Lec 23: I/O and Disks

Kavita Bala

CS 3410, Fall 2008

Computer Science

Cornell University

Traps

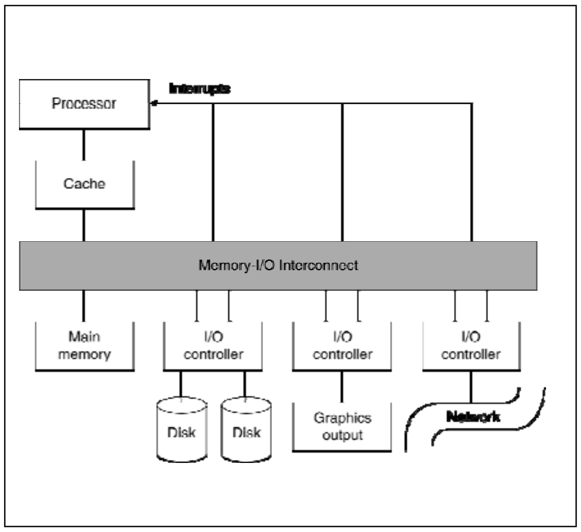
- Trap
  - Any kind of a control transfer to the OS
- Syscall
  - Synchronous, program-initiated control transfer from user to the OS to obtain service from the OS
  - e.g. SYSCALL
- Exception
  - Asynchronous, program-initiated control transfer from user to the OS in response to an exceptional event
  - e.g. Divide by zero
- Interrupt
  - Asynchronous, device-initiated control transfer from user to the OS
  - e.g. Clock tick, network packet

Reading

- Chapter 7 (VM) in 3rd edition (or Chapter 5)
- Chapter 8 (Disks) (or Chapter 6)

Challenge

- How do we interface to other devices
  - Keyboard
  - Mouse
  - Disk
  - Network
  - Printer
  - …



I/O System Characteristics

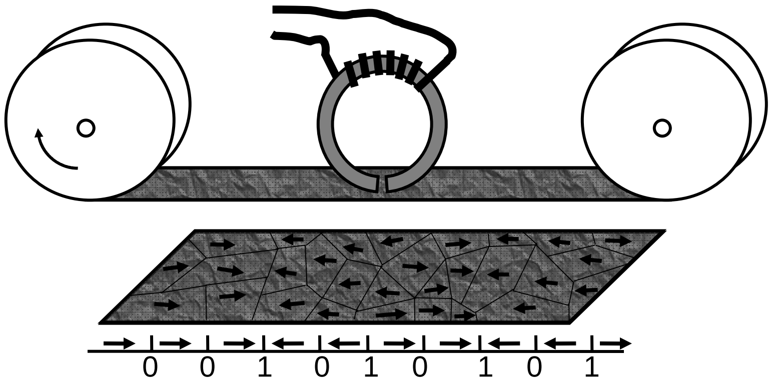
- Dependability is important
  - Particularly for storage devices
- Performance measures
  - Latency (response time)
  - Throughput (bandwidth)
  - Desktops & embedded systems
    - Mainly interested in response time & diversity of devices
  - Servers
    - Mainly interested in throughput & expandability of devices

Memory Hierarchy

16 KB	registers/L1	2 ns, random access
512 KB	L2	5 ns, random access
2 GB	DRAM	20-80 ns, random access
300 GB	Disk	2-8 ms, random access
1 TB	Tape	100s, sequential access

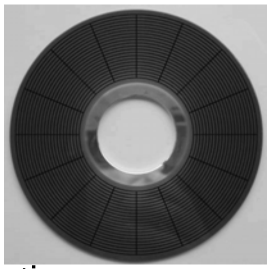
Tapes

- Same basic principle for 8-tracks cassettes, VHS, atari tape drive, tape storage

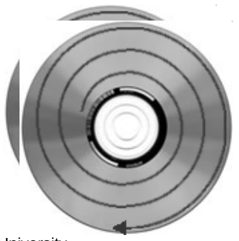


Disks & CDs

- Disks use same magnetic medium as tapes
  - concentric rings (not a spiral)

