Unit 4

Device Management

I/O System

- I/O system is a part of operating system which is responsible for controlling and managing all the types of input and output operations performed by I/O devices.

- The I/O system is needed because various I/O devices that are connected to computer differ in their functionality, data rate, software support and speed. In order to manage these peripherals the kernel needs a separate subsystem.
- The I/O system acts as an interface between end users and I/O devices. It hides all the device specific complexities from end users and provides a uniform interface to the outer layer of the operating system and to the programs.

I/O communication Techniques

1. Programmed I/O :

- The processor issues an I/O command, on behalf of a process.
- This I/O is issued to an I/O module.
- The execution of process starts only after the completion of I/O operation.

2. Interrupt Driver I/O :

- The processor issues an I/O command on behalf of a process.
- There are then two possibilities :
  1. If the I/O instruction from the process is non-blocking, then the processor continues to execute instructions from the process that issued the I/O command.
  2. If the I/O instruction is blocking, then the next instruction that the processor executes is from the OS, which will put the current process in a blocked state and schedule another process.

3. Direct Memory Access (DMA) :

- DMA module controls the transfer the data between main memory and I/O system without processor intervention.
- The processor sends a request for the transfer of data to the DMA module.
- The processor is interrupted only after the entire block of data has been transferred.
- DMA executes I/O operation without the help of processor the processor is free to do some other work.

I/O Hardware and I/O Controller

- In order to manage and control the various I/O device attached to computer, I/O system requires some hardware and software components.
- I/O devices commonly use certain hardware devices. These are system bus and ports.
- **Ports** are the plugs used to connect I/O devices to the computer.
- **Bus** is a set of wires to which these ports and I/O controllers are connected and through which signals are send for I/O command.
- **I/O controller** is a component that attaches with each device and is used to accept input and provide output to these devices. Applications access I/O devices with the help of these I/O controllers.

- I/O controllers have certain registers to store data and control signals. These registers are :
  1. Status registers
  2. Control registers
  3. Input registers
  4. Output register
  5. Program Counter

Interrupts

- Whenever a process needs to perform I/O operation it can use an interrupt. Interrupts stop the execution of a program to perform other tasks, such as numerical computation.
- Interrupt signal an event to occur. If an interrupt occurs, the CPU stores the current status of the process in the program registers and stop the program execution.
- CPU starts executing the interrupt. When CPU finishes its processing it regains the status of process and continue its execution.

Types of Interrupts

Types of interrupts :

1. **Program Interrupt.** Generated by a program when one of its statements causes error, such as division by zero, data mismatch and arithmetic overflow.
2. **Time Interrupt.** Generated by the inbuilt timer in a processor to perform certain functions that are required at regular time intervals.

