System Software

Unit 1:-System Software Basics

1.1 Software: Application and System Software
1.2 System Software: Need of System Software, Definition
1.3 Evolution of System Software and Operating System
1.4 Component of System Software
1.5 Evolution of Programming Languages
1.6 Fundamentals of Language Processing activities: Analysis and Synthesis phase
1.7 Types of Special System Software

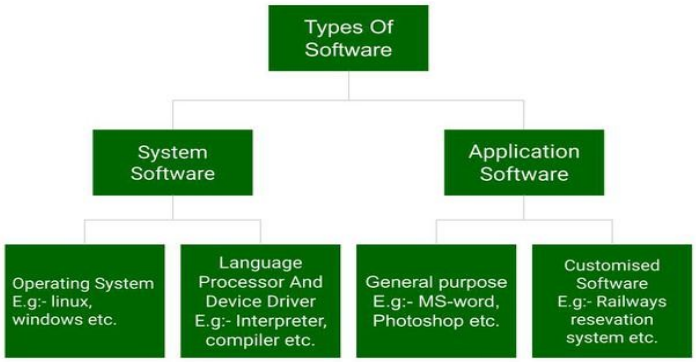
- **Software:**

- Software, is a kind of program that enable a user to perform some specific task or used to operate a computer.

- It directs all the peripheral devices on the computer system – what to do and how to perform a task. PC Software plays the role of mediator between the user and computer hardware.

- Without software, a user can't perform any task on a digital computer. A computer system can be divided into three components: the hardware, the software, and the users. Software can be further divide into mainly two parts: Application software and System Software.

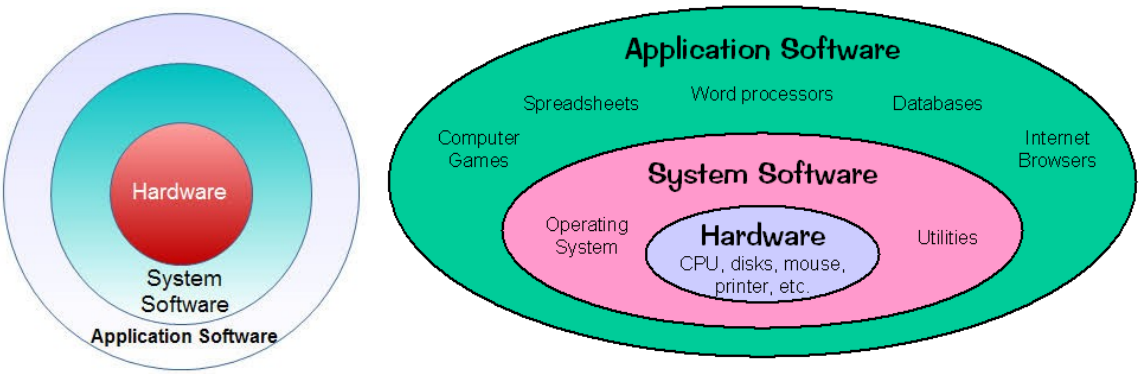
- **Types of Software**
    1. System Software
    2. Application Software



- **System Software**
    - System software is set of program that control and manage the operations of computer hardware. It also helps application programs to execute correctly.
    - System software are designed to control the operation and extend the processing functionalities of computer system.

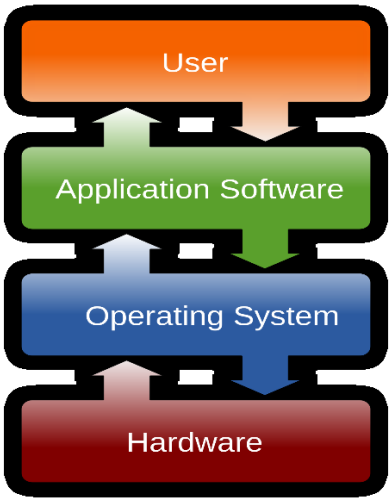
- System software makes the operation of computer more fast, effective and secure. E.g operating system, programming language processor, communication software etc
- System software are designed to manage the resource of the system, like memory m and process management, security etc
- System software is written in low-level language like a machine or assembly language. There are capable of running independently.
- System software starts running when the system is powered on and runs until the system is powered off.
- It is difficult to design and difficult to manipulate. The speed of system software is fast.