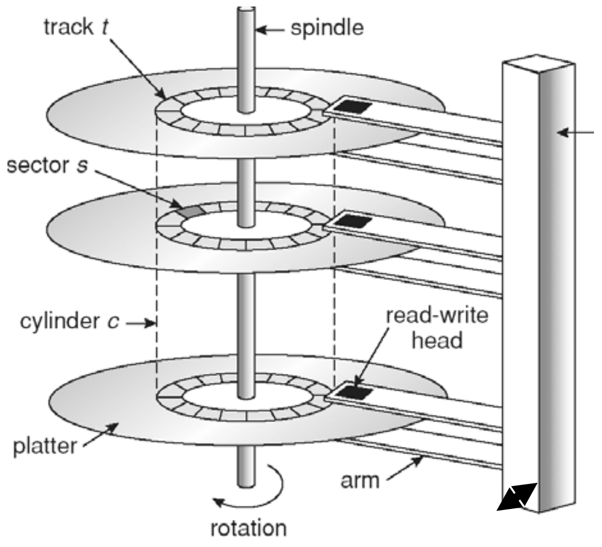


- CDs & DVDs use optics, and a single spiral track



- Non-volatile

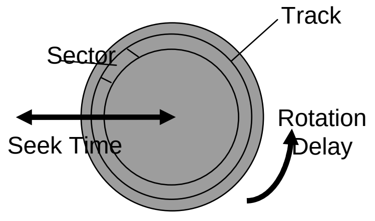
Disk Physics



Typical parameters :
1 spindle
1 arm assembly
1-4 platters
1-2 sides/platter
1 head per side
(but only 1 active head at a time)
5,400 – 15,000 RPM

Disk Accesses

- Accessing a disk requires:
  - – specify sector: C (cylinder), H (head), and S (sector)
  - – specify size: number of sectors to read or write
  - – specify memory address
- Performance:
  - seek time: move the arm assembly to track
  - Rotational delay: wait for sector to come around
  - transfer time: get the bits off the disk
  - Controller time: time for setup



Example

- Average time to read/write 512-byte sector
  - – Disk rotation at 10,000 RPM
  - – Seek time: 6ms
  - – Transfer rate: 50 MB/sec
  - – Controller overhead: 0.2 ms
- Average time:
  - – Seek time + rotational delay + transfer time + controller overhead
  - – 6ms + 0.5 rotation/(10,000 RPM) + 0.5KB/(50 MB/sec) + 0.2ms
  - – 6.0 + 3.0 + 0.01 + 0.2 = 9.2ms

Disk Access Example

- If actual average seek time is 2ms
  - Average read time = 5.2ms

Disk Scheduling

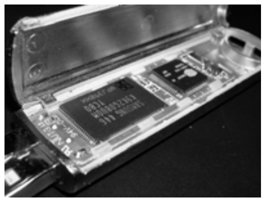
- Goal: minimize seek time
  - secondary goal: minimize rotational latency
- FCFS (First come first served)
- Shortest seek time
- SCAN/Elevator
  - First service all requests in one direction
  - Then reverse and serve in opposite direction
- Circular SCAN
  - Go off the edge and come to the beginning and start all over again

Disk Geometry: LBA

- New machines use *logical block addressing* instead of CHS
  - machine presents illusion of an array of blocks, numbered 0 to N
- Modern disks…
  - have varying number of sectors per track
    - roughly constant data density over disk
    - varying throughput over disk
  - remap and reorder blocks (to avoid defects)
  - completely obscure their actual physical geometry

Flash Storage

- Nonvolatile semiconductor storage
  - 100× – 1000× faster than disk
  - Smaller, lower power
  - But more $/GB (between disk and DRAM)



