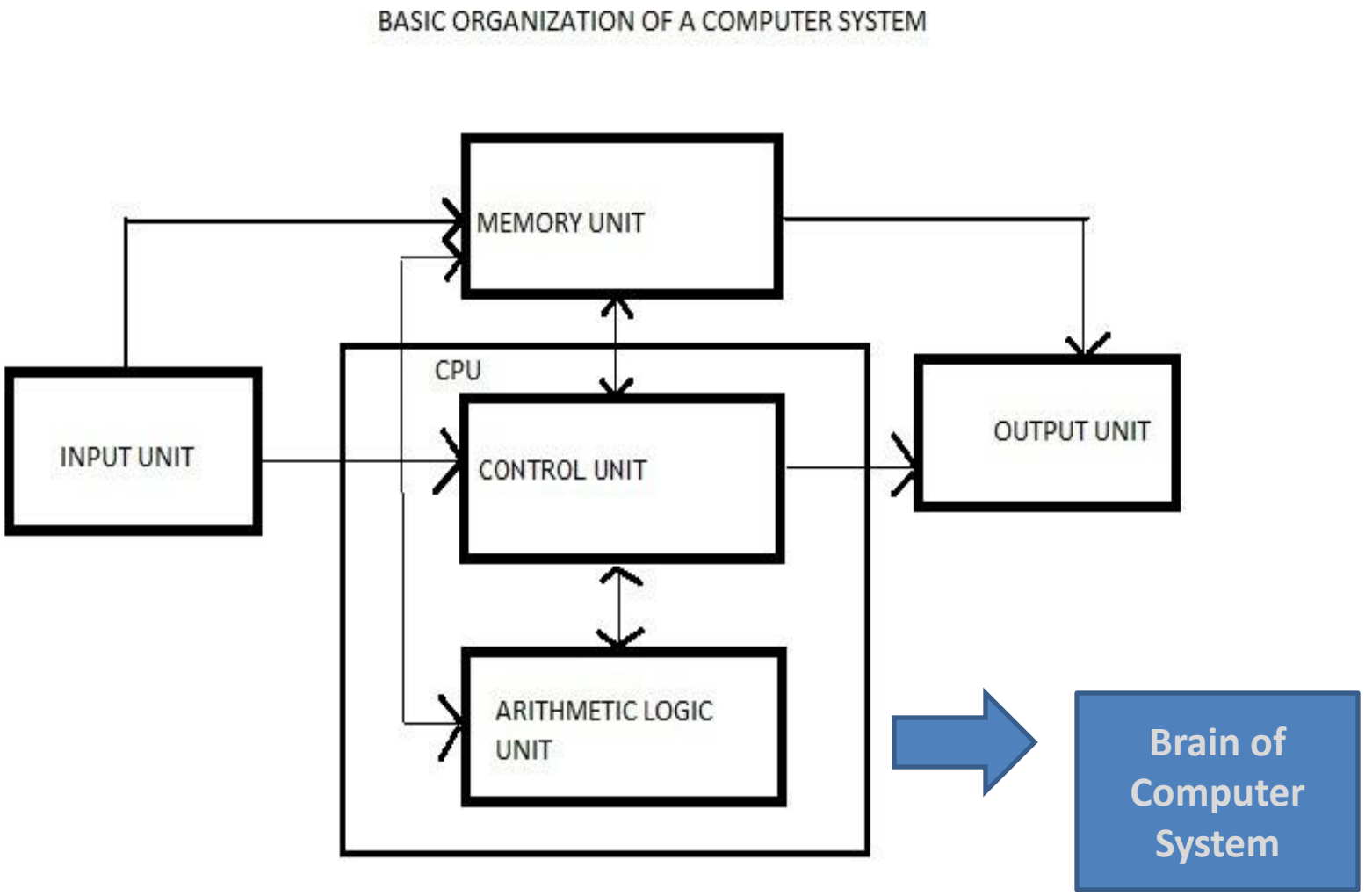
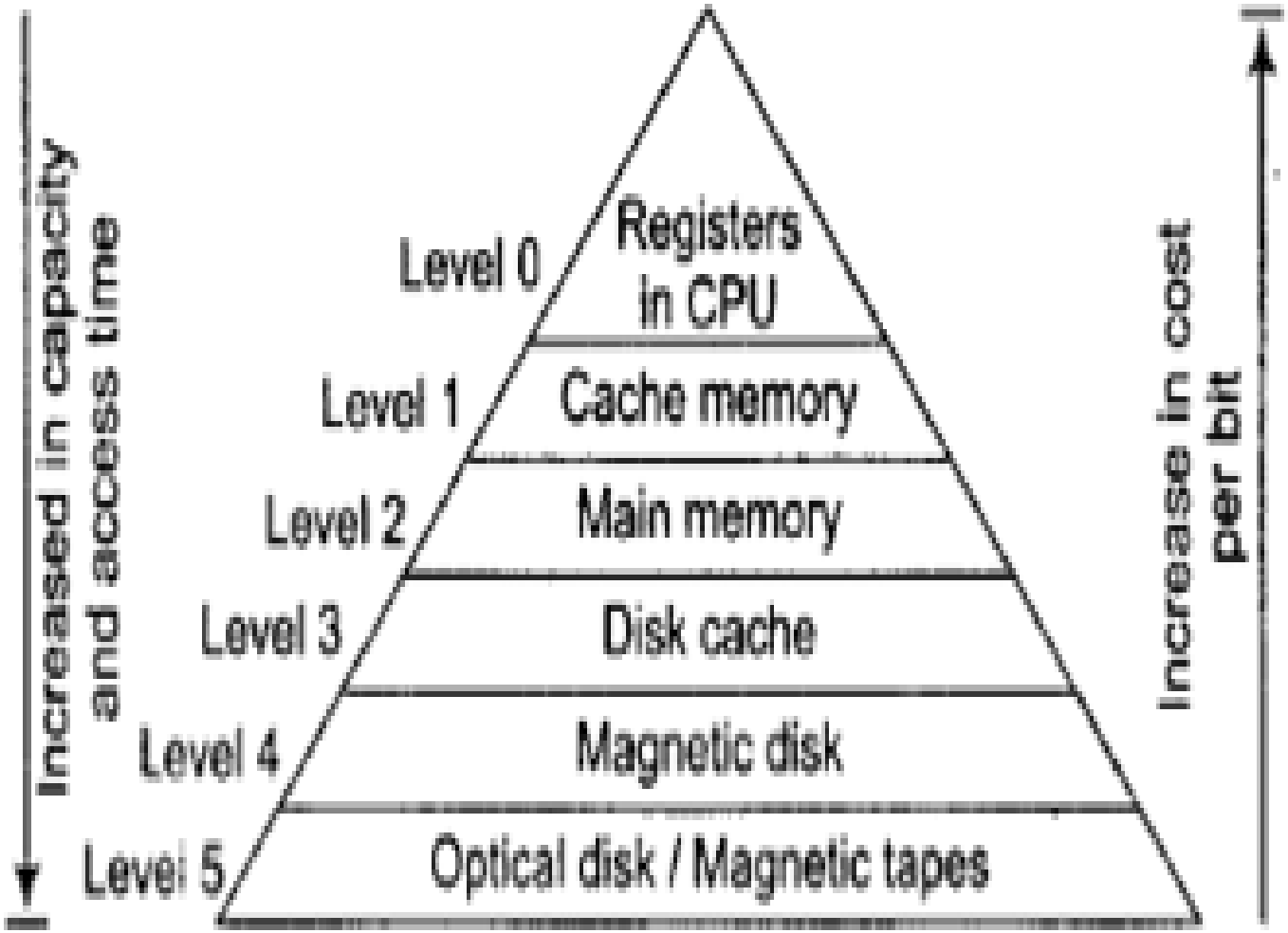
Basic Organization of a Computer