- **Application Software**
  - Application software is a program that does real work for the user. It is mostly created to perform a specific task for a user.
  - Application software acts as a mediator between the end user and system software. It is also known as Application Package.
  - This type of software is written using a high level language like C,java,Vb.net etc
  - It is a user-specific & is designed to meet the requirements of the user and also install multiple application software on a single system software
  - Example- word processing, spreadsheet, databases etc
  - Application software can't run independently. They needs system software to run.



- **Need of System software:**
  - System software is a collection of such files and utility program that are responsible for program that are responsible for running and smooth functioning of your computer system with other hardware.
  - It is acts as a platform for other software to work, such as antivirus software, OS , compiler, disk formatting software etc
  - It is the most important type of software required to administer the resources of the computer system
  - It handles all the hardware, software and network together and managed basic functions such as storing data, retrieving files and scheduling task etc.

- It converts all human instructions into machine understandable format.



- **Difference between System software and Application Software**

Sr.No	System Software	Application Software
1	System software is used to for operating computer hardware	Application software is used by user to perform specific task
2	System software's are installed on the computer when operating system is installed.	Application software's ate installed according to user's requirements
3	In general, the user does not interact with system software because it works in the background	In general, the user interacts with application software's
4	System software can run independently. It provides platform for running application software's	Application software can't run independently. They can't run without the presence of system software.
5	Example- Compiler, assemble, debugger, drivers, operating system etc	Example- word processor, web browser, media player etc
6	Debugging is difficult	Debugging is not difficult
7	Fault-fixing is not easy, but reliable	Fault-fixing is easy and reliable
8	System programming makes machine to do work	Application programming makes system program to do work

- **Evolution of System Software and Operating System**

Evolution of System software

1. The earliest computers were entirely programmed in the machine language.
2. Programmers would write out the symbolic program on sheets of paper, hand-assemble into machine code and then toggle the machine code into the computer.
3. Assemblers solved the problem by allowing the programmers to write program in terms of symbolic names and binding the names to machine addresses.
4. The relocating loader allowed the users of the sub-programs to write each sub-program as it starts at location zero.

5. Linkers and loaders divided up the work, with linker doing part of address binding, assigning address with each program and the loader doing a final relocation step to assign.
6. Linkers had to deal with object code generated by high level programming languages such as FORTRAN.
7. Multiple copies of the same program were frequently run at the same time, compilers and the assemblers were modified to create object code in multiple sections with one section for read only code another code for writable data.

Evolution of Operating System-

1. As the demands for computer till memory, devices, and files increased, the efficient management of these resources become more critical. These resources are valuable, and inefficient management of them can be costly. The management of each resource has evolved as the cost and sophistication of its use increased.
2. In simple batched system, the memory resource was allocated totally to a single program. Thus, if a program did not need the entire memory, a portion of that resource was wasted.
3. Multiprogramming operating systems with partitioned core memory were developed to circumvent this problem. Multiprogramming allows multiple program to reside in separate areas of core at the same time. Programs were given a fixed portion of core.
4. Often in such partitioned memory system some portion could not be used since it was too small to contain a program. The problem of "holes" or unused portions of core is called "fragmentation". Fragmentation has been minimized by the technique of relocatable partitions.
5. Paging is a method of memory allocation by which the program is subdivided into equal portions of pages, and core is subdivided into equal portions or blocks. The pages are loaded into blocks.
6. In simple paging all the pages of a program must be in core for execution. In demand paging a program can be executed without all pages being in core.
7. The traffic controller coordinates the processors and the processes. The resource of processor time is allocated by a program known as the scheduler.
8. The processor concerned with I/O is referred to as the I/O processor, and programming this processor is called I/O programming

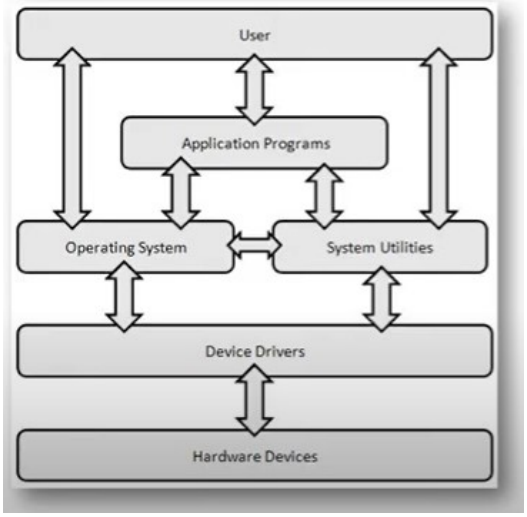
Types of Operating System –

- Batch Operating System- Sequence of jobs in a program on a computer without manual interventions.
- Time-sharing operating System- allows many users to share the computer resources. (Max utilization of the resources).
- Distributed operating System- Manages a group of different computers and makes appear to be a single computer.
- Network operating system- computers running in different operating systems can participate in a common network (It is used for security purposes).
- Real-time operating system – meant applications to fix the deadlines.