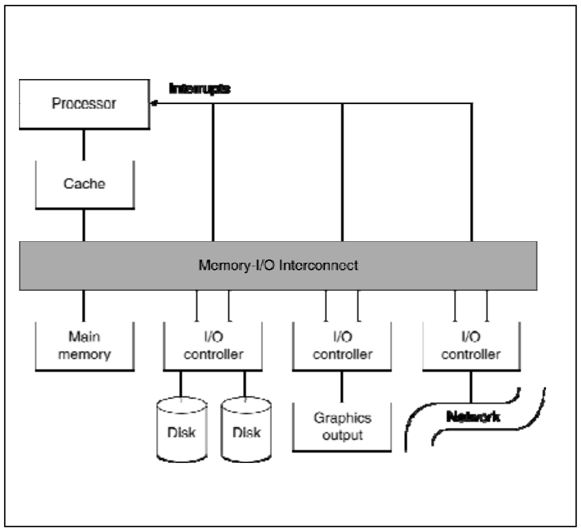
Flash Types

- NOR flash: bit cell like a NOR gate
  - Random read/write access
  - Used for instruction memory in embedded systems
- NAND flash: bit cell like a NAND gate
  - Denser (bits/area), but block-at-a-time access
  - Cheaper per GB
  - Used for USB keys, media storage, …
- Flash bits wears out after 1000’s of accesses
  - Not suitable for direct RAM or disk replacement

I/O vs. CPU Performance

- Amdahl’s Law
  - Don’t neglect I/O performance as parallelism increases compute performance
- Example
  - Benchmark takes 90s CPU time, 10s I/O time
  - Double the number of CPUs/2 years
    - I/O unchanged

Year	CPU time	I/O time	Elapsed time	% I/O time
now	90s	10s	100s	10%
+2	45s	10s	55s	18%
+4	23s	10s	33s	31%
+6	11s	10s	21s	47%



I/O Controllers

- We could place every device on the interconnect
  - We would have to replace devices as we improve/change the interconnect

- We could decouple them from the interconnect
  - Via an “I/O controller”
  - Need to replace only the I/O controller when the CPU/Memory interface changes

Interconnecting Components

- Need interconnections between
  - CPU, memory, I/O controllers
- Bus: shared communication channel
  - Parallel set of wires for data and synchronization of data transfer

Buses

- A bus is a shared collections of wires with multiple senders/receivers
  - Has an associated protocol, obeyed by senders and receivers, for sending and receiving data
  - Simple broadcast mechanism: all devices can see all transactions
  - Pros: cost-effective
  - Cons: communication bottleneck

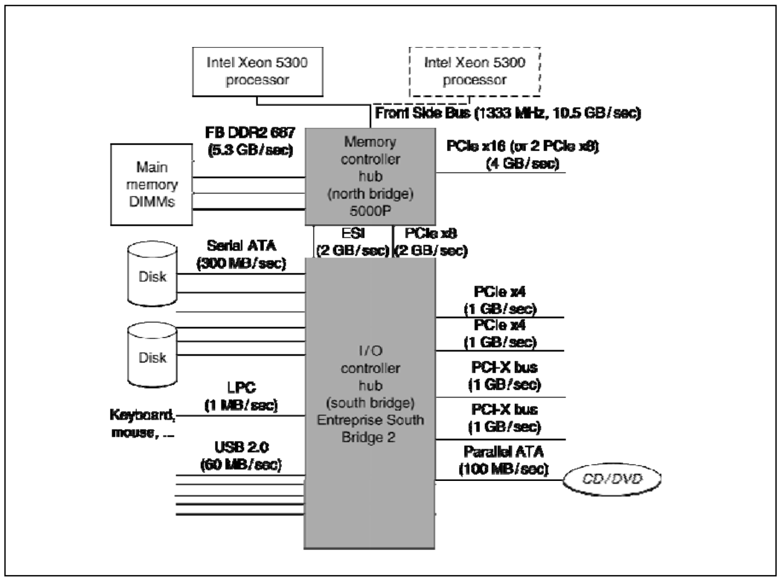
Bus Parameters

- Bus width
  - Number of wires, separate control/data?
- Data width
  - Number of bits per transfer
- Transfer size
  - Number of words per bus transaction
- Synchronous (clock), asynchronous (wider variety of devices)
  - Handshaking protocol
    - Sender/receiver proceed only if in agreement
- Buses: Firewire, SCSI, PCI Express, USB

