

System Software and Operating System

**Introduction
09/07/20 and 10/07/20**

Introduction

- ❖ **Software** is a set of programs, which is designed to perform a well-defined function. A program is a sequence of instructions written to solve a particular problem.
- ❖ Two types of software –
 - **System Software**
 - **Application Software**
- ❖ **System Software**
 - ✓ The system software is a collection of programs designed to operate, control, and extend the processing capabilities of the computer itself.
 - ✓ Acts as an interface between hardware and user applications.

Introduction

❖ Types of System Software

- ❑ Operating Systems
- ❑ Language Processors
- ❑ Device Drivers

❖ Application Software

❖ Application software products are designed to satisfy a particular need of a particular environment.

❖ Examples of Application software are the following –

Student Record Software

Income Tax Software

Railways Reservation Software

Microsoft Word

Introduction

❖ Utility Software:

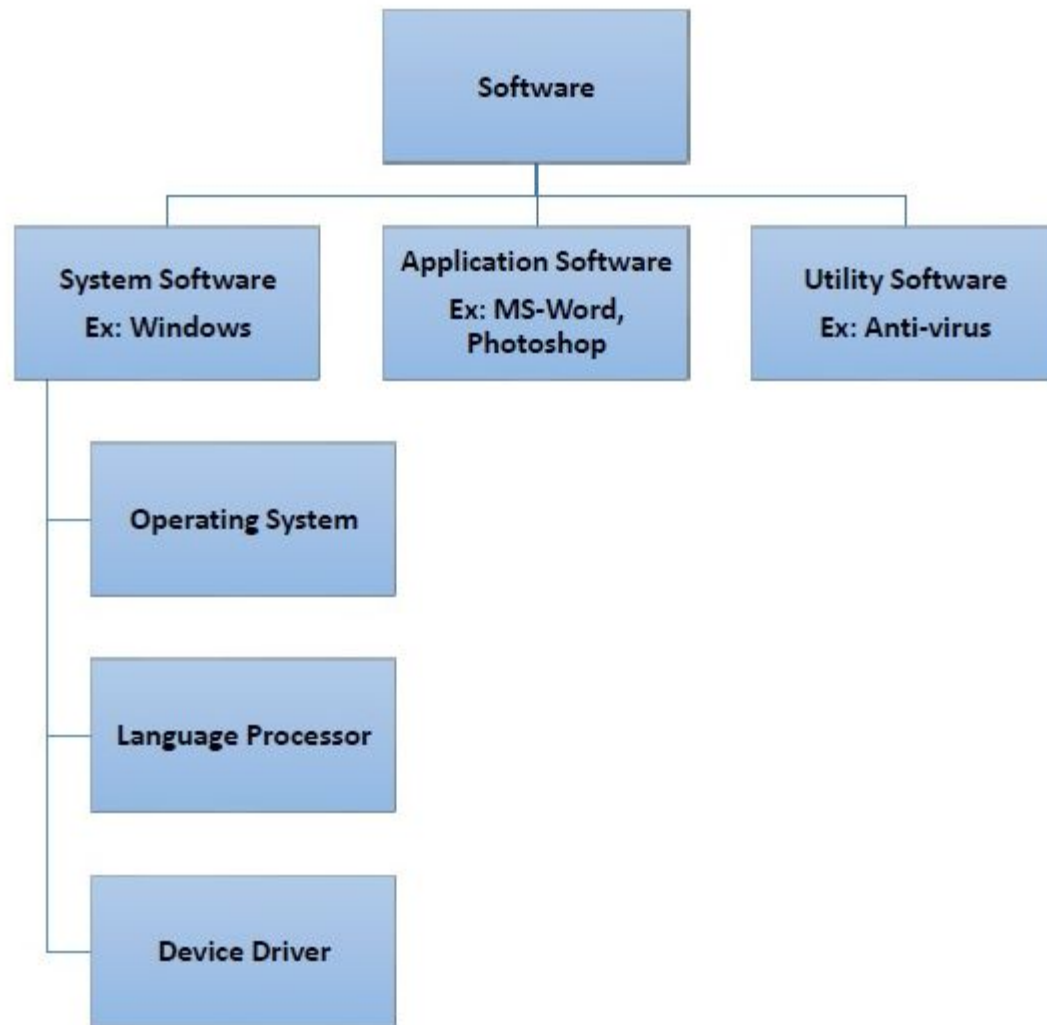
❖ Application software that assist system software in doing their work is called **utility software**.

Thus utility software is actually a cross between system software and application software.

- Antivirus software
- Disk management tools
- File management tools
- Compression tools
- Backup tools



Examples



Types of Computer Software

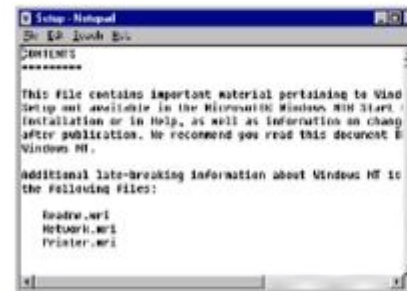


MS Office consists of various applications such as Word, Excel, PowerPoint etc., These are used for creating documents, spreadsheets, presentations etc.,

Copyright 2015 by Next Education India Pvt Ltd. All rights reserved.