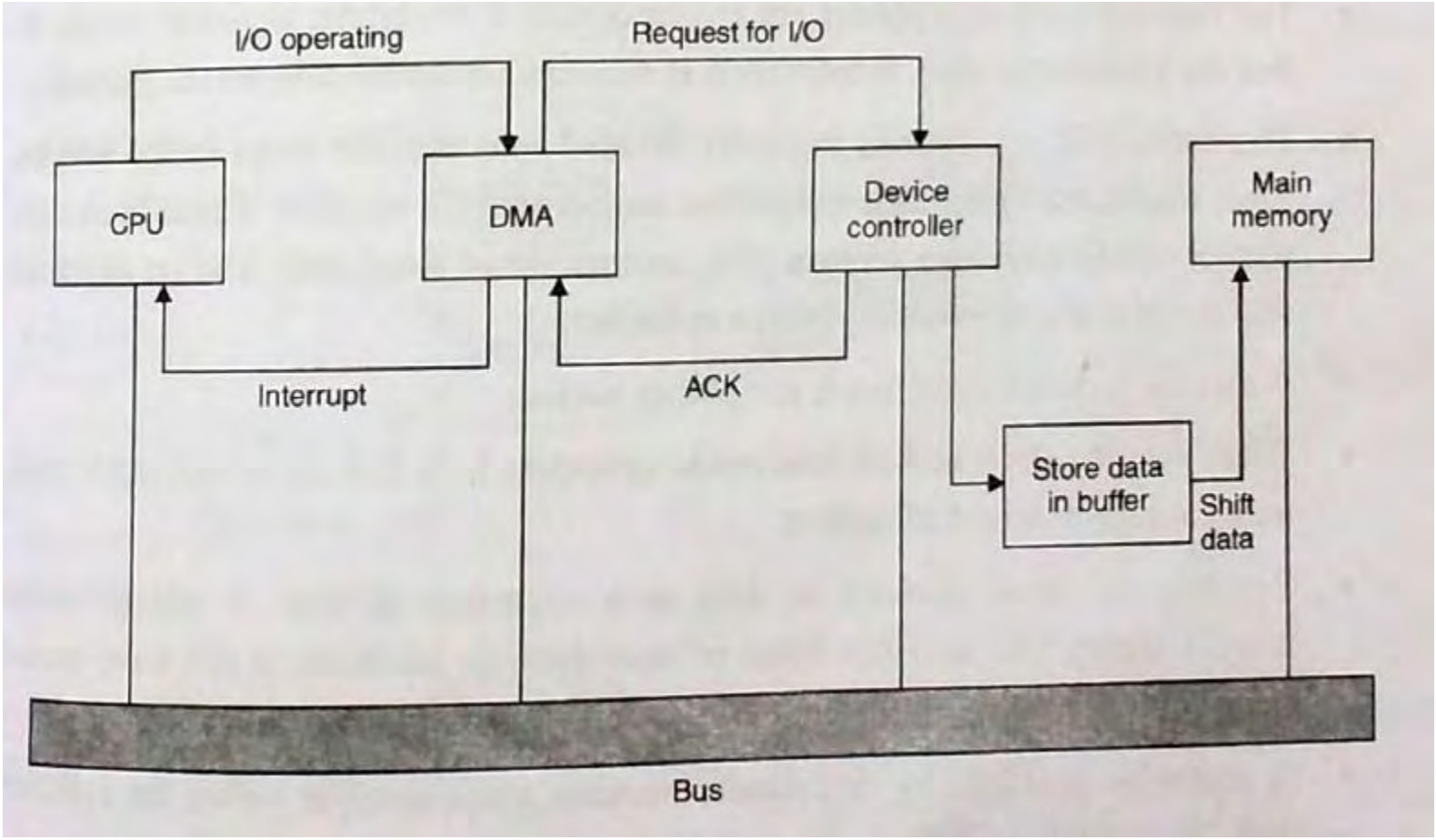
3. **I/O Interrupts.** Generated by I/O hardware or by the controller to inform an end user about the completion of an I/O operation or about errors, if any, that occurred at the time of execution of a program.
4. **Hardware Failure Interrupt.** Generated by the hardware if any problem occurs, such as parity error or power failure.
5. **Supervisor Call (SVC) Interrupts.** Generated by the process itself at the time of execution, if it needs an I/O operation to be performed.
6. **Restart Interrupts.** Generated when the restart button of the console is pressed.