Bus Types

- Processor-Memory buses
  - Short, high speed
  - Design is matched to memory organization
- I/O buses
  - Longer, allowing multiple connections
  - Specified by standards for interoperability
  - Connect to processor-memory bus through a bridge
- More recent alternative: high-speed serial connections with switches
  - Like networks

Typical x86 PC I/O System


Intel Xeon 5300 processor
Intel Xeon 5300 processor
Front Side Bus (1333 MHz, 10.5 GB/sec)
FB DDR2 667 (5.3 GB/sec)
Main memory DIMMs
Memory controller hub (north bridge) 5000P
PCIe x16 (or 2 PCIe x8) (4 GB/sec)
ESI (2 GB/sec)
PCIe x8 (2 GB/sec)
Serial ATA (300 MB/sec)
Disk
Disk
I/O controller hub (south bridge) Entreprise South Bridge 2
PCIe x4 (1 GB/sec)
PCIe x4 (1 GB/sec)
PCI-X bus (1 GB/sec)
PCI-X bus (1 GB/sec)
Parallel ATA (100 MB/sec)
CD/DVD
LPC (1 MB/sec)
Keyboard, mouse, ...
USB 2.0 (60 MB/sec)

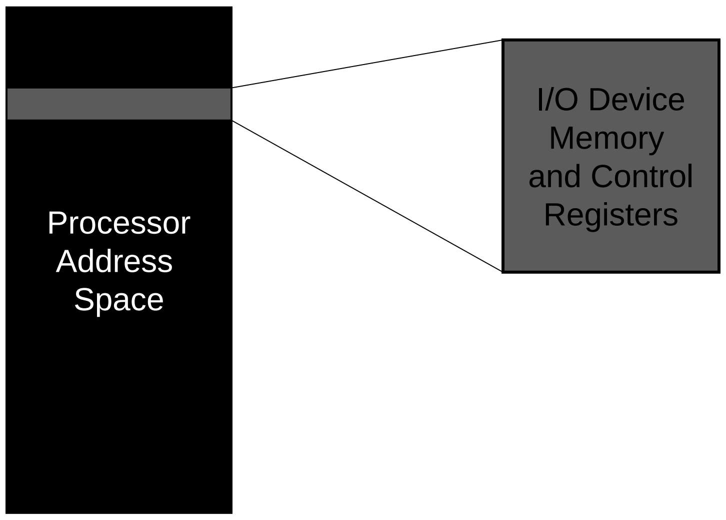
I/O Commands

- I/O devices are managed by I/O controller hardware
  - Transfers data to/from device
- Command registers
  - Cause device to do something
- Status registers
  - Indicate what the device is doing and occurrence of errors
- Data registers
  - Write: transfer data to a device
  - Read: transfer data from a device

Communication Interface

- Programmed I/O
  - CPU has dedicated, special instructions to access I/O registers
  - Instruction specifies device and operation
  - Protection: I/O instructions are privileged
- Memory-mapped I/O
  - Each device appears as if it is part of the memory address space
  - Reads/Writes to special addresses are converted into I/O operations
  - Device communication goes over the memory bus
  - Protection: device protected by the MMU

Memory-Mapped I/O



Memory-Mapped I/O

- Basic Idea – Make control registers and I/O device memory appear to be part of the system's main memory
  - Reads and writes to that region of the memory are translated by OS into device accesses
  - Easy to support variable numbers/types of devices
    - Managing new devices is now memory allocation
    - Example: accessing memory on a PCMCIA card
      - Once card memory mapped into address space, just hand out pointers like conventional memory

Processor Memory Maps

- Some OSes use fixed memory maps
- Others use variable mappings
  - Unix/Linux has virtual memory system, maps I/O devices onto memory only when requested. This allows more flexibility for a system to accommodate variable number I/O devices

Communication Method

- Polling

- Periodically check I/O status register
  - If device ready, do operation
  - If error, take action
- Common in small or low-performance real-time embedded systems
  - Predictable timing
  - Low hardware cost
- In other systems, wastes CPU time

