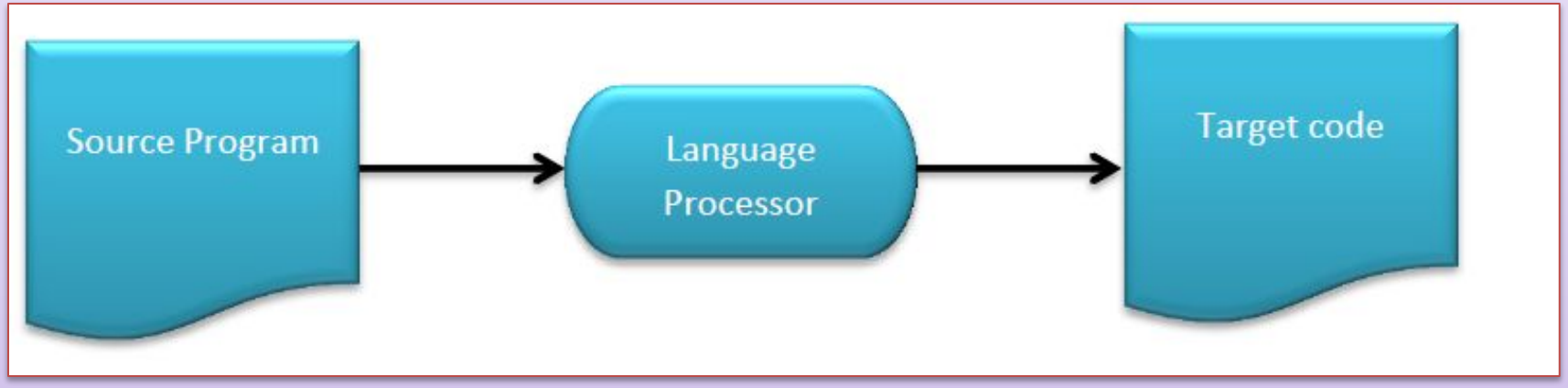
Language Processor

Language Processor

Language processors are programs that can read a program in one language and convert it to an equivalent program in another language.



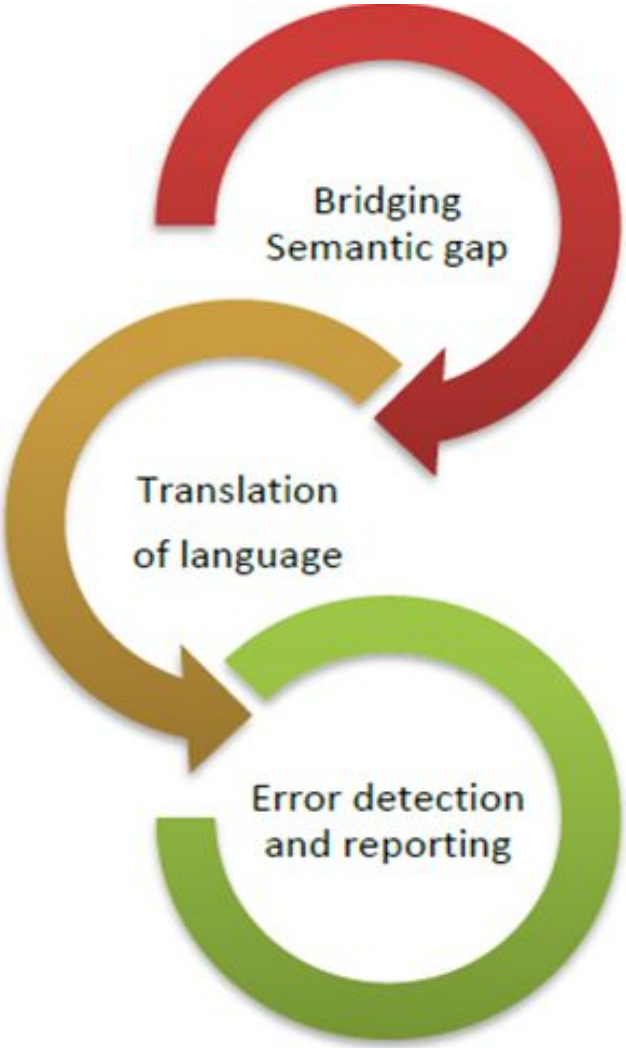
Language Processor

Language Processing activities arise due to the diff between the manner in which a software designer describes the ideas and the manner in which ideas are implemented in a computer system.

