

Operating system



Part IX: IO Devices

- Use HDD as instance to understand the mapping between file and blocks
- Trend: Attempt to manage devices uniformly

- **MAPPING 1** – map files into main memory
 - Basic: fundamental ideas and old ways
 - Advanced: so-called virtual memory
- **MAPPING 2** – map files into persistence storage medias (HDD space as instance)
 - Basic: fundamental understanding about HDD space
 - Advanced: File system
 - Other: Disk scheduling algorithms, RAID, Spooling, etc.

Goals

- Know the basic concepts related to IO
 - Types of devices
 - General framework to connect devices with computers
 - How to control the devices
- Know the techniques related to Hard Disk
 - So as to provide the basis for file system

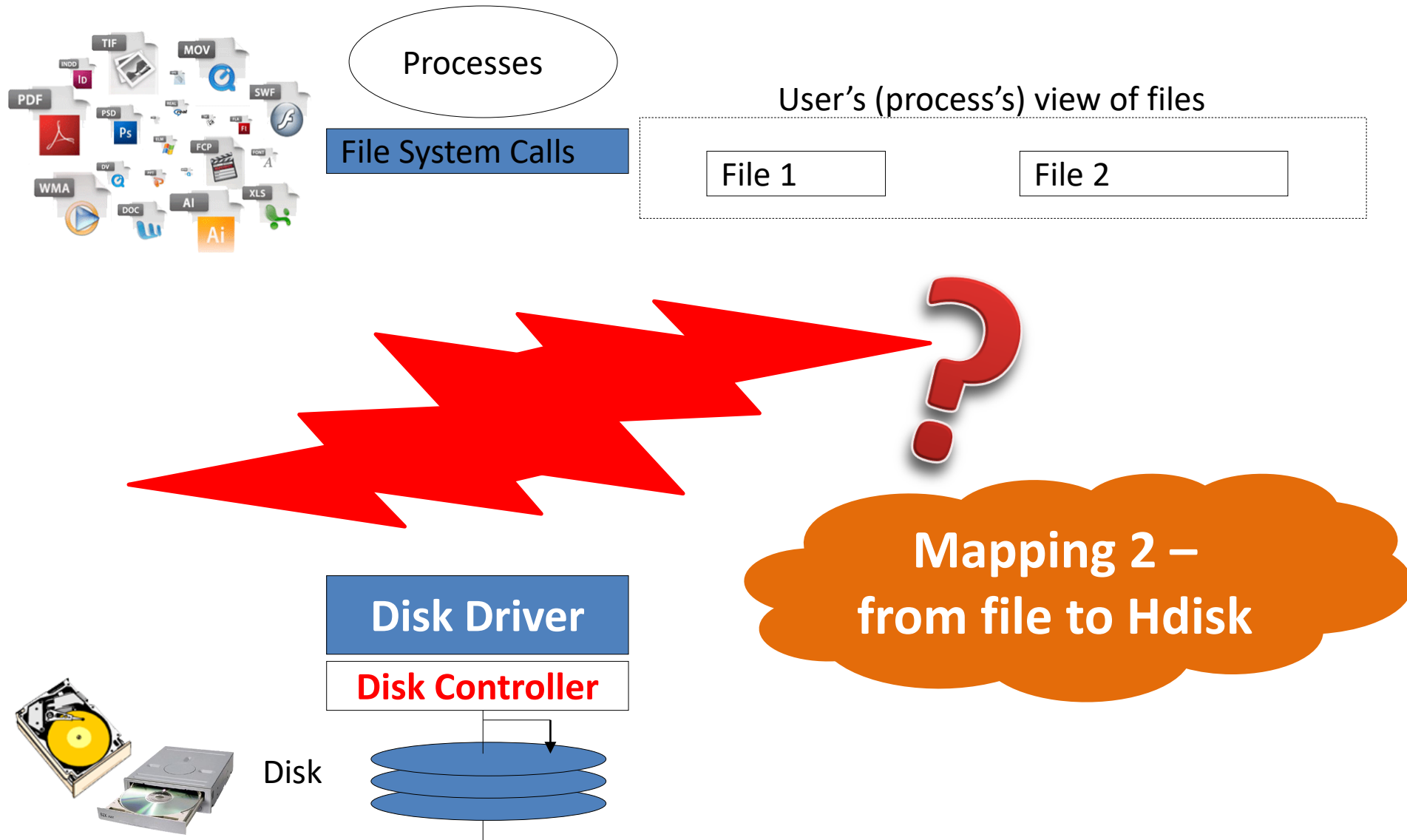
File concept

[We all know “**file**”, but what is it essentially?]

- Anything you are interested in and want to **store permanently** could be a “file”!
 - Long-term Information Storage
 - Must store large amounts of data
 - Information stored must survive the termination of the process using it
 - Multiple processes must be able to access the information concurrently
- Each file is a named collection of data stored in a permanent device



We need store file permanently in secondary storage media, like Hard Drive Disk(HDD)



I reorganize the IO + File system

- Because Mapping 2 and Mapping 1 (with HDisk as instance) could be understood following similar idea
- They both
 - **From logic program/file to linear**
 - Allocation and Address Translation
 - Indexing data structure (Tree)
 - Some data structures are also needed and kept in MM to locate those programs

I suggest that you'd better remember this and follow this to understand the rest parts!

- General structure to connect devices –
 - Abstraction/Interface in OS to communicate with the diverse devices
 - IO devices - **Categories** of devices
 - **IO operations**
 - **Connect** all together & structure of IO module in OS
 - **Communication** types between CPU and IO devices
- Taking (Magnetic) Disk for instance
 - So-called “linear address sector space”
 - Organize sectors into partitions, and so-called “linear addressed block space”
 - Optical disk is similar

- We have many different devices



Part X IO System (Basic)

Three Device Types

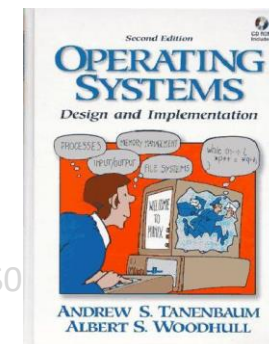
- Most operating system have 2 device types:

1. **Block devices:** [块设备]

- Block devices can only be written to and read from in multiples of the block size, typically 512 or 1024 bytes
 - Used for mass-storage (E.g. hard disks, tapes and CDROM)

2. **Character devices** [字符设备]

- Character devices are read and written directly without buffering
 - Used for serial-line types of devices (e.g. USB port)
- **Network Card**
 - Network devices are accessed via the socket interface
 - Used for network interfaces (E.g. Ethernet card)

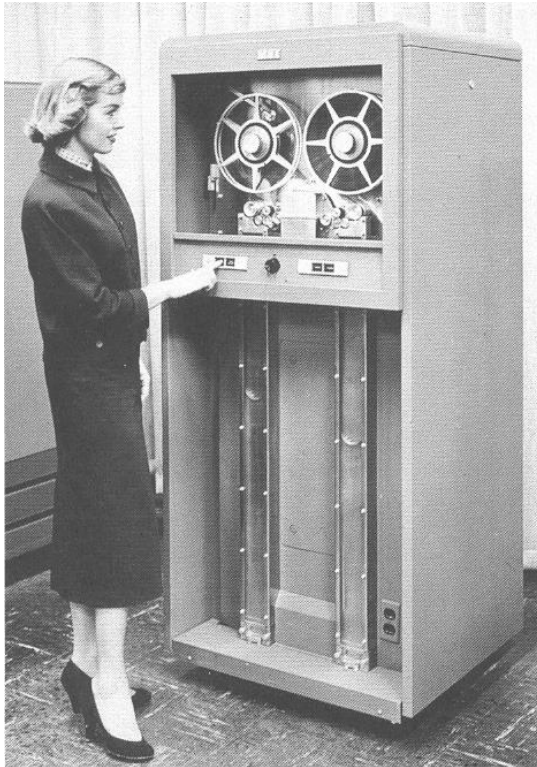


Sequential Access Devices

Sequential Access = In order to access specific information, the device must sequentially pass through all preceding information

- 9 Track Tape (Reel to Reel [逐卷地])
- Cartridge Tapes

PPTs\Part XII\Part XII magnetic media].ppt



Direct Access Devices

Direct Access = The specific information is accessed directly

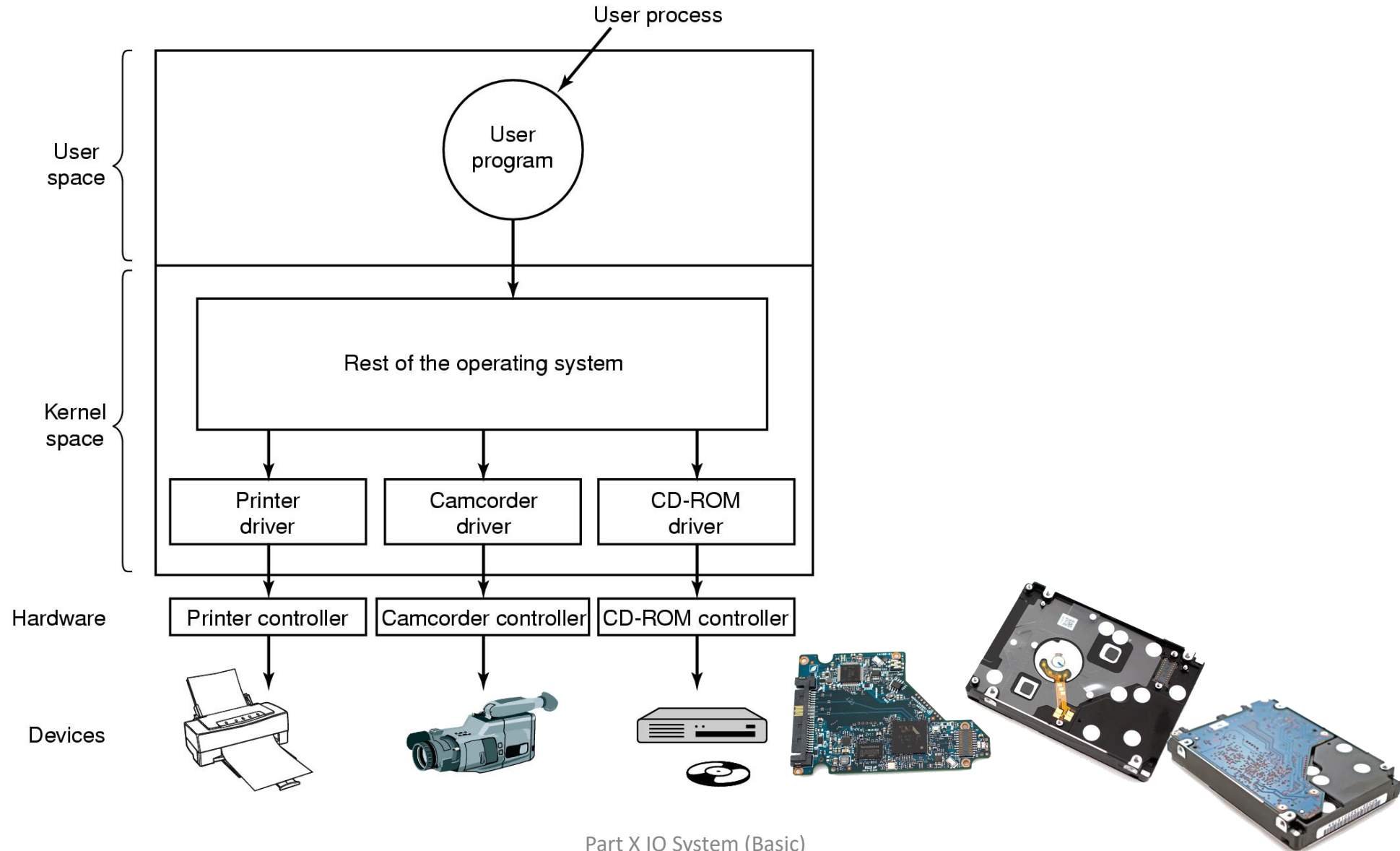
Examples

- floppy disk drives
- hard disk drives
- cartridge disk drives
- CD ROM and DVD drives

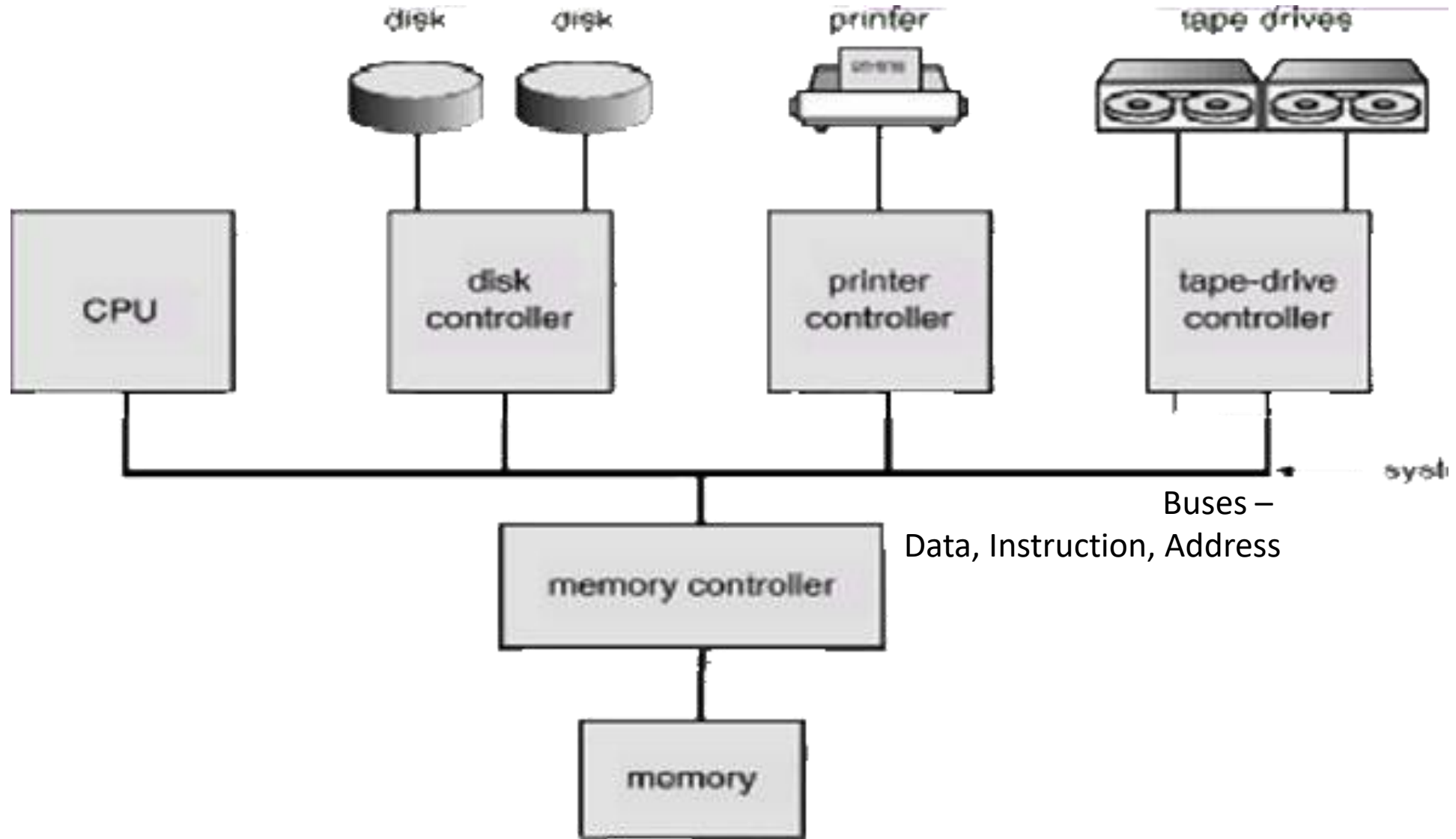
PPTs\Part XII\Part XII magnetic media].ppt



