

Unit-I: Introduction to Computer

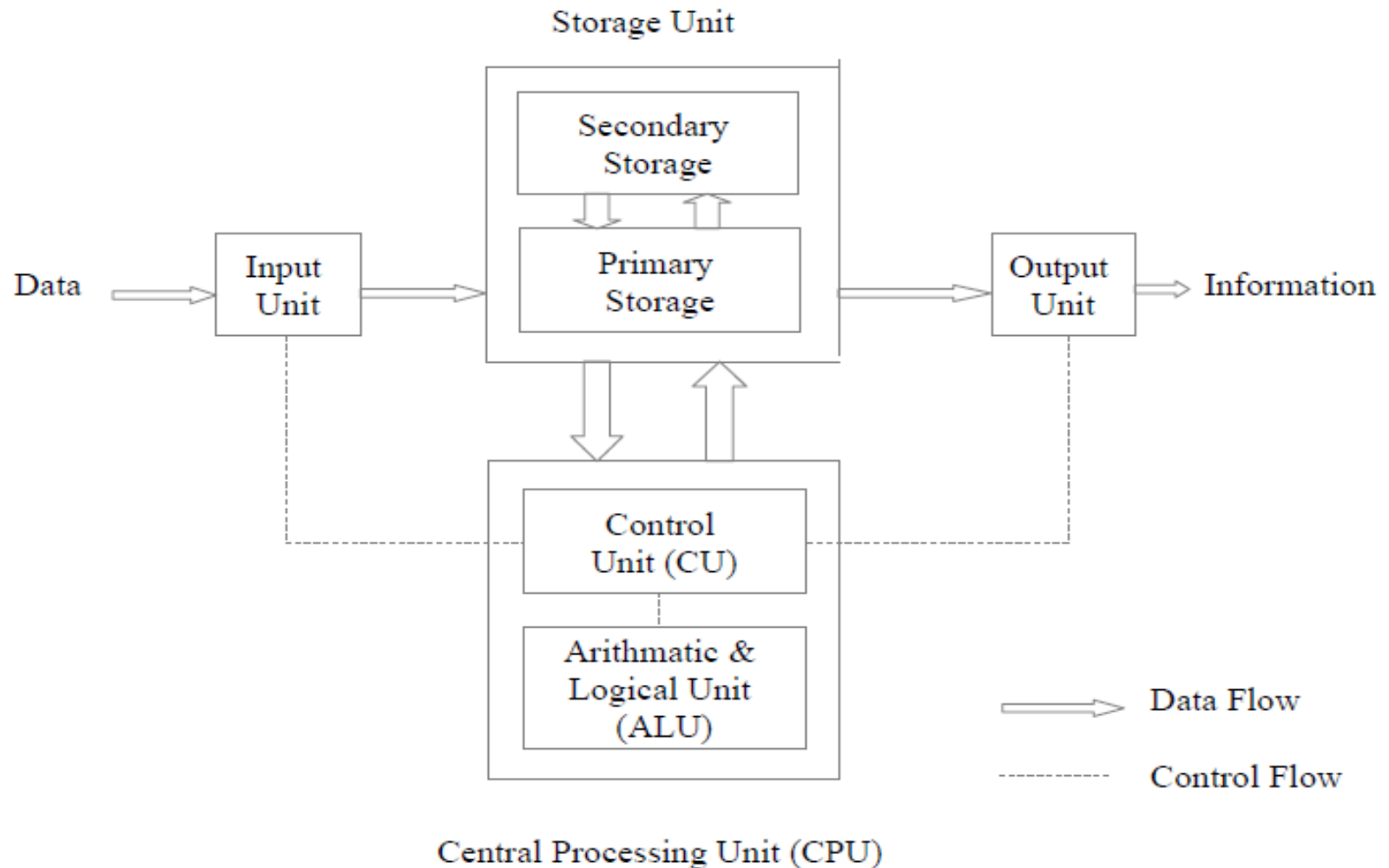
Objectives:

- To know the basic computer and its hardware and software.
- To know various system software and application software.
- To give introduction about computer programming & program planning tools.
- To aware about programming languages and its environments.
- To know about open source operating systems and programming languages.

Outline:

- Input/output devices - CPU, RAM, and Storage Devices.
- Classification of Software.- System Software, Application Software
- Program Planning Tools – Algorithm, Flowchart, Pseudo Codes.
- Software Development Life Cycle.
- Open Source Operating Systems, Programming Languages.
- Program Development Environments like BOSS and GCC.

Introduction:



Block Diagram of Computer:

Input Unit:

- The input unit of the computer system is used for **feeding data and instructions** to the computer.
- These data and instructions given to the computer are called as **input** and the devices used for giving input are called **input unit or devices**.
- The input devices of computer are **keyboard, mouse, scanner, Joystick, etc.**

Output Unit:

- The output unit of the computer system is used for **displaying or providing the information**. This information is the processed data by the **CPU**.
- This information is given **back to the user** in a **form of results** and the devices used for generating output are called **output unit or devices**.
- The output devices of computer are **monitor, printer, plotter, speaker etc.**

Storage Unit:

- Computer storage unit is often called **memory**.
- This unit consisting of **computer components** and **recording media** used to retain **digital data**.
- It is **a core function** and **fundamental component** of computers.

Primary storage

- Primary storage (also known as main memory or internal memory), often referred to simply as memory, is the only one directly accessible to the CPU.
- The CPU continuously reads instructions stored there and executes them as required.
- Any data actively operated on is also stored there in uniform manner.

Secondary storage

- Secondary storage (also known as external memory or auxiliary storage), differs from primary storage in that it is not directly accessible by the CPU.
- The computer usually uses its input/output channels to access secondary storage and transfers the desired data using intermediate area in primary storage.
- Secondary storage does not lose the data when the device is powered down—it is non-volatile.

Central Processing Unit (CPU):

- CPU is the electronic circuitry within a computer that carries out the instructions of a computer program by performing the basic arithmetic, logical, control and input/output (I/O) operations specified by the instructions.
- The fundamental operation of most CPUs, regardless of the physical form they take, is to execute a sequence of stored instructions that is called a program.
- The instructions to be executed are kept in some kind of computer memory.

Continued...

- Nearly all CPUs follow the **fetch, decode and execute** steps in their operation
- The control unit **does not execute program instructions**; rather, it directs other parts of the system to do so.
- The control unit communicates with both the ALU and memory.

ALU:

- The arithmetic logic unit (ALU) is a digital circuit within the processor that performs integer arithmetic and bitwise logic operations.
- The inputs to the ALU are the data words to be operated on (called operands), status information from previous operations, and a code from the control unit indicating which operation to perform.
- Depending on the instruction being executed, the operands may come from internal CPU registers or external memory, or they may be constants generated by the ALU itself.
- When all input signals have settled and propagated through the ALU circuitry, the result of the performed operation appears at the ALU's outputs.
- The result consists of both a data word, which may be stored in a register or memory, and status information that is typically stored in a special, internal CPU register reserved for this purpose.

