Device Management

Chapter 4

I/O Devices

- **BLOCK** device can be defined as one that stores information in fixed-size blocks, each one with its own address.

  e.g. Hard disks, CD-ROMs, and USB

- **CHARACTER** device delivers or accepts a stream of characters, without regard to any block structure.

  e.g. Printers, NIC, and mice

Sample devices

I/O devices span wide range of types:

- Keyboard
- Mouse
- Disk
- CRT
- Bit-mapped screen
- GPU
- LED
- Analog-to-Digital converter (ADC)
- Digital-to-Analog converter (DAC)
- Pushbutton
- On/Off switch
- One-bit digital output
- Rotary encoder
- Network interface card (NIC)
- Robot arm
- TV receiver
- Others…

Controllers

- Controller is electronic component that helps manage the I/O device.
- Its job is to convert the serial bit stream into a block of bytes and perform any error correction necessary.
- Block of bytes is typically first assembled, bit by bit, in a buffer inside the controller.
  - Simplest: ADCs, DACs, and some digital lines
  - Common: fairly fancy electronics to hide low-level messiness
  - Most complex: own CPU with millions of lines of codes

Talking to Devices

- Each controller has few registers used for communicating with the CPU.
- By writing into theses registers, the operating system can command the device.
- In addition, many devices have a data buffer that OS can read and write.
- Three approaches:
  - Special instructions, `e.g. IN, %eax, $80`
  - Memory mapping (device pretends to be memory)
  - Direct memory access (DMA) – device bypasses CPU entirely

Special instructions

- Each control register is assigned an I/O port number, an 8- or 16-bit integer.
- The set of all the I/O ports form the **I/O port space** and is protected to that ordinary user programs cannot access it.
- Using special I/O instructions, the CPU can **read** and **write** into the control register port and store the results in CPU register REG.

Memory mapping

- Mapping all the control registers into the memory space.
- Each control register is assigned a unique memory address to which no memory in assigned.
- Usually, the assigned addresses are at the top of the address space.

Direct Memory Access

- CPU needs to address the device controllers to exchange data with them.
- DMA controller has access to the system bus independent of the CPU.
- It contains several registers that can be written and read by the CPU.
- The control registers specify the I/O port to use, the direction of the transfer **(read or write to I/O)**, transfer unit **(byte or word),** and the number of bytes to transfer in one burst.

Direct Memory Access

1. CPU programs DMA controller by setting its registers to know what to transfer and where. It also issues a command to the disk controller telling it to read data from the disk into its internal buffer and verify the checksum.
2. DMA controller initiates the transfer by issuing a read request over the bus to the disk controller.
3. Write to memory bus cycle.
4. When the write is complete, the disk controller sends an acknowledgement signal to the DMA controller, also over the bus.
5. DMA controller increments the memory address to use and decrements the byte count.
6. DMA controller interrupts the CPU for process completion

DMA mode

- **Cycle stealing** – the device controller sneaks in and steals an occasional bus cycle from the CPU once in a while, delaying it slightly (word-at-a-time).
- **Burst mode** – DMA controller tells the device to acquire the bus, issues a series of transfers, then release the bus (block).
- **Fly-by mode** – DMA controller tells the device controller to transfer the data directly to main memory.

Hardware Interrupts

- When an I/O device has finished the work given to it, it causes an interrupt.
- Interrupt is done by asserting a signal on a bus line that it has been assigned.
- This signal is detected by the interrupt controller chip on the parentboard, which then decides what to do.
- To handle the interrupt, the controller puts a number on the address lines specifying which device wants attention and asserts a signal to interrupt the CPU.

Principles of I/O software

- **Device independence** – writing programs that can access any I/O device without specifying the device in advance.
- **Uniform naming** – the name of a file or a device should simply be a string/integer and not depending on the device.
- **Error handling** – errors should be handled as close to the hardware as possible.