Types of Computer Software

Application Software



Notepad and WordPad are two other examples of application software used for creating and editing documents.

Types of Computer Software

Application Software



Chrome, Mozilla Firefox, Internet Explorer, Opera etc., are different browsers used to access the internet.

Operating System



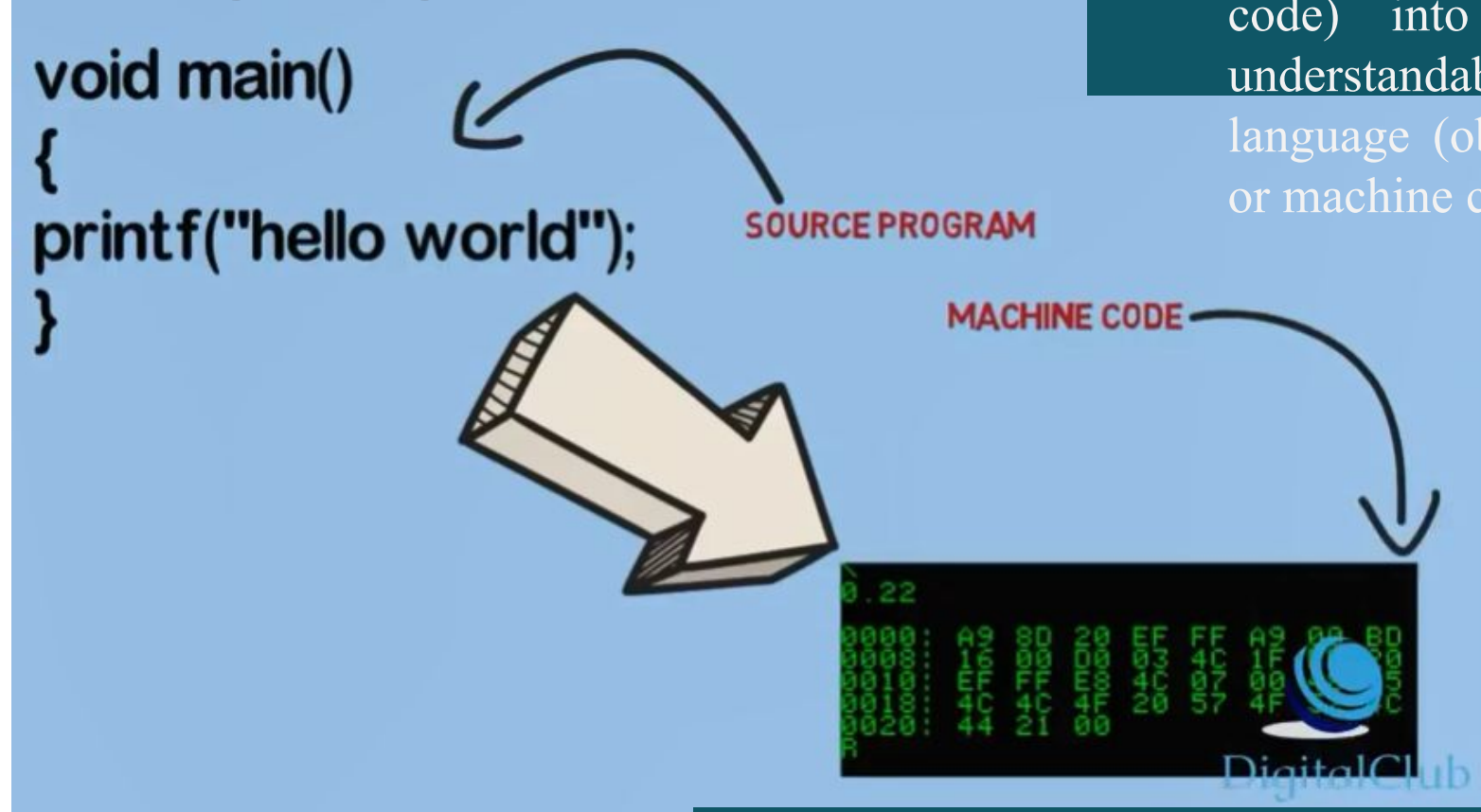
Operating system is a program that allows different applications and various pieces of hardware such as monitor, mouse, printer, keyboard, etc., to communicate with each other.

Copyright 2003 © New Education India Pvt Ltd. All Rights Reserved

- Responsible for functioning of all hardware parts and their inter-operability to carry out tasks successfully.
- First software to be loaded into computer memory
- Manages a computer's basic functions like storing data in memory, retrieving files from storage devices, scheduling tasks based on priority.