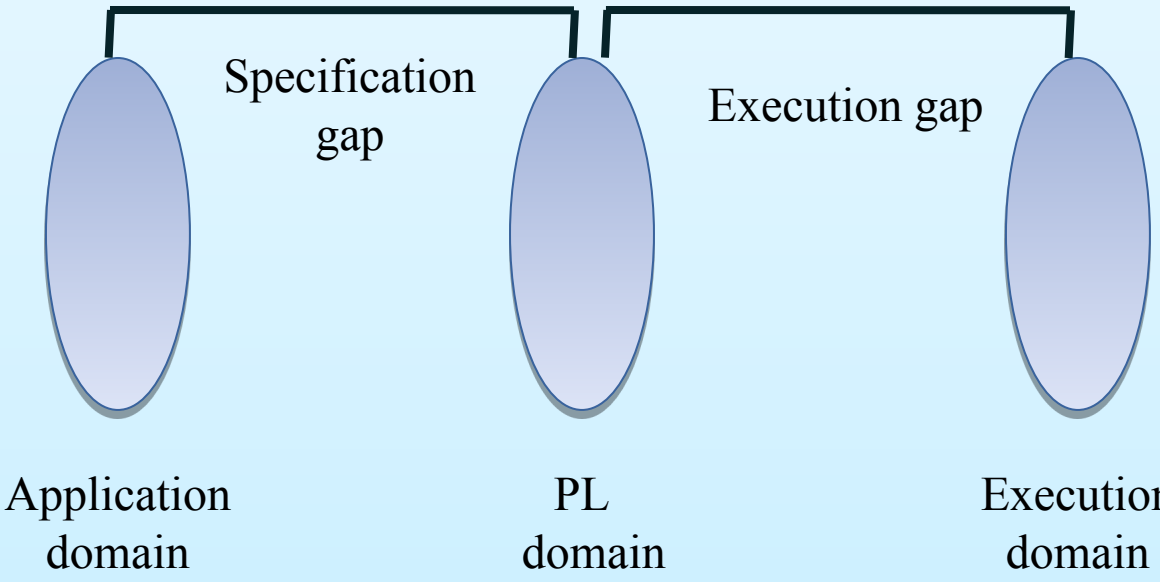
The designer express their ideas in terms of application domain of software domains. To implement these ideas, their description need to be interpreted in terms related to the Execution domain.