Ways to perform I/O

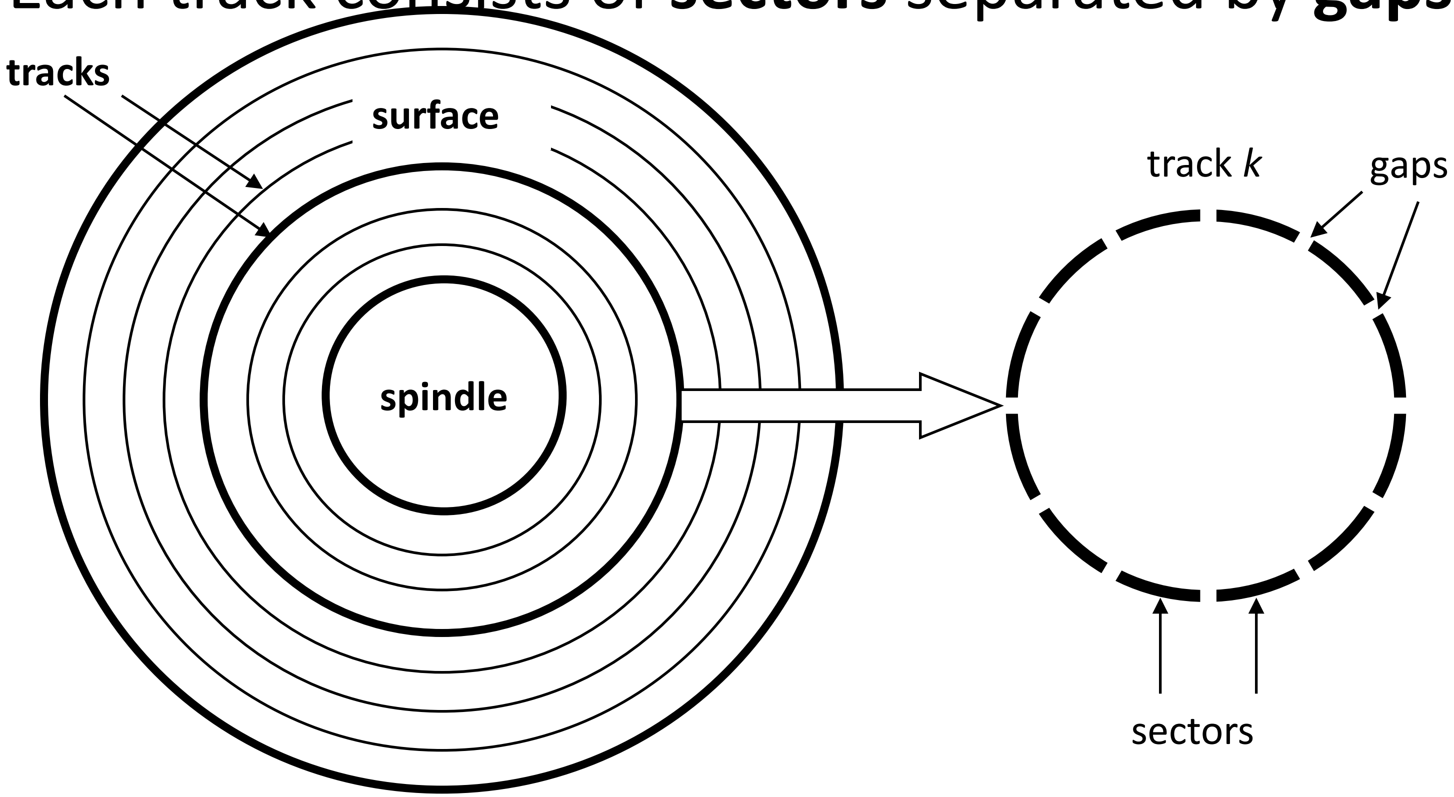
- **Programmed I/O :** In programmed I/O, the **data transfer is accomplished through an I/O port** and are controlled by software.
- **Interrupt driven I/O :** In interrupt driven I/O, the **I/O device will interrupt the processor**, and initiate data transfer.
- **Direct memory access (DMA) :** In DMA, the data transfer between memory and I/O can be performed by **bypassing the microprocessor**.

Abstracting Device differences

- Many devices have common characteristics; *e.g., different brands of disk or printer*
- Makes sense to abstract common parts
- Resulting structure is uniform driver sitting above specific one

