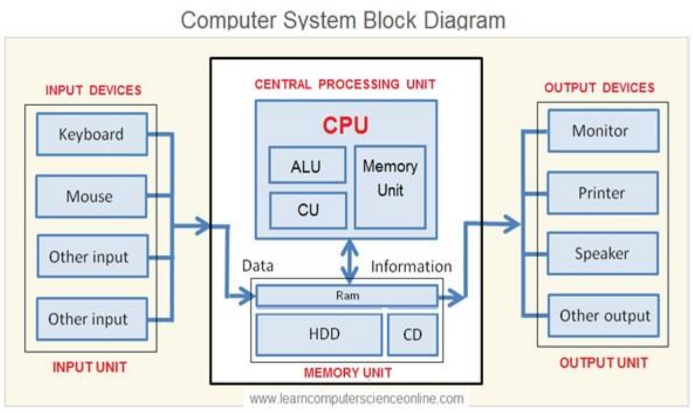
CHAPTER-0

Fundamental of a computer

0.1 Component of computer



- If we divide a computer into parts, we get five main components: input devices, output devices, memory, CPU, and the operating system
- Input devices, output devices, memory, and CPU are hardware, whereas the operating system (OS) is software (system software)
- Input **devices** – Used to give data and instructions to the computer (like keyboard, mouse).
- Output **devices** – Show the results from the computer (like monitor, printer).
- Memory – Stores data, both temporarily (RAM) and permanently (hard disk/SSD)
- Operating **System (OS)** – The Operating System (OS) is system software that acts as a bridge between application software and hardware, and also provides an interface for users to interact with the computer hardware

0.2 How computer Work

- CPU (Central Processing Unit) is called the brain of the computer. Internally, CPU **is connected to all hardware devices through physical wires called the *system bus***

- The **system bus (physical wire)** acts as a communication pathway between the CPU and other components like memory, input devices, output devices, and storage

- CPU decides what task each component should perform.
- Whenever a task needs to be done (such as fetching data from memory, saving to the hard disk, displaying something on the screen, or sending signals to the NIC (Network Interface Card) for network requests), the CPU sends a signal/instruction to that component through the system bus
- Then, the component completes the task and sends the result back to the CPU as a signal.
- However, the CPU does not generate signals on its own; it requires **instructions** to perform a task.
- This **set of instructions is called a program**.
- Programs are written in **high-level languages** such as Python, JavaScript, C, C++, etc.

If a computer understands only binary language, then why do we give instructions in high-level language?

Ans: Writing instructions directly in binary (low-level language) is very difficult and time-consuming for humans.
That's why we write programs in **high-level languages** like Python, Java, or C++.
These programs are then **translated (compiled or interpreted)** into binary (machine code) so the computer can understand and execute them.

0.3 Role of Operating system

- As we know, to perform any task in a computer we have to give an instruction to the CPU.
  For example, if we want to send a network request, the CPU must send signals to the **NIC (Network Interface Card)** through the system bus, and then the NIC does its job.
- But writing those low-level instructions directly would be very difficult, because we would need detailed knowledge of how every hardware device works.
- Instead, as application developers, we simply write **short, high-level instructions** in our program (often using a library).
  These instructions internally call the **detailed instructions of the Operating System**, which then communicates with the hardware on our behalf

- **So basically, in our application, we write *what* we want to do, not *how* to do it.**

Examples:

1 File Handling Instructions

- Reading and writing files, saving data, or opening documents.
- Example (Python):
- file = open ("data.txt", "w")
- file. Write ("Hello Students!")
- file. Close ()

👉 OS takes care of the disk access.

---

2. Memory Usage Instructions

- Defining variables, creating objects, using arrays, etc.
- Example (C):
- int age = 20;

👉 OS memory manager assigns actual RAM locations.

---

3. Network Communication Instructions (Network Request)

You want to send a request to google.com.

import requests

res = requests. Get("https://www.google.com")

print (res. status code)

- **You wrote only one line: requests. Get(...).**
- **Behind the scenes:**
    - **Python library → calls OS networking API**
    - **OS → instructs NIC card (network hardware)**
    - **NIC card → sends bits (0s and 1s) over the internet**

---

4. Input/Output Instructions

- Taking input from the user or showing output on screen.
- Example (Java):
- System.out.println("Enter your name:");

👉 OS decides how to use keyboard (input) and monitor (output).

---

6. User Interface (UI) Instructions

- Building screens, buttons, forms.
- Example (HTML):
- <button>Click Me</button>
- 👉 he **browser (app)** tells the **OS**:
  - "Draw a rectangle here."
  - "Put text inside."
- The OS then communicates with:
  - **Graphics hardware (GPU, video driver)** to draw on the screen.
  - **Input hardware (mouse, keyboard drivers)** to detect clicks.

👉 Result: Button appears on the screen, and we can click it.