Hardware:

- Hardware refers to the **physical elements of a computer.**
- This is also sometime called the **machinery or the equipment of the computer.** Examples of hardware in a computer are the **keyboard, the monitor, the mouse and the processing unit.**
- A computer's hardware is comprised of many different parts, but perhaps the most important of these is the **motherboard.**
- The motherboard is made up of even more parts **that power and control the computer.**

Software:

- Software, commonly known as **programs**, consists of all the **electronic instructions** that **tell the hardware how to perform a task**.
- These instructions come from a **software developer** in the form that will be accepted by the platform (operating system + CPU) that they are based on.
- Software is capable of performing many tasks, as opposed to hardware which only perform **mechanical tasks** that they are designed for.
- Software is the **electronic instructions** that tell the **computer to perform a task**.

Practical computer systems divide software systems into two major classes:

- **System software:**

Helps run computer hardware and computer system itself. System software includes **Operating System, Device Drivers, Editor, Compiler, Assembler, Linker, Loader** etc. System software is almost always pre-installed on your computer.

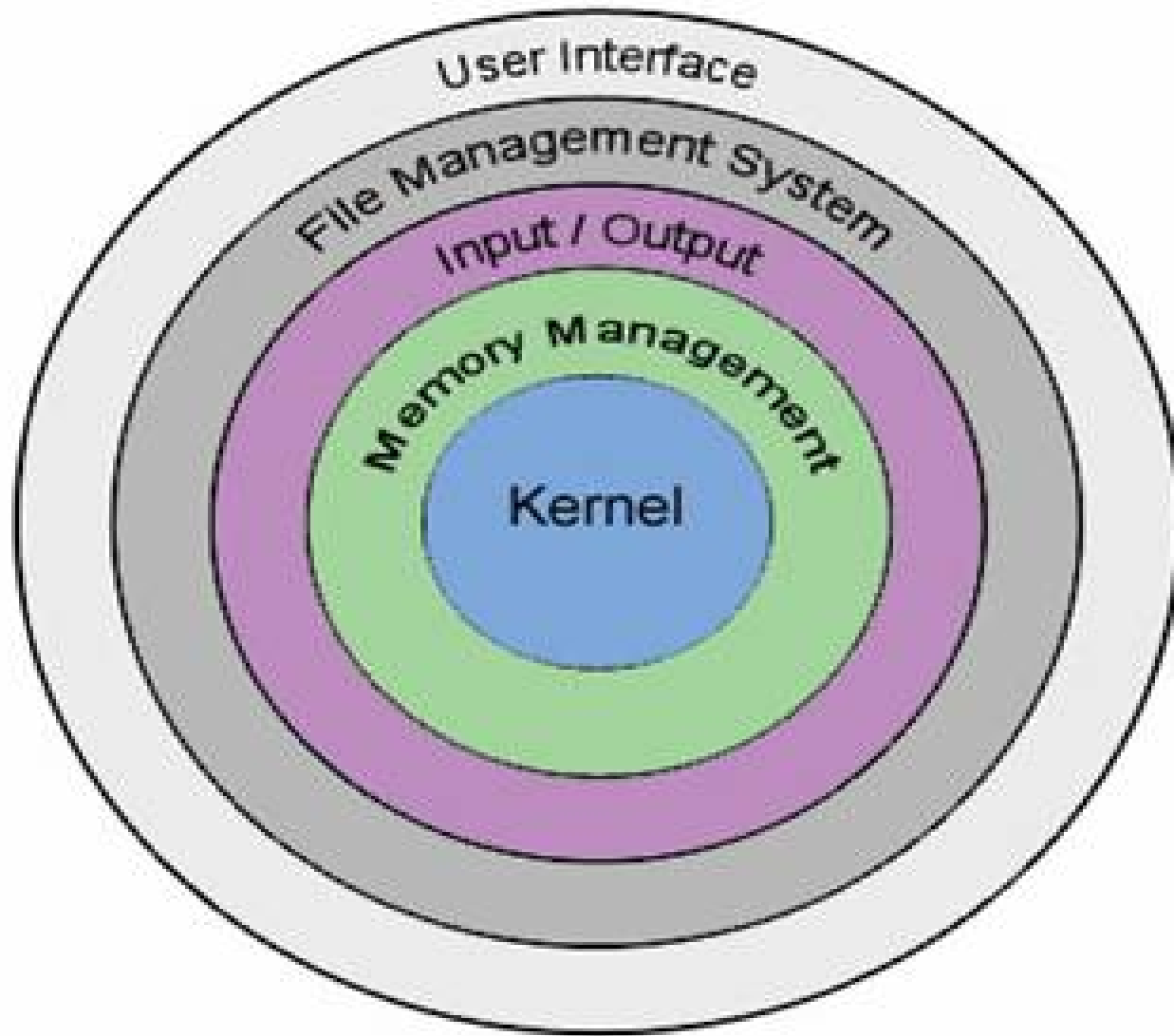
- **Application software:**

These allow users to accomplish one or more tasks. Includes **word processing, web browsing** and almost any other task for which you might install software. (Some application software is pre-installed on most computer systems)

Operating System:

- An operating system (OS) is system software that manages **computer hardware and software resources** and provides **common services for computer programs**.
- The operating system is a **component of the system software** in a computer system.
- **Application programs** usually require an **operating system to function**.
- It is one of the **strongest medium for communication between Hardware and Software of the computer system**.
- **Each and every operation** within a computer system is **carried out by Operating System**.
- Examples of popular desktop operating systems include **Apple OS X, Linux and its variants, and Microsoft Windows**. So-called mobile operating systems include **Android and iOS**.

Operating System Architecture:



Operating System Structure:

- Process Management
- Memory Management
- Secondary Storage Management
- I/O system Management
- File Management
- Protection System
- Networking
- Command Interpreter
- Operating System Services

System Calls:

- Interface between **running program** and **the OS**.
- Assembly Language Instruction (Macros and Subroutines)
- Some Higher Level Languages allow system calls to be **made directly** (e.g. C)
- Passing Parameter between running program and OS via registers, memory tables or stack
- UNIX has about 32 system calls (**e.g read(), write(), open(), close(), fork(), exec(), ioctl()...**)

Operating System Services:

- Services that provide user interfaces to OS.
- **Program Execution:** Load program into memory and run it
- **I/O Operations:** Since user can not execute I/O operations directly.
- **File System Manipulation:** read, write, create, and delete files.
- **Communication:** interprocess and intersystem

Continued...

- **Error Detection:** In hardware, I/O devices, user program Services for providing efficient system operations.
- **Resource Allocation:** for simultaneously executing jobs.
- **Accounting:** for account billing and usage statistics.
- **Protection:** ensure access to system resources is controlled.

