125,000 KB/s
  
  • Note: When discussing transfer rates, use $10^x$

• 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
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
Interrupt-Driven Data Transfer

1. I/O interrupt
2. Save PC
3. Jump to interrupt service routine
4. Perform transfer
5. Read, store, ...

Memory

add
sub
and
or

User program

Interrupt service routine

Cal

CS61C L26 IO & Performance I (9)
disks
Magnetic Disk – common I/O device

• A kind of computer memory
  • Information stored by magnetizing ferrite material on surface of rotating disk

• Nonvolatile storage
  • retains its value without applying power to disk.

• Two Types
  • Floppy disks – slower, less dense, removable, non-existent today
  • Hard Disk Drives (HDD) – faster, more dense, non-removable.

• Purpose in computer systems (Hard Drive):
  • Long-term, inexpensive storage for files
  • “Backup” for main-memory. Large, inexpensive, slow level in the memory hierarchy (virtual memory)
Photo of Disk Head, Arm, Actuator

- Spindle
- Arm
- Head
- Actuator
- Platters (1-12)
Administrivia

• Project 2 graded under way, if you haven’t signed up for a slot…. You should get on that.

• Project 3 out now
  • No partners for this project!

• Grading underway for rest of the assignments. Will get them done ASAP

• Don’t forget to register your iClicker
  • http://www.iclicker.com/registration/

• Final Thursday August 12th from 8am to 11am in 10 Evans.
Disk Device Terminology

- Several platters, with information recorded magnetically on both surfaces (usually)
- Bits recorded in **tracks**, which in turn divided into **sectors** (e.g., 512 Bytes)
- **Actuator** moves **head** (end of **arm**) over track ("**seek**"), wait for **sector** rotate under **head**, then read or write
Disk Device Performance (1/2)

- Disk Latency = Seek Time + Rotation Time + Transfer Time + Controller Overhead
  - Seek Time? depends on no. tracks to move arm, speed of actuator
  - Rotation Time? depends on speed disk rotates, how far sector is from head
  - Transfer Time? depends on data rate (bandwidth) of disk (f(bit density, rpm)), size of request
Disk Device Performance (2/2)

• Average distance of sector from head?
  • 1/2 time of a rotation
    • 7200 Revolutions Per Minute ⇒ 120 Rev/sec
    • 1 revolution = 1/120 sec ⇒ 8.33 milliseconds
    • 1/2 rotation (revolution) ⇒ 4.17 ms

• Average no. tracks to move arm?
  • Disk industry standard benchmark:
    ▪ Sum all time for all possible seek distances from all possible tracks / # possible
    ▪ Assumes average seek distance is random

• Size of Disk cache can strongly affect perf!
  • Cache built into disk system, OS knows nothing
Where does Flash memory come in?

• Microdrives and Flash memory (e.g., CompactFlash) are going head-to-head
  • Both non-volatile (no power, data ok)
  • Flash benefits: durable & lower power (no moving parts, need to spin µdrives up/down)
  • Flash limitations: finite number of write cycles (wear on the insulating oxide layer around the charge storage mechanism). Most $\geq 100K$, some $\geq 1M$ W/erase cycles.

• How does Flash memory work?
  • NMOS transistor with an additional conductor between gate and source/drain which “traps” electrons. The presence/absence is a 1 or 0.

en.wikipedia.org/wiki/Flash_memory
What does Apple put in its iPods?

- Toshiba flash 1, 2GB
- Samsung flash 4, 8GB
- Toshiba 1.8-inch HDD 80, 160GB
- Toshiba flash 8, 16, 32GB

shuffle, nano, classic, touch
RAID : Redundant Array of Inexpensive Disks

• Invented @ Berkeley (1989) by Patterson, Katz, et al.

• A multi-billion industry
  80% non-PC disks sold in RAIDs

• Idea:
  • Files are “striped” across multiple disks in a redundant manor
  • Redundancy yields high data availability
    ▪ Disks will still fail
  • Contents reconstructed from data redundantly stored in the array
    ▪ ⇒ Capacity penalty to store redundant info
    ▪ ⇒ Bandwidth penalty to update redundant info
Performance
Why Performance? Faster is better!

