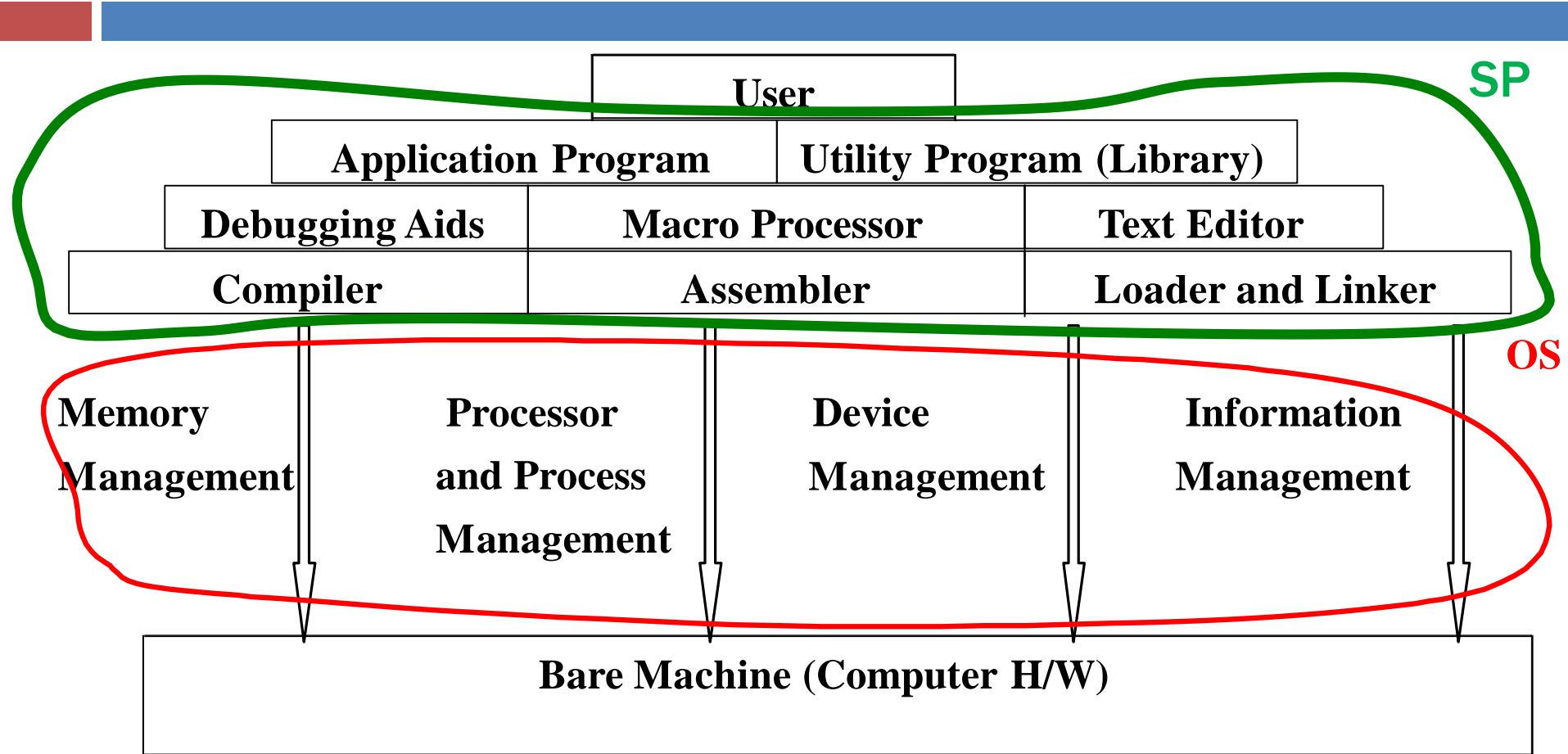


Modern Computer System



- CPU, Memory, I/O devices, Networking Support etc.
- Machine language (1's and 0's)
- User Unable to instruct using M/C language
- System Software
- Bridge gap between User and Computer System
- Like Software layer

The scope of the system program



What is System Software

- What is Computer System?
- School or College student
- User of Application package like administrative stuff
- Programmer
- Abstracts view
- Computer System is combination of
 - Computer H/W
 - Computer S/W

Abstract View

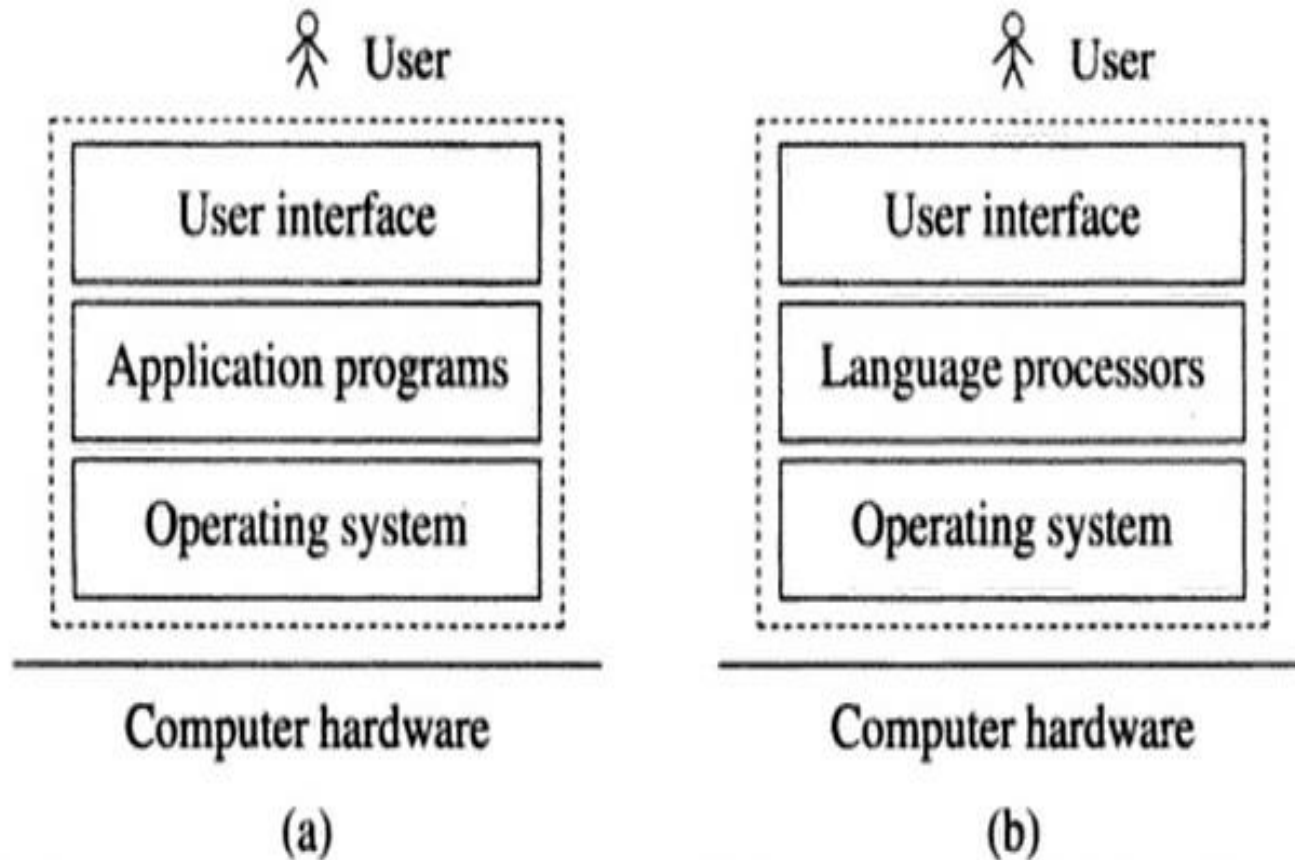



Figure 1.1 Abstract views of a computer system: (a) View of a student, (b) View of the programmer

- 
- **System Program**
 - **Environments**
 - **Editing, Compiling and arranging for execution**
 - **Linking, Protecting**
 - **Used to describe the collection of techniques used in design of system program.**

Computer Languages

- Machine languages
- Low-level languages
 - Assembly language: symbolic representation of machine instructions.
 - Assembler: a compiler which translates from an assembly language to a machine language.
- High-level languages
 - Java, Python, C, ASP, HTML, ...

High level language



Preprocessor



Compiler



Assembler



Loader/Linker



Executable File

Pure High level language

Assembly Code

Machine Code (relocatable)

Course Contents



- **Unit 1: INTRODUCTION TO SYSTEMS PROGRAMMING AND ASSEMBLERS**
- **Unit 2: MACROPROCESSORS, LOADERS AND LINKERS**
- **Unit 3: INTRODUCTION TO COMPILERS**
- **Unit 4: PARSERS**
- **Unit 5: SEMANTIC ANALYSIS AND STORAGE ALLOCATION**
- **Unit 6: CODE GENERATION AND OPTIMIZATION**

Recall



- **Course Structure and Contents**
- **Introduction**

Unit 1: INTRODUCTION TO SYSTEMS PROGRAMMING AND ASSEMBLERS

Outline

■ Introduction:

- ❑ Need of System Software
- ❑ Components of System Software
- ❑ Language Processing Activities
- ❑ Fundamentals of Language Processing

■ Assemblers :

- ❑ Elements of Assembly Language Programming
- ❑ A simple Assembly Scheme
- ❑ Pass structure of Assemblers
- ❑ Design of Two Pass Assembler
- ❑ Single pass assembler