- **Component of System Software**

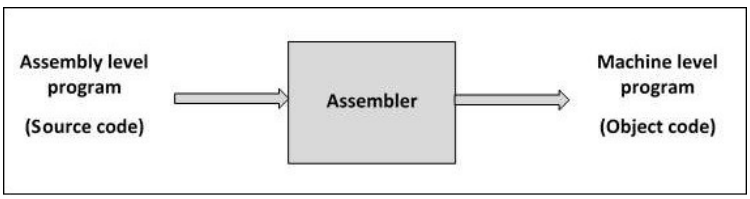
There are 10 components of system software

1. Assembler
2. Compiler
3. Interpreter
4. Editor
5. Loader
6. Linker
7. Debugger
8. Macro
9. Operating System
10. Device Driver



1. Assembler

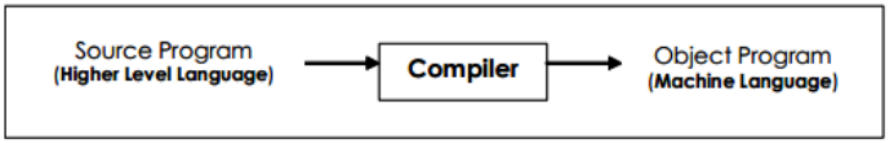
- An assembler is program that converts assembly language into machine code. It takes the basic commands and operations from assembly code and converts them into binary code that can be recognized by specific type of processor.
- An assembler enables software and application developers to access, operate and manage a computer's hardware architecture and components.
- An Assembler is sometimes referred to as compiler of assembly language. It also provides the services of an interpreter.



2. Compiler

- The compiler convert high level programming language to machine language. Example. Turbo compiler for C language.

- The compiler reads the whole source code at once, creates tokens, checks semantics, generates intermediate code, executes the whole program and may involve many passes.
- The process of compilation is very complex it is divided in to several phases. (A phase is a logical unit of work that takes as input one representation and generates another representation.)
  The phases are as follows
  1. Lexical Analysis
  2. Syntax analysis
  3. Semantic Analysis
  4. Intermediate code generation
  5. Code optimization
  6. Code Generation



3. Interpreter

- An Interpreter reads the source code one instruction or line at a this line into machine code or some intermediate form and executes it
- An Interpreter reads statement from input, converts it to intermediate code, executes it, then takes the next statement in sequence.
- If an error occurs, an interpreter stops execution and reports it.

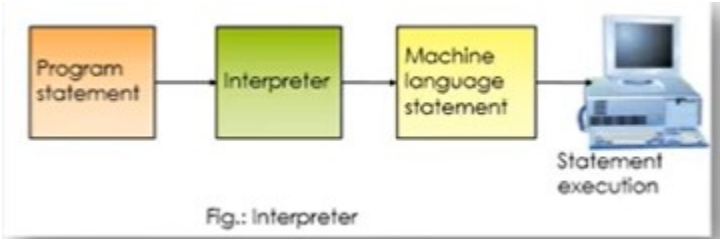


Fig.: Interpreter

4. Editor

- Editor or text editors are software programs that enable the user to create and edit text file. It is specially used for writing and editing code, traversing, viewing and displaying text.
- Example- Notepad, wordpad, line editior, screen editor, word processor etc

5. Linker

- It is a program that links user program (.c) to another program or libraries(.lib)
- It links two or more modules into the memory and preparing for execution (.obj). The task of integrating the program module together is called linking.
- It integrates necessary functions required by the program.(.obj->.exe)

