Rule 1: “Data processing requires data storage”. – I just made that up.

Data were originally stored on paper media; first as written documents but fairly soon (Hollerith, late 19th century) the storage medium was machine-readable.

In the 1950’s, New York Life Insurance Company was devoting an entire floor of its main building to the storage of punched cards. Something had to change.

IBM quickly came out with two magnetic media for storing data:

the magnetic tape

the DASD (disk)

The acronym “DASD” stands for Direct Access Storage Device.

Until recently, the standard disk drive was the only commercially viable example. (Magnetic drum memories became obsolete in the late 1950’s.)

We now have another very popular example, these USB “flash drives”. While different from standard disk drives, these are managed as if they were disk drives and are considered disk drives.
Removable Disk Packs and Sealed Drives

Early disk systems mimicked the magnetic tape strategy.
The disks were organized into packs. These were mounted into the disk drive, and removed when another pack was to be used. Here is an IBM–2311 system.
Disk Platters

Here is a picture showing the disk platters on a commercial hard drive, in which the disk pack was removable from the disk drive.
The Sealed Disk Pack

The IBM 3340, code named “Winchester” (because it called for two 30 MB removable disk packs) was the first to be based on sealed disk packs.

The sealed disk packs included the head and arm assembly. Packaging these with the disk pack and not the drive lead to significant cost savings and increased reliability.
The next evolutionary step was inclusion of the disk drive motor and all associated electronics in the disk pack. Now the entire unit was removable.
## Historical Cost Data for Disk Drives

<table>
<thead>
<tr>
<th>YEAR</th>
<th>MANUFACTURER</th>
<th>COST/GB</th>
</tr>
</thead>
<tbody>
<tr>
<td>1956</td>
<td>IBM</td>
<td>$10,00,000</td>
</tr>
<tr>
<td>1980</td>
<td>North Star</td>
<td>$193,000</td>
</tr>
<tr>
<td>1981</td>
<td>Morrow Designs</td>
<td>$138,000</td>
</tr>
<tr>
<td>1983</td>
<td>Davong</td>
<td>$119,000</td>
</tr>
<tr>
<td>1984</td>
<td>Pegasus (Great Lakes)</td>
<td>$80,000</td>
</tr>
<tr>
<td>1985</td>
<td>First Class Peripherals</td>
<td>$71,000</td>
</tr>
<tr>
<td>1987</td>
<td>Iomega</td>
<td>$45,000</td>
</tr>
<tr>
<td>1988</td>
<td>IBM</td>
<td>$16,000</td>
</tr>
<tr>
<td>1990</td>
<td>First Class Peripherals</td>
<td>$12,000</td>
</tr>
<tr>
<td>1991</td>
<td>WD</td>
<td>$9,000</td>
</tr>
<tr>
<td>1992</td>
<td>Iomega</td>
<td>$7,000</td>
</tr>
<tr>
<td>1994</td>
<td>Iomega</td>
<td>$2000</td>
</tr>
<tr>
<td>1995</td>
<td>Seagate</td>
<td>$850</td>
</tr>
<tr>
<td>1996</td>
<td>Maxtor</td>
<td>$259</td>
</tr>
<tr>
<td>1997</td>
<td>Maxtor</td>
<td>$93</td>
</tr>
<tr>
<td>1998</td>
<td>Quantum</td>
<td>$43</td>
</tr>
<tr>
<td>1999</td>
<td>Fujitsu IDE</td>
<td>$16</td>
</tr>
<tr>
<td>2000</td>
<td>Maxtor 7200 rpm UDMA/66</td>
<td>$9.58</td>
</tr>
<tr>
<td>2001</td>
<td>Maxtor 5400 rpm IDE</td>
<td>$4.57</td>
</tr>
<tr>
<td>2002</td>
<td>Western Digital 7200 rpm</td>
<td>$2.68</td>
</tr>
<tr>
<td>2003</td>
<td>Maxtor 7200 rpm IDE</td>
<td>$1.39</td>
</tr>
<tr>
<td>2004</td>
<td>Western Digital Caviar SE</td>
<td>$1.15</td>
</tr>
<tr>
<td>2011</td>
<td>WD Caviar Green (3 TB for $140)</td>
<td>$0.05</td>
</tr>
</tbody>
</table>
The typical large–capacity (and physically small) disk drive has a number of glass platters with magnetic coating. These spin at a high rate (7,200 rpm or 120 / second)

This drawing shows a disk with three platters and six surfaces. In general, a disk drive with N platters will have 2N surfaces, the top and bottom of each platter.

