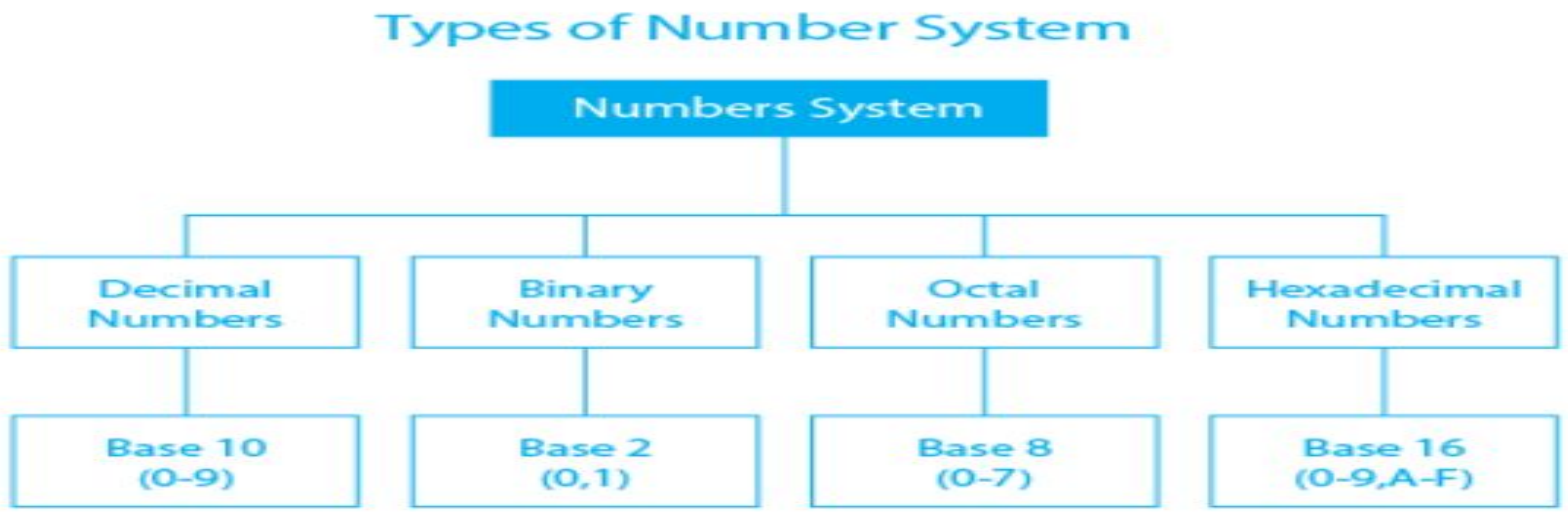
MODULE I

FUNDAMENTALS OF COMPUTER SYSTEMS

NUMBER SYSTEM

- A number system is defined as a system of writing to express numbers.
- All numbers can be formed using the digits 0 to 9. Anyone can generate an infinite number using these digits.
- The most commonly used number system is the **binary system**.
- The binary system uses only two digits, **0 and 1,** to represent all numbers and data.
- Computer programs and programming languages use number system concepts to perform **arithmetic operations, store data, and communicate with hardware devices.**

TYPES OF NUMBER SYSTEMS



Decimal Number System:

- It uses ten digits**, 0 through 9**, to represent numbers.
- **Example of Decimal Number System**:

The decimal number 1457 consists of the digit 7 in the units position, 5 in the tens place, 4 in the hundreds position, and 1 in the thousands place whose value can be written as:

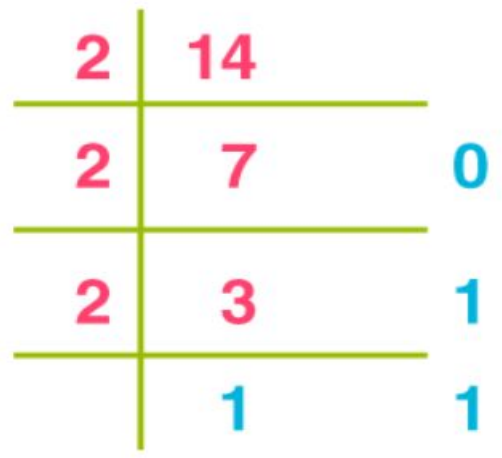
- (1×1000) + (4×100) + (5×10) + (7×1)
- 1000 + 400 + 50 + 7
- 1457

- **Binary Number System (Base 2 Number System)**
- The base 2 number system is also known as the Binary number system wherein, only two binary digits exist, i.e., 0 and 1.
- For example, 110101 is a binary number.
- We can convert any system into binary and vice versa.

Example

- Write $(14)_{10}$ as a binary number.

$(14)_{10} = 1110_2$



- **Octal Number System (Base 8 Number System)**
- In the octal number system, the base is 8 and it uses numbers from 0 to 7 to represent numbers.
- Octal numbers are commonly used in computer applications.

Example: Convert 215_8 into decimal.

Solution:

$215_8 = 2 \times 8^2 + 1 \times 8^1 + 5 \times 8^0$

$= 2 \times 64 + 1 \times 8 + 5 \times 1$

$= 128 + 8 + 5$

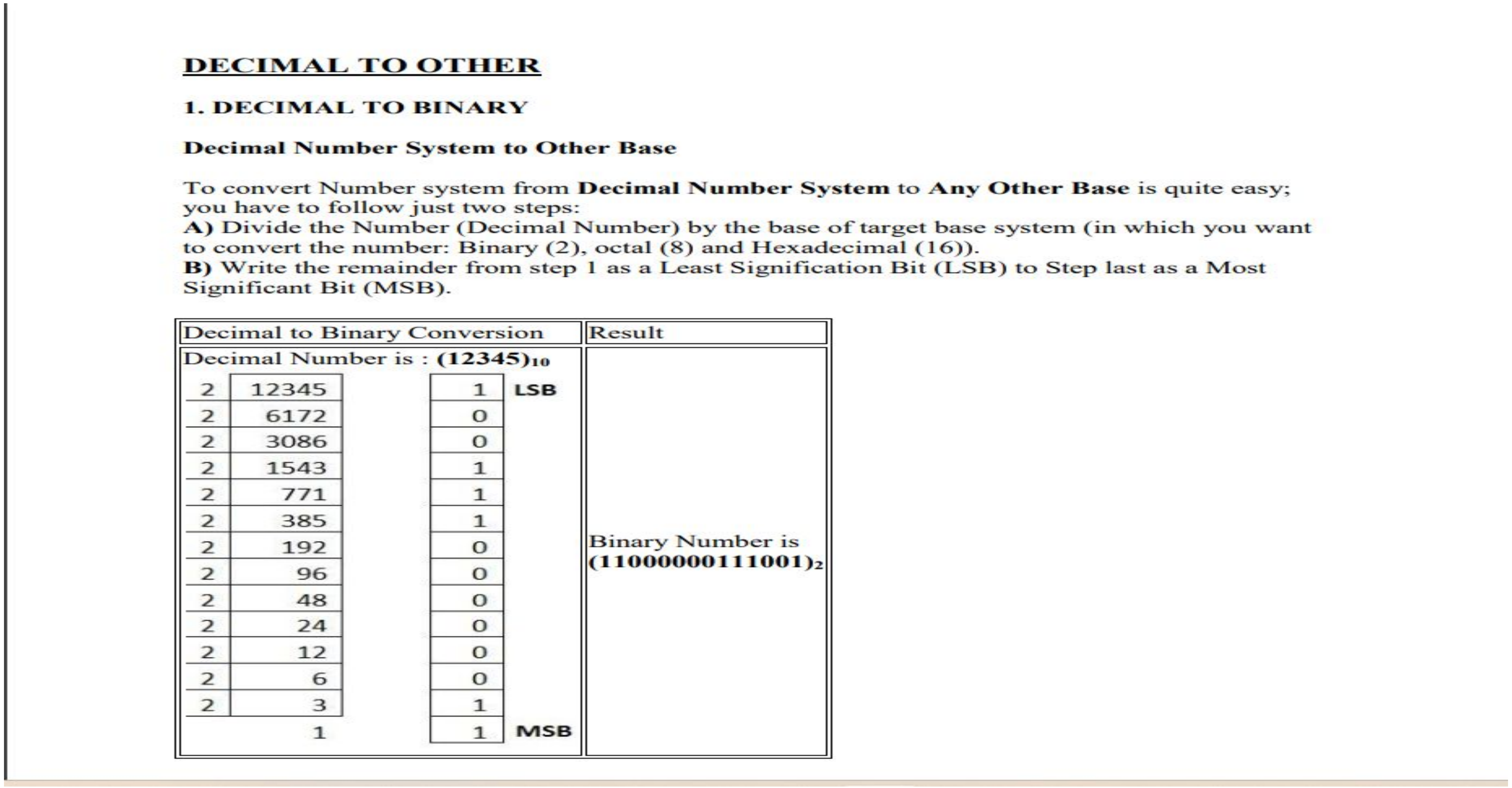
$= 141_{10}$

Hexadecimal Number System: The hexadecimal number system uses **16 digits, including 0 through 9 and A through F, to represent numbers.**

Each of these number systems has its advantages and disadvantages, and each is used in different applications.

- Binary number system is used to **represent and manipulate data in computer hardware.**
- Decimal number system is used for **everyday calculations.**
- The octal and hexadecimal systems are commonly used in **computer programming and digital electronics** because they are more compact and easier to read than binary.

NUMBER SYSTEM CONVERSIONS



DECIMAL TO OTHER

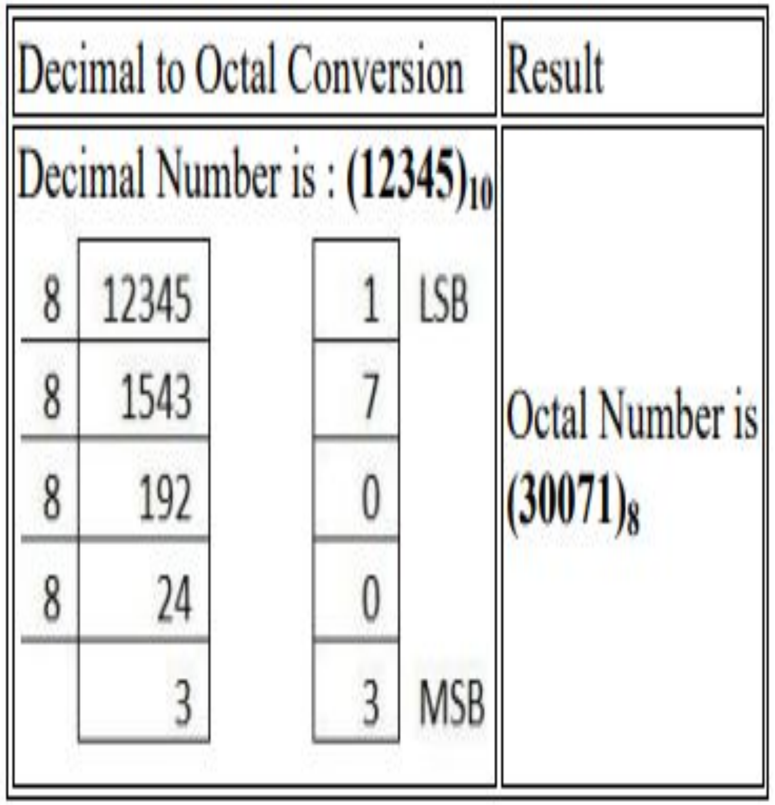
1. DECIMAL TO BINARY

Decimal Number System to Other Base

To convert Number system from **Decimal Number System** to **Any Other Base** is quite easy; you have to follow just two steps:
A) Divide the Number (Decimal Number) by the base of target base system (in which you want to convert the number: Binary (2), octal (8) and Hexadecimal (16)).
B) Write the remainder from step 1 as a Least Signification Bit (LSB) to Step last as a Most Significant Bit (MSB).

Decimal to Binary Conversion				Result
Decimal Number is : **$(12345)_{10}$**				Binary Number is **$(11000000111001)_2$**
2	12345	1	**LSB**	
2	6172	0		
2	3086	0		
2	1543	1		
2	771	1		
2	385	1		
2	192	0		
2	96	0		
2	48	0		
2	24	0		
2	12	0		
2	6	0		
2	3	1		
	1	1	**MSB**	

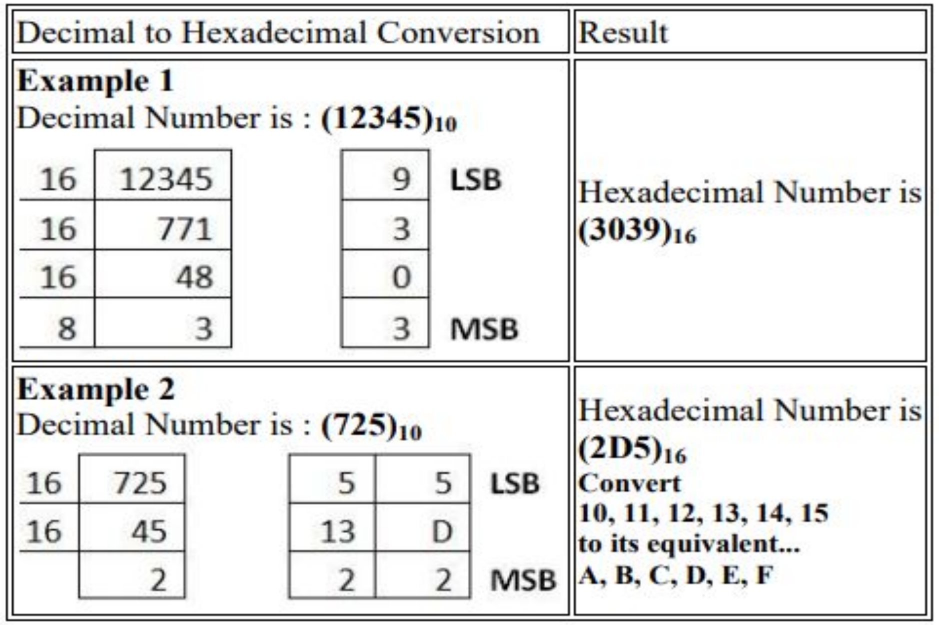
2. DECIMAL TO OCTAL



Decimal to Octal Conversion	Result
Decimal Number is : $(12345)_{10}$	Octal Number is $(30071)_8$

8	12345	1	LSB
8	1543	7	
8	192	0	
8	24	0	
	3	3	MSB

3. DECIMAL TO HEXADECIMAL



Decimal to Hexadecimal Conversion	Result
Example 1 Decimal Number is : $(12345)_{10}$	Hexadecimal Number is $(3039)_{16}$
Example 2 Decimal Number is : $(725)_{10}$	Hexadecimal Number is $(2D5)_{16}$ **Convert 10, 11, 12, 13, 14, 15 to its equivalent... A, B, C, D, E, F**

Example 1

16	12345	9	**LSB**
16	771	3	
16	48	0	
8	3	3	**MSB**

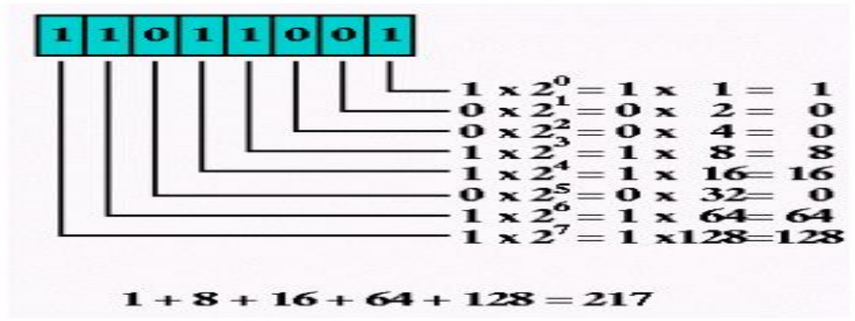
Example 2

16	725	5	5	**LSB**
16	45	13	D	
	2	2	2	**MSB**

BINARY TO OTHER

A) Multiply the digit with 2(with place value exponent). Eventually add all the multiplication becomes the Decimal number.

