Concept of Computer System

Introduction to computer system:

- Definition:

A computer is an electronic device that accepts input, processes it according to a set of instructions (called programs), stores the data, and provides output in a meaningful format.

- Body:

A computer performs four basic operations:

1. **Input** – It receives data and instructions through input devices like a keyboard and mouse.
2. **Processing** – The central processing unit (CPU) processes the data based on the given instructions.
3. **Storage** – It stores data either temporarily (RAM) or permanently (hard drive, SSD).
4. **Output** – It provides results to the user through output devices like a monitor or printer.

Computers work with high speed, accuracy, reliability, and can perform millions of calculations within seconds. They are used in various sectors such as education, government offices, business, hospitals, and research fields.

- **Conclusion:**

In conclusion, a computer is a powerful and versatile machine that has become an essential part of modern life. It helps in making tasks faster, easier, and more efficient.

Computer Software and its type:

❖ **Definition:**

Computer software refers to a set of instructions, data, or programs used to operate computers and perform specific tasks. It is the non-physical part of a computer system, in contrast to hardware, which refers to the physical components.

❖ Body:

There are **two main types of computer software**:

1. **System Software:**
   - ✓ This software manages and controls the hardware so that application software can function.
   - ✓ Examples: Operating systems (Windows, Linux), device drivers, and utility programs.
2. **Application Software:**
   - ✓ This is user-oriented software designed to perform specific tasks or solve problems.
   - ✓ Examples: Microsoft Word, Excel, web browsers, media players.

◆ Additionally, software can also be classified as:

- **Proprietary Software:** Licensed and sold by companies (e.g., Microsoft Office).
- **Open-source Software:** Free to use and modify (e.g., Linux, LibreOffice).

❖ **Conclusion:**

In conclusion, computer software is essential for the functioning and usability of a computer system. It helps users interact with hardware and perform various tasks efficiently, making it a core part of modern computing.

Difference Between System Software and Application Software

System Software	Application Software
It is designed to manage the resources of the computer system, like memory and process management, etc.	It is designed to fulfill the requirements of the user for performing specific tasks.
Written in a low-level language.	Written in a high-level language.
Less interactive for the users.	More interactive for the users.
System software plays vital role for the effective functioning of a system.	Application software is not so important for the functioning of the system, as it is task specific.
It is independent of the application software to run.	It needs system software to run.

Development of computer languages and its type

- Definition:

Computer languages are sets of instructions, commands, and syntax used to communicate with a computer. Their evolution has been driven by the need for more efficient, human-readable, and powerful ways to program.

1. Development of Computer Languages:

The development of computer languages can be broadly categorized into generations:

- **First Generation (1GL) - Machine Language:**
    - **Period:** Early 1940s - 1950s.
    - **Description:** Directly understood by the CPU, consisting of binary code (0s and 1s).
    - **Characteristics:** Extremely difficult to program, machine-dependent, error-prone, and time-consuming.
    - **Example:** 00101101 11010010 (representing an instruction).
- **Second Generation (2GL) - Assembly Language:**
    - **Period:** 1950s - 1960s.
    - **Description:** Uses mnemonics (symbolic codes) to represent machine language instructions. An assembler translates assembly code into machine code.
    - **Characteristics:** Easier to program than machine language, still machine-dependent, faster execution due to direct hardware interaction.
    - **Example:** ADD R1, R2 (add contents of Register 2 to Register 1).
- **Third Generation (3GL) - High-Level Languages:**
    - **Period:** 1950s onwards.
    - **Description:** Uses statements resembling human language (English) and mathematical notation. A compiler or interpreter translates these languages into machine code.
    - **Characteristics:** Machine-independent, easier to learn and write, less prone to errors, higher abstraction, and improved programmer productivity.
    - **Examples:** FORTRAN, COBOL, BASIC, C, C++, Java, Python.

