Unit-1 Introduction to Computer Fundamentals

- Computer System Components:
  - Input and Output Devices, Memory and Storage Devices,
  - Block diagram of Computer System,
  - Types of Software, Operating System with its types,
  - Compiler, Interpreter, Assembler, Linker, Loader.

1. Computer System Components

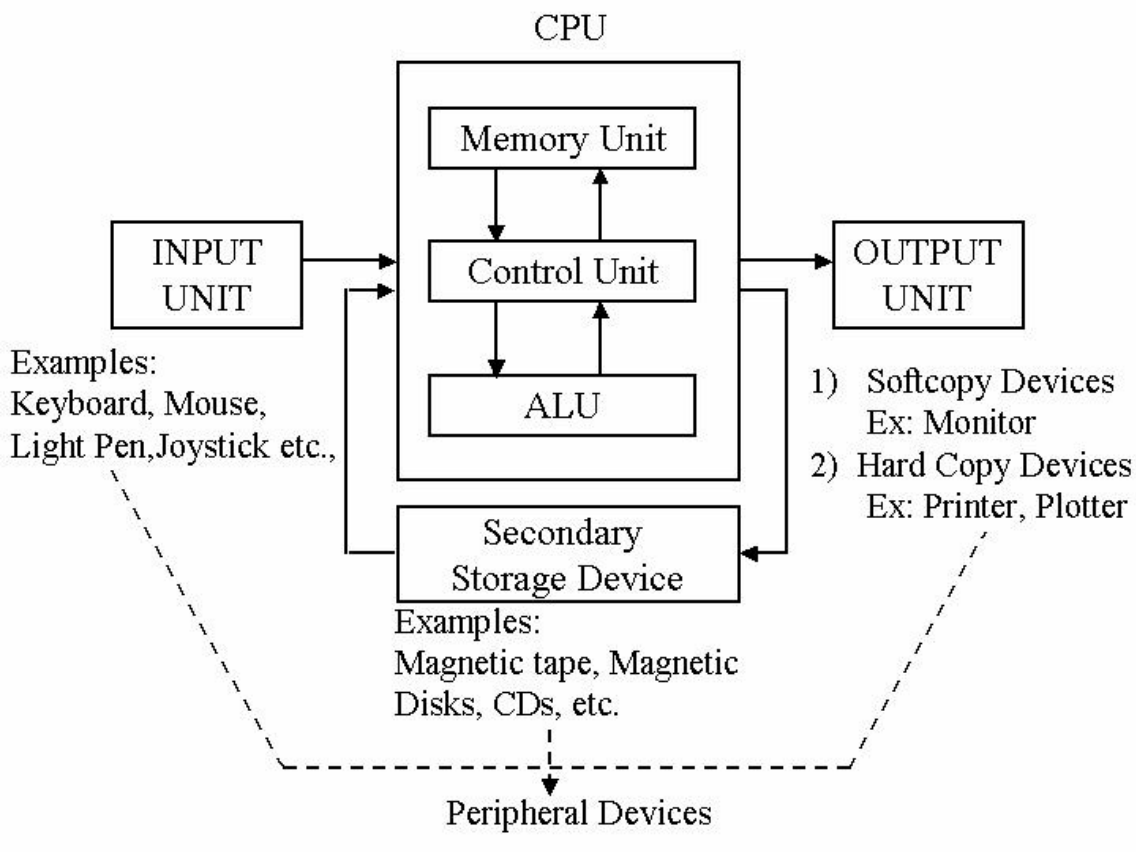
- **Input Devices**: Devices that allow users to input data into the computer.
  - **Examples**: Keyboard, Mouse, Microphone, Scanner, Touchscreen, Barcode Reader.
- **Output Devices**: Devices that output data from the computer to the user.
  - **Examples**: Monitor, Printer, Speakers, Projector, Headphones.

2. Memory and Storage Devices

- **Primary Memory (Volatile)**: Fast and temporary storage used by the CPU to store data currently being used.
  - **Examples**: RAM (Random Access Memory), Cache.
- **Secondary Memory (Non-volatile)**: Permanent storage for storing large amounts of data.
  - **Examples**: Hard Drive (HDD), Solid-State Drive (SSD), Optical Discs (CD, DVD), Flash Drives.

3. Block Diagram of a Computer System

A typical **block diagram of a computer system** shows the interaction between major components:



In the above diagram, both control **(control unit or CU)** and **arithmetic & logic unit (ALU)** combined called as Central Processing Unit (CPU).

Let’s describe all the parts as included within the above diagram one by one.

1. **The Processor Unit (CPU):** It is the brain of the PC system. All major calculations and comparisons are made inside the CPU and it’s also liable for activation and controlling the operation of another unit.

   This unit consists of two major components, that are **arithmetic logic unit (ALU)** and **control unit (CU)**.

2. **Arithmetic Logic Unit (ALU):** Here arithmetic logic unit performs all arithmetic operations like **addition, subtraction, multiplication,** and **division**. It also uses **logical operation** for comparison.

3. **Control Unit (CU):** And the control unit controls the whole operation of the PC. It also controls all devices like memory, input/output devices connected to the CPU. Following are the three steps of the instruction execution cycle:

   1. **Fetch** : This is the first step where the CPU 'fetches' an instruction from the primary memory (RAM).
   2. **Decode**: Right after fetching the instruction, the CPU 'decodes' it or translates it into a series of actions it can understand.
   3. **Execute**: Finally, the CPU 'executes' the instruction decoded in the previous step. It carries out the actions dictated by the instruction.