1. BINARY TO DECIMAL



1 1 0 1 1 0 0 1
$1 \times 2^0 = 1 \times 1 = 1$
$0 \times 2^1 = 0 \times 2 = 0$
$0 \times 2^2 = 0 \times 4 = 0$
$1 \times 2^3 = 1 \times 8 = 8$
$1 \times 2^4 = 1 \times 16 = 16$
$0 \times 2^5 = 0 \times 32 = 0$
$1 \times 2^6 = 1 \times 64 = 64$
$1 \times 2^7 = 1 \times 128 = 128$
$1 + 8 + 16 + 64 + 128 = 217$

2. BINARY TO OCTAL

An easy way to convert from binary to octal is to group binary digits into sets of three, starting with the least significant (rightmost) digits.

Binary: 11100101 =	11 100 101	
	011 100 101	Pad the most significant digits with zeros if necessary to complete a group of three.

Then, look up each group in a table:

Binary:	000	001	010	011	100	101	110	111

Octal:	0	1	2	3	4	5	6	7

Binary =	011	100	101	
Octal =	3	4	5	= 345 oct

3. BINARY TO HEXADECIMAL

An equally easy way to convert from binary to hexadecimal is to group binary digits into sets of four, starting with the least significant (rightmost) digits.

Binary: 11100101 = 1110   0101

Then, look up each group in a table:

Binary:	0000	0001	0010	0011	0100	0101	0110	0111
Hexadecimal:	0	1	2	3	4	5	6	7

Binary:	1000	1001	1010	1011	1100	1101	1110	1111
Hexadecimal:	8	9	A	B	C	D	E	F

Binary =	1110	0101	
Hexadecimal =	E	5	= E5 hex

OCTAL TO OTHER

1. OCTAL TO BINARY

Converting from octal to binary is as easy as converting from binary to octal. Simply look up each octal digit to obtain the equivalent group of three binary digits.

Octal:	0	1	2	3	4	5	6	7
Binary:	000	001	010	011	100	101	110	111

Octal =	3	4	5	
Binary =	011	100	101	= 011100101 binary

2. OCTAL TO HEXADECIMAL

When converting from octal to hexadecimal, it is often easier to first convert the octal number into binary and then from binary into hexadecimal. For example, to convert 345 octal into hex:

(from the previous example)

Octal =	3	4	5	
Binary =	011	100	101	= 011100101 binary

2. OCTAL TO HEXADECIMAL

When converting from octal to hexadecimal, it is often easier to first convert the octal number into binary and then from binary into hexadecimal. For example, to convert 345 octal into hex:

(from the previous example)

Octal =	3	4	5	
Binary =	011	100	101	= 011100101 binary

Drop any leading zeros or pad with leading zeros to get groups of four binary digits (bits):
Binary 011100101 = 1110 0101

Then, look up the groups in a table to convert to hexadecimal digits.

Binary:	0000	0001	0010	0011	0100	0101	0110	0111
Hexadecimal:	0	1	2	3	4	5	6	7

Binary:	1000	1001	1010	1011	1100	1101	1110	1111
Hexadecimal:	8	9	A	B	C	D	E	F

Binary =	1110	0101	
Hexadecimal =	E	5	= E5 hex

Therefore, through a two-step conversion process, octal 345 equals binary 011100101 equals hexadecimal E5.

3. OCTAL TO DECIMAL

The conversion can also be performed in the conventional mathematical way, by showing each digit place as an increasing power of 8.

345 octal = $(3 * 8^2) + (4 * 8^1) + (5 * 8^0) = (3 * 64) + (4 * 8) + (5 * 1) = 229$ decimal

OR

Converting octal to decimal can be done with repeated division.

1. Start the decimal result at 0.
2. Remove the most significant octal digit (leftmost) and add it to the result.
3. If all octal digits have been removed, you're done. Stop.
4. Otherwise, multiply the result by 8.
5. Go to step 2.

Octal Digits	Operation	Decimal Result	Operation	Decimal Result
345	+3	3	× 8	24
45	+4	28	× 8	224
5	+5	229	done.	

⇨ $(345)_8 = (229)_{10}$

HEXADECIMAL TO OTHER

1. HEXADECIMAL TO BINARY

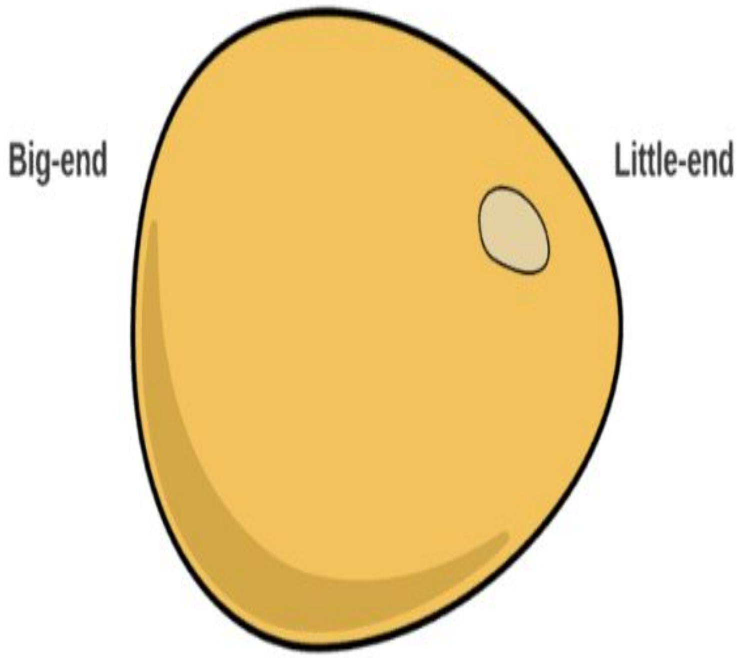
Converting from hexadecimal to binary is as easy as converting from binary to hexadecimal. Simply look up each hexadecimal digit to obtain the equivalent group of four binary digits.

Hexadecimal:	0	1	2	3	4	5	6	7
Binary:	0000	0001	0010	0011	0100	0101	0110	0111
Hexadecimal:	8	9	A	B	C	D	E	F
Binary:	1000	1001	1010	1011	1100	1101	1110	1111

Hexadecimal =	A	2	D	E	
Binary =	1010	0010	1101	1110	= 1010001011011110 binary

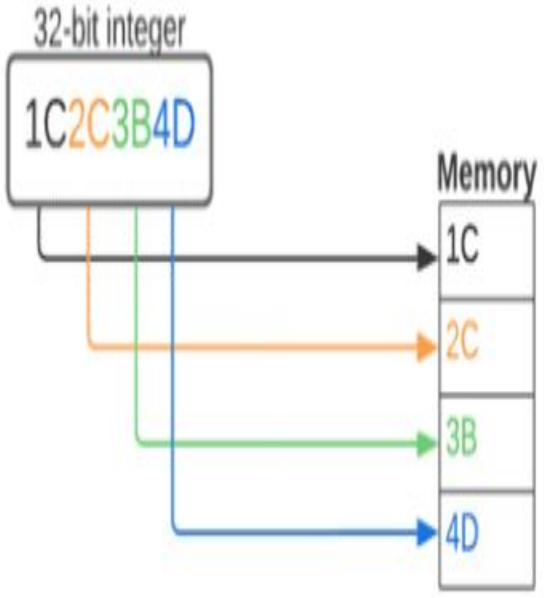
ENDIANESS

- Endianness refers to the **order in which bytes are arranged in memory**.
- Different languages read their text in different orders. for example, English reads from left to right, while Arabic reads from right to left.
- Endianness **helps ensure that computers read the bytes stored in memory in a specific order.**
- Two methods of reading bytes define endianness: **big endian and little endian**.
- In big-endian, the ‘big end’ of the byte is stored first.
- Conversely, Little Endian stores the little end of the byte first.

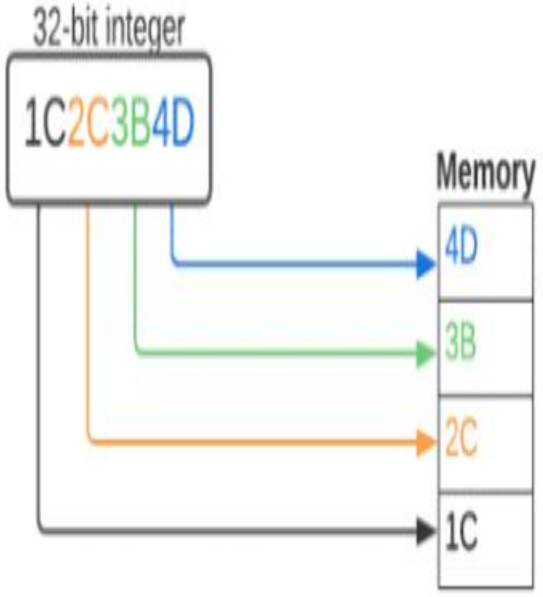


BIG-ENDIAN

- In a big-endian system, **the most significant byte (MSB) is stored at the lowest memory address.**
- This means the "big end" (the most significant part of the data) comes first.
- A big-endian computer would store the two bytes required for the hexadecimal number 4F52 as **4F52** in storage.
- For example, **if 4F is stored at storage address 1000, 52 will be at address 1001.**



Big-endian



Little-endian

LITTLE-ENDIAN

- **As we can see, in the case of big-endian, we locate the most significant byte of the 32-bit integer at the byte with the lowest address in the memory.**
- The rest of the data is located in order in the next three bytes in memory.
- On the other hand, when we look at the little-endian, in that case, **we locate the least significant byte of the data at the byte with the lowest address.**
- After that, we find the rest of the data in the order in the next three bytes in memory.

INTRODUCTION TO COMPUTER

- The term ‘Computer’ is derived from the Latin word ‘**Computare’ which means ‘to calculate’, ‘to count’, etc.**
- A computer can also be defined in terms of the functions it can perform. A computer can

i) accept data

ii) store data

iii) process data as desired

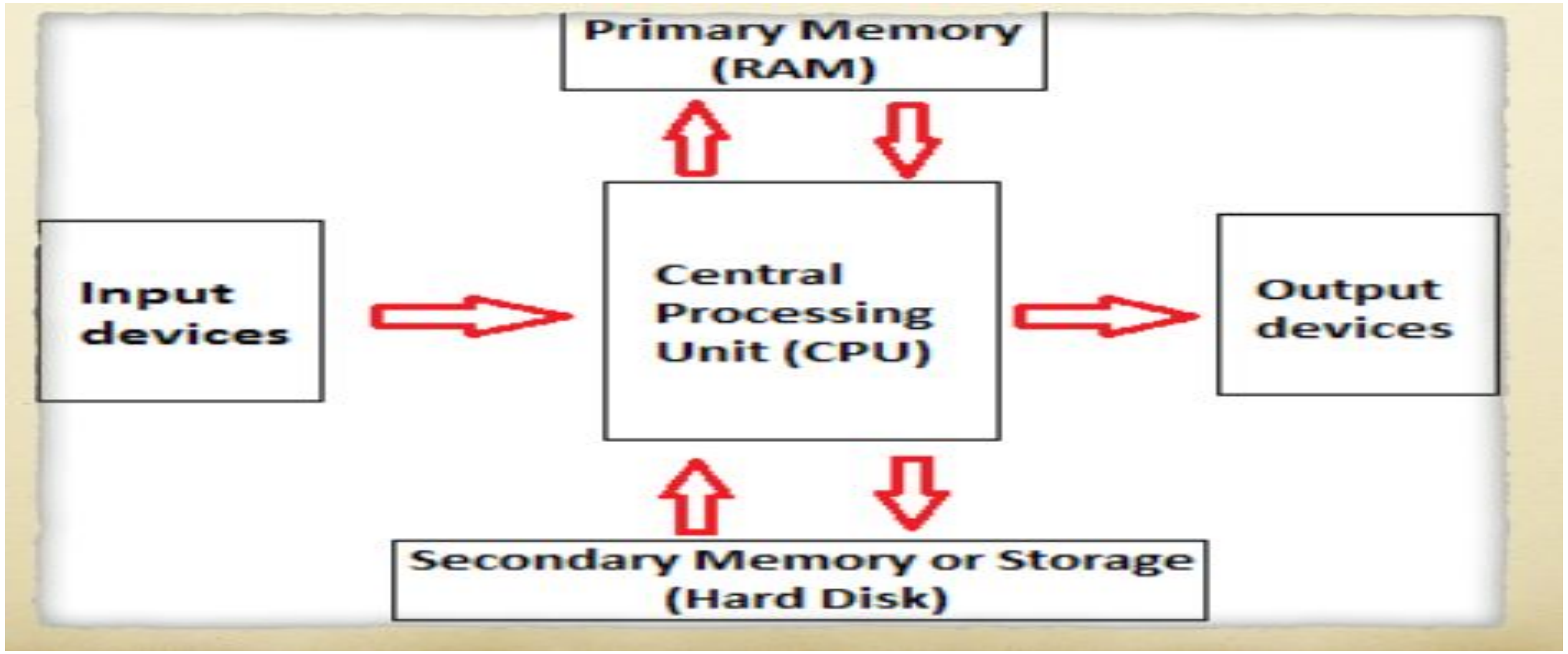
iv) retrieve the stored data as and when required and

v) print the result in desired format.

- **COMPUTER SYSTEM = HARDWARE + SOFTWARE+ USER**
- Hardware = Internal Devices + Peripheral Devices
- Software = Programs
- USER = Person, who operates computer.

BASIC COMPUTER MODEL

- The model of a computer can be described by four basic units:


Primary Memory (RAM)
Input devices
Central Processing Unit (CPU)
Output devices
Secondary Memory or Storage (Hard Disk)

- **Input Device:** The input device is used for entering information into memory. It enables the user to send data, information, or control signals to a computer. The Central Processing Unit (CPU) of a computer receives the input and processes it to produce the output.

- **Output Device:** The output device displays the result of the processing of raw data that is entered in the computer through an input device.

- **Memory:** The computer memory holds the data and instructions needed to process raw data and produce output.

- **The Central Processing Unit (CPU):** The CPU is used to perform computations on information.

CPU

- **“Brain” of the computer.**
- **Receives instructions from both the hardware and active software** and produces the output accordingly.
- Carries out the operations commanded by the instructions.

Two typical components of a CPU are:

- **The *control unit (CU)*,** which extracts instructions from memory and decodes and executes them, calling on the ALU when necessary.
- **The *arithmetic logic unit (ALU)*,** which performs arithmetic and logical operations.