- **Fourth Generation (4GL) - Very High-Level Languages:**
    - **Period:** 1970s onwards.
    - **Description:** Designed to be even closer to human language than 3GLs, often used for specific tasks like database management, report generation, and web development. Focus on "what to do" rather than "how to do it."
    - **Characteristics:** Non-procedural (or less procedural), rapid application development (RAD), reduced coding effort.
    - **Examples:** SQL, MATLAB, various scripting languages (e.g., Perl, Python for specific tasks), domain-specific languages.
- **Fifth Generation (5GL) - Natural Language/AI Languages:**
    - **Period:** 1980s onwards (under continuous research and development).
    - **Description:** Aims to allow computers to solve problems using constraints and knowledge bases, often associated with artificial intelligence and expert systems.
    - **Characteristics:** Still largely in research, focus on declarative programming, parallel processing, and natural language processing.
    - **Examples:** Prolog, OPS5, Mercury.

2. Types of Computer Languages:

Based on their level of abstraction and execution, computer languages are primarily categorized into:

- **Low-Level Languages:**
    - **Description:** Closest to the computer's hardware, providing minimal abstraction. Directly interact with memory and processor registers.
    - **Includes:** Machine Language and Assembly Language.

- **Advantages:** High speed, efficient memory utilization, direct hardware control.
- **Disadvantages:** Difficult to program, machine-dependent, complex for large applications.

- **High-Level Languages:**
  - **Description:** More abstract and human-readable, designed to be machine-independent. Use compilers or interpreters for translation.
  - **Includes:** 3GLs, 4GLs, and 5GLs.
  - **Advantages:** Easier to learn and write, faster development, portable across different systems, better for complex applications.
  - **Disadvantages:** Slower execution (due to translation process), less direct hardware control.

❖ Conclusion:

The progression of computer languages from binary machine code to highly abstract, human-like interfaces has fundamentally transformed computing. Each generation has addressed limitations of its predecessor, leading to more accessible, powerful, and efficient programming tools. This evolution continues to drive innovation in software development, enabling increasingly complex and intelligent applications.

Translators and Its type (Assembler, Compiler, Interpreter)

✅ Definition:

- A translator is a system software that converts a program written in one programming language (source code) into another programming language (object code or machine code).
- This conversion is essential because computers can only understand and execute instructions in their native machine language (binary 0s

and 1s), while programmers typically write code in higher-level, more human-readable languages.

❖ **Body Part:**

There are three main types of translators:

1. Assembler:

- Converts assembly language into machine code.
- It works line by line and is used for low-level programming.
- Example: Converting MOV A, B into binary instructions.

2. Compiler:

- Converts the entire high-level language program into machine code at once.
- It shows all errors after translation is complete.
- Faster during execution.
- Examples: GCC for C, javac for Java.

3. Interpreter:

- Translates and executes high-level language line by line.
- Slower than compiler but easier for debugging.
- Examples: Python Interpreter, PHP Interpreter.

✅ **Conclusion:**

In conclusion, translators play a vital role in programming by bridging the gap between human-readable code and machine-level code. Assembler, compiler, and interpreter each serve specific purposes based on the programming language and execution needs.

Linker and Loader:

✅ Definition:

Linker and **Loader** are system programs that play important roles in the process of program execution.

- A **Linker** combines multiple objects files and resolves references to create a single executable file.
- A **Loader** loads the executable file into memory and prepares it for execution.

✅ Body:

❖ Linker:

- ✓ Combines multiple **object files (.obj or. o)** into one executable program.
- ✓ Resolves **external references** (e.g., calling a function from another file or library).
- ✓ Example: Linking main.obj, math.obj, and io.obj into one .exe file.

Functions of Linker:

- ✓ Symbol resolution
- ✓ Address binding
- ✓ Library linking (e.g., linking standard libraries)

❖ Loader:

- ✓ Loads the executable file created by the linker into **main memory (RAM)**.
- ✓ Assigns memory addresses and starts program execution.

Functions of Loader:

- ✓ Allocates memory
- ✓ Loads code and data

- ✓ Starts execution by jumping to the program's starting address

✅ **Conclusion:**

In conclusion, the linker and loader are essential tools in the execution process. The linker prepares the program by combining files and resolving references, while the loader loads the program into memory and starts its execution. Without them, modern software development would not be possible.

