Chapter 7: Input/Output

Who Cares & What to Consider

Latency vs. Throughput

Device Characteristics and Types

Some Important Devices

- Disks
- Graphics Displays
- Networks
- Buses

High-Performance Disk Architectures

Why I/O?

Amdahl's Law

- Only speed up CPU and I/O becomes bottleneck

e.g.,

- Suppose I/O takes 10% of time
- Speed up CPU 10 times
  - System only speeds up 5 times

Throughput vs. Latency

Throughput

- Bandwidth
- I/Os per second

Latency

- Response time

Who cares about latency?

- Just context switch

FALLACY

- Requires more memory
- Requires more processors
- Human productivity increases super-linearly as response time decreases.

I/O Performance

$$Time_{JOB} = Time_{CPU} + Time_{I/O} - Time_{OVERLAP}$$

e.g.,

$$10 = 10 + 4 - 4$$

Speed up CPU by 2X

What is $Time_{JOB}$?

A) $Time_{JOB}$ = 5 + 4 - 4 = 5 (Best)

B) $Time_{JOB}$ = 5 + 4 - 0 = 9 (Worst)

C) $Time_{JOB}$ = 5 + 4 - 2 = 7 (Average)

I/O Benchmarks

Supercomputers

- Data transfer rate
- Megabytes per second for large files

Transaction Processing

- I/O rate
- Random disk I/Os per second

Timesharing Filesystems

- Small files
- Sequential access
- Many creates/deletes

Device Characteristics

Behavior

- Input (Read-Once)
- Output (Write-Once)
- Storage (Read many times; usually write)

Partner

- Human
- Machine

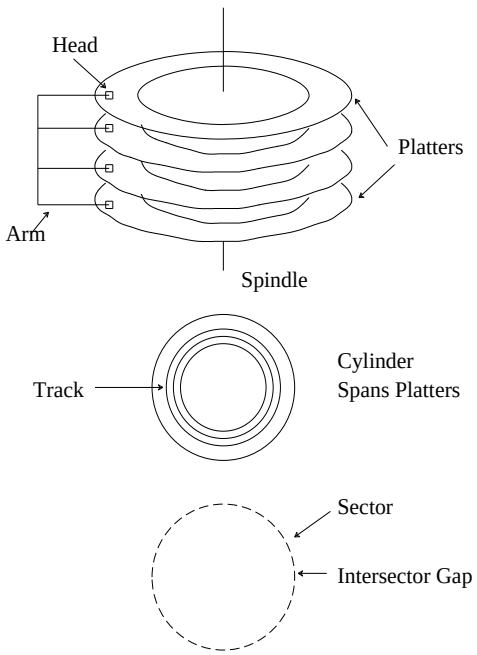
Data Rate

- Peak transfer rate

I/O Device Types

Device	*Behavior*	*Partner*	*Rate Kbytes/sec*
Keyboard	Input	Human	0.01
Mouse	Input	Human	0.02
Graphics Display	Output	Human	30,000
CPU → Frame Buffer	Output	Human	200
LAN	I/O	Machine	200
Optical Disk	Storage	Machine	500
Mag. Tape	Storage	Machine	2,000
Mag. Disk	Storage	Machine	2,000

Magnetic Disks



Disk Parameters

Spindles: 1 - 4 (most 1)

Platters per spindle: 1 - 20

RPM: 3000-6000 RPM (most 3600)

Platter Diameter: 2.5" - 14"

- Trend toward small disks
  - 1) higher RPM
  - 2) mass production

Tracks per surface: 500 - 2000

Sectors per track: 32 typical (software)

- Sector # — Gap — Data+ECC — Gap
- Fixed length sectors (except IBM)
- Typically fixed sectors per track
  - Recently constant bit density

Disk Operation

Seek: Move head to track

Avg. Seek Time = $\frac{1}{N}\sum_{i=1}^{N} SEEK_i$

N is # of tracks

$SEEK_i$ is time to seek to track i

Rotational Latency: Wait for sector

Avg. Rotation latency = $\frac{0.5}{3600RPM}$ = 8.3 ms

Transfer Time:

