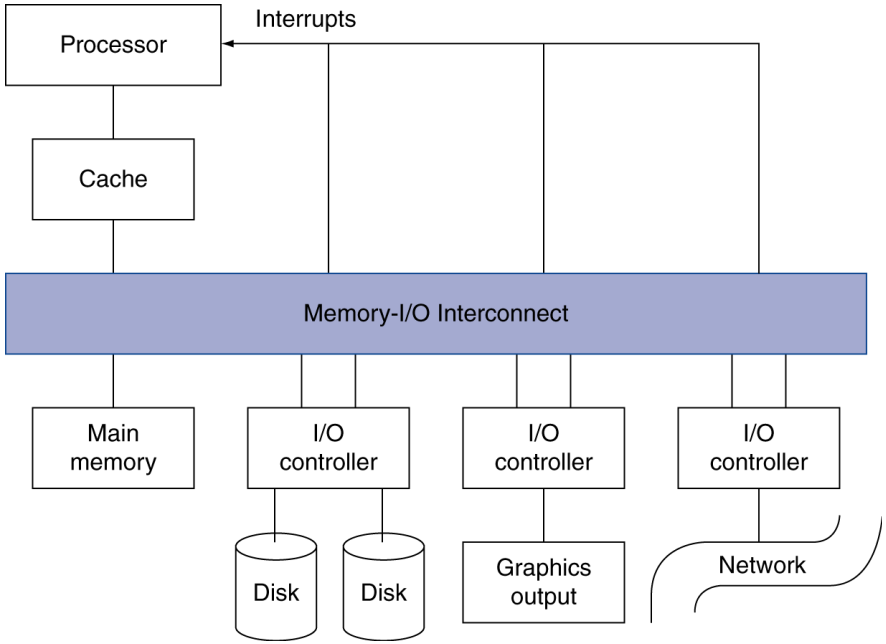
Computer Input & Output

Lecture for CPSC 5155

Edward Bosworth, Ph.D.
Computer Science Department
Columbus State University

Introduction

- I/O devices can be characterized by
  - Behaviour: input, output, storage
  - Partner: human or machine
  - Data rate: bytes/sec, transfers/sec
- I/O bus connections



Four Strategies for Managing I/O

- **Program Controlled I/O**
  The CPU manages every aspect of I/O processing. I/O occurs only when the program calls for it. If the I/O device is not ready to perform its function, the CPU waits for it to be ready; this is **"busy waiting"**.
- **Interrupt Driven I/O**
  The I/O device raises a signal called an **"interrupt"** when it is ready to perform the I/O. The I/O is managed by the CPU, but proceeds only when the device is ready for it.
- **Direct Memory Access**
  This variant elaborates on the two above. The I/O device interrupts and is sent a "word count" and starting address by the CPU. The individual transfers are managed by the device controller, not the CPU.
- **I/O Channel**
  This assigns I/O to a separate processor, which uses one of the above three strategies.

I/O Strategies: A Silly Example

- I am giving a party to which a number of people are invited. I know exactly how many people will attend.
- I know that the guests will not arrive before 6:00 PM.
- All guests will enter through my front door. In addition to the regular door (which can be locked), it has a screen door and a doorbell.
- I have ordered pizzas and beer, each to be delivered. All deliveries at the back door.
- I must divide my time between baking cookies for the party and going to the door to let the visitors in.
- I am a careless cook and often burn the cookies.

The Silly Example: Program Controlled I/O

- The guests are due, starting at 6:00 PM.
- Here I go to the door at 6:00 PM and wait.
- As each one arrives, I open the door and admit the guest.
- I do not leave the door until the last guest has arrived; nothing gets done in the kitchen.

The Silly Example: Interrupt-Driven I/O

- I continue working in the kitchen until the doorbell rings.
- When the doorbell rings, I put down my work, go to the door, and admit the guest. However, note the following:

1. I do not “drop” the work, but bring it to a quick and orderly conclusion. If I am removing cookies from the oven, I place them in a safe place to cool before answering the door.
2. If I am fighting a grease fire, I ignore the doorbell and first put out the fire. Only when it is safe do I attend to the door.
3. With a guest at the front door and the beer truck at the back door, I have a difficult choice, but I must attend to each.