System Program:

- Convenient environment for program development and execution. User view of OS is defined by system program, not system calls.
- **Command Interpreter** (sh, csh, tsh) – parses/executes other system programs.
- **File Manipulation** – copy (cp), print (lpr), compare (cmp, diff)
- **File Modification** – editing (ed, vi, emacs)
- **Application Programs** – send mail (mail), read news (rn),
- **Programming Language Support** (cc)
- **Status Information, Communication**

Editor:

- In general, an editor refers to **any program capable of editing files.**
- The term editor is commonly used to refer to **a text editor,**
- which is **a software program** that allows users to **create or manipulate plain text computer files.**
- They are often used in the **field of computer programming.**
- These text editors are often provided with **operating systems** or **software developing packages** and can be used to change **configuration files and programming language source code.**

Continued...

- Some text editors are small and simple, while other offer a broad and complex range of functionality.
- Some popular examples of editor are:
- **Windows OS:** Notepad, WordPad
- **UNIX and UNIX like OS:** vi, emacs
- **Mac OS:** SimpleText, TextEdit

Compiler:

- A compiler is a computer program (or a set of programs)
- That transforms source code written in a programming language (the source language) into another computer language (the target language),
- The latter often having a binary form known as object code.
- The most common reason for converting source code is to create an executable program.

Compiler Continued..

- 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 or machine code).
- A compiler is likely to perform many or all of the following operations:
 - **lexical analysis,**
 - **pre-processing,**
 - **parsing, semantic analysis,**
 - **code generation,**
 - **and code optimization.**

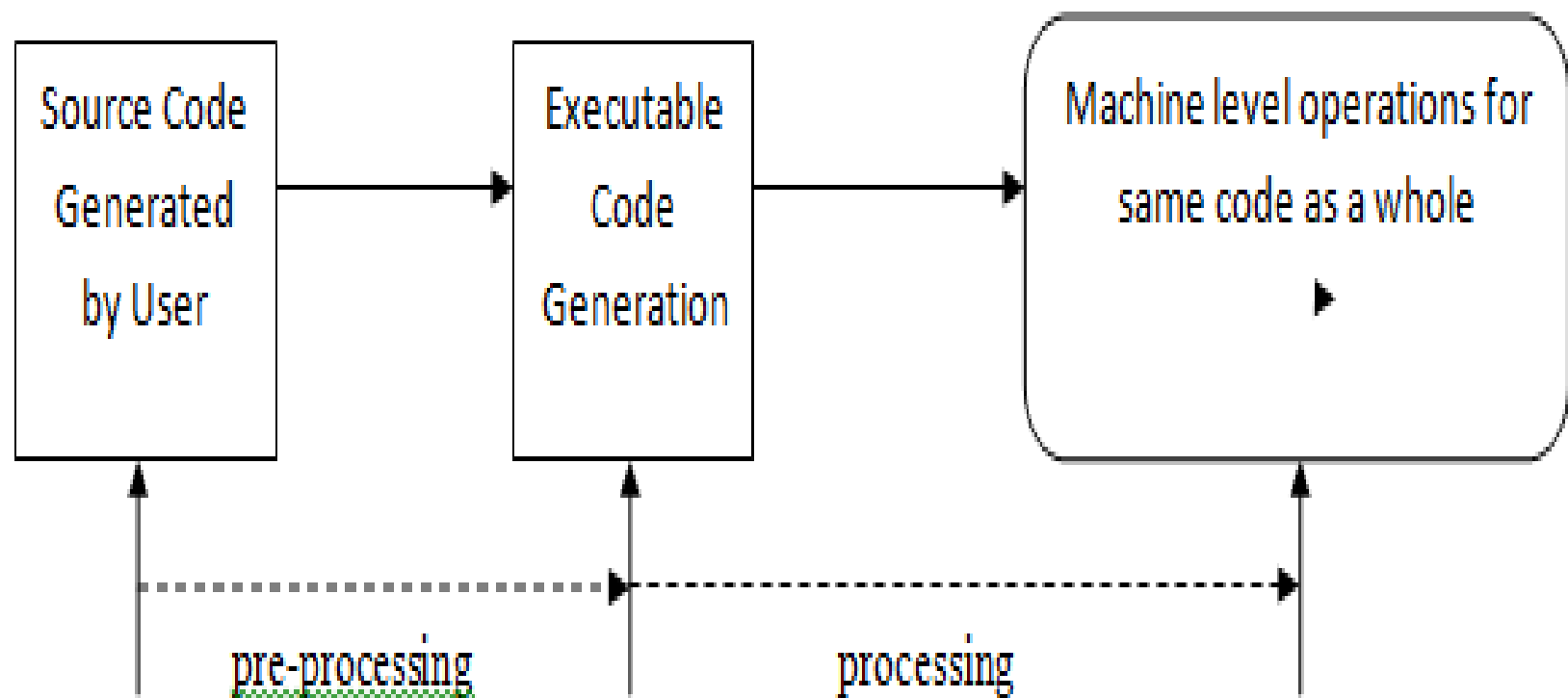
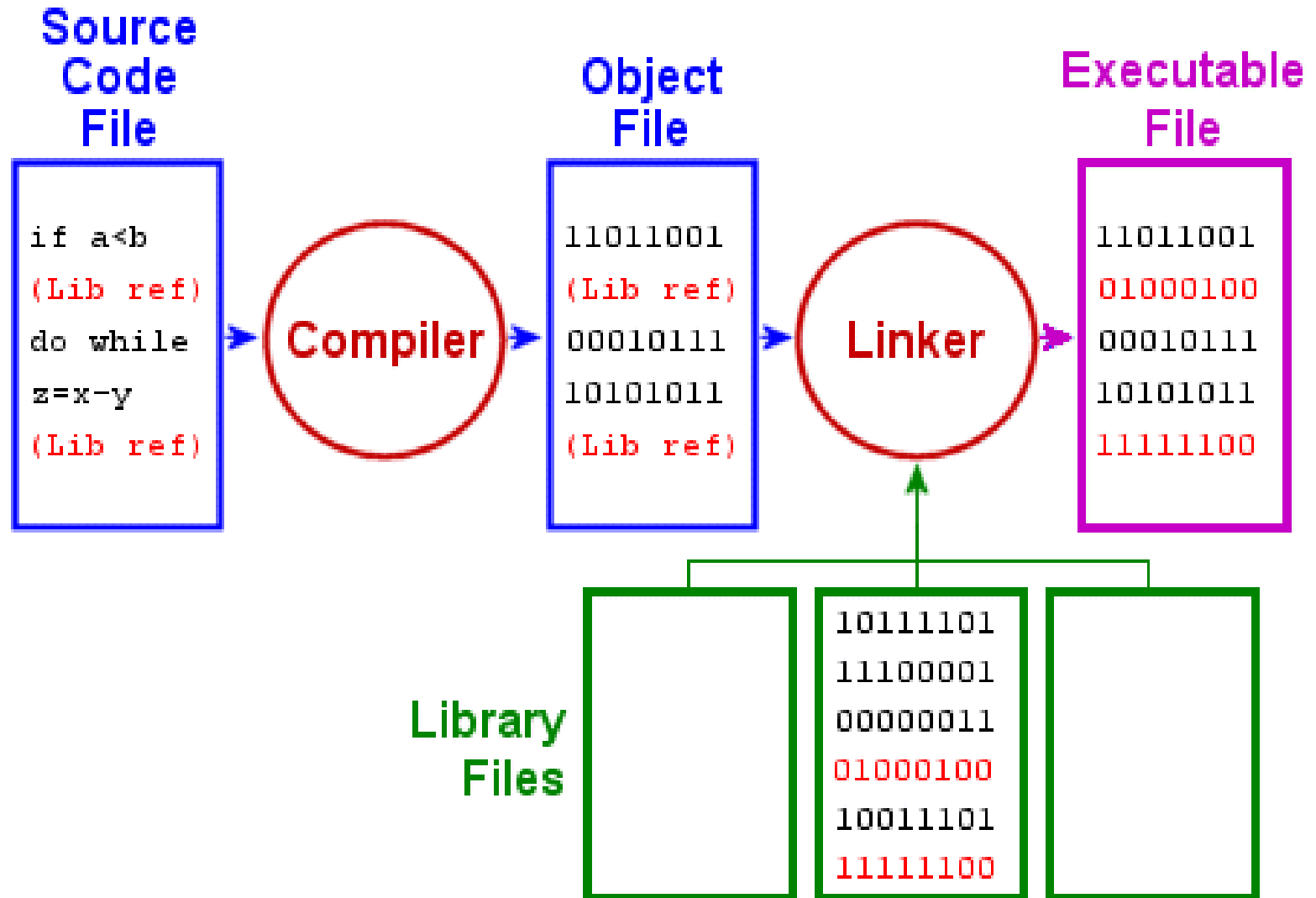