They share similar connection architecture



MM is also connected with controller

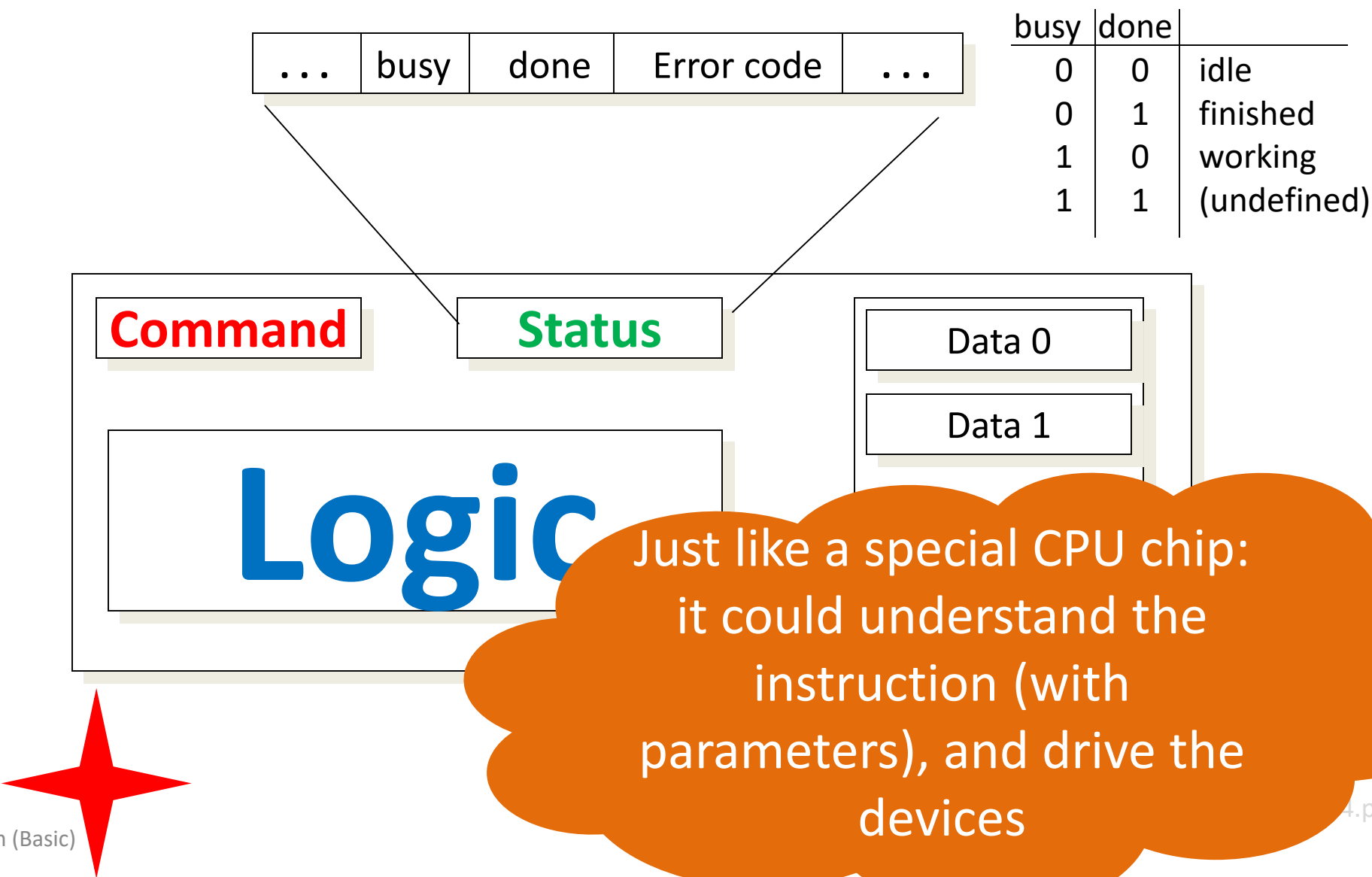


Device controller [控制器]

- Device controller: hardware that connects the device to the computer.
 - continuously monitors and controls the operation of the device.
 - provides an interface to the computer.
- Controller's tasks
 - Control the physical operation of the device
 - Convert serial bit stream to block of bytes
 - Perform error correction as necessary
- Since several devices need to be connected to a computer, they are connected through the **bus**.

PPTs from others\cms.dt.uh.edu\chap04.ppt

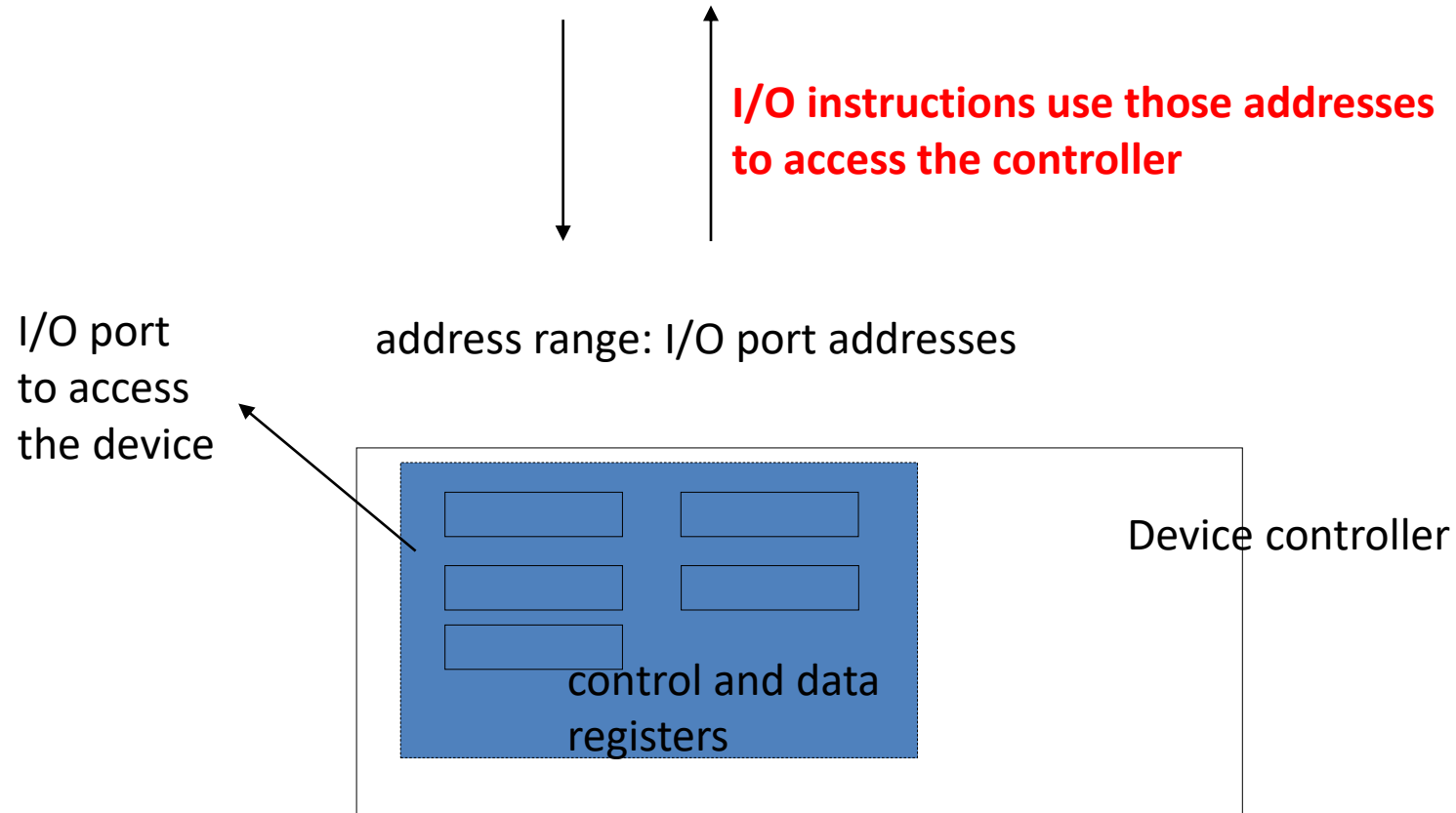
Device Controller Interface



- The device communicates with the computer via a communication point called a **port**. [端口号]
- Exchange data with CPU via registers
 - By writing into these registers
 - OS can command the device to deliver or accept data, to switch the device on or off
 - By reading from the registers
 - OS can learn the status of the device

I/O port concept

PPTs.2012\PPTs from others\www.cs.bilkent.edu.tr~korpe_courses_cs342spring2010\lecture13_io.ppt



Device I/O Port Locations on PCs (partial)

I/O address range (hexadecimal)	device
000–00F	DMA controller
020–021	interrupt controller
040–043	timer
200–20F	game controller
2F8–2FF	serial port (secondary)
320–32F	hard-disk controller
378–37F	parallel port
3D0–3DF	graphics controller
3F0–3F7	diskette-drive controller
3F8–3FF	serial port (primary)

We also need device Drivers [驱动程序]

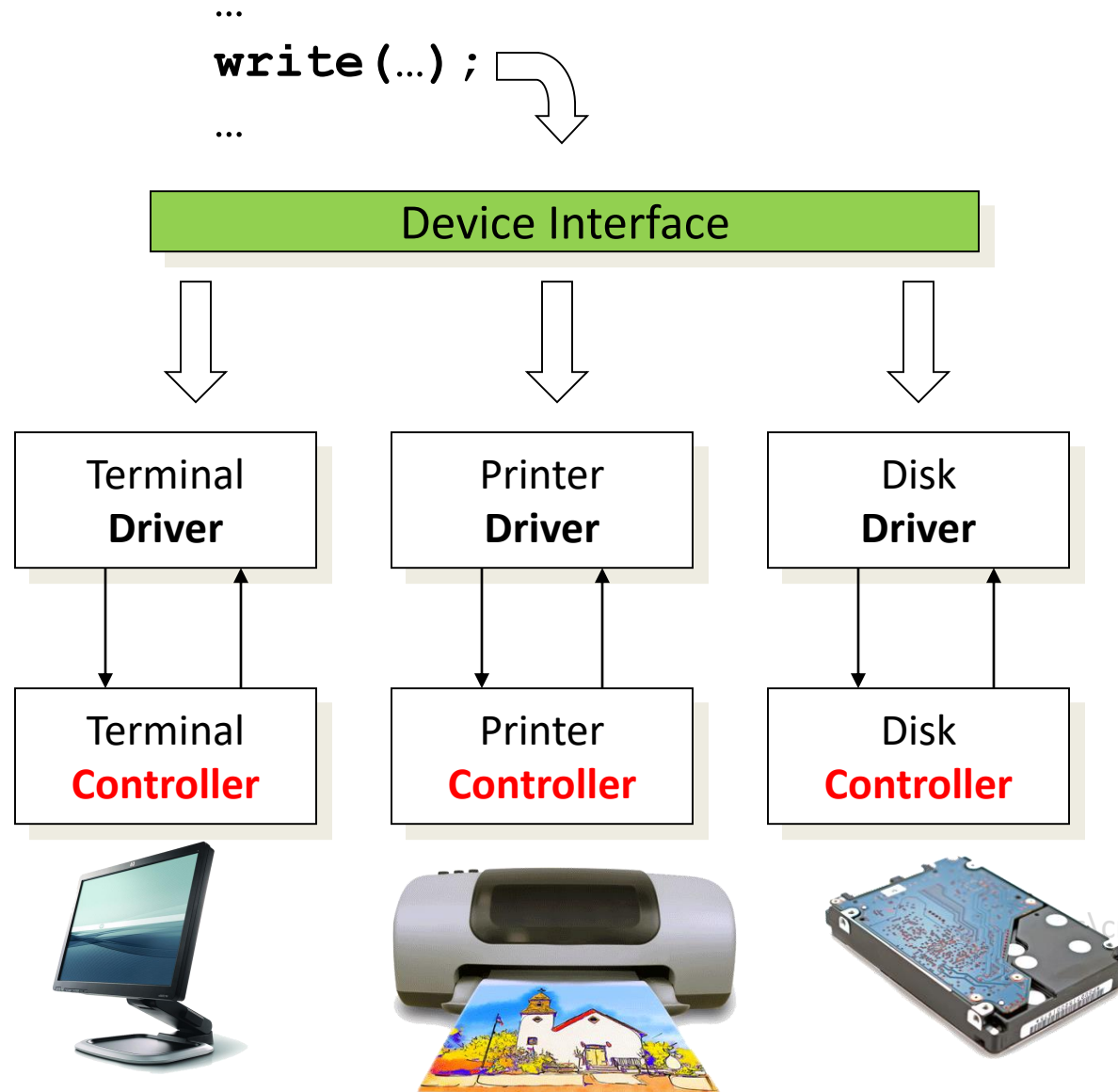
- The software that talks to the device controllers
 - Device specific
 - Tailored to individual device characteristics
 - Written by device manufacturers
 - Part of the OS Kernel
- Know about the details of the devices
 - Disk driver knows about sectors, tracks, cylinders, heads, arm motions, motor drives
 - Mouse driver knows about button pressed

User/OS method interface

- The same interface is used to access devices (like disks and network lines) and more abstract resources like files
- 4 main methods:
 - **open()**, **close()**, **read()**, **write()**
- Semantics depend on the type of the device (block, char, net)
 - These methods are system calls because they are the methods the OS provides to all processes.

