

# Module - 1

## Introduction to Computers:

### 1. History of Computing.

#### A. Early Calculating Devices (Ancient to 1800s)

**Abacus (c. 2400 BC):** One of the earliest known tools for calculations, the abacus was used in many ancient civilizations, including Mesopotamia, China, and Greece. It consisted of a wooden frame with beads that could be moved to represent numbers and perform basic arithmetic operations.

#### B. Mechanical Calculators (1600s-1800s):

- **Blaise Pascal** invented the **Pascaline** in 1642, an early mechanical calculator that could perform addition and subtraction.

- **Gottfried Wilhelm Leibniz** improved on Pascal's design by creating a calculator that could also multiply and divide in 1673, known as the **Stepped Reckoner**.

### 2. The Concept of Modern Computing (1800s):

**Charles Babbage (1791–1871):** Often called the “father of the computer,” Babbage conceptualized the **Difference Engine** in 1822, a mechanical device designed to calculate polynomial functions.

Later, he proposed the more advanced **Analytical Engine(1837)**, which had many of the basic components of a modern computer, including an Arithmetic Logic Unit (ALU), control flow in the form of loops and conditional branching, and memory. However, Babbage was unable to complete a fully functioning machine due to technological limitations of his time.

### 3. Electromechanical Computers (1930s-1940s)

#### A. Z3 (1941):

Developed by **Konrad Zuse** in Germany. The Z3 was the world's first programmable digital computer. It used electromechanical relays to automate calculations. It was programmable with a limited set of instructions.

B. **Harvard Mark I (1944)**: Built by **Howard Aiken** and engineers at **IBM**, this large electromechanical computer was capable of basic arithmetic operations and was used by the U.S. Navy for ballistic calculations during World War II.

### 4. The Era of Electronic Computers (1940s-1950s)

**Colossus (1943-1944)**: Developed by **Tommy Flowers** during World War II, Colossus was the world's first programmable, digital, electronic computer. It was used by British codebreakers to decipher encrypted messages from the German military.

A. **ENIAC (1946)**: The **Electronic Numerical Integrator and Computer** was the first general-purpose electronic digital computer, designed by **John Mauchly** and **J. Presper Eckert**. It was capable of performing complex calculations much faster than electromechanical devices. However, ENIAC required manual rewiring to change its program, which limited its flexibility.

B. **Stored Program Concept**: This concept, introduced by **John von Neumann** in 1945, was revolutionary in computer design. Von Neumann suggested that both data and programs could be stored in a computer's memory, allowing the machine to be reprogrammed without the need for physical reconfiguration. This architecture, known as the **Von Neumann architecture**, is still the foundation of most modern computers.

## 5. The First Generation (1940s-1950s)

**Vacuum Tube Technology:** The first generation of computers (1940s to 1950s) used vacuum tubes for circuitry and magnetic drums for memory. These machines were large, expensive, and required a lot of electricity.

- Example: UNIVAC I(1951), the first commercially available computer in the U.S., used by the U.S. Census Bureau.

## 6. The Second Generation (1950s-1960s)

**Transistor Technology:**

The second generation of computers replaced vacuum tubes with transistors, making machines smaller, faster, and more energy-efficient. Transistors were invented by **John Bardeen, Walter Brattain, and William Shockley** in **1947**, but were widely adopted in the 1950s.

- Example: IBM 1401(1959), a successful transistor-based computer widely used for business applications.

## 7. The Third Generation (1960s-1970s)

**Integrated Circuits (ICs):** The third generation of computers began using integrated circuits, developed by **Jack Kilby** and **Robert Noyce** in the late 1950s. ICs allowed for even smaller, faster, and more reliable machines.

- Example: IBM System/360 (1964), one of the first computers to use integrated circuits. It was notable for being compatible across a range of applications, marking a shift toward more versatile machines.

## 8. The Fourth Generation (1970s-present)

**Microprocessors:** The fourth generation was marked by the development of the microprocessor, which integrated all the processing functions of a computer onto a single chip. Intel's 4004, introduced in 1971, was the first commercially available microprocessor, designed by **Ted Hoff** and others at Intel.

## **Personal Computers:**

The invention of microprocessors made personal computers (PCs) possible. Notable developments include:

- Altair 8800(1975): Often considered the first personal computer.
- Apple I and Apple II (1976-1977): Developed by Steve Jobs and Steve Wozniak, these early PCs helped launch the personal computing revolution.
- IBM PC(1981): IBM's entry into the personal computer market set standards that influenced the design of subsequent personal computers.

## **9. The Fifth Generation (Present and Beyond)**

**-Artificial Intelligence (AI):** The fifth generation of computers is characterized by the development of machines that incorporate artificial intelligence. These computers are designed to process and understand natural language, recognize images, and perform complex tasks that require human-like reasoning. Key advancements include:

- Supercomputers like IBM's Watson, which beat human champions on the game show **Jeopardy**, and AI-based systems like **Google's AlphaGo**, which defeated top human players in the game of Go.