The Silly Example: Direct Memory Access

- I continue work in the kitchen until the first guest arrives and rings the doorbell.
- I go to the door, unlock it, admit the guest .
- I leave the main door open. Each guest now enters without ringing the doorbell.
- The last guest is asked to notify me, so that I can return to the front door and close it again.
- In the Interrupt Driven analog, I had to go to the door once for each guest.
- In the DMA analog, I had to go to the door only twice.

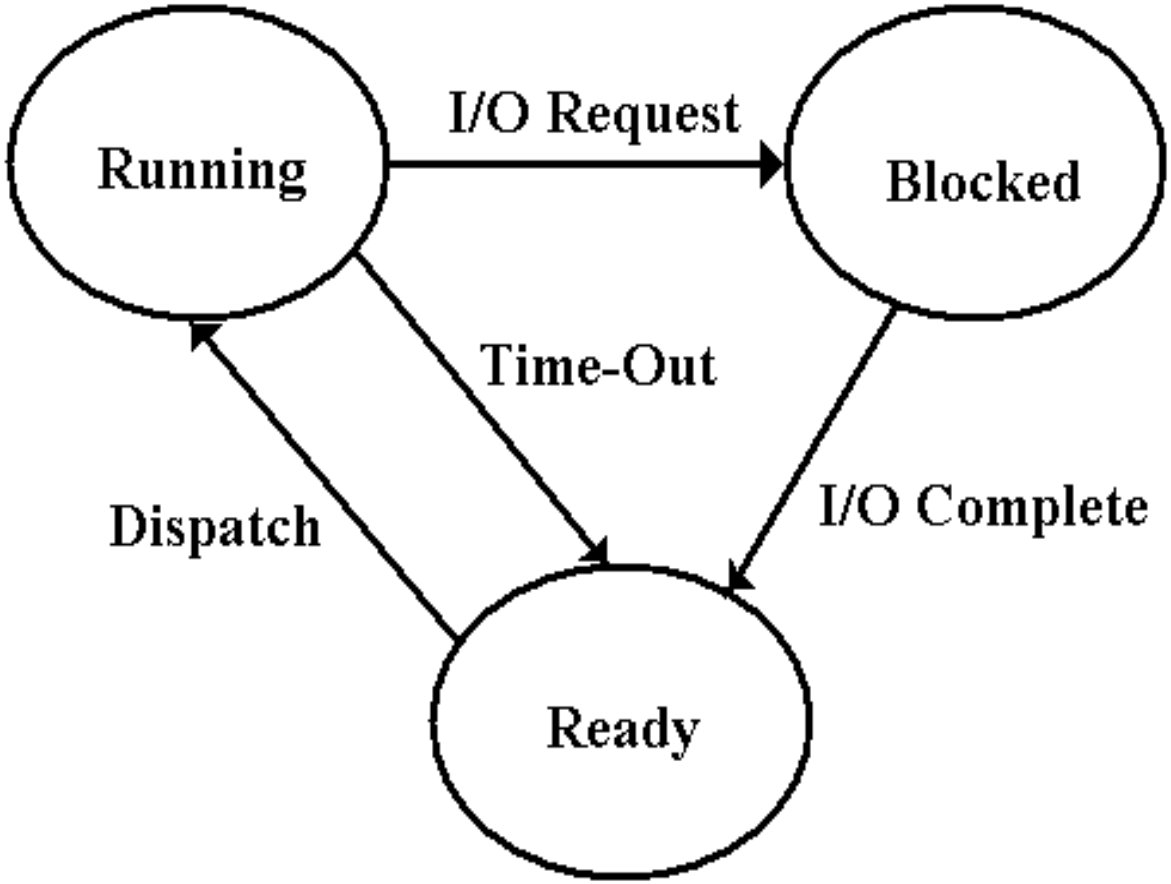
The Silly Example: I/O Channel

- Here, I hire a butler and tell him to manage the door any way he wants.
- He just has to get the guests into the party and keep them happy.

Device Management

- The primary issue in choosing a strategy for I/O is the relative speed of I/O and the CPU.
- Program controlled I/O may be appropriate when the I/O device is purely electronic and able to transfer data almost instantly.
- The context for the other strategies is that the CPU can execute other processes while waiting on the given I/O to complete.