PPTs from others\paul.rutgers.edu_cs519_S02\IO-File.ppt

The Device Driver Interface



Interacting with the Device controller

PPTs.2012\PPTs from others\www.cs.bilkent.edu.tr~korpe_courses_cs342spring2010\lecture13_io.ppt

- Host (CPU+Memory) and Device Controller interaction (data transfers and control) can be in one of 3 ways:
 - Polling
 - Interrupt driven I/O
 - Interrupt driven with help of DMA

I/O Code with Busy Waiting

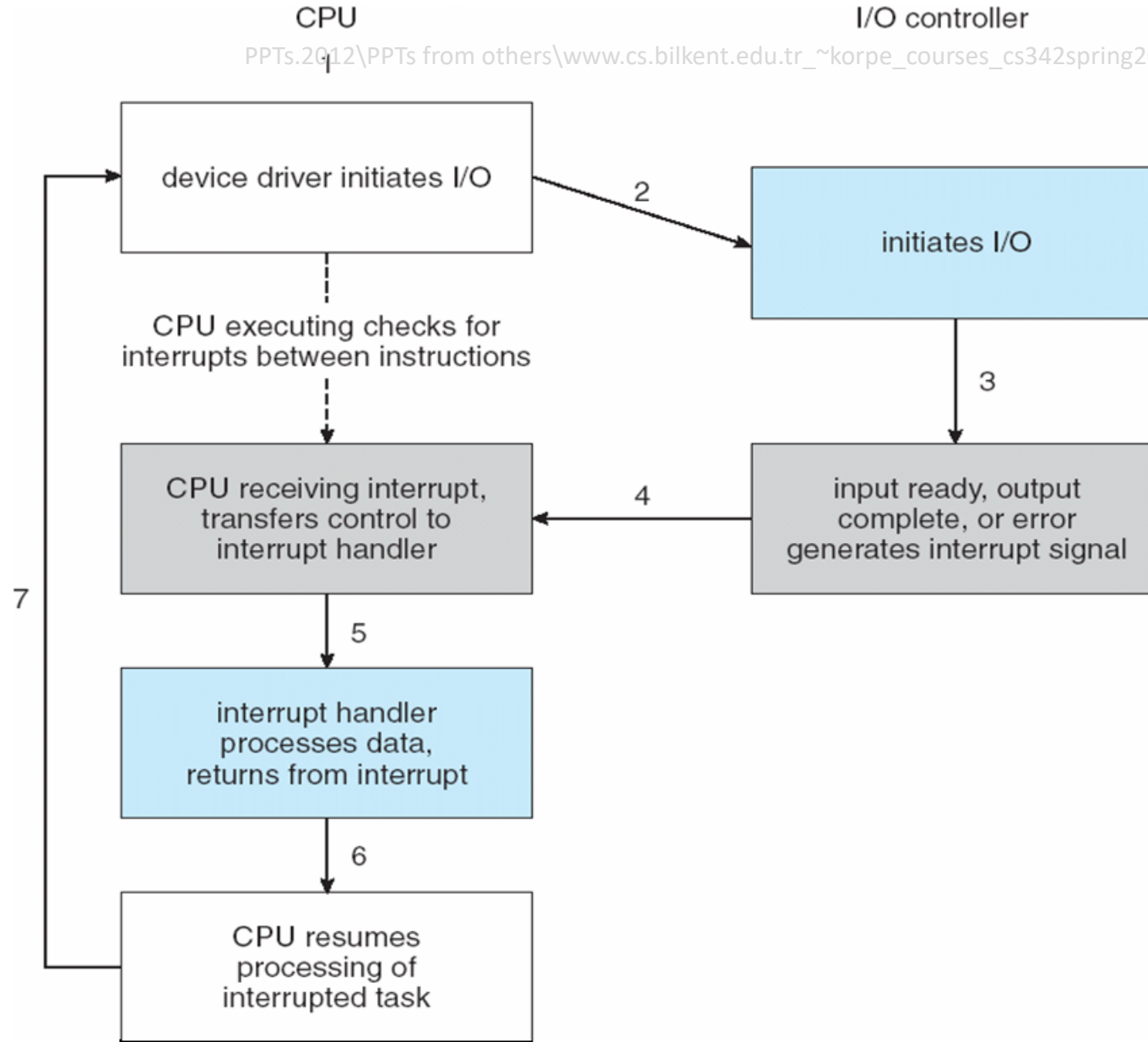
- Goal: user wants to send N bytes from Buffer to an I/O device (e.g. printer or floppy)
- Device controller has registers for status and for putting data
- Basic approach (too wasteful for CPU):

```
Copy data into kernel buffer B;  
for (i=0; i< N; i++) {  
    /* wait for device to be free */  
    while (*DeviceStatusReg != READY);  
    *DeviceDataReg = B[i];  
}
```

Transferring Data between host and device: Interrupts [中断]

- CPU Interrupt-request line triggered by I/O device
- **Interrupt handler** receives interrupts
- **Maskable** to ignore or delay some interrupts
- Interrupt vector to dispatch interrupt to the correct handler
 - Based on priority
 - Some **nonmaskable**
- Interrupt mechanism also used for exceptions

Interrupt-Driven I/O Cycle



I/O Code with Interrupts

- Goal: To let CPU schedule other processes while device is busy processing data
- Code for driver:
Copy data into kernel buffer B
while (*DeviceStatusReg != ready);
DeviceDataReg = B[0]; / Do only the first step */
InvokeScheduler();
- Code for interrupt handler:
if (N == 0) Wakeup(); /* unblock the process */
else {
 DeviceDataReg = B[i];
 N--; i++;
};
ResetInterrupt(); /* Notify device as an ack */

Intel Pentium Processor Event-Vector Table

vector number	description
0	divide error
1	debug exception
2	null interrupt
3	breakpoint
4	INTO-detected overflow
5	bound range exception
6	invalid opcode
7	device not available
8	double fault
9	coprocessor segment overrun (reserved)
10	invalid task state segment
11	segment not present
12	stack fault
13	general protection
14	page fault
15	(Intel reserved, do not use)
16	floating-point error
17	alignment check
18	machine check
19–31	(Intel reserved, do not use)
32–255	maskable interrupts

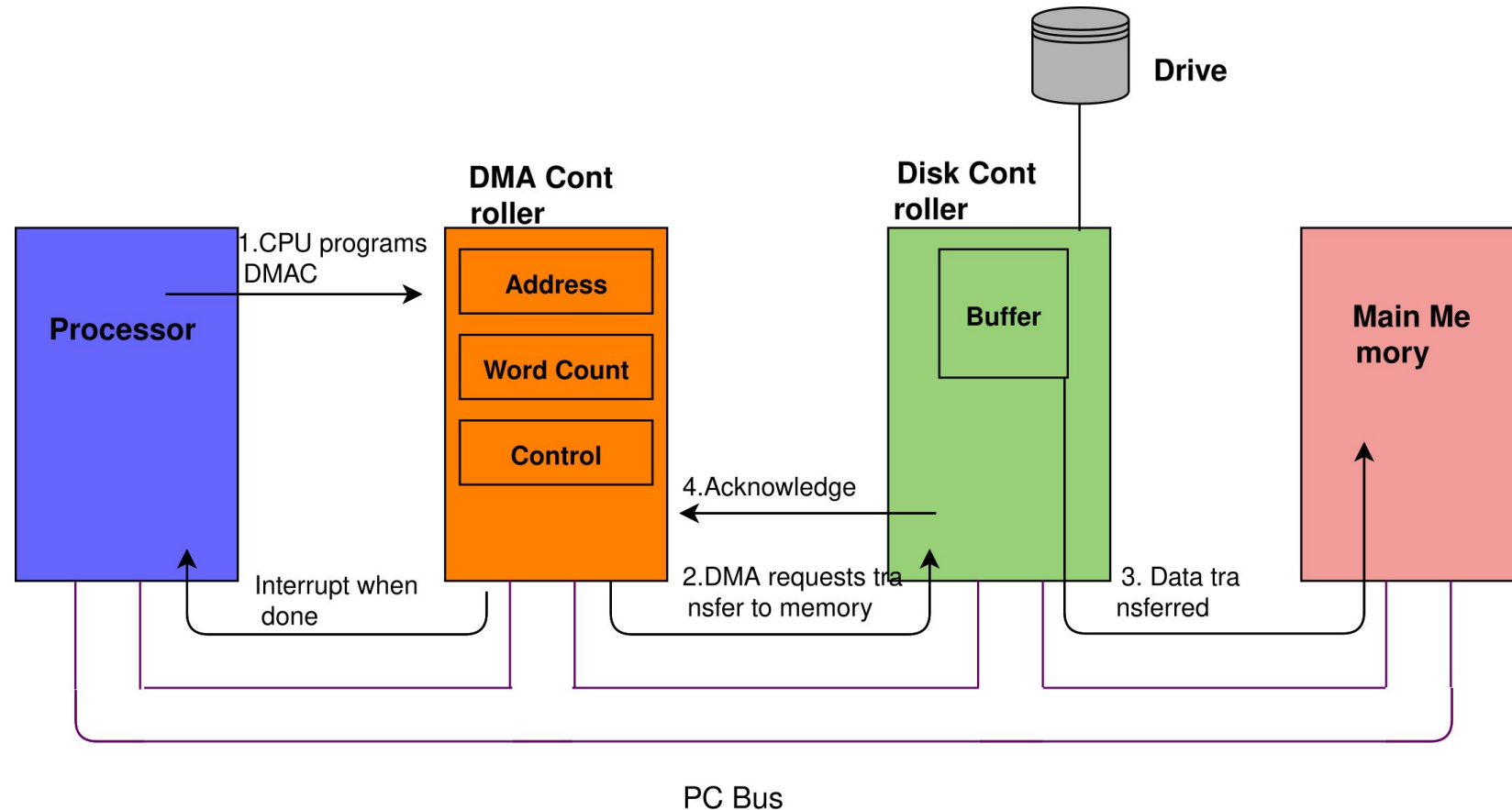
DMA: Direct Memory Access [直接存储器存取]

PPTs.2012\PPTs from others\www.cs.bilkent.edu.tr~korpe_courses_cs342spring2010\lecture13_io.ppt

- Used to avoid **programmed I/O** for large data movement

Requires **DMA** controller

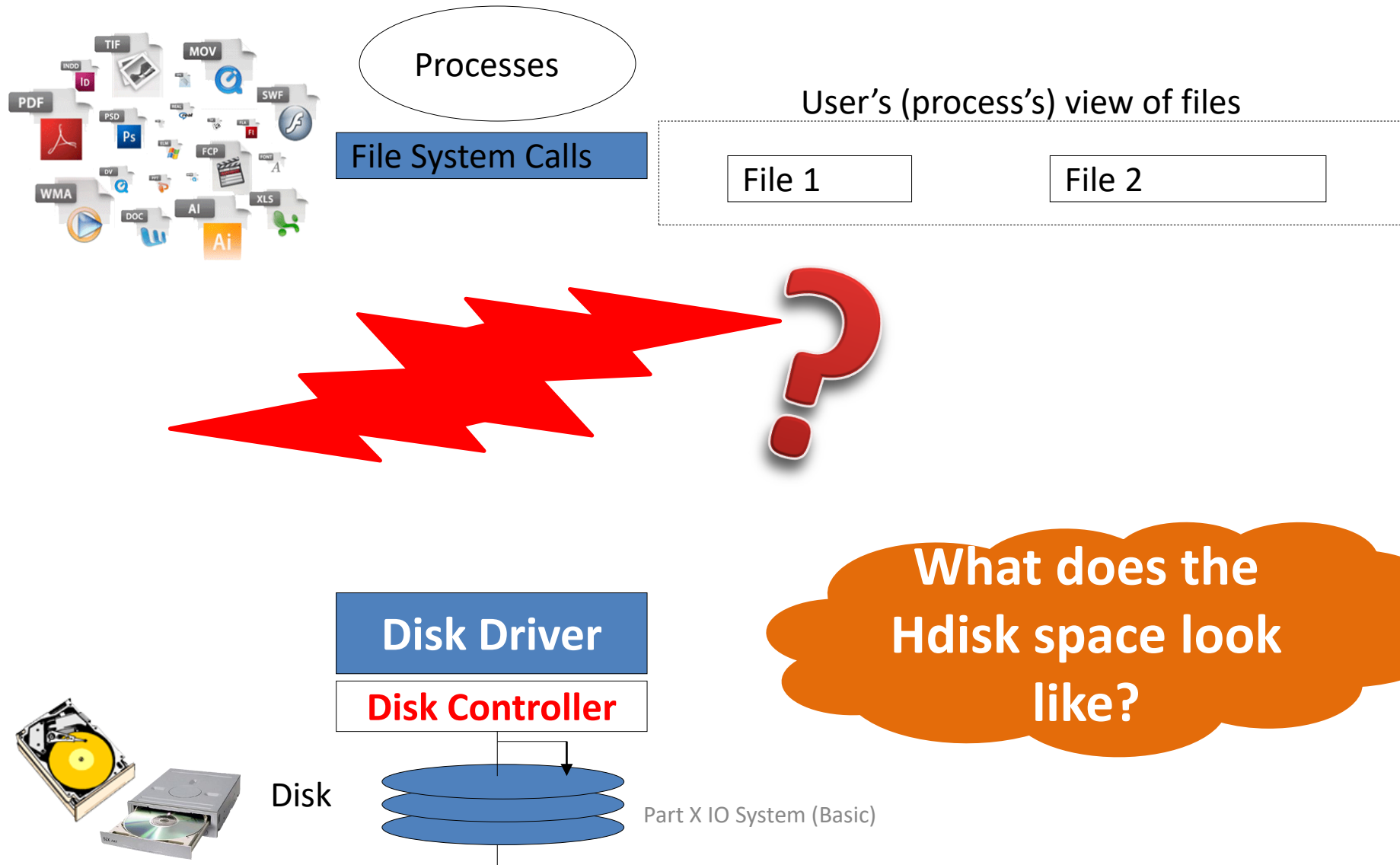
Bypasses CPU to transfer data directly between I/O device and memory



Mechanism of DMA Transfer

- A program running on CPU sets up the registers in **DMA controller** (e.g. write N bytes into memory starting at location XYZ)
- DMA controller sets up the registers in **device controller** to request the transfer
 - Original request may be divided into chunks
- Disk controller transfers the data **directly** into memory
 - Bus architecture should resolve contention for memory
- Disk controller notifies DMA controller
- When the original request is entirely processed, DMA controller interrupts the CPU