Communication Method

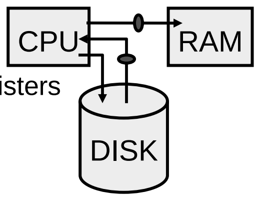
- Interrupts
  - Device sends interrupt
  - Cause information often identifies the interrupting device
  - OS responds by communicating with the device
  - Uses exception handling hardware
  - Interrupt priority levels
- Priority interrupts
  - Devices needing more urgent attention get higher priority

I/O Data Transfer

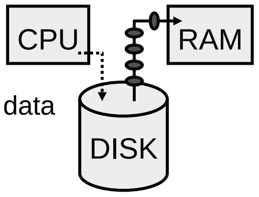
- Polling and interrupt-driven I/O
  - CPU transfers data between memory and I/O data registers
  - Time consuming for high-speed devices
- Direct memory access (DMA)
  - OS provides starting address in memory
  - I/O controller transfers to/from memory autonomously
  - Controller interrupts on completion or error

DMA: Direct Memory Access

- Non-DMA transfer: I/O device ←→ CPU ←→ RAM
  - **for (i = 1 .. n)**
    - CPU sends transfer request to device
    - I/O writes data to bus, CPU reads into registers
    - CPU writes data to registers to memory



- DMA transfer: I/O device ←→ RAM
  - CPU sets up DMA request on device
  - for (i = 1 .. n)
    - I/O device writes data to bus, RAM reads data



How to do DMA?

- DMA implemented with special controller
  - Transfers data between I/O device and memory
- Processor sets up DMA
  - Which device, operation, address, # of bytes
  - DMA
    - gets bus
    - Transfers data (maybe multiple requests)
  - DMA interrupts processor to signal end of I/O operation

DMA Issues (1): Addressing

- Problem: device operates on bus or physical addresses, programs operate with virtual addresses
- Solution: DMA uses physical addresses
  - Allocate contiguous physical pages for DMA
  - Or may need to break transfers into page-sized chunks, and chain multiple transfers

DMA Issues (1): Addressing

- Problem: device operates on bus or physical addresses, programs operate with virtual addresses
- Alternate solution: DMA uses virutal addresses
  - add translation hardware to dma controllers
  - a software-controlled, miniature TLB

DMA Issues (2): Virtual Mem

- Problem: DMA destination page may not be in memory (i.e. swapped to disk by virtual memory system)
- Solution:
  - *pin* the page before initiating DMA transfer
- Alternate solution:
  - DMA to a pinned kernel page, then memcpy elsewhere

DMA Issues (3): Legacy Cruft

- DMA controller (or bus) can't support full size addresses
  - many legacy DMA devices can use only 16-bit addresses
  - Solution: DMA to dedicated low-addressed physical pages, then memcopy elsewhere

DMA Issues (4): Caches

- What if L1 or L2 caches DMA-related data?
  - DMA write to device from RAM: device gets stale data if cache is dirty
  - DMA read from device to RAM: cache will have stale data
- Solution: (software enforced coherence)
  - flush entire cache (or part of cache) before DMA begins
  - Or, don't touch pages during DMA
  - Or, mark pages as *uncacheable* in VM page tables
    (and update the TLB entries for those pages)

DMA Issues (4): Caches

- What if L1 or L2 caches DMA-related data?
  - DMA write to device from RAM: device gets stale data if cache is dirty
  - DMA read from device to RAM: cache will have stale data
- Alternate solution: (hardware cache coherence, aka *snooping*)
  - cache controller listens on bus, and conspires with RAM
  - DMA write: cache services request if it has the data, otherwise RAM services
  - DMA read: cache invalidates data seen on the bus
    or, cache updates itself when it sees data on the bus

Summary

- Disks provide nonvolatile memory
- I/O performance measures
  - Throughput, response time
  - Dependability and cost very important
- Buses used to connect CPU, memory, I/O controllers
  - Polling, interrupts, DMA
- What we didn’t discuss: RAID
  - Redundancy for fault tolerance
  - Speed