The Classic Process Diagram



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

Interrupts

- When a device is ready or error occurs
  - Controller interrupts CPU
- Interrupt is like an exception
  - But not synchronized to instruction execution
  - Can invoke handler between instructions
  - Cause information often identifies the interrupting device
- Priority interrupts
  - Devices needing more urgent attention get higher priority
  - Can interrupt handler for a lower priority interrupt

The PDP-11 Priority Structure

- The I/O system of the old PDP-11 was based on eight priority levels: 0 through 7.
- User programs execute at priority level 0.
- The device handler programs execute at the device priority level.
- The disk is assigned priority level 7.
  The keyboard is assigned priority level 4.
- An interrupt is processed only if its priority is higher than that of the executing program.

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/Cache Interaction

- If DMA writes to a memory block that is cached
  - Cached copy becomes stale
- If write-back cache has dirty block, and DMA reads memory block
  - Reads stale data
- Need to ensure cache coherence
  - Flush blocks from cache if they will be used for DMA
  - Or use non-cacheable memory locations for I/O

DMA/VM Interaction

- OS uses virtual addresses for memory
  - DMA blocks may not be contiguous in physical memory
- Should DMA use virtual addresses?
  - Would require controller to do translation
- If DMA uses physical addresses
  - May need to break transfers into page-sized chunks
  - Or chain multiple transfers
  - Or allocate contiguous physical pages for DMA

DMA and Memory Protection

- The Operating System uses Virtual Memory as a tool to protect physical memory from access that is possibly harmful.
- If DMA bypasses the VM system, it bypasses this level of protection.
- Device driver software is usually written by the device vendor; it is suspect.
- The hardware can have a "**No DMA Table**".

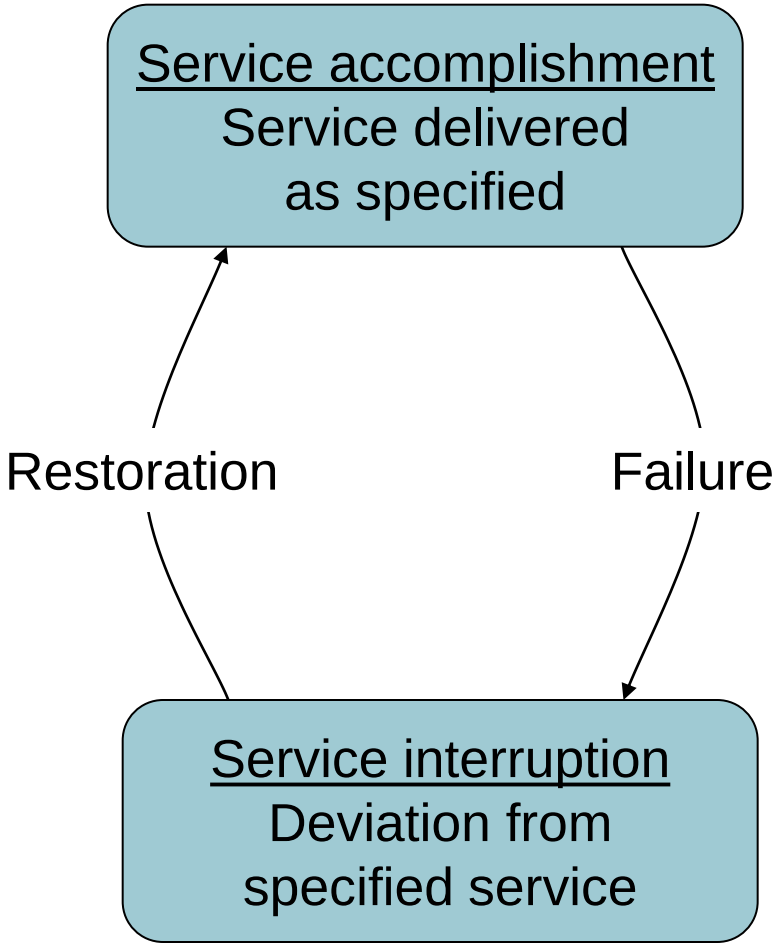
I/O Channels and Processors

- Manufacturers have elected to use attached I/O processors for a number of reasons.
- Balance of function: IBM 1401 and IBM 7094.
- Removal of I/O functionality from a CPU dedicated to numerical computations, as in the CDC 6600, CDC 7600, or Cray-1.
- Enterprise servers, such as the IBM S/360 are transaction processors with heavy I/O.