Language Processors

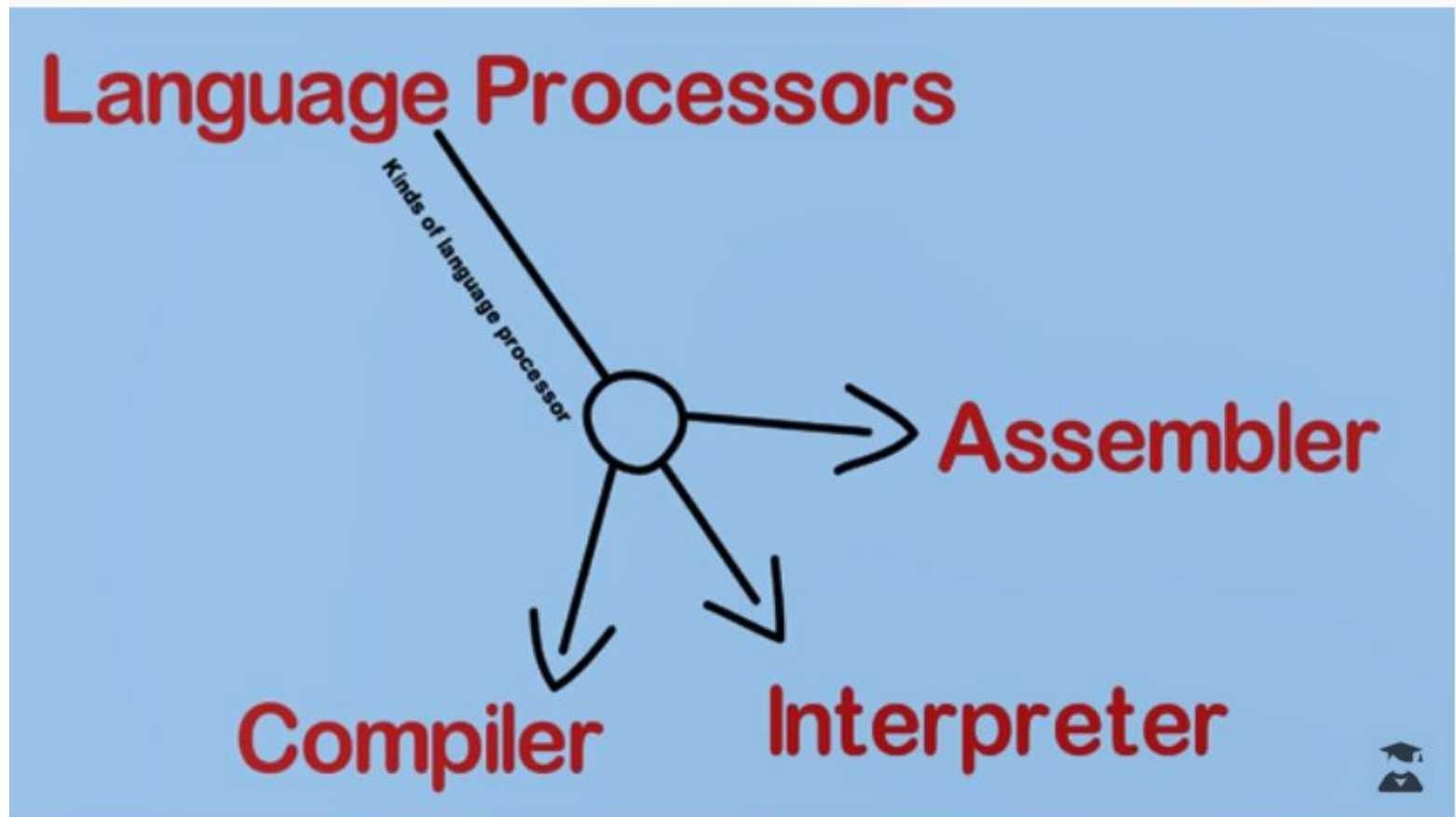
Converts all user instructions (source code) into machine understandable language (object code or machine code)



3 types of languages

- *Machine level* – a string of 0s and 1s (Machine code)
- *Assembly level* - mnemonics
- *High level* – English like statements

Types of Language Processors



Compiler is a program that translates high level language to assembly level language and generates machine code with help of linker and loader

Assembler is a program that translates assembly level language into machine code

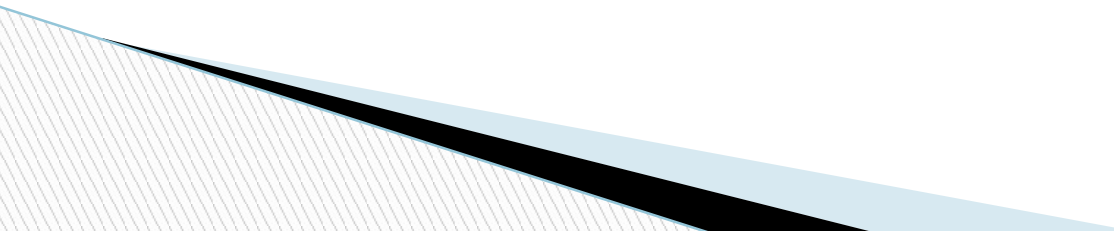
Interpreter is a program that translates high level language to machine code but one statement at a time unlike compiler who reads the whole program at once



Device Drivers

- It is the system software that controls and monitors functioning of a specific device on computer.
- Each device that needs to be attached externally to the system has a specific driver associated with it.
- When you attach a new device, you need to install its driver, so that the OS knows how it needs to be managed.