CHAPTER-01

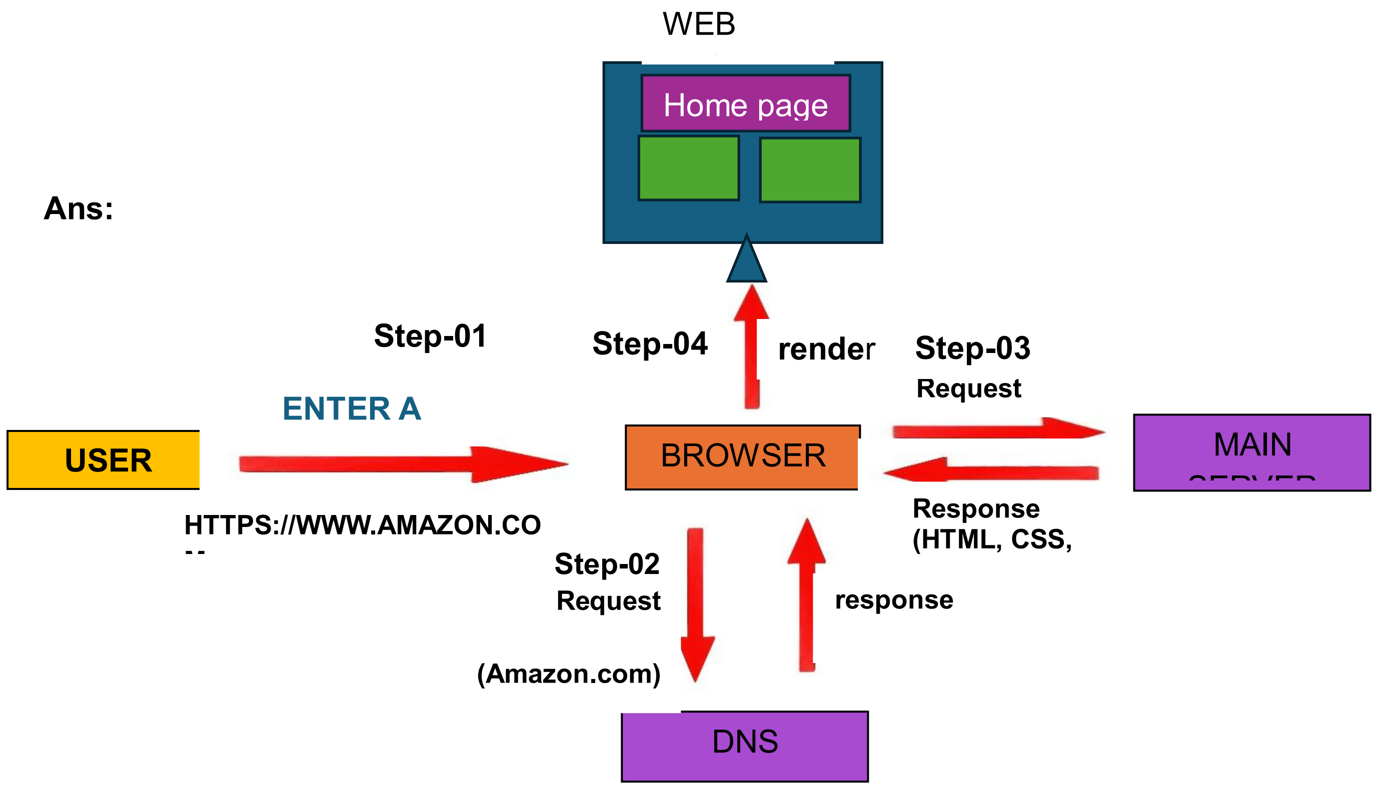
Q: What is a website?

Ans:
A **website** is a collection of **web pages** that are linked together under a common domain name.

- These pages can include text, images, videos, and other content.
- At the top of many websites, there is a **navigation bar (nav bar)** which allows users to move between different pages such as the **Home page, Contact page, About page, Services page**, etc.
- Websites are accessed using a web browser through the Internet.

How does a website work?

Ans:



Reqest-23.1.3.4, port (300), data

Response-html, CSS, JavaScript

1. **User Request –** When you type a website's URL (like www.amazon.com) in a browser, the browser sends a request to the DNS server
2. **DNS (Domain Name System) server**, which finds the IP address of the website's server.
3. **Server Response** – The request reaches the website's web server, where the website files (HTML, CSS, JavaScript, images, etc.) are stored.
4. **Data Transfer** – The server sends these files back to the browser through the Internet.
5. **Browser Rendering** – The browser takes those files, interprets them, and finally displays the webpage to the user.

👉 In short:
Browser → DNS → Server → Browser → User sees website

Parts of website

Frontend - parts of a website through which user interact, it is made in html, CSS and java script and we can also used frame work like React or Angular

Backend - backend is the server-side of a website.
It is the part that users don't see, but it powers everything happening behind the scenes.

Example:

Frontend (User side): You open Facebook and type your username + password.