On early disk drives, before the introduction of sealed drives, the top and bottom surfaces would not be used because they would become dirty.
Picture of a Disk Read/Write Arm

Here is a picture of a commercial disk, showing the read–write arm.

This drive seems to have two platters, with four recording surfaces.
More on Disk Drive Structure

Each surface is divided into a number of concentric tracks.
Each track has a number of sectors.

A sector usually contains 512 bytes of data, along with a header and trailer part.
Seek Time and Rotational Latency

In order to read from a disk track, the read/write heads must be moved to the track. This is a mechanical action, as the read/write heads are physical devices.

There are two seek times typically quoted for a disk.

**Track–to–track:** the time to move the heads to the next track over

**Average:** the average time to move the heads to any track.

The *rotational delay* is due to the fact that the disk is spinning at a fixed high speed. It takes a certain time for a specific sector to rotate under the read/write heads.

Suppose a disk rotating at 12,000 RPM. That is 200 revolutions per second. Each sector moves under the read/write heads 200 times a second, once every 0.005 second or every 5 milliseconds.

The *rotational latency*, or average rotational delay, is one half of the time for a complete revolution of the disk. Here it would be 2.50 milliseconds.
The Idea of a Cylinder

**Fixed head** disks have one head per track. The last time I heard of such a device was 1977, when working with a 1 MB fixed head disk on a PDP–11/45.

I claim that fixed head disks are obsolete. Revisit the picture of a typical disk.

![Diagram of a disk with arms, platters, and read/write heads]

Question: How many tracks can be read before the read/write heads must be moved?

Answer: One track per surface can be read without moving the heads. Here it is 6.

**Definition:** A cylinder is that set of tracks that can be read without moving the disk read write heads. A disk has as many cylinders as a surface has tracks. A cylinder has as many tracks as the disk has surfaces.
The File Allocation Table

Physically the disk contains a large number of sectors, each of which contains either data or programs. The term “cluster” will be defined in the next slide.

Logically the disk contains a large number of files, also program and data.

The disk will have one or more index structures that associate files with their sectors.

The two important structures are

- the disk directory associates a file with its first sector
- the File Allocation Table maintains the “linked list” of sectors for that file.

An example from the textbook shows the structure of the FAT.

Here the disk directory indicates that the first sector for a file is at address 121. The FAT entry at 121 indicates that the next sector is at address 124. The FAT entry at 126 indicates that the next sector is at address 122. The FAT entry at 122 indicates that sector 122 is the last for this file. Sector 125 is bad.
FAT–16 and Its Consequences

The FAT–16 system was implemented by Microsoft for early versions of MS–DOS. This system used a 16–bit index into the FAT.

As there is one FAT entry per sector, this makes this limits the sector count to $2^{16}$.

The maximum disk size is thus $2^{16} \times 512 = 2^{16} \times 2^9 = 2^{25} = 2^5 \times 2^{20} = 32$ MB.

In 1987, my brand–new PC/XT had a 20MB disk! FAT–16 worked very well.

What about a 40 MB disk? How about a 256 MB disk?

Few people in the late 1980’s contemplated disk drives with capacities of over 100 GB, but it was obvious that unmodified FAT–16 would not do the job.

We consider two short–term remedies to this problem, one transient and one with longer term consequences.
The first solution was to partition a larger physical disk drive into two logical disk drives.

A 40 MB disk drive would support two logical disks, each with its own directory structure and File Allocation Table.

- Drive C with capacity of 32 MB
- Drive D with capacity of 8 MB.

As a short-term fix, this worked well. However, it just raised the limit to 64 MB. This solution was obsolete by about 1992.
FAT–16 and Its Consequences
Part 3: Disk Clusters

The main problem with the FAT–16 system arose from the fact that each sector was individually addressable. With $2^{16}$ addresses available, we have a maximum of $2^{16}$ sectors or 32 MB.

The second solution was to remove the restriction that each sector be addressable.

Sectors were grouped into clusters, and only clusters could be addressed.