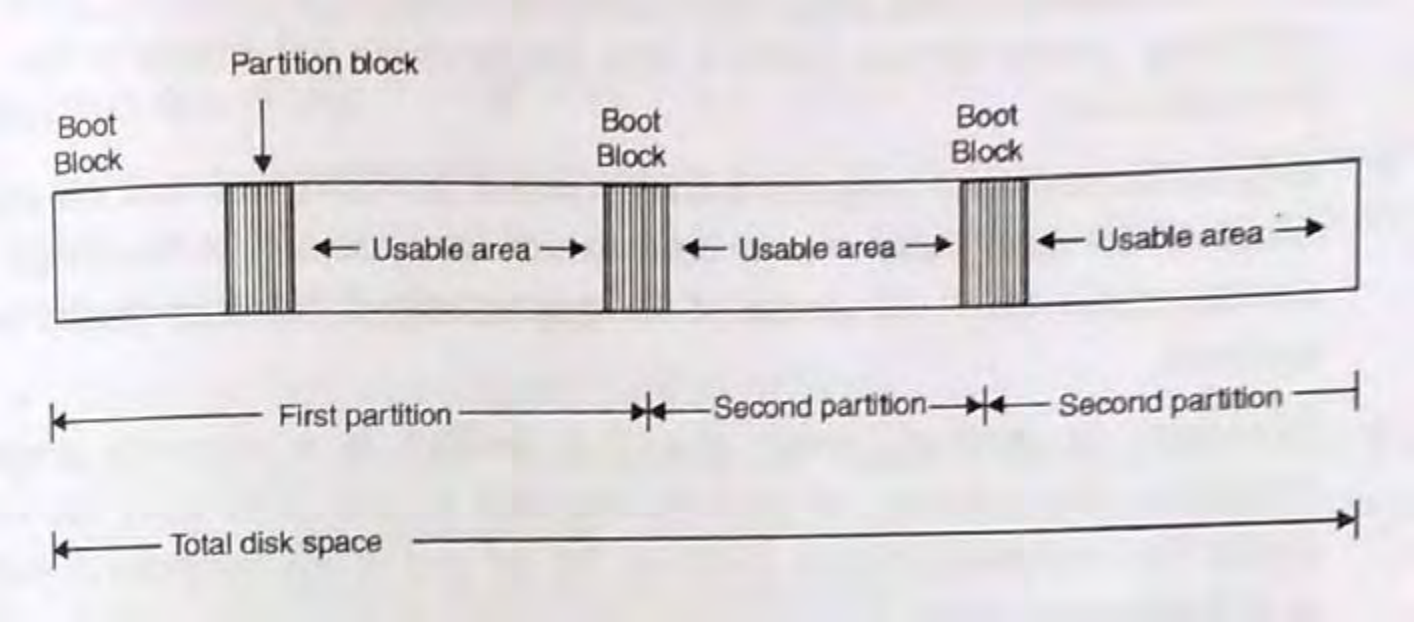
DMA

- In DMA, I/O system can transfer block of data directly to main memory without CPU's intervention.
- In such a case DMA, uses four basic things to perform its functions :
  1. Type of command *i.e.* read or write.
  2. Address of I/O device that needs to perform I/O.
  3. Location of the memory *i.e.* where data is read or written to.
  4. Total number of words read from or written to a file.
- Whenever any process needs to perform I/O the CPU issues a command with the above listed four information to DMA module.
  5. DMA module acquires the control of bus from CPU for transferring data, if CPU needs the bus it has to wait for the completion of DMA work.
  6. DMA sends a read request, with the memory address of the location

where data is to write through bus to the device controller for performing read request.

7. The device controller after receiving this request, sends the data directly to the desired memory location and acknowledge the DMA controller.
8. DMA interrupts the CPU about the process completion.
9. In such a way CPU is only interrupted twice in the whole process. DMA then returns the control of bus to the CPU.
10. For the next word, the DMA again get the hold of bus and start transferring the word, this process continues until the whole block is transferred. This is called **cycle stealing** because DMA gets the control of the bus forcefully from CPU for a short period of time.
11. The working of DMA is shown in figure 10.1


I/O operating
Request for I/O
CPU
DMA
Device controller
Main memory
Interrupt
ACK
Store data in buffer
Shift data
Bus


Partition block
Boot Block
Boot Block
Boot Block
Usable area
Usable area
Usable area
First partition
Second partition
Second partition
Total disk space

Booting From Disk

- In order to start the computer, the special program is required. This program is called *bootstrap*.
- This program initializes all the hardware component, like CPU registers device controllers and starts the operating system. For this it finds the operating system kernel, loads it into the memory and jumps to the first instruction in it.
- Originally, bootstrap program entirely resided in ROM and was not a disk concern. In case, there was a need to add new device or to change the characteristics of an old device of the system, only ROM chip needed to be replaced.
- However, in modern world, changing devices is a common practice. Therefore, the solution for modern computer system is to store the small loader for bootstrap program in ROM. All the rest of this program is written to a dedicated area on the hard disk. This area is known as *boot sector*. This area can be few dedicated sectors or the whole dedicated partition. In such a case, if a disk has a boot partition, it is called boot disk or system disk.
- Then, to start the computer, the loader in ROM is started. This loader find the boot sector on the disk, loads bootstrap program from the boot sector into memory and then transfers the control to the bootstrap program (now loaded in memory), which inturn does the initialization job. This procedure is called booting from the disk.

Bad Sector Recovery

- When one or more sectors on a disk become defective their data is lost, they are called **bad blocks.**
- In such a case special applications are required to recover data from bad sectors or to restore them from their backup copy.
- The bad blocks found are marked as bad through the file allocation methods. If the blocks are not marked as bad, the read/write errors may occur because of bad block reference.
- The various methods are used to handle bad blocks on the disk. These are :

Manual Handling of Bad Sectors

- Bad blocks on some disks with IDE (Integrated Drive Electronics) controllers can be handled manually.