Backend (Server side):

1. Takes your input.
2. Checks in the database if credentials are correct.
3. If correct → sends back your profile data.
4. If wrong → sends back "Invalid password."

Technologies used in Backend

- In backend we can write server-side logic in any one of them like in Java script (express framework) or java (spring boot), or Django (Python) or php

CHAPTER-02

INTRODUCTION TO HTML

2.1 What is html?

- html is a hypertext markup language, used to add like text, images, videos, link etc and give proper structure to its content on the webpage using tag

- tag-is a fundamental building block of an html document that instructs the browser on how to format and display the content of html element

- element-it consists opening tag, closing tag and it has some content b/w opening and closing tag this complete set is called element.

  exp- <p>hello I am learning html</p>

types of tag

- **closing tag-**closing tag exit in pair it has open tag with self-closing tag

  exp- <p></p>

- **open tag-**open tag doesn’t have self-closing tag

  exp- <img *src*="" *alt*="">, <br>

2.2 installation and set up

step-01-installing ide or code editor

- To write a html document we need a code editor like note pad or ide (integrated development environment) software like VS code, code blocks, IntelliJ IDEA etc

- in window we have default code editor called note pad but note pad has some limitation

```
-no autocomplete suggestion that is predictive text
-no colourful syntax highlighting
-not showing error if we make mistake
-we have to do everything manually
```

IDE SELECTION

- so, we should use ide like vscode, code blocks etc bcz
- it has autocompleted suggestion that is predictive text
- provide colourful syntax highlighting like keywords in blue, variable in green, error in red
- we can write, run, find debug and compiled the code in one place

step-02- installing Browser

- browser (google chrome, Firefox, micrsoftedge) helps in converting html document in to the webpage so that user can interact with it
- it allows changes to the webpage dynamically using java script
- makes the request to the server and get response from server

2.3 How to Download & Install Visual Studio Code (for Windows)

1. Open the Official Website or type vscode in browser
   Go to 👉 https://code.visualstudio.com

2. Click the big blue button:("Download for Windows")
   This will download the file VSCodeSetup.exe.

3. **Run the Installer**
   - Double-click on VSCodeSetup.exe.
   - Accept the license agreement.
   - Keep the default installation location.
   - ✅ Make sure to check:

- *Add to PATH*
- *Create a Desktop Icon*

Click Next → Install.

4. Finish Installation
After installation, click Finish.
Now you can open VS Code from the Desktop icon or Start menu.

4. Install Extensions

- On the left sidebar, click the Extensions icon (four squares).
  Search and install useful extensions like:
- Live Server (to run html document in browser).
- Prettier (to format code).
- ES7+ React/Redux Snippets (for React**).**

PRACTICE QUESTION

Ques-01 - *make a simple webpage and give title "first web page" and content "hey I have created my first webpage" by doing everything manually.*

Sol: -

```
  <!DOCTYPE html>
    <html>
        <head>
          <meta charset="UTF-8">
          <meta name="viewport" content="width=device-width, initial-
scale=1.0">
             <title>first web page</title>
        </head>


       <body>
          <p>😬 hey I am created my first web page</p>
        <body/>
   </html>
```

Ques02-what is html boiler code

Ans html boiler plate code is a basic must tag that has to be write in order to write a html document

```
  <!DOCTYPE html>
    <html>
        <head>
          <meta charset="UTF-8">
          <meta name="viewport" content="width=device-width, initial-
scale=1.0">
             <title>boiler code</title>
        </head>


       <body>

        <body/>
   </html>
```

*<!DOCTYPE html>, <html>, <meta>, <head>, and <body> are boilerplate code, you write them in **every HTML page**, even though the main content changes.*

Explanation of each tag in boiler code:

1. *<!DOCTYPE html> → Tells the browser this is an **HTML5 document**.*
2. *<html lang="en"> → Root element, language is English.*
3. *<head> → Contains meta information (not shown on page).*
   - *<meta charset="UTF-8"> → □ UTF-8 allows us to use **all languages + emojis** directly.*
   - *If we used **ASCII only**, non-English text or emojis wouldn't display properly.*
   - *<meta name="viewport" content="width=device-width, initial-scale=1.0"> → Makes site **responsive** on mobile.*
   - *<title> → Title shown on browser tab.*

4. *<body> → Actual **content** of the page (what users see).*

How to Generate boiler code in VS Code

1. *Type: exclamation mark(!) it gives emmit abbreviation*
2. *Click on emmit abbreviation → VS Code will auto-generate the full HTML boilerplate.*

Ques03- make a simple webpage by using boiler code and give title "first web page" and content "hey I have created my first webpage"

Ques-04 what is comment

*Ans A **comment** is a piece of text in code that is **ignored by the browser or compiler**.*
*It's written only for **developers** to explain the code, make notes, or temporarily disable some part of code*

Ques5- create a webpage showing" I am learning with techno globe" and use comments and some caseinsentive tags