ALU: The actual processing of the data and instruction are performed by Arithmetic Logical Unit.

- **The major operations performed by the ALU are addition, subtraction, multiplication, division, logic and comparison.**
- Data is transferred to ALU from storage unit when required.

Control Unit: The Control Unit **acts like the supervisor**. It extracts instructions from memory and decodes and executes them, and sends the necessary signals to the ALU to perform the operation needed.

- **The control unit determines the sequence in which computer programs and instructions are executed.**

MEMORY

- There are two kinds of computer memory: ***primary* and *secondary*.**
- Primary memory is accessible directly by the processing unit, hence, it can store and **retrieve data much faster.**
- Holds instructions and data needed for programs that are currently running.
- Primary Memory is of two types: **RAM and ROM**
- **RAM is usually a *volatile* type of memory.**
- Contents of RAM are lost when power is turned off.
- **ROM is a non-volatile memory that contains all important data and instructions needed to perform important tasks like the boot process.**

SECONDARY STORAGE

- A ***non-volatile* storage medium.**
- Contents are retained while power is off.
- The secondary storage devices built into the computer or connected to the computer.
- Also known as **external memory or auxiliary storage.**
- Examples are: HDD, SSD, Pen drive, CD, etc.

INPUT DEVICES

- Any type of device that provides **data to a computer** from the outside world.

For example:

- **–Keyboard**
- **–Mouse**
- **–Scanner**

OUTPUT DEVICES

- Any type of device that provides **data from a computer to** the outside world.

Examples of output data:

- **–A printed report**
- **–An image such as a picture**
- **–A sound**

Common output devices include:

- **–Monitor (display screen)**
- **–Printer**

SOFTWARE COMPONENTS

- **Software refers to programs, which are instruction codes that direct the computer to perform some actions.**
- In general, there are two basic types of software:
- **System software:** set of programs designed to **coordinate the activities and functions of the hardware and various programs** throughout the computer system.
- Three types of programs: **Operating System, Language Translators and Common Utility Programs**
- **Application software:** consists of programs that help users solve particular computing problems.

CPU

- **“Brain” of the computer.**
- Responsible for accepting data from input devices, processing the data into information, and transferring the information to memory and output devices.
- **The processor calculates (add, multiply and so on), performs logical operations (compares numbers and make decisions), and controls the transfer of data among devices.**
- The primary function is to execute sequences of instructions representing programs, which are stored in the Main Memory.

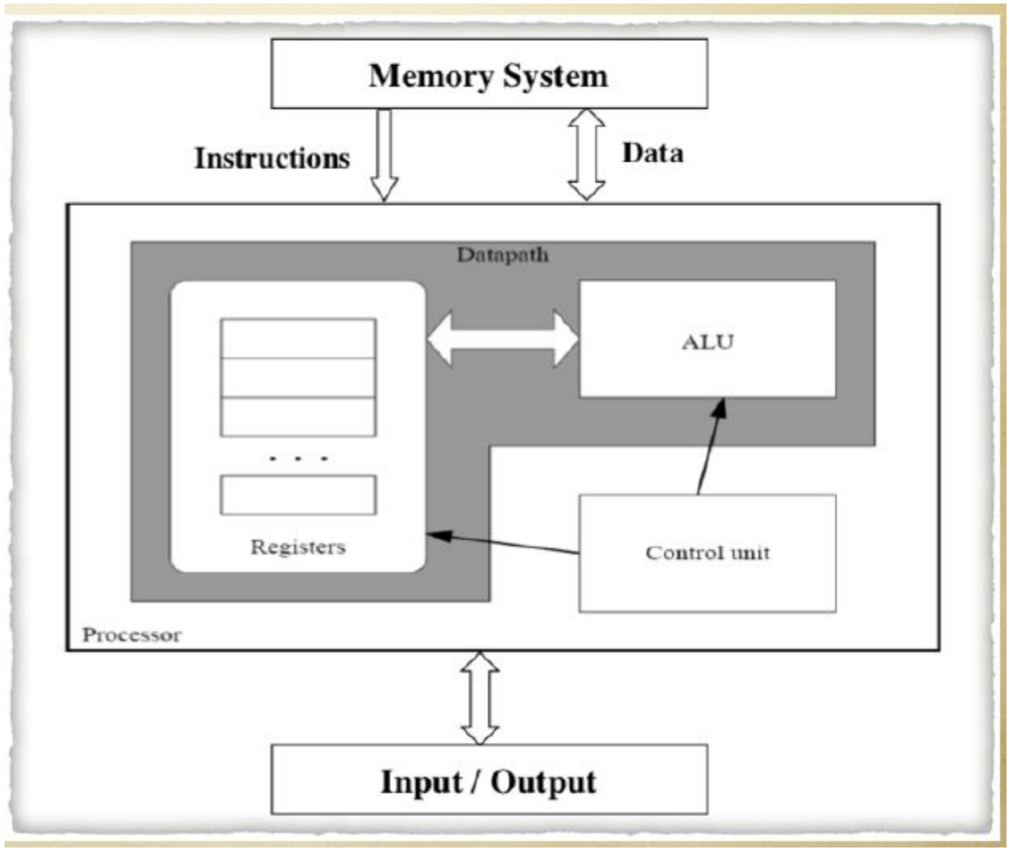
COMPONENTS OF CPU

- The CPU is organized into the following three major sections:

1. Arithmetic Logic Unit (ALU)

2. Control unit (CU)

3. Registers



ALU

- The function of the arithmetic logic unit (ALU) is to perform **arithmetic operations such as addition, subtraction, division, and multiplication and logic operations such as AND, OR, and NOT.**
- The ALU has the **status of inputs, outputs,** or both which convey the information about the previous operation or the current operation.
- The **inputs to the ALU are the data words to be operated on (called operands),** status information from previous operations, and **a code from the control unit indicating which operation to perform.**
- When all input signals have settled and propagated through the ALU circuitry, the result of the performed operation appears at the ALU's outputs.

CONTROL UNIT

- The Control Unit **instructs the computer system for executing already stored instructions in memory and then decodes and executes these instructions.**

- The function of the control unit is **to control input/output devices, generate control signals to the other components of the computer such as read and write signals, and perform instruction execution.**

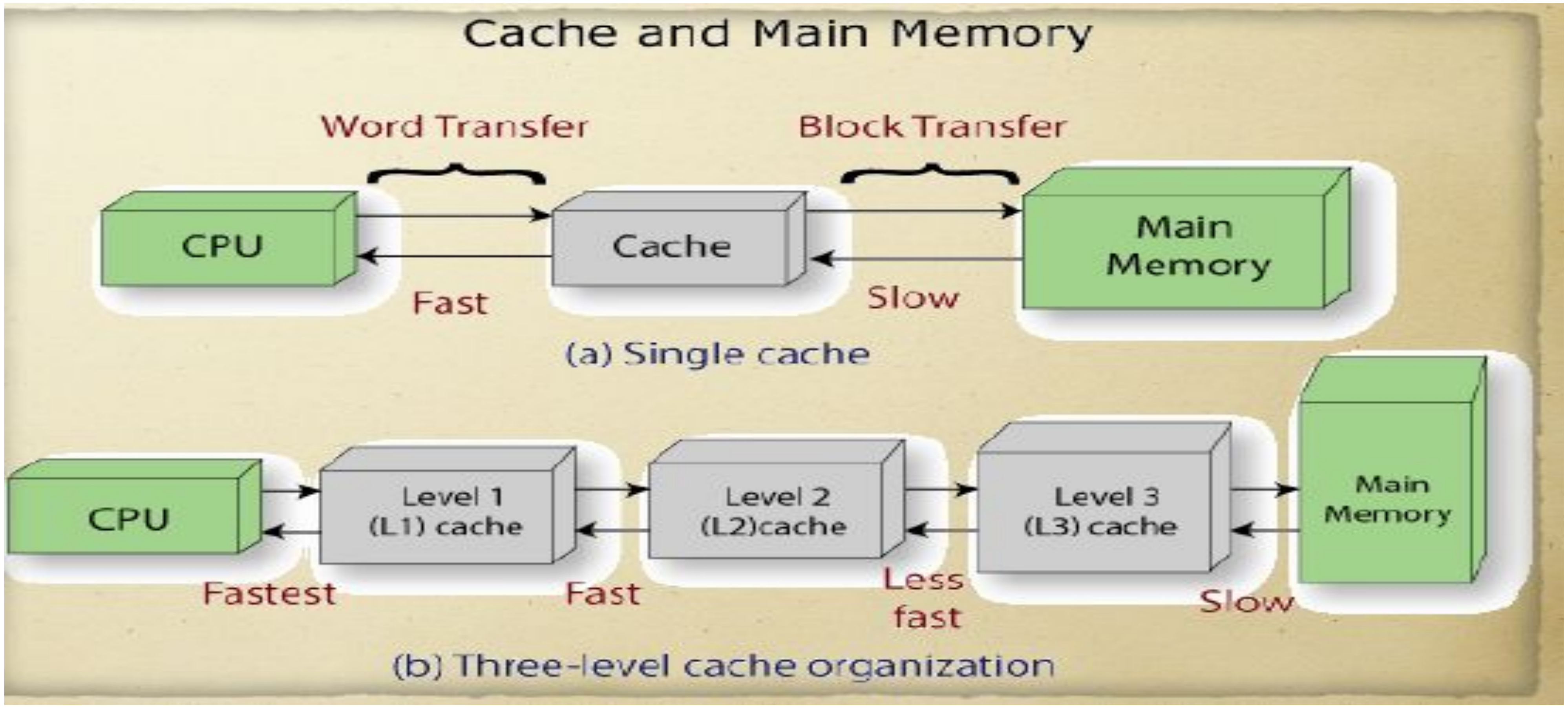
REGISTERS

- Registers are **temporary storage areas for instructions or data.**
- Registers work **under the direction of the control unit** to accept, hold, and transfer instructions or data and perform arithmetic or logical comparisons at high speed.
- **Information is moved from memory to the registers; the registers then pass the information to the ALU for logic and arithmetic operations.**
- The size of register plays an important role in the performance of CPU.
- A **32-bit processor** means it can perform operations on 32-bit data; therefore, the **size of registers is 32 bits** and ALU also performs 32-bit operations.
- A **64-bit CPU** performs operation in 64-bit data; therefore, **it contains 64-bit register and 64-bit ALU.**

CACHE MEMORY

- Cache memory is a **small amount of high-speed memory used for temporary data storage based between the processor and main memory.**
- The cache is a **volatile type of computer memory**.
- It can **store frequently used** computer programs, application, and data.
- **Data is transferred between memory and cache in blocks of fixed size, called *cache lines* or *cache blocks*.**
- When trying to read from or write to a location in the main memory, the processor checks whether the data from that location is already in the cache.
- If so, the processor will read from or write to the cache instead of the much slower main memory.

- Most CPUs have a hierarchy of multiple cache levels (L1, L2, often L3, and rarely even L4), with separate instruction-specific and data-specific caches at level 1.



SYSTEM BUSES

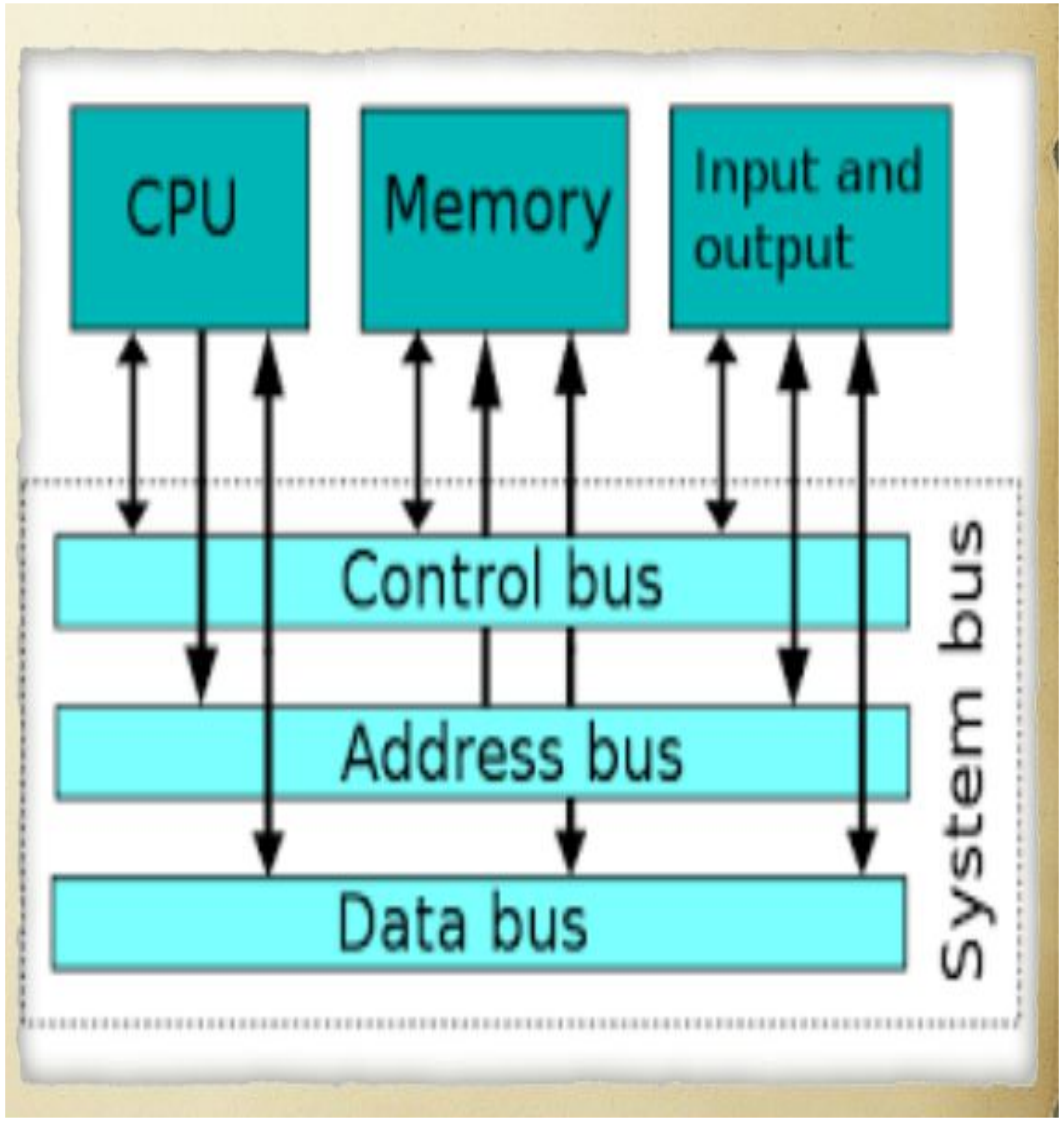
- **Buses** are **collections of wires or groups of conductors that connect several devices within a computer system.**
- Buses **allow for the transfer of electrical signals** between different parts of the computer system and thereby transfer information from one device to another.
- The *interface* for the System Bus in the CPU is referred to as **Bus Control block.**
- The System Bus is formed of several sub-busses each with its particular tasks.

Most important are :

- **Address Bus**
- **Data Bus and**
- **Control Bus.**

Data Bus: The **bidirectional** data bus, handles the transfer of all **data** and **instructions** between the units of the computer.