Goals of System Software

- ❑ User convenience
 - ❑ Efficient use
 - ❑ Non-interference
- 

Language Processor

13/07/20

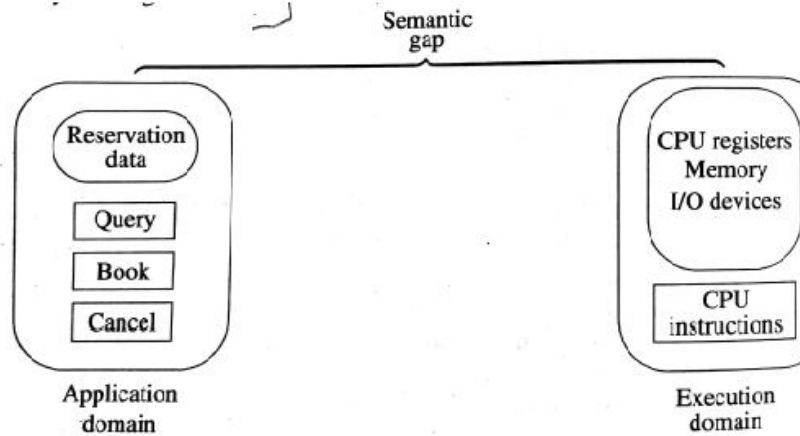


Figure 2.1 Semantic gap between domains

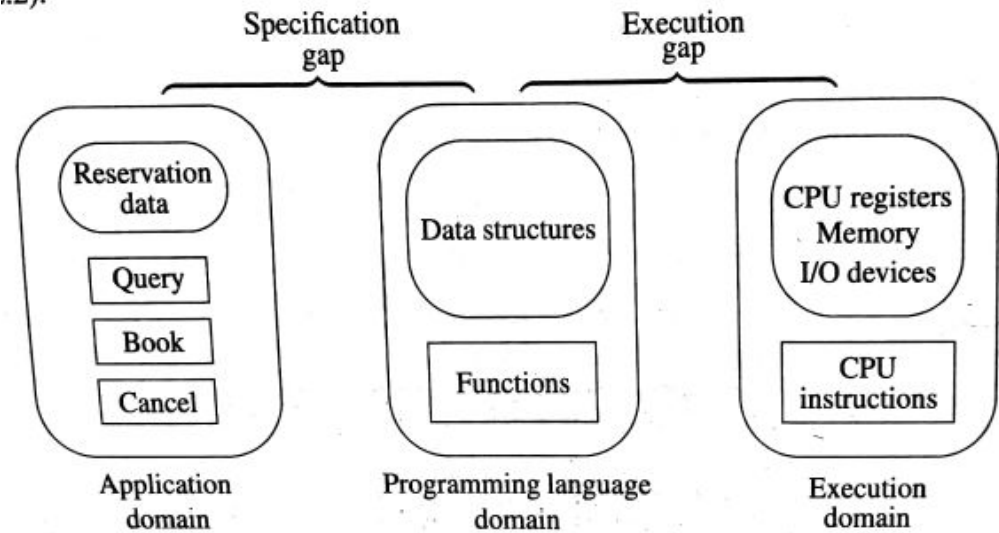


Figure 2.2 Specification and execution gaps

LANGUAGE PROCESSOR

A software that bridges a specification gap or execution gap. A similar thing, called language translator, bridges an execution gap with machine language.

=Or=

Allows the development of applications in the language most appropriate to the task, removing the need for developing at the appropriate to the task, removing the need for developing at the machine level. Programmers can ignore the machine-dependent details of programming.