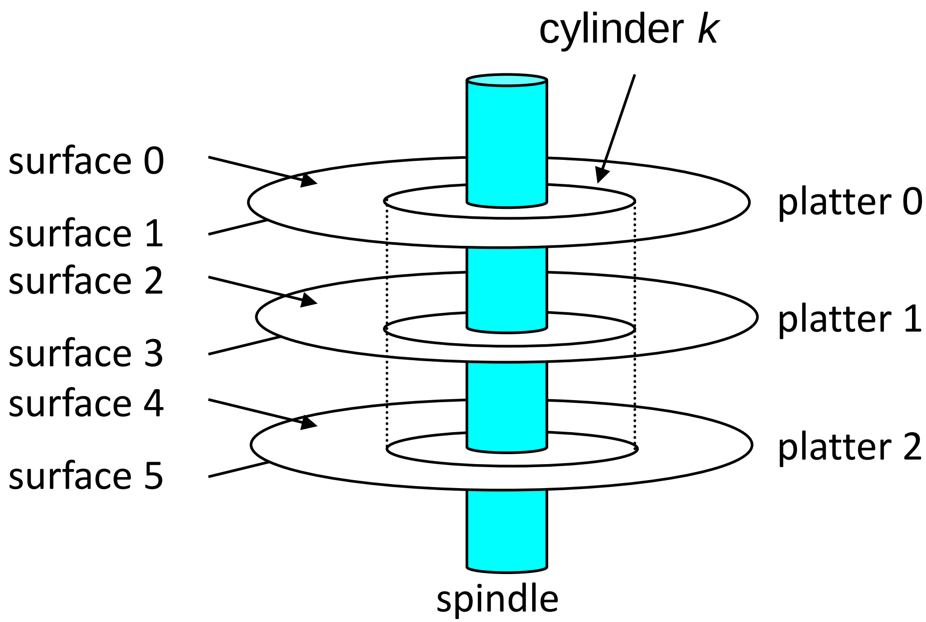
Disk Geometry

- Disks consist of **platters**, each with two **surfaces**.
- Each surface consists of concentric rings called **tracks**.
- Each track consists of **sectors** separated by **gaps**.



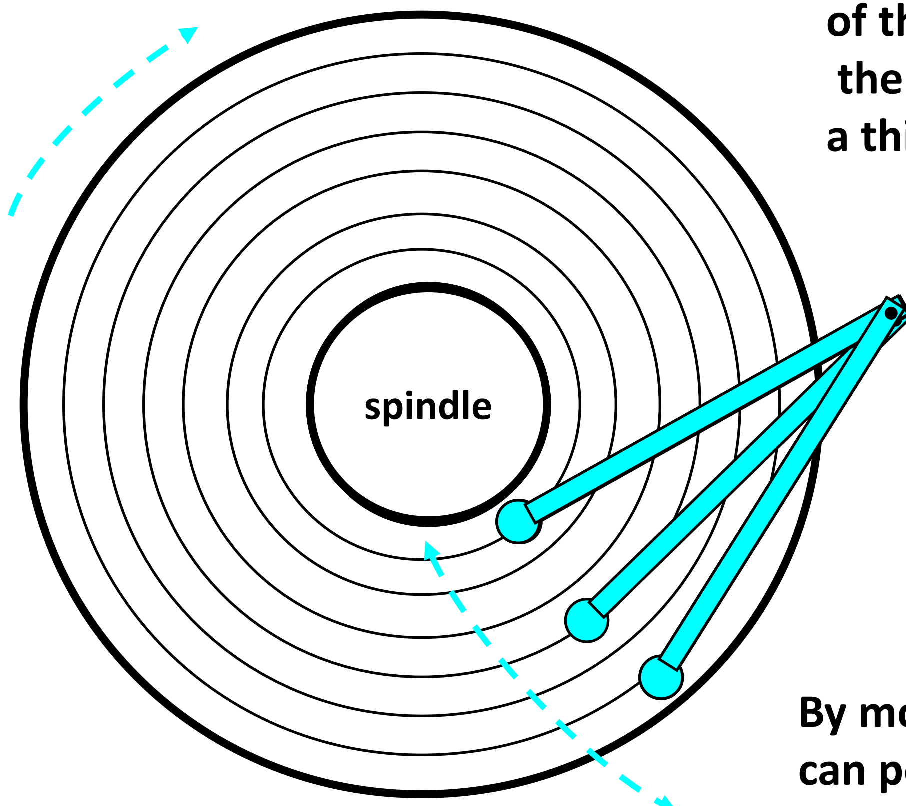
Disk Geometry (Muliple-Platter View)

- Aligned tracks form a cylinder.



Disk Operation (Single-Platter View)

The disk surface spins at a fixed rotational rate



The read/write *head* is attached to the end of the *arm* and flies over the disk surface on a thin cushion of air.

By moving radially, the arm can position the read/write head over any track.

Optical Disks

- Most commonly used optical disks are
  - **CD-ROM**
  - **CD-R**
  - **CD-RW**
  - **DVD- ROM**
  - **DVD-R**
  - **DVD-RW**

CD-ROM (Compact Disk- Read Only Memory)

- On this optical disk, **once data is imprinted, the user cannot erase it, change it or write on the disk.**
- The user can only "**read**" the data.
- This type of optical disk is used primarily for making huge amounts of prerecorded data, such as
  - operating systems, government statistics, Encyclopedias, medical reference books, dictionaries and legal libraries.