- It **transfer instructions coming from or going to the processor.**
- The size of data bus plays important factor on CPU performance.
- **A CPU with 32-bit data bus means it can read or write 32 bits of data in and from memory.**



Address Bus:

- It is a **unidirectional bus** (called the *memory bus*) **transports memory addresses which the processor wants to access in order to read or write data.**

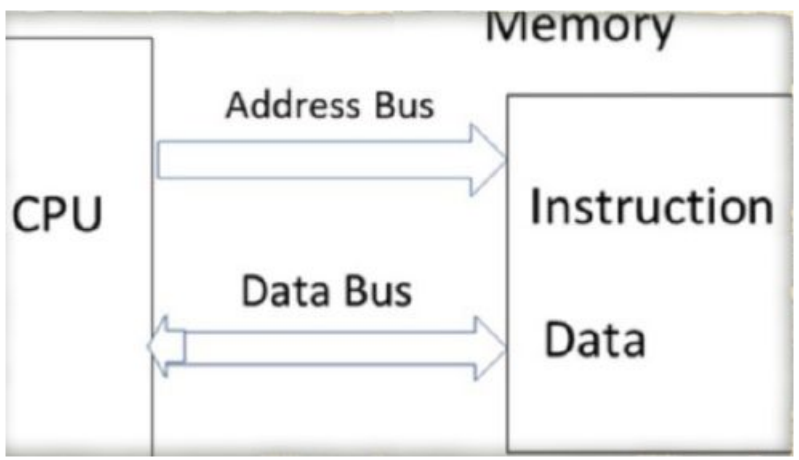
Control Bus:

- It is a **bidirectional bus** (or *command bus*) **transports orders and signals coming from the control unit and traveling to all other hardware components, as it also transmits response signals from the hardware.**
- Control bus is used by the **CPU to direct and monitor the actions of the other functional areas of the computer.**

COMPUTER ARCHITECTURE

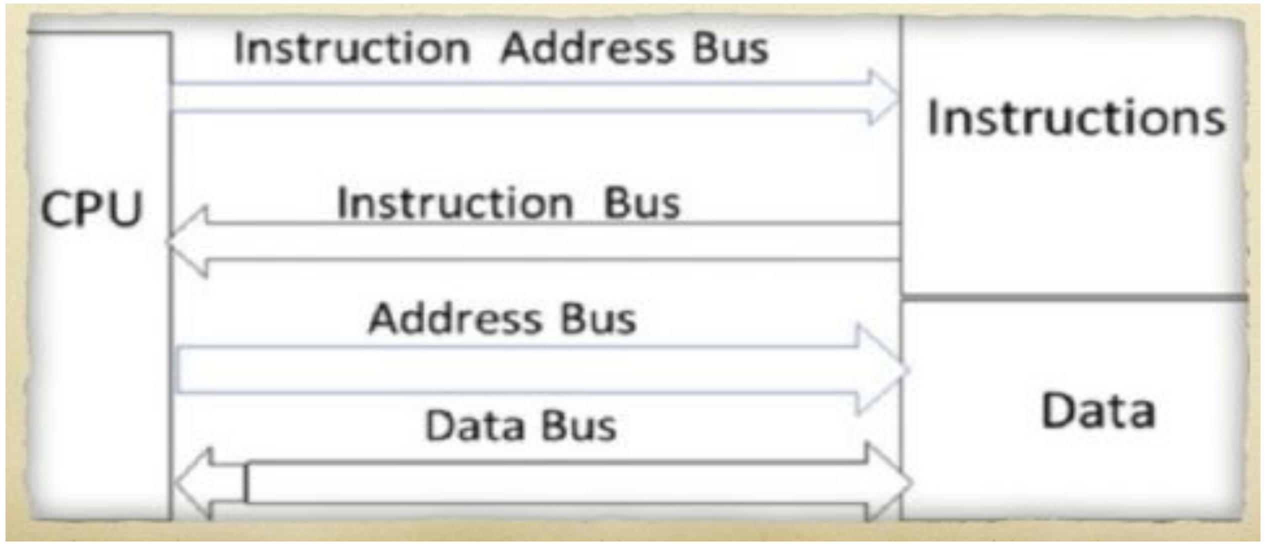
Von Neumann Architecture

- This idea decrees that **instructions are held sequentially in the memory** and that the processor executes each one, in turn, from the **lowest address in memory to the highest address in memory, unless otherwise instructed.**
- It is a program which consists of code (instructions) and data.
- Von Neumann uses the **data bus to transfer data and instructions** from the memory to the CPU.



Harvard Architecture

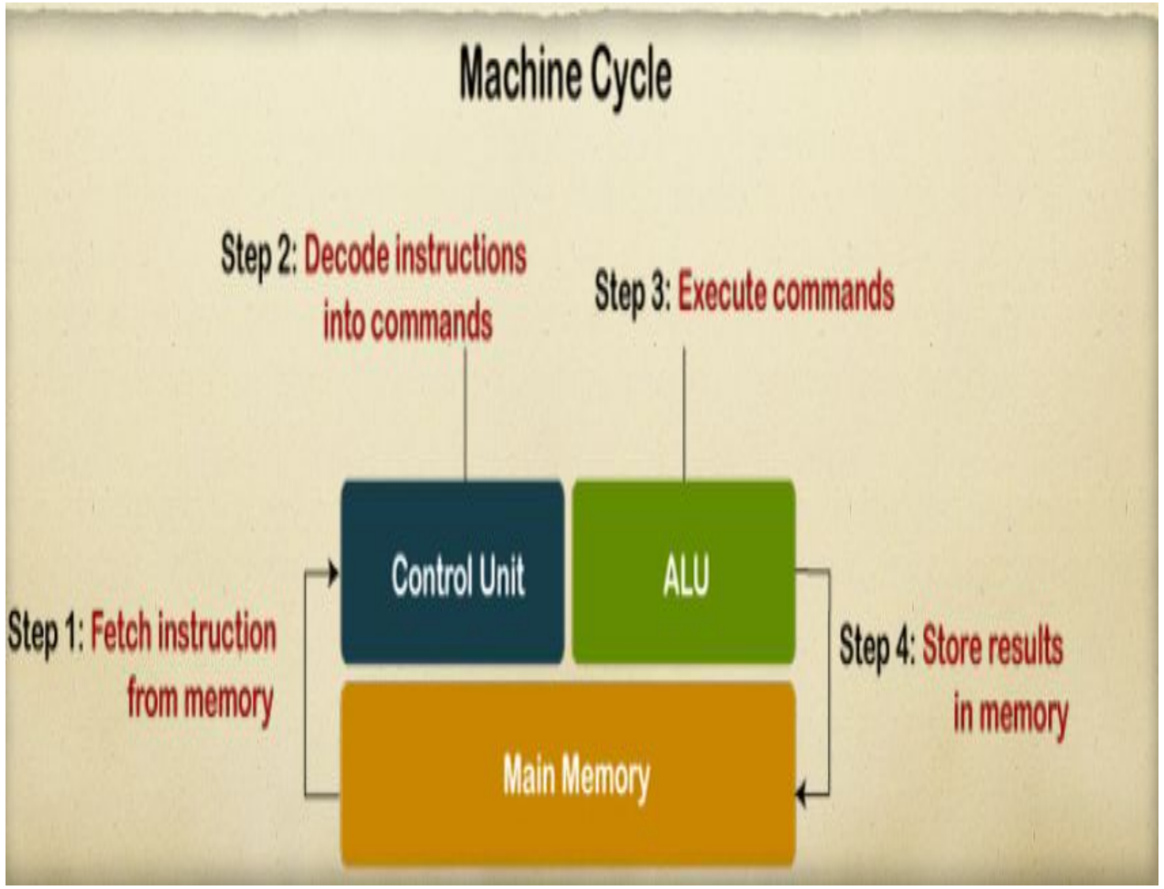
- Harvard architecture **uses separate buses for instructions and data.**
- The **instruction address bus and instruction bus are used for reading instructions from memory.**
- The **address bus and data bus are used for writing and reading data to and from memory.**



CPU OPERATIONS

- In general, CPU performs the following steps to execute one instruction:

1. **Fetch instruction (F):** Moving Instruction from memory to CPU.
2. **Decode instruction (D):** Determine types of instruction and Store operands into registers if needed.
3. **Execute instruction (E):** Execute the instruction.
4. **Write results (R):** Store the result of execution into register or memory.


Machine Cycle
Step 2: Decode instructions into commands
Step 3: Execute commands
Control Unit
ALU
Step 1: Fetch instruction from memory
Step 4: Store results in memory
Main Memory

MEMORY

- In a computer, memory **holds instructions (code) and data.**
- Semiconductor memory can be **volatile or non-volatile memory.**
- **Volatile memory loses its contents when power is removed from it (RAM).**
- **Non-volatile memory will keep its contents without power (ROM).**
- Memory chips are made up transistor type cells each of which can be in one of two states: either on or off, **representing the values 1 or 0.**

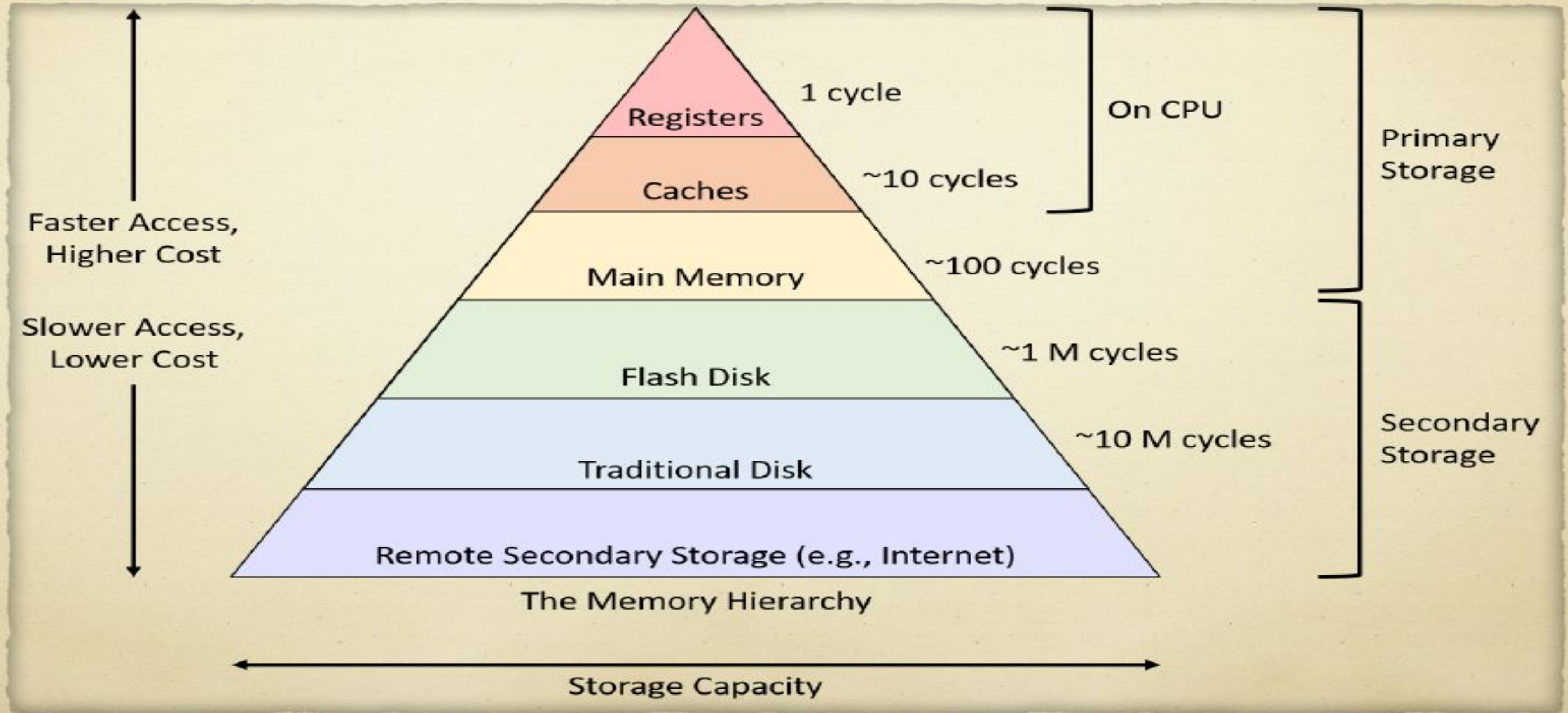
RAM

- RAM, which stands for **Random Access Memory,** is a hardware device generally located on the motherboard of a computer .
- Acts as an **internal memory of the CPU and allows it to store data, program, and program results when the computer is on.**
- It is the read and write memory of a computer.
- RAM is a **volatile memory, which means it does not store data or instructions permanently.**

ROM

- ROM, stands for **read only memory**.
- **Stores information permanently (non-volatile).**
- It is also the primary memory unit of a computer along with the random access memory (RAM).
- Known as **read only memory as we can only read the programs and data stored on it but cannot write on it.**
- ROM is **filled at the time of manufacturing and its content can't be altered**, which means you can't reprogram, rewrite, or erase its content later.
- However, there are other types of ROM which can be programmed.

Memory Heirarchies



The Memory Hierarchy

STORAGE DEVICES

- Physical components or materials **on which data is stored** are called **storage media.**
- Hardware components that **read/write to storage media** are called s**torage devices**.
- Purpose of storage devices is to **hold data even when the computer is turned off so the data can be used whenever needed.**
- Storage involves **writing data to the medium** and **reading from the medium.**
- **Writing data =** recording the data on the surface of the disk where it is stored for later use.
- **Reading data =** retrieving data from the surface and transferring it into the computers memory for use.