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

Dependability



- Fault: failure of a component
  - May or may not lead to system failure

Dependability Measures

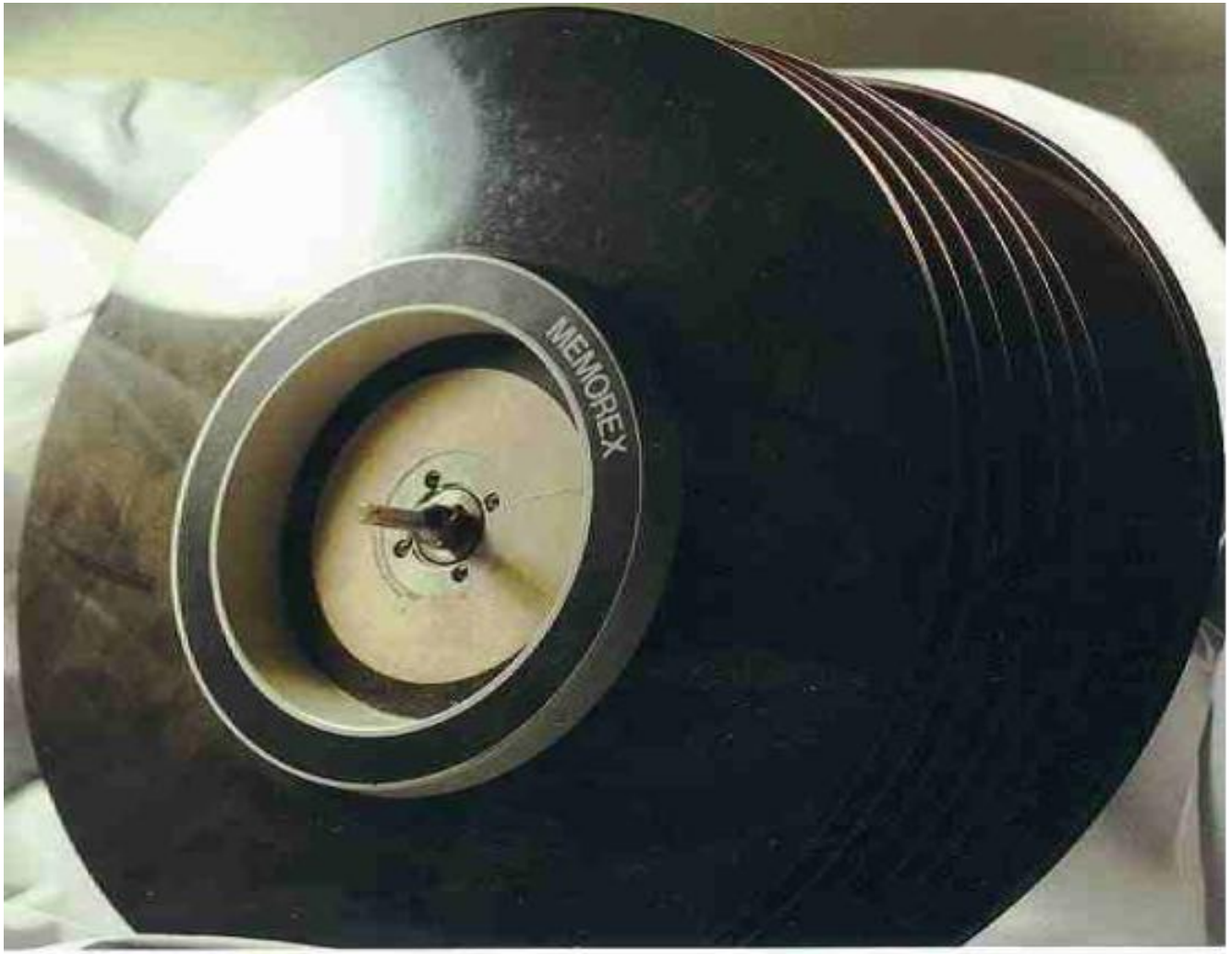
- Reliability: mean time to failure (MTTF)
- Service interruption: mean time to repair (MTTR)
- Mean time between failures
  - MTBF = MTTF + MTTR
- Availability = MTTF / (MTTF + MTTR)
- Improving Availability
  - Increase MTTF: fault avoidance, fault tolerance, fault forecasting
  - Reduce MTTR: improved tools and processes for diagnosis and repair