System Software

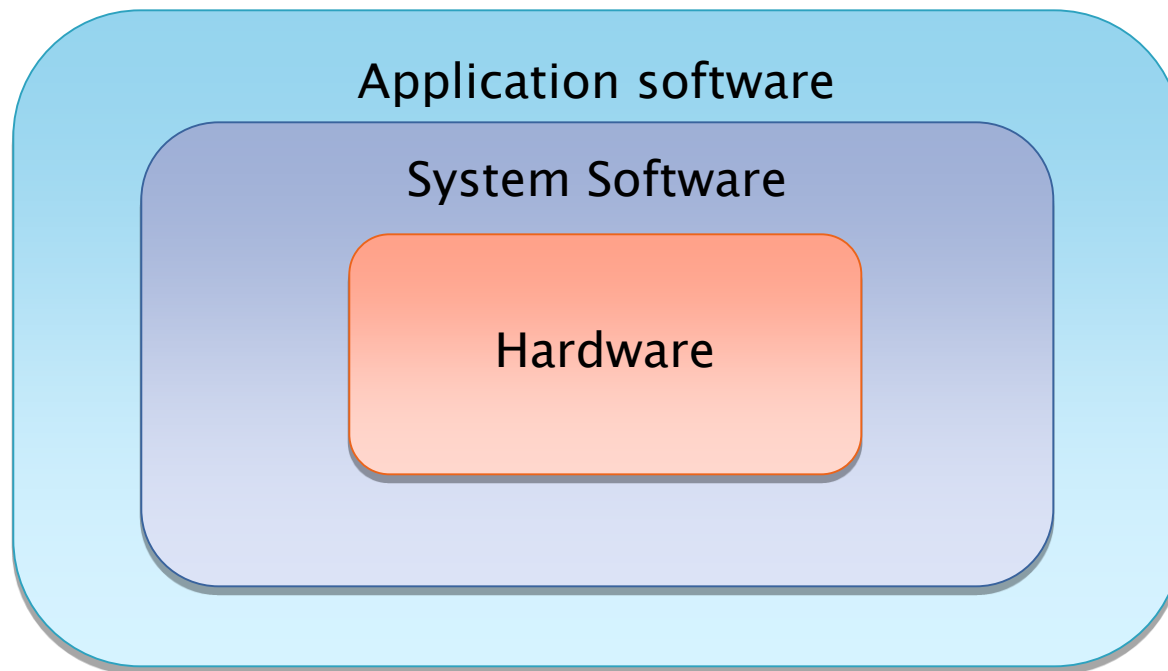
- ▶ **Computer software**, or simply **software**, refers to the non-tangible components of computers, known as computer programs. The term is used to contrast with computer hardware, which denotes the physical tangible components of computers.

Software classification

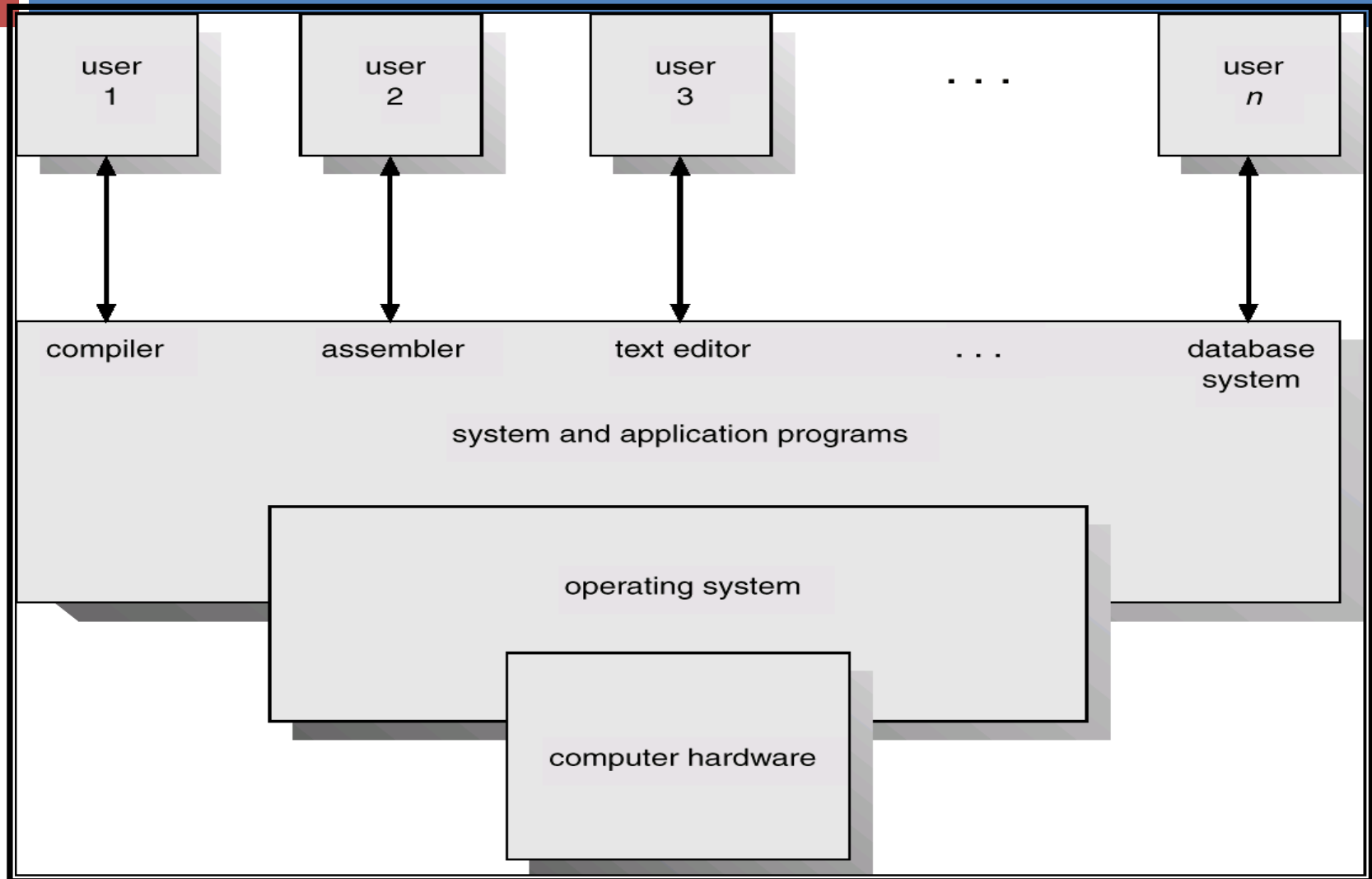
- ▶ Software can be classified into
 - System software:
 - **System software** (or **systems software**) is [computer software](#) designed to operate and control the [computer hardware](#) and to provide a platform for running [application software](#).
 - System software is collection of software program that perform a variety of functions like IO management, storage management, generation and execution of programs etc.
 - Operating Systems
 - Compiler / Assembler (utility software's)
 - Device Drivers
 - Application software:
 - Application software is kind of software which is designed for fulfillment specialized user requirement.
 - MS Office
 - Adobe Photoshop
 - Audio/Media Player

Cont.

- ▶ The system software work as middleware between application software and hardware.



Abstract View of System Components



Introduction

- Collection of system programs
- facilitate execution of programs and use of resources in computer system
- System Software consists of a variety of programs that support the **operation of a computer**.
- The software makes it possible for the users to focus on an application or other problem to be solved, **without needing to know the details of how the machine works internally**.

System Software and Machine Architecture

- One characteristic in which most system software differs from application software is **machine dependency**.
- System programs are **intended to support the operation and use of the computer itself**, rather than any particular application.
- e.g. of system software
 - Text editor, assembler, compiler, loader or linker, debugger, macro processors, operating system, database management systems, software engineering tools, ...

1.2 Goal of System Software



- **User Convenience**
 - **Convenient Methods of Use**
- **Efficient use**
 - **Efficient use of Computer Resources**
- **Non-interference**
 - **Prevent Interference**

System Softwares/Language Processors

■ Text editor

- ❑ To create and modify the program

■ Compiler and assembler

- ❑ You translated these programs into machine language

■ Loader or linker

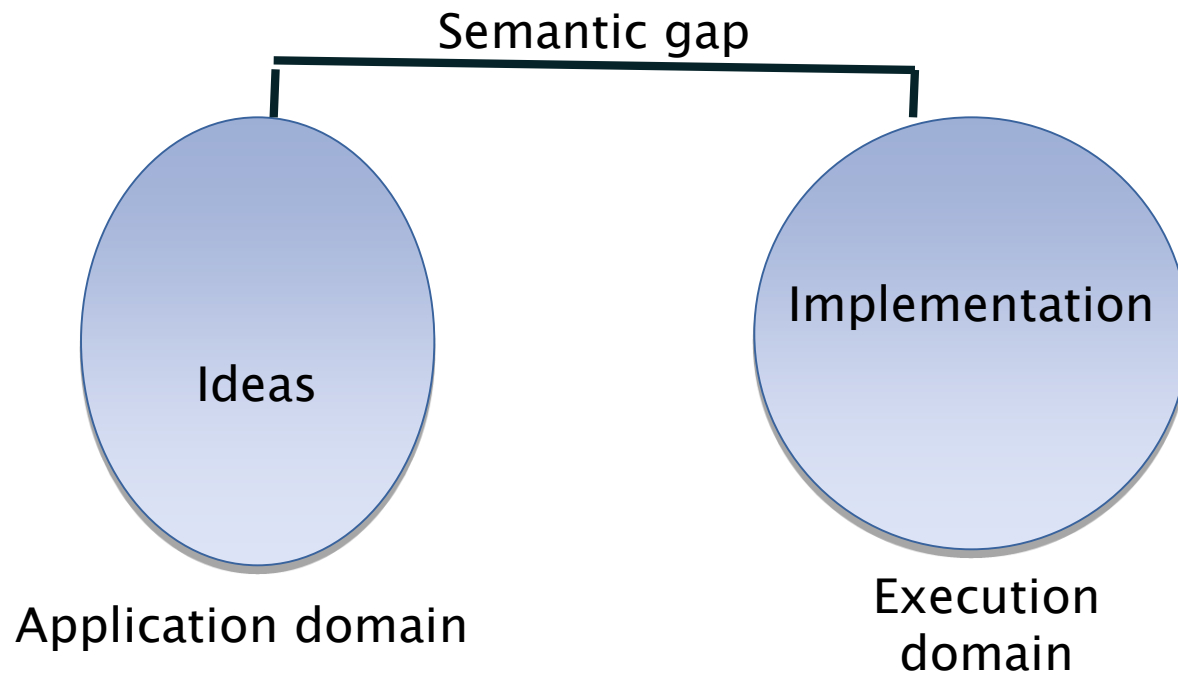
- ❑ The resulting machine program was loaded into memory and prepared for execution

■ Debugger

- ❑ To help detect errors in the program

System Software

- ▶ **Language processors (Why?)**
 - **Language processing activities arise due to the differences between the manner in which a software designer describes the ideas concerning the behavior of software and the manner in which these ideas are implemented in computer system.**
 - **The designer expresses the ideas in terms related to the application domain of the software.**
 - **To implement these ideas, their description has to be interpreted in terms related to the execution domain.**



- ▶ The term semantics to represent the rules of meaning of a domain, and the term semantic gap to represent difference between the semantics of two domains.