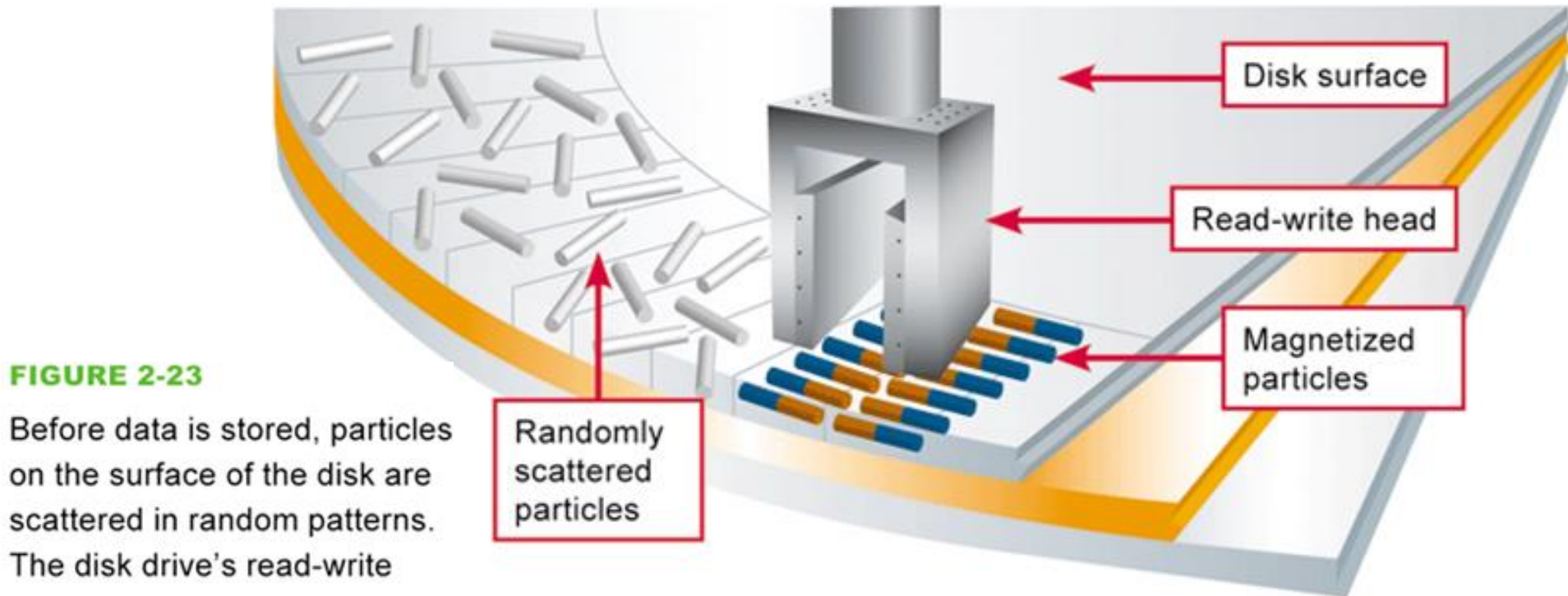
Now we know how to connect Hdisk



- General structure to connect devices –
 - Abstraction/Interface in OS to communicate with the diverse devices
 - IO devices - **Categories** of devices
 - **IO operations**
 - **Connect** all together & structure of IO module in OS
 - **Communication** types between CPU and IO devices
- Taking (Magnetic) Disk for instance
 - So-called “linear address sector space”
 - Organize sectors into partitions, and so-called “linear addressed block space”
 - Optical disk is similar

The basis of magnetic storage media

- Magnetic storage stores data by magnetizing microscopic particles on the disk or tape surface



Magnetic Disk

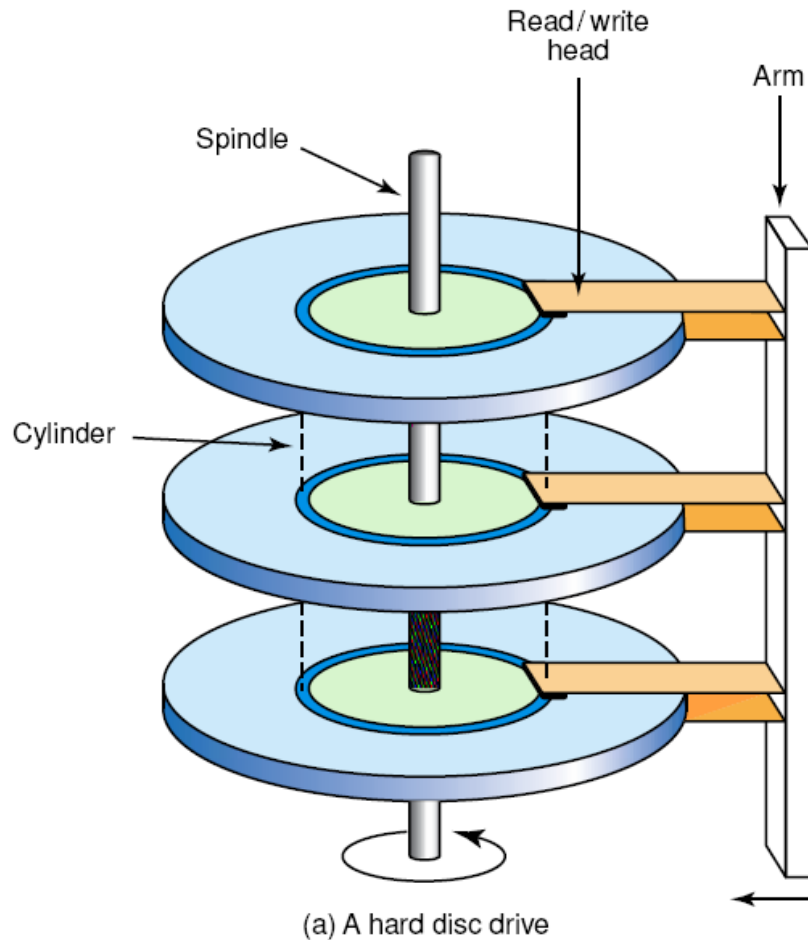


Figure 5.5 The organization of a magnetic disk

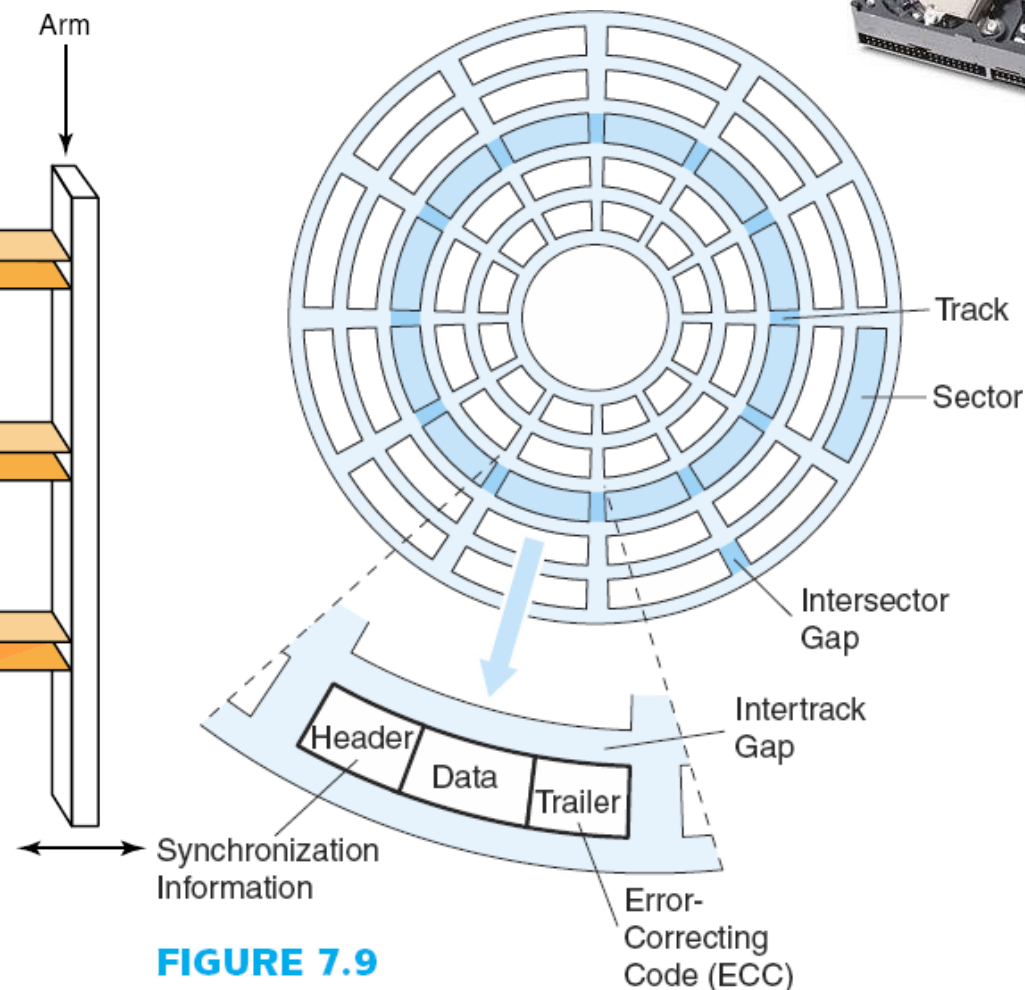


FIGURE 7.9

Disk Sectors Showing Intersector Gaps and Logical Sector Format

Cont'

- Numbering for the sectors
 - (**c**ylinder #, **h**ead #, **s**ector #)

Sectors are organized by the predefined rules, like the inner most one is numbered 0 sector.

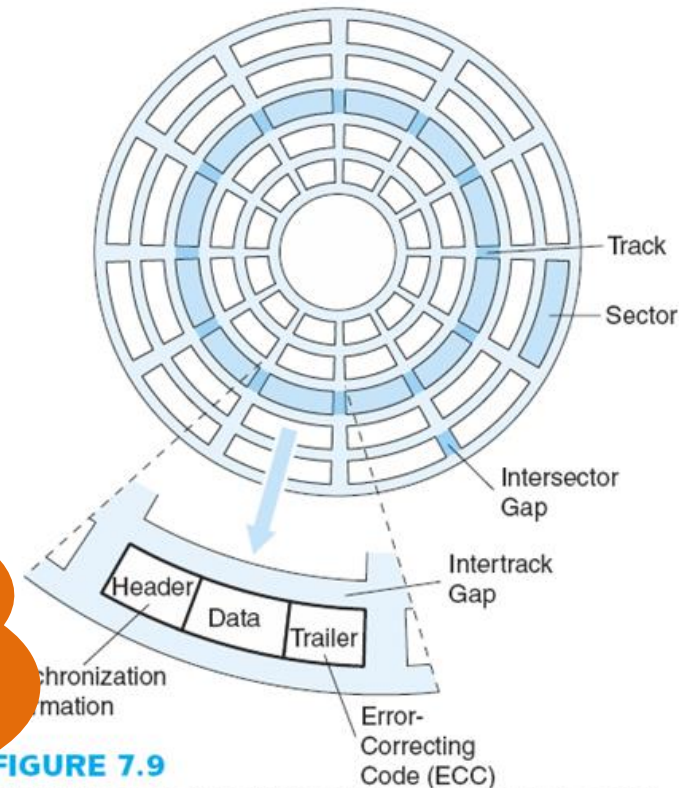
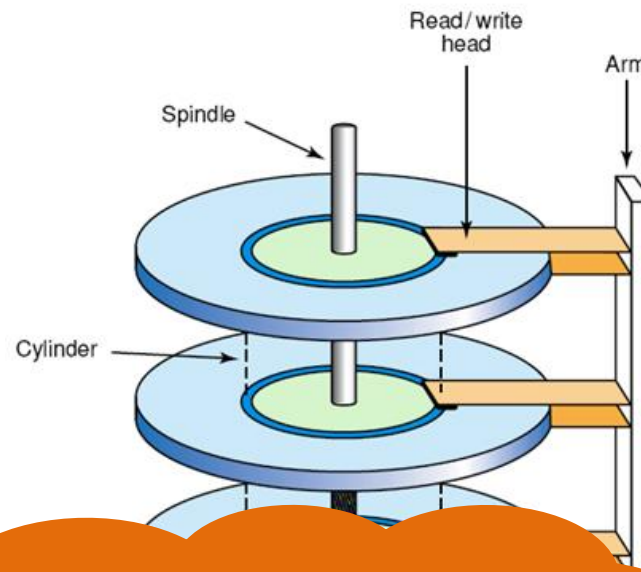


FIGURE 7.9
Disk Sectors Showing Intersector Gaps and Logical Sector Format

The sector number is recorded in the header of each sector.

LBA now for CHS

https://en.wikipedia.org/wiki/Logical_block_addressing

- Logical block addressing
 - Logical block addressing (LBA) is a common scheme used for specifying the location of blocks of data stored on computer storage devices, generally secondary storage systems such as hard disk drives.

