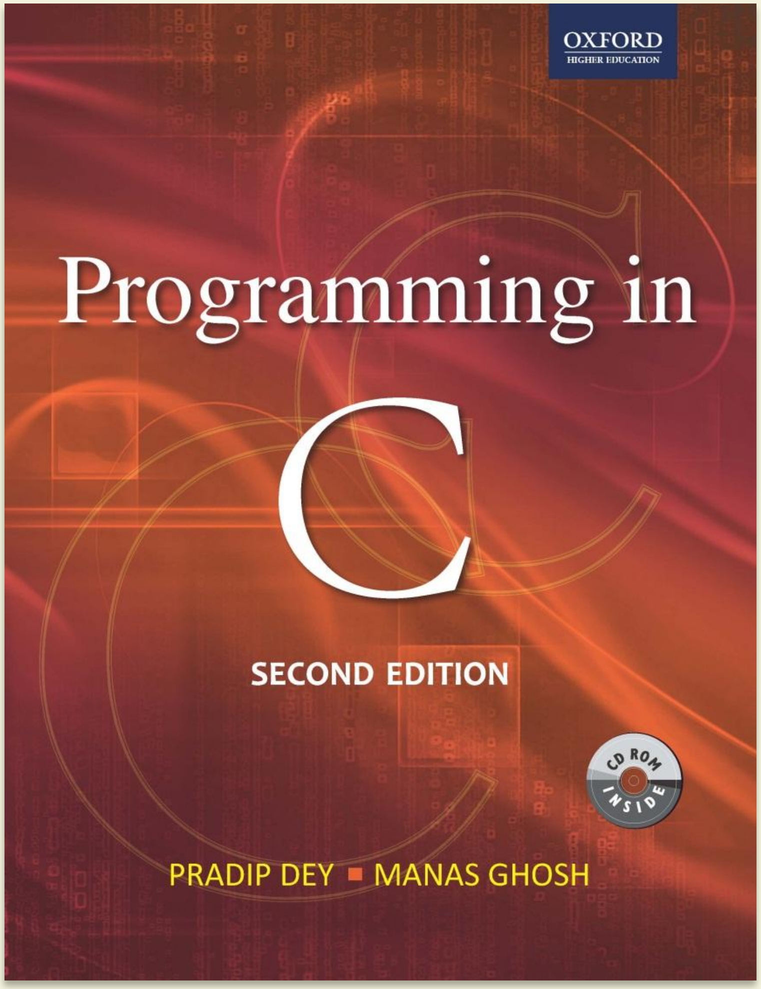


Programming in C

Pradip Dey & Manas Ghosh

CHAPTER 1

Introduction to Programming, Algorithms & Flow Charts

PROGRAM & PROGRAMMING

- A program is a set of logically related instructions that is arranged in a sequence that directs the computer in solving a problem.
- The process of writing a program is called programming.

- Software is a collection of computer programs and related data that provides the instructions for telling a computer what to do and how to do it.
- Computer software can be broadly classified into two categories :