- **Quantum Computing:** Although still in its early stages, quantum computers use the principles of quantum mechanics to perform calculations at unprecedented speeds, promising breakthroughs in fields such as cryptography, medicine, and complex simulations.

## Key Trends in Modern Computing

- A. **Cloud Computing:** The use of the internet to store and process data on remote servers, allowing for on-demand access to resources and applications.
- B. **Mobile Computing:** The rise of smartphones and tablets, powered by powerful processors like **ARM** chips, which have revolutionized how we interact with technology in our daily lives.
- C. **Internet of Things (IoT):** The increasing interconnectivity of devices, from smart thermostats to connected cars, is creating vast networks of data-driven interactions.
- D. **Artificial Intelligence & Machine Learning:** AI is becoming integrated into nearly every aspect of computing, from personal assistants (e.g Siri, Alexa) to predictive analytics and self-driving cars.

## Data Storage:

Data storage involves representing and storing information using bits, which are the fundamental units of data in computing. Here's an overview:

### 1. Bits and Their Storage

- A. **Bit:** A bit is a binary digit, either 0 or 1, which represents the smallest unit of data in computing. All forms of data (text, images, videos) are ultimately stored as a sequence of bits.
- B. **Byte:** 8 bits form a byte. Most data sizes are measured in multiples of bytes (kilobytes, megabytes, gigabytes, etc.).

### Storage Devices and Methods for Bits :

- 1. **Magnetic Storage:** Hard drives (HDDs) use magnetic materials to store data as bits, with the orientation of magnetic particles representing 0 or 1.
- 2. **Optical Storage:** CD, DVD, and Blu-ray store data by encoding bits as microscopic pits and lands on the disc surface, which are read by lasers.
- 3. **Solid-State Storage:** SSDs, USB drives, and memory cards store bits using flash memory, where electrical charges represent 0s and 1s.

4. **Cloud Storage:** Bits are stored in data centers on large servers, accessed via the internet, making it appear that data is remotely available.

## 2. Main Memory (Primary Memory)

Main memory, often called RAM (Random Access Memory), is a form of volatile storage used by computers to store data that is currently being processed. Unlike permanent storage (like hard drives), the data in RAM is lost when the computer is turned off.

1. **Dynamic RAM (DRAM):** The most common type of RAM, where each bit is stored in a tiny capacitor that must be periodically refreshed to maintain its data.
2. **Static RAM (SRAM):** Faster but more expensive than DRAM, SRAM doesn't require refreshing, making it ideal for cache memory in processors.
3. **Cache Memory:** A small, high-speed type of memory located close to the CPU. It stores frequently accessed data to speed up processing.

### Characteristics of Main Memory :

- **Volatile:** Data is lost when power is turned off.
- **Faster than secondary storage:** RAM is much faster than hard drives or SSDs, allowing for quick read/write operations.
- **Limited capacity:** It usually has less capacity than secondary storage (HDD/SSD).

Main memory plays a critical role in ensuring smooth execution of programs, storing temporary data and instructions required by the CPU for current tasks.

## Mass Storage:

Mass storage refers to systems or devices designed to store large amounts of data. These storage systems are typically used in enterprises, data centers, cloud computing environments, and personal computing to hold extensive data sets. Here is a detailed look at the various aspects of mass storage:

### 1. Types of Mass Storage Devices

Mass storage devices can be broadly categorized into different types based on their technology and use cases:

- **Hard Disk Drives (HDD):** HDDs are magnetic storage devices that use spinning platters and a moving read/write head to access data. They offer high capacity at

relatively low cost but are slower than modern alternatives due to mechanical moving parts.

- **Solid State Drives (SSD):** SSDs use flash memory, a type of non-volatile storage. They have no moving parts, making them faster and more durable than HDDs, but they are often more expensive per unit of storage.
- **Optical Storage:** This includes CDs, DVDs, and Blu-ray discs, which are less commonly used in modern mass storage but still relevant in some archival applications. These media store data using laser light to read and write information.
- **Tape Storage:** Tape drives use magnetic tape to store data and are often used for backup and archival purposes. Tape offers extremely high capacity at a low cost, though it has slower read/write speeds compared to HDDs and SSDs.
- **Network-Attached Storage (NAS):** NAS devices are dedicated file storage devices that connect to a network, allowing multiple users and devices to access data over a local area network (LAN). They are typically easy to manage and scale.
- **Cloud Storage:** Cloud storage allows data to be stored remotely on servers managed by third-party providers. This type of mass storage is highly scalable, and users can access their data via the internet. Examples include services like Amazon S3, Google Drive, and Microsoft Azure.
- **RAID Systems:** RAID (Redundant Array of Independent Disks) systems use multiple drives in various configurations to ensure redundancy, improve performance, and prevent data loss. Common RAID levels include RAID 0 (striping), RAID 1 (mirroring), RAID 5 (striping with parity), and RAID 6 (double parity).

## 2. Storage Technologies

- **Magnetic Storage:** HDDs and tape drives use magnetic technology to store data. Data is written to and read from a spinning platter or tape using magnetic fields. Though slower, magnetic storage is cost-effective and has been the standard for large capacity for many years.
- **Flash Storage:** SSDs use NAND flash memory cells to store data. This technology offers significantly faster read/write speeds compared to HDDs because data is

stored electronically rather than magnetically. SSDs also have lower latency and higher endurance.