Purpose of Language Processor

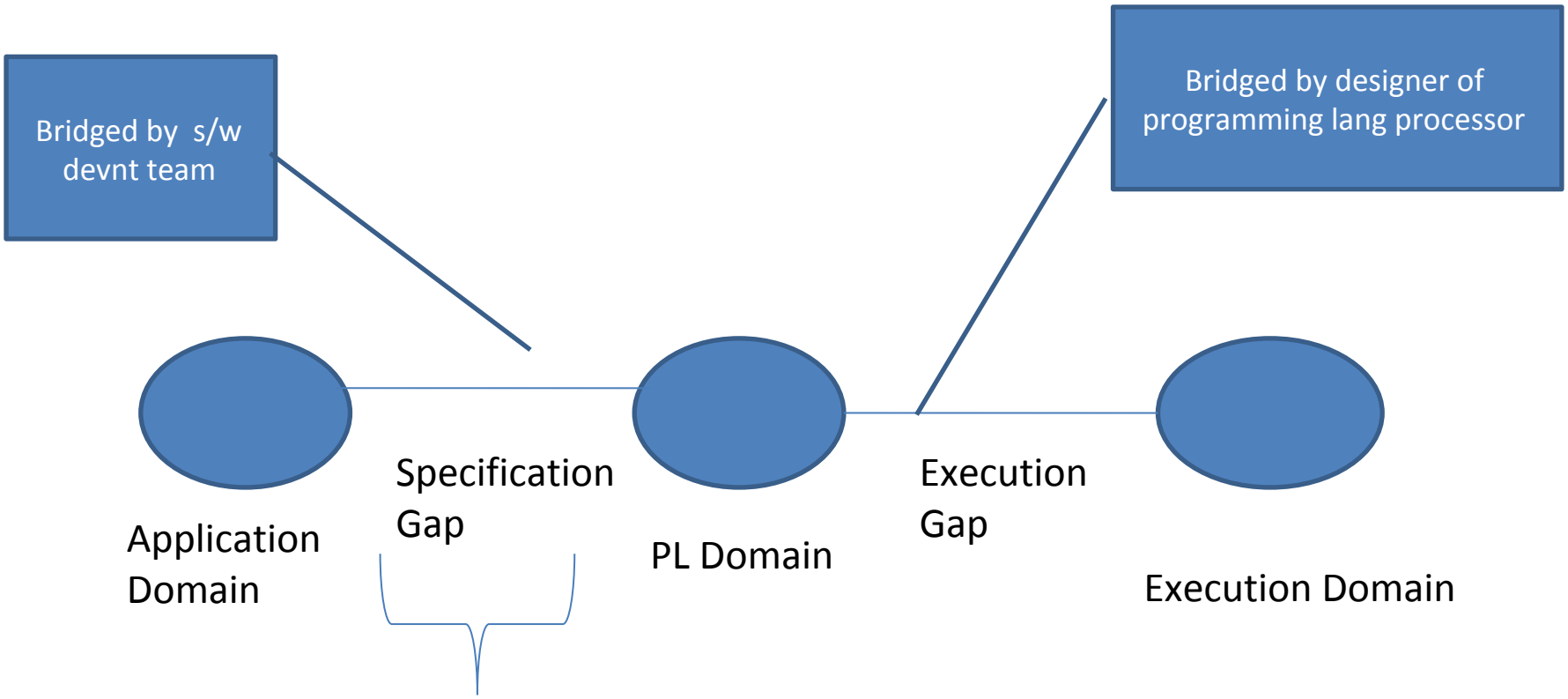
- Language Processors has mainly Three purposes:
- Bridge gap between Application Domain and Execution Domain
- Translation from one language to another
- To detect error in source during translation.



Introduction to System Programming



Types Of Software



Spectrum of language processors:

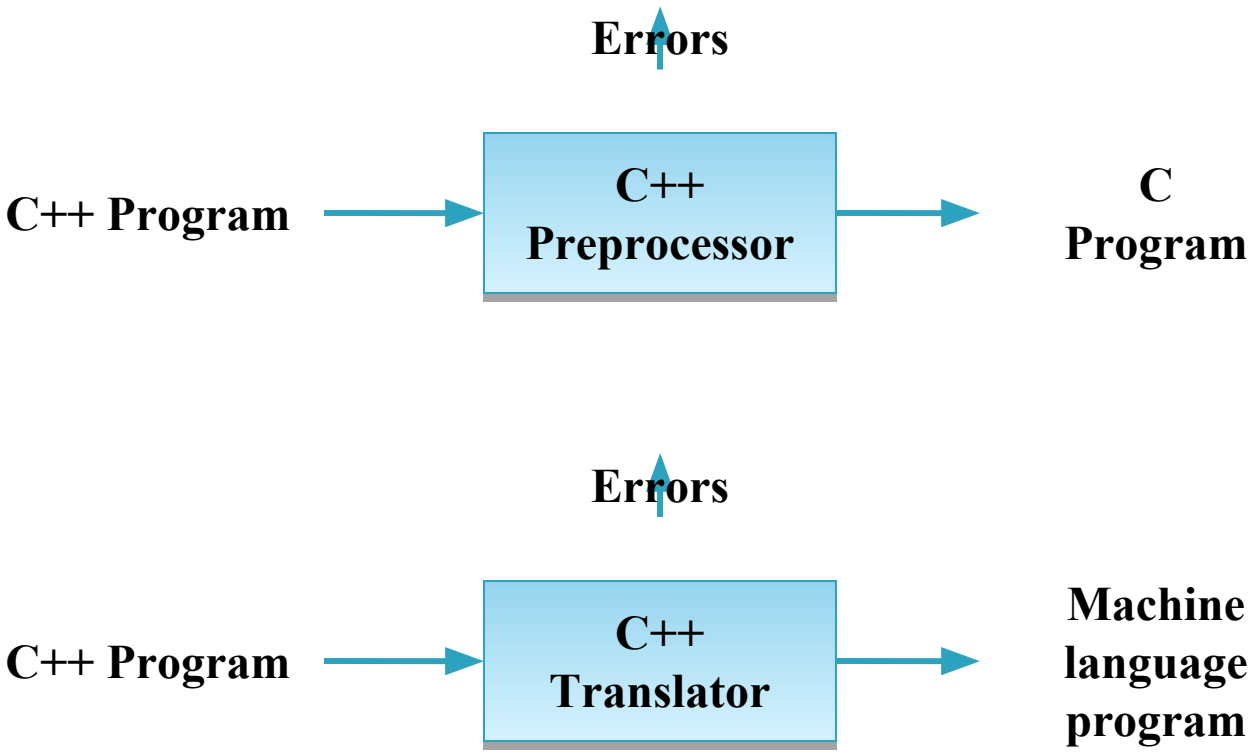
1. **Language translator :** Bridges an execution gap to the machine language. (e.g. .Net CLR)
2. **Assembler :**
3. **Compiler :**
4. **Detranslator :** Translator working in either direction.
5. **Preprocessor :** Is a language processor which bridges execution gap but is not a language translator.
6. **Language migrator :** bridges the specification gap between two PLs.
7. **Interpreter :** Is a language processor which bridges an execution gap without generating a machine language program.

Language Processing Activities

Fundamental language processing activities can be divided into those that bridges the specification gap and those that bridges the execution gap.
i.e.