Computer Memory

- Computer memory is the storage space in the computer, where data is to be processed and instructions required for processing are stored.

- Each location has a unique address in memory.


Increased in capacity and access time
Level 0
Registers in CPU
Level 1
Cache memory
Level 2
Main memory
Level 3
Disk cache
Level 4
Magnetic disk
Level 5
Optical disk / Magnetic tapes
Increase in cost per bit

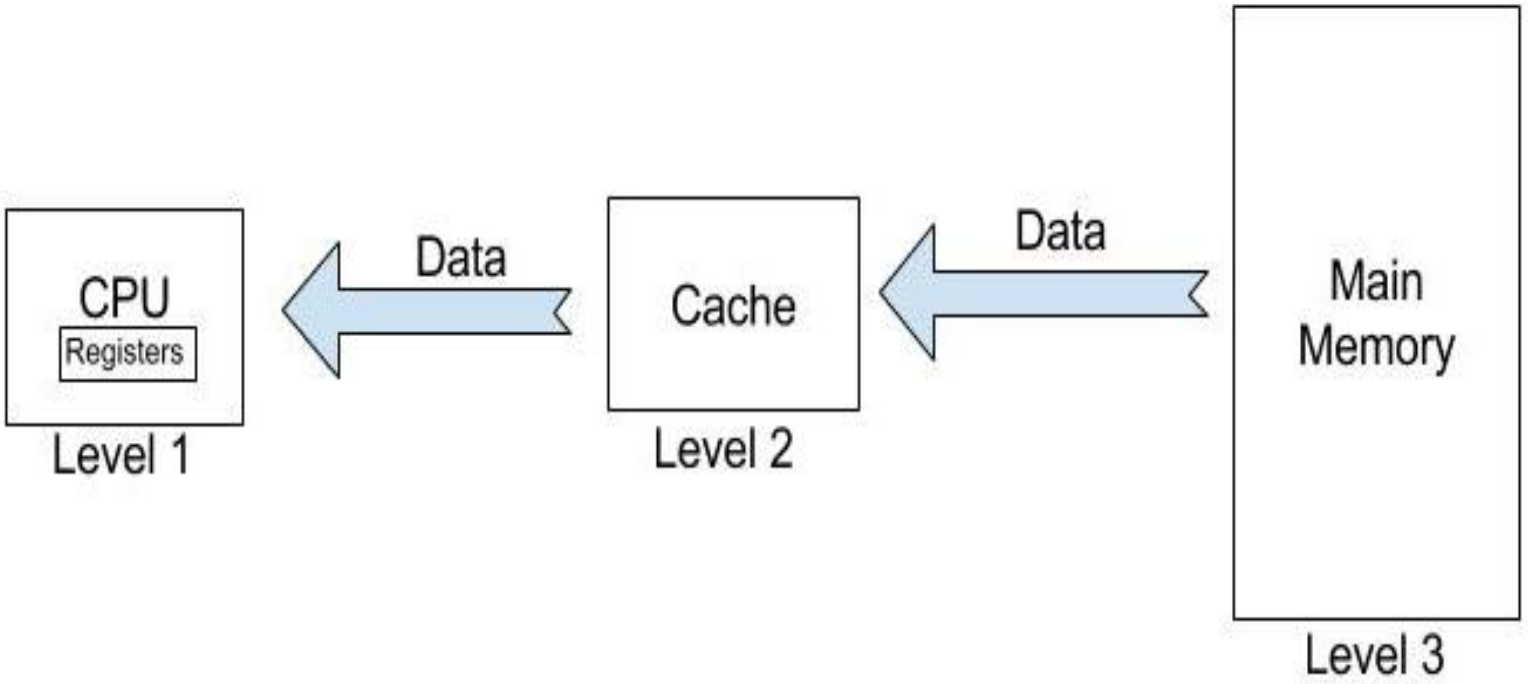
CPU Registers

- **CPU Register or processor register** is a quickly accessible location available to a computer's CPU.

- It is fastest among the all types of data storage.

