NOTES

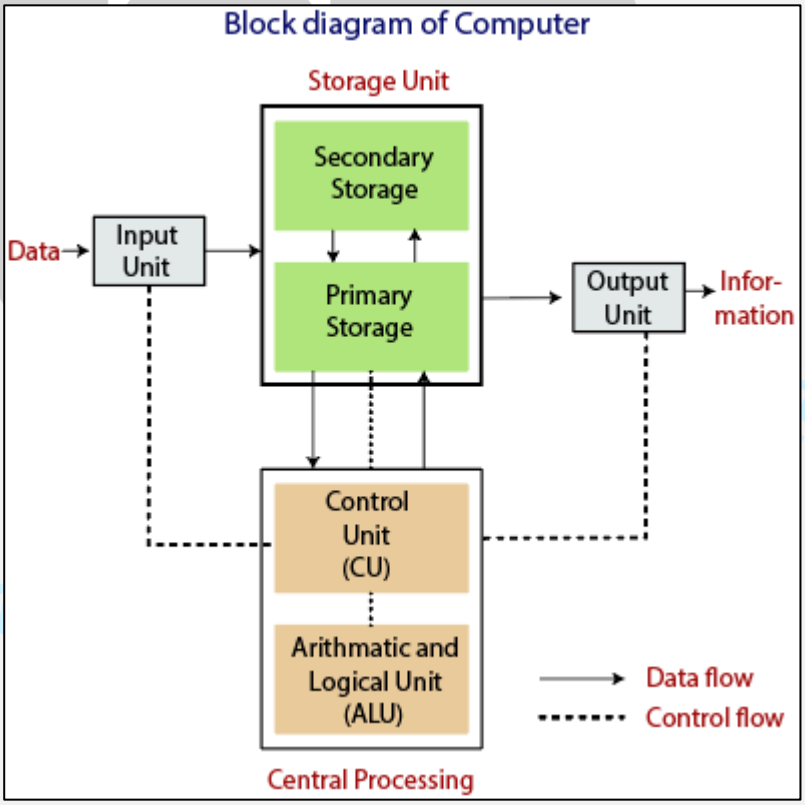
Computer Fundamentals and Programming

Unit 1: Computer Fundamentals

- **Basic Computer Organizations:**

This part of the chapter contain the overview introduction to the computer and the parts of the computer and how it will perform the different functions or we can say processes.

- **Block diagram of Computer System:**



NOTES

A block diagram of a computer represents the major components of a computer system and how they are interconnected. Here's an explanation of each component:

1. Input Unit:

- The input unit is responsible for taking data and instructions from external sources and transferring them into a form that the computer can understand. This can include input devices like keyboards, mice, scanners, and microphones.
- The data and instructions provided by users or other devices are the raw materials for the computer's processing.

2. Storage Unit (Memory):

- The storage unit, often referred to as memory, is where the computer temporarily holds data, programs, and intermediate results during its operation. It is divided into two main types:
  - RAM (Random Access Memory): This is volatile memory and is used for temporarily storing data and program instructions that the CPU (Central Processing Unit) needs during its operations.
  - ROM (Read-Only Memory): This is non-volatile memory and typically stores the computer's firmware and startup instructions.

3. Control Unit:

- The control unit is like the "brain" of the computer. It coordinates and controls the operations of all the other components. It manages the execution of program instructions and controls the data flow between the other units.
- The control unit fetches instructions from memory, decodes them, and executes them in a sequence.

4. Arithmetic and Logical Unit (ALU):

- The Arithmetic and Logical Unit is responsible for performing mathematical operations (addition, subtraction, multiplication, division) and logical operations (e.g., comparisons) on data.
- It works in conjunction with the control unit to execute instructions and manipulate data according to the program's requirements.

5. Output Unit:

- The output unit is responsible for taking the results of the computer's processing and presenting them in a form that is understandable to humans or other external devices. Common output devices include monitors, printers, and speakers.
- Output devices convert the digital information generated by the computer into a format that users can interpret.

- **Functional Units of Computer:**

1)CPU (Central Processing Unit):

The CPU is the brain of the computer. It is responsible for executing instructions and performing calculations. The CPU is made up of several components, including the following:

- Arithmetic Logic Unit (ALU): The ALU is responsible for performing arithmetic and logical operations. It is the part of the CPU that actually does the "computing".
- Control Unit (CU): The CU is responsible for fetching instructions from memory, decoding them, and executing them. It tells the ALU what to do and when to do it.
- Registers: Registers are small, high-speed storage units that are used to hold data temporarily. The CPU uses registers to store operands, intermediate results, and addresses.

2) I/O (Input/Output) Devices:

I/O devices allow the computer to interact with the outside world. They are responsible for sending data to and from the computer. Some common I/O devices include the following:

- Keyboard: The keyboard allows the user to enter text into the computer.
- Mouse: The mouse allows the user to point and click on things on the screen.
- Monitor: The monitor displays the output of the computer.
- Printer: The printer prints out text and graphics.
- Modem: The modem allows the computer to connect to the internet.

3) Hardware:

NOTES

Hardware is the physical components of a computer. It includes the CPU, memory, I/O devices, and other physical parts of the computer. Hardware is often contrasted with software, which is the instructions that tell the hardware what to do.

4) Software

It is the instructions that tell the hardware what to do. It includes the operating system, applications, and other software programs. Software is stored in memory and executed by the CPU.