Fig. 1.3 - Compilation Process

Compiler Characteristics:

- Spends a lot of time analyzing and processing the program
- The resulting executable is some form of machine specific binary code
- The computer hardware interprets (executes) the resulting code
- Program execution is fast

Role of Compiler with Linker



Interpreter:

- An interpreter is a program which translates statements of a program into machine code. It translates only one statement of the program at a time.
- It reads only one statement of program, translates it and executes it.
- Then it reads the next statement of the program again translates it and executes it.
- In this way it proceeds further till all the statements are translated and executed.
- On the other hand, a compiler goes through the entire program and then translates the entire program into machine codes.

Continued....

- A compiler is 5 to 25 times faster than an interpreter.
- By the compiler, the machine codes are saved permanently for future reference.
- On the other hand, the machine codes produced by interpreter are not saved.
- An interpreter is a small program as compared to compiler.
- It occupies less memory space, so it can be used in a smaller system which has limited memory space.

Interpreter Characteristics:

- Relatively **little time is spent** analyzing and processing the program
- The resulting code is some **sort of intermediate code**
- The resulting code is interpreted by **another program**
- Program execution is relatively **slow**

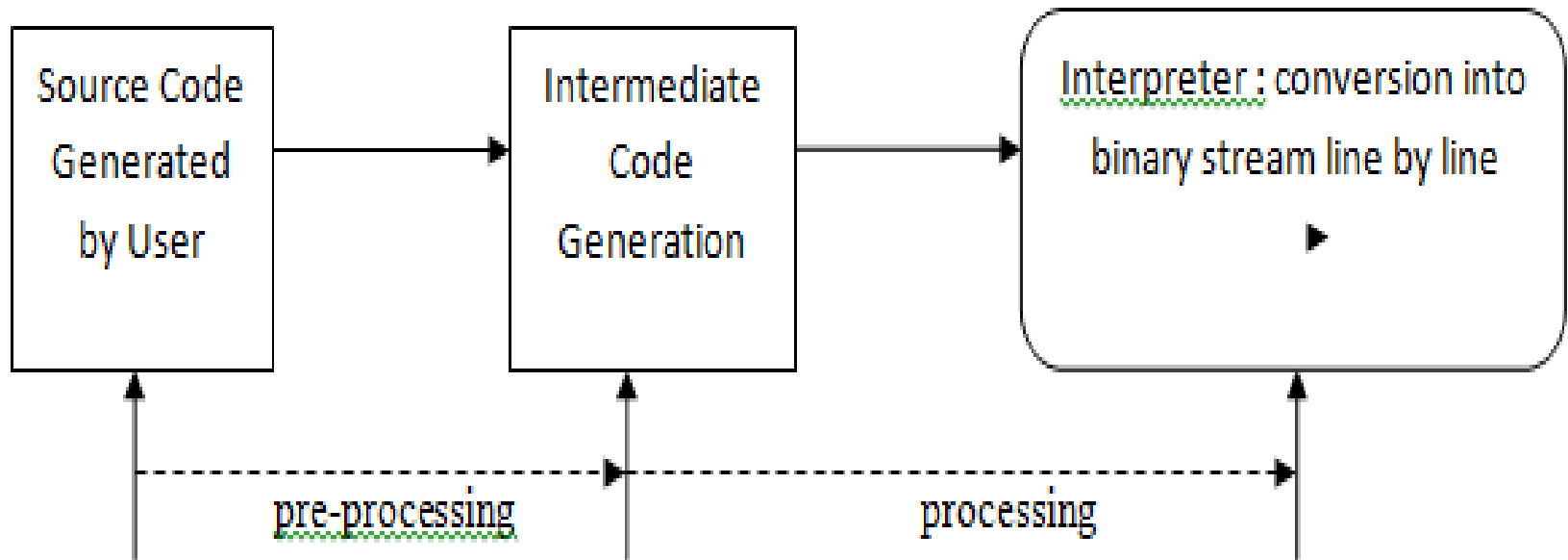


Fig. 1.5 - Interpretation Process

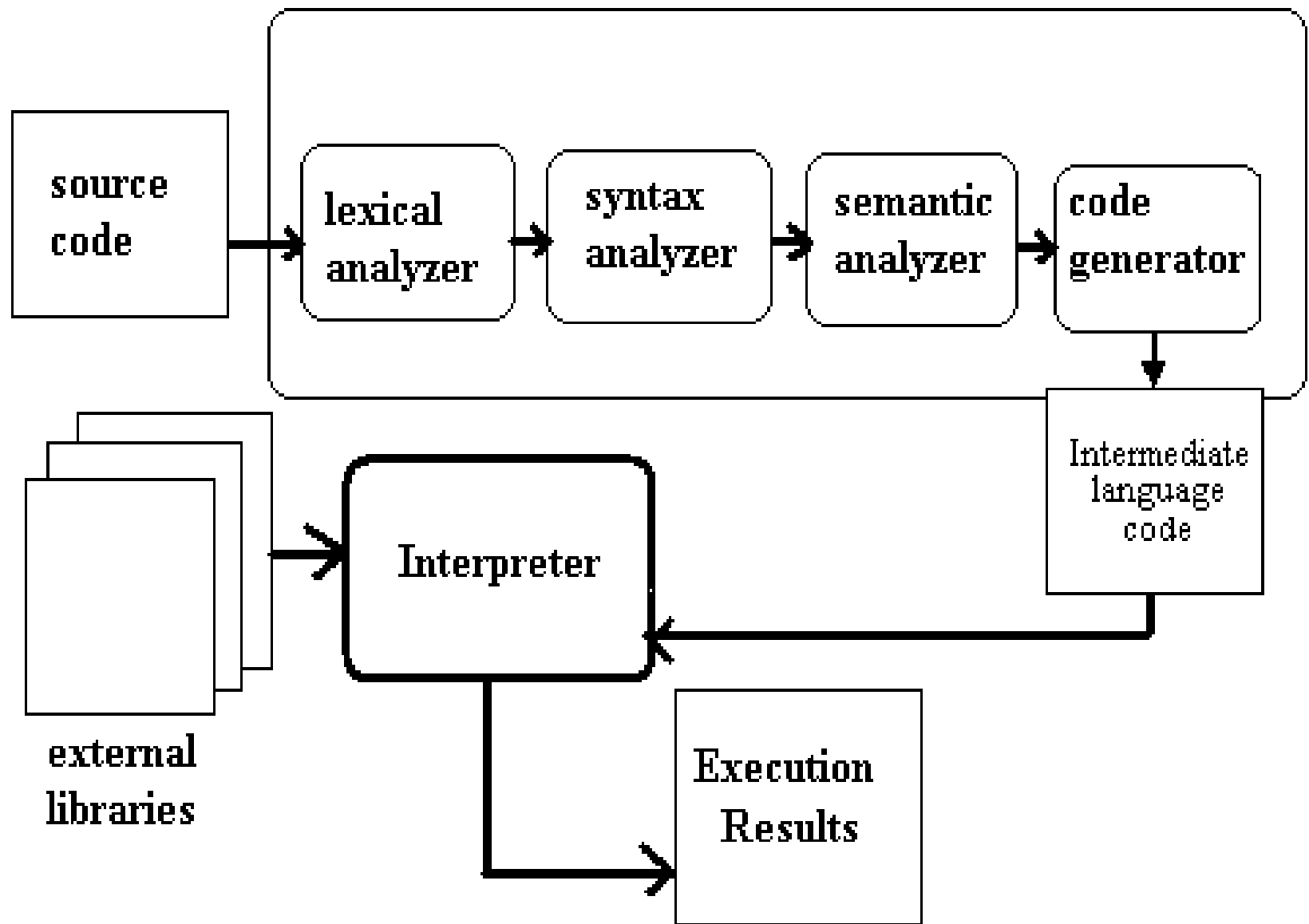


Fig. 1.6 – Role of Interpreter with Compiler