Cache Memory

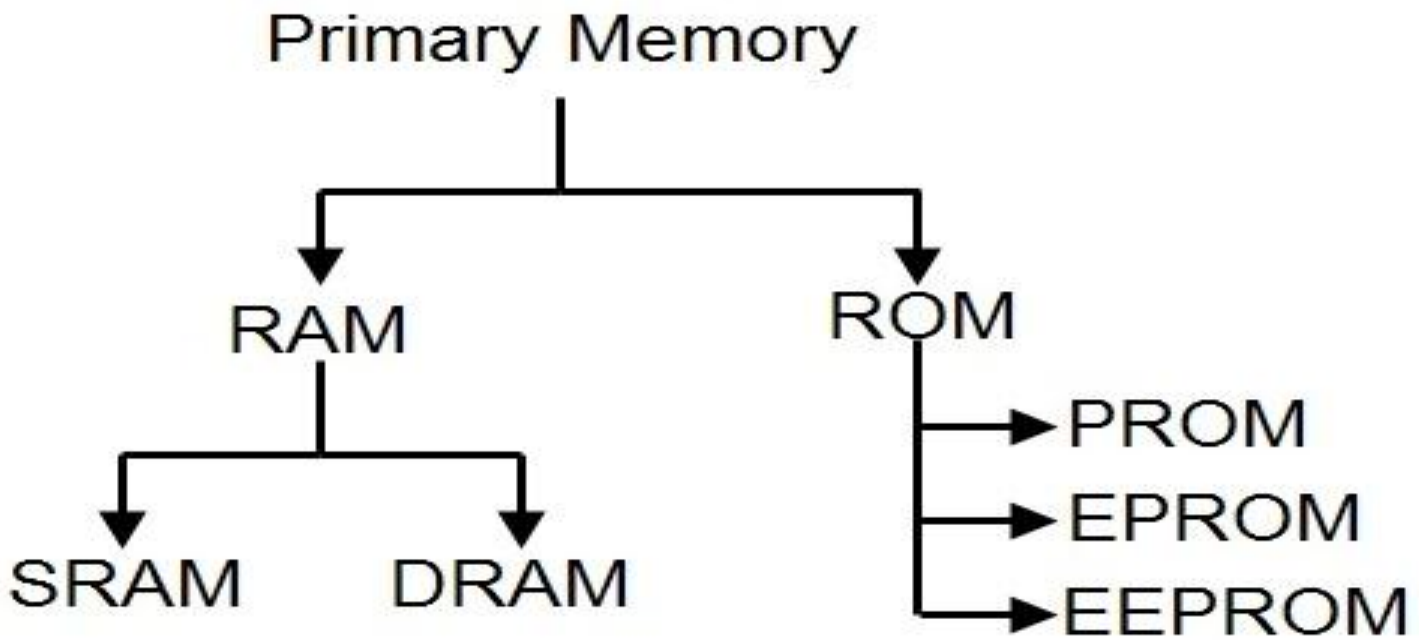
- **It is used to hold those parts of data and program which are most frequently required to execute program.**
- It consumes less access time as compared to main memory, so it is faster than main memory.
- **It is the portion of memory made of high speed RAM (SRAM).**
- Cache memory has limited capacity to store data.
- Widely used for Memory Caching.
- Works on the "**Principle of Locality of Reference**".



It acts as a buffer between the CPU and the main memory.

Primary Memory

- Primary memory is computer memory that is accessed directly by the CPU.



Random Access Memory (RAM)

- RAM is used to hold the program and data during computation i.e. stores temporary data.
- Volatile memory: data retains as long as continuous power supply is provided.
- Any cell can be accessed in any order at same speed if address is known.

Read Only Memory (ROM)

- Non- volatile in nature.
- Information can simply be read by the user but cannot be modified.
- Generally stores BIOS(Basic Input Output System)

1. PROM – Programmable Read Only Memory
2. EPROM – Erasable Programmable Read Only Memory
3. EEPROM – Electrically Erasable Programmable Read Only Memory

Units to measure computer Memory

UNIT	ABBREVIATION	STORAGE
Bit	B	Binary Digit, Single 1 or 0
Nibble	-	4 bits
Byte/Octet	B	8 bits
Kilobyte	KB	1024 bytes
Megabyte	MB	1024 KB
Gigabyte	GB	1024 MB
Terabyte	TB	1024 GB
Petabyte	PB	1024 TB
Exabyte	EB	1024 PB
Zettabyte	ZB	1024 EB
Yottabyte	YB	1024 ZB

Secondary Memory

- Communicates indirectly with CPU via main memory. So, It is slower than the main memory.
- Non- volatile in nature. So, store data permanently.

1. **Magnetic Storage Devices:** it is sequential access memory.
2. Hard Disk
3. Floppy Disk
4. Magnetic Tape

- **Optical Storage Devices:**

1. CD-ROM
2. CD-Recordable
3. CD-Rewritable
4. DVD-ROM

- **USB Flash Drive**
- **Memory Cards**
- **Solid State Drive**

Processing Devices

- Processing devices are parts of the computer that are responsible for processing or converting data into meaningful information.

1. Processor
2. Buses
3. System Clock

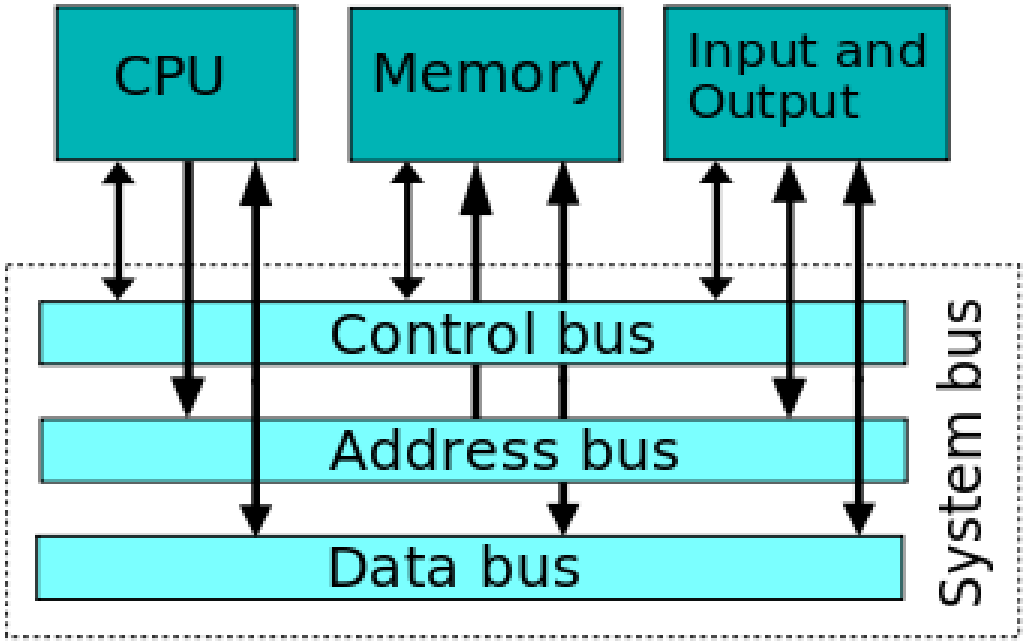
1. Processor:

A. The CPU is traditionally referred to as a Processor.
B. The CPU is a computer chip located on the motherboard.
C. Performs processing and control activities performed by different parts of computer.
D. Main electronic circuitry in the computer.
E. Carries out the instructions contained in a computer program by performing arithmetic, logical, control and input/output operations.
F. Most modern CPUs are contained on a single Integrated Circuit (IC) chip and as such are called microprocessors.
G. A processor can have two or more CPUs or independent processing units called “cores” on a single chip and such processor is called a multi-core processor.

2. **Buses**

- Electrical pathway that transfer data and instructions among different parts of computer.
- Main memory is directly/indirectly connected to the processor via a bus.

1. Data Bus
2. Address Bus
3. Control Bus



- Address bus is **Unidirectional** because the microprocessor is addressing a specific memory location.

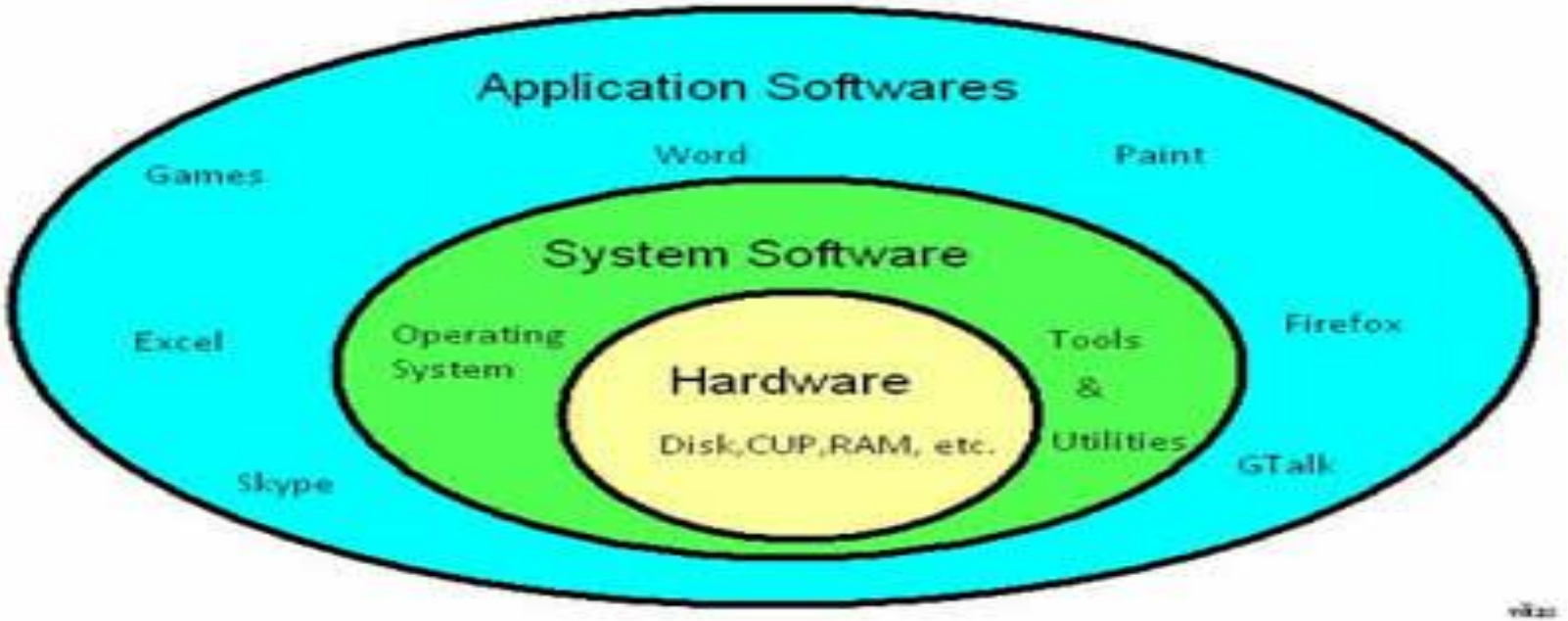
- Data **bus** is **Bidirectional** because the Microprocessor can read data from memory or write data to the memory.

3. Clock

- Used to synchronizing the activities performed by the computer.

Software

- Set of computer programs which includes instructions, used for performing a particular task using hardware.
- Software tells hardware: what to do? How to do?



1. **System Software**:

- System software is "Background" software that helps the computer manage its own internal resources.
- It enables the application software to interact with the computer hardware. Eg. Operating System, Device Drivers, Utility Software, Translators etc..

2. **Application Software**: Collection of programs written for a specific application.