Typically 1-4 Megabytes per second

Overhead

- Controller Delay
- Queuing Delay

Alternatives to Disks

DRAMS

- SSD — Solid State Disk
  - Standard Disk Interface
  - DRAM and battery backup
- ES — Expanded Storage
  - Software controlled cache
  - Large (4K) blocks
- + No seek time
- + Fast transfer rate
- + Safer than main memory
- - Cost

Optical Disks

Read-Only

- CD ROM
- Cheap + Reliable
- Slow

Write-Once

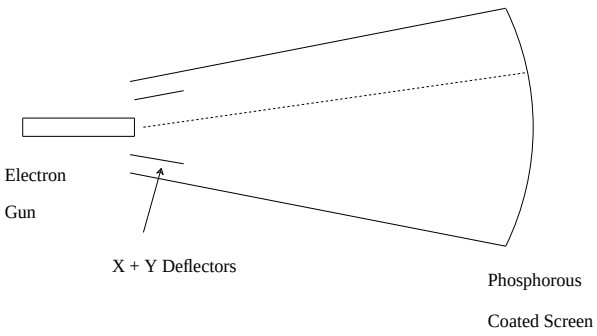
- Not as cheap
- Slow (especially write)

Write-Many

- Expensive
- Slow

Graphics Displays

Cathode Ray Tube



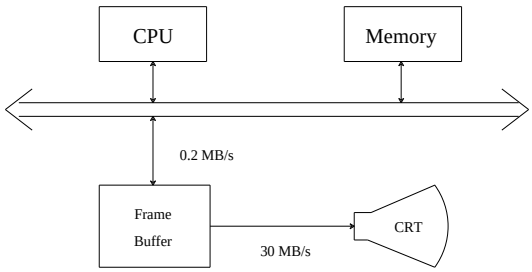
Phosphorous acts as capacitor

- Refresh 30-60 times per second

Screen consists of many ***scan lines***

- Each scan line consists of many ***pixels***

Graphics Displays — Frame Buffer



Frame buffer stores bit map

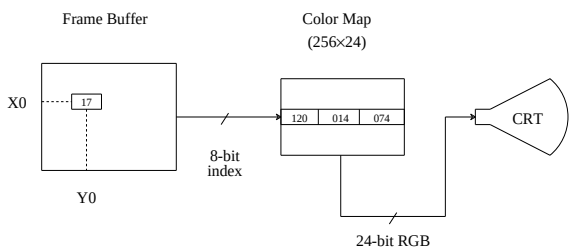
- One entry per pixel
  - Black and White: 1 bit per pixel
  - Gray-scale: 4-8 bits per pixel
  - Color (RGB): 8 bits per color
- Typical Size: 1560 $\times$ 1280 pixels
  - Black and White: 250 KB
  - Color (RGB): 5.7MB

Reducing Cost of Frame Buffer

Key Idea: Only a small number of colors are used in any one image

Color Map:

- Frame buffer stores color map index
- Color map translates index to full 24-bit color



1560 $\times$ 1280 RGB with 256-entry color map

- factor of 3 reduction

Frame Buffer Operations

Logically Output Only

- ***BUT***, read as well

BIT BLTS: Bit Block Transfers

- Read-Modify-Write Operations
- e.g., Read-XOR-Write
- Used for cursors, etc.

Research Question

- Operating System only?
  Or Direct User Access?
  - If user access, protection?

Frame Buffer Implementation

1560 × 1280 RGB Display

Bandwidth required = 1560×1280×24×30 = 171 MB/sec

How can we implement this?

Video DRAMS

Dual-ported DRAM

Regular Random Access Port

Serial video port

Use 24 in parallel for RGB

What about bandwidth

Interleave video DRAMS to increase effective bandwidth

Other Issues in Displays

Double Buffering

Duplicate frame buffer to prevent displaying incomplete updates

May be necessary for animation

Z-Buffer

For displaying 3D images

Assign Z-dimension for each pixel

Store Z-dimension in frame buffer

BIT BLTS compare Z-dimension

Note: Z-dimension not sent to CRT

Networks

Terminal Network

Machine-Terminal
Star (point-to-point)
0.3-19.2 Kbits per sec
RS232 protocol, etc.