Assembler:

- A computer, other than its machine **will not understand any program written in a language.**
- The programs written in other languages **must be translated into the machine language.** Such translation is performed with the **help of software.**
- A program which translates an **assembly language program into a machine language program** is called an assembler.
- If an assembler which runs on a computer and produces the machine codes for the **same computer** then it is called **self assembler or resident assembler.**
- If an assembler that runs on a computer and produces the machine codes for **other computer** then it is called **Cross Assembler.**

Continued...

- Assemblers are further divided into two types:
One Pass Assembler and Two Pass Assembler.
 - One pass assembler is the assembler which assigns the memory addresses to the variables and translates the source code into machine code in the first pass simultaneously.
 - A Two Pass Assembler is the assembler which reads the source code twice. In the first pass, it reads all the variables and assigns them memory addresses. In the second pass, it reads the source code and translates the code into object code.

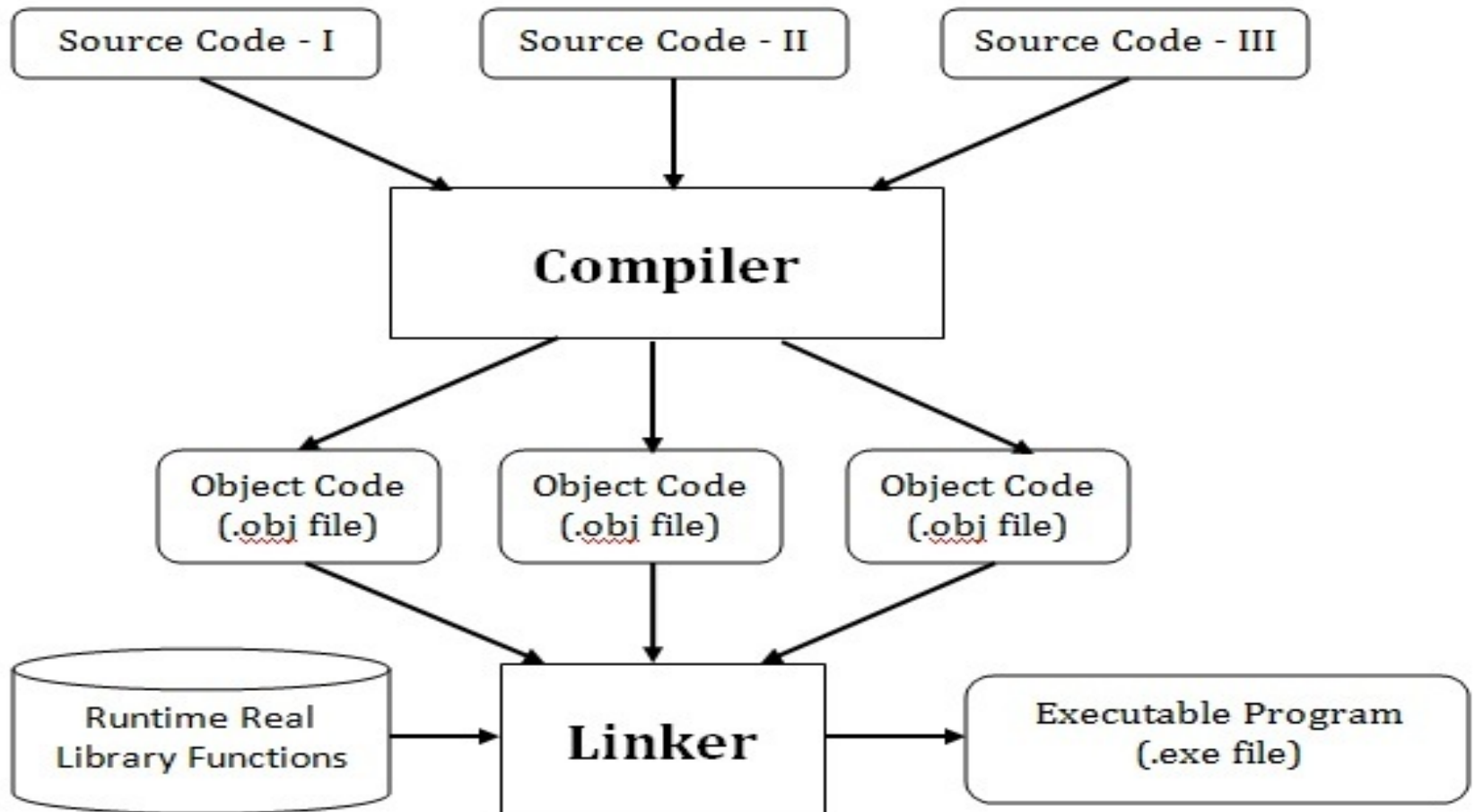
Linker:

- In high level languages, some built in header files or libraries are stored.
- These libraries are predefined and these contain basic functions which are essential for executing the program.
- These functions are linked to the libraries by a program called Linker.
- If linker does not find a library of a function then it informs to compiler and then compiler generates an error.
- Ex:(`stdio.h`) for `Printf` and `Scanf` functions.