Operating System

- System software, helps in managing the resources of a computer.
- Primary goal: make computer convenient and efficient to use.
- Eg. MS-DOS, MS-Windows, UNIX, Linux, Mac OS etc…

Tasks performed by OS

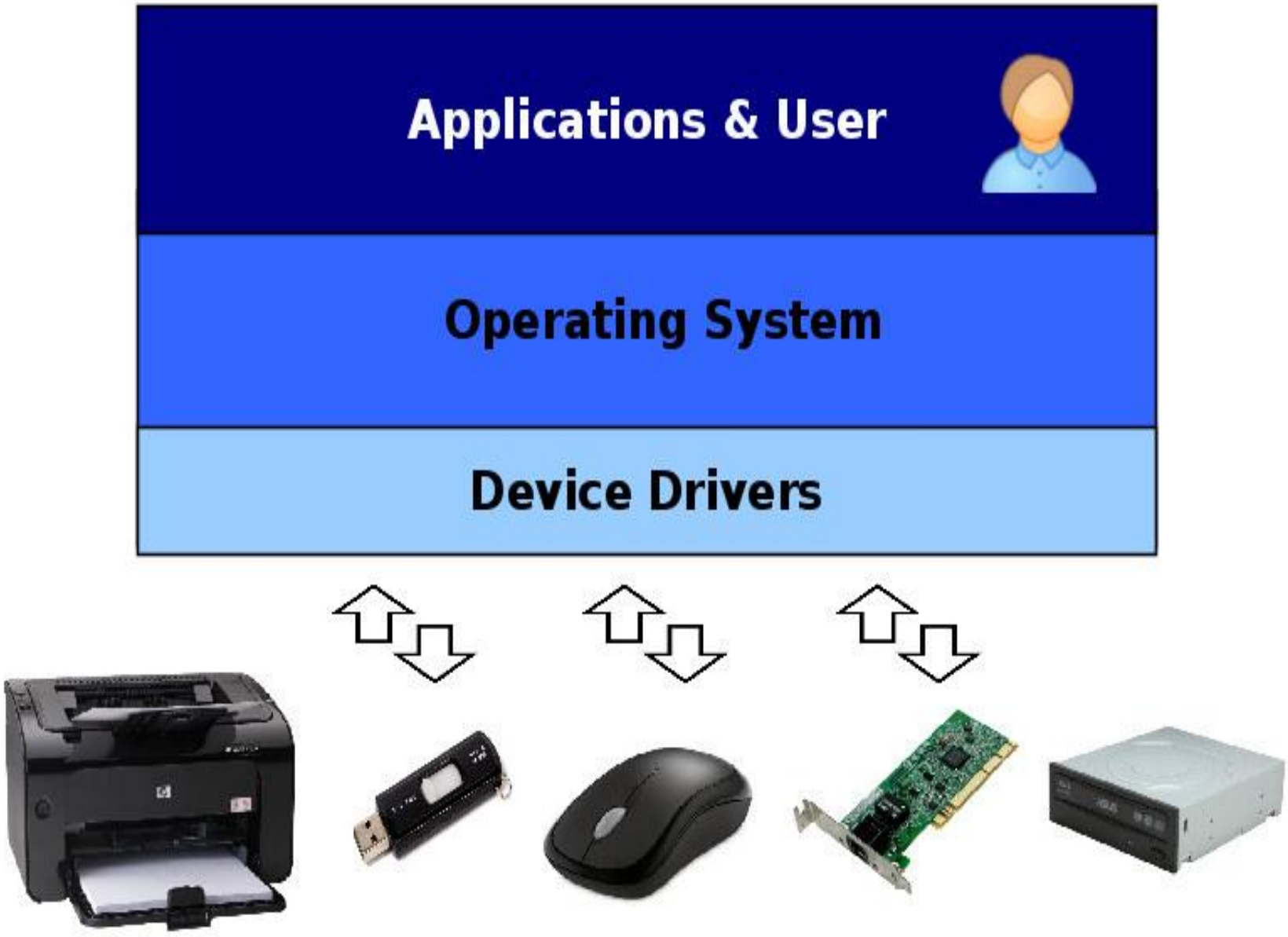
- Process Management
- Memory Management
- File Management
- Device Management
- Security and user Interface
- Servicing the request by user etc…

Types of OS

- Batch Processing OS
- Multiuser OS
- Multi tasking OS
- Multithreading OS
- Time Sharing OS

Device Drivers

- System Software, responsible for proper functioning of devices.
- Each device has a device driver associated with it.
- Whenever computer system needs the use of device, the processor issues general commands to the driver of device.
- When you buy an operating system, many device drivers are built into the product. In Windows operating systems, a device driver file usually has a file name suffix of DLL or EXE


Applications & User
Operating System
Device Drivers

Utility Software

- Used to analyze, configure and maintain the computer system.
- Examples of utility programs are antivirus software, backup software and disk cleaners, clean up tools, defragmentation tool etc…

Levels of Programming Languages

1. Low level Language
   A. Machine Language
   B. Assembly Language

2. High level Language

Machine Language

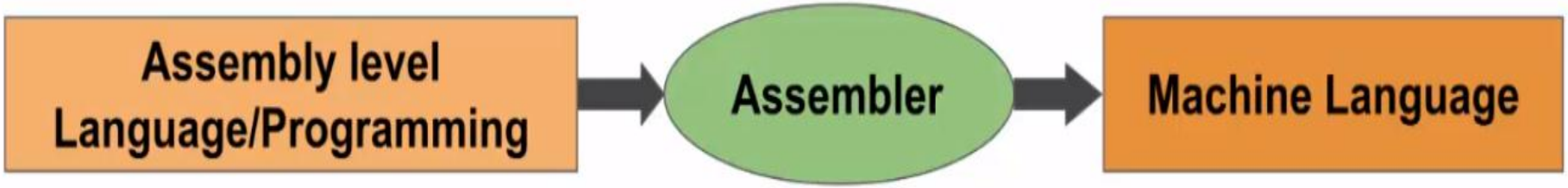
- First Generation Language.
- Instructions are represented by combinations of 0’s and 1’s.
- Programs are executable, can be run directly.
- Requires memorization of binary codes. So, difficult to learn.
- Machine Dependent Codes, not portable.

Assembly Language

- Second Generation Language.
- Machine language instructions are replaced with simple pneumonic abbreviations (e.g. ADD, MUL, DIV etc…).
- Programs needs to translate into machine language.

What is assembly language?