Local-Area Networks

Machine-Machine
Bus, Ring, Star
0.1-100 MBits per sec
< 10 kilometers
Ethernet, etc.

Long-Haul Network

Machine-Machine
Tree (DAG)
50-2000 Kbits per sec
> 10 kilometers
ARPANET

Buses

Option	High Performance	Low Cost
Address & data lines separate?	yes	no
Data lines	wider	narrower
Transfer size	multiple words	single word
Bus masters	multiple	one
Split transaction?	yes	no
Clocking	synchronous	asynchronous

(H&P Figure 9.24)

CPU-Memory Buses

Want speed!

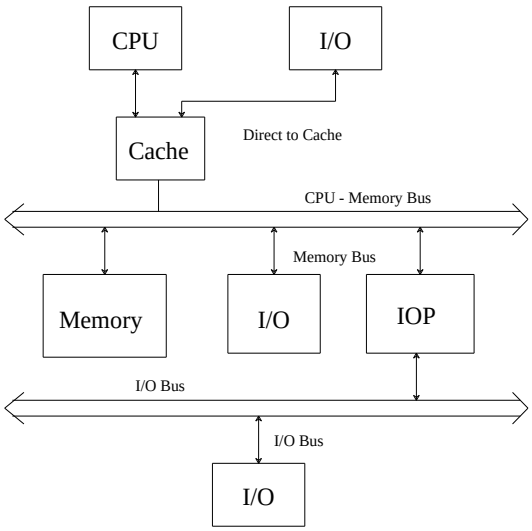
Usually custom designs

I/O Buses

Want compatibility!

Usually standard designs (Unibus, SCSI)

CPU Interface



CPU Interface

Physical Connection

- Direct to Cache
  - + No coherence problems
  - - ‘‘Pollutes’’ the cache
  - - CPU and I/O arbitrate for cache
- CPU—Memory Bus
  - + Direct Memory Access
  - - May not be standard bus
- I/O Bus
  - + Industry standard bus
  - - Slower than CPU-Memory bus
  - - Indirection through IOP

CPU Interface, contd.

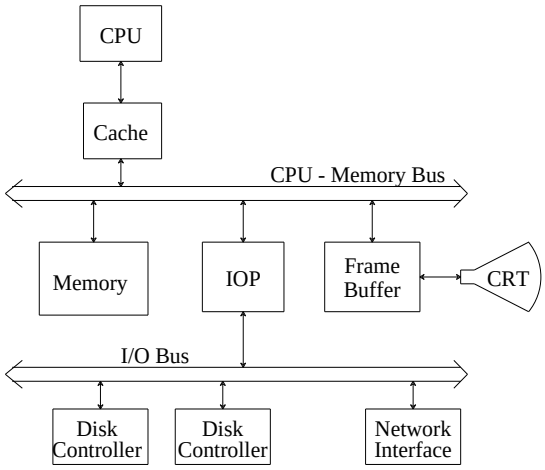
I/O Control

- Memory-Mapped
  - Reading/writing to assigned addresses causes operations to occur
- I/O Instructions
  - Special I/O instructions initiate I/O operations

I/O Completion

- Polling
  - Wait for status bit to change
  - Periodic checking
- Interrupt
  - I/O completion interrupts CPU

Example



Bus Arbitration

Centralized ‘‘Star’’ connection

- High cost
- High Performance

Daisy Chain

- Cheap
- Low Performance

Distributed Arbitration

- Medium Price
- Better Performance

Arbitration for next bus mastership is typically overlapped with current transfer.

Distributed Arbitration

Set of wired-OR priority lines

Set of wired-OR timing and control lines

Each requesting device indicates its priority

Device removes its less-significant bits if higher priority device present

Eventually only highest priority device remains.