- **Hybrid Storage:** Hybrid drives, or SSHDs (Solid State Hybrid Drives), combine HDD and SSD technologies. They store frequently accessed data on faster flash memory, while larger data sets are stored on magnetic platters, offering a balance between speed and cost.

### 3. Storage Capacities

- **HDD Capacities:** HDDs can range from 500 GB to several terabytes (TB). In enterprise environments, some specialized drives can go beyond 20 TB.
- **SSD Capacities:** SSDs typically range from 128 GB to 8 TB in consumer devices, but enterprise-grade SSDs can reach capacities of up to 100 TB, though the cost per gigabyte is higher.
- **Tape Capacities:** Modern tape drives, such as LTO (Linear Tape-Open) technology, can store up to 12 TB per cartridge (uncompressed) and more with compression.
- **Cloud Storage Capacities:** Cloud storage is highly flexible, offering virtually unlimited capacity. Users can scale their storage up or down as needed, paying based on the amount of storage used.

### 4. Performance and Speed

- **HDDs:** Typically offer read/write speeds of 100–150 MB/s, though this can vary based on factors like RPM (Revolutions Per Minute), ranging from 5,400 RPM to 15,000 RPM in enterprise-class drives.
- **SSDs:** Have significantly faster read/write speeds, often exceeding 500 MB/s for SATA-based SSDs and up to 7,000 MB/s for NVMe (Non-Volatile Memory Express) SSDs.
- **NAS:** The speed of a NAS device depends on the network connection (Ethernet, Wi-Fi, etc.) and the type of drives it uses (HDD, SSD). NAS over gigabit Ethernet can typically deliver speeds of 100 MB/s or more.

- **Tape:** Tape storage is relatively slow, with speeds generally ranging from 100 MB/s to 400 MB/s depending on the technology and compression. It is typically used for archival purposes rather than frequent data access.
- **Cloud Storage:** The performance of cloud storage depends on the internet connection speed and the cloud provider's infrastructure. Latency may be higher compared to local storage, but cloud storage can scale performance based on usage.

## 5. Redundancy and Reliability

- **RAID Arrays:** RAID configurations are used to protect against data loss and improve reliability. RAID 1 mirrors data across drives, while RAID 5 and RAID 6 use parity information to allow recovery from one or two disk failures, respectively.
- **Backup Solutions:** Mass storage systems are often used in conjunction with backup strategies to ensure data safety. Regular backups can be made to tape, cloud, or another mass storage device, depending on business needs.
- **Data Replication:** In enterprise environments, mass storage often involves data replication—either synchronously or asynchronously—across multiple locations to ensure redundancy and disaster recovery capabilities.

## 6. Use Cases for Mass Storage

- **Enterprise Data Centers:** Large-scale storage for databases, business applications, virtual machines, and file storage. Reliability, scalability, and redundancy are critical here.
- **Cloud Storage Solutions:** Cloud providers store massive amounts of user data, including personal files, backups, and enterprise-level applications. Customers can access these via APIs or web interfaces.
- **Media and Content Delivery:** Mass storage is crucial for media companies that need to store large amounts of video, audio, and other multimedia content. High throughput and fast access times are critical for content delivery.

- **Backup and Archiving:** Tape storage or cloud services are often used for long-term data retention and disaster recovery purposes, ensuring critical data is preserved in case of a failure.

## 7. Challenges of Mass Storage

- **Data Security:** As mass storage systems store vast amounts of sensitive information, they are often targets for cyberattacks. Encryption, access control, and regular monitoring are crucial to maintaining security.
- **Cost:** The cost of mass storage can be significant, especially when factoring in redundancy (e.g., RAID), backups, and high-speed access technologies like SSDs. Cloud storage introduces operational expenses (OPEX), where users pay on a subscription basis, which may grow with increasing data usage.
- **Scalability:** While cloud storage offers the easiest scalability, on-premise mass storage systems must be carefully planned to allow for future growth. Expanding storage infrastructure can involve significant investment in hardware, space, and energy.
- **Management Complexity:** Managing mass storage, especially in environments with mixed storage types (HDDs, SSDs, NAS, and cloud), can be complex. This involves monitoring storage utilization, ensuring redundancy, managing backups, and optimizing for performance.

### *Key Concepts in Mass Storage:*

1. **Capacity:** The total amount of data that a storage device can hold. It is typically measured in gigabytes (GB) or terabytes (TB).
2. **Read/Write Speed:** The speed at which data can be written to or read from a storage device. SSDs tend to be significantly faster than HDDs due to the absence of moving parts.
3. **Durability:** The ability of the storage device to withstand physical damage or wear over time. SSDs and flash-based storage tend to be more durable than HDDs due to the lack of mechanical components.