Continued...

- The compiler automatically invokes the linker as the last step in compiling a program.
- Not built in libraries, it also links the user defined functions to the user defined libraries.
- Usually a longer program is divided into smaller subprograms called modules. And these modules must be combined to execute the program.
- The process of combining the modules is done by the linker.

Role of Linker with Compiler



Loader:

- Loader is a program that loads machine codes of a program into the system memory.
- In Computing, 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.
- Because it places programs into memory and prepares them for execution.

Continued...

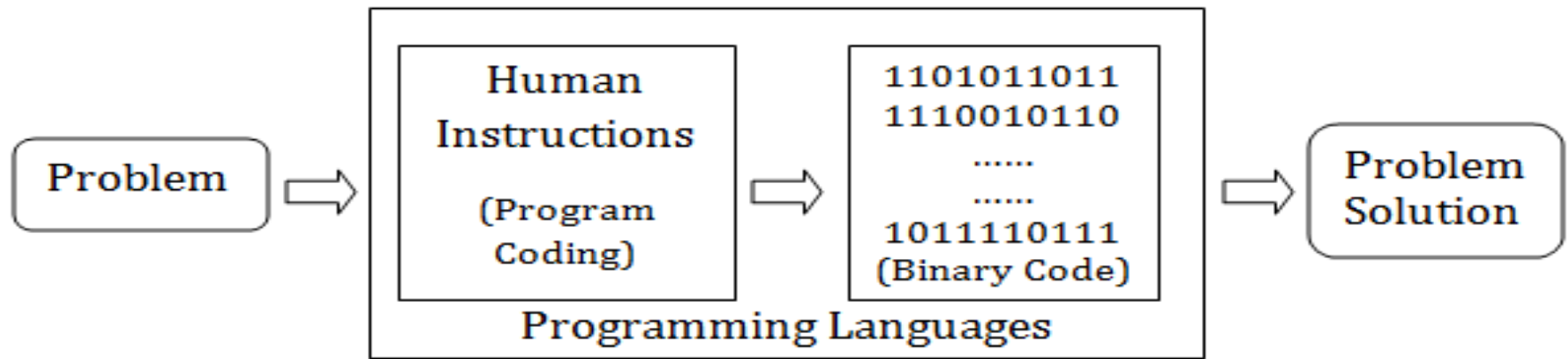
- Loading a program involves reading the contents of executable file into memory.
- Once loading is complete, the operating system starts the program by passing control to the loaded program code.
- All operating systems that support program loading have loaders.
- In many operating systems the loader is permanently resident in memory.

Introduction to computer programming

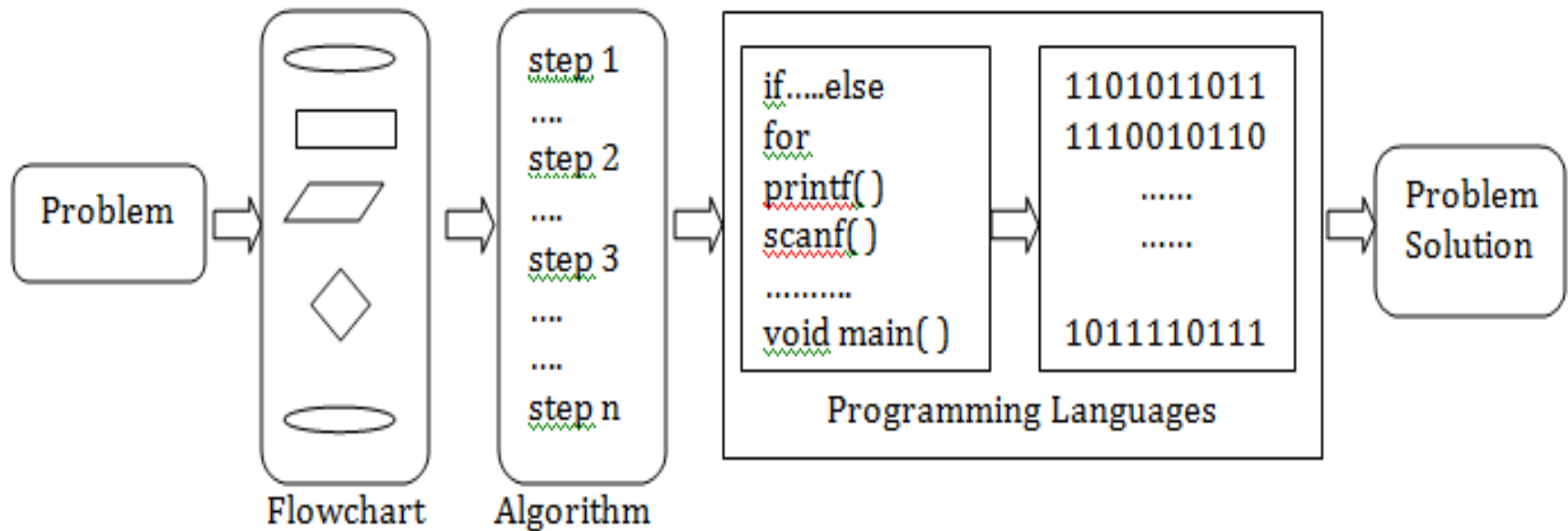
- Computer programmers write programs in a variety of computer languages, such as **C, C++ and Java.**
- Computer programmers write code **to create software programs.**
- They turn the program designs created by software developers and engineers **into instructions that a computer can follow.**
- Basically, writing software (computer program) involves describing **various processes and procedures;**

Continued...

- It involves designing that software with the help of **algorithms, flowcharts, pseudo codes.**
- Computer programming involves developing **lists of instructions** - the source code representation of software.
- These instructions use different types of objects, e.g., **numbers, words, images, sounds, etc...**



Using Programming Languages



Using Programming Languages along with Flowchart and Algorithm

Introduction to program planning tools:

- The program planning tools provides a **systematic and a simplified programming approach** for any problem.
- These tools give step by step guidance to **solve or to rectify that problem.**
- They help to **design and develop a program** for the problem.
- Program Planning Tools includes **Algorithm, Flowchart and Pseudocode.**

A typical programming task can be divided into two phases:

–Problem solving phase

- It produces an ordered sequence of steps that describe solution of problem; this sequence of steps is called an algorithm.

–Implementation phase

- Implement the program in some programming language.

Algorithm:

- It is most general sense; an algorithm is **any set of detailed instruction** which result in a **predictable end-state** from a **known beginning**.
- Algorithm is only as good as the instruction given, however, and the **result will be incorrect if the algorithm is not properly defined**.
- A common example of an algorithm would be instruction for **assembling a model airplane**.