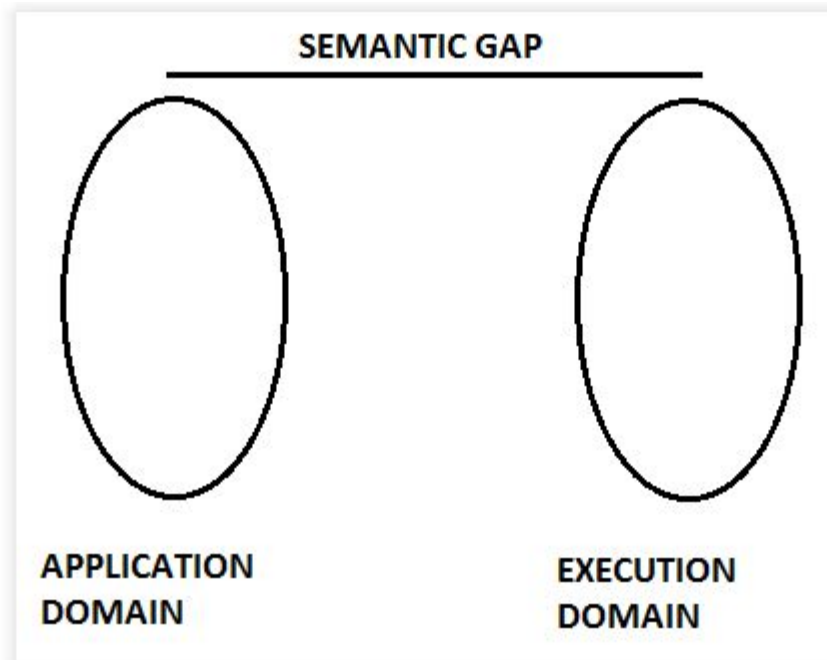
SEMANTICS & SEMANTIC GAP

Semantics : To represent the rules of meaning of a domain.

Semantic Gap : The difference between the semantics of two domains.

Application Domain : The designer describes the ideas concerning the behaviour of software in terms related to application domain of the software.

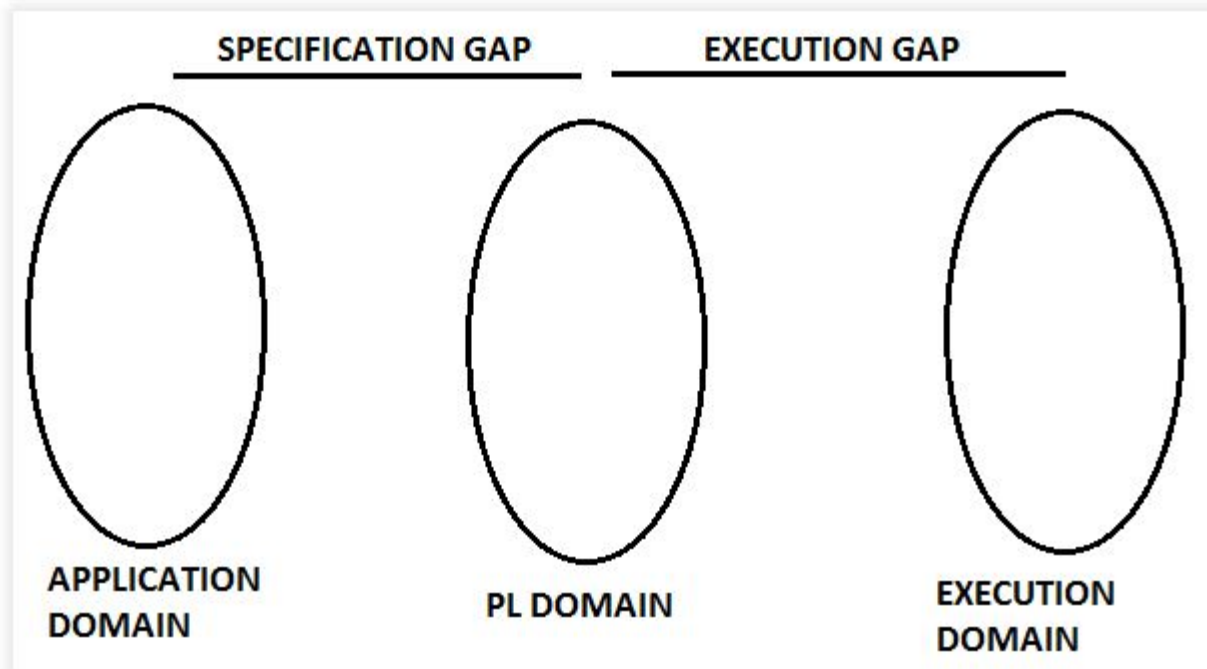
Execution Domain : To implement these ideas, their description has to be interpreted in terms related to the execution domain of the computer system.



A semantic gap is tackled using software engineering steps like

- 1) Specification, design and coding
- 2) PL implementation steps

The software implementation using the PL domain introduces the new domain called the PI domain which comes between the application domain and the execution domain.



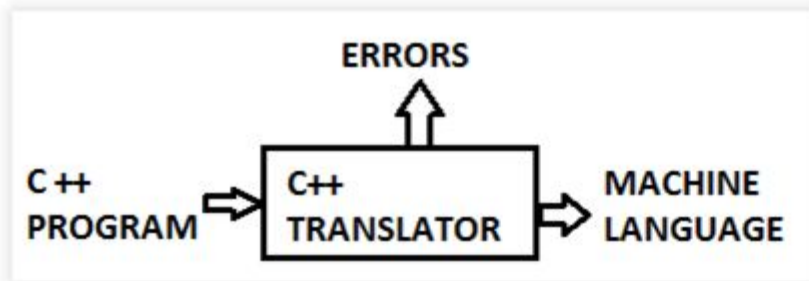
Specification Gap : It is the semantic gap between the two specifications of the same task.