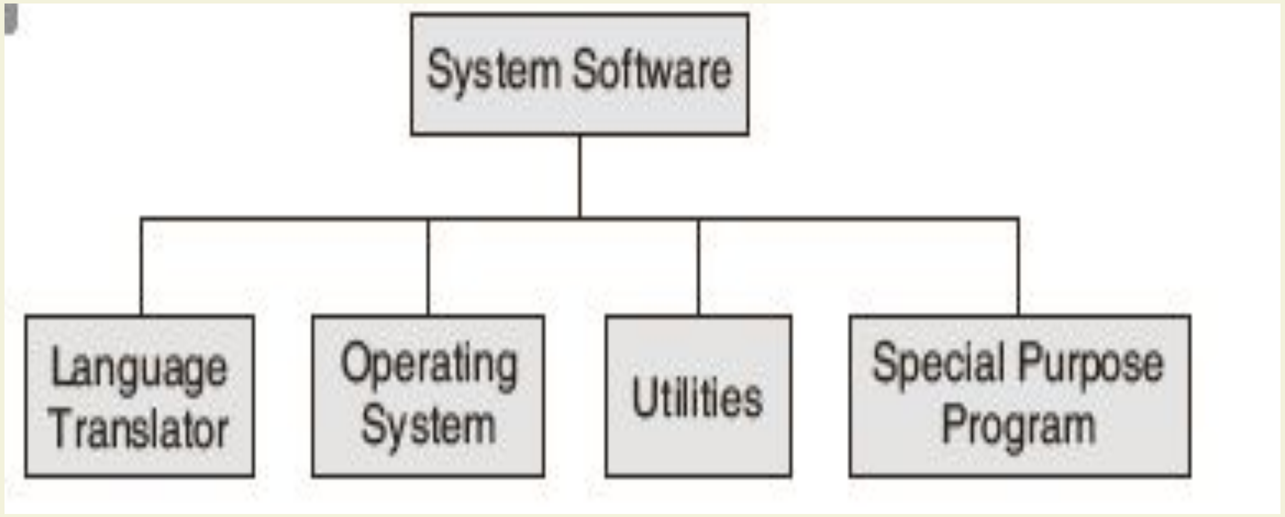
  (a) . system software

  &

  (b) . application software

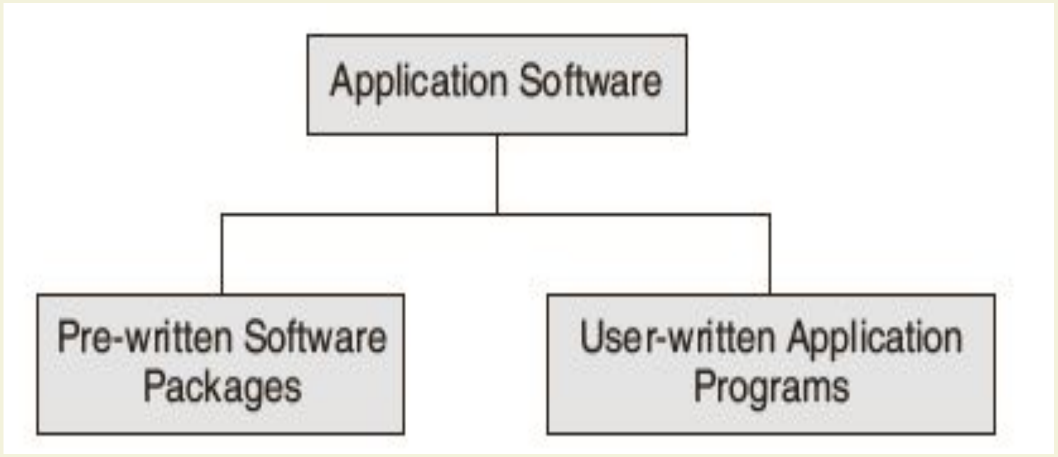
SYSTEM SOFTWARE

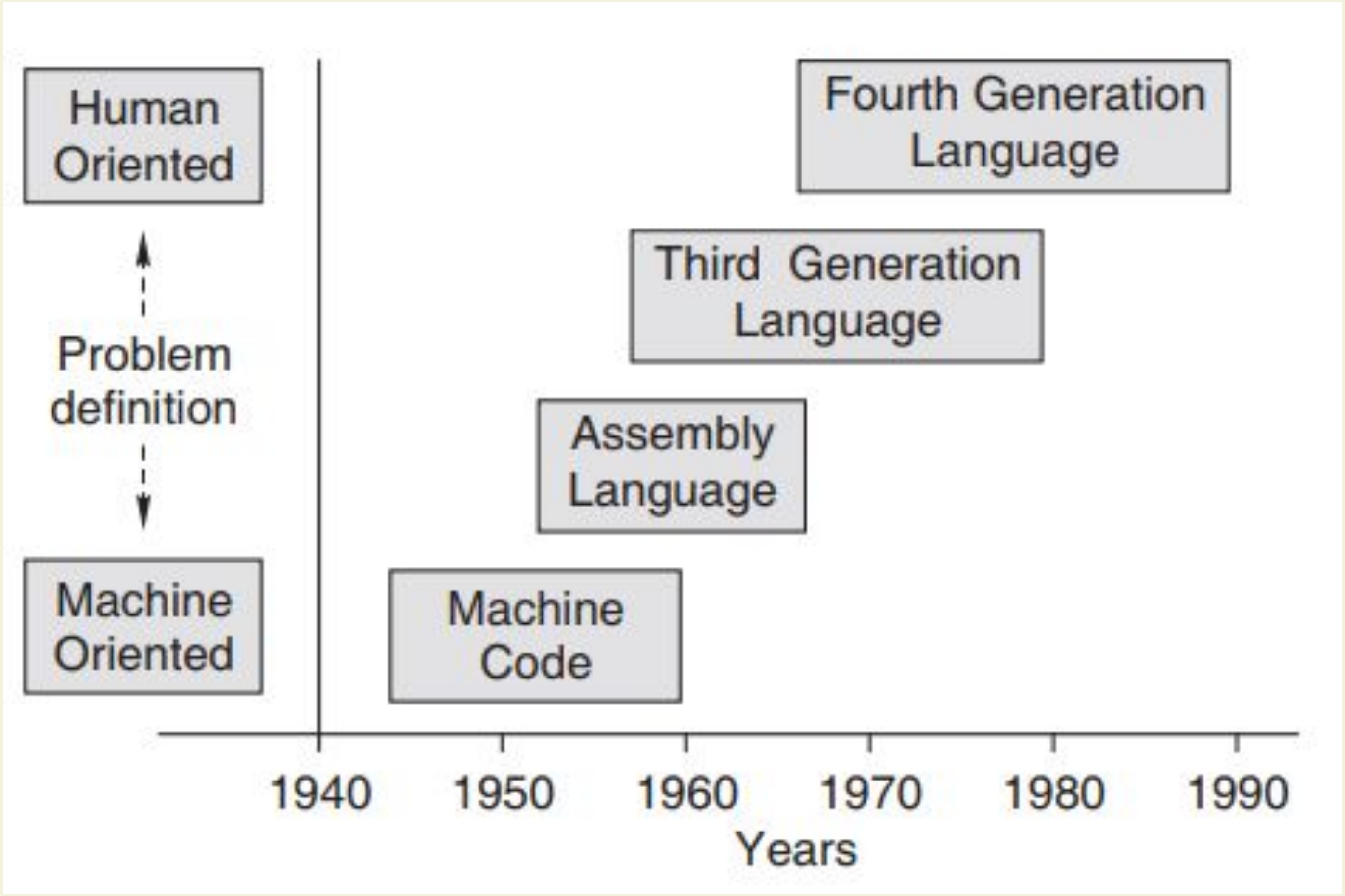
- System software is a collection of programs that interfaces with the hardware.
- Categories of system software :



APPLICATION SOFTWARE

- Application software is written to enable the computer to solve a specific data processing task.
- Categories of application software :


Application Software
Pre-written Software Packages
User-written Application Programs


Human Oriented
Problem definition
Machine Oriented
Fourth Generation Language
Third Generation Language
Assembly Language
Machine Code
1940
1950
1960
1970
1980
1990
Years

PROGRAMMING LANGUAGE

- A programming language is composed of a set of instructions in a language understandable to the programmer and recognizable by a computer.

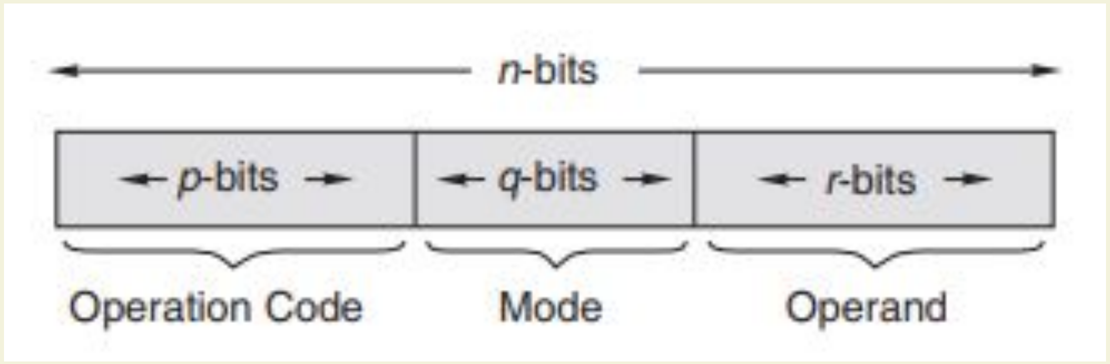
- Programming languages can be classified as

  (a) High-level language - BASIC, COBOL & FORTRAN(application programs).

  (b) Middle level language - C (application & system programs).