4. **Input/output Unit:** These devices are used to enter data and instructions into the computer. They act as a bridge between the user and the computer system.

```
**Input Devices:** Examples: Keyboard, Mouse, Scanner, Microphone.
**Output Devices:** Examples: Monitor, Printer, Speakers.
```

5. **Memory Unit:** The memory unit is an important component of a computer . it's where all data intermediate and finds results are stored. The data read from the most storage or an input unit are transferred to the computer's memory where they're available for processing. This memory unit is employed to carry the instructions to be executed and data to be processed.

6. **Disk Storage Unit:** Data and instruction enter into a computing system through data input device need to stored inside the pc before actual processing start. Two sorts of storage units are the first and auxiliary storage units.

```
 i. **Primary Storage Unit:** Primary memory features a direct link with the input unit and output unit. It stores the input file , calculation result.
 ii. **Secondary Storage Unit:** The primary storage isn't ready to store data permanently for future use. So another sorts of storage technology are required to store the info permanently for an extended time, it's called secondary or external storage .
```

4. Types of Software

Software can be broadly classified into two main categories:

1. **System Software**:
   - Controls the hardware and provides a platform for running application software.
   - **Examples**: Operating System, Device Drivers, Utilities.
2. **Application Software**:
   - Designed to perform specific tasks for the user.
   - **Examples**: Word Processor (MS Word), Web Browser (Chrome), Games, Databases.

5. Operating System (OS) and its Types

The **Operating System** is system software that manages computer hardware and software resources, providing services for computer programs.

- **Types of Operating Systems**:

  1. **Batch OS**: Processes batches of jobs without manual intervention.
  2. **Time-Sharing OS**: Allows multiple users to share system resources simultaneously.
  3. **Real-Time OS (RTOS)**: Processes tasks in real-time with stringent timing constraints.
  4. **Distributed OS**: Manages a group of independent computers and makes them appear to be a single computer.
  5. **Network OS**: Manages data, users, groups, and security in a networked environment.
  6. **Embedded OS**: Used in embedded systems like appliances, phones, and cars.

  - **Functions of an Operating System**

1. **Process Management:** Manages processes, including their execution, scheduling, and termination.
2. **Memory Management:** Allocates and manages system memory for processes and applications.
3. **File System Management:** Manages files, directories, and file permissions.
4. **Device Management:** Manages communication with hardware devices like printers, disks, and monitors.
5. **Storage Management:** Organizes data on storage devices and manages data retrieval and backups.
6. **Security and Access Control:** Ensures system security through user authentication and access control.
7. **Network Management:** Manages network connections and communication between devices.

6. Compiler, Interpreter, Assembler, Linker, Loader

- **Compiler**:
  - Translates high-level programming language (e.g., C, C++) into machine code in one go.
  - Creates an object file or executable.
- **Interpreter**:
  - Translates and executes high-level language line by line.
  - Does not produce an intermediate machine code file.
  - Example languages: Python, JavaScript.
- **Assembler**:
  - Converts assembly language into machine code (object code).
  - Machine-specific.
  - Example: Converts x86 assembly into machine code for an x86 processor.
- **Linker**:
  - Combines multiple object files (compiled code) into a single executable.
  - Resolves external references between different object files and libraries.
- **Loader**:
  - Loads the executable into memory, performs address binding, and starts the program's execution.

Feature	Compiler	Interpreter	Assembler	Linker	L
Definition	Translates the entire high-level program into machine code in one go.	Translates and executes the high-level code line by line.	Converts assembly language code into machine code.	Combines object files and libraries into a single executable.	Load exec into and it f exec

Execution Time	Faster execution since it translates the code once.	Slower execution as it translates the code line by line during runtime.	Doesn't execute code; only converts assembly to machine code.	No direct execution; it prepares the executable.	No tran it p the for exec
Translation Method	Converts entire source code into machine code.	Translates one line of source code at a time.	Converts assembly code into object code (machine code).	Combines machine code modules.	Load mach into for exec
Output	Produces an object file or executable after translation.	No intermediate object file, directly executes instructions.	Produces an object file (machine code).	Produces an executable file from object files.	Does proc outp file simp into
Error Handling	Displays all errors after compilation, stopping at none.	Stops execution when the first error is encountered.	Reports assembly errors during translation.	Handles errors related to unresolved references between object files.	Does hand erro reli the this
Example Languages	C, C++, Java	Python, Ruby, JavaScript	Assembly Language	Used with compiled languages like C, C++	Work exec file plat
Memory Usage	Generally requires more memory since the entire program is compiled at once.	Efficient use of memory as only one line is processed at a time.	Works with machine code, memory depends on the code size.	Requires memory for the linking process.	Allo memo the duri exec
Platform Dependency	The output is machine-specific (dependent on the target architecture).	Not machine-specific, often platform-independent if run on an appropriate interpreter.	Machine-specific (dependent on architecture).	Platform-specific, links machine code for a specific platform.	Plat spec load exec mach on a part syst
Examples	GCC, Clang	CPython, Node.js, Ruby Interpreter	NASM, MASM	GNU Linker (ld),	Oper Syst load

				Microsoft Linker	(Wir load Linu load
———					
### Summary of Components and Functions:					
Component	**Description**				
---------- ----------	---------- ---------- ---------- ---------- ---------- ---------- ---------- ----------				
Input Devices	Devices to provide data to the system (e.g., keyboard, mouse).				
Output Devices	Devices that show the result of processing (e.g., monitor, printer).				
Memory	Primary and secondary memory that stores data and instructions.				
Operating System	Manages hardware resources and provides an environment for applications.				
Compiler	Converts the entire high-level source				

	code into machine code.				
Interpreter	Converts and executes code line-by-line during runtime.				
Assembler	Converts assembly language into machine code.				
Linker	Combines object files into a single executable, resolving references.				
Loader	Loads an executable into memory and prepares it for execution.				