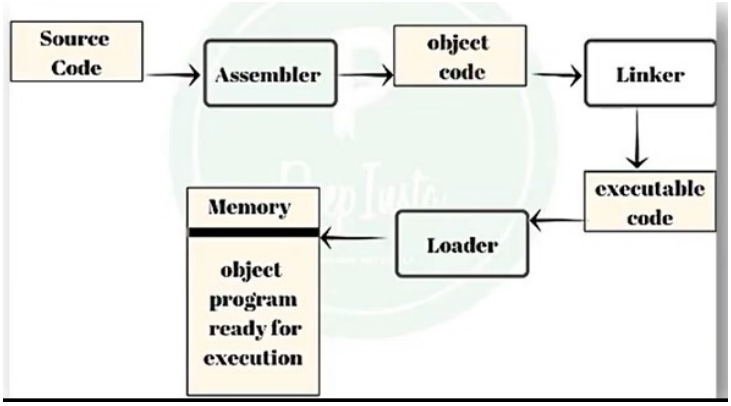


Fig. Linker & Loader

6. Loader

- A Loader reads the executable code into memory, does some address translation and ties to run the program resulting in running program or an error message (or both)
- It placing object code into main memory for execution purpose. It translate object code into executable formats(.exe)
- Functions of loader :-
- a. Allocation - The loader determines and allocates the required memory space for the program to execute properly
- b. Linking - The loader analyses and resolve the symbolic references made in the object modules
- c. Relocation- The loader maps and relocates the address references to correspond to the newly allocated memory space during execution.
- d. Loading- The loader actually loads the machine code corresponding to the object modules into the allocated memory space and makes the program ready to execute.

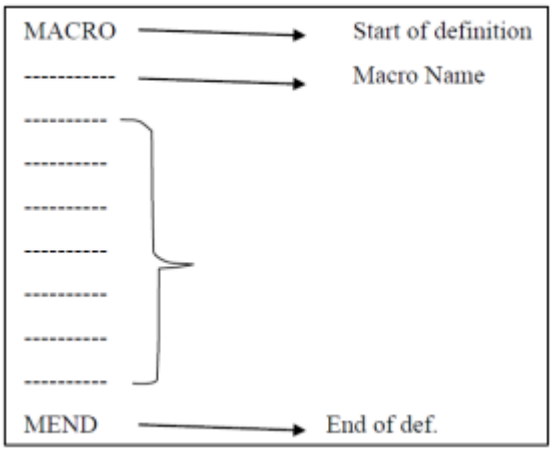
7. Debugger

- A debugger is a computer program used to find out errors which are also called as bugs in source program.
- It Provides facility to halt the program at any certain point & check the changes made in program
- It allows the user to view program line by line to identify incorrect code and find out how a program flows.
- A debugger is very useful to find sematic error in program.

8. Macro preprocessor

- It is group of instruction which can be replaced when is called by macro preprocessor.
- Provides an easy and efficient solution to be problem of repeatedly needing the sequence of instruction
- The assembly language programmer often finds that certain set of instructions get repeated often in the code.

- Instead of repeating the set of instructions the programmer can take the advantage of macro facility where macro is defined to be as "Single line abbreviation for a group of instructions" The template for designing a macro is as follows.



9. Operating System

- An operating system is the most important software that runs on a computer. It manages the computer's memory and processes, as well as all of computer without knowing how to speak the computer's language.
- Following are functions of OS:
    - Memory management
    - Processor management
    - Device management
    - File management
    - Security
    - Control over system performance
    - Job accounting
    - Error detecting aids.

10. Device Driver

- It is system program used to control number of device which are attached to the computer.
- Device driver tells OS that how the device will work on certain commands which are generated by the user. To communicate with any device such as mouse, printer, keyboard device driver are loaded in OS.
- The purpose of device drivers is to allow smooth functioning of the hardware for which it is created and to allow it to be used with different operating system.
- Example- WiFi driver, Bluetooth Driver etc.

• Evolution of Programming Language

- **1883: The Journey starts from here…!!**

– In the early days, Charles Babbage had made the device, but he was confused about how to give instructions to the machine, and then Ada Lovelace wrote the instructions for the analytical engine.

– The device was made by Charles Babbage and the code was written by Ada Lovelace for computing Bernoulli's number.
– First time in history that the capability of computer devices was judged.

- **1949: Assembly Language**
  - It is a type of low-level language.
  - It mainly consists of instructions (kind of symbols) that only machines could understand.
  - In today's time also assembly language is used in real-time programs such as simulation flight navigation systems and medical equipment eg – Fly-by-wire (FBW) systems.
  - It is also used to create computer viruses.

- **1957: FORTRAN**
  - Developers are John Backus and IBM.
  - It was designed for numeric computation and scientific computing.
  - Software for NASA probes voyager-1 (space probe) and voyager-2 (space probe) was originally written in FORTRAN 5.