- Design of Airline Application System
 - Query
 - Book
 - Cancel
- The Description of reservation data and operations form specification of application.
- System S/W help to implement this specification on the computer using machine language.
- System S/W has to implement his specification in the execution domain of computer system.
- Entities in Execution domains are cells in memory and registers in CPU and operations are the instruction of the CPU

Semantic gap

Reservation
Data

Query

Book

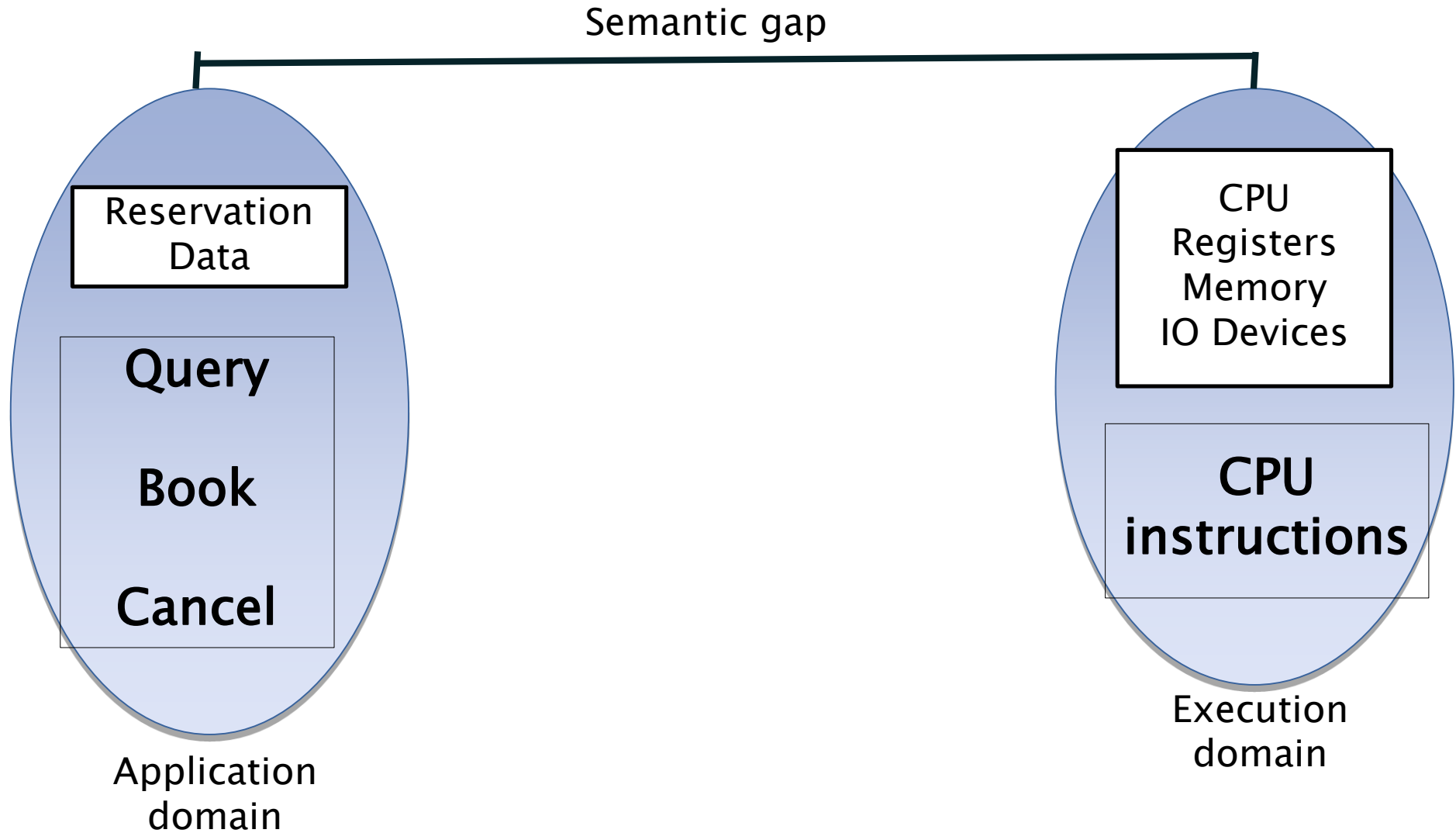
Cancel


Application
domain

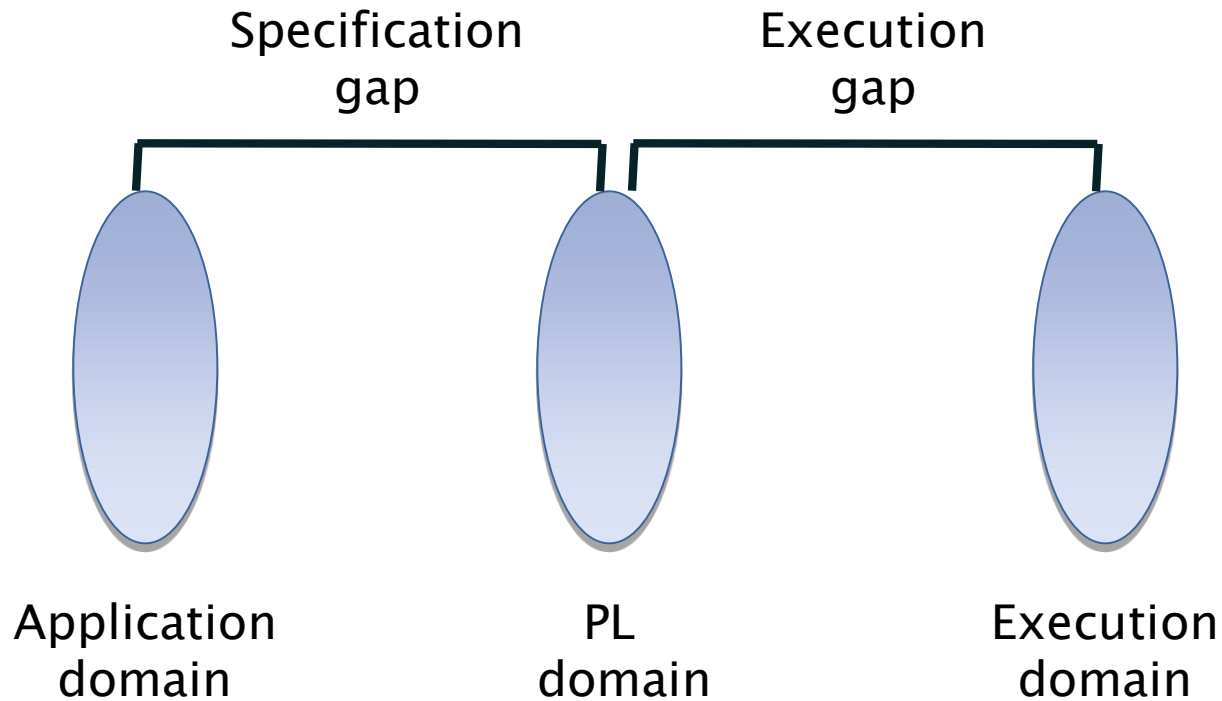
CPU
Registers
Memory
IO Devices

CPU
instructions

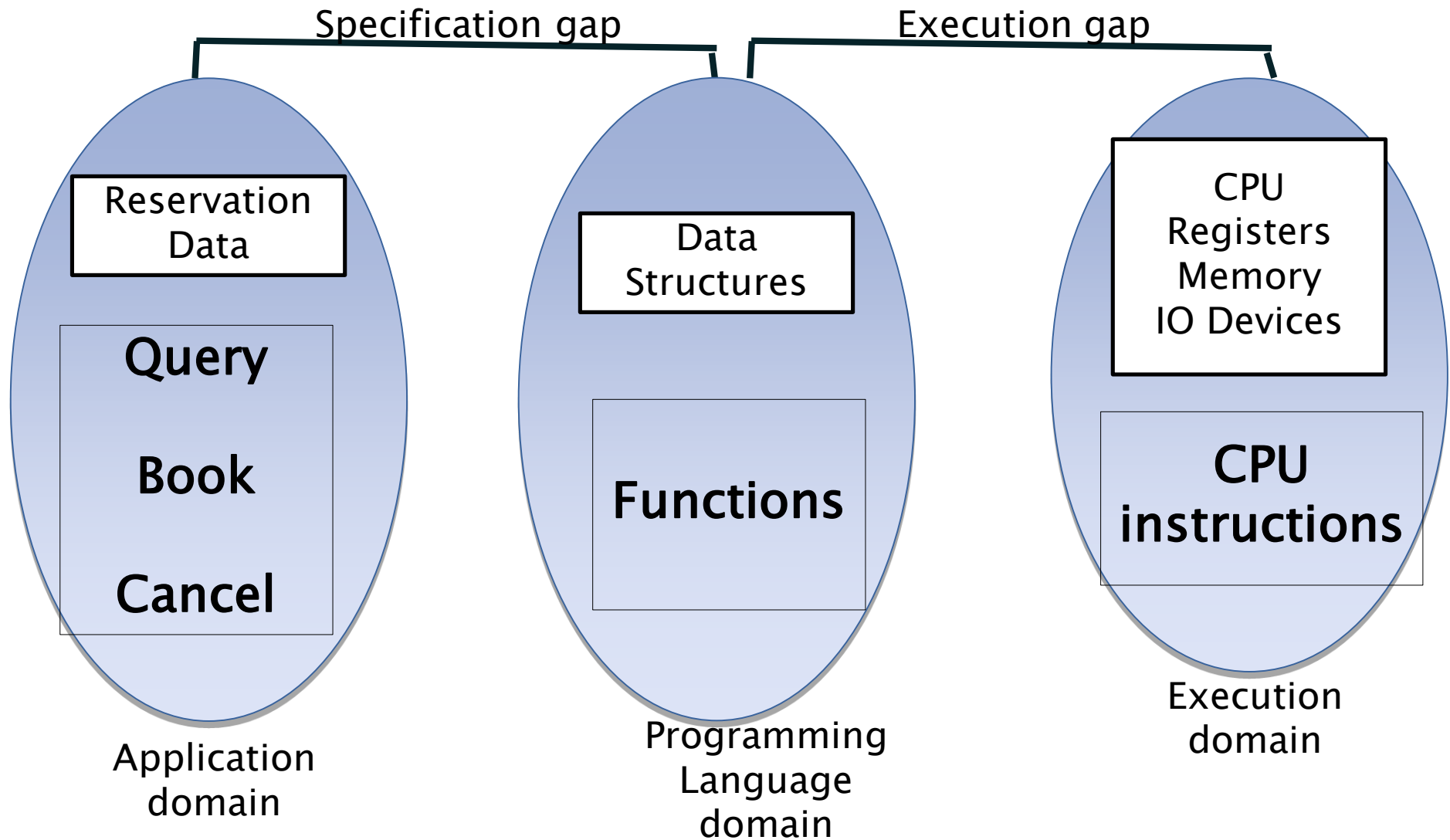
Execution
domain



- 
- ▶ Semantic gap is bridged when specification in one of the domains is converted into specification in the other domain.
 - ▶ these issues are tackled by software engineering thru' use of methodologies and programming languages.
 - ▶ Programming Language
 - ▶ Language Processor



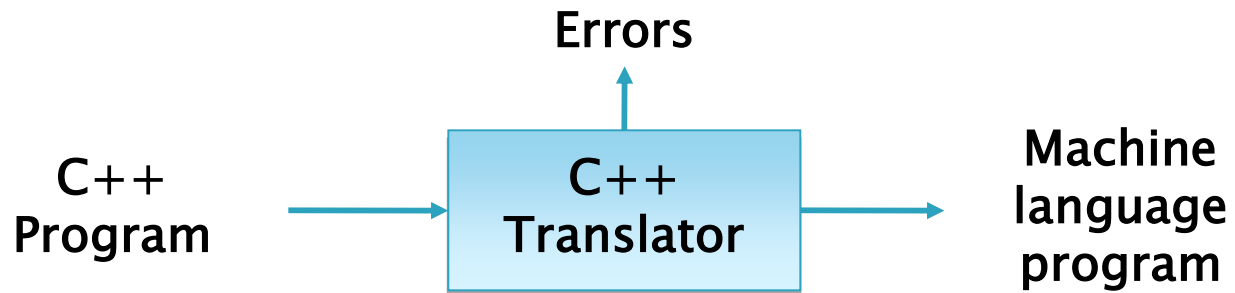
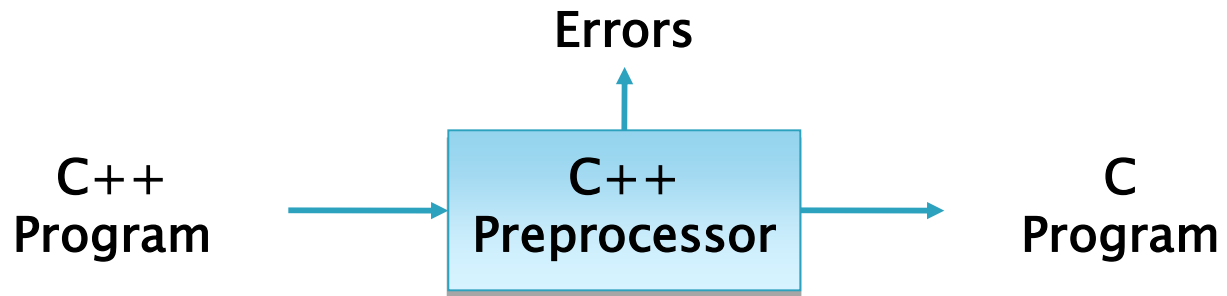