Execution Gap : The gap between the semantics of programs written in the different programming languages.

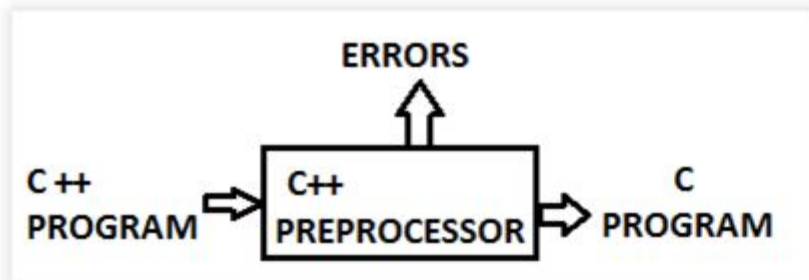
Language Processors : It is a software which bridges a specification or execution gap. The input program of a language processor is a source program and the output program is the target program.

TYPES OF LANGUAGE PROCESSOR

1. Language Translators:- Bridges an execution gap to the machine language of a computer system.



1. Assemblers : A language translator whose source language is assembly language.
2. Compiler : A language translator whose source language is a high level language.
2. Detranslator : Converts machine language to assembly level language. Bridges the same execution gap as in language translator, but in reverse direction.
3. Preprocessor : Which bridges an execution gap and is not a language translator.
4. Language Migrator : bridges the specification gap between two PLs



Preprocessor

1. Remove comments
2. File inclusion
3. Macro expansion

5. Interpreter : It is a language processor which bridges an execution gap without generating a machine language program that means the execution gap vanishes totally.

Language Processing Activities

14/07/20

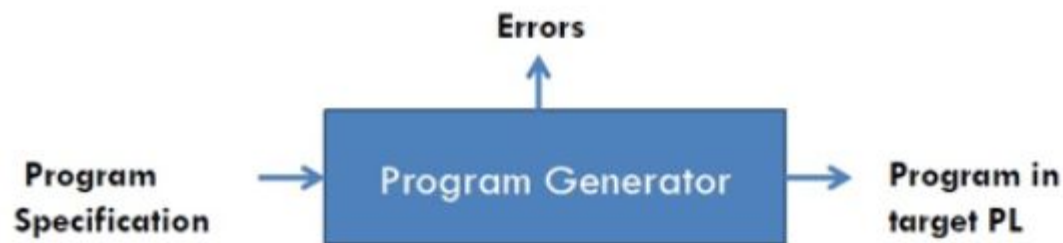
□ Fundamental activities divided into those that bridge the specification gap and execution gap.

- Program generation activities
- Program execution activities

□ *Program Generation:*

- A program generation activity aims at automatic generation of a program.
- A source language is a specification language of an application domain and the target language is procedure oriented PL.

- Program generator introduces a new domain between the application and PL domain, call this the program generator domain.
- Specification gap is now between Application domain and program generation domain, reduction in the specification gap increases the reliability of the generated program (simple to write program).
- A program generator is software that enables an individual to create a program with less efforts and prog. Knowledge.



or



```
Employee name : char : start (line=2,position=25)
               end (line=2,position=80)
Married       : char : start (line=10,position=25)
               end (line=10,position=27)
               value ('yes','No') default ('Yes')
Age           : numeric : start (line=12,position=25)
               end (line=12,position=26)
```

Employee Name	<input type="text"/>
Address	<input type="text"/>
	<input type="text"/>
	<input type="text"/>
Married	<input type="button" value="Yes"/>
Age	<input type="text"/>

□ *Program Execution:*

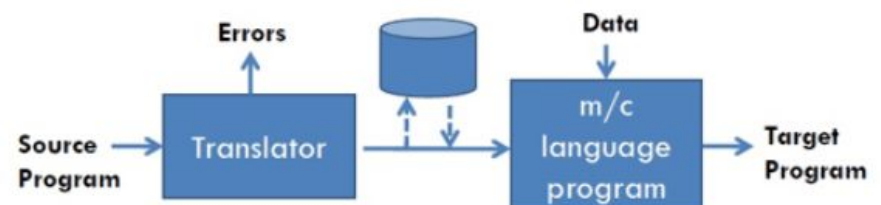
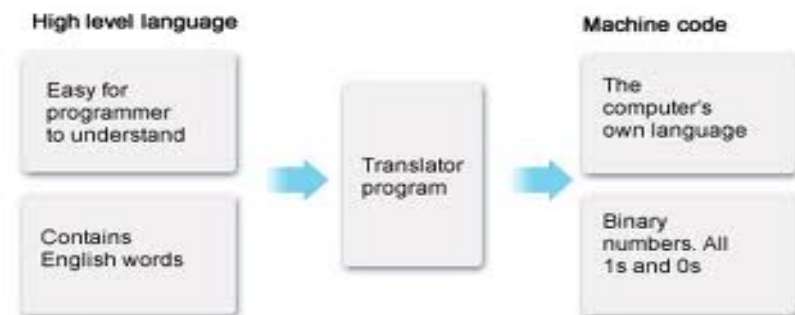
- There are 2 popular models of program execution