- **1958: ALGOL**
  - ALGOL stands for **ALGO**rithmic **L**anguage.
  - The initial phase of the most popular programming languages of C, C++, and JAVA.
  - It was also the first language implementing the nested function and has a simple syntax than FORTRAN.
  - The first programming language to have a code block like "begin" that indicates that your program has started and "end" means you have ended your code.

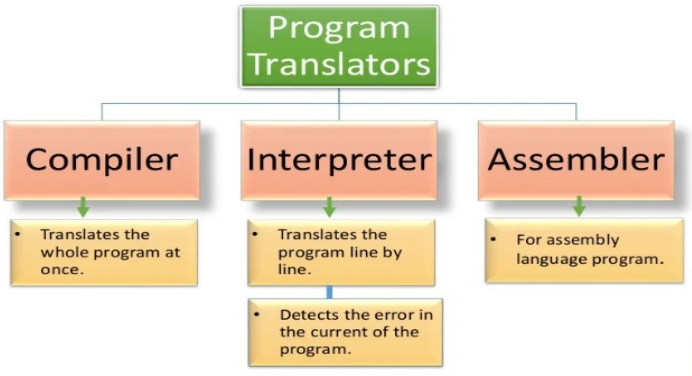
- **1959: COBOL**
  - It stands for **CO**mmon **B**usiness-**O**riented **L**anguage.
  - In 1997, 80% of the world's business ran on Cobol.
  - The US internal revenue service scrambled its path to COBOL-based IMF (individual master file) in order to pay the tens of millions of payments mandated by the coronavirus aid, relief, and economic security.
- **1964: BASIC**
  - It stands for beginners All-purpose symbolic instruction code.
  - In 1991 Microsoft released Visual Basic, an updated version of Basic
  - The first microcomputer version of Basic was co-written by Bill Gates, Paul Allen, and Monte Davidoff for their newly-formed company, Microsoft.

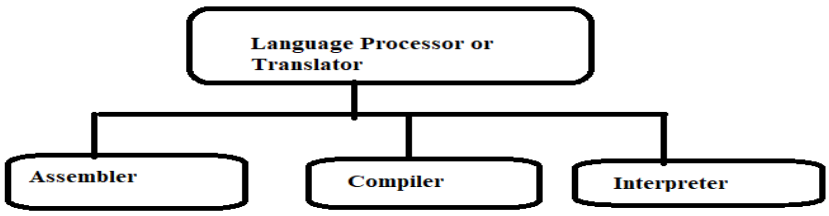
- **1972: C**
  - It is a general-purpose, procedural programming language and the most popular programming language till now.
  - All the code that was previously written in assembly language gets replaced by the C language like operating system, kernel, and many other applications.
  - It can be used in implementing an operating system, embedded system, and also on the website using the Common Gateway Interface (CGI).

- C is the mother of almost all higher-level programming languages like C#, D, Go, Java, JavaScript, Limbo, LPC, Perl, PHP, Python, and Unix's C shell.

YEAR	LANGUAGES	FACTS
1972	SQL	SQL was developed at IBM by Donald D. Chamberlin and Raymond F. Boyce. The earlier name was SEQUEL (Structured English Query Language).
1978	MATLAB	It stands for MATrix LABoratory. It is used for matrix manipulation, implementation of an algorithm, and creation of a user interface.
1983	Objective-C, C++	C++ is the fastest high-level programming language. Earlier, Apple Inc uses Objective-C to make applications.
1990	Haskell	It is a purely functional programming language.
1991	Python	The language is very easy to understand. Famous language among data scientists and analysts.
1995	JAVA, PHP, JavaScript	JAVA is everywhere. JAVA is the platform-independent language. PHP is a scripting language mainly used in web programming for connecting databases. JavaScript enables interactive web pages. JS is the most popular programming language. JS is famous for building a web application. It makes our page interactive.
2000	C#	C#(C-sharp) is mainly used for making games. Unity engine uses C# for making amazing games for all platforms
2009	GO	GO language is developed in Google by Robert Griesemer, Rob Pike, and Ken Thompson.
2011	Kotlin	Kotlin is developed by JetBrains. It is used for making an android application.
2014	Swift	Swift language is developed by Apple Inc. It is a general-purpose programming language.