Optical Storage:

- Compact Disks(CD)
- DVDs
- Blu-ray disks

Magnetic Storage:

- Floppy Disks
- Hard disks
- Magnetic Tapes

Solid State Storage:

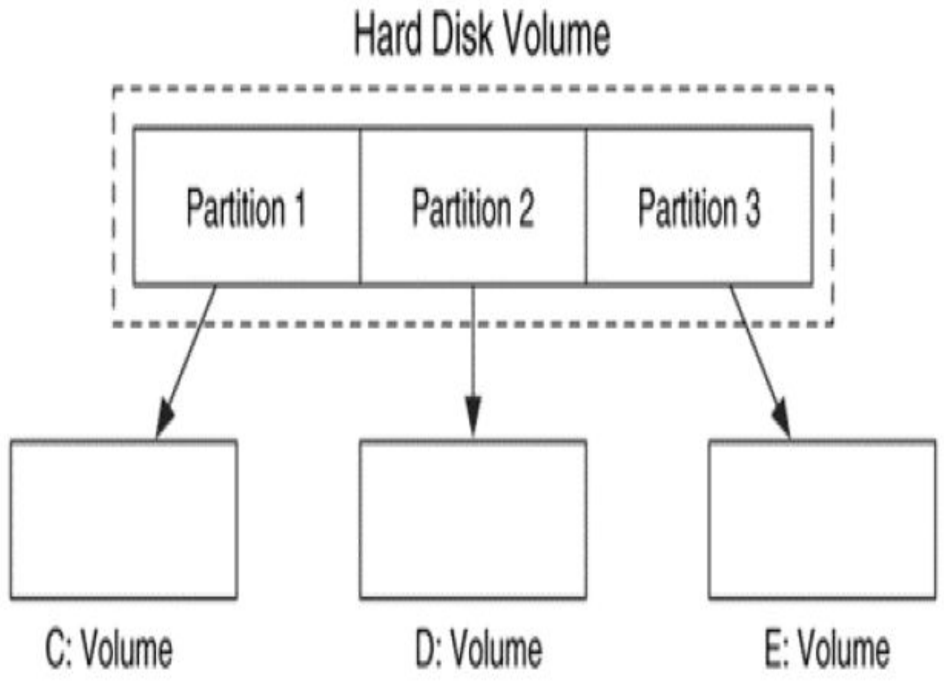
- Solid State Drives(SSDs, USB Flash Drives, Memory Cards)

VOLUME

- A volume is a **collection of addressable sectors that an Operating System (OS) or application can use for data storage.**
- In the simplest case, **a single disk equals one volume.**
- A volume consists of **file system information, a collection of files, and any additional unallocated space remaining on the volume that can be allocated to files.**
- Frequently, a disk is divided into partitions, with **each partition functioning as a separate volume.**

PARTITION

- A *partition* is a **collection of consecutive sectors in a volume.**
- It is a **logical division** of the hard disk, allowing a single hard disk to function as though it were one or more hard disks on the computer.
- **The purpose of a partition system is to organize the layout of a volume.**
- •Once a partition is set, it can be given a drive letter (such as C:, D:, etc.) and formatted to use a file system.
- **When an area of the hard disk is formatted and issued a drive letter, it is referred to as a *volume.***



MAGNETIC DISKS

- A magnetic disk consists of **one or more aluminium platters with a coating of material capable of being magnetized.**

- **Bits of data (0s and 1s) are stored on circular magnetic platters.**

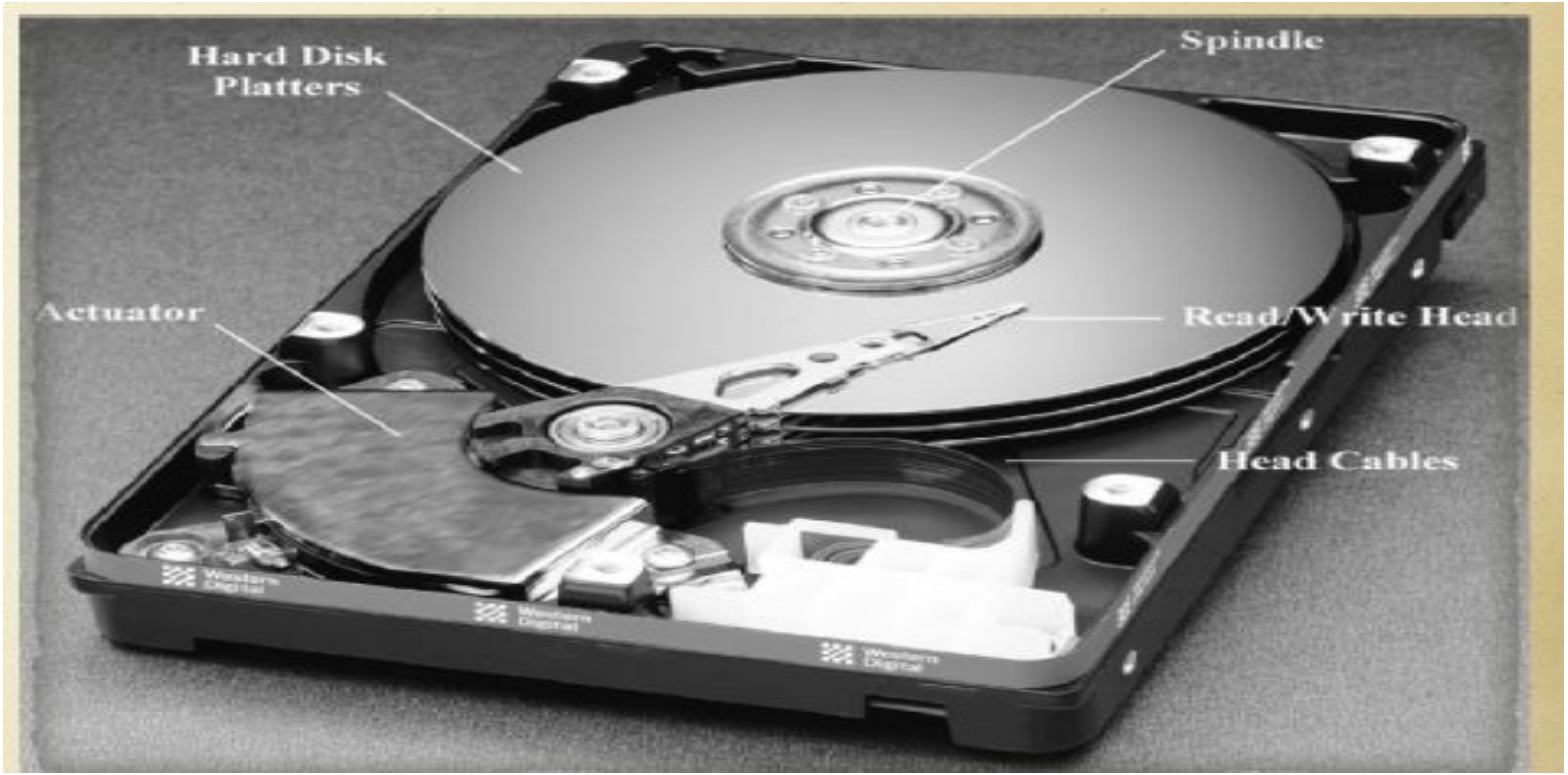
- A **disk rotates rapidly** and a **disk head reads and writes bits of data as they pass under the head.**

- The **read/ write heads of a magnetic disk drive contain electromagnets** that generate magnetic fields in the iron on the storage medium as the head passes over the disk.

HARD DISK GEOMETRY & INTERNALS

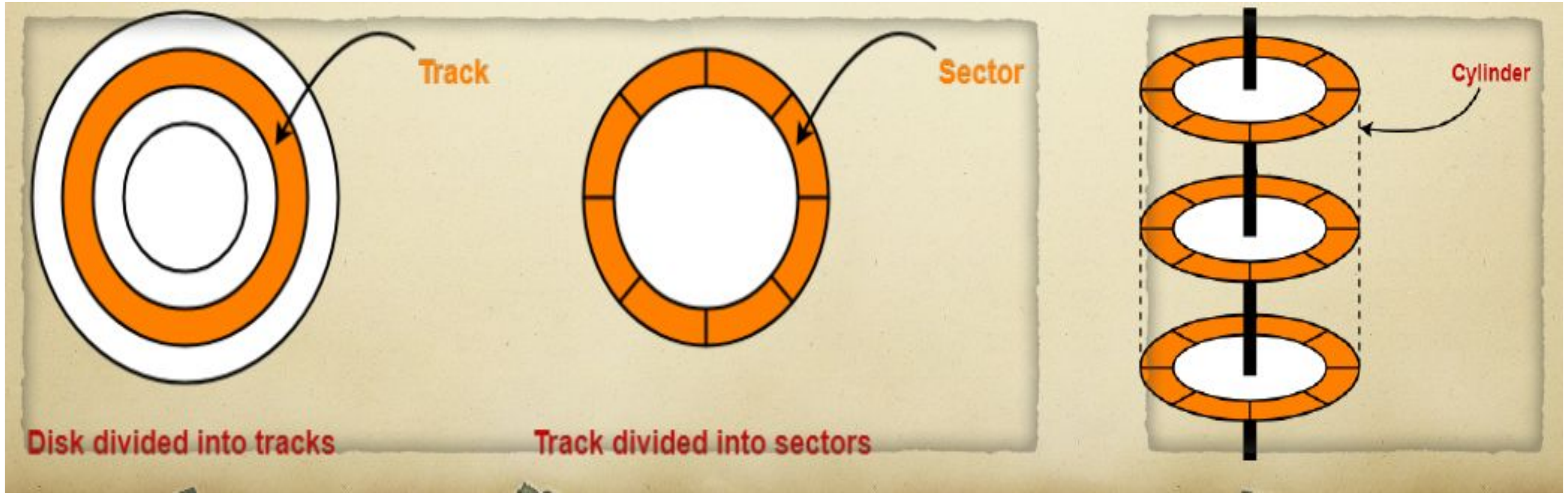
- Hard disks **contain one or more circular platters** that are stacked on top of each other and spin at the same time.
- **Each platter has 2 sides**, each of which is called a **surface**.
- The **bottom and top of each platter is coated with a magnetic media, and when the disk is manufactured, the platters are uniform and empty.**
- The platters are all bound together around the **spindle**, which is connected to a **motor that spins the platters around (while the drive is powered on) at a constant (fixed) rate.**

- The platter of a HDD rotates in a **speed of 4,500 RPM to 15,000 RPM.**
- The rate of rotation is often measured in **revolutions per minute (RPM).**
- It has an **Actuator** that moves the read-write arm.
- As the platters spin**, the drive heads move in toward the centre surface and out toward the edge.**
- In this way, **the drive heads can reach the entire surface of each platter.**


Hard Disk Platters
Spindle
Actuator
Read/Write Head
Head Cables

- **A low-level format is performed on the blank platters to create data structures.**
- The **process of mapping a disk surface is called formatting or initializing.**
- Formatting draws **tracks** (Concentric circles) on the disk, which are divided into **sectors**.
- In all diskettes and most hard disks, a **sector can store up to 512 bytes (0.5 KB).**
- Each **track and sector is labelled with numbers.**
- Each **track on the hard disk is given an address from the outside inward, starting with 0.**

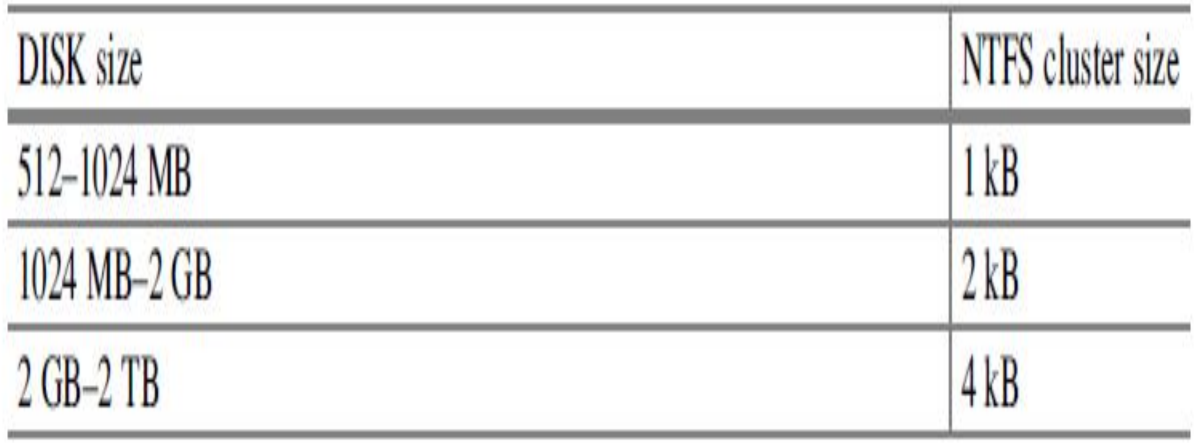
- The term **cylinder** is used to **describe all tracks at a given address on all platters.**
- The **heads in the disk are given an address so that we can uniquely identify which platter and on which side of the platter we want to read from or write to.**
- We can **address a specific sector** by using the **cylinder address (C) to get the track, the head number (H) to get the platter and side, and the sector address (S) to get the sector in the track.**



