System Programming

Lecture 2

LANGUAGE PROCESSING ACTIVITIES

Divided into those that bridge specification gap and those that bridge execution gap

1. Program generation activities
   Aims at automatic generation of a program

2. Program execution activities
   Organizes execution of a program written in a PL on computer system

Program Generation – 1/5

- A software system which accepts the specification of a program to be generated, and generates a program in the target PL
- New domain between the application and PL domains
  Program Generator Domain
- Specification gap is now the gap between the application domain and program generator domain

Program Generation – 2/5

- Reduction in specification gap increases the reliability of the generated program
- Since generator domain is close to application domain, it is easy for the designer or programmer to write the specification of the program to be generated
- The harder task of bridging the gap to the PL domain is performed by the generator

Program Generation – 3/5

- Only verify the correctness of the specification input to program generator
- Simpler task than verifying correctness of generated program
- Good diagnostic (error indication)
- Execution gap between target PL domain and execution domain is bridged by the compiler or interpreter for PL

Program Generation – 4/5

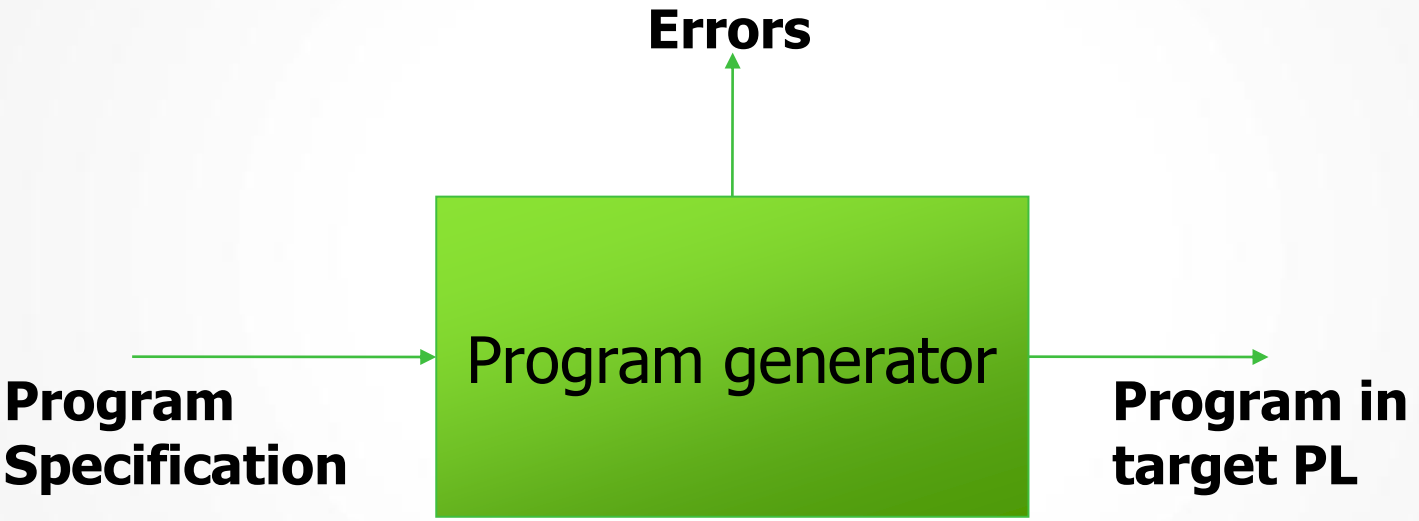

Fig : Program generation

Program Generation – 5/5

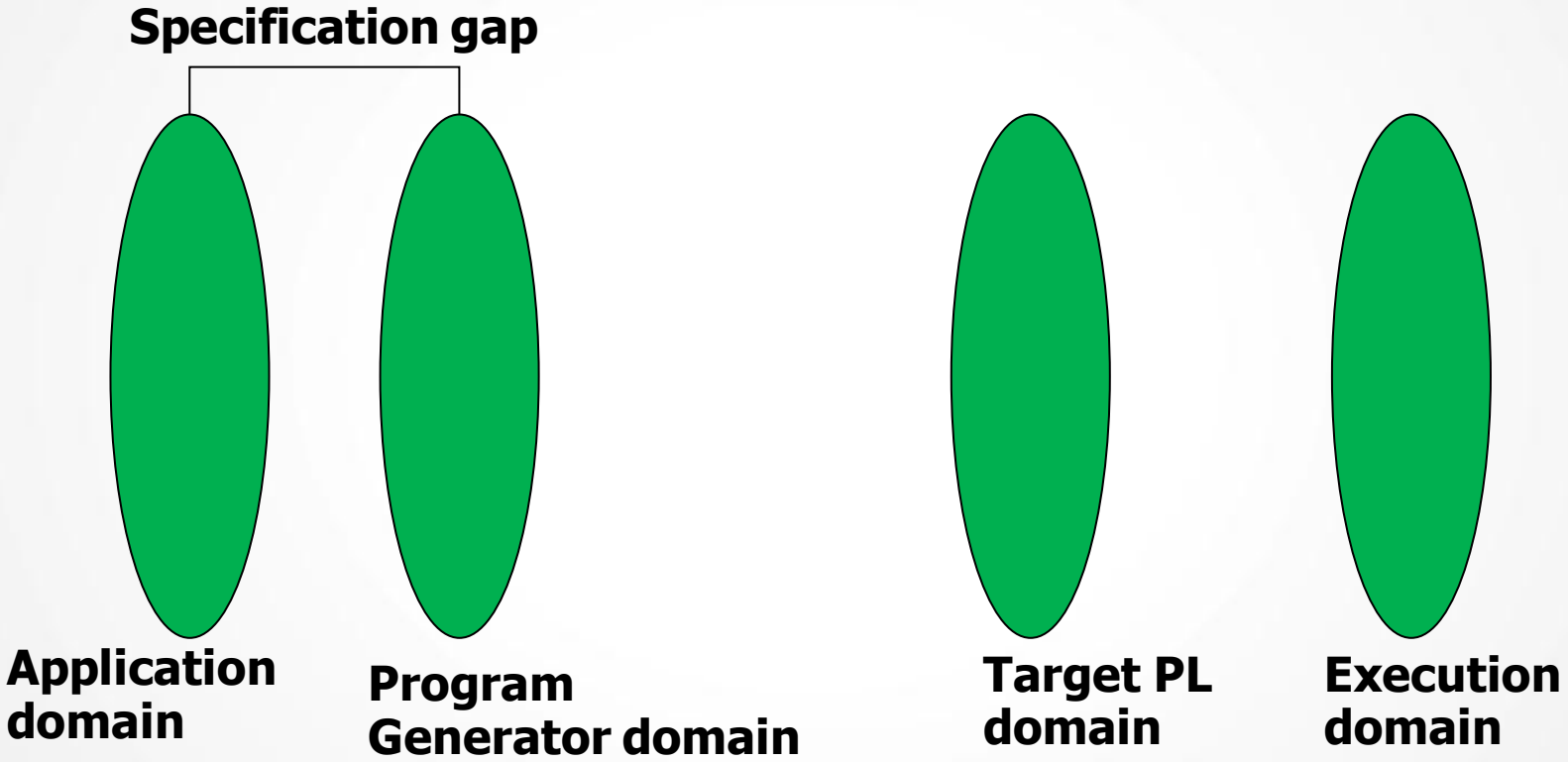

Fig : Program generator domain

Program execution

Two popular models for program execution are

- Translation
- Interpretation

Program translation – 1/2

- The program translation model bridges the execution gap by translating a program written in PL called the source program (SP), into an equivalent program in the machine or assembly language of the computer system, called the target program (TP)

- Characteristics
  - 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 translation – 2/2