1. Program generation activities
2. Program execution activities

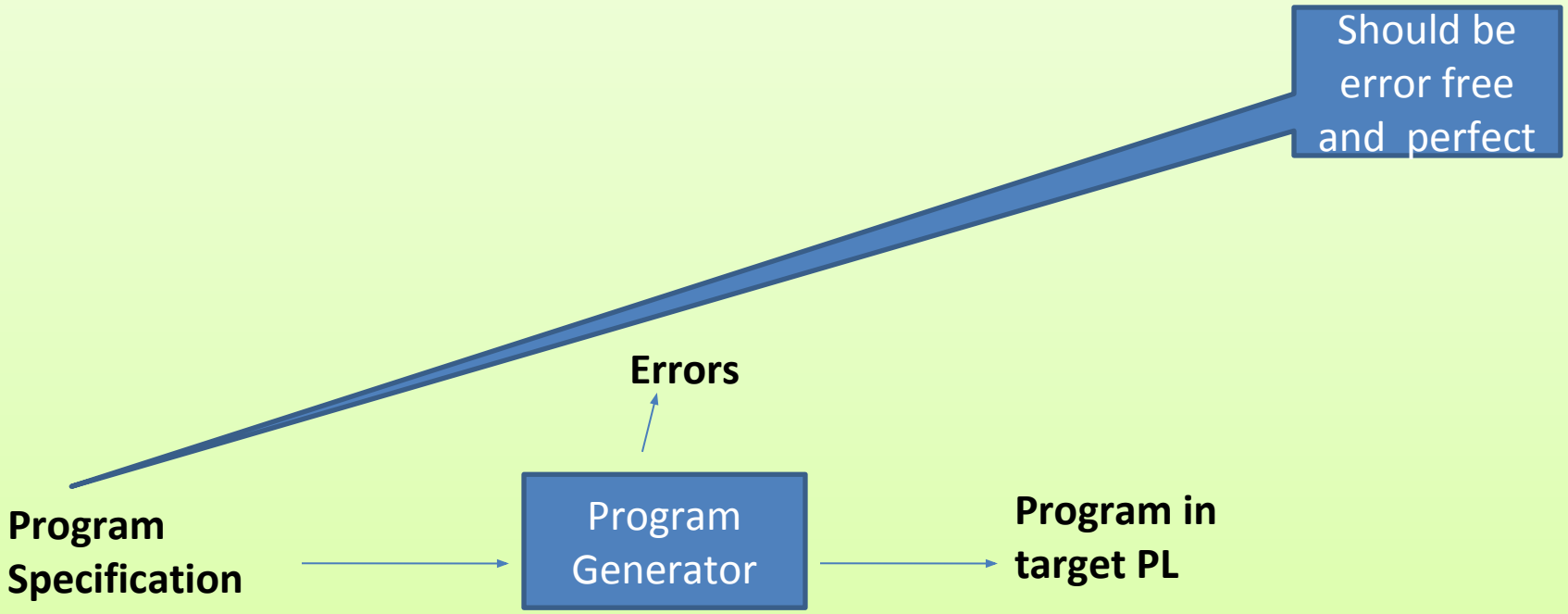
Example


Errors
C++ Program
C++ Preprocessor
C Program
Errors
C++ Program
C++ Translator
Machine language program

Language Processing Activities

Program Generation Activities:

-It aims at automatic generation of a program.



Language Processing Activities

Program Execution Activities

- Two popular model
  - Program Translation
  - Program Interpretation

Program Translation

- It bridges the execution gap by translating a program written in a PL, called source Program (SP), into an equivalent program in the machine or assembly language, called target program.
- **Characteristics of the program translation model:**

- A program must be translated before it can be

Language Processing Activities

Program Execution Activities

- Two popular model
  - Program Translation
  - Program Interpretation

Program Interpretation:

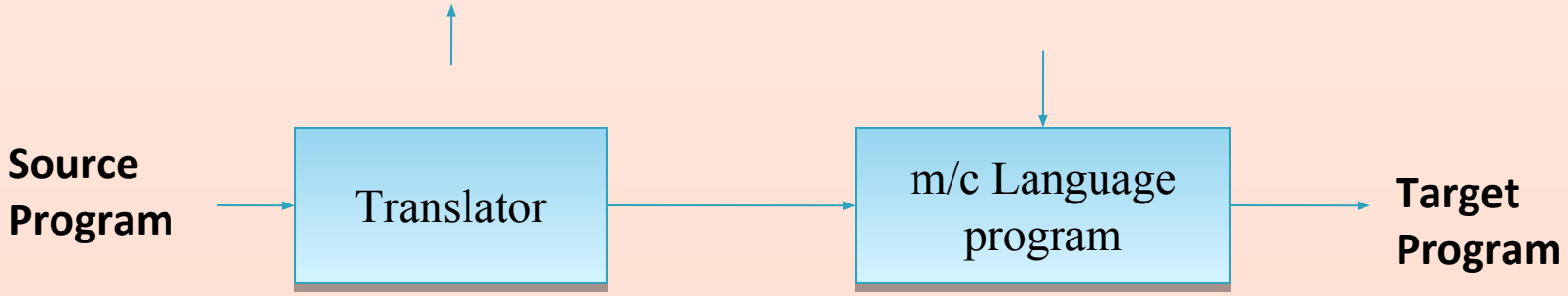
- Interpretation cycle

- 1. Fetch the statement.
- 2. Analyze the statement and determine its meaning
- 3. Execute the meaning of statement
- 4. Change PC and repeat the cycle