- Assembly language is an intermediate language which lies between low and high level language. It is higher than machine language (low-level language comprised of numbers) and lower than high level languages (human understandable programming languages like Java, C++, FORTRAN etc.)



Assembler

- Translate assembly language statements into machine language codes.

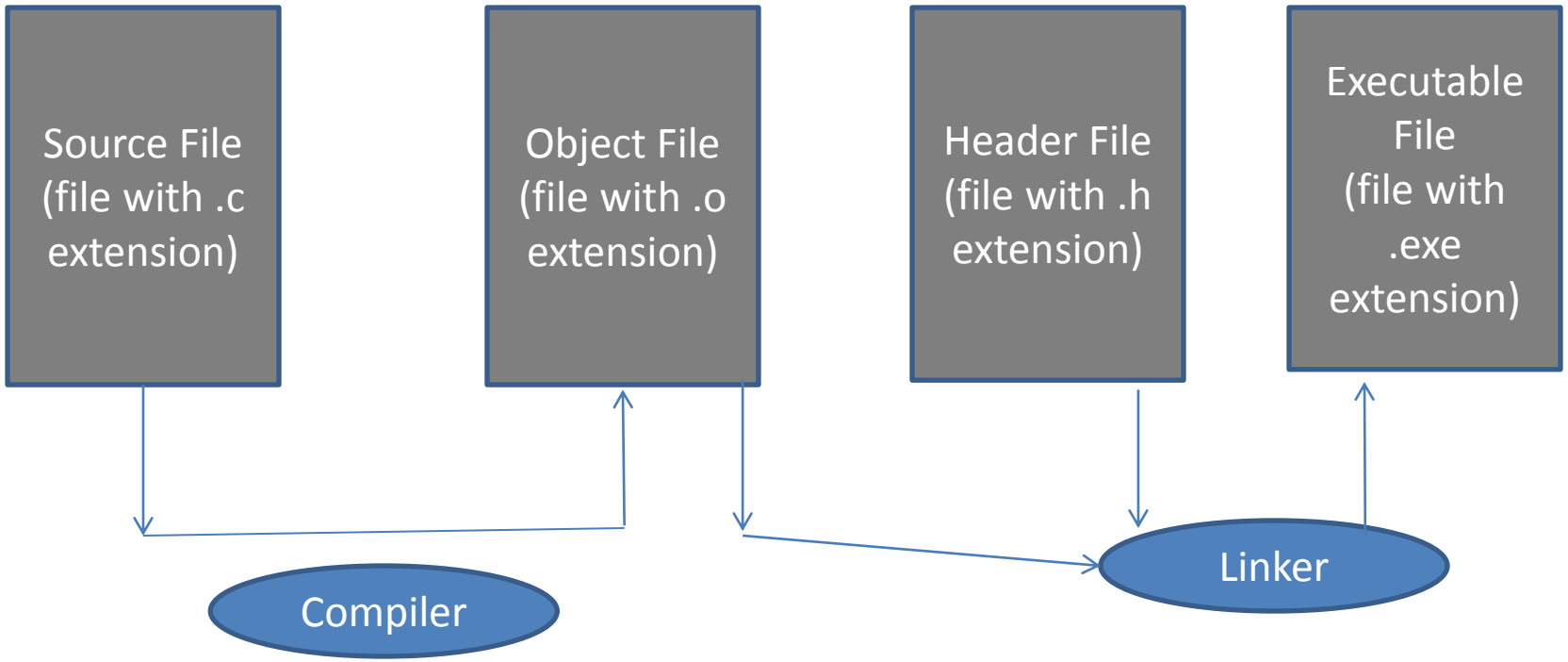
An Assembler refers to a program which converts basic computer instructions written in assembly language to machine code that can be easily understood by the machine

- Types of Assembler –
  1. Single-pass Assembler
  2. Two-pass Assembler

High-Level Language

- Consist of set of English like statements, it makes programming easier and less error-prone.
- Languages are not closely related to internal characteristics of computer.
- Two Categories:
  1. Specific Purpose Languages (LISP, Prolog etc…)
  2. General Purpose Language(C++, JAVA, C etc…)

