Plan for Today

• Reminders
  – Project report due Dec 7\textsuperscript{th}
    • Bring hardcopy to class, AND
    • Send to Patrick & Shakir via email
  – Please complete a course evaluation:
    • evals.wustl.edu

• Questions

• End of semester discussion

• Today’s discussion:
  – Disks
  – I/O
  – embedded memory organizations
Semester’s End

• Final lecture: Wednesday, Dec 2

• Come to class on Monday, Dec 7
  – Turn in final report hard copy
  – Pick up final exam

• Final exam due to me via email by 4pm on Dec 14
Magnetic disks (hard disks)

- Non-volatile storage for files
- Outer-most level of memory hierarchy
  - Backing store for virtual memory
Magnetic Disk Organization
Magnetic Disk Organization (cont’d)

- **Platters**
  - 1 to 20
  - Diameter from 1.0 to 8 inches (both sides are used to store information)
  - 3,600–15,000 RPM

- **Tracks**
  - Thousands to tens of thousands per platter

- **Sectors**
  - 100–500 per track
  - Typically 512 bytes per sector
  - Trend: constant bit density: outer tracks have more sectors

- Example:
  
  Capacity = \( \text{# platters} \times \text{# tracks} \times \text{# sectors} \times \text{sector size} \)
  
  \[ = 12 \times 28,200 \times 424 \times 512 = \text{approx. 68 Gbytes} \]
Disk Access Time

- Two dynamic actions
  - Spinning the disk
  - Positioning arm, with read/write head, to proper location

- Four components in an access
  - **Seek time**
    - to move the arm to the correct cylinder.
    - from 0 to 12 ms.
  - **Rotation time**
    - to move on the correct sector
    - on the average, $\frac{1}{2}$ rotation
    - @ 3600 RPM, 8.3 ms (Recall: 3600 RPM - 10000 RPM).
  - **Transfer time**
    - to read-write data
    - depends on RPM, amount to transfer, recording density, and disk/mem connection.
    - current transfer rates: from 2 MB/s to 40MB/s.
  - **Disk controller time**
    - overhead to perform an access
    - around 1ms.
Disk Improvements

• **Read ahead:**
  – Amortize long access by reading more than one request
  – Idea:
    • Leverage spatial locality
    • Use buffers on the disk acting as caches (0.125 MB to 4MB)
    • Transfers to/from buffer at the speed of I/O bus

• **Capacity** has improved (via areal density)
  – Traditionally seen same growth rate as DRAMs, 60% per year
  – 100 % per year since 1997.

• **Price** has improved at least that much
  – Recently ~$0.06/GB

• **Access times** improve slowly
  – Higher density → smaller drives → smaller seek times
  – RPM has increased (3600 to 10K+)
  – Transfer time is increasing
Disk Improvements (cont’d)

- CPU speed vs. DRAM access is one “memory wall”

- DRAM access time vs. Disk access time is “memory gap”
  - 150 ns vs. 15 ms: 100,000 difference
  - No technology has yet filled this gap; Flash might!

- NAND Flash recently 10x cheaper v. DRAM, 1000x faster v. Disk
  - Samsung OneNAND: ~19 us access time
  - Intel SS72 NAND: 25 us random access time, 30ns serial
  - Recent prices
    - Flash SSD: ~$3.65/GB
    - USB stick: ~$1.63/GB
Price Trends (cont’d)
Cost vs. Access Time

© 2003 Elsevier Science (USA). All rights reserved.
Scheduling of Disk Requests

• If many requests arrive at “the same time”, it’s possible for the controller to schedule requests to reduce seek time.

• Scheduling algorithms
  – **FIFO**: simple, no optimization
  – **SSTF** (shortest seek time first): great on average, but poor variance; possibility of starvation for requests on the center and on the periphery
  – **Scan**: repeatedly have the arm move from the center to the periphery and back, meeting requests as an elevator does
  – **Elevator extension**: improvement on Scan; do not go to extremes unless needed
Buses

• *Shared* communication links
  – *Pros:*
    • *Low cost:* single shared interconnect
    • *Versatility:* Easy to add devices
      – Although this comes with a cost (shorter, homogeneous buses are faster)
  – *Cons:*
    • Communication bottleneck
      – A single shared resource can be saturated
Buses (cont’d)

• Two basic types
  – CPU/memory bus
    • maximizes memory-CPU bandwidth
    • high speed
    • short

  – I/O bus
    • possibly long
    • wide range of data bandwidth
    • wide range of connected devices
    • follows some industry standard
Connecting CPU, Memory and I/O: Buses
Bus Transactions

- Read, Write, Atomic Read-Modify-Write
  - Send an *address*
  - In the case of a read return the *data*

- Arbitration: who gets the bus, and when?
  - Only one master
    - centralized arbitration
  - Multiple masters (most common case)
    - centralized arbitration (FIFO, round-robin, etc.)
    - decentralized arbitration (each device assigned a priority)

- Communication protocol between master and servant
  - Synchronous
    - for short buses, no clock skew
    - fast
    - i.e., CPU-memory
  - Asynchronous
    - Hand-shaking finite state machine
    - Easier to accommodate many devices
Split-transaction Buses

- Scenario: multiple masters

- Idea: use packets, instead than holding bus for full transaction

- Split a transaction into
  - Send address (CPU is bus master)
  - Send data (Memory is bus master)

- Can have more concurrency by
  - having different transactions use the data and address lines concurrently
  - Making bus available to other masters while waiting for a reply

- Useful for multiprocessor systems and I/O

- Control overhead: higher bandwidth but higher latency
I/O Operations

• Special I/O instructions
  – Intel x86 (I/O has a separate address space)
    • IN(S) and OUT(S) between EAX register and I/O port whose address
      is either an immediate or in the DX register

• Memory-mapped I/O
  – Portions of address space devoted to I/O devices
    • read/write to these addresses transfer data
    • some addresses use to control I/O devices

• I/O invocation
  – Polling
    • CPU periodically checks status bit on devices
    • is implemented in certain devices
  – Interrupt-driven I/O
    • CPU works on some process while waiting from I/O
    • common case
    • operating system overhead
Direct Memory Access (DMA)

- Having long blocks of I/O go through the processor is often inefficient

- DMA controller:
  - Specialized processor for transfer of blocks between memory and I/O devices without intervention from CPU, except at the beginning and end
  - Has registers set up by CPU
    - base memory address and count
  - DMA device interrupts CPU at end of transfer
  - DMA device is a master on the bus
  - Extensions: I/O or channel processors
    - Can handle a table or list of DMA transactions
    - Generally only move and do not modify data
Memory Structures for Embedded Computers

• Generally
  – Make heavy use of DMA-like distributed I/O processing
  – Usually support large numbers and types of interfaces
  – Small, deeply embedded systems usually do not use virtual memory
AMD’s Alchemy Au1000

- 400 MHz @ .5 Watt
  - 266 MHz and 500 MHz
- OS: Win CE, Linux, VxWorks
- Used to implement “devices”
  - Printers
  - WAPs
  - Gateways
  - Terminals
  - Set-top boxes
Assignment

• Readings
  – For Wednesday
    • No reading

  – For Monday
    • Final project report due