Basic computer architecture and its components

- ❖ Definition:

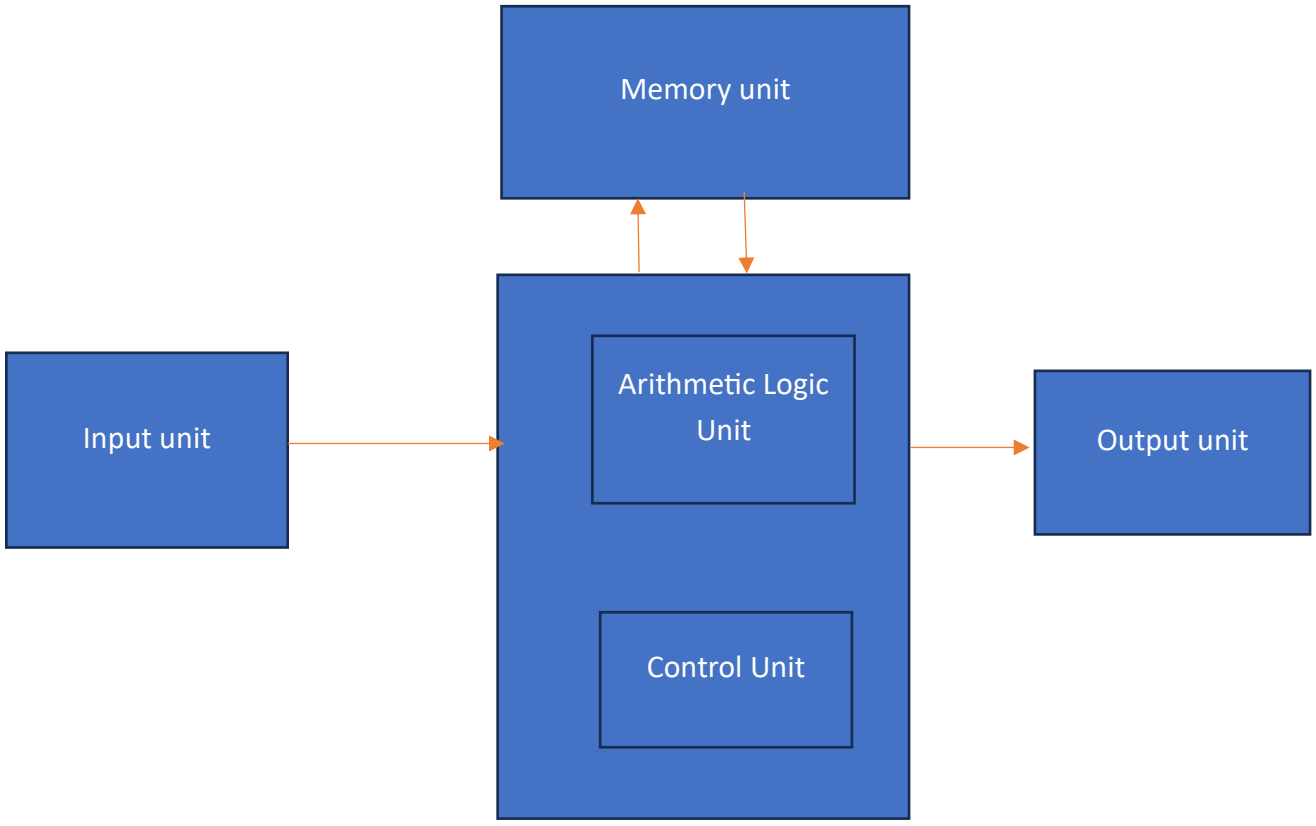
Computer Architecture refers to the design and organization of a computer's core components and how they interact to perform tasks. It defines how the computer system is structured and how data flows within the system.

Main Components

A basic computer architecture typically includes the following **main units**:

1. **Input Unit**
   - Allows users to enter data and instructions into the computer.
   - Examples: Keyboard, Mouse, Scanner.
2. **Central Processing Unit (CPU)**
   - Often called the **brain of the computer**. It processes all instructions.
   - It has two main parts:
     - **a. Arithmetic and Logic Unit (ALU):** Performs calculations and logical comparisons.
     - **b. Control Unit (CU):** Directs the flow of data and controls the operations of all components.

3. **Memory Unit (Storage)**
    - Stores data and instructions temporarily or permanently.
    - **Primary Memory (RAM & ROM):** Used during processing.
    - **Secondary Memory (Hard Drive, SSD):** Used for long-term storage.
4. **Output Unit**
    - Converts processed data into human-readable form.
    - Examples: Monitor, Printer, Speaker.