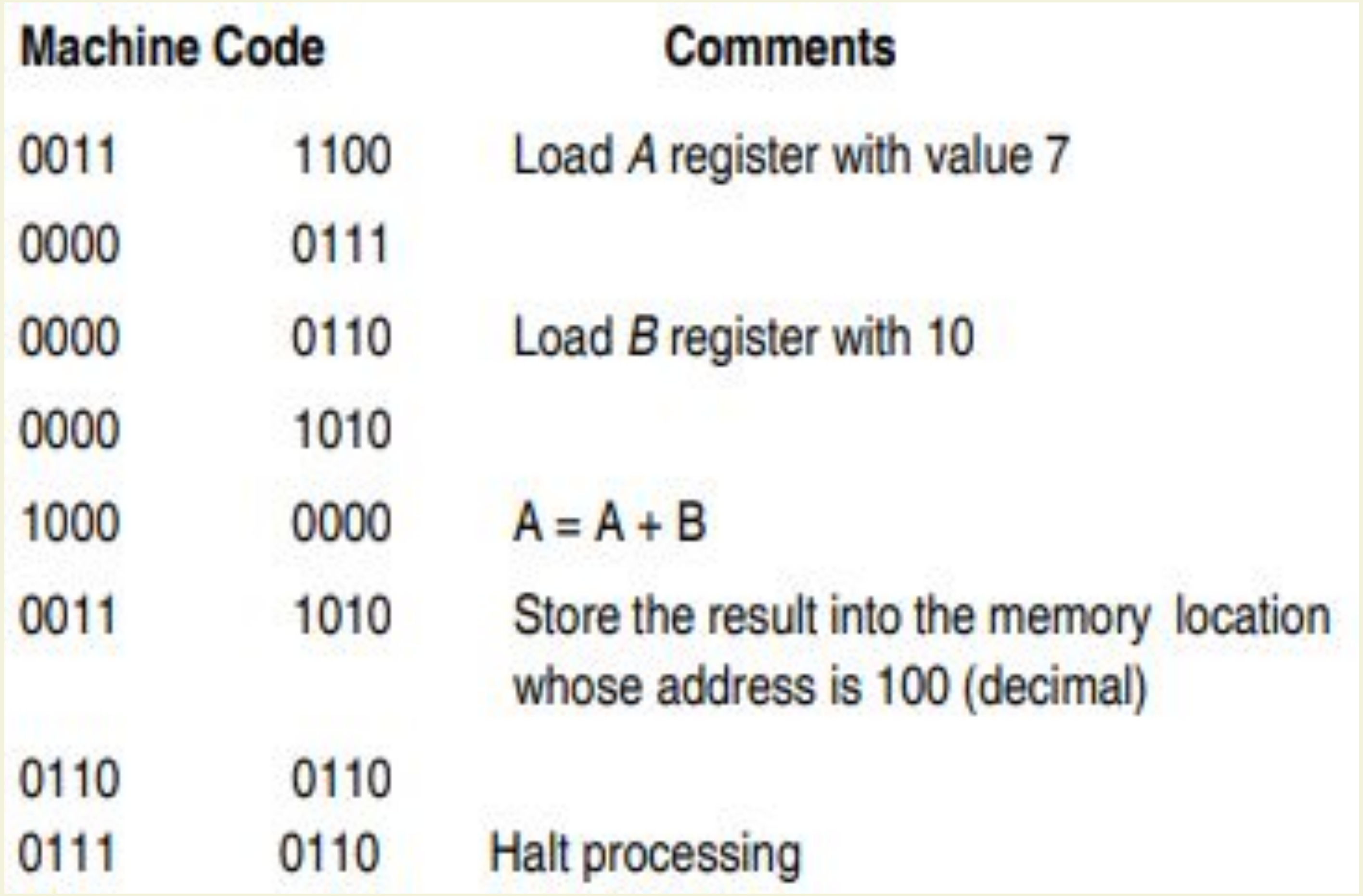
  (c) Low level language – assembly language (system programs).

Machine Language



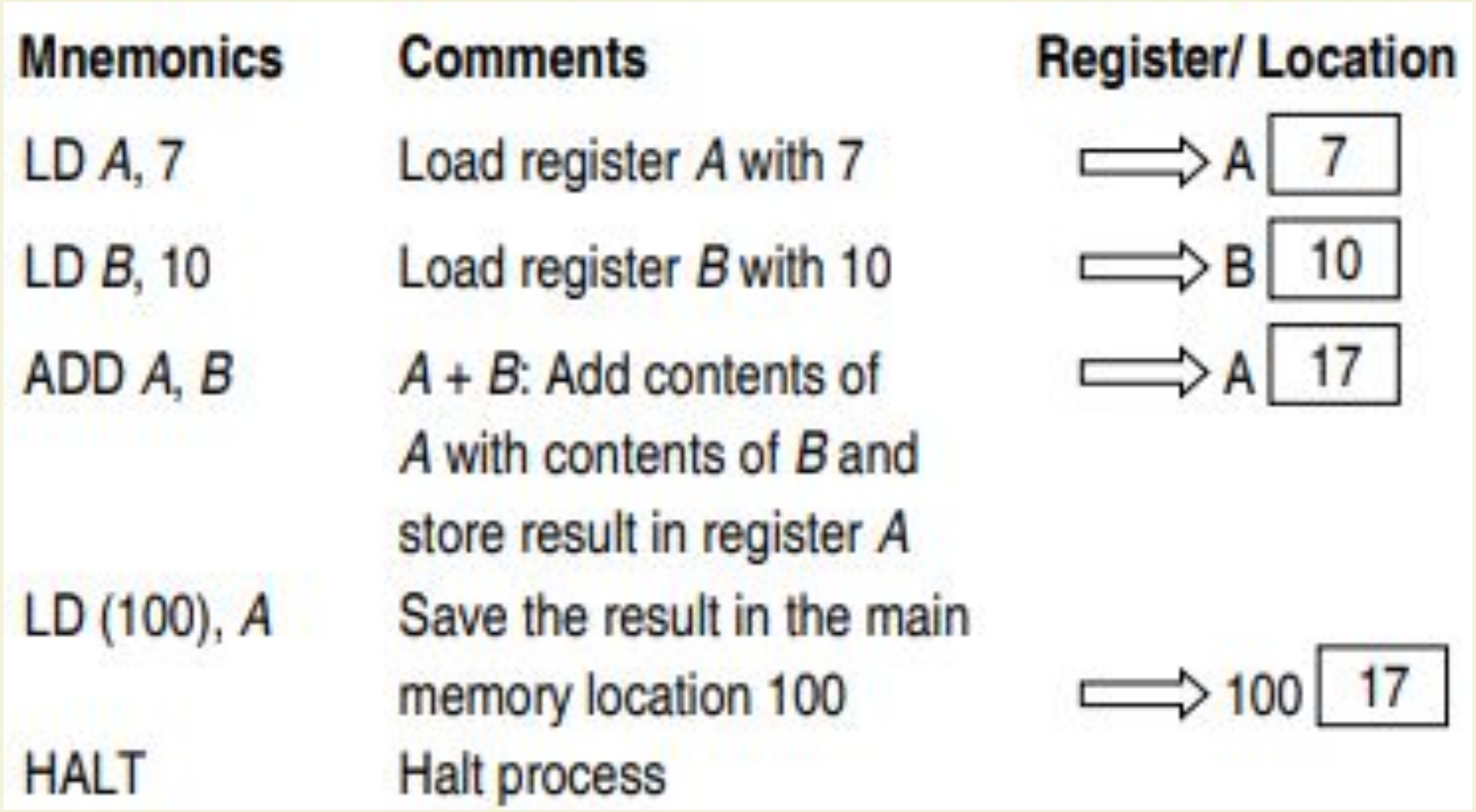
- Difficult to use
- Machine dependent
- Error prone
- Difficult to debug and modify