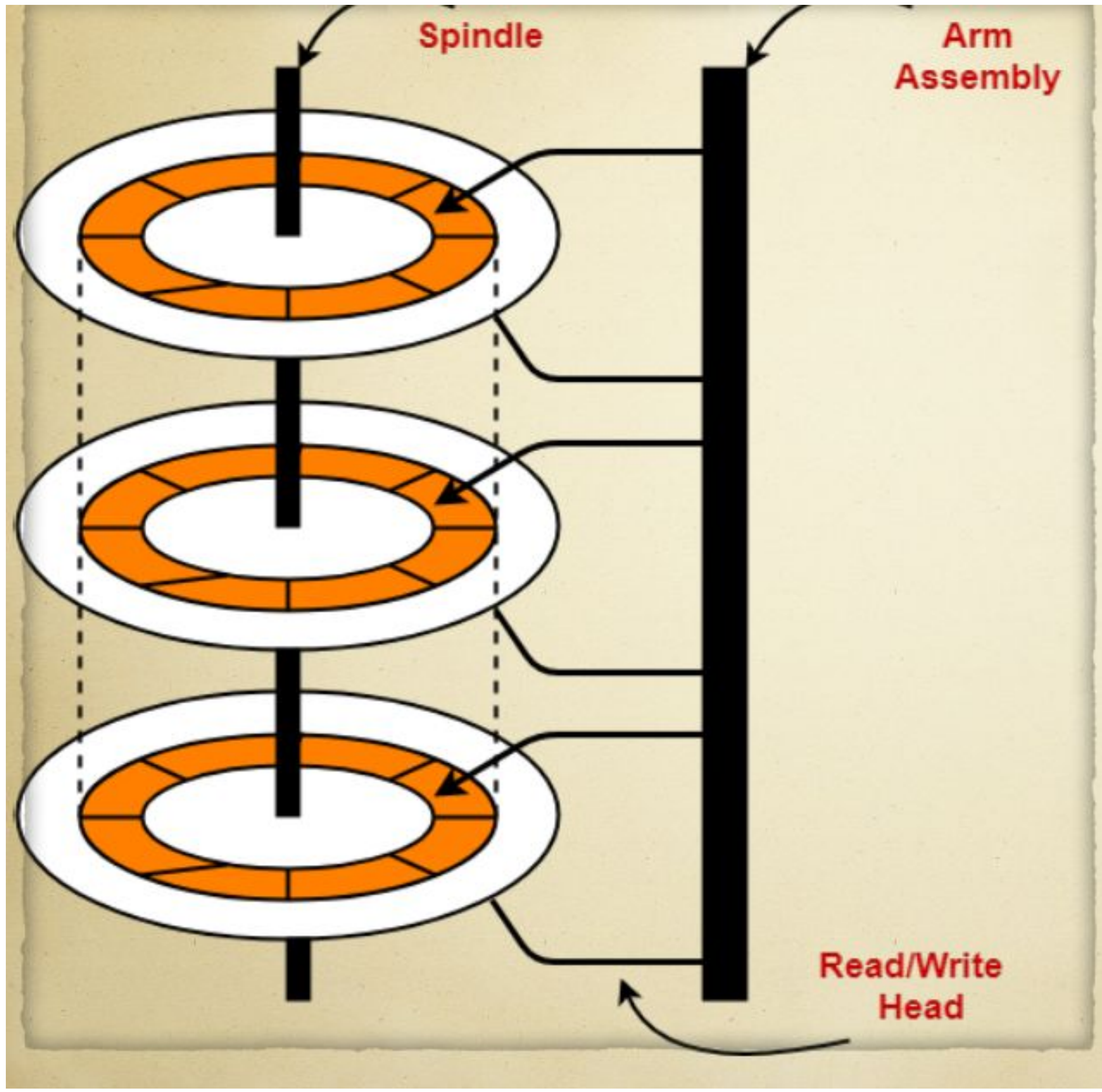
CLUSTERS

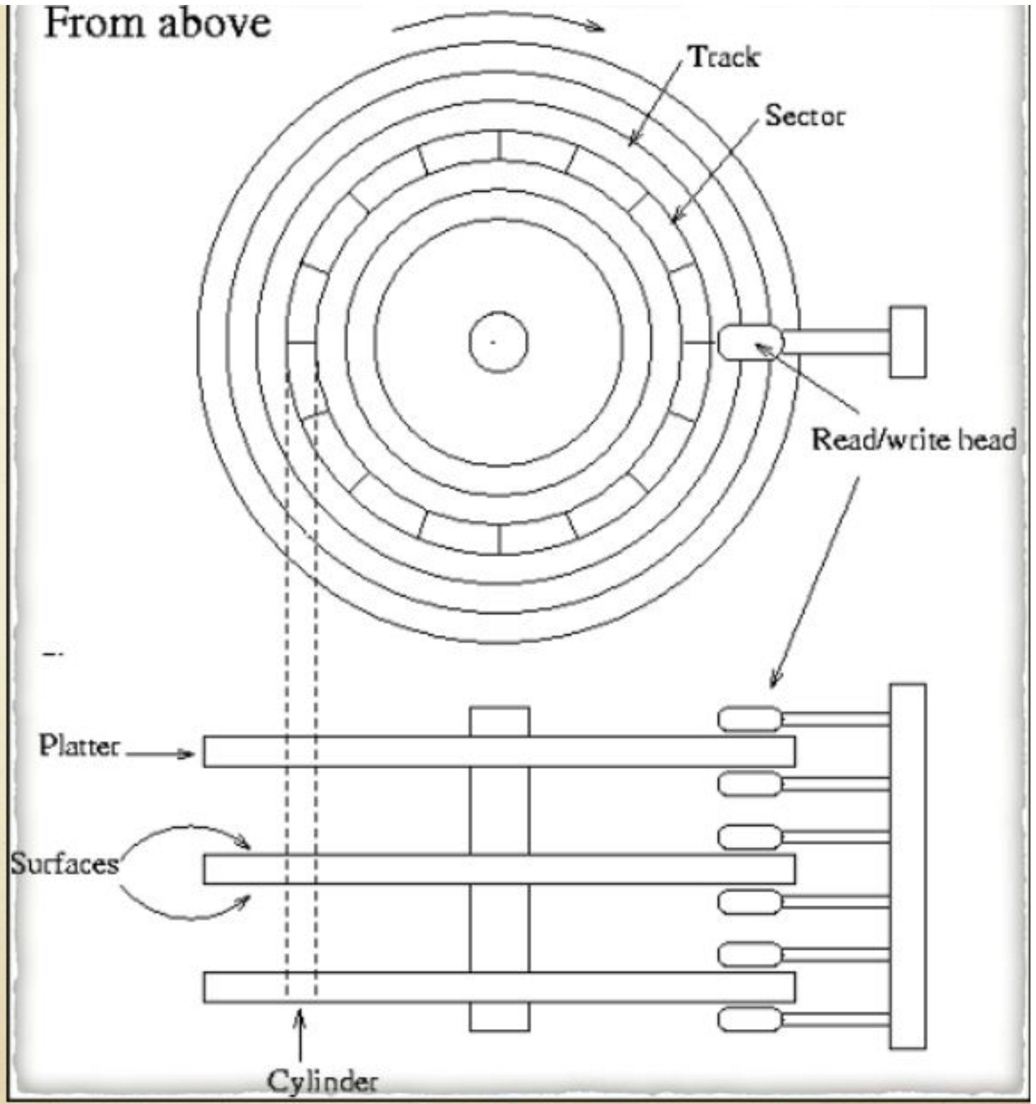
- Each sector of a disk is 512 bytes (B), and cluster is made of **one or more sectors**, if a cluster is 1 kB, then it is made of two sectors. The 2 kB cluster is made of 4 sectors.

Table 7.3 Default cluster size



DISK size	NTFS cluster size
512–1024 MB	1 kB
1024 MB–2 GB	2 kB
2 GB–2 TB	4 kB


Spindle
Arm Assembly
Read/Write Head


From above
Track
Sector
Read/write head
Platter
Surfaces
Cylinder

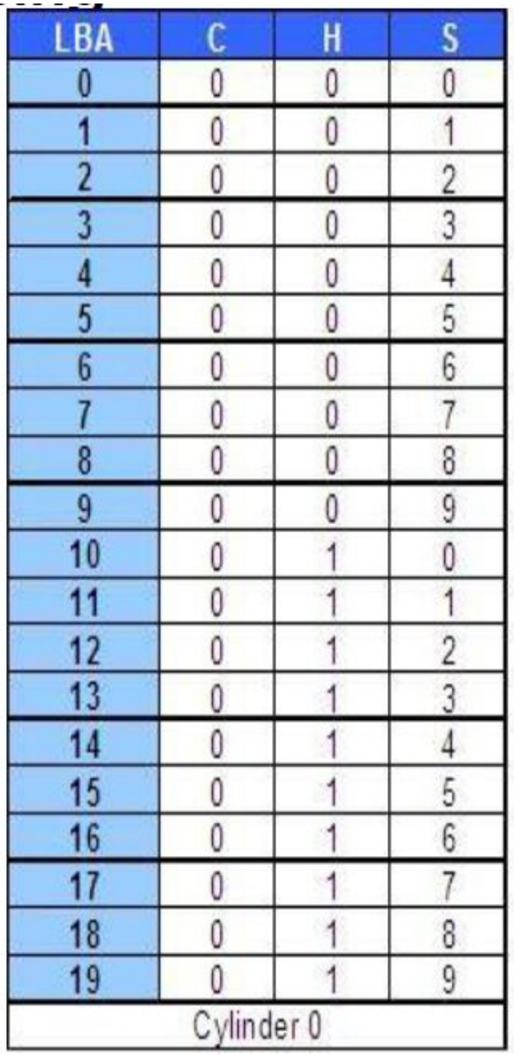
TYPES OF SECTOR ADDRESSING

CHS (Cylinder/Head/Sector) Addressing type:

- CHS is a method of **giving addresses to each physical block of data on a hard drive.**
- CHS divides the disk into cylinder, heads and sector.
- **Number of cylinders = Number of tracks**
- Sector: The capacity of every sector is 512 bytes.
- We can address a specific sector by using the cylinder address (C) to get the **track**, the head number (H) to get the **platter and side**, and the **sector address (S)** to get the sector in the track.

Logical Block Addressing (LBA)

- LBA uses a single number, **starting at 0, to address each sector.**
- With LBA, the software does not need to know anything about the geometry; it needs to know only a single number.



LBA	C	H	S
0	0	0	0
1	0	0	1
2	0	0	2
3	0	0	3
4	0	0	4
5	0	0	5
6	0	0	6
7	0	0	7
8	0	0	8
9	0	0	9
10	0	1	0
11	0	1	1
12	0	1	2
13	0	1	3
14	0	1	4
15	0	1	5
16	0	1	6
17	0	1	7
18	0	1	8
19	0	1	9
Cylinder 0			

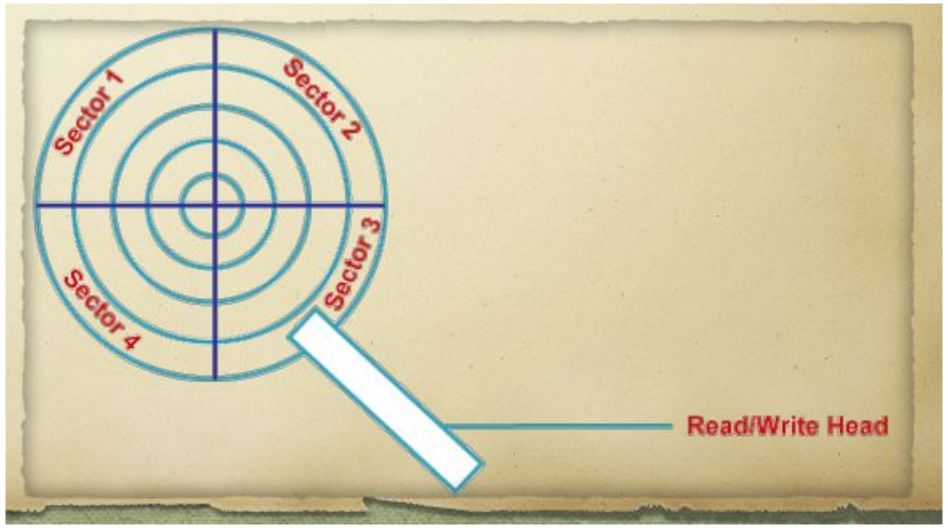
DISK CHARACTERISTICS

Seek Time

- To read or write a sector, first the arm must be moved to the right track. This action is called a seek.
- **Seek Time describes the time taken to position the R/W head on the desired track.**
- The average seek time on a modern disk is typically in the **range of 3 to 15 milliseconds.**
- The seek, has many phases: first an ***acceleration* phase** as the disk arm gets moving; then ***coasting*** as the arm is moving at full speed, then ***deceleration*** as the arm slows down; finally ***settling*** as the head is carefully positioned over the correct track.

Radial/Rotational Latency

- Once the head is positioned on the particular track, **there is a delay, called the rotational latency, until the desired sector rotates under the head.**
- **The time taken by the platter to rotate and position the sector under the R/W head is called radial/rotational latency.**
- It depends on the speed of the rotation, which is in revolutions per minute (RPM).
- **Average rotational latency = 1 / 2 x Time taken for full rotation.**



Data Transfer Rate

- The amount of data that passes under the read / write head in a given amount of time is called as **data transfer rate**.
- The data rate is the **number of bytes per second that the drive can deliver to the CPU.**
- The time taken to transfer (read or write the contents) the data is called as **transfer time**.
- Rates between **5 and 40 megabytes per second are common**.

Disk Access Time

- It is the **total time required to access the data.**
- It means total time required to read/write the data from the disk.
- **Disk Access Time = Seek time + Rotational Delay + Data Transfer Time.**

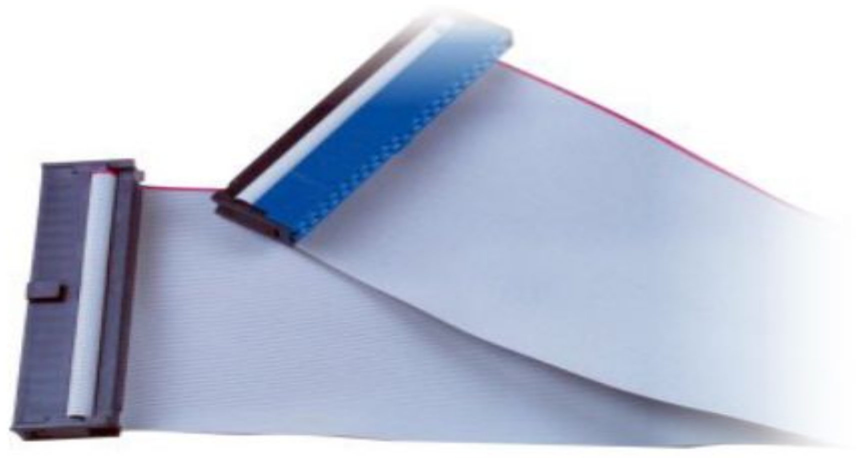
Disk Capacity

- Hard disk capacity is determined by elements of the disk, including the number of tracks, sectors, and surfaces on which data can be written.
- **Capacity of disk = Total number of surfaces x Number of tracks per surface x Number of sectors per track x Number of bytes per sector.**

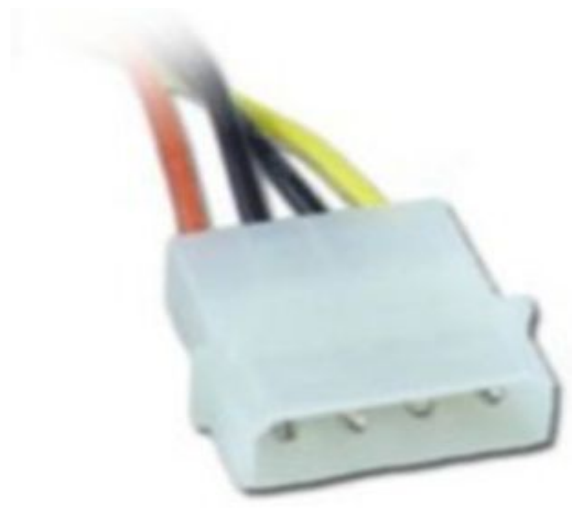
HARD DISK CONNECTOR TYPES

- Computer hard disks can be **categorized using the types of connectors that they use to connect to the motherboard.**
- When it comes to connecting the internal hard drive to the motherboard there are **4 main hard disk interface types** used.
- **They are used to move data from the HD to the main memory (RAM) and computer processor.**

- **Parallel Advanced Technology Attachment (PATA):** They are also known as Integrated Drive Electronic (IDE). PATA was the initial technology used to connect hard disk drives. They had a data rate of around **133MegaByte/Second.**



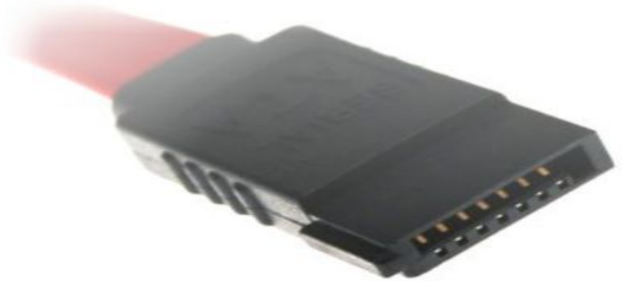
Pic. 12. IDE (PATA) data cable.



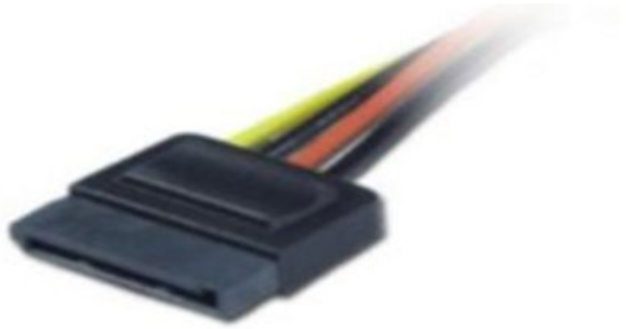
Pic. 13. Molex power cable connector.

In this picture, the **blue connector** is used to connect the cable to the mainboard of the computer, while the **black one** is attached to the drive.