LBA and CHS equivalence
with 16 heads per
cylinder

LBA value	CHS tuple
0	0, 0, 1
1	0, 0, 2
2	0, 0, 3
62	0, 0, 63
63	0, 1, 1
945	0, 15, 1
1007	0, 15, 63
1008	1, 0, 1
1070	1, 0, 63
1071	1, 1, 1
1133	1, 1, 63
1134	1, 2, 1
2015	1, 15, 63
2016	2, 0, 1
16,127	15, 15, 63
16,128	16, 0, 1
32,255	31, 15, 63
32,256	32, 0, 1
16,450,559	16319, 15, 63
16,514,063	16382, 15, 63

CHS tuples can be mapped to LBA address with the following formula:^{[5][6]}

$$LBA = (C \times HPC + H) \times SPT + (S - 1)$$

where

- *C*, *H* and *S* are the cylinder number, the head number, and the sector number
- *LBA* is the logical block address
- *HPC* is the maximum number of heads per cylinder (reported by disk drive, typically 16 for 28-bit LBA)
- *SPT* is the maximum number of sectors per track (reported by disk drive, typically 63 for 28-bit LBA)

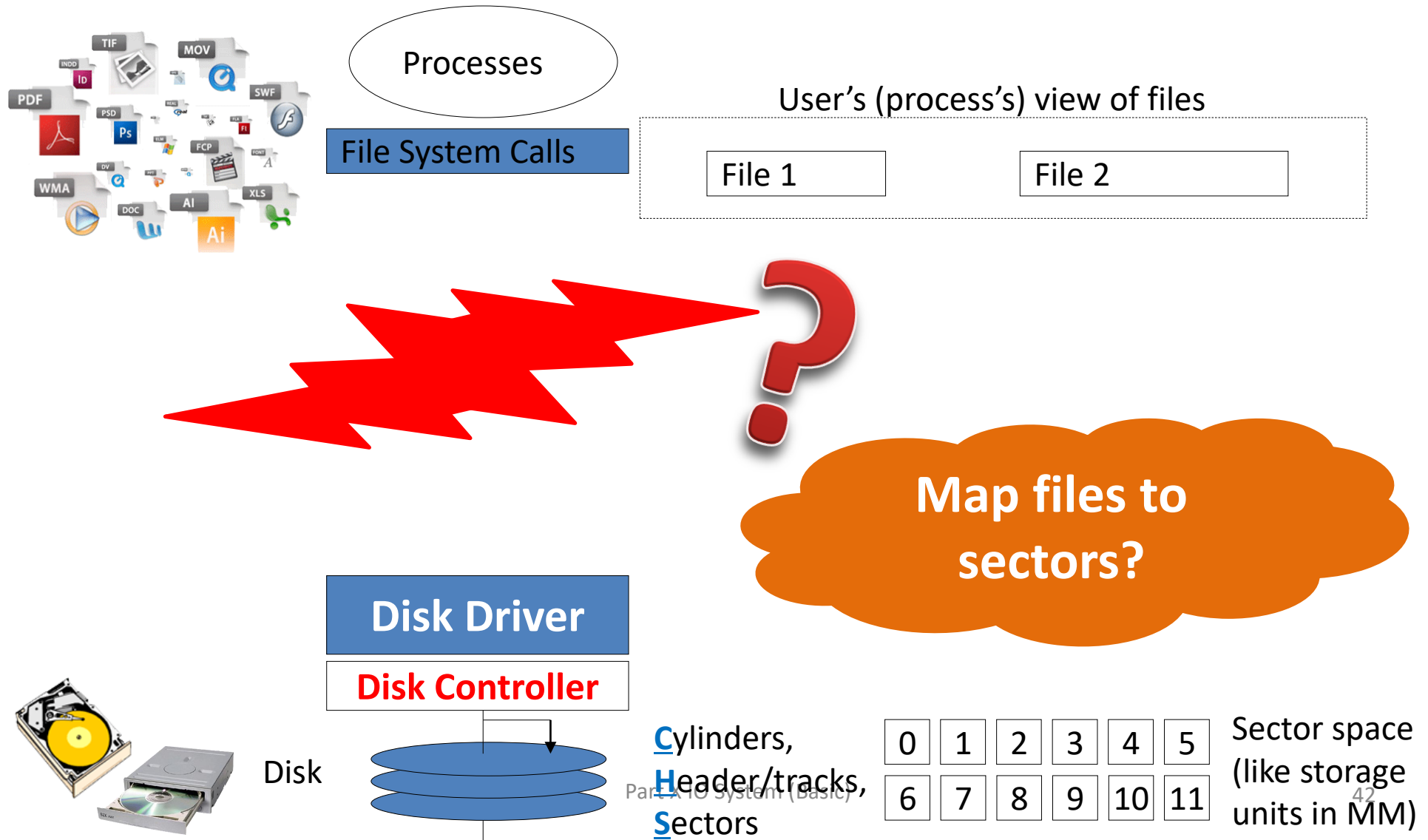
LBA addresses can be mapped to CHS tuples with the following formula (“mod” is the modulo operation, i.e. the remainder, and “÷” is integer division, i.e. the quotient of the division where any fractional part is discarded):

$$C = LBA \div (HPC \times SPT)$$

$$H = (LBA \div SPT) \bmod HPC$$

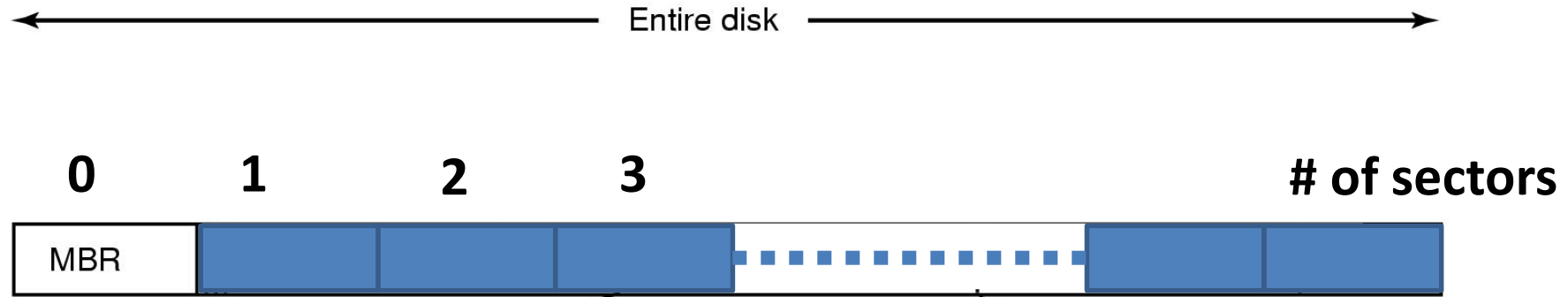
$$S = (LBA \bmod SPT) + 1$$



Mapping files to blocks and sectors?



- General structure to connect devices –
 - Abstraction/Interface in OS to communicate with the diverse devices
 - IO devices - **Categories** of devices
 - **IO operations**
 - **Connect** all together & structure of IO module in OS
 - **Communication** types between CPU and IO devices
- Taking (Magnetic) Disk for instance
 - So-called “linear address **sector** space”
 - Like the storage units in MM
 - Organize sectors into partitions, and so-called “linear addressed **block** space”
 - Optical disk is similar

Some special sectors in HD



- After LLF (low level formatting, which has usually been carried out by HD manufacturer like  , by default your disk is a collection of numbered sectors
 - Divide a disk into sectors that **the controller can read and write** (write C.H.S into header, etc.)
- Sector 0 is called the **Master Boot Record (MBR : 主引导记录)** which is used when booting the computer
 - Some information about the manufacturing – date, size ...
 - At the end, the table of PARTITIONs – “logic disks” in the HD

Sector is too small!

Several sectors are combined to create clusters or blocks

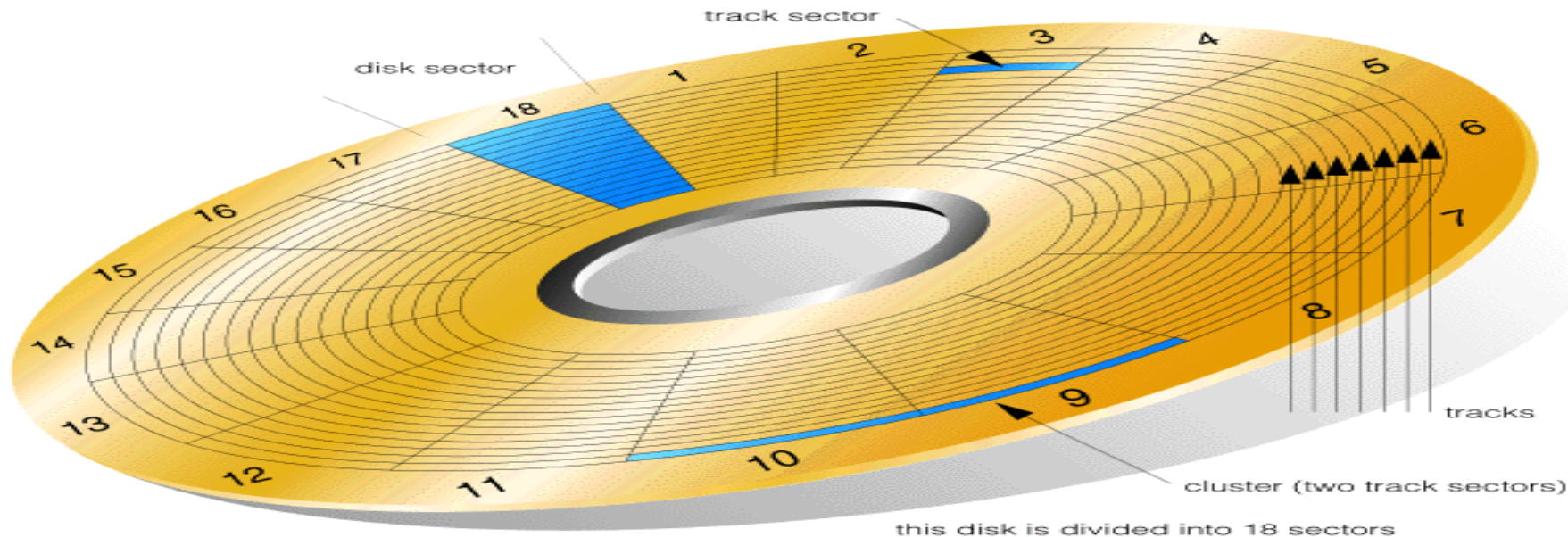
cluster (Windows and Macs) or **block** (UNIX) = **The number of sectors which is allocated on the disk each time a file needs space on the disk.**

Windows 95 (later versions) and Windows 98 using FAT32

- 1 cluster = 8 sectors (4K bytes)
- Recognizes disk drives up to 2 terabytes (2 trillion bytes)

NTFS?

PPTs\Part XII\Part XII magnetic media].ppt



Disk Partition [磁盘分区]

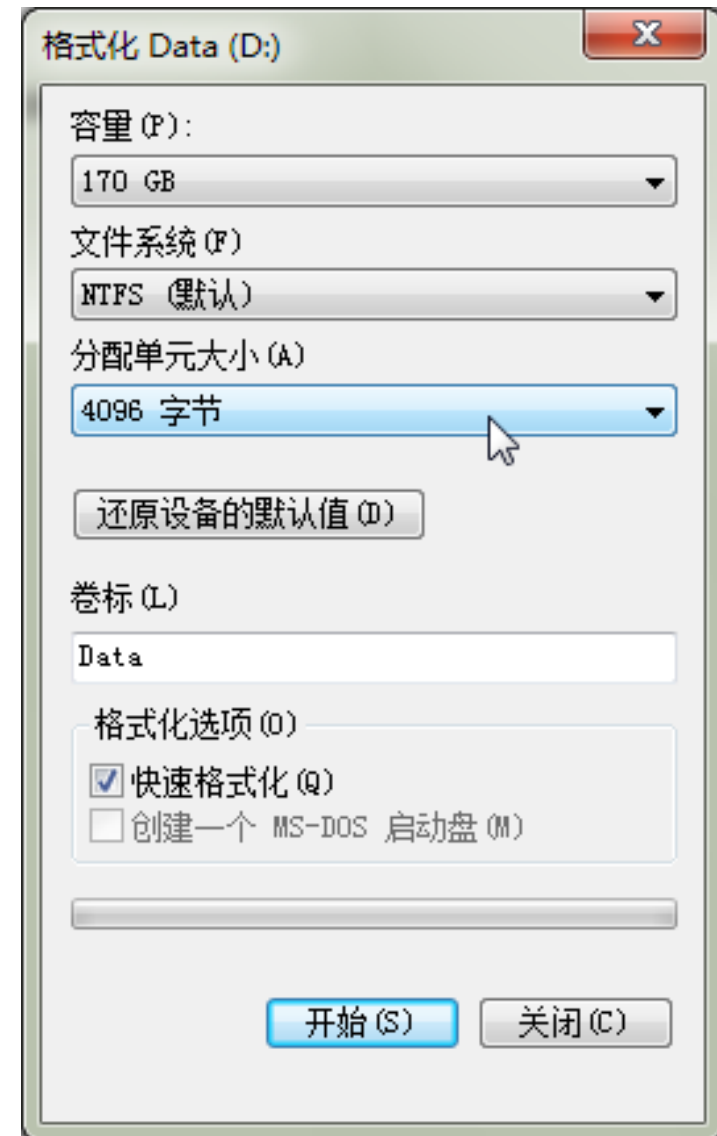
- To use a disk to hold files, OS still needs to record its own data structures on the disk
- **Partition** the disk into cylinders
 - Each partition can be used as a separate disk
- **Logical formatting** or “making a file system”
 - Store the initial file-system data structures on the disk...
 - Maps of free and available space (used later)
 - Initial empty directory

We'll learn FAT, i-node, NTFS – the way to organize files - later

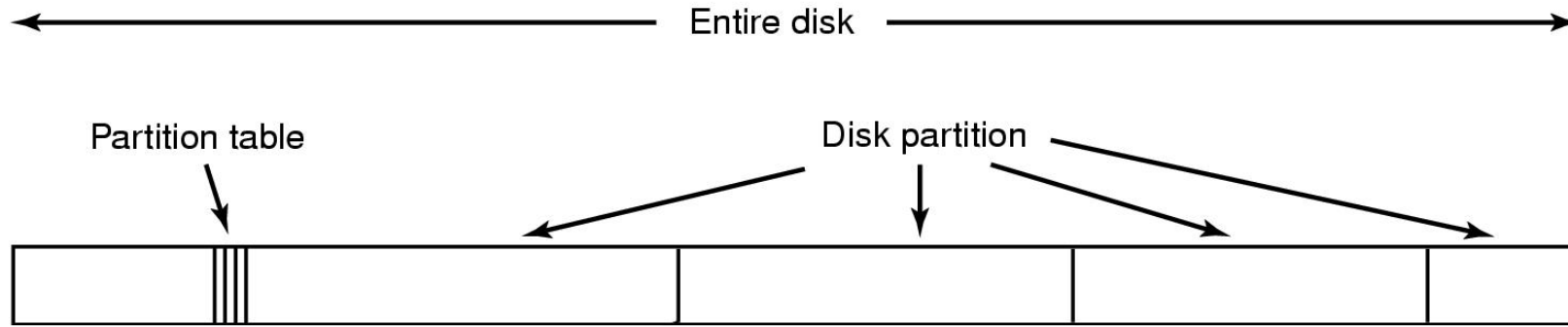
In fact this is like organize storage units of MM into frames

First, organize sectors in a partition into blocks

- The popular size in Windows now is 4KB (4096 bytes)
 - Which means a block usually contains 8 sectors
- You can set the File System you want to use for this partition
 - NTFS (NT File System) here
- You can also label it
 - Volume index here “Data”