- ▶ s/w development team
- ▶ Programming language processor



Cont.

- ▶ Language processor: A language processor is software which bridge a specification or execution gap.
- ▶ System Program that bridges the gap between how a use describes a computation—call it specification of computation and how computer executes a program.
 - A Language Translator
 - De-translator
 - Preprocessor
 - Language migrator

Example

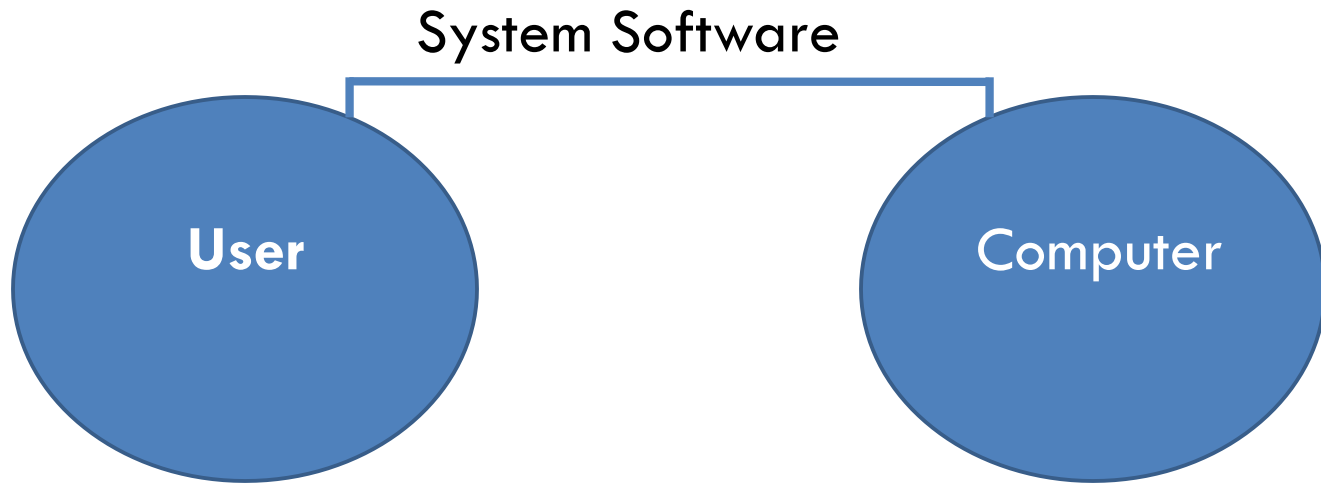


C++ Program converted to C Program by C++ Preprocessor
C++ Program converted to machine language by C++ translator.

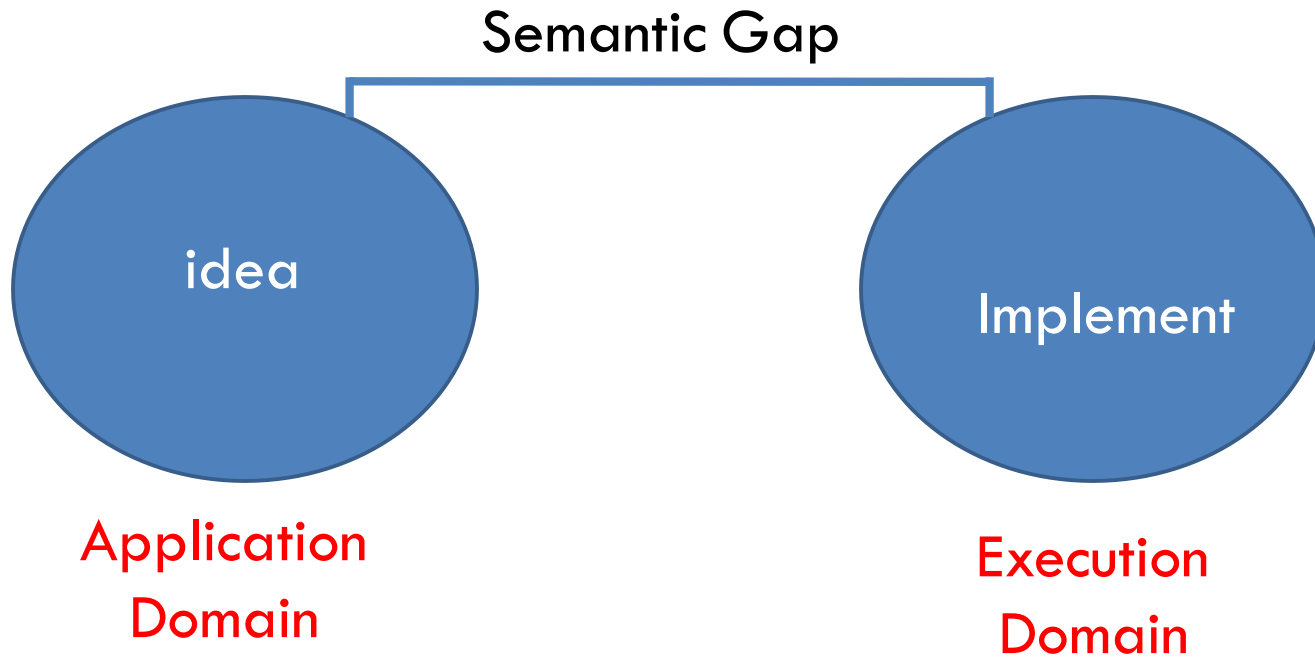
Recall

- **Language Processing activities** arise due to the differences between the manner in which a software designer describes *the ideas concerning the behavior of a software* and the manner in which *these ideas are implemented in a computer system*.
- The designer expresses the ideas in terms related to the *application domain* of the software. To implement these ideas, their description has to be interpreted in terms related to the *execution domain*.