Machine Language



Machine Code		Comments
0011	1100	Load *A* register with value 7
0000	0111	
0000	0110	Load *B* register with 10
0000	1010	
1000	0000	A = A + B
0011	1010	Store the result into the memory location whose address is 100 (decimal)
0110	0110	
0111	0110	Halt processing

Assembly Language



Mnemonics	Comments	Register/ Location
LD *A*, 7	Load register *A* with 7	$\Rightarrow$ A $\boxed{7}$
LD *B*, 10	Load register *B* with 10	$\Rightarrow$ B $\boxed{10}$
ADD *A*, *B*	*A* + *B*: Add contents of *A* with contents of *B* and store result in register *A*	$\Rightarrow$ A $\boxed{17}$
LD (100), *A*	Save the result in the main memory location 100	$\Rightarrow$ 100 $\boxed{17}$
HALT	Halt process	

Assembly Language

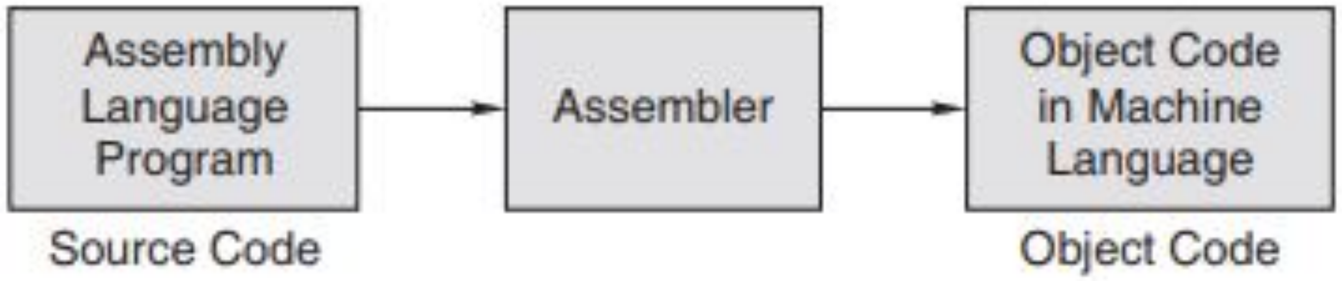
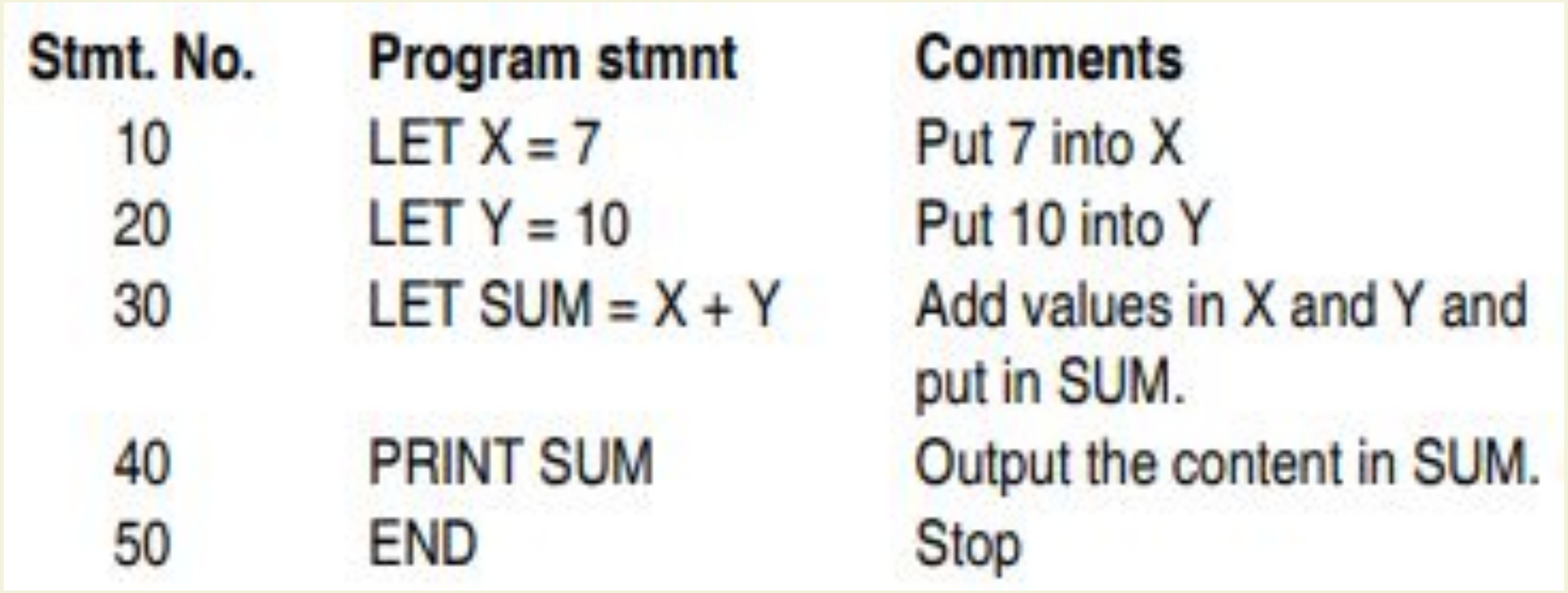


Figure 1.6 Assembler

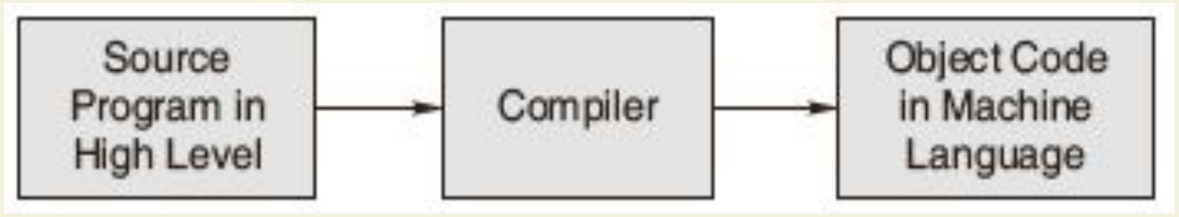
High Level Language



Stmt. No.	Program stmnt	Comments
10	LET X = 7	Put 7 into X
20	LET Y = 10	Put 10 into Y
30	LET SUM = X + Y	Add values in X and Y and put in SUM.
40	PRINT SUM	Output the content in SUM.
50	END	Stop