PARTITIONS in HD



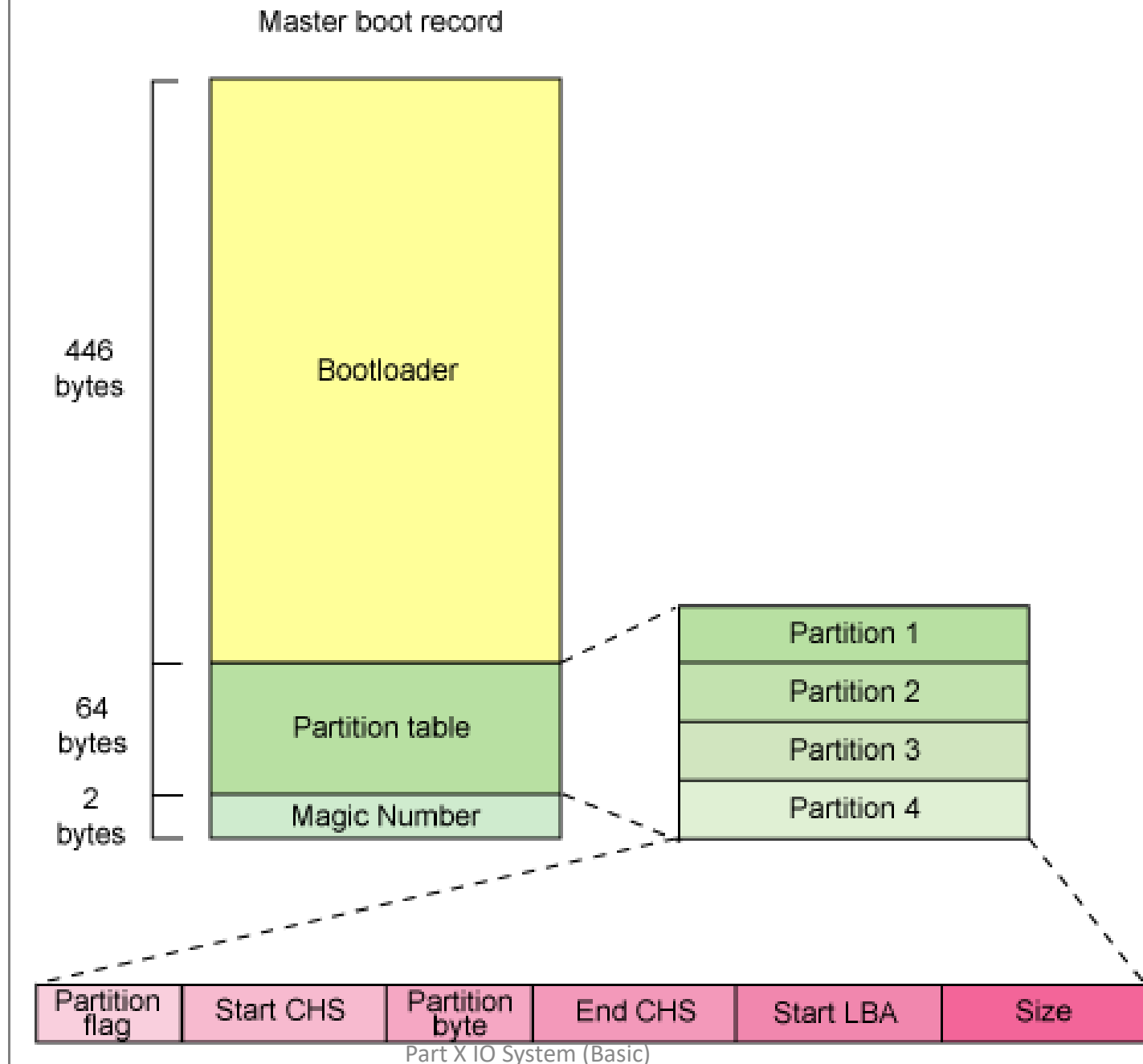
MBR

- Do you remember “What does OS do after turning-on but before your OS?”
 - After POST, BIOS (CMOS) will check the HDs to locate the OS(s) by reading the MBR of each HD
 - The partition table in MBR contains all the location information (starting CHS) of partitions
 - Each partition is further organized to contain the files based on a complex data structure called **FILE SYSTEM**
 - If in primary partition, bootstrap loader reads necessary OS’s codes with help of file system into main memory, and CPU executes those programs – you can use OS now

MBR – for demonstration

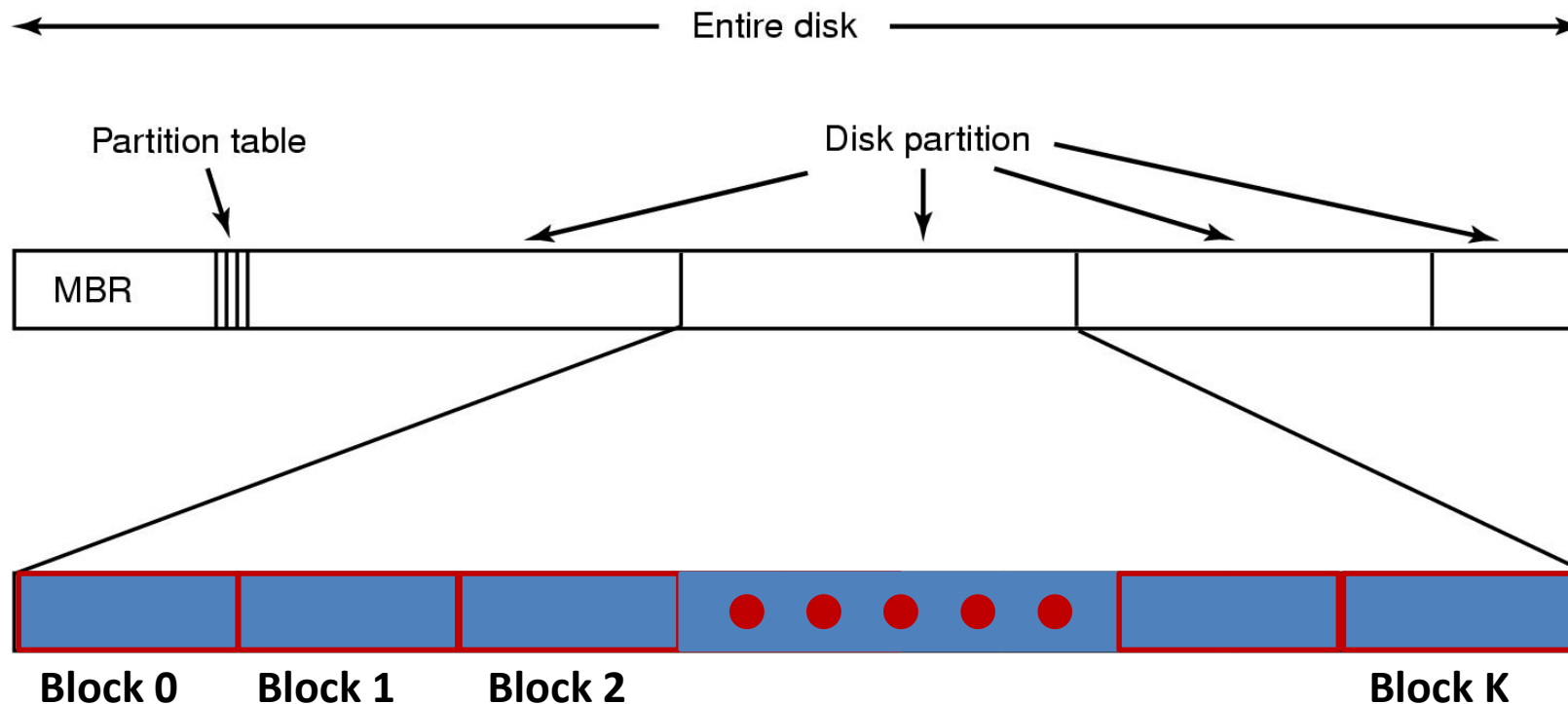
Structure of a classical generic MBR

Address		Description		Size in bytes
Hex	Dec			
+000h	+0	Bootstrap code area		446
+1BEh	+446	Partition entry #1	Partition table (for primary partitions)	16
+1CEh	+462	Partition entry #2		16
+1DEh	+478	Partition entry #3		16
+1EEh	+494	Partition entry #4		16
+1FEh	+510	55h	Boot signature ^[nb 1]	2
+1FFh	+511	AAh		
Total size: 446 + 4*16 + 2				512



Disk Space Organization

- Disk can be **PARTITIONED**
 - Each partition can have a different OS and/or different file system
 - One partition can be swap space for main memory
- Each partition has

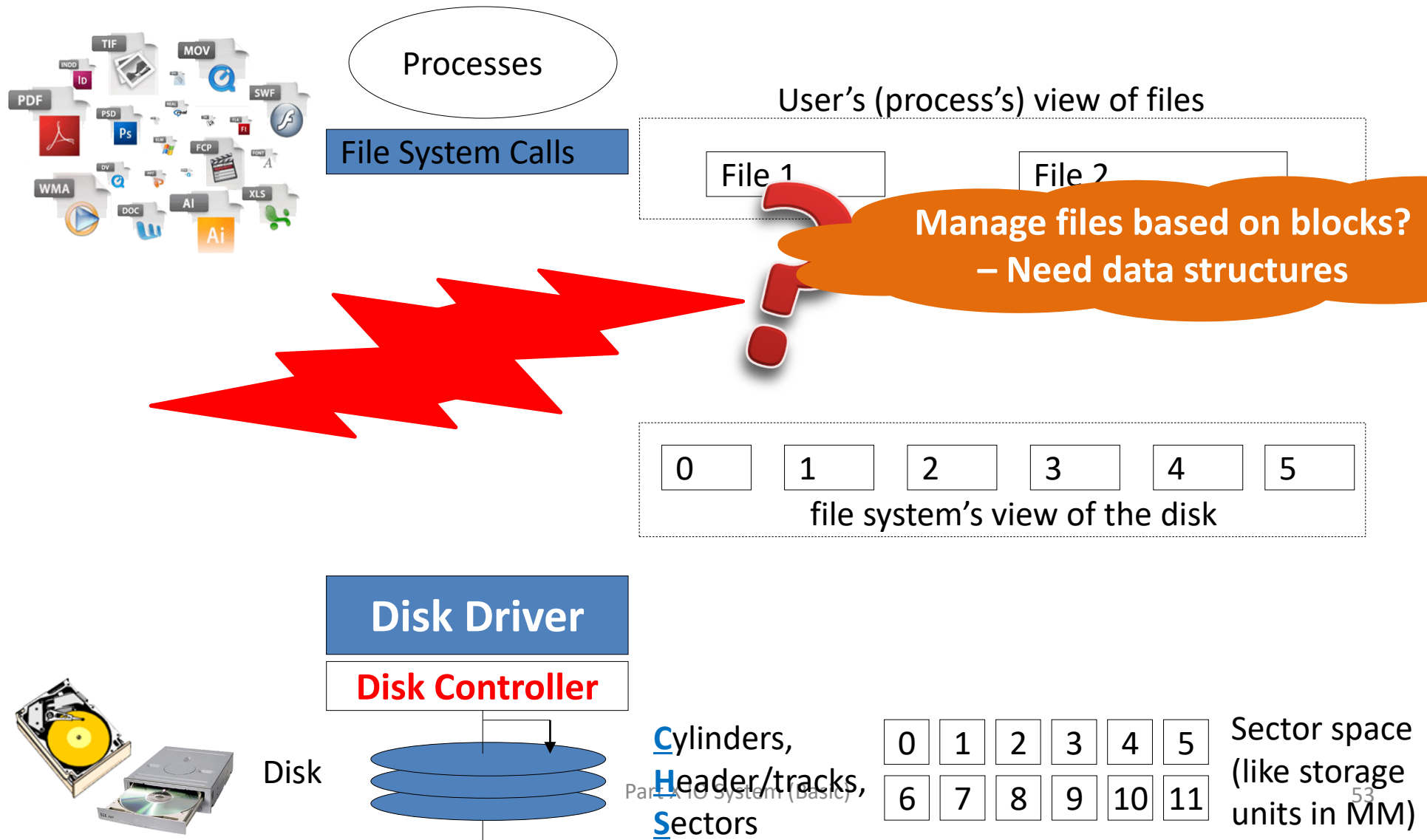


Organize sectors into blocks?

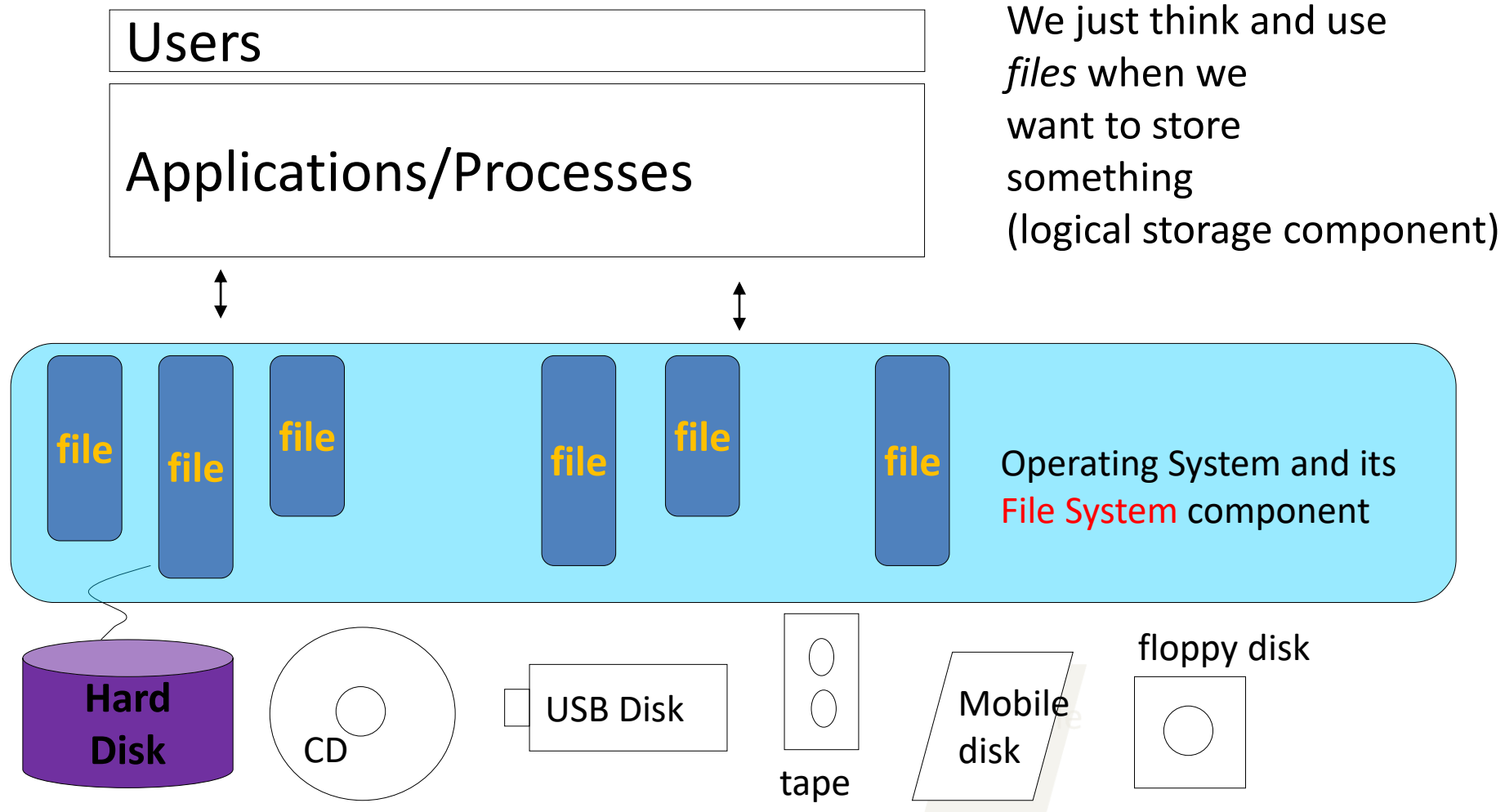
- Given a partition whose starting sector is 1024, and the size of a block is defined as 4KB, could you determine the sectors for a **block 7** in this partition?
 - Since $4\text{ KB} = 8 * 512\text{B} = 8\text{ sectors}$, the 1st sector of the **block 7** should be: $7 * 8 = 56^{\text{th}}$ sector
 - So the 1st sector of **block 7** in this partition is just: $1024 + 56 = 1080\text{ sector}$
- If you want to store a file of 17 KB, it's easy to know we need $\lceil 17/4 \rceil = 5\text{ blocks}$.
 - Of course we need record some information to have the blocks used to store your file.