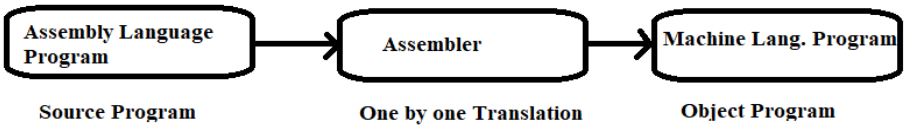
- **Language Processor(Translator)**
  - A Language processor is a special type of software program that has the potential to translate the program codes into machine codes.
  - Language such as COBOL and FORTAN have language processors, which are generally used to perform tasks like processing source code to object code.
  - A specific description of syntax, lexicon, and semantics of the high level language is required to design a language processor.





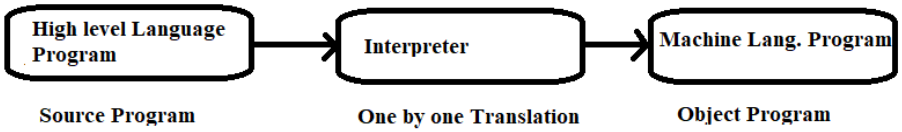
- **Assembler**

- An Assembler is a translator which is used to translate the assembly language code into machine language code.
- An Assembler language is a low level programming language where we use the symbols called Mnemonics in place of machine codes.
- The Assembler performs a one to one mapping from Mnemonic statements into machine codes and data.
- The translated program is called an Object program



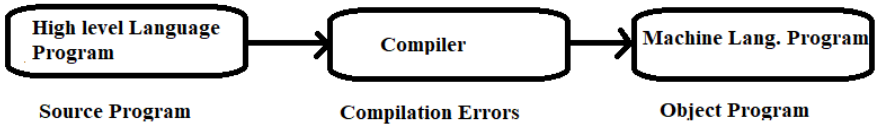
- **Interpreter**

- Interpreter converts the source program written in high level language into machine Language
- An Interpreter converts each statements of the program line by line into machine code. It means an interpreter translates the one line at a time into machine language and executes it. Then moves towards the next line. This goes on till the end of the program if no error encounters.
- Interpreter stops and highlights the problem and will not move to next line when any errors are encountered.
- An interpreter requires a less storage space in primary memory than a compiler



- **Compiler**

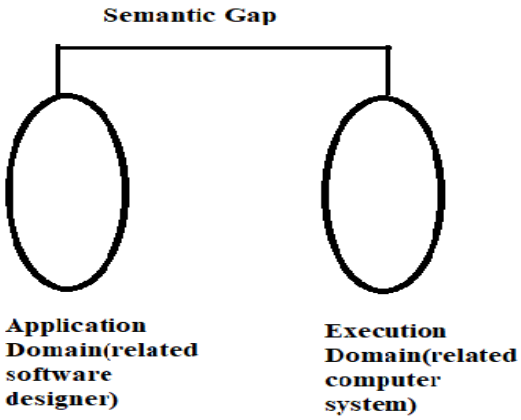
- The purpose of compiler is same as interpreter but Compiler are translator, which translate the entire program into machine codes.
- If source code program contain errors, the compiler highlights a list of errors at the end of the execution of the program
- A compiler is larger program and occupies more memory space. It is costlier than Interpreter.



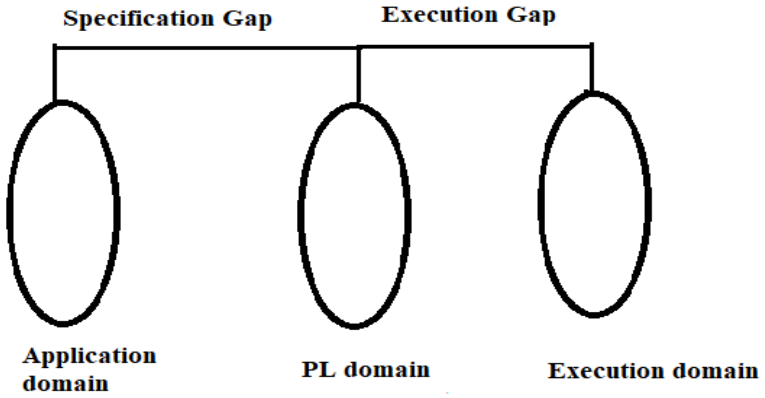
-

- **Fundamentals of Language processing activities: Analysis and Synthesis phase**

- Language processing activities arise due to the difference between the manner in which a software designer describe the ideas concerning the behavior of software and the manner in which these a software are implemented in a computer system.