4. **Latency:** The delay between a request for data and the delivery of that data. Lower latency is better, and SSDs typically have much lower latency compared to HDDs.
5. **Portability:** The ease with which a storage device can be transported. USB drives, memory cards, and external drives are highly portable, while internal drives and NAS systems are less so.
6. **Redundancy and Backup:** Many storage systems, especially in enterprise settings, use redundant configurations (like RAID) to prevent data loss in case of hardware failure. Backups are essential for ensuring data safety.

**Note:** Mass storage is a fundamental aspect of computer systems, providing long-term data retention and access. The type of storage selected depends on the specific needs of the user, including capacity, speed, portability, and reliability. With advances in technology, storage devices continue to become faster, smaller, and more efficient.

### Binary Number system:

The number system with base 2 is called as Binary number system. It is widely used in digital system such as computer. It has two digits e.g. 0 and 1 , and are called as bits. The base

of this number system is 2.

Binary numbers are formed very similar to decimal numbers as-

0,1,10,11,100,101,110,111,- - - - ,1000,- - - - ,10000,- - - -

Binary number is indicated as 11102 and read as 'one- one -one-zero'

Decimal weights of binary digits are as shown in the table below.

-	-	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$	.	$2^{-1}$	$2^{-2}$	$2^{-3}$	$2^{-4}$	-	-
							↑						
							↓						
-	-	16	8	4	2	1	.	0.5	0.25	0.125	0.0625	-	-

Binary number can be

expressed as sum of powers of 2 as follows:

$$1101_2 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$$

Group of 4 bits is known as a 'Nibble' and that of 8 bits as a 'Bytes'. A group of bits simultaneously processed by a digital system such as computer is known as a 'word'

e.g. 8-bit word,16-bit word,32-bit word, etc.

Advantages of binary number system are:

- Bits can be used to designate the two voltage levels in digital Electronics.
- Binary arithmetic is simple compared to decimal arithmetic.
- It is also called as Natural or Straight binary code.

Binary Number Examples

$$- 11 = 1 \times 2^0 + 1 \times 2^1 = 3_{10}$$

$$- 101 = 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 4 + 1 = 5_{10}.$$

$$- 1001 = 1 \times 2^3 + 1 \times 2^0 = 8 + 1 = 9_{10}.$$

$$- 1100 = 1 \times 2^3 + 1 \times 2^2 = 8 + 4 = 12_{10}.$$

$$- 0.1 = 1 \times 2^{-1} = \frac{1}{2} = 0.5_{10}$$

$$- 0.111 = 1 \times 2^{-1} + 1 \times 2^{-2} + 1 \times 2^{-3} = 0.5 + 0.25 + 0.125 = 0.875_{10}$$

$$- 0.10001 = 1 \times 2^{-1} + 1 \times 2^{-5} = 0.5 + 0.03125 = 0.53125_{10}$$

$$- 1101.01 = 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^0 + 1 \times 2^{-2} = 8 + 4 + 1 + 0.25 = 13.25_{10}$$

$$- 11.001 = 1 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-3} = 2 + 1 + 0.125 = 3.125_{10}$$

$$- 10.0011 = 1 \times 2^1 + 1 \times 2^{-3} + 1 \times 2^{-4} = 2 + 0.125 + 0.0625 = 2.1875_{10}$$

$$- 11101 = 1 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^0 = 16 + 8 + 4 + 1 = 29_{10}.$$

Table below shows some decimal numbers with their equivalent numbers of other system

Decimal	Binary	Octal	Hexadecimal
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

## Binary to Decimal conversion: Steamline Method:-

Procedure:

- 1) Rewrite the given binary number with little space between the bits.
- 2) Write equivalent decimal weights below each bit.
- 3) Decimal weights corresponding to bits zero are cancelled out
- 4) Remaining weights are added and decimal point is appropriately placed to get decimal equivalent of a given binary number.

Ex.  $1011.1012 = (x)_{10}$

$$= 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2} + 1 \times 2^{-3}$$

$$= 8 + 2 + 1 + 0.5 + 0.125$$

$$1011.1012 = 11.625_{10}$$

Decimal to Binary conversion:(Double-Dabble Method)

Ex.  $(14.25)_{10} = (x)_2$

Procedure :

( for Decimal Integer)

- 1) Given decimal integer is progressively divided by 2
- 2) Remainders after each division are recorded on one side
- 3) This is continued till the division is not possible.(Quotient =0)
- 4) Remainders taken in reverse order i.e. last remainder as MSB and first as LSB forms the binary integer of the given decimal integer.

## 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:

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 =  $3\ 4\ 5 = 345_{oct}$

## 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

### Binary to Decimal

They say there are only 10 people in this world: those that understand binary and those that don't. Ha ha.

If you don't get that joke, you'll need a method to convert from binary to decimal. One method involves addition and multiplication.