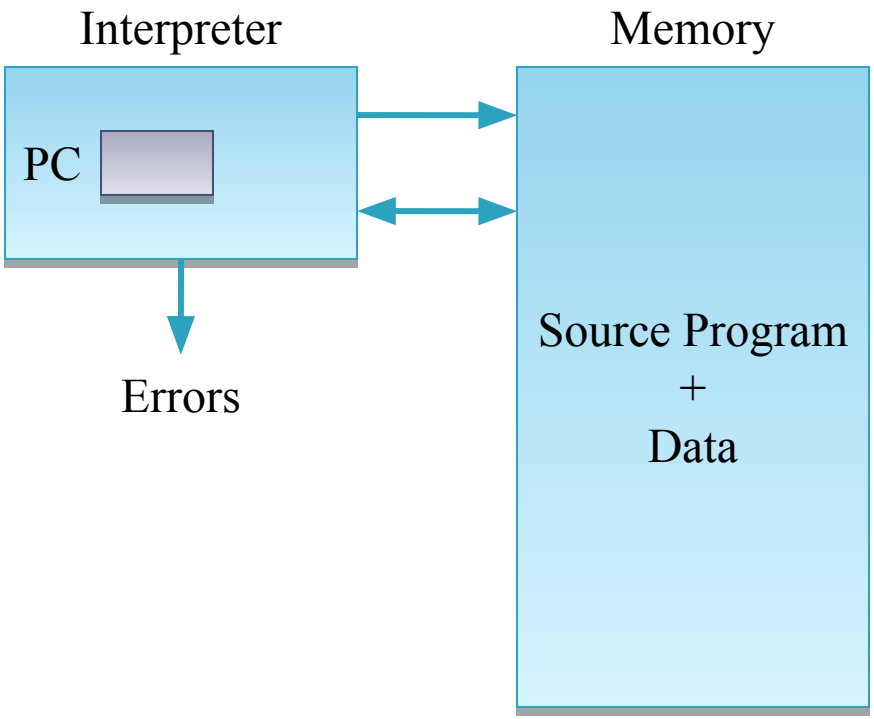
i.e. After interpretation, source program is retained in the source form itself, means, no target program form exists.

Language Processing Activities

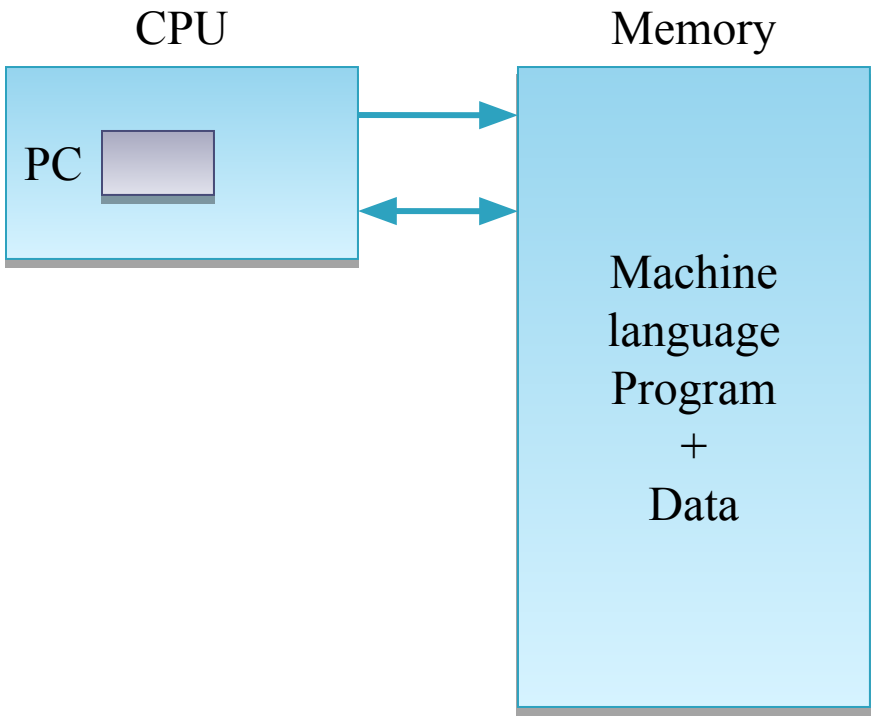
- The program translation model bridges the execution gap by translating a sources program into program in the machine or assembly language of the computer system, called target program.