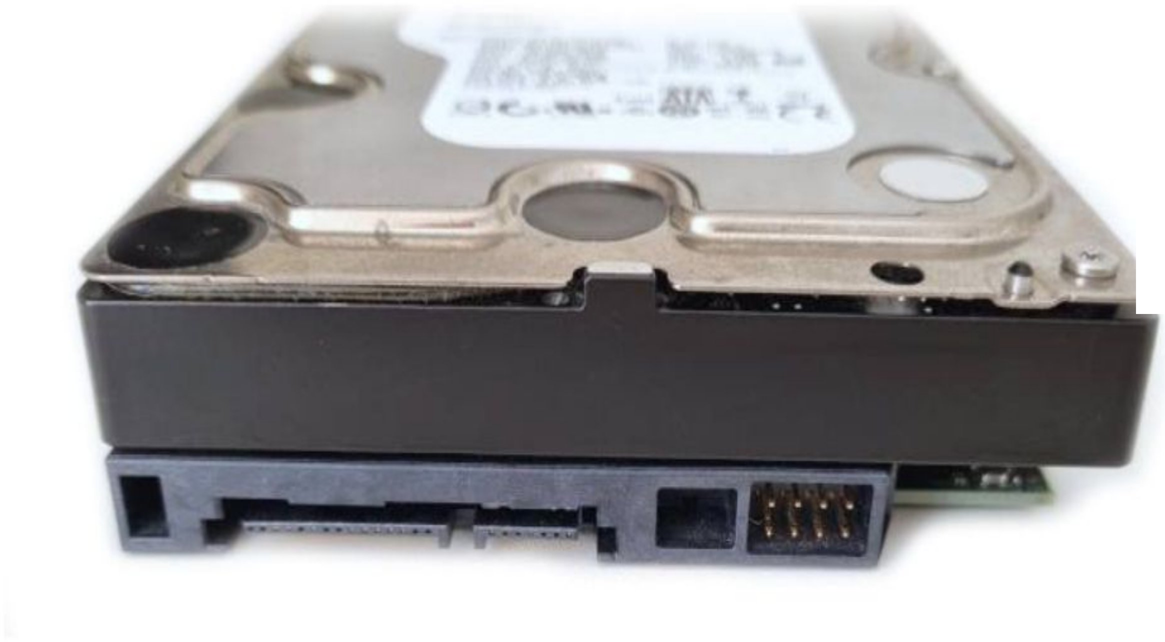
- **Serial ATA storage drive:** The types of connection remain the same but **data signalling changed from parallel to serial.** Data was transmitted one bit at a time and **faster than in the PATA. They also have smaller cables and consumed less power.**



Pic. 1. SATA data cable connector.

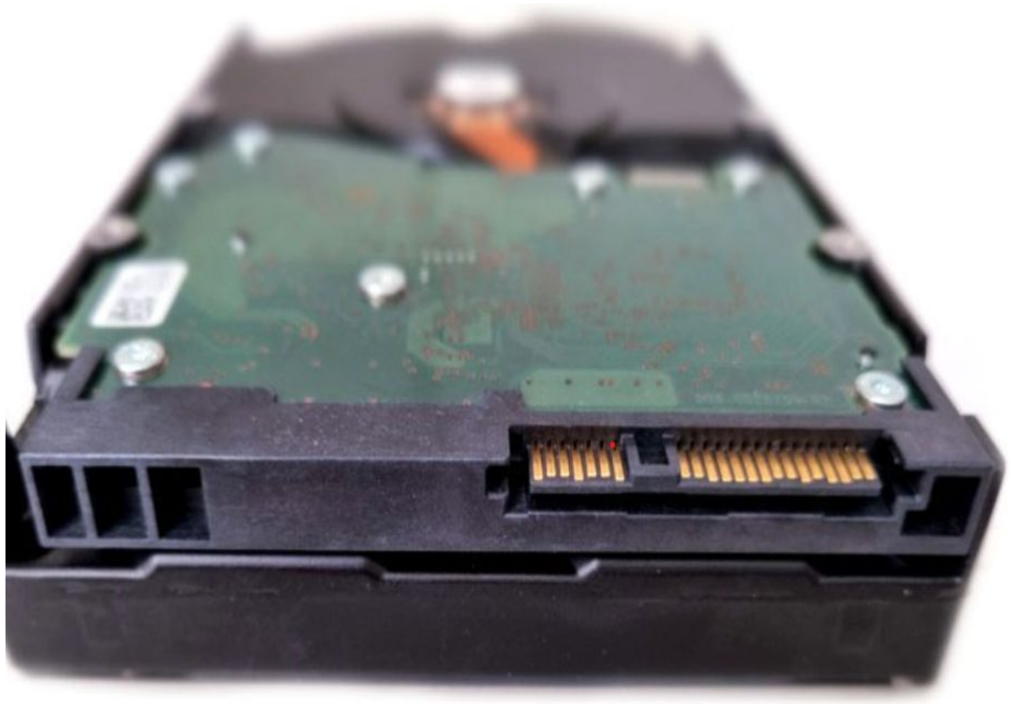


Pic. 2. SATA power cable connector.



Pic. 3. Back panel of the SATA hard drive.

- **Small Computer System Interface (SCSI):** This connection is meant to connect peripheral devices and it’s faster than the SATA.



Pic. 5. Back panel of a SAS drive.

Pic. 4. SAS data and power cable.

- **Universal Serial Bus (USB):** This is the latest type of connector that is replacing others. It is mostly used to **connect an external hard disk to the motherboard.** Most internal drives are not using a USB for connection.

HPA AND DCO

- The Host Protected Area (HPA) is an **area of memory on a hard drive that is not normally visible to a computer's Operating System (OS)** – for example, it would not be available for the user to store files on.

- **It was implemented so information could be stored that is not easily modified, changed, or accessed by the user, BIOS, or the OS.**

- **This means that a drive may contain more than the 'advertised' capacity. This part can only be made accessible using specific tools or commands.**

There are numerous reasons why an HPA is created on a drive:

- A vendor can store the necessary files to install or recover the computer’s operating system i.e. **perform a factory reset.**
- Other software may be installed here such as **diagnostic programs or other utilities**.
- **Some malware** may hide inside the HPA to avoid detection.
- Users who wish to go to great lengths **to hide data may utilise the HPA.**

- Computer system vendors may use the **Device Configuration Overlay (DCO)** when they wish to have a series of hard drives of varying capacities that all exhibit the **exact same storage volume from the perspective of the OS.**

- For example, the hard drives that a vendor wishes to deploy in their systems may contain more addressable areas (i.e. storage capacity) than is required.

- Using the DCO feature, the **vendor could modify these drives so they consistently display the same amount of usable storage.**

- **They effectively hide the additional areas from the user's view without any indication that they even exist.**

FEATURES OF HDD

- **Non-volatile:** HDD is a **non-volatile memory device. This makes it an ideal storage medium for long-term data storage.**
- **High Capacity:** HDDs can **store a large amount of data.** Modern HDDs can store terabytes of data, making them an **ideal choice for storing large files such as videos, photos, and audio recordings.**
- **Relatively Slow Speed:** Compared to primary memory devices such as RAM, HDDs are relatively slow. The data access time for an HDD is typically measured in **milliseconds**, while the **access time for RAM is measured in nanoseconds**. This makes HDDs better **suited for long-term storage rather than for frequently accessed data.**

- **Mechanical Parts:** Unlike solid-state drives (SSDs), **HDDs have mechanical parts that can wear out over time, leading to reduced performance or even failure**. HDDs contain spinning disks and moving read/write heads, which can be susceptible to damage if the drive is bumped or dropped.
- **Cost-effective:** HDDs are **generally less expensive** than SSDs, making them a popular choice for budget-conscious users. This is particularly true for larger capacity drives.
- **Reliable:** While HDDs are not as reliable as SSDs due to their mechanical components, they are still **considered to be a reliable storage medium**. HDDs are designed to withstand heavy use and are often used in enterprise-level storage solutions.

SSD INTERNALS AND GEOMETRY

- Traditional hard disk drives (HDDs) are mostly based on moving parts, like a read/write head that goes back and forth to gather data.
- SSD uses a **simple memory chip called NAND flash memory**, which has no moving parts and near-instant access times.
- **Flash memory is non-volatile, which means that it doesn't lose its contents when it loses power.**
- **SSD access times are measured in nanoseconds.**

1. Outer Shell

- The outer shell could be of metal or plastic and it helps in absorbing most of the heat from inside the flash memory.

2. NAND Flash

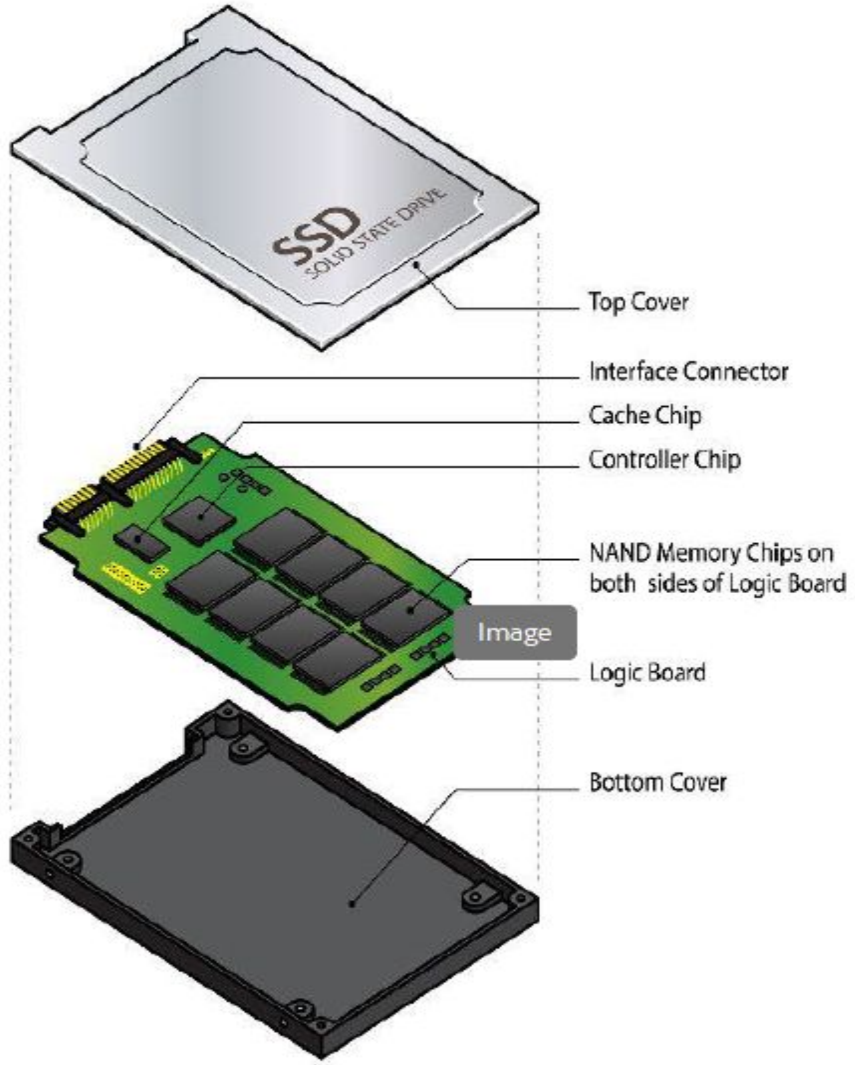
- **Data is held in memory chips called NAND Flash**.

3. DRAM Buffer

- A bit of DRAM is included in every SSD for the process of buffering. **Similar to hard drive's cache, data is stored on it for some time temporarily before it is being written to the device.**

4. Controller

- The controller, normally a multicore processor, is the **heart of the SSD** and it is the one that performs all the functions.



Top Cover
Interface Connector
Cache Chip
Controller Chip
NAND Memory Chips on both sides of Logic Board
Image
Logic Board
Bottom Cover

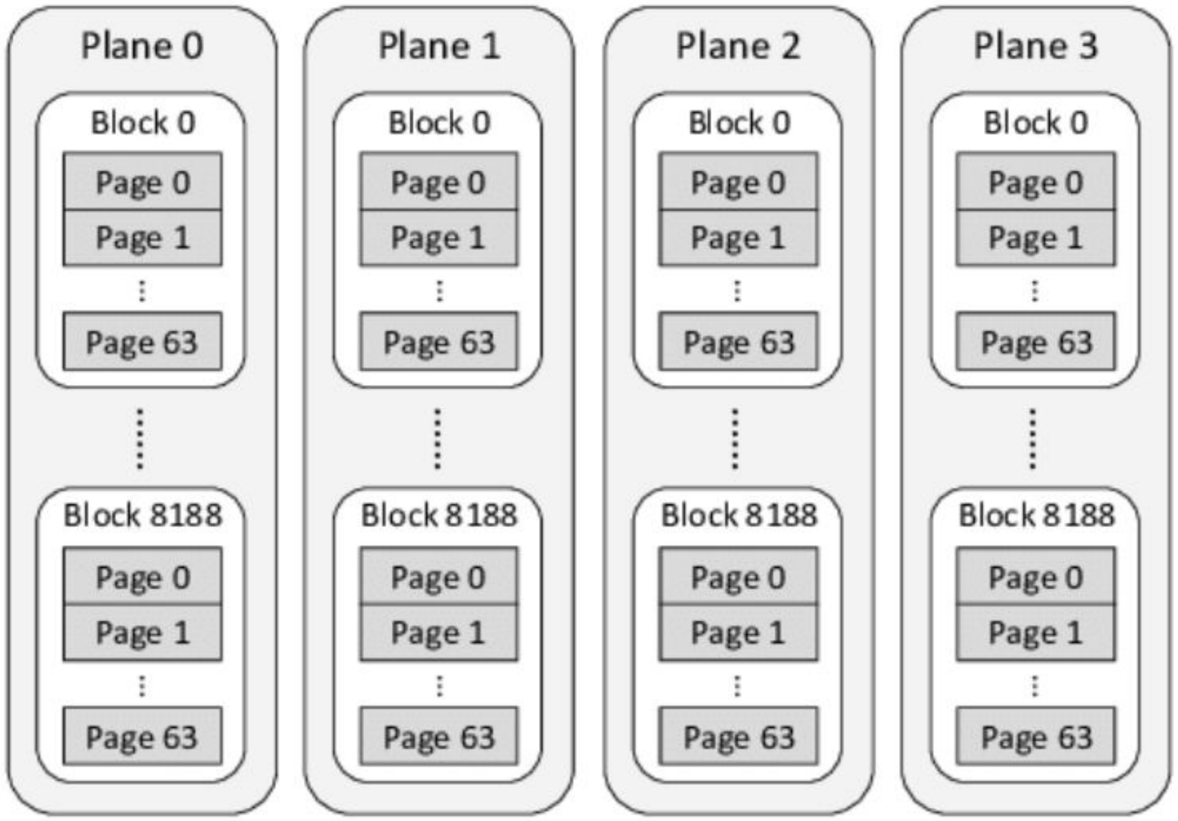
1
2
3
4

SSD ORGANISATION

- At the lowest level, a **solid state drive stores bits.**
- Eight bits make up a byte, and while on the typical mechanical hard drive 512 bytes would make up a sector, **solid state drives do not have sectors.**
- Solid state drives have a similar physical data object called a **page**.
- The **page is the smallest object that can be read or written on a solid state drive.**
- Unlike mechanical hard drives, pages do not have a standard capacity.

- **A page's capacity depends on the architecture of the solid state memory chip.**
- **Typical page capacities are 4 KB, 8 KB, and 16 KB.**

- A solid state drive **block is made up of pages.**
- **A block may have 32, 64, or 128 pages.** 32 is a common block size.