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

Here is an example of converting 11100000000 binary to decimal:

Binary Digits Operation Decimal Result Operation Decimal Result

11100000000 +1 1 × 2 2

11000000000 +1 3 × 2 6

1000000000 +1 7 × 2 14

000000000 +0 14 × 2 28

00000000 +0 28 × 2 56

000000 +0 56 × 2 112

### Binary addition

works in the same way, except that only 0's and 1's can be used, instead of the whole spectrum of 0-9. This actually makes

binary addition much simpler than decimal addition, as we only need to remember the following:

0 + 0 = 0

0 + 1 = 1

1 + 0 = 1

1 + 1 = 10

As an example of binary addition we have,

101  
+101

a) To add these two numbers, we first consider the "ones" column and calculate  $1 + 1$ , which (in binary) results in 10. We

"carry" the 1 to the "tens" column, and leave the 0 in the "ones" column.

b) Moving on to the "tens" column, we calculate  $1 + (0 + 0)$ , which gives 1. Nothing "carries" to the "hundreds" column, and we leave the 1 in the "tens" column.

c) Moving on to the "hundreds" column, we calculate  $1 + 1$ , which gives 10. We "carry" the 1 to the "thousands" column, leaving the 0 in the "hundreds" column.

101  
+101  
1010

Another example of binary addition:

1011  
+1011  
10110

Note that in the "tens" column, we have  $1 + (1 + 1)$ , where the first 1 is "carried" from the "ones" column. Recall that in binary,

$1 + 1 + 1 = 10 + 1$   
 $= 11$

### Binary subtraction

is simplified as well, as long as we remember how subtraction and the base 2 number system. Let's first look at an easy example.

111  
- 10  
101

Note that the difference is the same if this was decimal subtraction.

Also similar to decimal subtraction is the concept of "borrowing." Watch as "borrowing" occurs when a larger digit, say 8, is subtracted from a smaller digit, say 5, as shown below in decimal subtraction.

35  
- 8  
27

For 10 minus 1, 1 is borrowed from the "tens" column for use in the "ones" column, leaving the "tens" column with only 2. The following examples show "borrowing" in binary subtraction.

```

10 100 1010
- 1 - 10 - 110
1 10 100

```

## Binary multiplication

is actually much simpler than decimal multiplication. In the case of decimal multiplication, we need to remember  $3 \times 9 = 27$ ,

$7 \times 8 = 56$ , and so on. In binary multiplication, we only need to remember the following,

$$0 \times 0 = 0$$

$$0 \times 1 = 0$$

$$1 \times 0 = 0$$

$$1 \times 1 = 1$$

Note that since binary operates in base 2, the multiplication rules we need to remember are those that involve 0 and 1 only. As

an example of binary multiplication we have 101 times 11,

```

101

```

```

x11

```

First we multiply 101 by 1, which produces 101. Then we put a 0 as a placeholder as we would in decimal multiplication, and

multiply 101 by 1, which produces 101.

```

101

```

```

x11

```

```

101

```

1010 <-- the 0 here is the placeholder

The next step, as with decimal multiplication, is to add. The results from our previous step indicates that we must

add 101 and 1010, the sum of which is 1111.

```

101

```

```

x11

```

```

101

```

```

1010

```

```

1111

```

## Binary division

is almost as easy, and involves our knowledge of binary multiplication. Take for example the division of 1011 into 11.

```

11 R=10

```

```

11)1011

```

```

-11

```

```

-11

```

```

101

```

-11

10 <-- remainder, R

To check our answer, we first multiply our divisor 11 by our quotient 11. Then we add its' product to the remainder 10, and compare it to our dividend of 1011.

11

x 11

11

11

1001 <-- product of 11 and 11

1001

+ 10

1011 <-- sum of product and remainder

The sum is equal to our initial dividend, therefore our solution is correct.

### Rules of Binary Addition

- $0 + 0 = 0$
- $0 + 1 = 1$
- $1 + 0 = 1$
- $1 + 1 = 0$ , and carry 1 to the next more significant bit

For example,

$00011010 + 00001100 = 00100110$  1 1 carries

$00011010 = 26(\text{base } 10)$

$+ 00001100 = 12(\text{base } 10)$

$00100110 = 38(\text{base } 10)$

$00010011 + 00111110 = 01010001$  1 1 1 1 1 carries

$00010011 = 19(\text{base } 10)$

$+ 00111110 = 62(\text{base } 10)$

$01010001 = 81(\text{base } 10)$

Note: The rules of binary addition (without carries) are the same as the truths of the XOR gate.

### Rules of Binary Subtraction

- $0 - 0 = 0$
- $0 - 1 = 1$ , and borrow 1 from the next more significant bit
- $1 - 0 = 1$
- $1 - 1 = 0$

For example,

$00100101 - 00010001 = 00010100$  0 borrows

$001100101 = 37(\text{base } 10)$

$- 00010001 = 17(\text{base } 10)$

$$00010100 = 20(\text{base } 10)$$

$$00110011 - 00010110 = 00011101 \text{ 0 10 1 borrows}$$

BCS-I SEM – I DIGITAL ELECTRONICS NCK Notes | Multiplication & Division

$$001101011 = 51(\text{base } 10)$$

$$-00010110 = 22(\text{base } 10)$$

$$00011101 = 29(\text{base } 10)$$

### Rules of Binary Multiplication

- $0 \times 0 = 0$
- $0 \times 1 = 0$
- $1 \times 0 = 0$
- $1 \times 1 = 1$ , and no carry or borrow bits

For example,

$$00101001 \times 00000110 = 11110110 \text{ 0 0 1 0 1 0 0 1} = 41(\text{base } 10)$$

$$\times 00000110 = 6(\text{base } 10)$$

$$00000000$$

$$00101001$$

$$00101001$$

$$0011110110 = 246(\text{base } 10)$$

$$00010111 \times 00000011 = 01000101 \text{ 0 0 0 1 0 1 1 1} = 23(\text{base } 10)$$

$$\times 00000011 = 3(\text{base } 10)$$

$$11111 \text{ carries}$$

$$00010111$$

$$00010111$$

$$001000101 = 69(\text{base } 10)$$

Note: The rules of binary multiplication are the same as the truths of the [AND](#) gate.