The number of sectors a cluster contained was constrained to be a power of 2; so we had 2, 4, 8, 16, etc. sectors per cluster. The effect on disk size is easy to see.

<table>
<thead>
<tr>
<th>Sectors in Cluster</th>
<th>1</th>
<th>2</th>
<th>4</th>
<th>8</th>
<th>16</th>
<th>32</th>
<th>64</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bytes in Cluster</td>
<td>512</td>
<td>1024</td>
<td>2048</td>
<td>4096</td>
<td>8192</td>
<td>16384</td>
<td>32768</td>
</tr>
<tr>
<td>Disk Size</td>
<td>32 MB</td>
<td>64 MB</td>
<td>128 MB</td>
<td>256 MB</td>
<td>512 MB</td>
<td>1 GB</td>
<td>2 GB</td>
</tr>
</tbody>
</table>

In the early 1990’s, it seemed that this solution would work for a while.

Nobody back then envisioned multi–gigabyte disk drives on personal computers.

There is a problem associated with large clusters; it is called “internal fragmentation”.

FAT–16 and Its Consequences
Part 4: Internal Fragmentation

This problem arises from the fact that files must occupy an integer number of clusters.

Consider a data file having exactly 6,000 bytes of data, with several cluster sizes.

<table>
<thead>
<tr>
<th>Sectors in Cluster</th>
<th>1</th>
<th>2</th>
<th>4</th>
<th>8</th>
<th>16</th>
<th>32</th>
<th>64</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bytes in Cluster</td>
<td>512</td>
<td>1024</td>
<td>2048</td>
<td>4096</td>
<td>8192</td>
<td>16384</td>
<td>32768</td>
</tr>
<tr>
<td>Clusters needed</td>
<td>12</td>
<td>6</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>File size on disk</td>
<td>6144</td>
<td>6144</td>
<td>6144</td>
<td>8192</td>
<td>8192</td>
<td>16384</td>
<td>32768</td>
</tr>
<tr>
<td>Disk efficiency</td>
<td>97.7%</td>
<td>97.7%</td>
<td>97.7%</td>
<td>73.2%</td>
<td>73.2%</td>
<td>36.6%</td>
<td>18.3%</td>
</tr>
</tbody>
</table>

There is also the consideration of the time required to read a cluster.

Consider a disk for which the maximum data rate is 8 MB per second. Here, we are saying that 8MB = $2^{23}$ bytes. More on this later.

<table>
<thead>
<tr>
<th>Sectors in Cluster</th>
<th>1</th>
<th>2</th>
<th>4</th>
<th>8</th>
<th>16</th>
<th>32</th>
<th>64</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bytes in Cluster</td>
<td>$2^9$</td>
<td>$2^{10}$</td>
<td>$2^{11}$</td>
<td>$2^{12}$</td>
<td>$2^{13}$</td>
<td>$2^{14}$</td>
<td>$2^{15}$</td>
</tr>
<tr>
<td>Transfer time</td>
<td>$2^{-14}$</td>
<td>$2^{-13}$</td>
<td>$2^{-12}$</td>
<td>$2^{-11}$</td>
<td>$2^{-10}$</td>
<td>$2^{-9}$</td>
<td>$2^{-8}$</td>
</tr>
<tr>
<td>Time in msec.</td>
<td>0.06</td>
<td>0.122</td>
<td>0.244</td>
<td>0.488</td>
<td>0.977</td>
<td>1.95</td>
<td>3.91</td>
</tr>
</tbody>
</table>
Marketing Hype: Sizing Large Disk Drives

All large disk drives are sized in gigabytes.

Traditionally $1\text{GB} = 2^{30}\text{ bytes} = 1,073,741,824\text{ bytes}$.

Recent commercial practice uses $1\text{GB} = 10^9\text{ bytes} = 1,000,000,000\text{ bytes}$.

Let’s look at some comparative sizes.

<table>
<thead>
<tr>
<th>Size in GB with $1\text{GB} = 10^9\text{ bytes}$</th>
<th>Size in GB with $1\text{GB} = 2^{30}\text{ bytes}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>9.314</td>
</tr>
<tr>
<td>40</td>
<td>37.25</td>
</tr>
<tr>
<td>80</td>
<td>74.51</td>
</tr>
<tr>
<td>200</td>
<td>186.3</td>
</tr>
</tbody>
</table>

Some disk utilities will report free disk space using both units. This is the reason for the discrepancies.
Computing Disk Capacities

The capacity of a disk drive is expressed in a number of equivalent ways.