1. Program Translation

2. Program Interpretation

- **Program Translation**: The program translation model bridges the execution gap by translating a source program into program in the machine or assembly language of the computer system, called target program.
- **Characteristics of the 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 can be loaded and executed when desired.
 - A program must be **retrans**

■ Program Translation



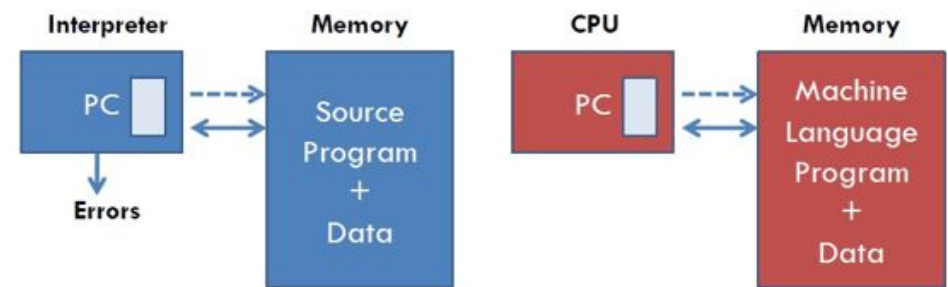
- ❑ **Program Interpretation:** During interpretation interpreter takes source program statement, determines its meaning and performs actions which implement it.

The function of an interpreter is same as the execution of machine language program by CPU.

- In the program interpretation, the CPU use a *program counter*(PC) to note the address of the next instruction to be executed. This instruction is subjected to the ***Instruction execution cycle*** consisting following step:

1. Fetch the instruction
2. Decode the instruction to determine
3. Execute the instruction.

◆ Program Interpretation



Interpretation

Follow interpretation
cycle

Program execution

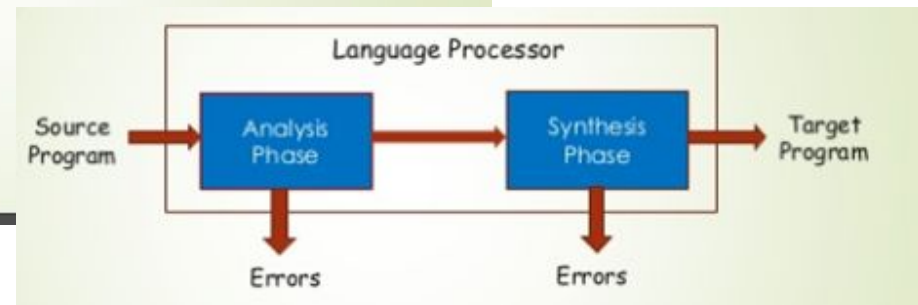
Follow Instruction
execution cycle

▣ Comparison

- In translator whole program is translated into target and if modified the source program, whole source program is translated irrespective to size of modification.
- That not the in case of interpreter, interpretation is slower than execution of m/c language program.

Fundamentals of Language Processing

- Language Processing = Analysis of SP + Synthesis of TP
- Collection of Language Processing components engaged in analysis a source program as the analysis phase and components engaged in synthesizing a target program constitute the synthesis phase.



- Language Processing= Analysis of Source program+ Synthesis of Target program



Analysis Phase

- The specification consists of three components:
 1. Lexical rules which govern the formation of valid lexical units in the source language. (such as operators, identifiers and constants)
 2. Syntax rules which govern the formation of valid statements in the source language.
 3. Semantic rules which associate meaning with valid statements of the language.

Analysis Phase

- Consider the following example:

`percent_profit = (profit * 100) / cost_price`

- Lexical units** identifies `=`, `*` and `/` operators, `100` as constant, and the remaining strings as identifiers.
- Syntax analysis** identifies the statement as an 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

- The synthesis phase is concerned with the construction of target language statements which have the same meaning as a source statement.

- It performs two main activities:

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

Data declaration
statements and
imperative statements

Synthesis of target program

MOVER AREG, PROFIT

MULT AREG, HUNDRED

$\text{percent_profit} = (\text{profit} * 100) / \text{cost_price}$

DIV AREG, COST_PRICE

MOVEM AREG, PERCENT_PROFIT

.....