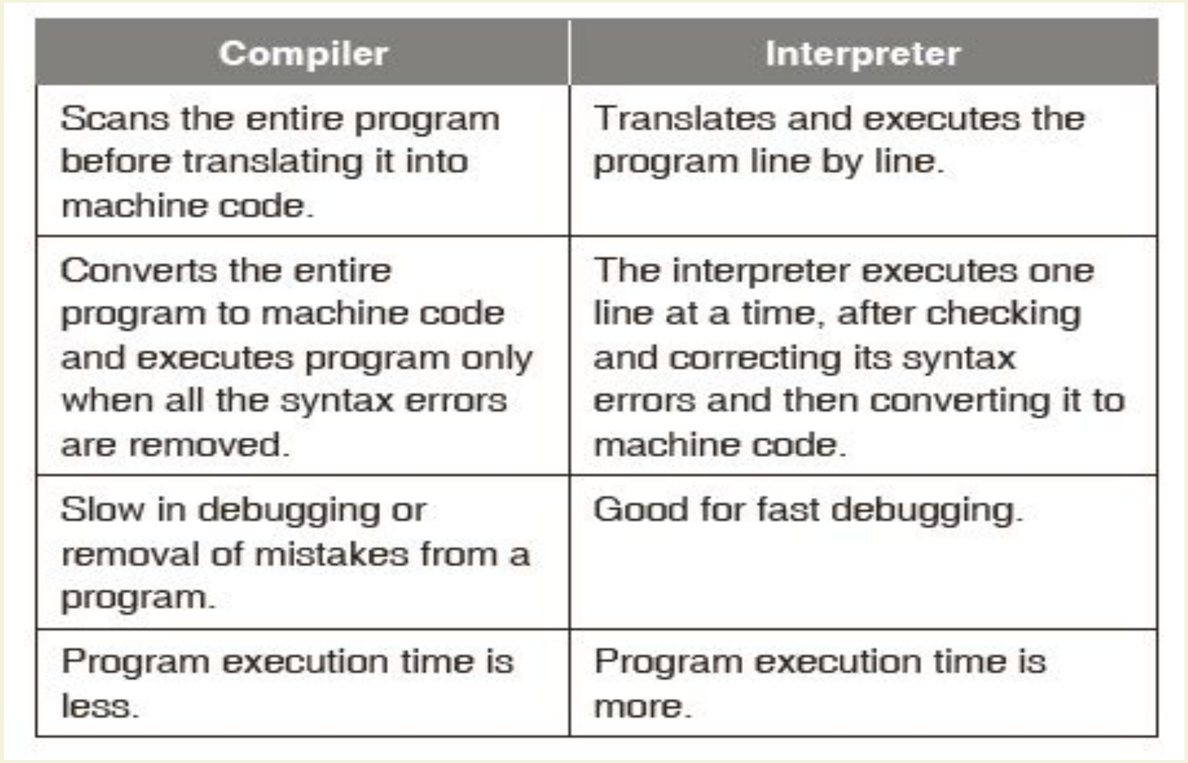
COMPILER

- For executing a program written in a high-level language, it must be first translated into a form the machine can understand. This is done by a software called the *compiler.*
- The compiling process consists of two steps:
  - a . The analysis of the source program and
  - b . The synthesis of the object program in the machine language of the specified machine.
- Compiler action :



INTERPRETER

- During the process of translation There is another type of software that also does translation. This is called an interpreter.
- Differences between compiler and interpreter :

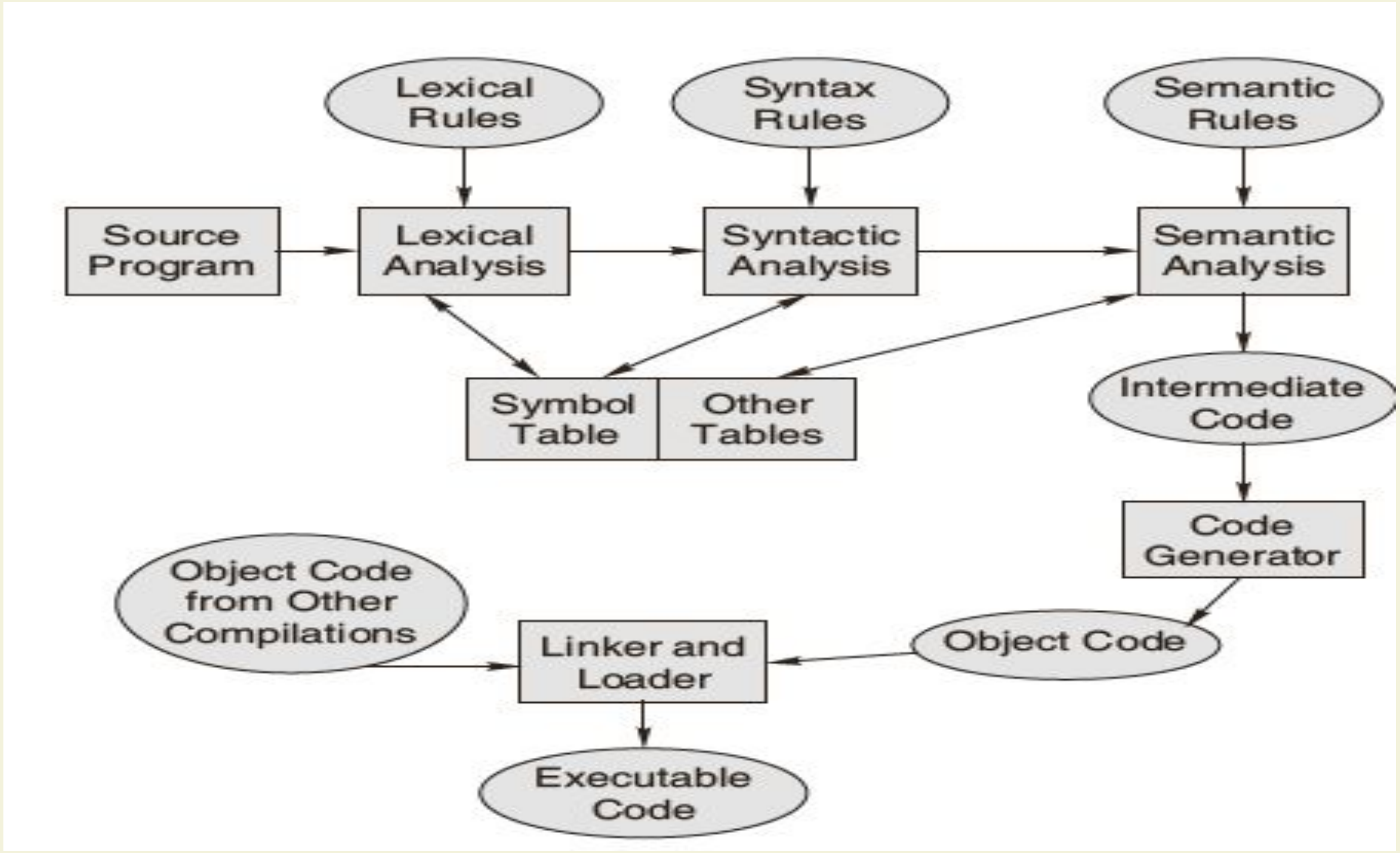


Compiler	Interpreter
Scans the entire program before translating it into machine code.	Translates and executes the program line by line.
Converts the entire program to machine code and executes program only when all the syntax errors are removed.	The interpreter executes one line at a time, after checking and correcting its syntax errors and then converting it to machine code.
Slow in debugging or removal of mistakes from a program.	Good for fast debugging.
Program execution time is less.	Program execution time is more.