Conclusion:

Computer architecture forms the foundation of how a computer operates. Understanding its components input, output, CPU, and memory is essential to grasp how data is processed and results are produced efficiently.

Expansion Cards and Peripherals

Definition:

Expansion cards and **peripherals** are additional hardware components used to **enhance** the functionality of a computer system.

- **Expansion Cards** are installed inside the computer.
- **Peripherals** are usually external devices connected to the computer.

Body:

- **Expansion Cards:**

Expansion cards are circuit boards inserted into the motherboard to add new features or improve performance.

Common types:

1. **Graphics Card (GPU):** Enhances video/graphics processing (used in gaming, design).
2. **Sound Card:** Improves audio output/input.
3. **Network Interface Card (NIC):** Enables wired or wireless internet connection.
4. **TV Tuner Card:** Lets users watch and record TV on the computer.

- **Peripherals:**

Peripherals are **external** devices that connect to the computer to perform input, output, or storage functions.

Types of peripherals:

- **Input Devices:** Keyboard, Mouse, Scanner, Webcam
- **Output Devices:** Monitor, Printer, Speaker
- **Input/Output Devices:** Touchscreen, External Drives, USB Devices

Conclusion:

Expansion cards improve a computer's internal capabilities, while peripherals allow users to interact with the system or expand its functionality. Both are essential in customizing and enhancing a computer's performance.

UPS (Uninterruptible Power Supply):

Definition:

A UPS, or Uninterruptible Power Supply, is an electrical apparatus that provides emergency power to a load when the input power source, typically the utility mains, fails. It differs from an auxiliary or emergency power system or standby generator in that it provides near-instantaneous protection from input power interruptions by supplying power from batteries and associated circuitry.

Body Part:

The primary function of a UPS is to protect electronic equipment from power disturbances. It achieves this through several key mechanisms and components:

Primary Functions:

- **Power Outage Protection:** Provides continuous power from its internal battery when the main power supply is cut off, allowing connected devices to continue operating for a short period. This is crucial for saving work and performing a graceful shutdown.
- **Voltage Regulation:** Most UPS units include features to regulate voltage fluctuations (sags, swells, brownouts). They can boost low voltage or trim high voltage to ensure a stable power supply to connected equipment.
- **Surge Protection:** Protects sensitive electronics from power surges and spikes that can cause significant damage.
- **Frequency Regulation:** Ensures a stable frequency (e.g., 50 Hz or 60 Hz) for the output power, which is important for some sensitive equipment.

Key Components:

- **Battery:** The core component, storing electrical energy. Typically, sealed lead-acid or lithium-ion batteries are used. The battery capacity determines how long the UPS can supply power during an outage.
- **Inverter:** Converts the DC (Direct Current) power from the battery into AC (Alternating Current) power, which is what most electronic devices use.
- **Rectifier/Charger:** Converts AC power from the utility into DC power to charge the battery when mains power is available.
- **Transfer Switch:** Automatically switches the power source from the utility mains to the battery/inverter when a power failure is detected.

Types of UPS Systems:

- **Standby/Offline UPS:** The simplest and most common type for home/office use. The battery inverter is off during normal operation and only turns on when power fails.
- **Line-Interactive UPS:** Offers better protection than standby, as the inverter is always connected to the output, allowing for voltage regulation without switching to battery mode.
- **Online/Double-Conversion UPS:** Provides the highest level of protection. The utility AC power is continuously converted to DC to charge the battery, and then the battery's DC power is constantly converted back to AC for the output. This isolates the load from any incoming power disturbances.

Conclusion:

A UPS is a critical investment for protecting valuable electronic equipment and ensuring data integrity, especially in regions prone to power fluctuations or outages. By providing immediate backup power, voltage stabilization, and surge protection, it safeguards devices like computers, servers, and networking equipment from potential damage and data loss, thereby ensuring business continuity and personal productivity.