- Given the starting set of a number of marked pieces, one can follow the instruction given to result in a predictable end-state:
 - the completed airplane.
 - Misprints in the instruction, or a failure to properly follow a step will result in a faulty end product
- It is an effective method that can be expressed within **a finite amount of space and time.**
- These are a **set of rules** that precisely defines a sequence of operations.
- There are common classes that algorithm are frequently agreed to belong to. Among these are:

Types of Algorithms

- *Dynamic Programming Algorithm:* This class remembers older result and attempts to use this to speed the process of finding new result.
- *Greedy Algorithm:* Greedy algorithm attempt not only to find a solution, but to find the ideal solution to any given problem.
- *Brute Force Algorithm:* The brute force approach starts at some random point and iterates through every possibility until it finds the solution.

- *Randomized Algorithm:* Randomized algorithm includes any algorithm that uses a random number at any point during its process.
- *Branch and Bound Algorithms:* Branch and bound algorithms forms a tree of sub problems to the primary problem, following each branch until it is either solved or lumped in with another branch.
- *Simple Recursive Algorithms:* This type of algorithm goes for a direct solution immediately, and then backtracks to find a simpler solution.
- *Backtracking Algorithms:* Backtracking algorithm test for a solution, if one is found the algorithm has solved, if not it recurs once and tests again, continuing until a solution is found.

- *Divide and Conquer Algorithms*: A divide and conquer algorithm is similar to a branch and bound algorithm, except it uses the backtracking method of recurring a set of problem and with dividing a problem into sub problems.
- In addition to these types of algorithms, they may also be divided into two primary groups.
 - *Serial Algorithms*: Which are designed for serial execution, where as each operation is enacted in a linear order.
 - *Parallel Algorithms*: Used with computers running parallel processors, where as a number of operations are run in parallel.




Example Of Algorithm

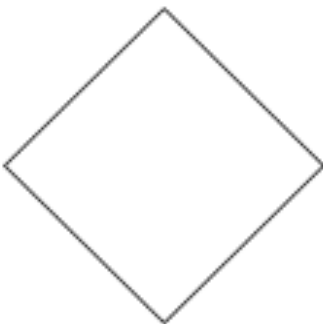

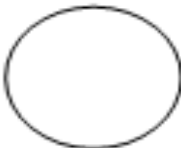
- The below example shows an algorithm to print sum of digits for a given number.
- Step 1: Input N
- Step 2: Sum = 0
- Step 3: While (N != 0)
 - $\text{Rem} = \text{N} \% 10;$
 - $\text{Sum} = \text{Sum} + \text{Rem};$
 - $\text{N} = \text{N} / 10;$
- Step 4: Print Sum

Flowchart:

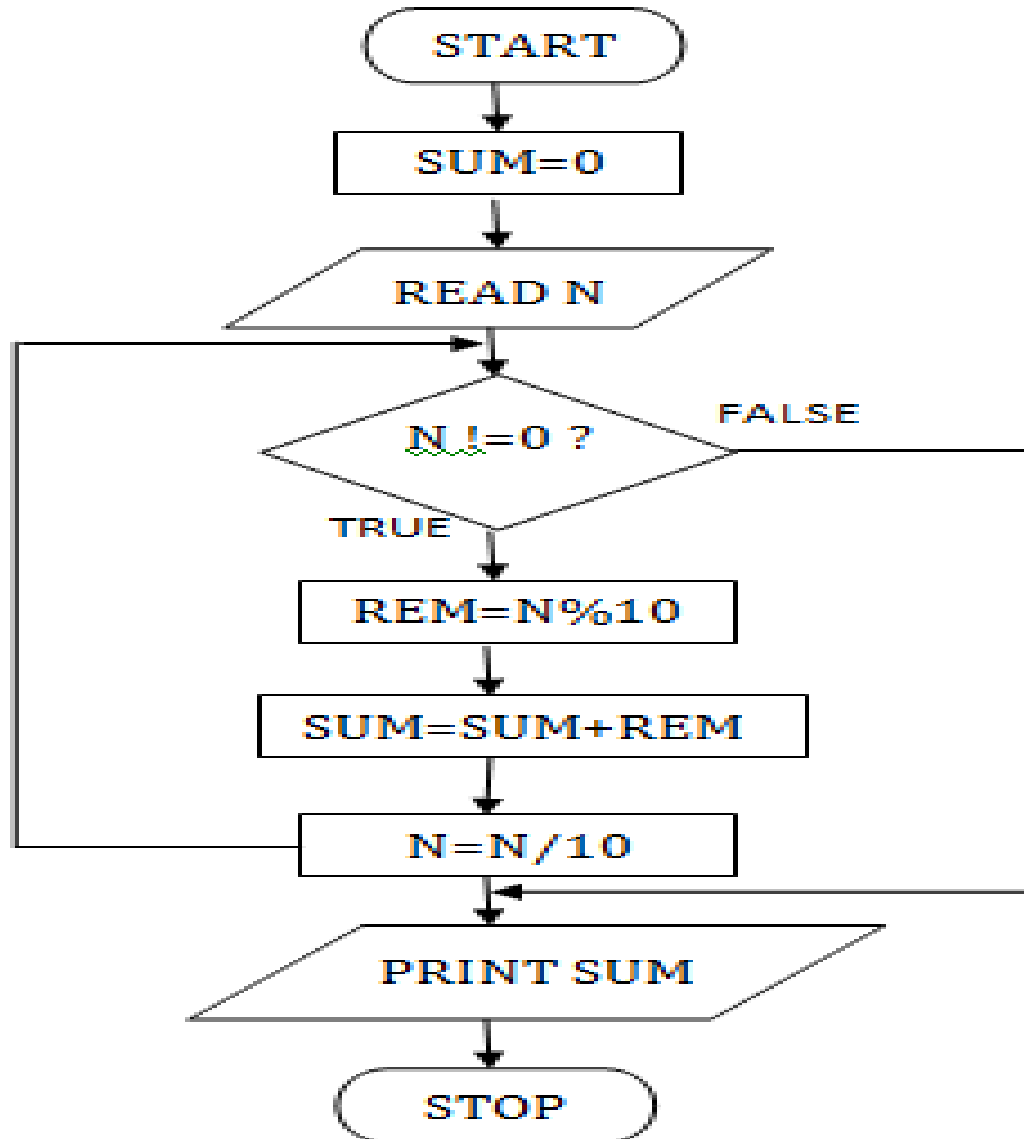
- A schematic representation of a sequence of operations, as in a computer program;
- More over a graphical representation of the sequence of operations in an information system or program.
- Information system flowcharts show how data flows from source documents through the computer to final distribution to users.
- Program flowcharts show the sequence of instructions in a single program or subroutine.
- Different symbols are used to draw each type of flowchart.
- A Flowchart shows logic of an algorithm and emphasizes individual steps and their interconnections. It shows control flow from one action to the next.