Interpretation vs Execution



Interpretation



Program execution

Program Interpretation Cycle

- Interpretation cycle
  - 1. Fetch the statement.
  - 2. Analyze the statement and determine its meaning
  - 3. Execute the meaning of statement
  - 4. Change PC and repeat the cycle

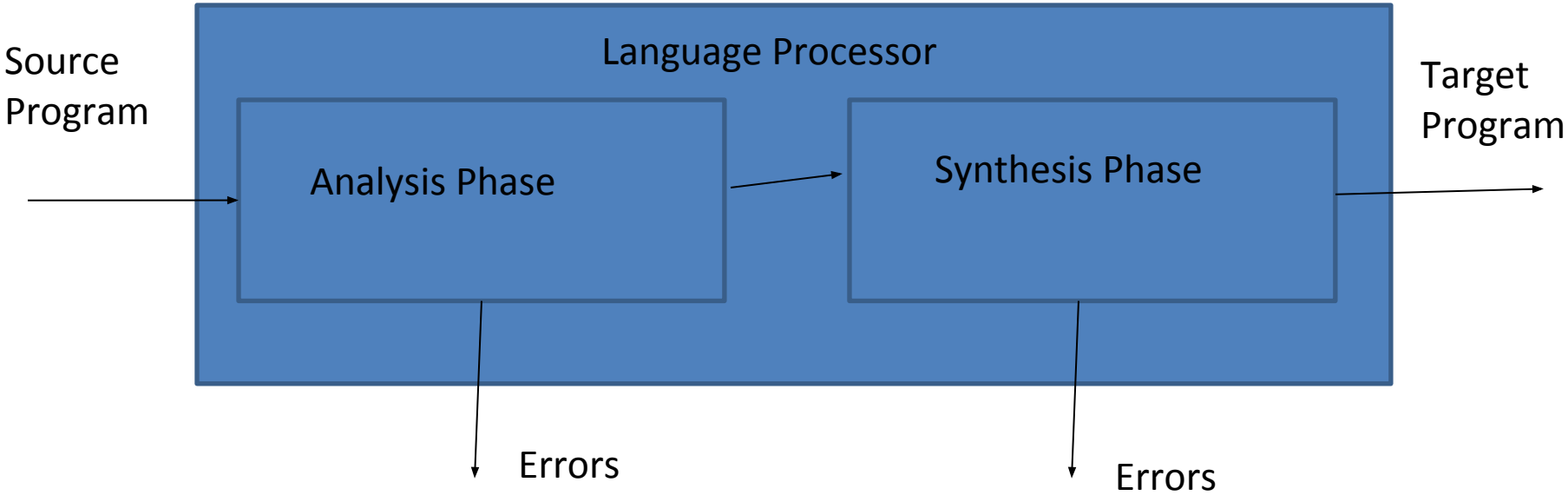
  - i.e. After interpretation, source program is retained in the source form itself, means, no target program form exists.

DIFFERENCE BETWEEN

COMPILER	INTERPRETER
READS ENTIRE PROGRAM AND LISTS ALL ERRORS AFTERWARDS.	READS PROGRAM LINE BY LINE AND STOPS EXECUTION ON ENCOUNTERING ERROR
MEMORY REQUIRED IS MORE DUE TO INTERMEDIATE OBJECT CODE	MEMORY EFFICIENT AS NO INTERMEDIATE CODE IS GENERATED
OVERALL EXECUTION TIME IS FASTER	EXECUTION IS SLOWER AS AFTER EVERY STATEMENT THE INTERPRETER CHECKS FOR ERRORS
DEBUGGING IS DIFFICULT AS YOU HAVE TO COMPILE EVERYTIME YOU CORRECT AN ERROR	DEBUGGING IS EASY AS THE INTERPRETER IMMEDIATELY INDICATES THE ERRORS
EXAMPLE : C, C++	EXAMPLE: BASIC, PYTHON

Phases of Language Processor

- **Language Processing = Analysis of SP + Synthesis of TP**



Phases of Language Processor

Analysis Phase

- The specification consists of three components:
  1. ***Lexical rules*** which govern the formation of valid lexical units in the source language.
  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.