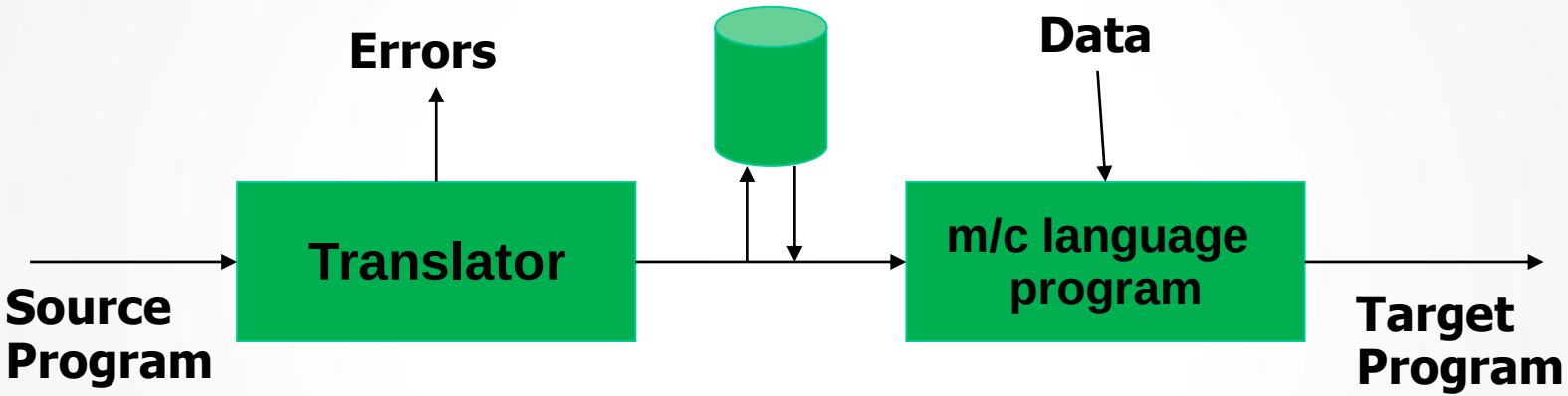

Fig : Program Translation Model

Program interpretation – 1/6

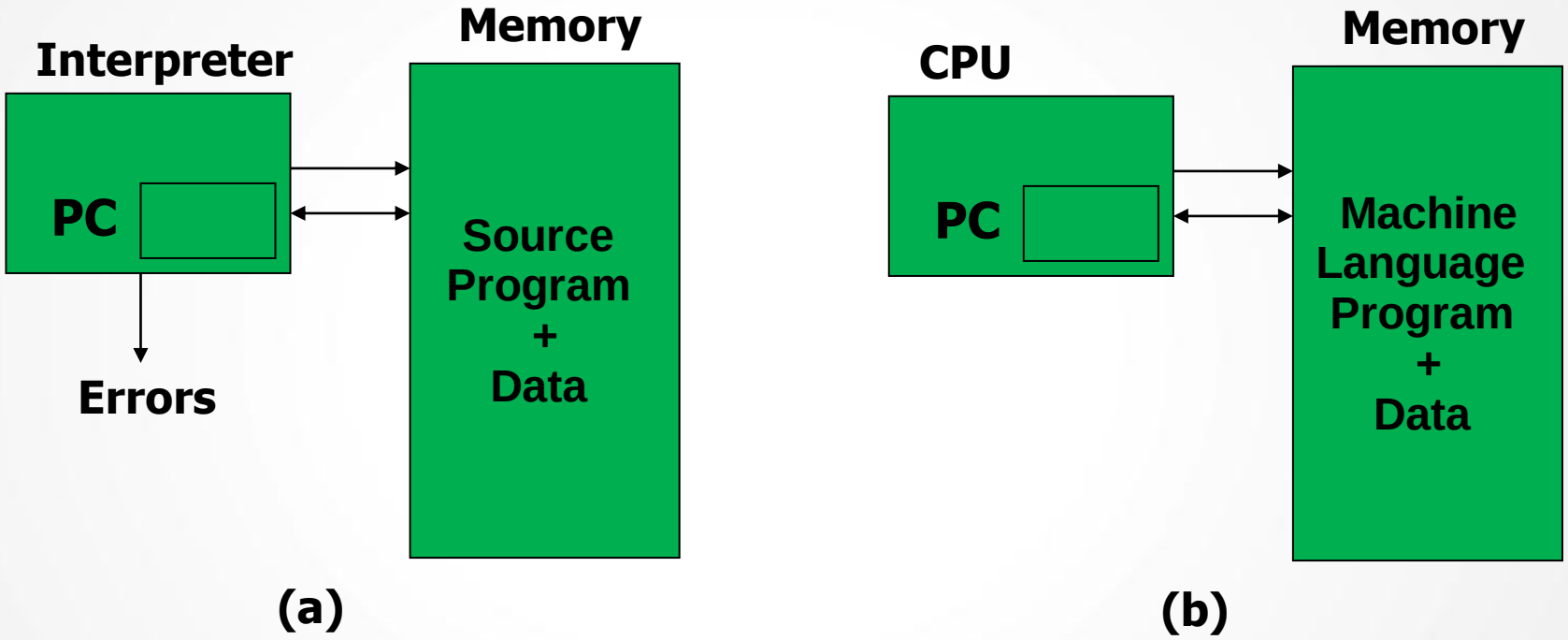

Fig : Schematics of (a) interpretation and (b) program execution

Program interpretation – 2/6

Fig (a) shows a schematic of 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 performs actions which implement it
- This includes computational and input-output actions

Program interpretation – 3/6

- To understand the functioning of an interpreter, note the striking similarity between the interpretation schematic (Fig(a)) and a schematic of execution of machine language program by the CPU of a computer system (Fig(b))

- Program counter (PC) is used to note the address of the next instruction to be executed

Program interpretation – 4/6

- This instruction is subjected to the instruction execution cycle consisting of following steps:-
  - Fetch the instruction
  - Decode the instruction to determine the operation to be performed, and also its operands
  - Execute the instruction

- At the end of cycle, the instruction address in PC is updated and the cycle is repeated for next instruction
- Thus, PC can indicate which statement of the source program is to be interpreted next

Program interpretation – 5/6

- The statement would be subjected to interpretation cycle, which consists of following steps
  - Fetch the statement
  - Analyze the statement and determine its meaning, viz. the computation to be performed and its operands
  - Execute the meaning of statement

Program interpretation – 6/6

- The source program is retained in the source form itself, i.e. no target program form exists
- A statement is analyzed during its interpretation