Another Method: Binary multiplication is the same as repeated binary addition;

add the multicand to itself the multiplier number of times.

For example,

$$00001000 \times 00000011 = 00011000 \text{ 1 carries}$$

$$00001000 = 8(\text{base } 10)$$

$$00001000 = 8(\text{base } 10)$$

$$+ 00001000 = 8(\text{base } 10)$$

$$00011000 = 24(\text{base } 10)$$

### Binary Division

Binary division is the repeated process of subtraction, just as in decimal division.

For example,

$$00101010 \div 00000110 = 00000111 \text{ 1 1 1} = 7(\text{base } 10)$$

$$110)001101010 = 42(\text{base } 10)$$

$$-110 = 6(\text{base } 10)$$

1 borrows

1 0 1 0 1

- 1 1 0

1 1 0

- 1 1 0

0

$10000111 \div 00000101 = 00011011 \ 1 \ 1 \ 0 \ 1 \ 1 = 27(\text{base } 10)$

$1 \ 0 \ 1 \ ) \ 1 \ 0 \ 0 \ 10 \ 0 \ 1 \ 1 \ 1 = 135(\text{base } 10)$

$- \ 1 \ 0 \ 1 = 5(\text{base } 10)$

1 1 1 0

- 1 0 1

BCS-I SEM – I DIGITAL ELECTRONICS NCK Notes | Multiplication & Division

1 1

- 0

1 1 1

- 1 0 1

1 0 1

- 1 0 1

0

### Notes

Binary Number System

System Digits: 0 and 1

Bit (short for binary digit): A single binary digit

LSB (least significant bit): The rightmost bit

MSB (most significant bit): The leftmost bit

Upper Byte (or nybble): The right-hand byte (or nybble) of a pair

Lower Byte (or nybble): The left-hand byte (or nybble) of a pair

Binary Equivalents

1 Nybble (or nibble) = 4 bits

1 Byte = 2 nybbles = 8 bits

1 Kilobyte (KB) = 1024 bytes

1 Megabyte (MB) = 1024 kilobytes = 1,048,576 bytes

1 Gigabyte (GB) = 1024 megabytes = 1,073,741,824 bytes

### One's Complementing

Before we discuss this representation further, we need to introduce two operations.

The first is called one's complementing or taking the one's complement of an integer. The operation can be applied to any integer, positive or negative. This operation simply reverses (flips) each bit. A 0-bit is changed to a 1-bit, a 1-bit is changed to a 0-bit.

The following shows how we take the one's complement of the integer 00110110.

Original pattern	0	0	1	1	0	1	1	0
After applying one's complement operation	1	1	0	0	1	0	0	1

The following shows that we get the original integer if we apply the one's complement operations twice.

### Two's Complementing

Original pattern	0	0	1	1	0	1	1	0
One's complementing once	1	1	0	0	1	0	0	1
One's complementing twice	0	0	1	1	0	1	1	0

### Two's Complementing

The second operation is called two's complementing or taking the two's complement of an integer in binary. This operation is done in two steps. First, we copy bits from the right until a 1 is copied; then, we flip the rest of the bits.

The following shows how we take the two's complement of the integer 00110100.

Original integer	0	0	1	1	0	1	0	0
	↓	↓	↓	↓	↓	↓	↓	↓
Two's complementing once	1	1	0	0	1	1	0	0
	↓	↓	↓	↓	↓	↓	↓	↓
Two's complementing twice	0	0	1	1	0	1	0	0

An alternative way to take the two's complement of an integer is to first take the one's complement and then add 1 to the result.

## Representing information as bit patterns

Representing information as bit patterns is a foundational concept in computer science, where all types of data are stored and processed using sequences of binary digits (bits). A bit is the smallest unit of data, which can be either a 0 or a 1. By arranging multiple bits in specific sequences, we can encode various forms of information, such as numbers, text, images, and sound, as binary data that computers can manipulate.

### 1. Numbers

**Binary Representation:** The simplest representation of numbers, where each bit represents a power of 2. For example, the decimal number 5 is represented as 101 in binary ( $4 + 1$ ).

- **Signed Numbers:** For negative numbers, computers often use “two’s complement,” which allows a bit pattern to represent both positive and negative integers.
- **Floating-Point Numbers:** Real numbers are represented using the IEEE 754 standard, which divides bits into sections for the sign, exponent, and mantissa, allowing for a wide range of values and decimals.