The DASD

- The term “DASD” is an IBM term that stands for “**D**irect **A**ccess **S**torage **D**evice”.
- One hears the term “DASD” only in IBM enterprise server shops.
- There used to be two DASDs: magnetic disks and magnetic drums.
- Magnetic drums are obsolete.
- A flash drive may be called a DASD.

The IBM 2311

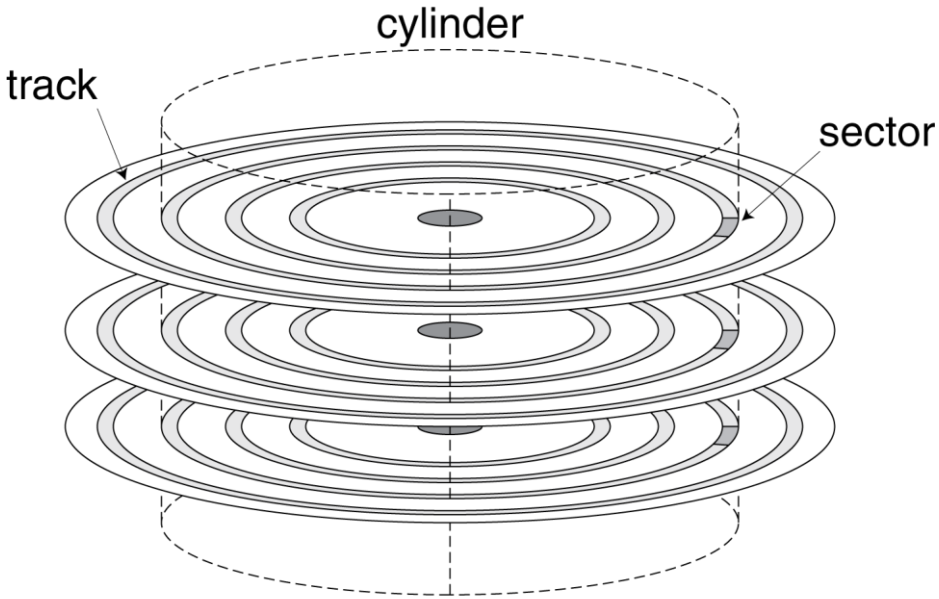
Disk Platters



The Sealed Disk Drive

Disk Storage

- Nonvolatile, rotating magnetic storage

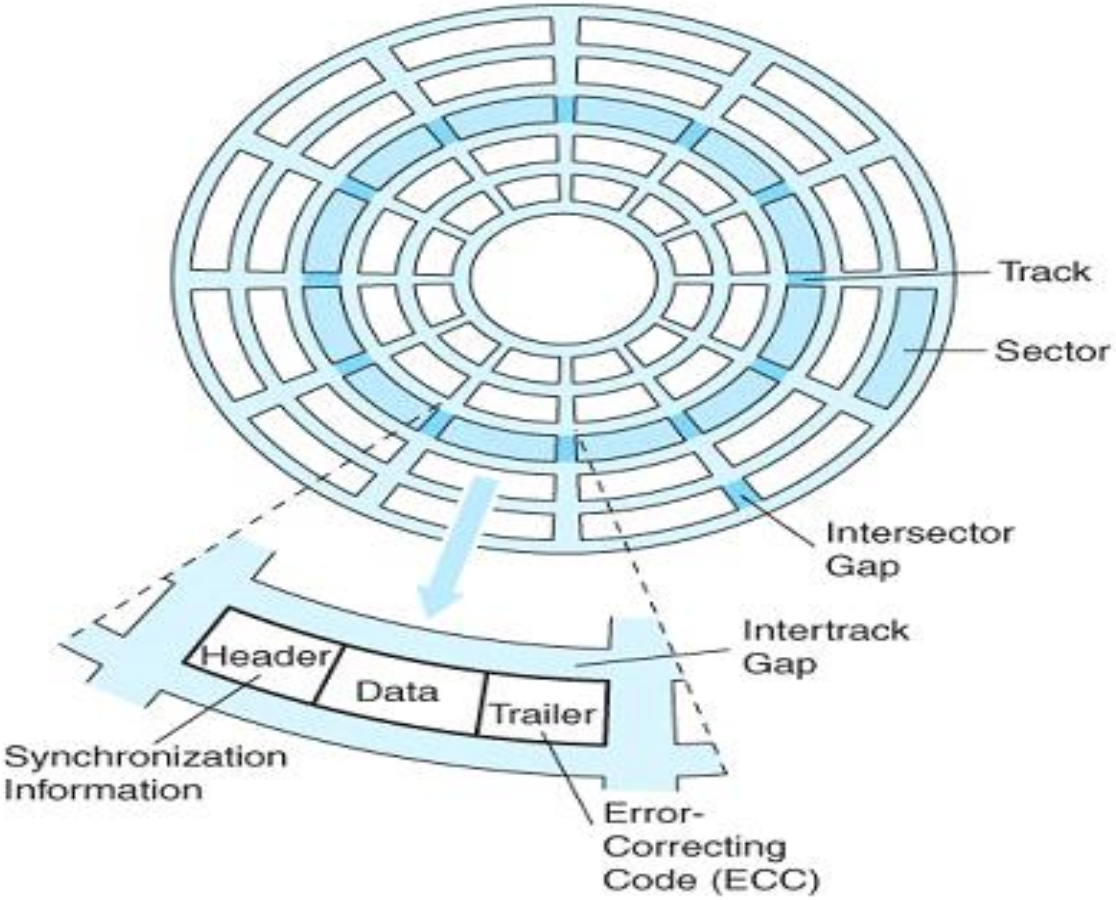


The Read/Write Heads

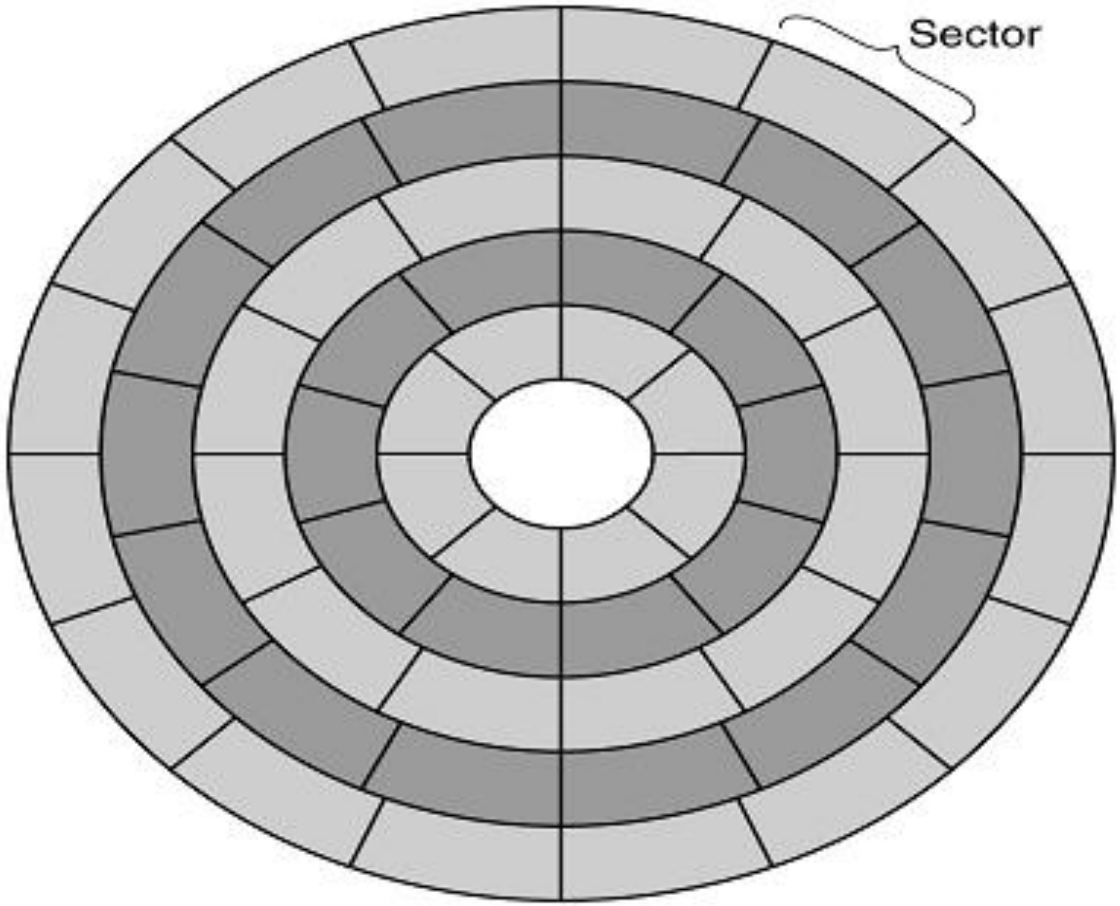
Disk Sectors and Access

- Each sector records
  - Sector ID
  - Data (512 bytes, 4096 bytes proposed)
  - Error correcting code (ECC)
    - Used to hide defects and recording errors
  - Synchronization fields and gaps
- Access to a sector involves
  - Queuing delay if other accesses are pending
  - Seek: move the heads
  - Rotational latency
  - Data transfer