Files used in c programming

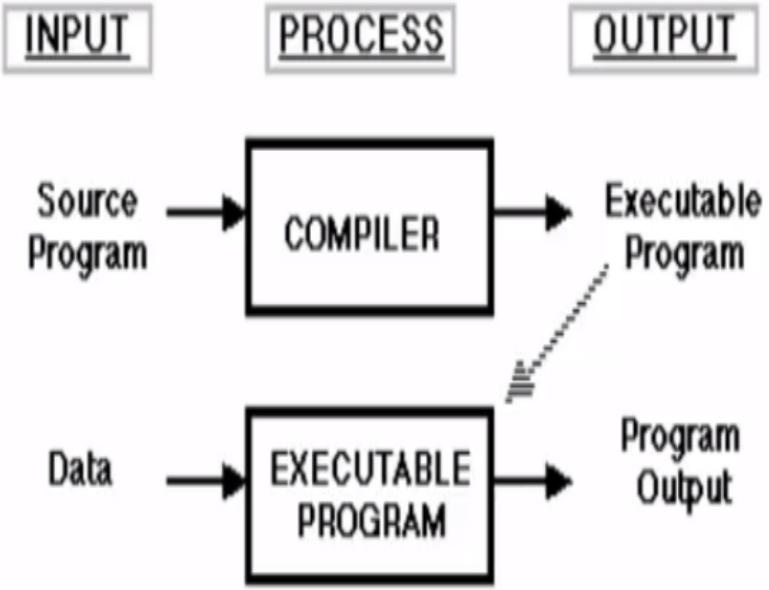


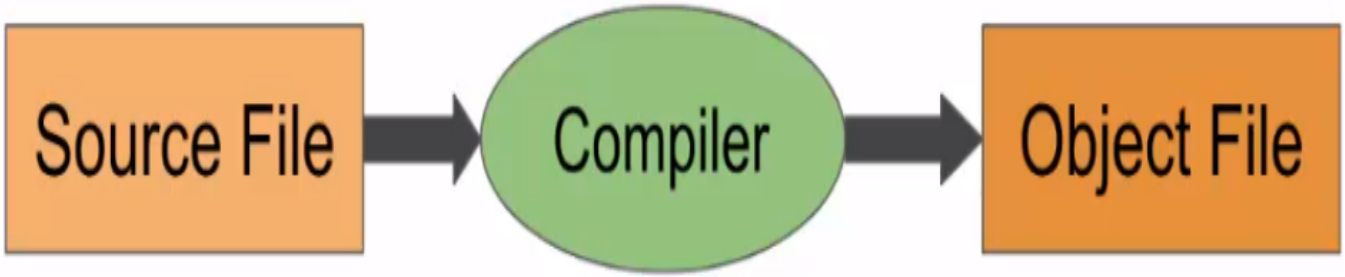
Compiler

- The name *compiler* is primarily used for programs that translate source code from a high-level programming language to a lower level language (e.g., assembly language, object code, or machine code) to create an executable program.

- Compiler resides on a disk or other storage media, when a high-level program is to be compiled, compiler is loaded into main-memory.

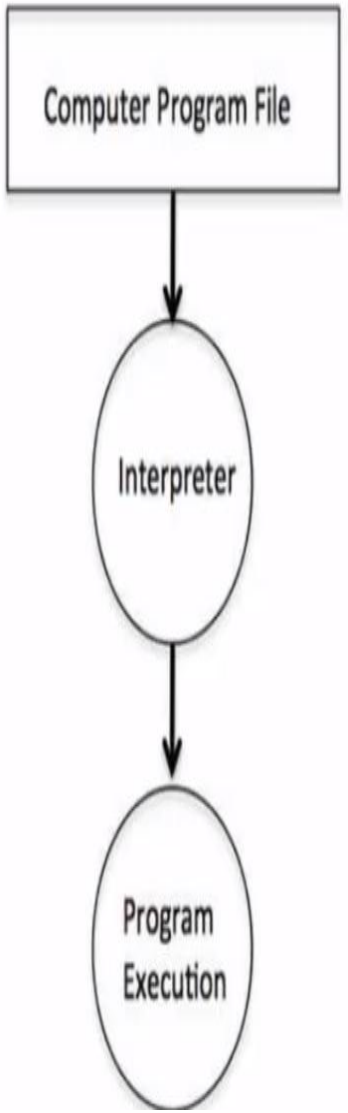
A Compiler refers to a software or program which converts code written in high level language (Java, C++, Python etc.) into machine code or bytecode.





Interpreter

- Converted high-level language code into corresponding machine code.
- Instead of entire program, one statemen at a time is translated and executed immediately.



Difference between Compiler and Interpreter

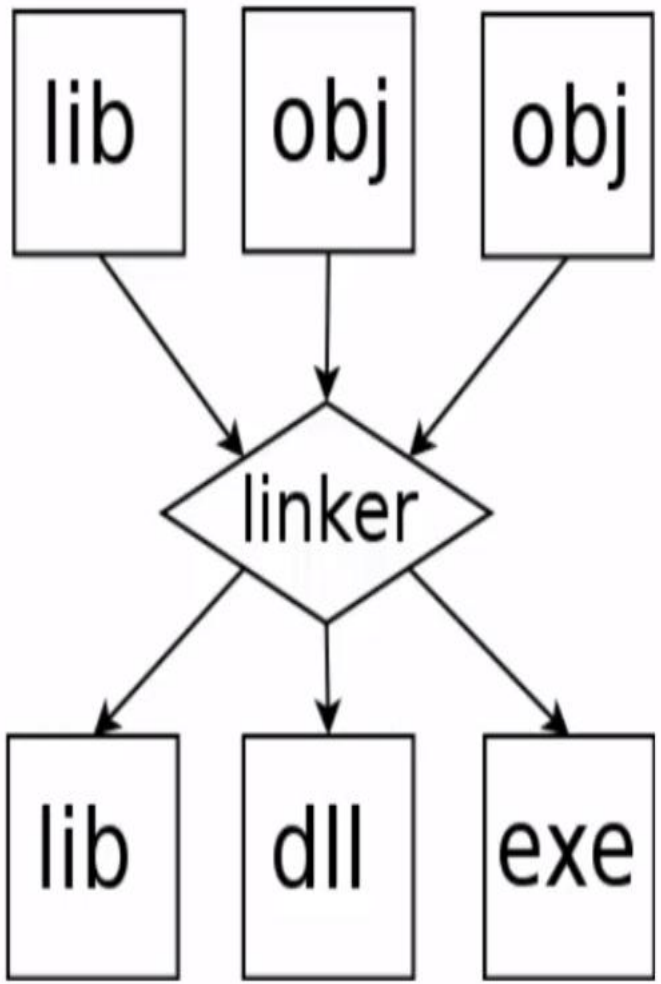
No	Compiler	Interpreter
1	Compiler Takes **Entire** program as input	Interpreter Takes **Single** instruction as input .
2	Intermediate Object Code is **Generated**	**No** Intermediate Object Code is **Generated**
3	Conditional Control Statements are Executes **faster**	Conditional Control Statements are Executes **slower**
4	**Memory Requirement** : **More** (Since Object Code is Generated)	**Memory Requirement** is **Less**
5	Program need not be **compiled** every time	Every time higher level program is converted into lower level program
6	**Errors** are displayed after **entire program** is checked	**Errors** are displayed for **every instruction** interpreted (if any)
7	**Example** : C Compiler	**Example** : BASIC

Linker/Link Editor/Binder

- In high-level language built-in library functions need to be linked to the library. This is done by Linker.
- Sometimes, programs are divided into modules. These modules are combined and assembled and object module is generated.
- Linker has the responsibility to combine / Link all modules and generate a single executable file of the source program.

The linker links all the functions and files required by the object code and converts that to executable code.