Special care to ensure ‘‘fairness’’

Bus Switching Method

Pended or circuit-switched Buses

- Bus is held until request is complete
  - + simple protocol
  - - latency of device contributes to bus utilization

Split transaction, or pipelined, or packet-switched

- Bus is released after request is initiated; others can use bus until reply comes back
  - - more complex bus control
  - + better utilization of bus

Higher-Performance Disk Architectures

Extensions to Conventional Disks

Disk Arrays

Redundant Arrays of Inexpensive Disks (RAIDs)

Extensions to Conventional Disks

Fixed Head Disk

Head per track; head does not seek

+ seek time eliminated

• rotational latency unchanged

- low track density

- not economical

Parallel Transfer Disk

Read from multiple surfaces at a time

- difficulty in locking onto different tracks on multiple surfaces

• lower cost alternatives possible (disk arrays)

Extensions to Conventional Disks

Increasing Disk Density

Requires sophisticated lock-on control mechanisms

- increases cost

Solid-State Disks

20% of data receives 80% of accesses

see earlier discussion

Extensions to Conventional Disks

Disk Caches

RAM buffers to buffer data between device and host

+ ‘‘fast writes’’: buffer acts as write buffer

+ better utilization of host-to-device path

- increase in request latency if buffer not effective

Disk Scheduling

Schedule a set of simultaneous I/O requests to reduce total latency, e.g., schedule request with shortest seek time

- works best for unlikely cases (long queues of pending requests)

Disk Arrays

Collection of individual disks

Each disk has own arm/head

Degree of Interleaving

Granularity at which data is interleaved

bit, byte, block, track, or cylinder

Arm independence

actuators of individual disks seek as a unit or independently

Rotation independence

same sector of each disk under head at same time (synchronous) or not (asynchronous)

Synchronous rotation and independent arms is not very useful

Synchronous Rotation/Unit Arms

Logical disk block is interleaved among multiple physical disks (a la memory interleaving)

- Logical seek = N physical seeks
- Transfer time reduced by factor of N

Problems

- Single failed disk = array failure
  - Likely single point of failure for entire array (e.g., sync. clock)
- Handling bad sectors

Asynchronous Rotation/Unit Arms

Similar to synchronous rotation/unit arms but rotation of individual disks is not synchronized

Plus: easy to construct since no need to synchronize multiple spindles

Minus: Rotational latency tends to worst case rather than average case

Asynchronous Rotation/Independent Arms/Block Interleaved

Interleaving at the block level (declustering). Logical block in one device. Logically adjacent blocks placed on separate devices.

- Filesystem help required to place adjacent logical blocks on separate devices.

Plus: concurrent operation of multiblock I/O as well as independent I/Os.

Minus: additional complexity of multiple request queues plus fault-tolerance issues.

Redundant Arrays

Asynchronous Rotation/Independent Arms/Redundant, Non-Interleaved

- Also called disk shadowing
- Disk image maintained on another physical disk
- Write bandwidth sacrificed
  - Each logical write is multiple physical writes
- Too much redundancy

Asynchronous Rotation/Independent Arms/Redundant,Block-Interleaved

- Also called interleaved-declustering
- Also suffers from degraded write performance

Redundant Arrays of Inexpensive Disks (RAIDs)

Arrays of small, cheap, disks to provide high performance and high reliability.

G = number of ‘‘data’’ disks in group

C = number of ‘‘check’’ disks in group

Level 1: Mirrored Disks (***G*** = 1; ***C*** = 1)

Overhead is too high

Level 2: Bit Interleaved Array (e.g., ***D*** = 10; ***C*** = 4)

Like ECC codes for DRAMs

Read all bits across group

Merge update bits with those not being updated; recompute parities

Rewrite full group, including checks

Redundant Arrays of Inexpensive Disks (RAIDs)

Level 3: Hard Failure Detection and Parity (e.g., ***D*** = 4; ***C*** = 1)

Key: failed disk is easily identified, no need for special code to identify failed disk

Level 4: Intragroup Parallelism

Level 3 plus ability to do more than one small I/O at a time.

Redundant Arrays of Inexpensive Disks (RAIDs)

Level 5: Rotated Parity to Parallelize Writes

Parity spread out across disks in a group: different updates of parities go to different disks.

Level 6: Two-Dimensional Array

Array is a two-dimensional array, with row and column parities