Introduction to Program Planning Tools:

Program planning tools, such as algorithms and flowcharts, are essential elements in the software development and problem-solving process. They help developers and programmers design, visualize, and communicate the logic of a program or a solution to a problem. Let's explore these tools in more detail:

1. Algorithm:

An algorithm is a step-by-step, well-defined set of instructions for solving a particular problem or accomplishing a specific task. Algorithms are used in various fields, including computer science, mathematics, and engineering. They provide a systematic way to solve problems and can be implemented in various programming languages. Key characteristics of algorithms include:

- Precision: Algorithms must be clear, unambiguous, and precise, with each step defined in a straightforward manner.

- Finiteness: Algorithms must have a finite number of steps, meaning they eventually terminate.

- Input and Output: Algorithms take input data, process it through a sequence of steps, and produce an output.

- Determinism: Each step in an algorithm should be deterministic, meaning it produces the same result for a given set of inputs

NOTES

Algorithms serve as the foundation for program planning and are usually designed before writing the actual code. They provide a high-level overview of the problem-solving process.

2. Flowcharts:

A flowchart is a graphical representation of an algorithm or a process, using various symbols and shapes to depict the flow of control within the program or system. Flowcharts are particularly helpful for visualizing the logical sequence of steps and decision points in a program. Here are some common flowchart symbols and their meanings:

- Start/End: Represents the beginning or end of the process.

- Process: Depicts a specific task or action.

- Decision: Represents a condition or decision point, with arrows indicating different possible paths based on the condition.

- Input/Output: Signifies data input or output points.

- Flowlines: Connect symbols and show the direction of flow.

Flowcharts provide a visual way to communicate the algorithm's logic, making it easier for developers to understand and implement. They are particularly useful for planning complex programs and for documenting existing processes. The example for the flowchart is the computer system block diagram.

Introduction to System Software:

System software plays a crucial role in the functioning of a computer system. It serves as an intermediary between the hardware and application software, managing various tasks and resources to ensure the efficient operation of the computer. Let's explore some key components of system software and related concepts:

1. System Software:

System software is a type of software that controls and manages the hardware components of a computer system. It includes various programs and utilities that facilitate the execution of

application software and ensure the smooth operation of the computer. Some common examples of system software include operating systems, device drivers, and utility programs.

2. IDE (Integrated Development Environment):

An Integrated Development Environment is a software application that provides a comprehensive set of tools for software development. It typically includes a code editor, debugger, compiler, and other features that streamline the software development process, making it more efficient and user-friendly.

3. Assembler:

An assembler is a program that translates assembly language code into machine code, which can be directly executed by the computer's CPU. It converts human-readable mnemonics into binary instructions.

4. Compiler:

A compiler is a software tool that translates high-level programming language code into machine code or a lower-level intermediate code. Unlike interpreters, compilers generate an executable file that can be run independently of the source code.

5. Linker:

A linker is a program that combines multiple object files (produced by a compiler) and libraries to create a single executable program. It resolves references between different parts of the code and links them together.

6. Loader:

A loader is a program responsible for loading an executable program into memory for execution. It takes care of memory allocation, initializing program components, and preparing the program for execution.

NOTES

7. Debugger:

A debugger is a tool used by developers to identify and rectify errors in software code. It provides features such as step-by-step code execution, variable inspection, and breakpoints to aid in the debugging process.

8. Breakpoints:

Breakpoints are markers set by a developer within the code. When the program execution reaches a breakpoint, it pauses, allowing the developer to inspect the program's state, variables, and other details. This is a valuable feature for debugging and testing.

9. Syntax:

Syntax refers to the set of rules and conventions that dictate the structure and format of programming languages. Following the correct syntax is crucial for writing error-free code, as syntax errors occur when the code deviates from these rules.

10. Logical Errors:

Logical errors, also known as semantic errors, occur when the code's logic is flawed, leading to undesired or incorrect program behavior. Unlike syntax errors, logical errors do not generate error messages but require careful debugging and analysis to identify and fix.

- ## Types of Languages:

1) High level language VS Low level Language:

NOTES

High-Level Languages	Low-Level Languages
High-Level Languages are easy to learn and understand.	Low-Level Languages are challenging to learn and understand.
They are executed slower than lower level languages because they require a translator program.	They execute with high speed.
They allow much more abstraction.	They allow little or no abstraction.
They do not provide many facilities at the hardware level.	They are very close to the hardware and help to write a program at the hardware level.
For writing programs, hardware knowledge is not required.	For writing programs, hardware knowledge is a must.
The programs are easy to modify.	Modifying programs is difficult.
A single statement may execute several instructions.	The statements can be directly mapped to processor instructions.
BASIC, Perl, Pascal, COBOL, Ruby etc are examples of High-Level Languages.	Machine language and Assembly language are Low-Level Languages.

2) Machine Language:

Machine Language is the lowest-level programming language and is considered the first-generation programming language. It consists of binary code, which is a series of 0s and 1s that directly correspond to machine instructions. Machine language is specific to the computer's architecture and is the only language that a computer's central processing unit (CPU) can directly understand and execute.

Machine language instructions are very basic and typically correspond to operations at the hardware level, such as moving data between memory locations, performing arithmetic calculations, and controlling the flow of program execution. While machine language is highly efficient for the computer, it is not human-friendly, making it challenging for programmers to work with directly.

NOTES

In practice, most programmers use higher-level programming languages, such as assembly language or high-level programming languages, to write software because they are more readable and maintainable. These higher-level languages are then translated into machine code by an assembler or compiler for execution on the computer's hardware.

- **Interview Questions:**
  1. What is the role of the Arithmetic and Logical Unit (ALU) in the Central Processing Unit (CPU)?
  2. Can you explain what the Control Unit (CU) does in the CPU?
  3. What is the main difference between hardware and software in a computer system?
  4. What is the main difference between hardware and software in a computer system?
  5. What do you understand about the high level , low level and machine language?