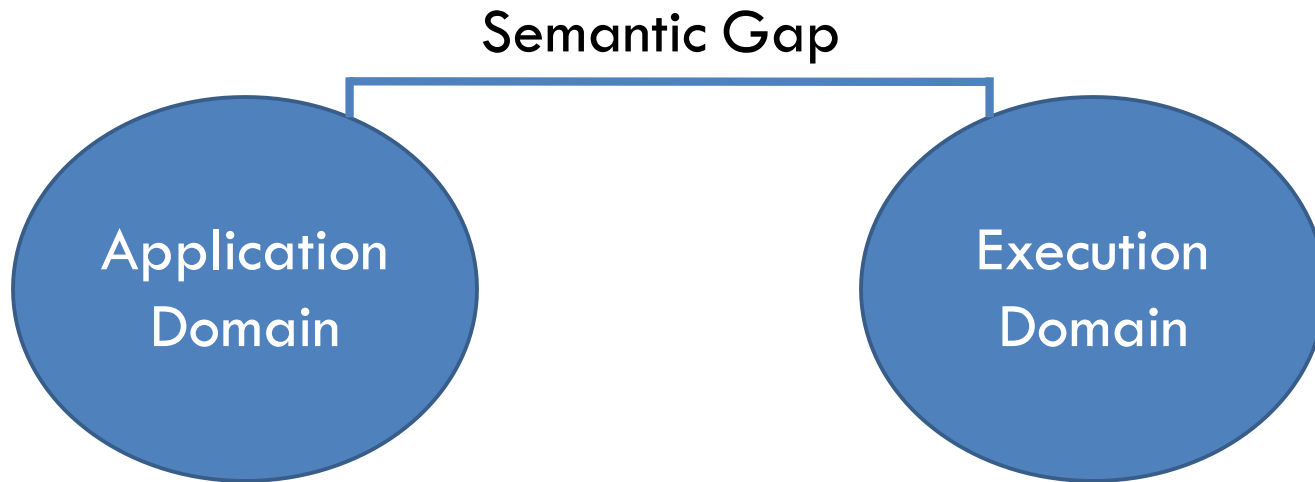
System Programming



Semantic Gap

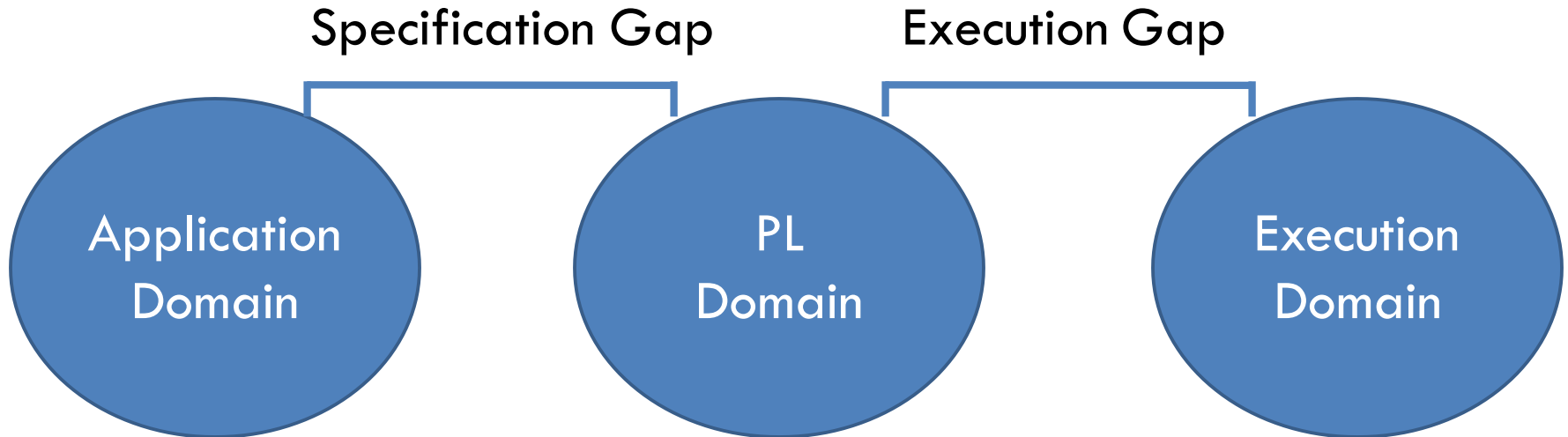


Semantic Gap



- ❑ Semantic Gap has many consequences
 - ▣ Large development time
 - ▣ Large development effort
 - ▣ Poor quality of software

Specification and Execution Gaps



- ❑ The software engineering steps aimed at the use of a PL can be grouped into
 - ▣ Specification, design and coding steps
 - ▣ PL implementation steps

Specification and Execution Gaps

□ Specification Gap

- ▣ It is the semantic gap between two specifications of the same task.

□ Execution Gap

- ▣ It is the gap between the semantics of programs (that perform the same task) written in different programming languages.

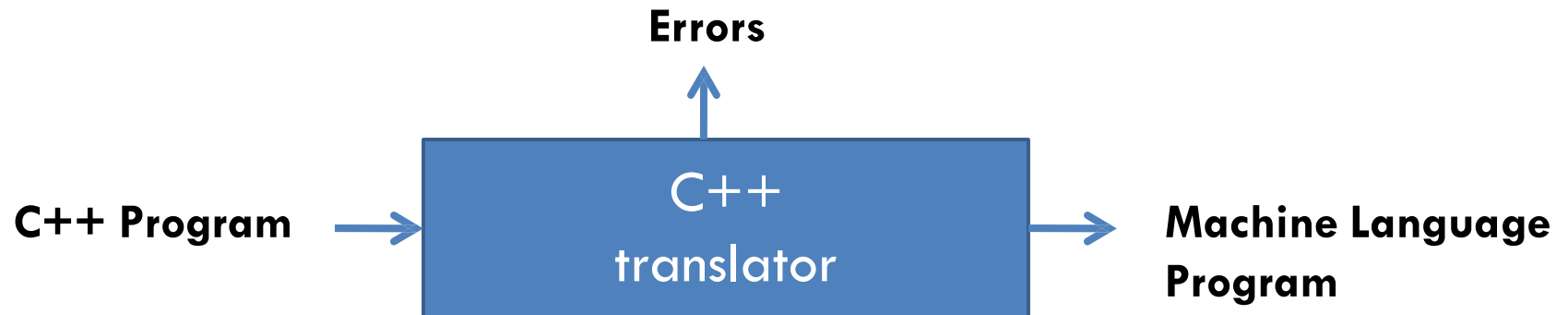
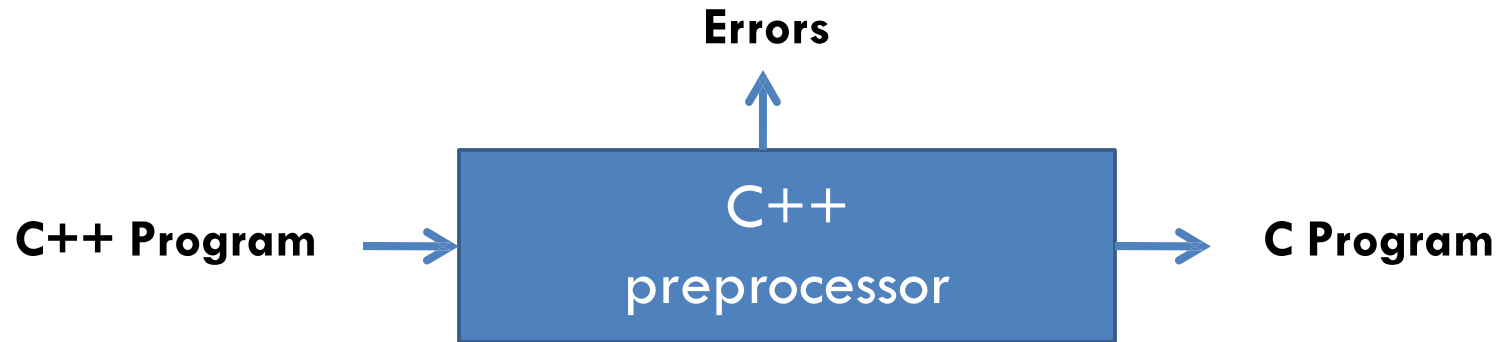
Language Processors

- “A *language processor* is a software which bridges a *specification or execution gap*”.
- The program form input to a language processor as the *source program* and to its output as the *target program*.
- The languages in which these programs are written are called *source language* and *target language*, respectively.

Types of Language Processors

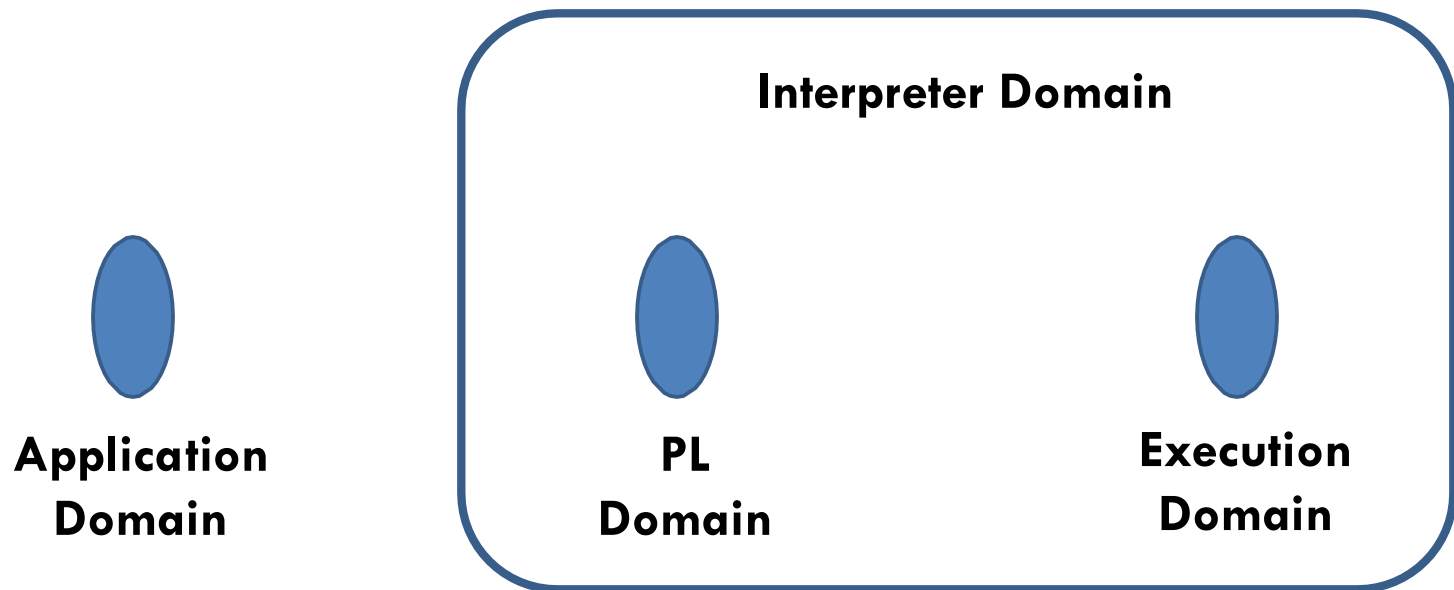
- A **language translator** bridges an execution gap to the machine language (or assembly language) of a computer system. E.g. Assembler, Compiler.
- A **detranslator** bridges the same execution gap as the language translator, but in the reverse direction.
- A **preprocessor** is a language processor which bridges an execution gap but is not a language translator.
- A **language migrator** bridges the specification gap between two PLs.