Phases of Language Processor

- 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 *

Phases of Language Processor

Synthesis Phase

The specification consists of three components:

1. ***Lexical rules*** which govern the formation of valid lexical units in the source language.
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.

Phases of Language Processor

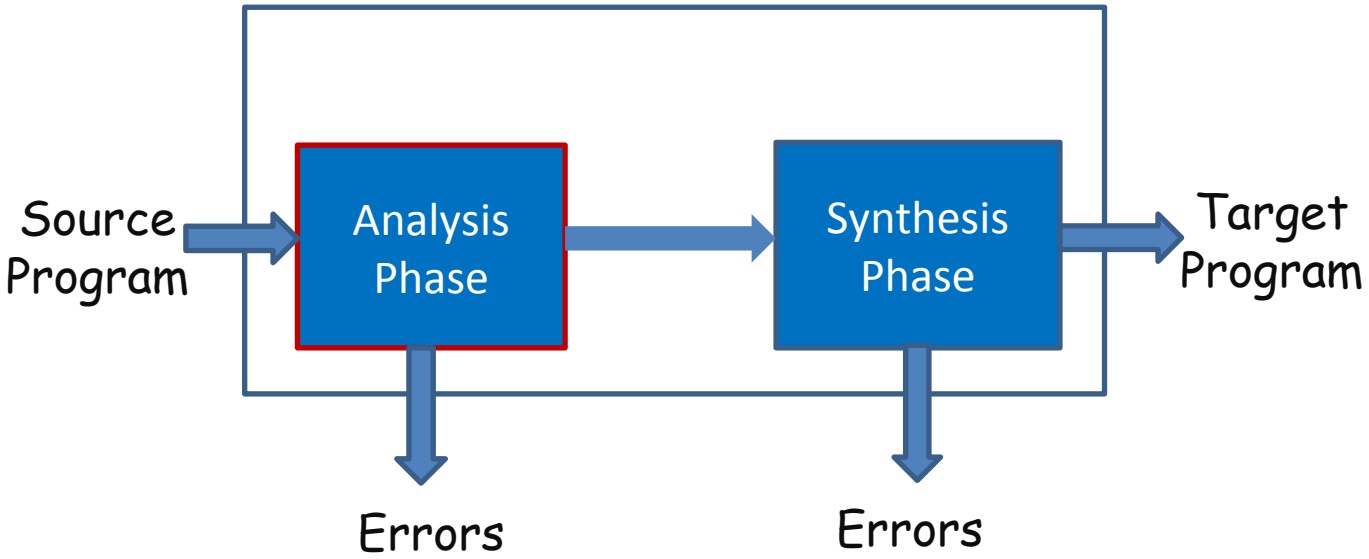
Synthesis Phase

- 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**)

Phases of Language Processor

Language Processor



- 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

Forward Reference

- It is a reference to the entity which precedes its definition in the program. Using a variable before a value is assigned to it is called a forward reference.

- Forward referenced variables have the scope of the location where they are first referenced. They cannot be read before a value has been assigned to them, but they can be used for further forward reference.

Language Processor Pass

- Pass I : performs analysis of Sorce Program and notes relevant information.
- Pass II : performs synthesis of target program.

Pass I analysis Source Program and generates IR which is given as input to Pass II to generate target code.

Intermediate Representation (IR)

- An IR reflects the effect of some but not all, analysis and synthesis tasks performed during language processing.