We'll learn those data structures later

Now we have linear addressed block space for files

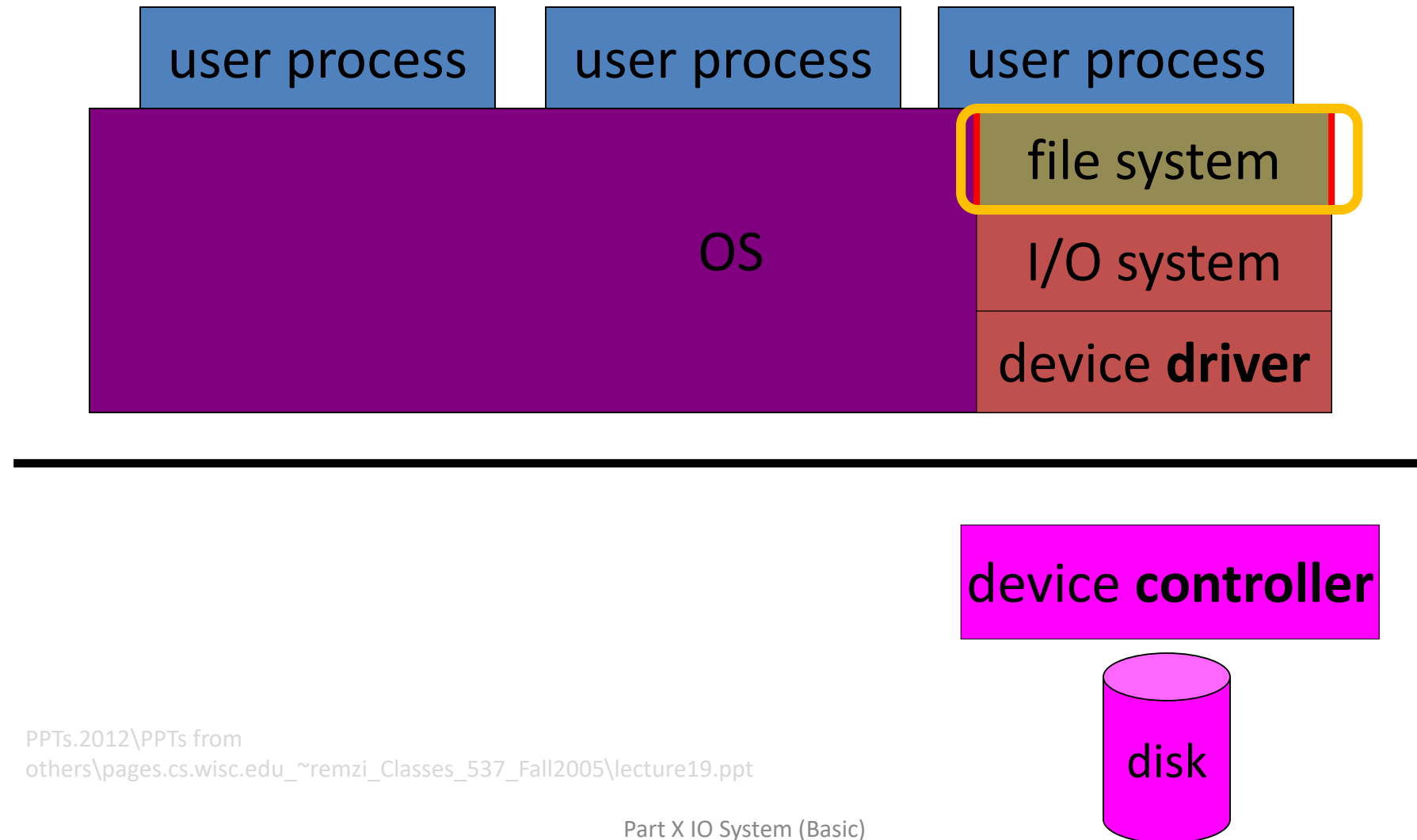


Those data structures are maintained by File system



We just think and use *files* when we want to store something (logical storage component)

I/O System – connect devices together



- General structure to connect devices –
 - Abstraction/Interface in OS to communicate with the diverse devices
 - IO devices - **Categories** of devices
 - **IO operations**
 - **Connect** all together & structure of IO module in OS
 - **Communication** types between CPU and IO devices
- Taking (Magnetic) Disk for instance
 - So-called “linear address sector space”
 - Organize sectors into partitions, and so-called “linear addressed block space”
 - Optical disk is similar

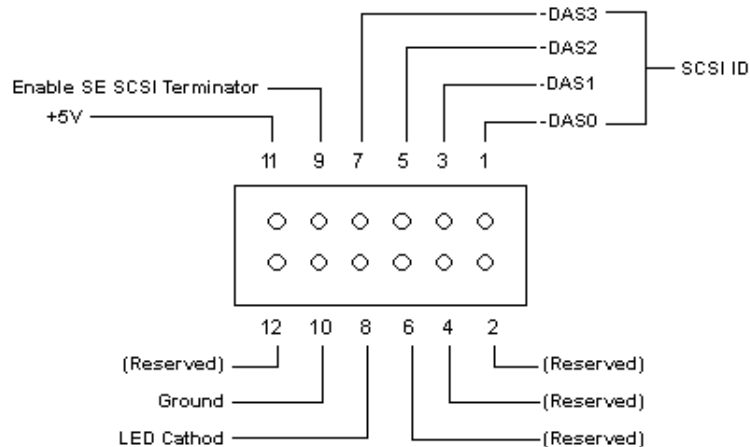
IDE Hard Disk

Most Hard disks & CD/DVD drivers are **IDE** (整合式电子设备, **Integrated Device Electronics**)



SCSI Hard disk

- **SCSI**（小型计算机系统接口， **Small Computer Systems Interface**） provide mode flexibility, but is more expensive。
- Usually used for servers or workstations

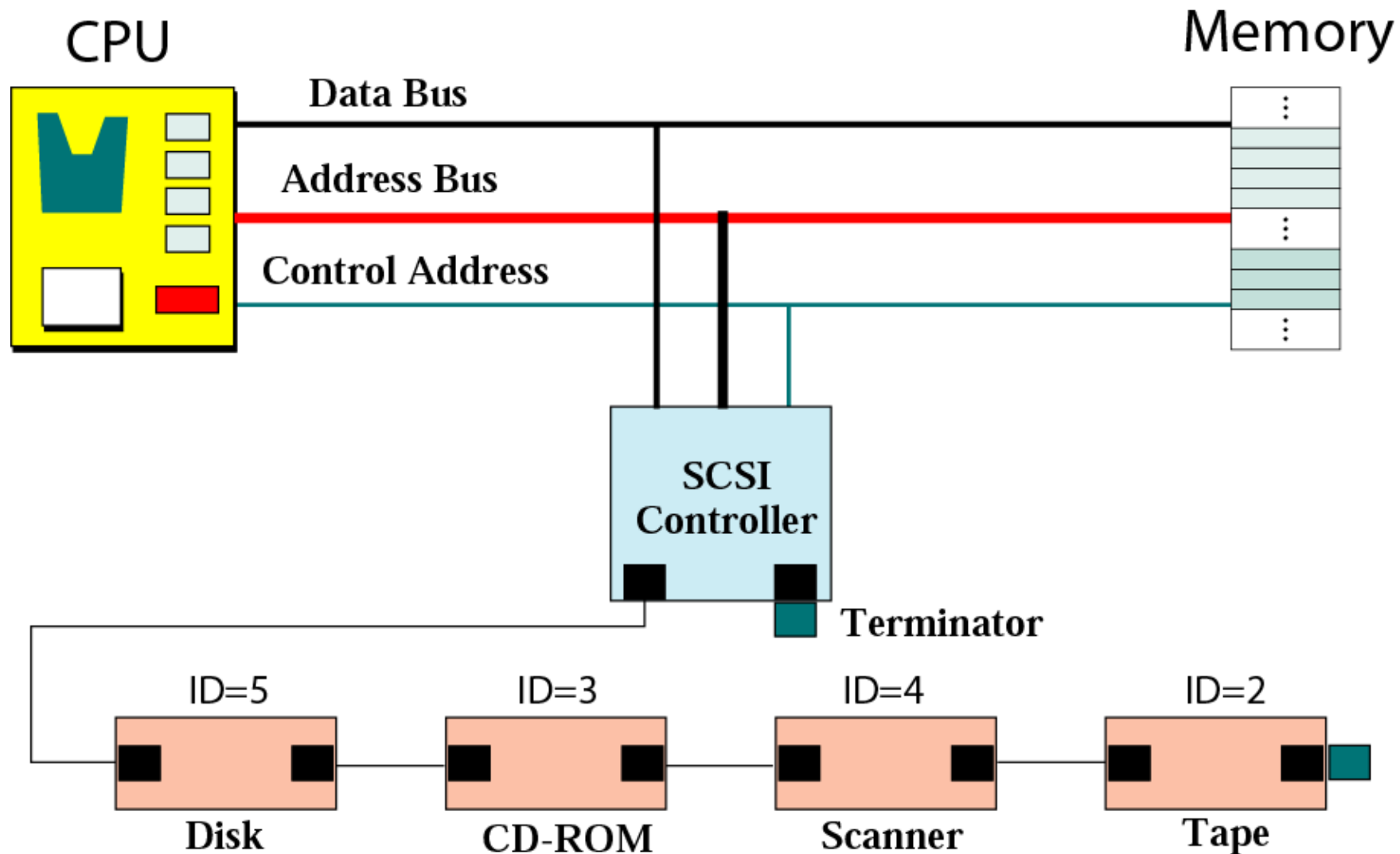


SCSI Address Bits

3 2 1 0	Address 0	3 2 1 0	Address 8
0000	Address 0	1000	Address 8
0001	Address 1	1001	Address 9
0010	Address 2	1010	Address 10
0011	Address 3	1011	Address 11
0100	Address 4	1100	Address 12
0101	Address 5	1101	Address 13
0110	Address 6	1110	Address 14
0111	Address 7	1111	Address 15

00.Computer & Related\PPts.2009\Part II Computer architecture.pptx

SCSI controller (a chain)

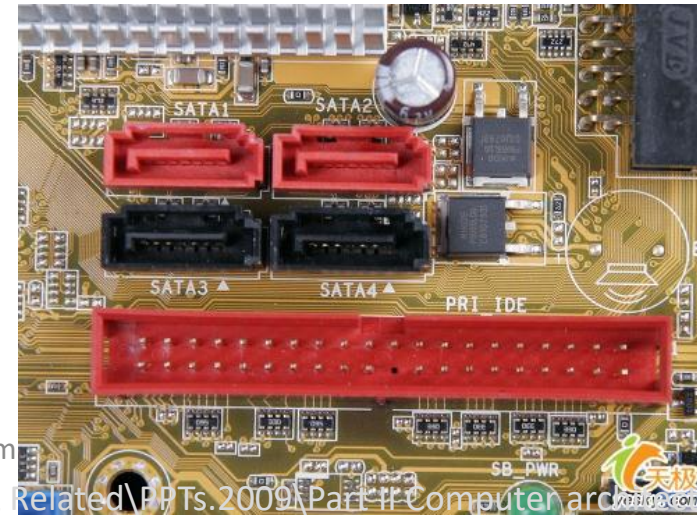


SATA



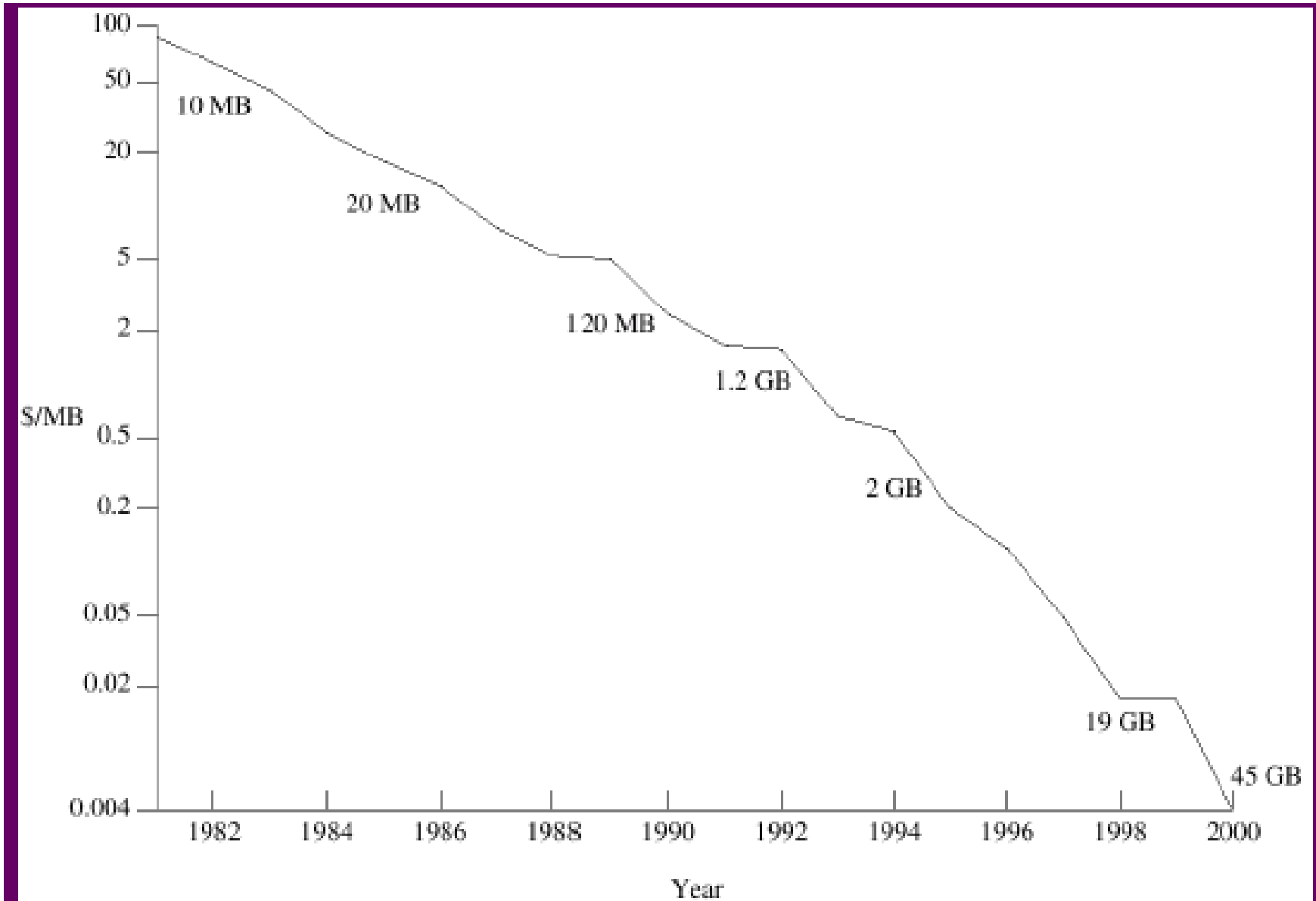
SATA: The serial ATA, or SATA computer bus, is a storage-interface for connecting host bus adapters to mass storage devices such as hard disk drives and optical drives. Serial ATA was designed to replace the older **ATA** (AT Attachment) standard (also known as **EIDE**).