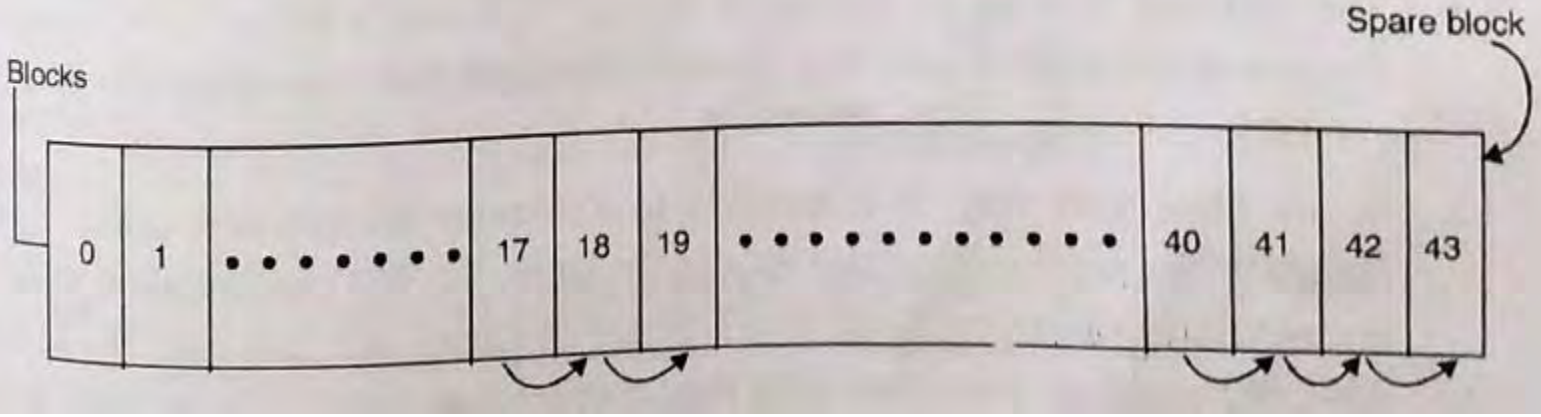
- This approach make use of certain command to locate bad blocks and to mark them as bad.
- For example, in MS-DOS, *format* command is used to find bad blocks and mark/flag them as bad in FAT so that they are not allocated at any file. Also *chkdsk* comment is also used to manually search for bad blocks and to lock them.

Sector Sparing

- In this method, a list of bad sectors and a list of spare sectors is maintained. Spare sectors are the sectors that are not use for general allocation and are set aside by low-level formatting.
- These two lists are updated from time to time.
- Whenever a controller encounters a bad sector, it replaces each bad sector with one of the spare sectors. Thus each block from the list of bad blocks is mapped to the block from the list of spare block.
- In such a way, whenever a bad sector is addressed, the request is translated to the corresponding spare sector by the controller.
- This technique is called **sector sparing** or sector formatting.
- This technique of sector sparing is used by more sophisticated disks such as the SCSI (Small Computer System Interface) disks used in high-end PCs and most workstations and servers.

Sector Slipping

- Some controllers replaces the bad sector or block by shifting one sector forward. This technique is called **sector slipping.**



I/O Scheduler

- The I/O scheduler implements policy algorithm used to allocate channel control unit and device access to I/O requests from jobs. Thus it works like process scheduler.
- If there are more I/O requests pending then available paths, it is necessary to choose which I/O requests to satisfy first.
- Unlike processes, I/O requests are not preempted *i.e.* once channel program has started it is allowed to continue towards competition. Also, I/O requests are not time-sliced.
- The I/O requests are also not preempted, even when higher priority I/O requests enters the queue. This is feasible because most channel programs are relatively short (50 to 100 ms)
- Many different policies may be used for the I/O schedule. For example, if a process scheduler has assigned a high priority to a process, it to reasonable also to assign high priority to its I/O requests of this process. This will help complete that program as fast as possible.
- It is important that I/O scheduler should synchronize its work with I/O traffic controller to make sure that a path is available to satisfy I/O requests.