Environmental Conditioning Requirements for Computer Installation

Definition:

Environmental conditioning for computer installation refers to establishing and maintaining specific atmospheric and physical conditions within a computer room, server room, or even a typical office space to ensure the optimal performance, reliability, and longevity of computer hardware. Computers are sensitive electronic devices that can be adversely affected by extreme or fluctuating environmental factors.

Body Part:

The critical environmental factors that need to be controlled for proper computer installation include:

A. Temperature Control:

- **Requirement:** Computers generate a significant amount of heat during operation. Excessive heat can lead to overheating, component damage, reduced lifespan, and system instability (e.g., crashes, throttling). Conversely, extremely low temperatures can cause condensation.
- **Optimal Range:** For general computer installations and server rooms, an ambient temperature range of **18◦C to 27◦C (64◦F to 80◦F)** is generally recommended. A narrower range of **20◦C to 24◦C (68◦F to 75◦F)** is often considered ideal for critical server environments.
- **Solution:** Proper ventilation and dedicated cooling systems (e.g., Air Conditioning units like CRAC - Computer Room Air Conditioner) are essential to dissipate heat and maintain stable temperatures. Hot-aisle/cold-aisle containment strategies are used in data centers.

B. Humidity Control:

- **Requirement:** Both excessively high and extremely low humidity levels can be detrimental to computer hardware.
  - **High Humidity:** Can lead to condensation on circuit boards, causing short circuits, corrosion, and eventual component

failure. It can also cause dust to clump and clog cooling systems.
        - **Low Humidity:** Increases the risk of Electrostatic Discharge (ESD), a static electricity spark that can irreversibly damage sensitive electronic components.
    - **Optimal Range:** A relative humidity level between **40% and 60% (non-condensing)** is typically recommended.
    - **Solution:** Dehumidifiers are used in high-humidity environments, while humidifiers might be necessary in very dry conditions. Proper airflow also helps prevent localized humidity buildup.

C. Dust and Air Quality Control:

- **Requirement:** Dust particles can accumulate on components, acting as an insulating layer that traps heat, impeding cooling. They can also clog fans, vents, and moving parts, leading to overheating and mechanical failures. Airborne contaminants (like corrosive gases) can also damage circuitry over time.
- **Solution:**
    - Maintain a clean environment with regular cleaning.
    - Use air filtration systems (e.g., HVAC systems with appropriate filters) to minimize dust ingress.
    - Consider raised floors in server rooms to manage airflow and prevent dust accumulation under equipment.
    - Seal off computer rooms from general office areas where possible.

D. Power Quality and Stability:

- **Requirement:** Computers are highly susceptible to power fluctuations, surges, sags, and complete outages. Unstable power can cause data corruption, hardware damage, and system downtime.
- **Solution:**
    - **Uninterruptible Power Supply (UPS):** Provides immediate, temporary backup power during outages and regulates incoming power.
    - **Surge Protectors:** Protect against sudden spikes in voltage.
    - **Voltage Regulators:** Maintain a consistent voltage level.

- **Backup Generators:** For critical installations (like data centers), generators provide long-term power in case of extended outages.
- **Proper Grounding:** Essential to dissipate electrical surges and static electricity safely.

E. Physical Security and Location:

- **Requirement:** The physical location should be secure and free from vibrations, direct sunlight, and potential water leaks.
- **Solution:**
  - Choose a stable and dry location away from windows, plumbing, or chemical storage.
  - Ensure adequate ventilation space around equipment.
  - Implement physical security measures like restricted access, surveillance cameras, and fire suppression systems (e.g., inert gas systems instead of water sprinklers for server rooms).

Conclusion:

Maintaining precise environmental conditions is fundamental to the reliable and efficient operation of computer systems. By diligently controlling temperature, humidity, air quality, and power supply, and ensuring appropriate physical security, organizations can significantly reduce the risk of hardware failures, data loss, and costly downtime, thereby maximizing the lifespan and performance of their computer installations.

Computer Performance Testing Methods

Definition:

Computer performance testing is a non-functional testing process that evaluates the speed, responsiveness, and stability of a computer system, component, or application under various workloads. Its primary goal is to ensure the system meets performance requirements and identifies bottlenecks, ensuring optimal user experience and system efficiency.