- **Purchasing Perspective**: given a collection of machines (or upgrade options), which has the
  - best performance?
  - least cost?
  - best performance / cost?

- **Computer Designer Perspective**: faced with design options, which has the
  - best performance improvement?
  - least cost?
  - best performance / cost?

- All require basis for comparison and metric for evaluation!
  - Solid metrics lead to solid progress!
Two Notions of “Performance”

<table>
<thead>
<tr>
<th>Plane</th>
<th>DC to Paris</th>
<th>Top Speed</th>
<th>Passengers</th>
<th>Throughput (pmph)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boeing 747</td>
<td>6.5 hours</td>
<td>610 mph</td>
<td>470</td>
<td>286,700</td>
</tr>
<tr>
<td>BAD/Sud Concorde</td>
<td>3 hours</td>
<td>1350 mph</td>
<td>132</td>
<td>178,200</td>
</tr>
</tbody>
</table>

• Which has higher performance?
  • Interested in time fly one plane?
  • Interested in delivering as many passengers per day as possible?

• In a computer, time for one task called **Latency** or **Response Time** or **Execution Time**

• In a computer, tasks per unit time called **Throughput** or **Bandwidth**
Definitions

• Performance is in units of things per sec
  • bigger is better

• If mostly concerned with response time
  • \( \text{performance}(x) = \frac{1}{\text{execution\_time}(x)} \)

• “F(ast) is \( n \) times faster than S(low)” means:

\[
\frac{\text{performance}(F)}{\text{execution\_time}(S)} = n = \frac{\text{performance}(S)}{\text{execution\_time}(F)}
\]
Example of Response Time v. Throughput

• Time of Concorde vs. Boeing 747?
  • Concord is $\frac{6.5 \text{ hours}}{3 \text{ hours}} = 2.2$ times faster than the Boeing
  • Concord is 2.2 times (“120%”) faster in terms of flying time (response time)

• Throughput of Boeing vs. Concorde?
  • Boeing 747: 286,700 pmph / 178,200 pmph = 1.6 times faster
  • Boeing is 1.6 times (“60%”) faster in terms of throughput

• We will focus primarily on response time.
What is Time?

• Straightforward definition of time:
  • Total time to complete a task, including disk accesses, memory accesses, I/O activities, operating system overhead, ...
  • “real time”, “response time” or “elapsed time”

• Alternative: just time processor (CPU) is working only on your program (since multiple processes running at same time)
  • “CPU execution time” or “CPU time”
  • Often divided into system CPU time (in OS) and user CPU time (in user program)
How to Measure Time?

- **Real Time** ⇒ Actual time elapsed

- **CPU Time**: Computers constructed using a clock that runs at a constant rate and determines when events take place in the hardware
  
  - These discrete time intervals called clock cycles (or informally clocks or cycles)
  
  - Length of clock period: clock cycle time (e.g., ½ nanoseconds or ½ ns) and clock rate (e.g., 2 gigahertz, or 2 GHz), which is the inverse of the clock period; use these!
Measuring Time using Clock Cycles (1/2)

- CPU execution time for a program
  - Units of \([\text{seconds} / \text{program}]\) or \([\text{s/p}]\)

  = Clock Cycles for a program \(\times\) Clock Period
    - Units of \([\text{s/p}] = [\text{cycles} / \text{p}] \times [\text{s} / \text{cycle}] = [\text{c/p}] \times [\text{s/c}]\)

- Or

  = \(\frac{\text{Clock Cycles for a program}}{\text{Clock Rate}}\)\[c / p\]
Measuring Time using Clock Cycles (2/2)

• One way to define clock cycles:

Clock Cycles for program \([c/p]\)

= Instructions for a program \([i/p]\)
  (called “Instruction Count”)

× Average Clock cycles Per Instruction \([c/i]\)
  (abbreviated “CPI”)

• CPI one way to compare two machines with same instruction set, since Instruction Count would be the same
Peer Instruction

1) Processors must communicate with I/O devices via special instructions.

2) For devices with low I/O event frequency polling is a better choice than interrupts.
“And in conclusion...”