PATA: Parallel ATA (PATA) is an interface standard for the connection of storage devices such as hard disks, solid-state drives, and CD-ROM drives in computers. The standard is maintained by X3/INCITS committee. It uses the underlying AT Attachment and **ATA Attachment Packet Interface (ATA/ATAPI)** standards

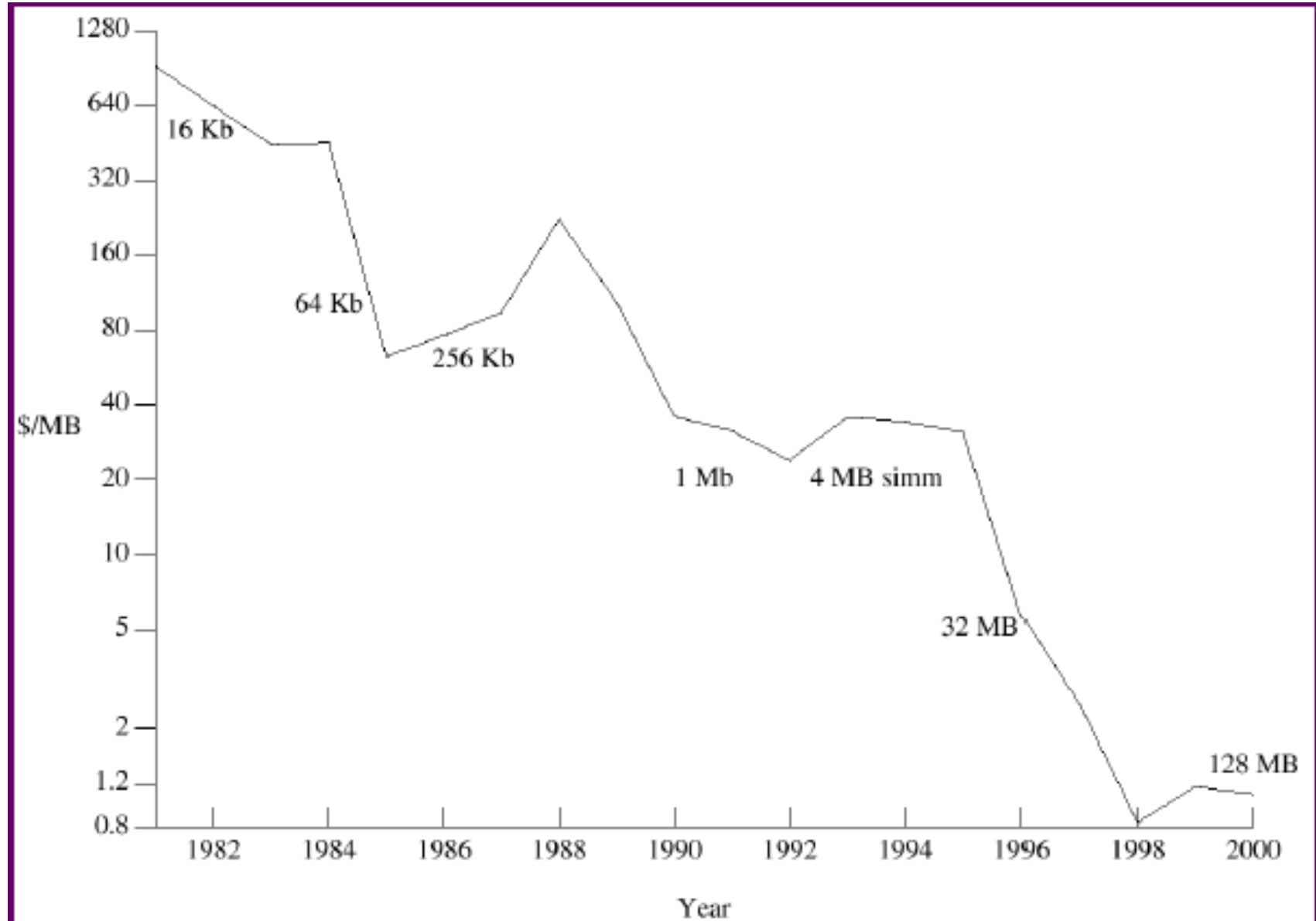


http://en.wikipedia.org/wiki/Serial_ATA

Price per Megabyte of Magnetic Hard Disk, From 1981 to 2000



Price per Megabyte of DRAM, From 1981 to 2000



CD (Compact Disk)

crater ['kreɪtə]
n. 弹坑; 陨石坑
pit n. 坑

FIGURE 2-26

As seen through an electron microscope, the pits on an optical storage disk look like small craters. Each pit is less than 1 micron (one millionth of a meter) in diameter—1,500 pits lined up side by side are about as wide as the head of a pin.

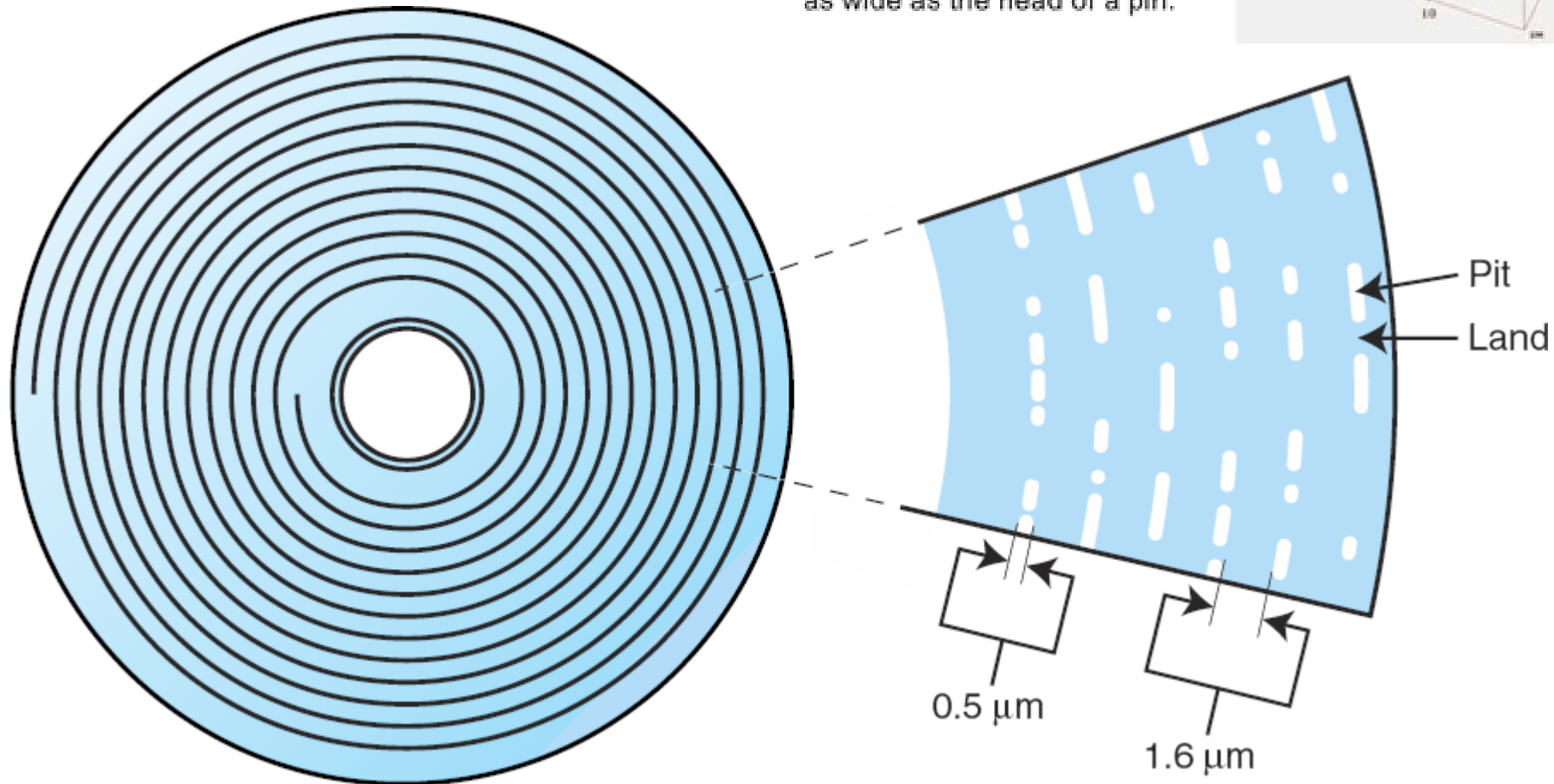
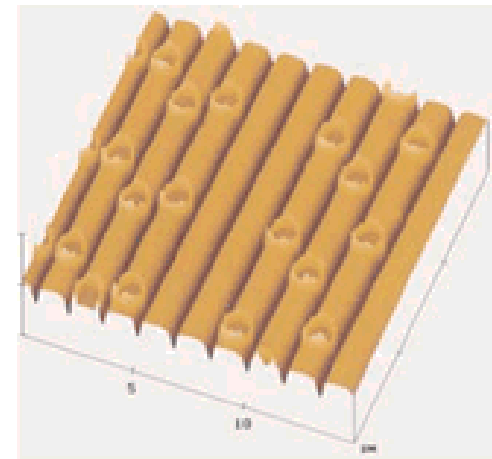
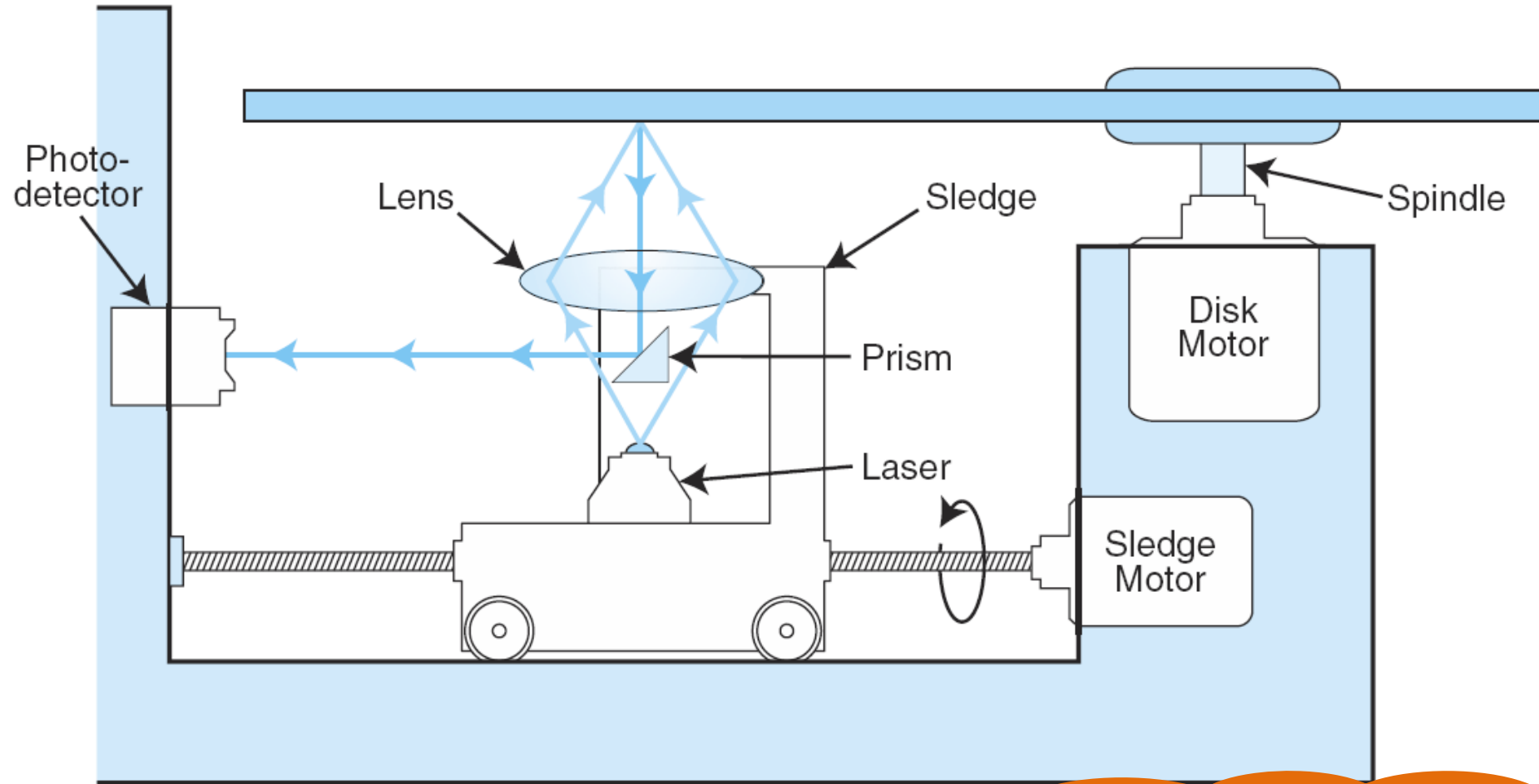


FIGURE 7.14 CD Track Spiral and Track Enlargement

The Internals of a CD-ROM Drive



sledge: 雪橇, 雪车

prism: (几何)棱柱 (体), 角柱 (体)

00.Comput

Part Y

**Your responsibility to
know others.**