Disk Capacity  = (number of surfaces) • (bytes per surface)
= (number of surfaces) • (tracks per surface) • (bytes per track)
= (number of surfaces) • (tracks per surface) • (sectors per track) • 512

Data from an earlier disk drive (now rather small).

8 surfaces
3196 tracks per surface
132 sectors per track
512 bytes per sector

Surface capacity = 3196 • 132 • 512 = 421872 • 512 = 210936 • 1024 bytes
= 210936 KB ≈ 206.0 MB

Disk capacity = 8 • 210936 KB = 1,687,488 KB ≈ 1.61 GB
Computing Disk Maximum Transfer Rate

Disk rotation rates are given in RPM (Revolutions per Minute). Common values are 3,600 RPM, 7,200 RPM, and higher.

3,600 RPM is 60 revolutions per second. 7,200 RPM is 120 per second.

Disks can transfer at maximum rate while reading from a single cylinder. Moving to another cylinder takes time, called the track-to-track seek time.

Consider our sample disk. Suppose it rotates at 7,200 RPM. One revolution every (1/120) second.

One track contains 132•512 bytes = 66•1024 bytes = 66KB.

This track can be read in (1/120) of a second.

The maximum data rate is 66 KB in (1/120) of a second.

120 • 66 KB in one second.

7,920 KB per second = 7.73 MB per second.
Sustaining the Maximum Data Transfer Rate

Recall the definition of a cylinder as a set of tracks, one per surface.

The number of tracks per cylinder is always exactly the same as the number of surfaces in the disk drive.

Our sample drive has 8 surfaces; each cylinder has 8 tracks.

Each track can be read in one revolution of the disk drive.

The data rate can be sustained for as long as it takes to read all tracks from the cylinder.

In our sample drive, rotating at 7200 RPM or 120 per second:

Each track can be read in 1/120 second.

The cylinder, containing 8 tracks, is read in 8/120 second or 1/15 second.

The maximum data transfer rate can be sustained for 1/15 second.
Modern Disk Organization

Older disk drives divided the surface into a number of equally sized tracks.

This facilitated design of the disk controller, but made poor use of the disk surface, as the inner tracks (with maximum linear density) determined the number of sectors.

Modern disk drives divide the disk surface into a number of zones. Each zone has the same number of sectors per track.

This gives only a modest increase in complexity of the controller.
Security Issues: Erasing and Reformatting

Remember the disk directory and the FAT.

What happens when a file is erased from the disk? Are the file data removed?

Answer: No, the file data are not removed. What actually happens is:

1. The file name is removed from the disk directory.
2. The FAT is modified to place all the sectors (clusters) for that file into a special file, called the “Free List”.
3. Sectors do not have their data erased or changed until they are allocated to another file and that a program writes data to that file.

For this reason, many companies sell utilities to “Wipe the File” or “Shred the File”.

How about reformatting the disk? Certainly, that removes data.

Answer: No, the file data are not removed.

The directory structure and FAT are reinitialized.

The free list is built in an efficient form.

The sectors containing data are not overwritten.
This lecture is based on the Peripheral Component Interconnect Express, which is a standard for computer expansion cards.

More specifically, this is a standard for the communication link by which a PCIe device communicates with the CPU.

According to Wikipedia, PCIe 3.0 (August 2007) is the latest standard.

The PCIe standard is an outgrowth of the original PCI standard, but is not compatible with PCI at the hardware level.

The PCIe standard is based on a new protocol for electrical signaling. This protocol is built on the concept of a lane, which we must define.

Here are some capacity quotes from Wikipedia

<table>
<thead>
<tr>
<th>Version</th>
<th>Per Lane</th>
<th>16–Lane Slot</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version 1</td>
<td>250 MB/s</td>
<td>4 GB/s</td>
</tr>
<tr>
<td>Version 2</td>
<td>500 MB/s</td>
<td>8 GB/s</td>
</tr>
<tr>
<td>Version 3</td>
<td>1 GB/s</td>
<td>16 GB/s</td>
</tr>
</tbody>
</table>
What is a Lane?