- **CD-R**
  - One **can add extra data** on these type of disks, if there is free space left.
  - But, once the data have been written on, however, they can only be read from them on, no changes can be made on the data.
- **CD-RW**
  - In contrast to CD –ROM and CD-R disks, CD-RW disks **can be written on and erased more than one time.**

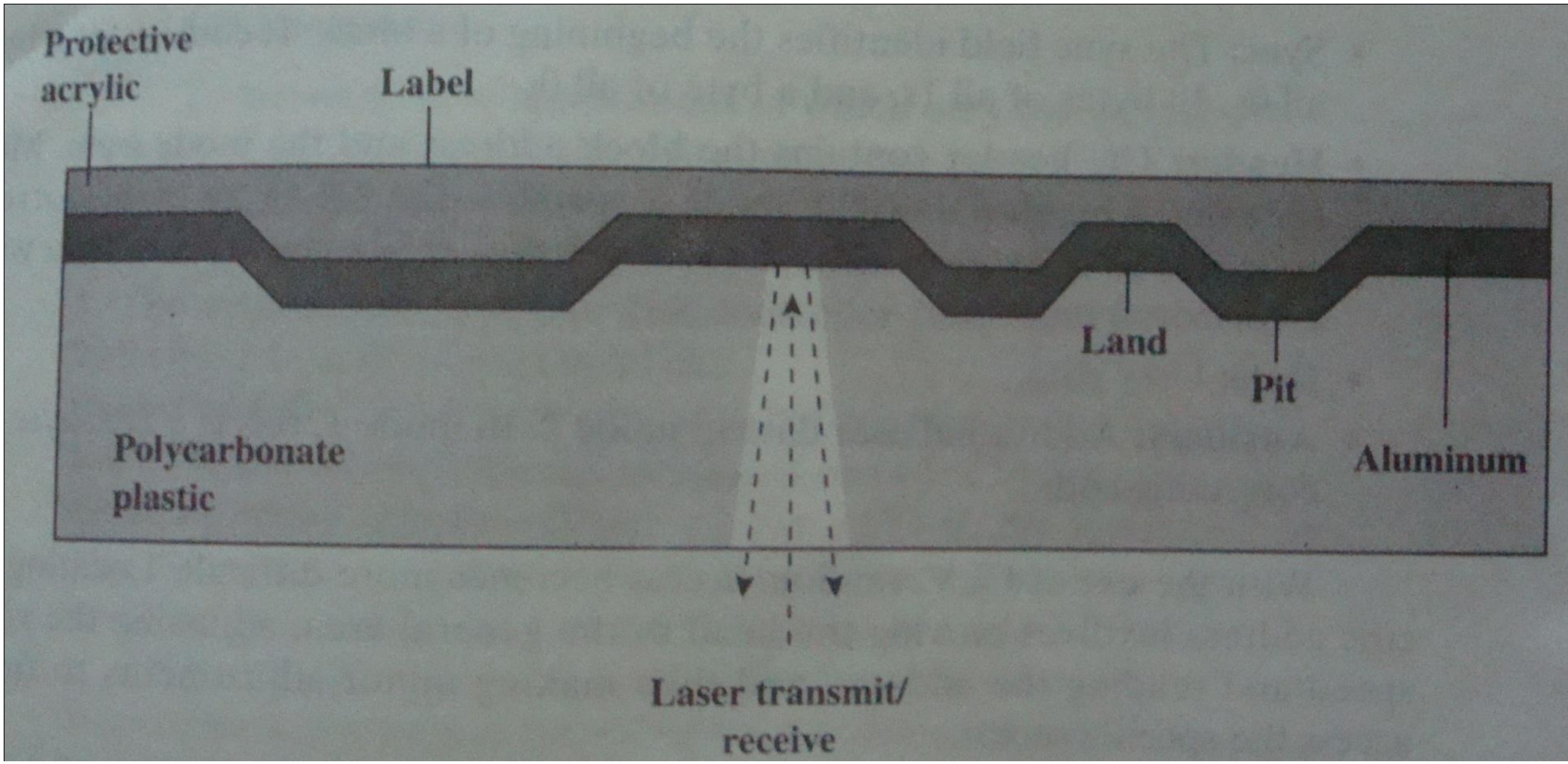
- **DVD**
  - Stands for **Digital Video Display**.
  - It holds huge data as compared with CDs. It has **4.7 GB up to 17 GB storage capacity**.
  - Nowadays, in addition to recordable, it comes with rewritable form.