## 2. Text

**ASCII and Unicode:** Characters and text are represented by assigning each character a unique bit pattern. ASCII uses 7 or 8 bits, while Unicode (such as UTF-8) can use varying numbers of bits to support a larger set of characters, covering many languages and symbols worldwide.

## 3. Images

- **Bitmaps:** Each pixel in an image is represented by a bit pattern corresponding to its color. In grayscale images, fewer bits per pixel are needed (often 8 bits per pixel), whereas color images use more (like 24 bits per pixel for RGB).
- **Compressed Formats:** Formats like JPEG and PNG compress images by encoding patterns and reducing redundancy, but they still fundamentally store information as bit patterns.

## 4. Sound

### Digital Audio:

Sound waves are sampled at regular intervals and each sample is represented as a bit pattern. Higher sample rates and bit depth (e.g., 16-bit, 24-bit) improve sound quality by capturing more detail.

## 5. Video

**Frames and Compression:** Video is a sequence of images (frames), each represented by bit patterns. Video compression algorithms like H.264 store only the differences between frames to save space, but the fundamental data remains binary.

### Note:

In all cases, the type of data determines how the bit patterns are interpreted. Different encoding and compression techniques allow these patterns to store complex information efficiently, making it possible to represent vast amounts of diverse data using just 0s and 1s

# Data Manipulation

## Computer Architecture:

Computer architecture is a specification describing how computer software and hardware connect and interact to create a computer network. It determines the structure and function of computers and the technologies it is compatible with – from the central processing unit (CPU) to memory, input/output devices, and storage units.

## Components of Computer Architecture

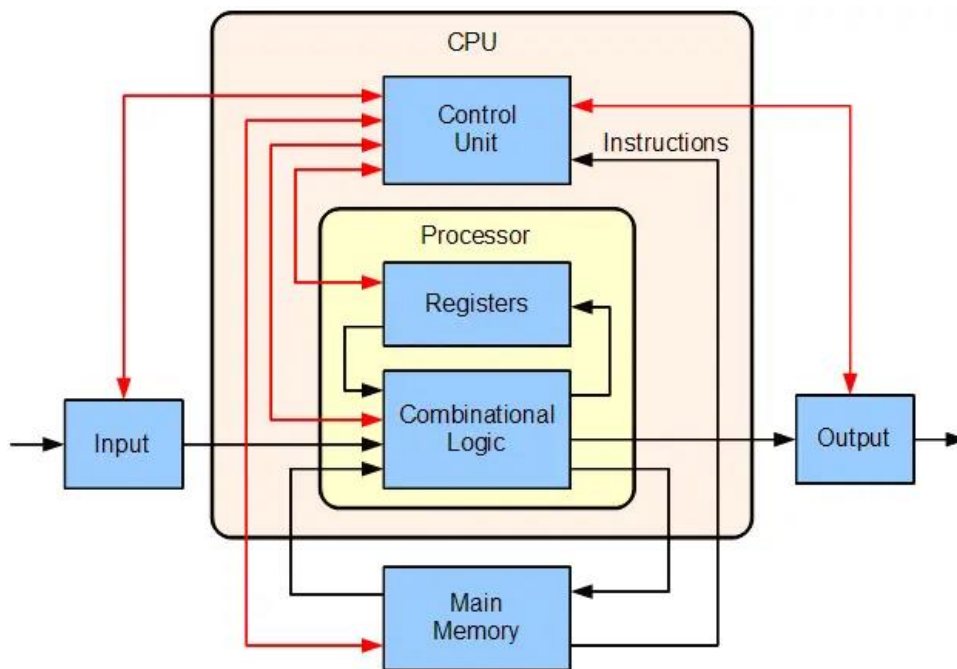
While computer architectures can differ greatly depending on the purpose of the computer, several key components generally contribute to its structure. These include:

1. **Central Processing Unit (CPU)** - Often referred to as the "brain" of the computer, the CPU executes instructions, performs calculations, and manages data. Its architecture dictates factors such as instruction set, clock speed, and cache hierarchy, all of which significantly impact overall system performance.
2. **Memory Hierarchy** - This includes various types of memory, such as cache memory, random access memory (RAM), and storage devices. The memory hierarchy plays a crucial role in optimizing data access times, as data moves between different levels of memory based on their proximity to the CPU and the frequency of access.
3. **Input/Output (I/O) System** - The I/O system enables communication between the computer and external devices, such as keyboards, monitors, and storage devices. It involves designing efficient data transfer mechanisms to ensure smooth interaction and data exchange.
4. **Storage Architecture** - This deals with how data is stored and retrieved from storage devices like hard drives, solid-state drives (SSDs), and optical drives. Efficient storage architectures ensure data integrity, availability, and fast access times.
5. **Instruction Pipelining** - Modern CPUs employ pipelining, a technique that breaks down instruction execution into multiple stages. This allows the CPU to

process multiple instructions simultaneously, resulting in improved throughput.

6. **Parallel Processing** - This involves dividing a task into smaller subtasks and executing them concurrently, often on multiple cores or processors. Parallel processing significantly accelerates computations, making it key to tasks like simulations, video rendering, and machine learning.