- The Semantic Gap has many consequences some of the important one being large development times, large development efforts and poor quality of software.
- These issues are tackled by software engineering through the use of methodologies and programming languages (PLs).
- Semantic gap to represent the difference between the semantics of two domains.



- Software implantation using PL introduces a new domain.-PL domain

- **Specification Gap**: - The gap between the application and PL domains as specification gap or specification-design-gap. It is bridged by the software development term
- **Execution Gap: -** The gap between the PL and execution domains as Execution gap. It is a bridged by the designer of the programming language processor.

- **A Language processor is software which bridges a specification or execution gap.**

- Language processing is describe the activity performed by a language processor and assume a diagnostic capability as an implicit part of any form of language processing.

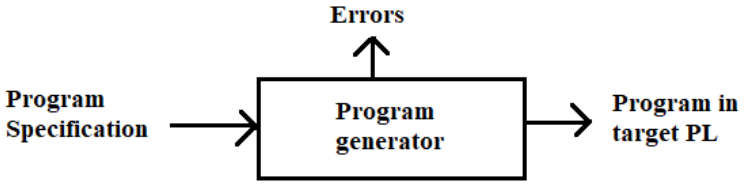
- A **Language translator** bridges an execution gap to the machine language of the computer system.

- **A Detranslator bridges** the same execution gap as the language translator but in the reverse direction

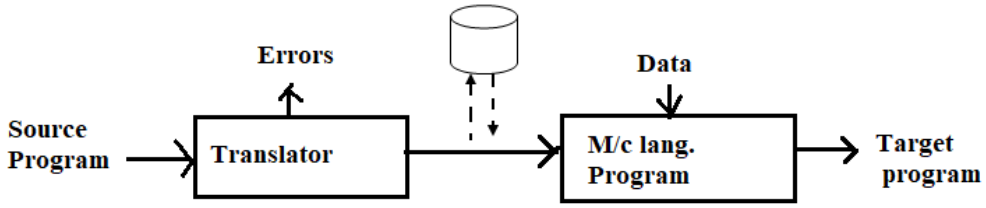
- **A preprocessor** is a language processor which bridges an execution gap but not a language translator.

- **A Language migrator** bridges the specification gap between 2 PLs

- Language Processing activity divided into 2:-
  - **Program generation activities**
  - **Program execution activities**
- **Program Generation**
  - The program generator is software system which accepts the specification of program to be generated and generates a program in the target PL.



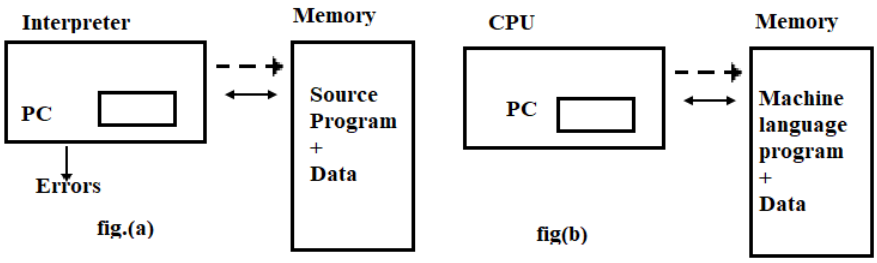
- **Program Execution- 2 model – Translation and interpretation**
  - ❖ **Program Translation**
  - The program translation model bridges the execution gap by translating a program written in PL, call source program (sp) into an equivalent program in the machine or assembly language of computer system called target program (tp).
  - Characteristics of program translation model-
    - A program must be translated before it can be executed.
    - The translated program may be saved in a file. The saved program may be executed repeatedly.
    - A program must be retranslated following modifications



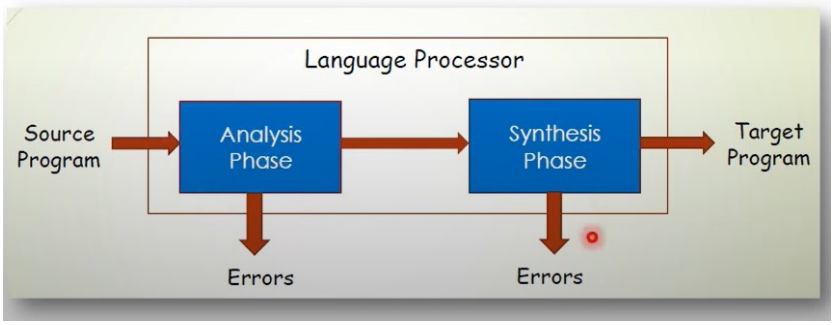
  - ❖ **Program Interpretation:** The interpreter reads the source program and stores it in its memory. During interpretation it takes a source statement, determines its

meaning and performing actions which implement it. This includes computational and input-output actions