  - Controller overhead

The Disk Platter



Modern Disk Organization



Seek Time and Rotational Latency

- The **seek time** is defined as the time to move the read/write heads to another track.
- There are two values: track-to-track (time to move to an adjacent track) and average.
- The **rotational latency** is the average time to rotate the sector under the read/write heads.
- At 12,000 rpm (200/second), the disk revolves every 5 milliseconds; latency = 2.5 msec.

The Data Transfer Rate

- The maximum data transfer rate for a disk is set by its rotational speed.
- The contents of one track being read for one rotation sets the maximum data transfer rate.
- At 12,000 RPM, there is one revolution every 1/200 of a second.
- If the track has 1.2 MB of data, the maximum transfer rate is 1.2 MB / (1/200) = 240 MB/s.

Disk Access Example

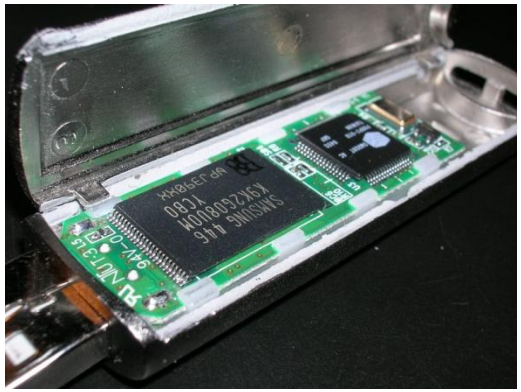
- Given
  - 512B sector, 15,000rpm, 4ms average seek time, 100MB/s transfer rate, 0.2ms controller overhead, idle disk
- Average read time
  - 4ms seek time
    + ½ / (15,000/60) = 2ms rotational latency
    + 512 / 100MB/s = 0.005ms transfer time
    + 0.2ms controller delay
    = 6.2ms
- If actual average seek time is 1ms
  - Average read time = 3.2ms

Disk Performance Issues

- Manufacturers quote average seek time
  - Based on all possible seeks
  - Locality and OS scheduling lead to smaller actual average seek times
- Smart disk controller allocate physical sectors on disk
  - Present logical sector interface to host
  - SCSI, ATA, SATA
- Disk drives include caches
  - Prefetch sectors in anticipation of access
  - Avoid seek and rotational delay

Flash Storage

- Nonvolatile semiconductor storage
  - 100× – 1000× faster than disk
  - Smaller, lower power, more robust
  - 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
  - Wear leveling: remap data to less used blocks