A lane is a pair of point–to–point serial links.

It is a full–duplex link, able to communicate in two directions simultaneously. Each of the serial links in the pair handles one of the two directions.

By definition, a serial link transmits one bit at a time.

By extension, a lane may transmit two bits at any one time, one bit in each direction.

One may view a parallel link, transmitting multiple bits in one direction at any given time, as a collection of serial links.

The only difference is that a parallel link must provide for synchronization of the bits sent by the individual links.
Data Transmission Codes

The PCIe standard is byte oriented, in that it should be viewed logically as a full–duplex byte stream. What is actually transmitted?

The association of bits (transmitted or received) with bytes is handled at the Data Link layer. Suppose a byte is to be transmitted serially.

The conversion from byte data to bit–oriented data for serial transmission is done by a shift register. The register takes in eight bits at a time and shifts out one bit at a time.

The bits, as shifted out, are still represented in standard logic levels.

The serial transmit unit takes the standard logic levels as input, and converts them to voltage levels appropriate for serial transmission.
Three Possible Transmission Codes

The serial transmit unit sends data by asserting a voltage on the serial link.

On simple method would be as follows.
   To transmit a logic 1, assert +5 volts on the transmission line.
   To transmit a logic 0, assert 0 volts on the transmission line.

This simple method has so many difficulties in practice that it cannot be used. Two of the most obvious are transmission of power and lack of data framing.

Standard methods for link management use codes that avoid these problems. Two of the more common methods used are NRZ and NRZI.

**Non–Return–to–Zero** coding transmits by asserting the following voltages:
   For a logic 1, it asserts a positive voltage (3.0 – 5.0 volts) on the link.
   For a logic 0, it asserts a negative voltage (–3.0 to –5.0 volts).

**Non–Return–to–Zero–Invert** is a modification of NRZ, using the same voltage levels.
The Problem of Noise

One problem with these serial links is that they function as antennas. They will pick up any stray electromagnetic radiation if in the radio range. In other words, the signal received at the destination might not be what was actually transmitted. It might be the original signal, corrupted by noise.

The solution to the problem of noise is based on the observation that two links placed in close proximity will receive noise signals that are almost identical.

To make use of this observation, we use **differential transmitters** to send the signals and **differential receivers** to reconstruct the signals.
Differential Transmitters and Receivers

In differential transmission, rather than asserting a voltage on a single output line, the transmitter asserts two voltages: $+V/2$ and $-V/2$.

A $+6$ volt signal would be asserted as two: $+3$ volts and $-3$ volts.

A $-8$ volt signal would be asserted as two: $-4$ volts and $+4$ volts.

Here is a standard diagram of a differential transmitter.

\[ \text{V} \quad \text{V/2} \quad \text{V/2} \]

The standard differential receiver is an analog subtractor.

\[ \text{A} \quad \text{A-B} \]

\[ \text{B} \]

For a 6 volt transmitted signal, we have $A = 3$, $B = -3$; $A - B = 3 - (-3) = 6$. 
Noise in a Differential Link

We now assume that the lines used to transmit the differential signals are physically close together, so that each line is subject to the same noise signal.

Here the received signal is the difference of the two voltages input to the differential receiver.

The value received is $V/2 - (-V/2) = V$, the desired value.
Ground Offsets in Standard Links

All voltages are measured relative to a standard value, called “ground”.

Here is the complete version of the simple circuit that we want to implement.

![Circuit Diagram]

Basically, there is an assumed second connection between the two devices. This second connection fixes the zero level for the voltage.

There is no necessity for the two devices to have the same ground. Suppose that the ground for the receiver is offset from the ground of the transmitter.

![Circuit Diagram with Offset]

The signal sent out as +V(t) will be received as V(t) – V_o.
Ground Offsets in Differential Links

Here again, the subtractor in the differential receiver handles this problem.

The signal originates as a given voltage, which can be positive, negative, or 0. The signal is transmitted as the pair \((+V/2, -V/2)\).

Due to the ground offset for the receiver, the signal is taken in as \((+V/2 - V_o, -V/2 - V_o)\). The signal is interpreted as:

\[
(+V/2 - V_o) - (-V/2 - V_o) = +V/2 - V_o + V/2 + V_o = V.
\]

The differential link will correct for both ground offset and line noise at the same time.
References