Language Processors - Examples



Interpreters

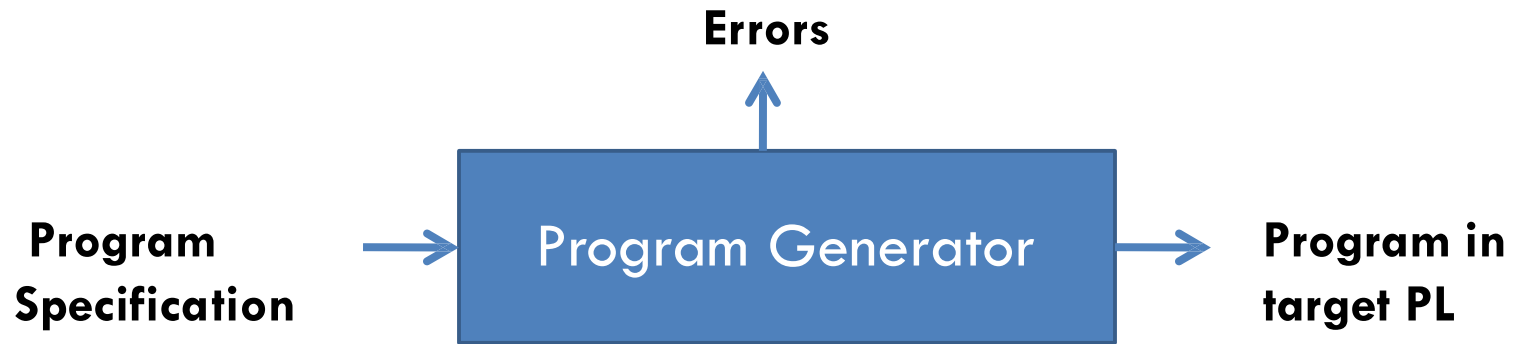
- An **interpreter** is a language processor which bridges an execution gap without generating a machine language program.
- An interpreter is a language translator according to classification.



Language Processing Activities

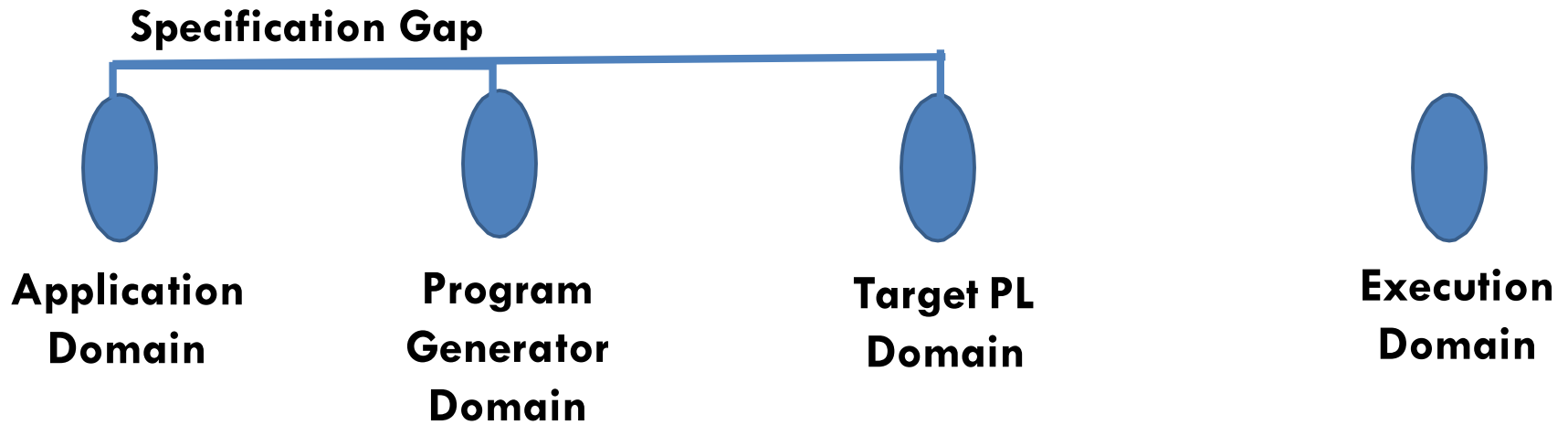
- Program Generation Activities
- Program Execution Activities

Program Generation



- The program generator is system program which accepts the specification of a program in some specification language and generates a program in target language that fulfills the specification.
- Program Generator domain

Program Generation



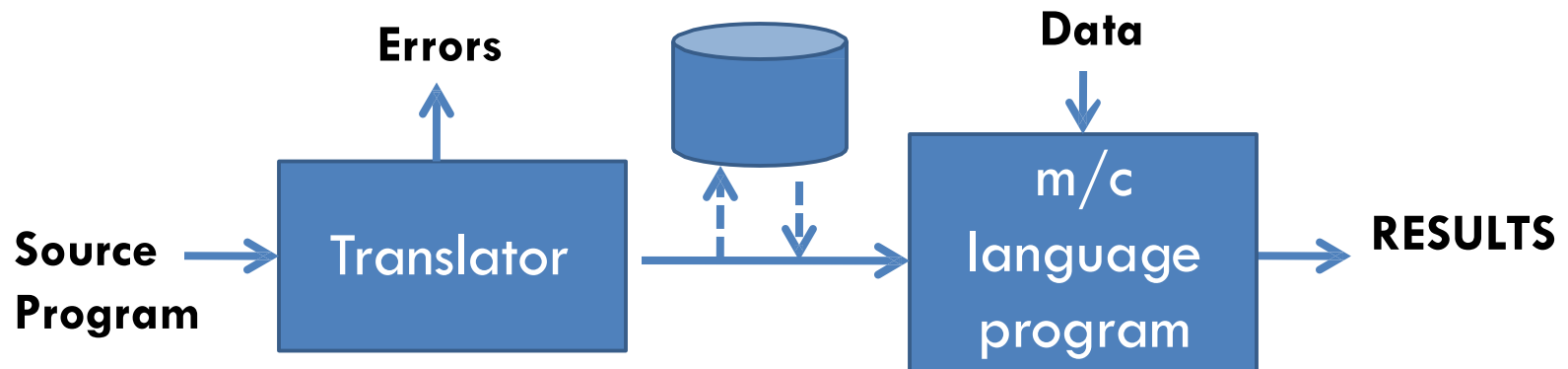
Data Entry Environment: Screen Handling Program

end [line=12,position=26]

QUESTION

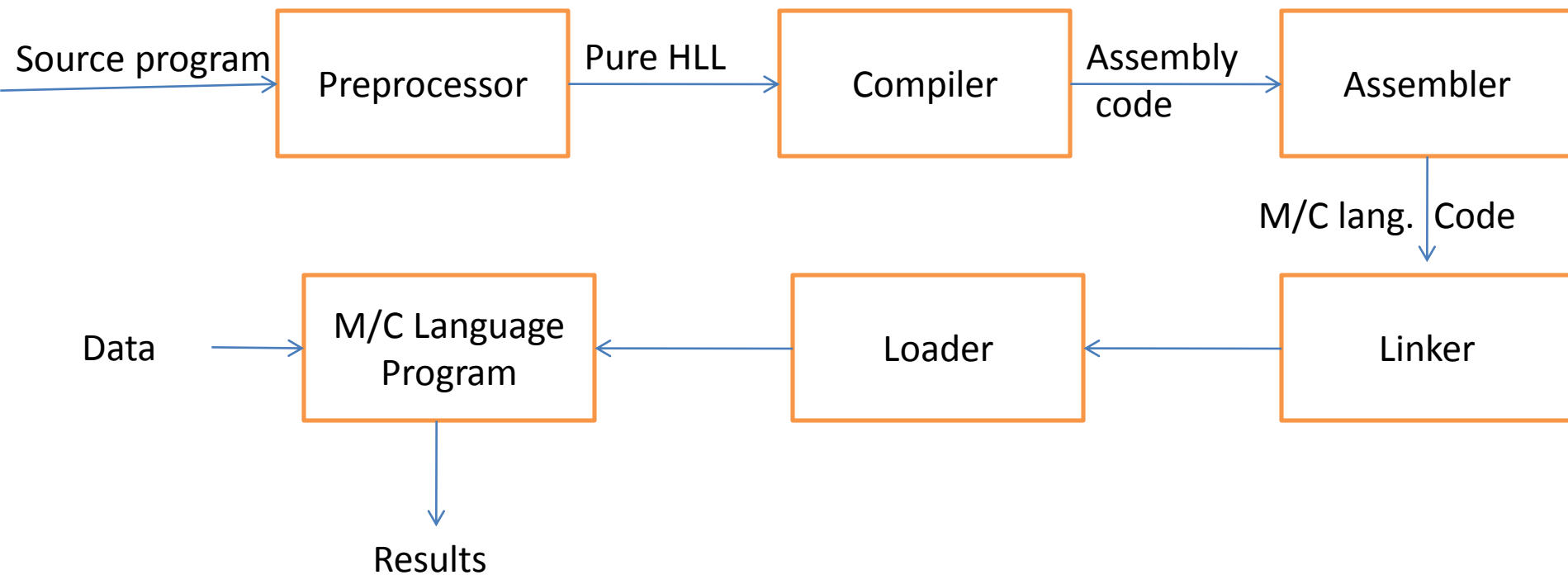
Program Execution

- ❑ Two popular models for program execution are translation and interpretation.
- ❑ **Program translation**