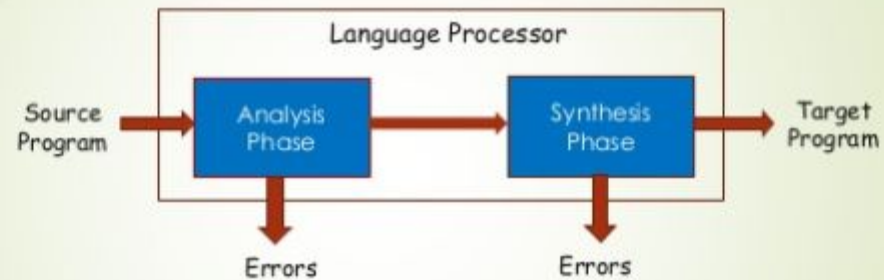
PERCENT_PROFIT DW 1

PROFIT DW 1

COST_PRICE DW 1

HUNDRED DC '100'

Phases & Passes of LP



- Analysis of source statements can not be immediately followed by synthesis of equivalent target statements due to following reasons:
 1. Forward References
 2. Issues concerning memory requirements and organization of a LP

PASS IN LANGUAGE PROCESSOR

Need of Pass

1. **Forward Reference:** The program entity having reference to the entity which is precedes its definition in the program. i.e Declaration Later, Reference First.

Eg: Percent_Profit= (Profit * 100)/cost_price;

```
-----  
-----  
long Profit;
```

2. Issue concerning **memory requirement** and **organization** of a language processor.

PROBLEM

Analysis of Source Statement is followed by synthesis of equivalent target statement.

SOLUTION

Multi-pass model of language processor

A language processor pass is a processing of every statement in a source program, or its equivalent representation, to perform a language processing function/s.

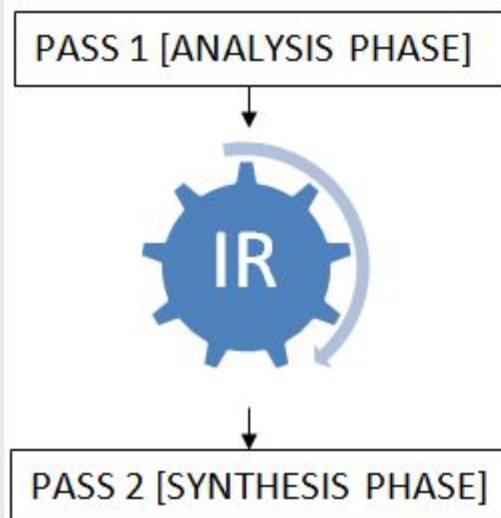
- Pass1 : Performs analysis of source program and note relevant information.
- Pass2 : Performs synthesis of target program.

- Example : $\text{Percent_Profit} = (\text{Profit} * 100) / \text{cost_price};$
 - Pass 1: Performs notification of type of "profit".
 - Pass 2: Using information of Pass-1 code generation is performed.

Problems with Multi-Pass Model

- Such processors perform certain operations more than once., Example
 - Pass 1 : Analysis of SP
 - Pass 2 : Analysis of SP + Synthesis of TP
- **SOLUTION** : Intermediate Representation

INTERMEDIATE REPRESENTATION (IR)



- An IR is a representation of a source program which reflects the effect of some, but not all, analysis & synthesis task performed during language processing.
- Benefit of IR 1. Memory Requirement Reduced, 2. Now; No need of co-existence of front end & back end in memory.
- Desired Property of IR :
 1. Ease of Use: IR should be easy to construct & analysis.
 2. Processing Efficiency: Efficient algorithm must exist for constructing & analyzing IR.
 3. Memory Efficiency: IR must be compact.

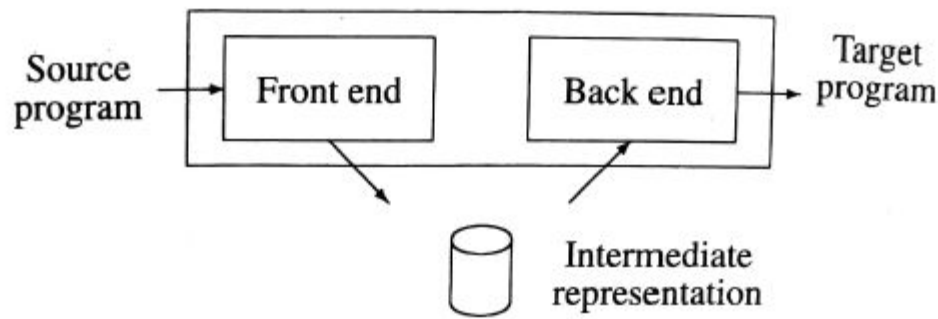


Figure 2.13 A two-pass schematic for language processing

Figure 2.13 depicts the classic two-pass schematic of a language processor. The first pass performs analysis of the source program and reflects its results in the intermediate representation. The second pass reads and analyzes the intermediate representation to perform synthesis of the target program. The first pass is concerned exclusively with source language issues. Hence it is called the *front end* of the language processor. The second pass is concerned with synthesis of the target program; it is called the *back end* of the language processor. The front and back ends of a language processor need not coexist in memory. This feature can be used to reduce the

memory requirements of a language processor as follows: The code for the front end is first loaded in memory. It analyzes the source program, generates the intermediate representation, and writes it to a disk. Now the code of the front end is removed from memory and the code of the back end is loaded. It reads the intermediate representation and synthesizes the target program.