Some of the basic flowchart symbols are:

| S. No. | Notation | Symbol | Use in Flowchart |
|--------|---------------|--|---|
| 1 | Oval |  | Represented as circle, Ovals or rounded rectangles, usually containing the word “start” or “End” ,or another phrase signaling the start or end of process, such as “Submit enquiry ” or “receive product” |
| 2 | Parallelogram |  | Denotes input or output operation; Get X from the user; display X |
| 3 | Rectangle |  | Represented as rectangle. Example: “Add 1 to X ”; “replace identified <u>part</u> ”; “ <u>save changes</u> ” or similar, Indicates processing activity |

| | | | |
|---|----------------|---|--|
| 4 | Diamond |  | Represented as a diamond (rhombus). These typically contain a YES/NO question or True/False test. The arrows should always be labeled. More than two arrows can be used, but this is normally a clear indicator that a complex decision is being taken, in which case it may need to be broken-down further, or replaced with the “pre-defined process” symbol |
| 5 | Flow Line |  | Showing what's called “flow of control” in computer science. An arrow coming from one symbol and ending at another symbol represents that control passes to the symbol the arrow points to. |
| 6 | Circle (small) |  | Denotes a connector which connects two different flows to each other |

Example:



Pseudocode:

- It is a refinement that algorithm is successively **converted into step by step detailed algorithm** that is very close to a computer language.
- Pseudocode is an artificial and **informal language** that helps programmers **develops algorithms**.
- Pseudocode is very similar to **everyday English**.
- Pseudocode is an **informal high-level description of the operating principle** of a computer program or other algorithm.

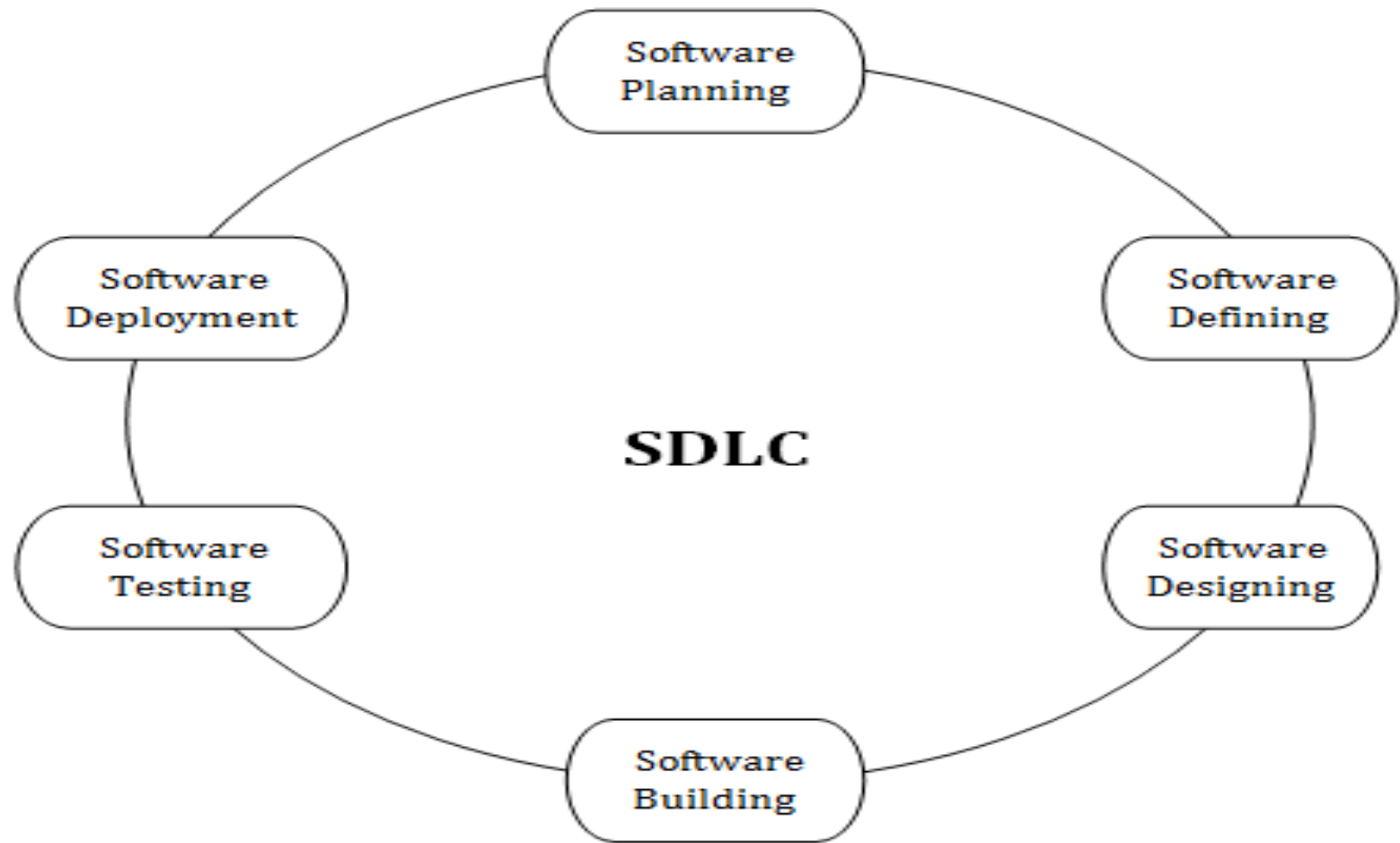
Continued...

- It uses the **structural conventions** of a normal programming language, but is intended for **human reading rather than machine reading**.
- The purpose of using Pseudocode is that it is **easier for people to understand than conventional programming language code**.
- The below example shows a pseudocode to print sum of digits for a given number.

Example

- Input a Number
- Initialize Sum to zero
- While Number is not zero
- Get Remainder by
Number Mod 10
- Add
Remainder to Sum
- Divide Number by 10
- Print sum

Software Development Life Cycle (SDLC):



Introduction to open source operating systems and programming languages:

- **Open source** refers to a program in which the **source** code is available to the general public for use and/or **modification** from its original design free of charge, i.e., **open**.
- **Open source** software is software in which the **source** code used to create the program is freely available for the public to view, edit, and redistribute.

Continued..

- Open-source software is the most prominent example of open-source development.
- Most popular open source operating systems are **Linux, Chrome OS, SteamOS, Android, Mac OS, FreeBSD** etc.
- And the most popular open source programming

BOSS

- Bharat Operating System Solutions (BOSS) is a free and open source Linux distribution developed by the Centre for Development of Advanced Computing (CDAC) of India.
- **BOSS Linux** is also known by the acronym **BOSS**.
- The latest version is 6.0. It is customized to suit Indian's digital environment & supports most of the Indian languages.
- BOSS Linux is available as Desktop edition, Server edition and as EduBOSS.
- BOSS Linux is targeted towards Government departments, schools and the first time user.
- A second target group is the Linux distribution developers so that they can develop a new distribution based on BOSS Linux.