Body Part:

Performance testing employs various methods, each designed to assess different aspects of a system's behavior under specific conditions:

A. Load Testing:

- **Purpose:** To measure the system's performance under **normal and expected user loads**. It simulates a realistic number of concurrent users or transactions to understand how the system behaves under anticipated conditions.
- **Objective:** To confirm that the system can handle the expected traffic while maintaining acceptable response times and throughput. It helps identify potential bottlenecks under typical usage.
- **Metrics Monitored:** Response time, throughput (transactions per second), resource utilization (CPU, memory, disk I/O), error rates.
- **Example:** Simulating 1,000 concurrent users Browse an e-commerce website during normal business hours.

B. Stress Testing:

- **Purpose:** To evaluate the system's performance and stability under **extreme and unusual load conditions**, often pushing it beyond its normal operational limits.
- **Objective:** To identify the system's "breaking point" – the load at which it starts to degrade significantly or fail. It also assesses how gracefully the system recovers from such stressful situations.
- **Metrics Monitored:** System stability, error handling, resource utilization at peak load, recovery time.
- **Example:** Continuously increasing the number of users to 5 times the expected maximum to see when the server crashes or becomes unresponsive.

C. Endurance Testing (Soak Testing):

- **Purpose:** To assess the system's performance and stability over an **extended period** under a sustained, normal load.

- **Objective:** To detect issues that might only appear after prolonged use, such as memory leaks, performance degradation over time, resource exhaustion, or database connection pooling issues.
- **Metrics Monitored:** Memory usage patterns, consistent response times and throughput over hours/days, resource utilization trends.
- **Example:** Running a banking application with a consistent load for 24-48 hours to check for any gradual performance decline or resource depletion.

D. Spike Testing:

- **Purpose:** To evaluate the system's ability to handle **sudden, sharp increases and decreases in load** over a short period.
- **Objective:** To determine if the system can cope with abrupt traffic surges (e.g., during a flash sale, viral event, or breaking news) without crashing or significantly degrading performance, and how quickly it recovers.
- **Metrics Monitored:** Recovery time, error rates during and immediately after the spike, system responsiveness.
- **Example:** Simulating a sudden jump from 100 users to 10,000 users within a few minutes, then dropping back down.

E. Volume Testing:

- **Purpose:** To test the system's performance when processing or dealing with **large amounts of data**.
- **Objective:** To identify performance issues related to database size, file uploads, data retrieval, or batch processing, ensuring the system handles data effectively.
- **Metrics Monitored:** Database query response times, data processing speed, disk I/O.
- **Example:** Testing a data analytics application's performance when processing a database containing terabytes of information.

F. Scalability Testing:

- **Purpose:** To determine the system's capability to **scale up or down** to accommodate an increasing or decreasing user load or data volume.

- **Objective:** To find the optimal strategy for expanding the system (e.g., adding more servers, CPU, RAM) to maintain performance as demand grows.
- **Metrics Monitored:** Performance metrics as resources are added or removed (e.g., requests per second vs. number of servers).

Conclusion:

Computer performance testing is critical for delivering robust and efficient software systems. By systematically applying methods like load, stress, endurance, spike, volume, and scalability testing, developers and QA teams can thoroughly evaluate a system's behavior under diverse conditions. This proactive approach helps in identifying and resolving performance bottlenecks before deployment, leading to improved user satisfaction, system reliability, and business continuity.

Model Questions (5 Marks Each)

1. **Define computer system. Explain the basic components of a computer system with a neat diagram.**
2. **What is computer software? Explain different types of computer software with examples.**
3. **Discuss the evolution of computer programming languages. Explain different generations of computer languages with examples.**
4. **What are translators in computer systems? Differentiate between Assembler, Compiler, and Interpreter with suitable examples.**
5. **Write short notes on: (Any Two)**
   a) Linker
   b) Loader
   c) Expansion Cards
6. **Explain various hardware components of a computer with suitable classification.**
7. **Classify input and output devices with examples. Write the functions of any two input and two output devices.**
8. **What are expansion cards? List and explain any three types of expansion cards used in computers.**

9. **Describe the environmental and power requirements for proper computer installation.**
10. **What is UPS? Explain its importance in computer systems. Also, mention different types of UPS used.**
11. **What is meant by computer performance testing? Describe the common parameters used to test computer performance.**