COMPILING & EXECUTING HIGH LEVEL LANGUAGE

- The compiling process consists of two steps: the analysis of the source program and the synthesis of the object program in the machine language of the specified machine.
- The analysis phase uses the precise description of the source programming language.
- A source language is described using (a) *lexical rules, (b)syntax rules, and (c)semantic* rules.

THE PROCESS OF COMPILATION



EXECUTION STEPS OF A PROGRAM

- Steps :

1. ***Translation of the program resulting in the object*** program.
2. ***Linking of the translated program with other object*** programs needed for execution, thereby resulting in a binary program.
3. ***Relocation of the program to execute from the specific*** memory area allocated to it.
4. ***Loading of the program in the memory for the purpose*** of execution.

LINKER

- Linking resolves symbolic references between object programs. It makes object programs known to each other.
- Linking makes the addresses of programs known to each other so that transfer of control from one subprogram to another or a main program takes place during execution.
- In FORTRAN/COBOL , all program units are translated separately.

RELOCATION

- Relocation is more than simply moving a program from one area to another in the main memory.
- Relocation means adjustment of all address-dependent locations, such as address constant, to correspond to the allocated space, which means simple modification of the object program so that it can be loaded at an address different from the location originally specified.

LOADER

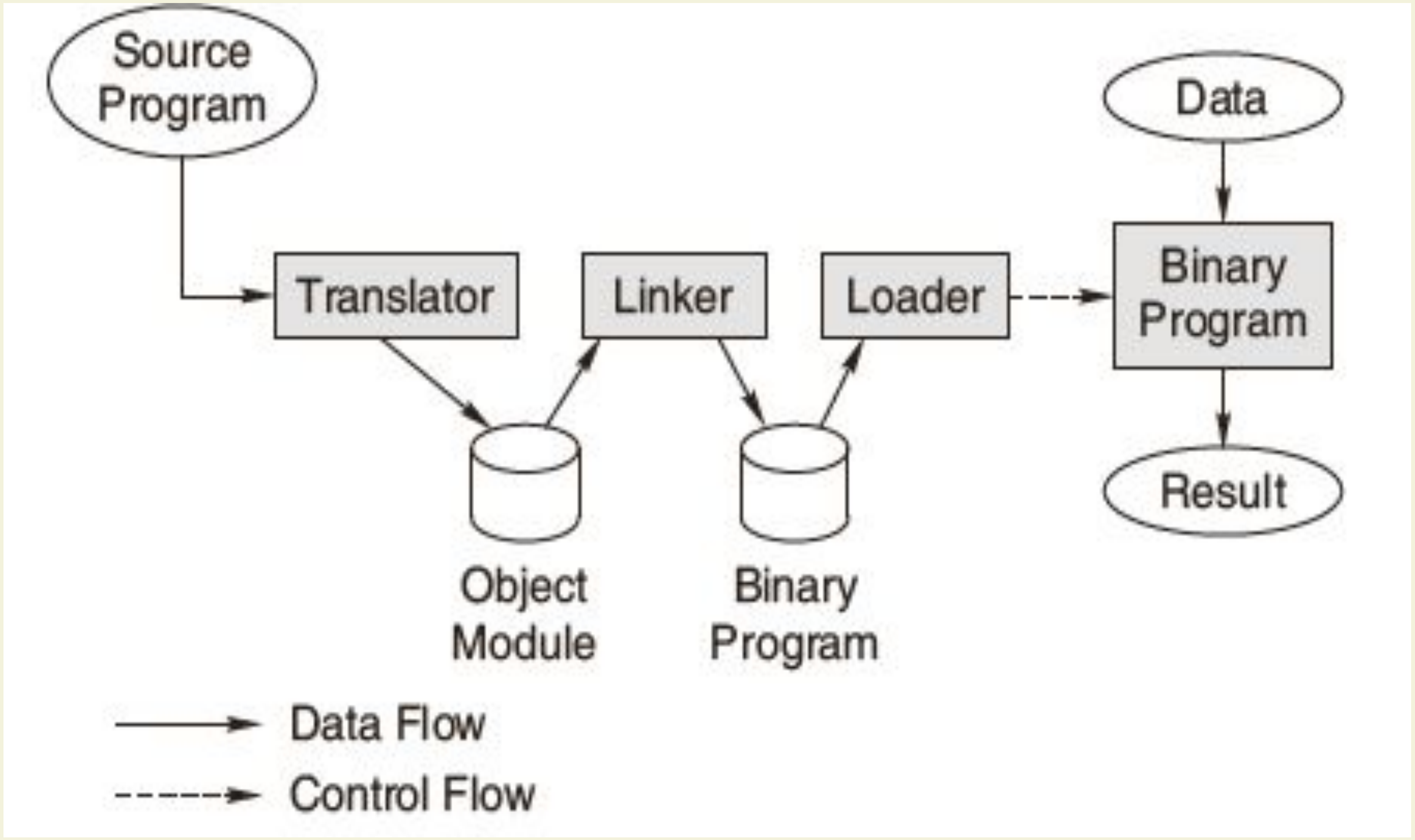
- Loading means physically placing the machine instructions and data into main memory, also known as primary storage area.
- <u>The functions performed by the loader are</u> :

a. Assignment of load-time storage area to the program
b. Loading of program into assigned area
c. Relocation of program to execute properly from its load time storage area
d. Linking of programs with one another

PROGRAM EXECUTION

- When a program is compiled and linked, each instruction and each item of data is assigned an address.
- At execution time, the CPU finds instructions and data from these addresses.
- The program counter, is a CPU register that holds the address of the next instruction to be executed in a program.
- The CPU has random access capability to any and all words of the memory, no matter what their addresses.