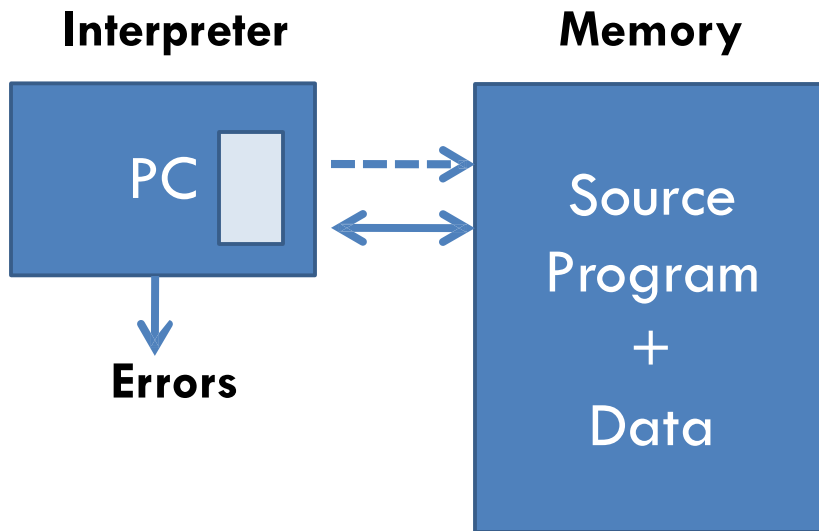
- ❑ A program must be translated before it can be executed.
- ❑ The translated program may be saved in a file. The saved program may be executed repeatedly.
- ❑ A program must be retranslated following modifications.

Practical Arrangement of Language Processor

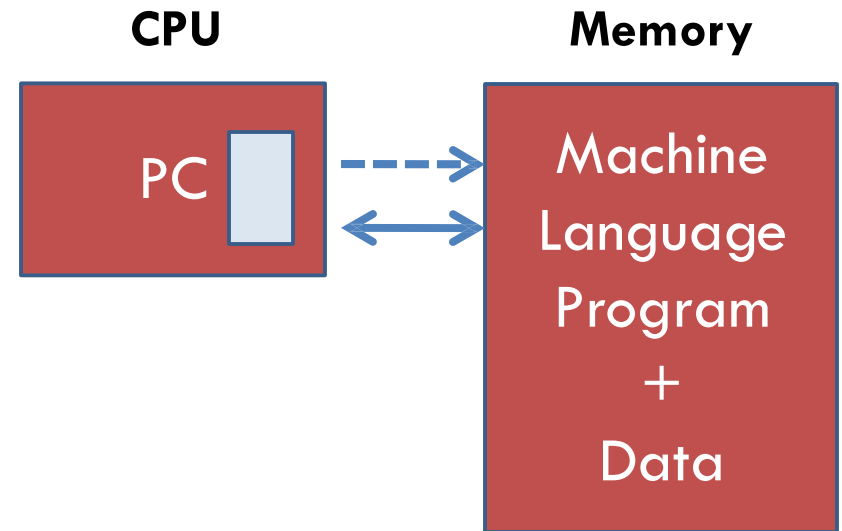


Program Execution

□ Program interpretation



Interpretation

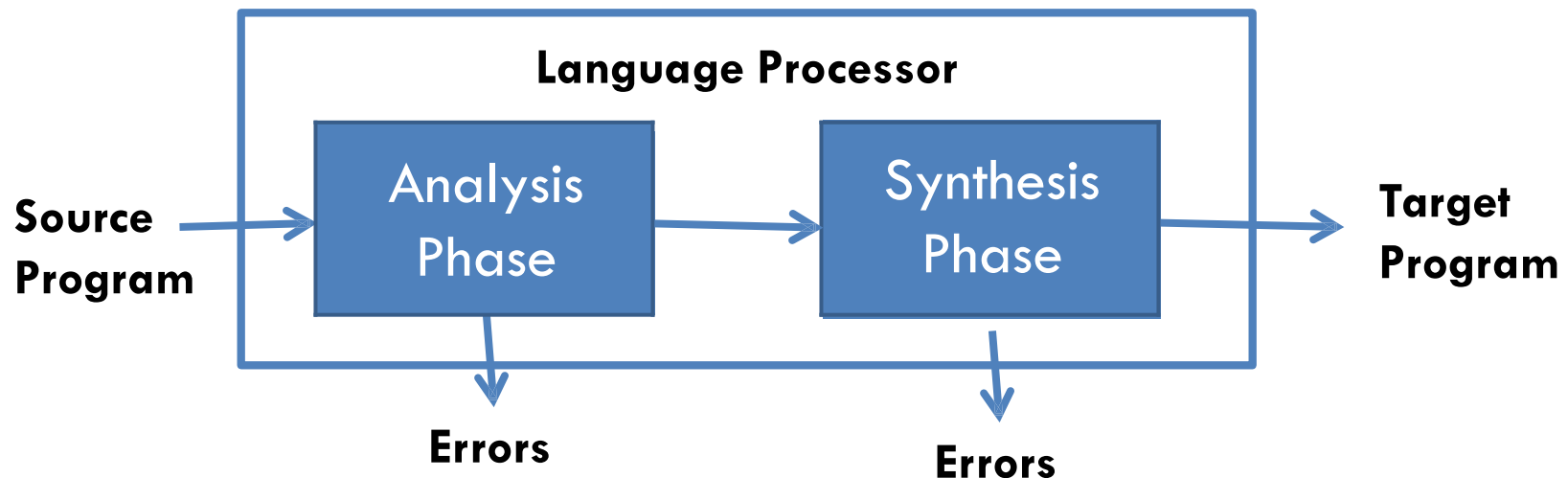


Program execution

Fundamentals of Language Processing

Language Processing = Analysis of SP + Synthesis of TP

Collection of LP components engaged in analysis a source program as the analysis phase and components engaged in synthesizing a target program constitute the synthesis phase.



Analysis Phase

- The specification consists of three components:
 - ▣ **Lexical rules** which govern the formation of valid lexical units in the source language.
 - ▣ **Syntax rules** which govern the formation of valid statements in the source language.
 - ▣ **Semantic rules** which associate meaning with valid statements of the language.

- 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 X 100 / cost_price to percent_profit.

Synthesis Phase

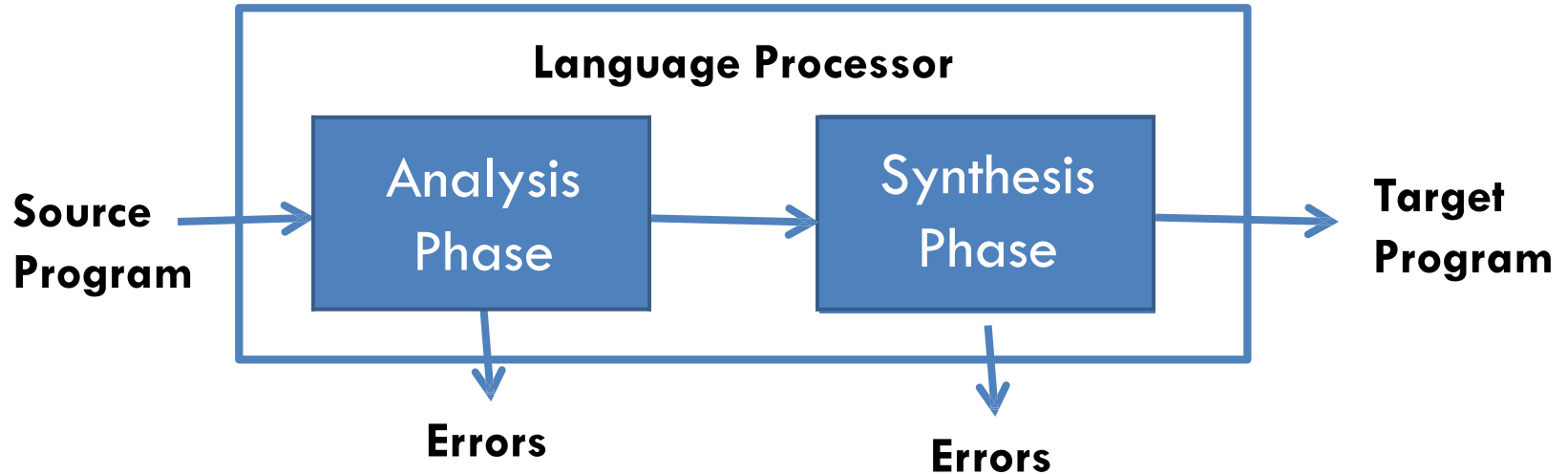
- The synthesis phase is concerned with the construction of target language statements which have the same meaning as a source statement.
- It performs two main activities:
 - ▣ Creation of data structures in the target program (**memory allocation**)
 - ▣ Generation of target code (**code generation**)
- Example

```
MOVER  AREG, PROFIT
MULT   AREG, 100
DIV    AREG, COST_PRICE
MOVEM  AREG, PERCENT_PROFIT
```

...