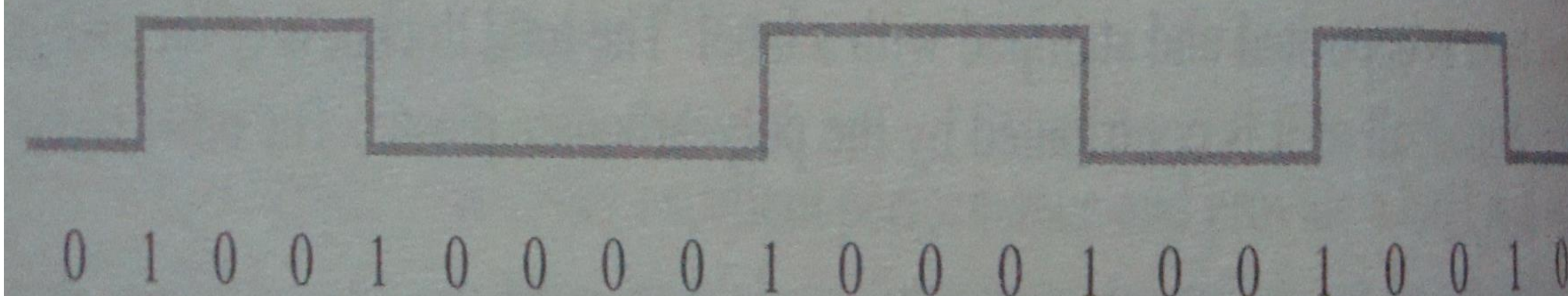
- The DVD’s greater capacity is due to 3 differences from CDs. They are

1. **Bits are packed more closely** on a DVD.
2. DVD **employs a second layer** of pits and lands on top of the first layer.
3. The **DVD ROM can be two sided** ; where as the data is recorded on only one side of a CD-ROM. this brings total capacity up to 17GB.

CD Operations

- Information from a CD-ROM is retrieved by a low-powered laser beam housed in an optical disk player or drive unit.
- The binary data recorded on the surface of a CD-ROM is in the form of pits and lands
- If the laser beam falls on a **pit** which has somewhat a rough surface the light scatters and a **low intensity** is returned back to the source. The areas between two pits are called **lands**, and is a smooth surface, which reflects back at **high intensity**.
- The change between **pits and lands** is detected by a photosensor and converted into a **digital signal**.
- **The beginning or end of a pit represents a '1' and when no change in elevation occurs between intervals a '0' is recorded**.


Protective
acrylic
Label
Land
Pit
Aluminum
Polycarbonate
plastic
Laser transmit/
receive



(c) Stored binary pattern

Error handling

- Disk manufacturers are constantly pushing the limits of the technology by increasing linear bit densities.
- This requires an extremely uniform substrate and a very fine oxide coating.
- Unfortunately, it is not possible to manufacture a disk to such specifications without defects.
- Manufacturing defects introduce **bad sectors**, that is, sectors that do not correctly read back the value just written to them.
- There are two general approaches to bad blocks: deal with them in **controller** or **operating system**.

Using the Controller

- Before the disk is shipped from the factory, it is tested and a **list of bad sectors** is written on the disk. For each bad sector, one of the spares is substituted for it.
- There are two ways to do substitution:
  - Substituting a spare for the bad sector.
  - Shifting all the sectors to bypass the bad one.
- The controller has to **keep track** of the information of each sector through **internal tables.**

Using the Operating system

- Operating system must acquire a list of bad sectors.
- Once it knows which sectors are bad, it can build **remapping tables**.
- It must **shif**t all the data starting from the bad sector up to the spare sector by one.

Stable storage

- Disks sometimes make errors, good sectors suddenly become bad, and whole drives die unexpectedly.
- RAIDS protects against a few sectors going bad but do not protect against write errors laying down bad data. They do not also protect against crashes.
- It is essential that the data never be lost or corrupted, the disk should work all the time with no error. Unfortunately, that is not achievable.
- What is achievable is a disk subsystem that has the following property: when a write issued to it, the disk either **correctly writes the data** or **it does nothing**, leaving the existing data intact. Such a system is called **stable storage**.

- Stable storage uses a pair of identical disks with the corresponding blocks working together to form one error-free block.
- In the absence of errors, the corresponding blocks on both drives are the same.
- Either one can be read to get the same result.
- To achieve this goal, the following three operations are defined:
  - Stable writes
  - Stable reads
  - Crash recovery

Stable writes

- A stable write consists of first writing the block on drive 1, then reading it back to verify that it was written correctly.
- If it was NOT written correctly, the write and reread are done again up to n times until they work.
- After n consecutive failures, the block is remapped onto a spare and the operation repeated until it succeeds.