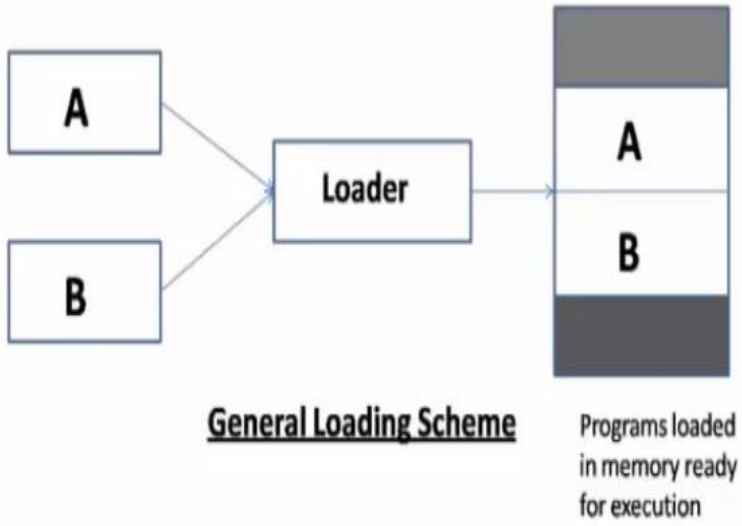
The converted code is stored with an executable (.exe on windows) extension. If the file or function that has to be linked, does not exist, it produces an error



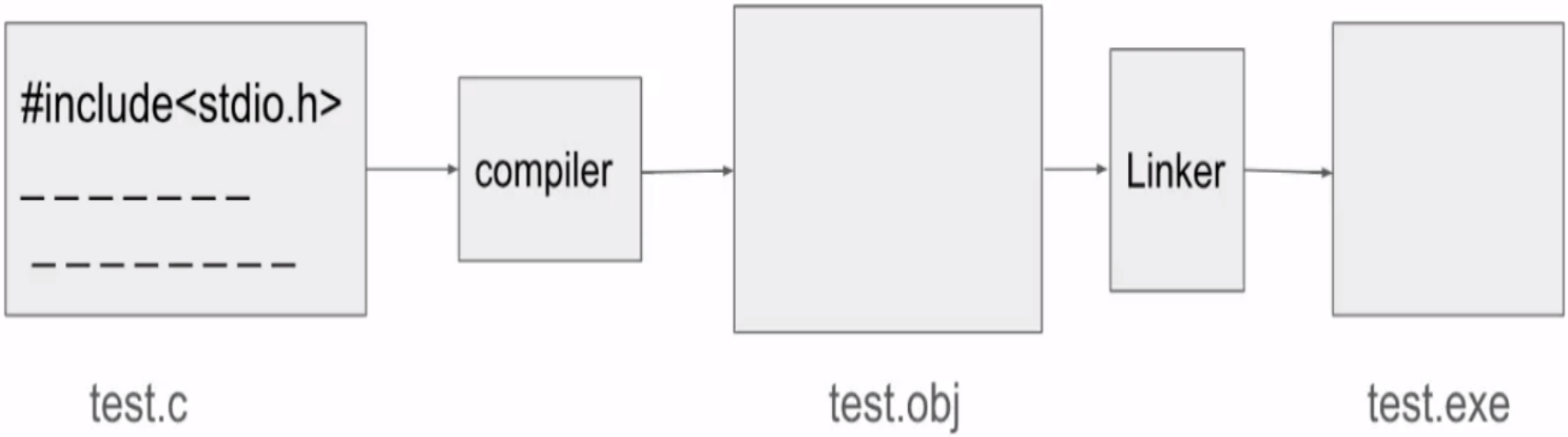
Loader

A loader is the part of an operating system that is responsible for loading programs. It is one of the essential stages in the process of starting a program, as it places programs into memory and prepares them for execution.

Errors generated at this stage are called as runtime error



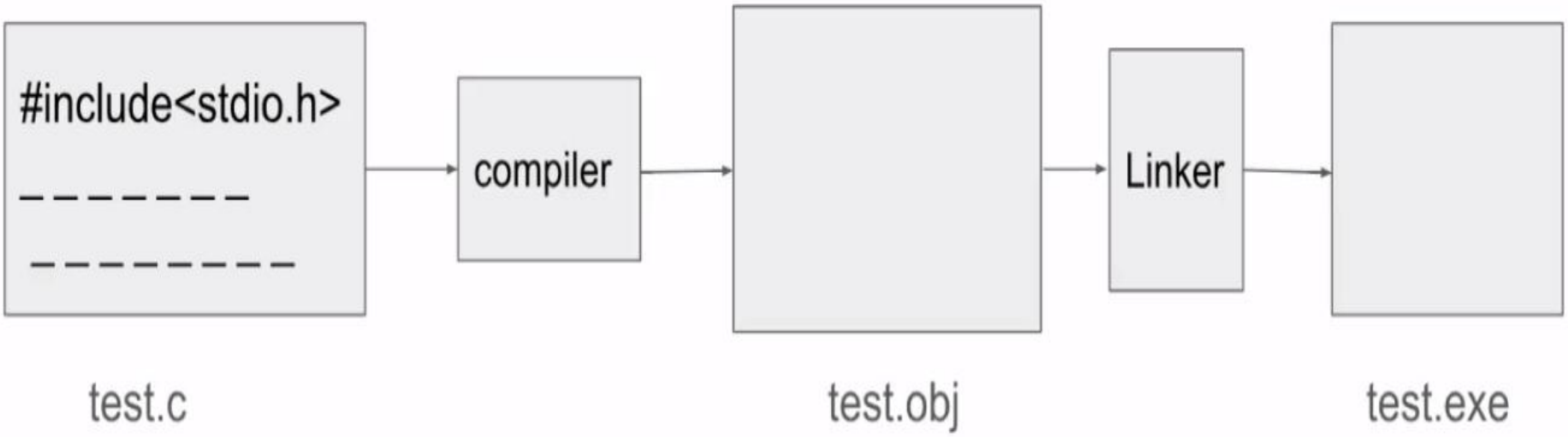
What is the complete process of compilation?



What is the complete process of compilation?

1. Compiler
2. Interpreter
3. Assembler
4. Linker
5. Loader

High level language
Assembly language
Low level language



What is complete process of creating and running code?