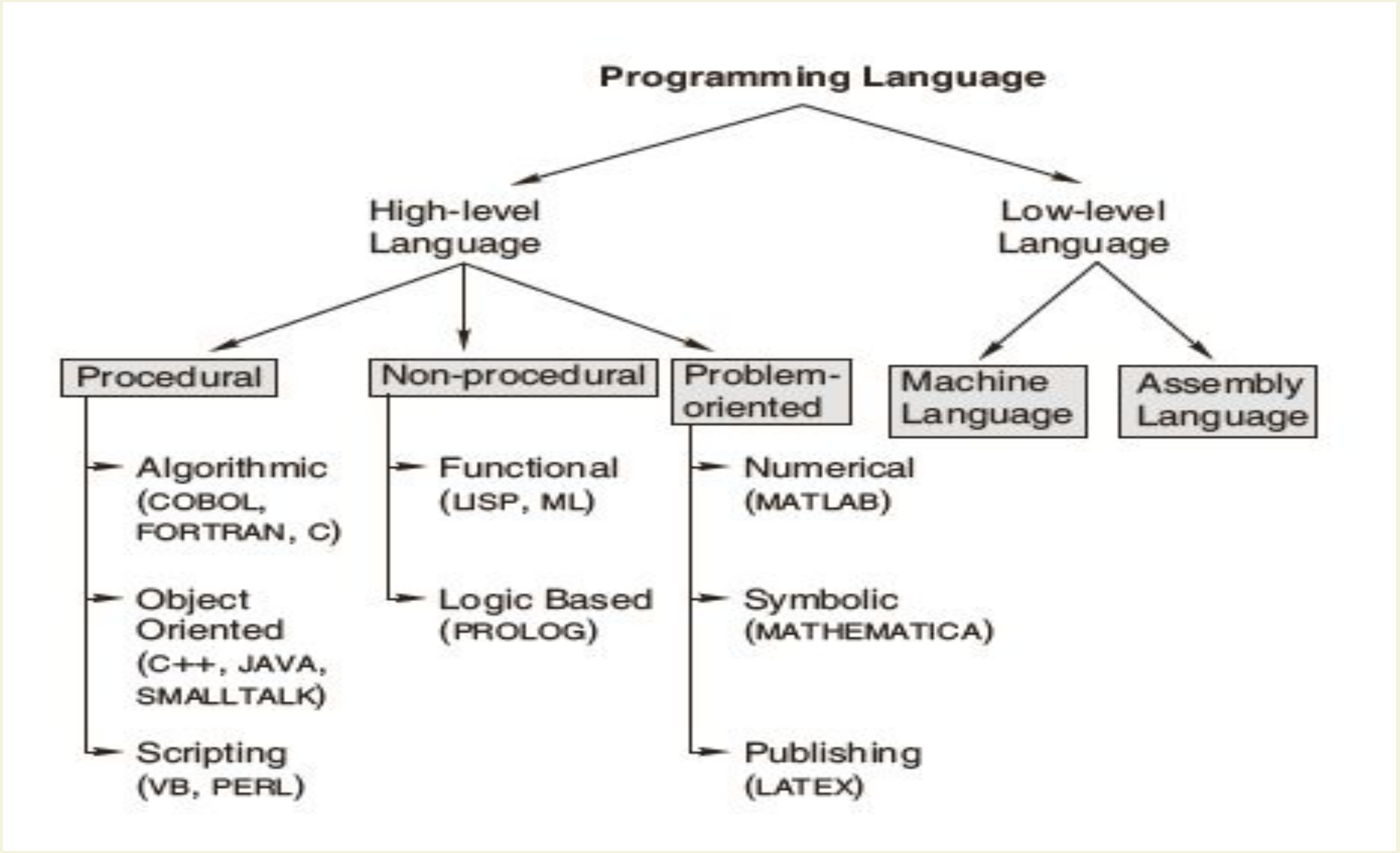
BLOCK DIAGRAM OF PROGRAM EXECUTION



THIRD,FORTH & FIFTH GENERATION LANGUAGE

- Third generation programming language specifies how to perform a task using a large number of procedural instructions and is file oriented.
- Fourth generation programming language specifies what task has to be performed using fewer instructions and is database oriented.
- Fifth generation programming language resembles human speech and eliminates the need for the user or programmer to learn a specific vocabulary, grammar ,or syntax.

CLASSIFICATION OF PROGRAMMING LANGUAGES



STUCTURED PROGRAMMING

- Structured programming involves top–down analysis for program solving, modularization of program structure and organizing structured code for individual module.
- Top-down analysis breaks the whole problem into smaller logical tasks and defines the hierarchical link between the tasks.
- Modularization of program structure means making the small logical tasks into independent program modules that carries out the desired tasks.
- Structured coding is structured programming which consists of writing a program that produces a well organized module.

ALGORITHM

- An algorithm is ‘an effective procedure for solving a problem in a finite number of steps’.
- A well-designed algorithm has termination and correctness properties.
- The four common ways of representing an algorithm are the Step-form, Pseudo-code, Flowchart and Nassi-Schneiderman .
- algorithms show these three features:

a. Sequence (also known as process)

b. Decision (also known as selection)

c. Repetition (also known as iteration or looping)

VARIABLE & SUBROUTINE

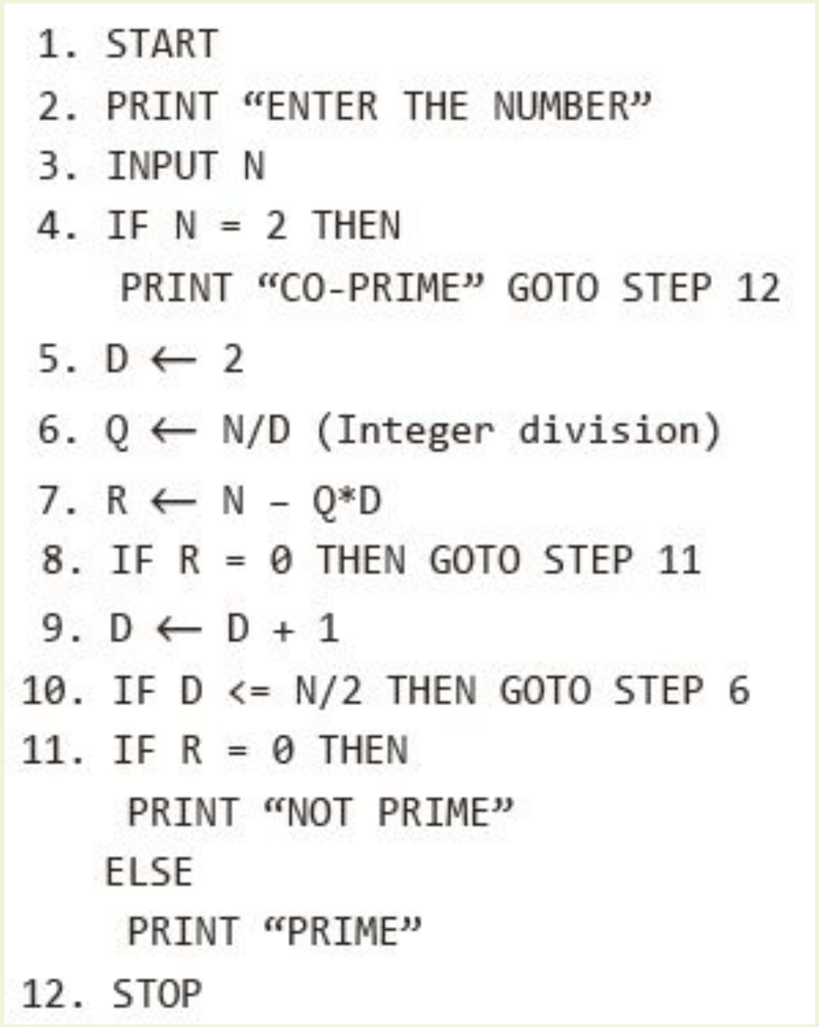
- A variable, which has a name, is a container for a value that may vary during the execution of the program.
- A subroutine is a logical collection of instructions that is invoked from within a larger program to perform a specific task.
- The subroutine is relatively independent of the remaining statements of the program that invokes it & can be invoked several times from several places during a single execution.
- After completing the specific task, a subroutine returns to the point of invocation in the larger program.