PERCENT_PROFIT	DW	1
PROFIT	DW	1
COST_PRICE	DW	1

Phases and Passes of LP



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

Forward Reference

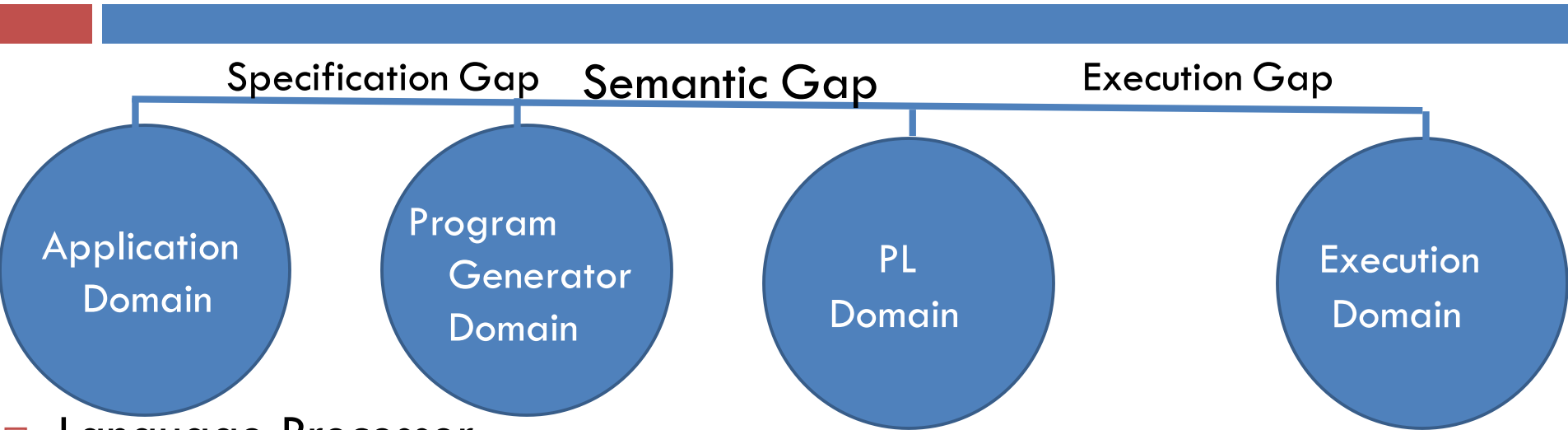
- A forward Reference of a program entity is a reference to the entity in some statements of the program that occurs before the statement containing the definition or declaration of the entity.
- $\text{Percent_profit} = (\text{profit} * 100) / \text{cost_price};$
-
- `float profit;`
- Tackle the problem using Multi-pass organization

Language Processor Pass



- A language Processor pass is the processing of every statement in a source program or in its equivalent representation to perform a language processing function.
- “Pass” is Abstract Noun denote processing done by language processor.
- Pass I:-perform Analysis of the source program and note the deduced information.
- Pass II:-Perform Synthesis of target program.

Recall



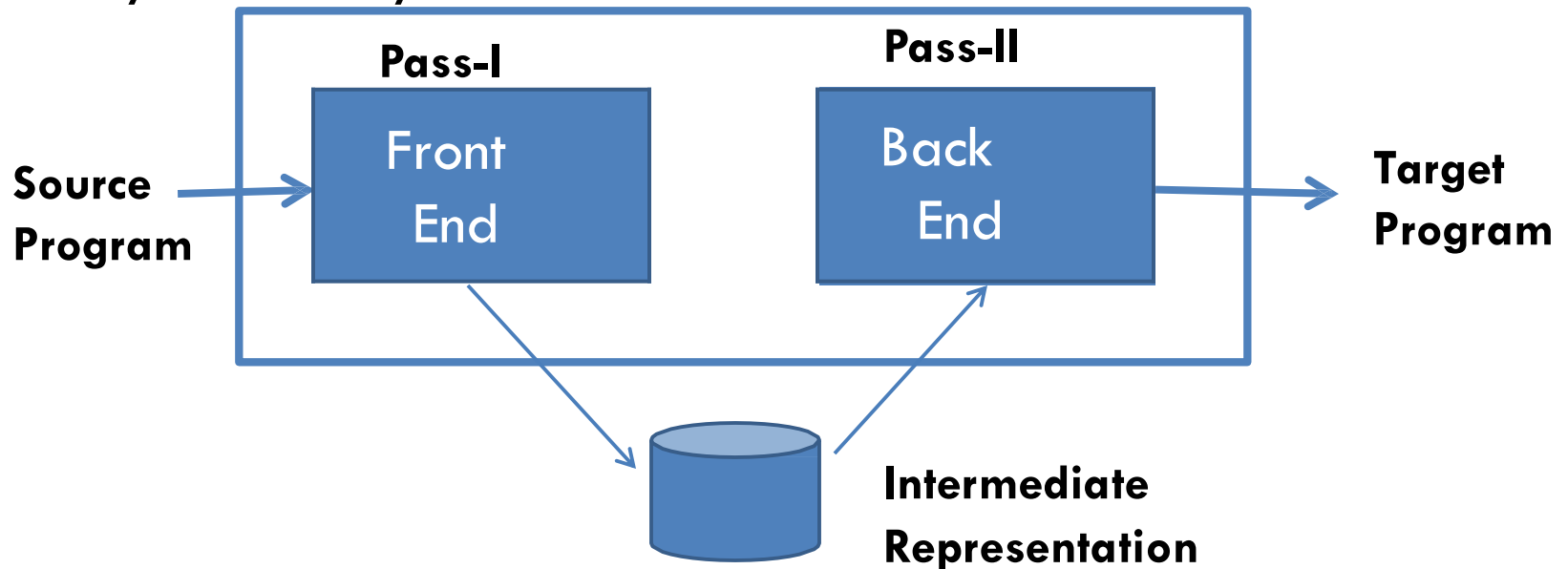
- ❑ Language Processor
- ❑ Language Processing Activities
 - ▣ Program generation Activity
 - ▣ Program Execution Activity
- ❑ Fundamental of Language Processing
 - ▣ Analysis Phase
 - ▣ Synthesis Phase
- ❑ Lexical rules, Syntax rules and Semantic rules
- ❑ Memory allocation and Code Generation


Recall

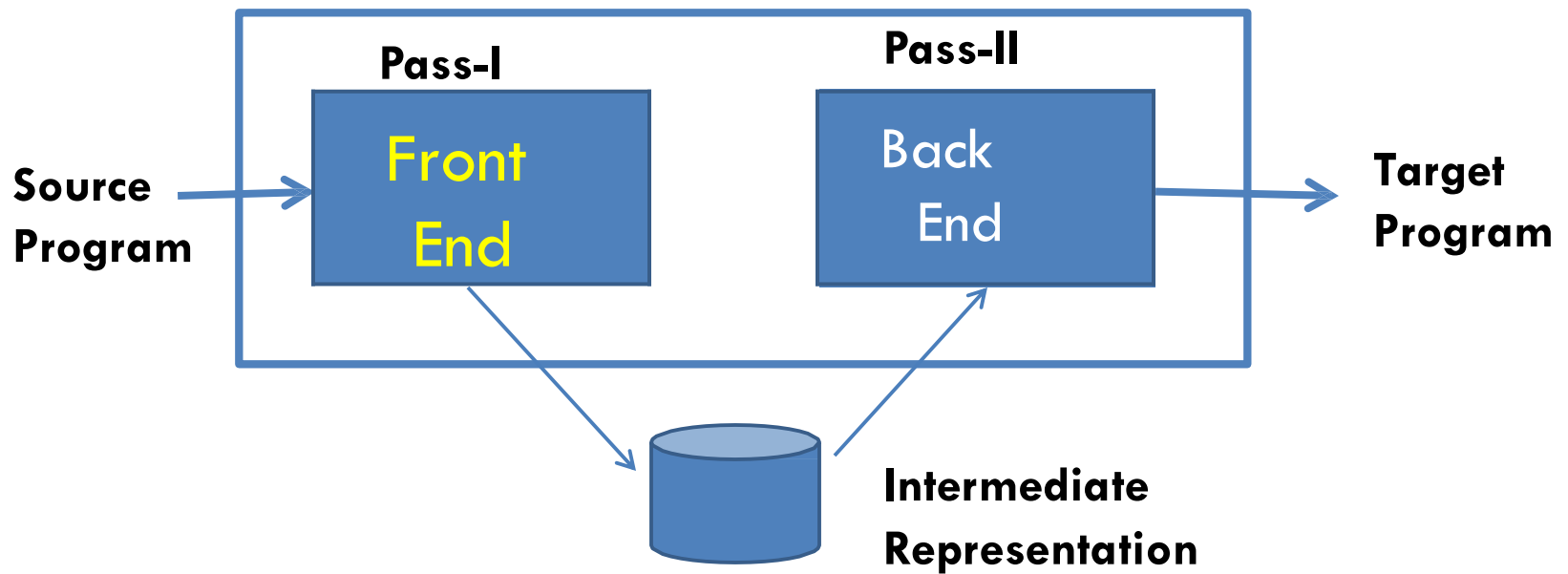
- ❑ If Analysis Phase immediately followed by Synthesis Phase
 - ❑ Forward reference
 - ❑ High Amount Memory Requirements
- ❑ Multi-pass organization (language processor Pass)
- ❑ Pass I- Performs Analysis of Source Program and note relevant information.
- ❑ Pass II-Performs synthesis of target program.
- ❑ Some Language Processor processors perform certain operations more than once.,
- ❑ Example
 - ❑ Pass 1 : Analysis of SP to find type of each variable.
 - ❑ Pass 2 : Analysis of SP to generate target code(Synthesis phase) by using type information noted in pass I.
- ❑ This Duplication and Overhead can be avoided using
- ❑ Intermediate Representation

Intermediate Representation (IR)

- Representation of a source program which reflects the effects of some , but not all, analysis and synthesis functions performed during language processing.
- Ease of use
- Processing Efficiency
- Memory Efficiency



- 
- Pass I perform Analysis of the source program and reflects its result in Intermediate Representation.
 - Pass II reads and analyzes the IR to perform synthesis of target program.
 - Pass I concerned with source language issues hence called front end.
 - Pass II concerned with Synthesis of target program so called back end of language processor.
 - Reduce the memory requirements as follows.
 - Front end firstly loaded into memory, it analyzes the source program, generates intermediate representation and writes it to disk.
 - Now code front end is removed from memory and code of back end is loaded.
 - It reads the IR and synthesis the target program.



Front end



- Performs Lexical ,syntax and semantic analysis of source program.
- Each Kind of Analysis Involves following functions:
 - ✓ Check validity of source statement from the viewpoint of analysis.
 - ✓ Determine 'content' of source statement .
 - ✓ Construct the suitable representation of source statement.
- Use generic form for Representation of content of various stages of language processor. It consists two important Components
 - 1) Tables (The most important Symbol Table which contain information about identifiers used in source program)
 - 2) Intermediate Code