- Fig (a) is functioning of interpreter (interpretation scheme) and fig (b) is a schematic of the execution of a machine language program by the CPU of the Computer system.
- The CPU uses a program counter(PC) to note the address of the next instruction execution cycle consisting, of following steps:
  1. Fetch the instruction
  2. Decode the instruction to determine the operation to be performed, and also its operands.
  3. Execute the Instruction.
- At end of cycle, the instruction address in PC is updated and the cycle is repeated for next instruction.
- Program interpretation can proceed in an analogous manner. Thus, the PC can indicate which statement would be subjected to the Interpretation cycle, which could consist of following steps:
  1. Fetch the statement
  2. Analyse the statement and determine its meaning, the computation to be performed and its operands.
  3. Execute the meaning of the statement



- **Fundamentals of Languages Processing Activates:**
  Language Processing = Analysis of Source Program + Synthesis of Target Program



❖ Analysis Phase:-

- The collection of language processor components engaged in analyzing a source program as the analysis phase of the language processor.
- The specification consists of 3 components:
    1. **Lexical Rules** which check the valid lexical units or tokens in the source language.
    2. **Syntax Rules** which check the information of valid Syntax in the Source language.
    3. **Semantic Rules** which check meaning with valid statement of the language.
- The analysis phase uses each component of the source language specification to determine relevant information concerning a statement in the source program. Thus, analysis of a source statement consists of lexical, syntax and semantic analysis.

Example: percent_profit=(profit*100)/cost_price

- **Lexical Units: identifiers =, *, / operators, 100 as constant,string**
- **Syntax Analysis: assignment statement with percent_profit as the left hand side and (profit* 100)/cost_price as the expression on the right hand side.**
- **Semantic Analysis: determines the meaning of the statement to be the assignment of profit* 100/ cost_price to percent_profit**

❖ **Synthesis Phase**

- Components engaged in synthesizing a target program constitutes the synthesis phase.
- The Synthesis Phase is concerned with the construction of target language statements which have the same meaning as a source statement. Typically, this contains of two main activities:

1. Creation of data structures in the target program (memory allocation)
2. Generation of target code (code generation).

• Types of System Software

Different types of system software. The details are as follows:

1**. Operating Systems**

The operating system is a system software kernel between the computer hardware and the end-user. Operating system software helps you effectively utilize all hardware and software components of your computer system.

The desktop in a modern operating system is a graphical workspace, which contains menus, icons, and applications. These menus, icons, and applications are operated by the user through a mouse-driven cursor or finger touch. Popular OSs for computers are -Windows 10, Mac OS X, Ubuntu.

2. Device Drivers

Driver software is system software that makes computer equipment and peripherals come to life. The driver enables all connected components and external accessories to perform their expected tasks by the instructions of the operating system. Without the driver, the operating system will not assign any responsibilities.

The devices that require drivers including the mouse, keyboard, soundcard, display card, network card, printer. Generally, operating systems come with drivers for most devices on the market. By default, input devices (such as mouse and keyboard) will have their drivers installed.

3. Firmware

The third type of system software is firmware. It is operating software embedded in flash memory, ROM, or EPROM memory chips, and the operating system can recognize it. It directly manages and controls all activities of any single hardware.

Traditional firmware is installed on a non-volatile chip and can only be upgraded by swapping with a new pre-programmed chip. Today, the firmware is stored in the flash memory chip and can be upgraded without replacing the semiconductor chip.

BIOS and UEFI

Today, the most important firmware in a computer has been installed on the motherboard by the manufacturer and can be accessed through the old BIOS (Basic Input/Output System) or the new UEFI (Unified Extended Firmware Interface) platform. It is the configuration interface, which is first loaded when the computer is turned on and passes the POST (Power On Self Test).

4. Programming Language Translators

These are intermediate programs that software programmers rely on to convert high-level language source code into machine language code. The former is a collection of programming languages that are easy for humans to understand and code (ie Java, C++, Python, PHP, BASIC). The latter is a complex code that only the processor can understand.