- I/O gives computers their 5 senses
- Vast I/O speed range
- 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
- Latency v. Throughput
- **Real Time**: time user waits for program to execute: depends heavily on how OS switches between tasks
- **CPU Time**: time spent executing a single program: depends solely on design of processor (datapath, pipelining effectiveness, caches, etc.)
Bonus slides

• These are extra slides that used to be included in lecture notes, but have been moved to this, the “bonus” area to serve as a supplement.

• The slides will appear in the order they would have in the normal presentation.
The Internet (1962)

www.computerhistory.org/internet_history

• Founders
  • JCR Licklider, as head of ARPA, writes on “intergalactic network”
  • 1963 : ASCII becomes first universal computer standard
  • 1969 : Defense Advanced Research Projects Agency (DARPA) deploys 4 “nodes” @ UCLA, SRI, Utah, & UCSB
  • 1973 Robert Kahn & Vint Cerf invent TCP, now part of the Internet Protocol Suite

• Internet growth rates
  • Exponential since start!

www.greatachievements.org/?id=3736
en.wikipedia.org/wiki/Internet_Protocol_Suite

Revolutions like this don't come along very often
Why Networks?

• Originally sharing I/O devices between computers
  • E.g., printers

• Then communicating between computers
  • E.g., file transfer protocol

• Then communicating between people
  • E.g., e-mail

• Then communicating between networks of computers
  • E.g., file sharing, www, …

- “System of interlinked hypertext documents on the Internet”

- History
  - 1945: Vannevar Bush describes hypertext system called “memex” in article
  - 1989: Tim Berners-Lee proposes, gets system up ’90
  - ~2000 Dot-com entrepreneurs rushed in, 2001 bubble burst

- Wayback Machine
  - Snapshots of web over time

- Today: Access anywhere!

en.wikipedia.org/wiki/History_of_the_World_Wide_Web

Tim Berners-Lee

World’s First web server in 1990

WayBack Machine

www.archive.org

Cal

CS61C L26 IO & Performance I (35)

Pearce, Summer 2010 © UCB
Shared vs. Switched Based Networks

- **Shared vs. Switched:**
  - **Switched:** pairs (“point-to-point” connections) communicate at same time
  - **Shared:** 1 at a time (CSMA/CD)

- **Aggregate bandwidth (BW) in switched network is many times shared:**
  - point-to-point faster since no arbitration, simpler interface
What makes networks work?

- **links** connecting **switches** to each other and to computers or devices

- ability to **name** the components and to **route** packets of information - messages - from a source to a destination

- Layering, redundancy, protocols, and encapsulation as means of **abstraction** (61C big idea)
“RAID 0”: No redundancy = “AID”

- Assume have 4 disks of data for this example, organized in blocks

- Large accesses faster since transfer from several disks at once

This and next 5 slides from RAID.edu, http://www.acnc.com/04_01_00.html
http://www.raid.com/04_00.html also has a great tutorial
RAID 1: Mirror data

- Each disk is fully duplicated onto its "mirror"
  - Very high availability can be achieved
- Bandwidth reduced on write:
  - 1 Logical write = 2 physical writes
- Most expensive solution: 100% capacity overhead
RAID 3: Parity

- Parity computed across group to protect against hard disk failures, stored in P disk
- Logically, a single high capacity, high transfer rate disk
- 25% capacity cost for parity in this example vs. 100% for RAID 1 (5 disks vs. 8 disks)
Inspiration for RAID 5 (RAID 4 block-stripping)

- **Small writes (write to one disk):**
  - Option 1: read other data disks, create new sum and write to Parity Disk (access all disks)
  - Option 2: since P has old sum, compare old data to new data, add the difference to P:
    - 1 logical write = 2 physical reads + 2 physical writes to 2 disks

- **Parity Disk is bottleneck for Small writes:**
  Write to A0, B1 € both write to P disk
RAID 5: Rotated Parity, faster small writes

- Independent writes possible because of interleaved parity
  - Example: write to A0, B1 uses disks 0, 1, 4, 5, so can proceed in parallel
  - Still 1 small write = 4 physical disk accesses

en.wikipedia.org/wiki/Redundant_array_of_independent_disks
Words, Words, Words…

• Will (try to) stick to “n times faster”; its less confusing than “m % faster”

• As faster means both decreased execution time and increased performance, to reduce confusion we will (and you should) use “improve execution time” or “improve performance”