All of the above parts are connected through a system bus consisting of the address bus, data bus and control bus. The diagram below is an example of this structure:



## Machine languages

Machine languages, also called machine code or low-level languages, are basic in computer science and important to the study of how computers function. These languages are basically composed of binary code-1s and 0s-that a computer's CPU directly understands.

### 1. Definition of Machine Language

Binary Code: Machine language is composed of binary numbers, 0s and 1s, that are instructions and data.

**Direct CPU Communication:** It is the only language that a CPU can execute directly without translation or interpretation.

## **2. Properties of Machine Language**

**Hardware Specific:** Machine code is highly specific to a CPU's architecture. Each CPU has its own machine language, meaning programs written in one machine language generally cannot run on a different type of CPU without modification.

**Fast and Efficient:** Machine language can process quickly because instructions are carried out straight by the hardware without translation or interpretation.

**Difficult to Read and Write:** As machine language is in binary format, it is not easy to read or write for a human being with much chance of errors.

## **3. Machine Language Structure**

**Instruction Set:** Every CPU has an instruction set, a listing of all the operations a given CPU can execute such as addition, subtraction, data movement, etc.

**Operands and Opcodes:** Most machine language instructions consist of two components

**Opcode:** That portion of a machine code instruction that indicates the operation that is to be carried out, such as "add," "load," "store."

**Operand:** Tells the CPU which data items are to be used during an operation or which data location is to be employed for storing the outcome of an operation.

## **4. Advantages of Machine Language**

**High Speed:** As there is no translation process, the CPU executes the machine language very quickly.

**Precision and**

**Control:** Direct access to hardware provides the exact amount of control needed over resources.

## **5. Disadvantages of Machine Language**

**Complexity:** It is not easily understandable to the human mind because it does not resemble any of the higher-level languages in any way and is almost impossible to write and debug without using some kind of interpretation or simulation.

**Not Portable:** Programs written in machine language are not portable between different CPUs and have to be rewritten for each kind of hardware.

## **6. Uses of Machine Language**

**Embedded Systems:** Most embedded systems employ the use of machine language so as to optimize performance in specialized hardware workloads.

**Operating Systems and Kernels:** Sometimes, certain components of operating systems are implemented in machine language in an attempt to achieve maximum efficiency and

speed.

System-Level Programming: Critical applications where speed is of the essence can use machine language, including real-time systems and hardware device drivers.

## **7. Comparison with Assembly Language**

Assembly Language: Assembly is a low-level language that is very close to machine language but uses mnemonics (symbols) rather than binary codes, hence easier for humans to read and write.

**Assembler:** Assembler translates assembly language into machine language. It offers a connection between high-level programming languages and machine language. It provides a more readable and controlled environment.

**Note:** Machine language is important in understanding computer architecture and operations, even though it is seldom written directly by programmers; however, every compiled code executed by the CPU has machine language hidden beneath it.

## **Program execution Arithmetic and Logic Instructions.**

In program execution, arithmetic and logic instructions are crucial for performing calculations and making decisions.

These instructions are processed by the Arithmetic Logic Unit (ALU) of the CPU which executes basic operations like addition, subtraction, bitwise operations, and comparisons.

### **1. Arithmetic Instructions**

Arithmetic instructions perform basic mathematical operations on binary data. Common operations include:

**Addition:** Adds two binary values. If the result is too large for the available bits, an overflow flag is set.

**Subtraction:** Subtracts one binary value from another, using a similar process to addition but with two's complement (for signed numbers).

**Multiplication and Division:** The CPU may have instructions for multiplying and dividing, though complex multiplications or divisions can take more cycles or may need additional logic.

**Increment and Decrement:** Simple operations that add or subtract one from a value, often used in loops and counters.

## **2. Logic Instructions**

Logic instructions perform operations at the bit level, which can manipulate specific bits in a binary word. Common logical operations include:

- **AND:** Sets each bit in the result to 1 if the corresponding bits in both operands are 1.
- **OR:** Sets each bit in the result to 1 if the corresponding bit in either operand is 1.
- **XOR (Exclusive OR):** Sets each bit in the result to 1 if the corresponding bits in the operands are different.
- **NOT:** Inverts each bit in the operand (0 becomes 1, and 1 becomes 0).

## **3. Conditional and Comparison Instructions**

**Comparison:** Compares two values and sets flags (e.g., zero, carry, sign) in the status register, which the program can use to decide its next step.

**Conditional Jumps:** Use flags to control the flow of execution based on conditions like equal, not equal, greater than, etc.

Execution Flow of Arithmetic/Logic Instructions

1. **Fetch:** The CPU fetches the instruction from memory.
2. **Decode:** The CPU decodes the instruction to determine the operation (e.g., ADD, AND).
3. **Execute:** The ALU performs the operation on the data in specified registers or memory locations.
4. **Store:** The result is stored in a destination register, and the CPU proceeds to the next instruction.

These instructions enable the CPU to perform complex calculations, make decisions, and control program flow, forming the basis of all computational tasks in a system.