BASIC FLASH OPERATIONS

Read (a page):

- Client of the flash chip can read any page (e.g.,2KB or 4KB), simply by specifying the **read command** and appropriate page number to the device.
- Typically quite fast, **10s of microseconds** or so.
- Able to access any location uniformly quickly means the device is a **random access device.**

Erase (a block):

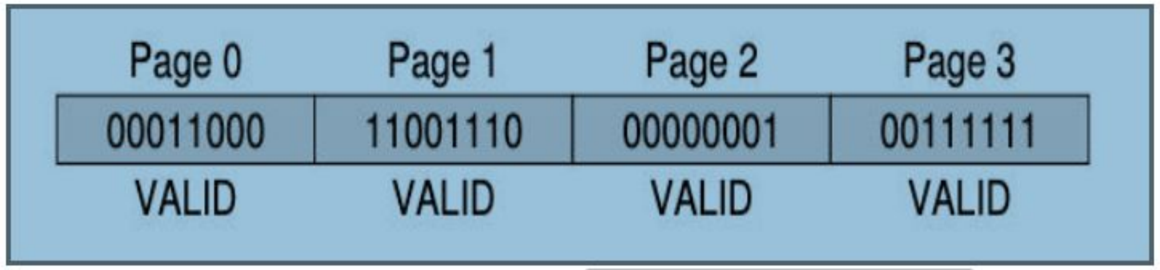
- Before writing to a page within a flash, the **nature of the device requires that you first erase the entire block the page lies within.**
- Erase, importantly, **destroys the contents of the block (by setting each bit to the value 1).**
- Therefore, you must be sure that any data you care about in the **block has been copied elsewhere before executing the erase.**
- Once finished, the entire block is reset and each page is ready to be programmed.

Program (a page):

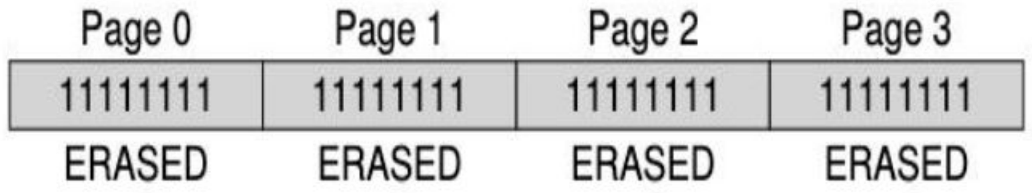
- Once a block has been erased, the **program command can be used to change some of the 1’s within a page to 0’s,** and write the desired contents of a page to the flash.
- Usually taking around **100s of microseconds** on modern flash chips.

AN EXAMPLE TO UNDERSTAND WORKING OF SSD

- Imagine we have the four 8-bit pages, within a 4-page block (both unrealistically small sizes, but useful within this example);
- Each **page is VALID** as each has been previously programmed.



- Say we wish to write to page 0, filling it with new contents.
- To write any page, we **must first erase the entire block.**



- We can now go ahead and program page 0 with required content, e.g., 00000011, overwriting the old page 0 (old contents 00011000) as desired.

- **Bad news: the previous contents of pages 1, 2, and 3 are all gone.**

GARBAGE COLLECTION

- The goal is to enable the flash storage device to have **enough spare blocks so that whenever data must be written**, the device does not have to wait for a block to be erased and made available for the new data.

- **When you modify the data after it is stored in pages, the SSD just writes the changes into a new page and marks the old page where the modified data is stored as stale page**.

- In order to avoid filling the SSD with stale pages, the **garbage collection comes out.**

- When idle pages in the SSD become less than a certain amount, **garbage collection** will be triggered. Its process is as follows:

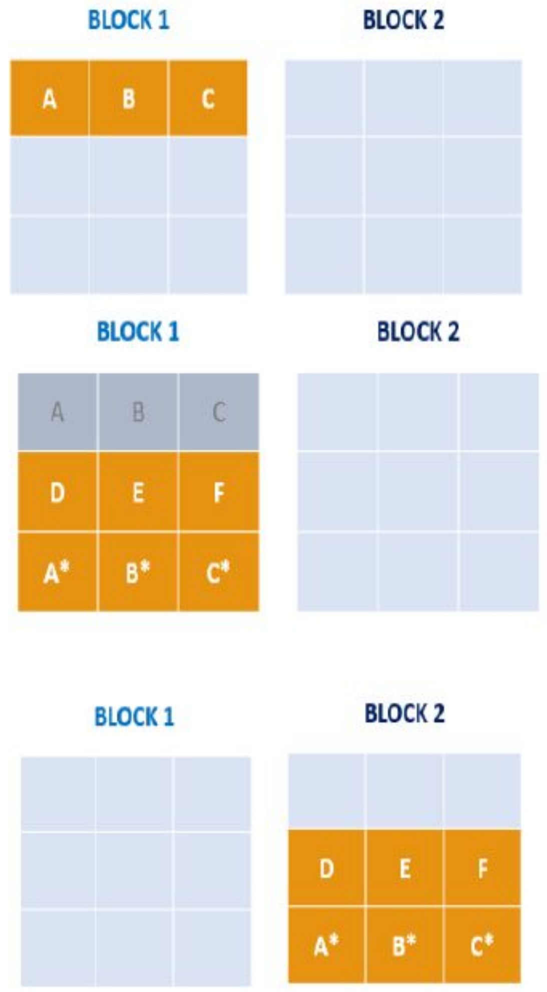
(1) Copying the valid data to a reserved block, just leaving stale data in the original block.

(2) Erasing the original block to get a new available block.

EXAMPLE TO UNDERSTAND GARBAGE COLLECTION

- Pages A, B & C in Block 1 contain active data.
- The **remaining pages in the block are free and available whenever new data needs to be written or stored.**

- Data in pages A, B and C in Block 1 have been modified and replaced as **A*, B* and C*.**
- **New data have been written to pages D, E and F.**
- **Pages A, B and C are now marked as invalid or stale**; however, they cannot be written with new data unless the entire Block 1 is erased.
- **Garbage collection is performed.**
- **Valid data in pages D, E & F as well as A*, B* & C* are copied to spare Block 2.**
- **All pages in Block 1 are erased and the block is now available to be written with new data.**



DIFFERENCES B/W HDD AND SSD

Feature	HDD	SSD
Access Time	Higher	100 times faster
Price	Cheaper	Costlier
Reliability	Low	High
Capacity	Less	More
Power Consumption	More	Less
Noise	More	Less
Size	Bigger	Smaller
Heat Generation	More	Lesser
Magnetic Effect	Affected	Not Affected
Vintage	Old	New Release
Weight	Heavier	Lighter

HDD	SSD
Easy to retrieve deleted data	Difficult to retrieve deleted data
Traditional Forensic can be applied	Data extraction not possible through HDD technique
New data can be overwritten over existing data	Data can not be overwritten. Old data has to be erased before writing fresh data
Data can be written nearly infinite times till there is a bad sector or physical damage	Data can be written finite number of times
Evidence is preserved	Evidence is destroyed

ADVANTAGES OF SSD OVER HDD

- **Faster speed.** As SSD uses flash memory, reading speed is relatively faster than that of traditional hard disk. In addition, **there is no head in SSD, so seeking time is nearly 0 .**

- **More resistance to physical shock.** Any physical occurrences do not affect SSDs to the same degree as HDDs because there are no moving parts to break.

- **Lower power consumption and less heat production.** Many SSDs require lower power and produce less heat.

- **No noise.** Solid-state drives don't have mechanical motor and fan, so the working **noise value is 0 Db.**
- **Wider range of working temperature.** A typical hard drive can only work within the scope of 5 to 55 degrees Celsius. However, most of the **SSD can work within -10 to 70 degrees Celsius.**

HDD FORENSICS

- **Actual file not deleted.**
- **New file overwritten over old file.**
- Existing Forensic Tools like Encase, FTK, Autopsy, Volatility can be used to extract data.
- **Well established procedures** as per NIST guidelines.
- **Evidence hash** is preserved and hence admissible in Court of Law.
- **High data recovery rate** unless anti-forensic techniques are applied.

SSD FORENSICS

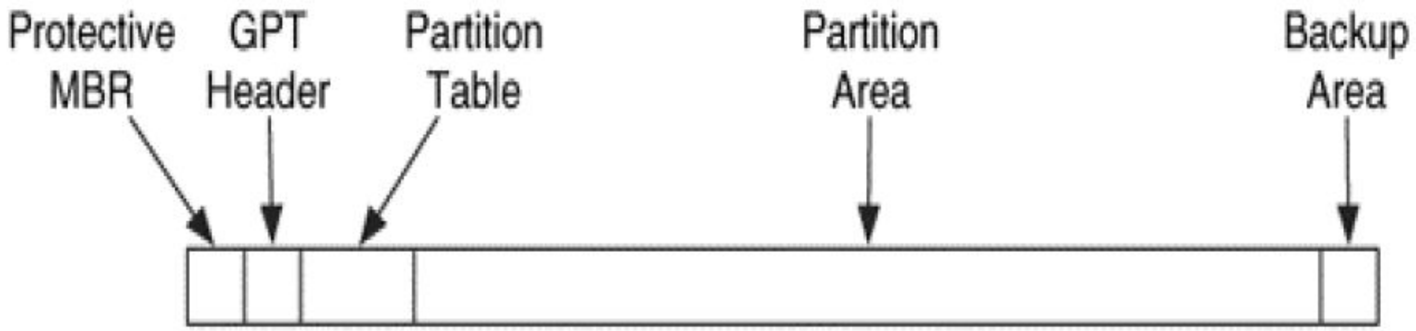
- SSD comprises of **Multiple chips.**
- **Difficult to apply Chip Off technique.**
- **Different addressing & memory management applied by different manufacturers**.
- Internal routines of SSD controllers is not made public hence **data recovery difficult**.

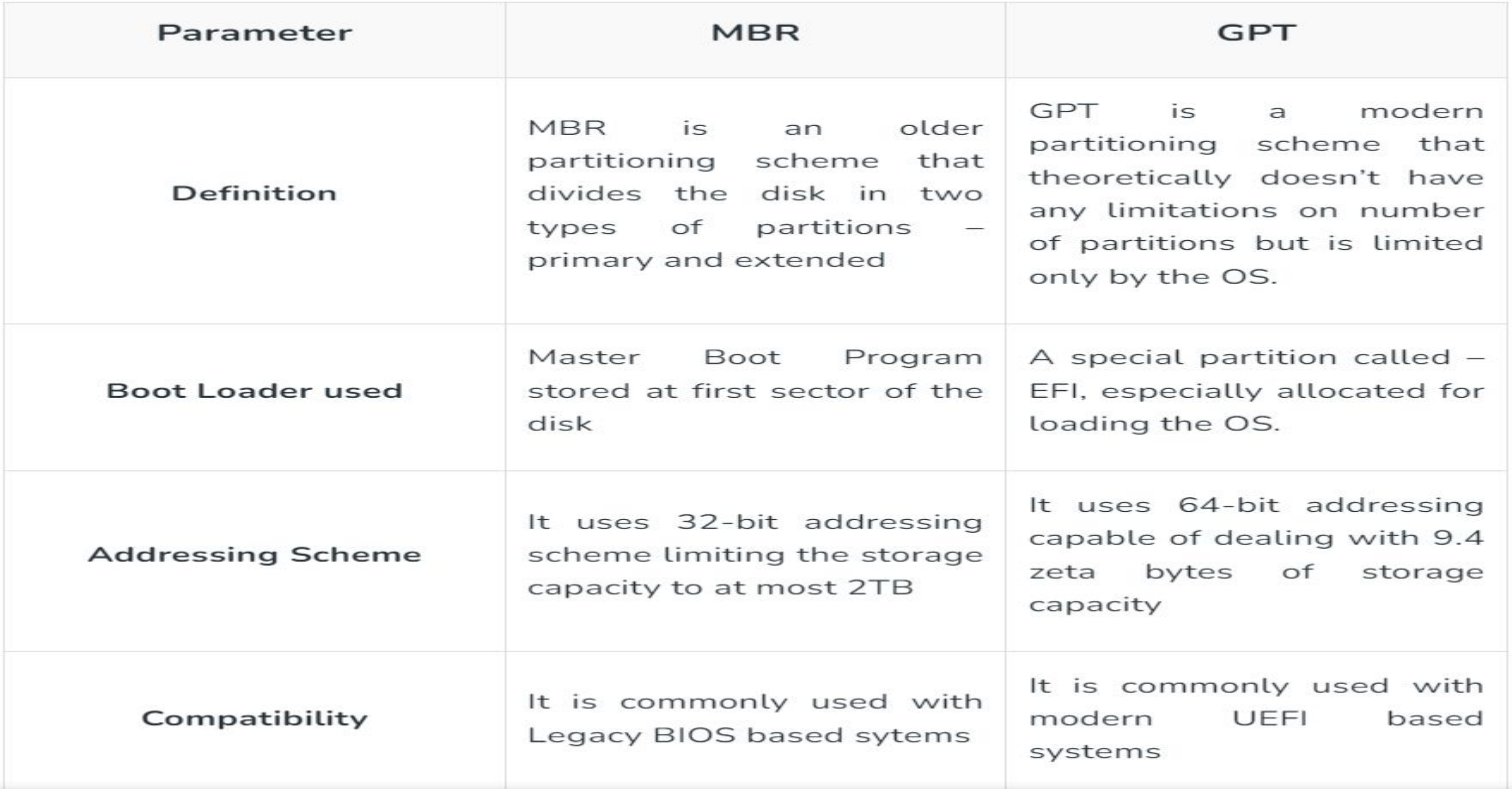
INTRODUCTION TO MBR AND GPT

- **The first sector (sector 0)** on a hard disk is always the **boot sector.**
- This sector contains codes that the computer uses to start the machine.
- The boot sector is also referred to as the **Master Boot Record (MBR).**
- The MBR contains the
- (i) **boot code** which contains the instructions that tell the computer how to process the partition table and **locate the operating system;**
- (ii) **a partition table**, which stores information on which primary partitions have to be created on the hard disk, so it can then use this information to start the machine; and
- (iii) **a signature value**

GPT

- The **GUID (Globally Unique Identifier) Partition Table** was introduced as a part of the Extensible Firmware Interface (EFI) initiative.
- GPT provides a mo**re flexible mechanism for partitioning disks** than the older Master Boot Record (MBR) partitioning scheme that has been common to PCs.
- A GPT disk has five major areas to it:





Parameter	MBR	GPT
Definition	MBR is an older partitioning scheme that divides the disk in two types of partitions – primary and extended	GPT is a modern partitioning scheme that theoretically doesn't have any limitations on number of partitions but is limited only by the OS.
Boot Loader used	Master Boot Program stored at first sector of the disk	A special partition called – EFI, especially allocated for loading the OS.
Addressing Scheme	It uses 32-bit addressing scheme limiting the storage capacity to at most 2TB	It uses 64-bit addressing capable of dealing with 9.4 zeta bytes of storage capacity
Compatibility	It is commonly used with Legacy BIOS based sytems	It is commonly used with modern UEFI based systems

Partition Limitation	It supports a maximum of 4 primary partitions	Upto 128 partitions
Data Integrity	It doesn't support error detection mechanism and is less secure	It supports CRC-32 error detection mechanism
Data Recovery	No built-in mechanism for data recovery in this scheme	GPT stores the identical copy at the end of the disk of the partition table which is allocated in the beginning of the disk. It can be used incase of data corruption for recovery.