PSEUDO CODE & FLOW CHART

- Like step-form, Pseudo-code is a written statement of an algorithm using a restricted and well-defined vocabulary.
- A flowchart comprises of a set of standard shaped boxes that are interconnected by flow lines to represent an algorithm.
- There should be a logical start and stop to the flowchart.
- The usual direction of the flow of a procedure or system is from left to right or top to bottom.
- The intersection of flow lines should be avoided.
- Flowcharts facilitate communication between programmers and users.

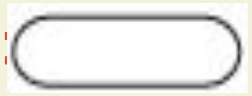
EXAMPLE: PSEUDO CODE

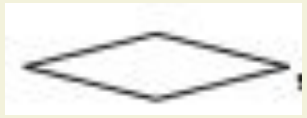
- **Problem:**
  - Write an algorithm to find out whether a given number is a prime number or not.
- **Solution:**
  - The algorithm for checking whether a given number is a prime number or not is as follows:



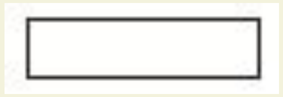
```
1.  START
2.  PRINT "ENTER THE NUMBER"
3.  INPUT N
4.  IF N = 2 THEN
      PRINT "CO-PRIME" GOTO STEP 12
5.  D ← 2
6.  Q ← N/D (Integer division)
7.  R ← N – Q*D
8.  IF R = 0 THEN GOTO STEP 11
9.  D ← D + 1
10. IF D <= N/2 THEN GOTO STEP 6
11. IF R = 0 THEN
      PRINT "NOT PRIME"
    ELSE
      PRINT "PRIME"
12. STOP
```

FLOW CHARTS : SYMBOLIC REPRESENTATION

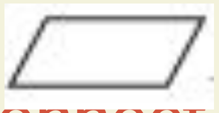
- The START and STOP are represented by an ellipse-like figure : 
- Decisions construct by the rhombus-like figure :



- The processes by rectangles :

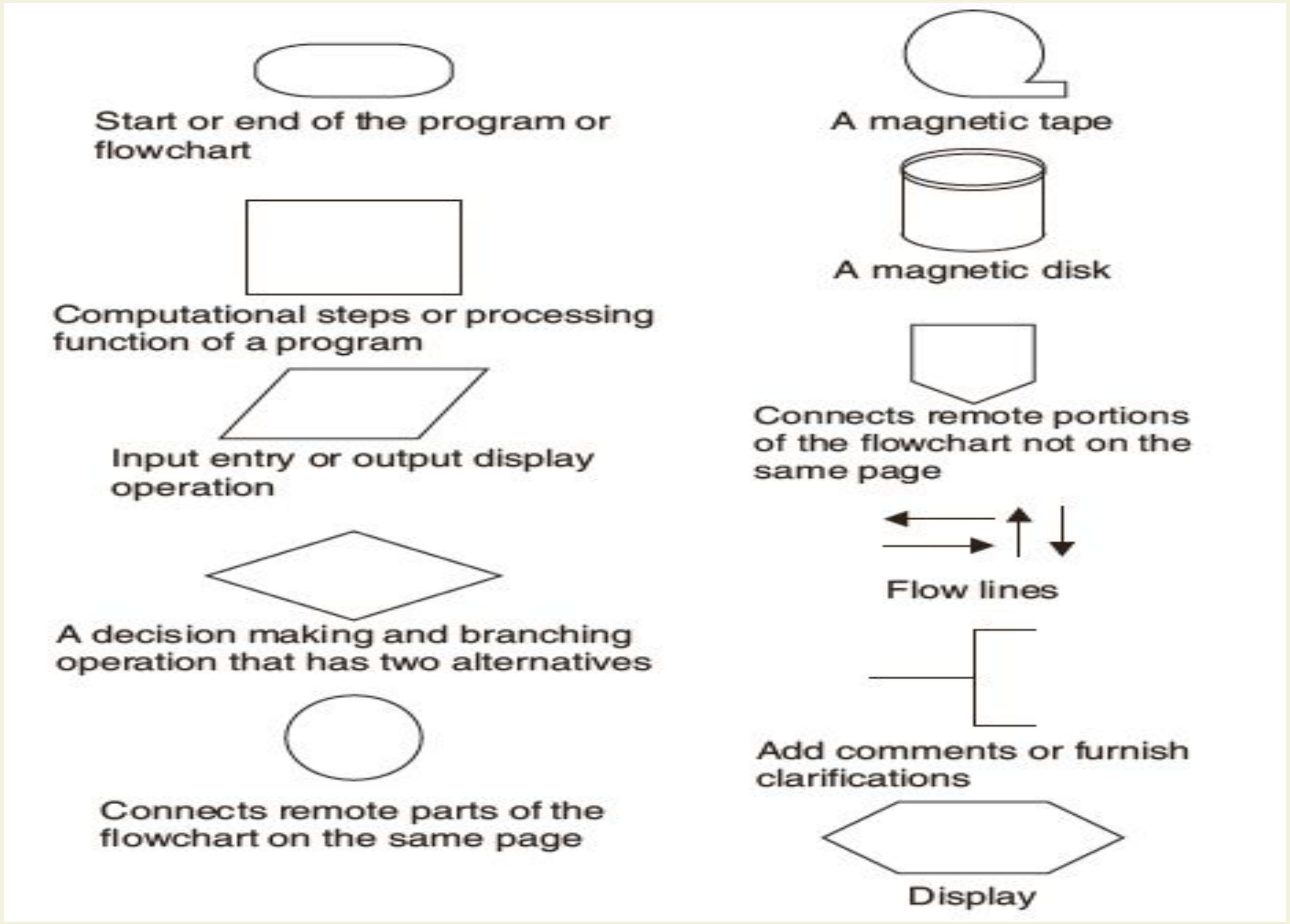


- Input / Output by parallelograms :



- Lines and arrows connect these blocks.

FLOW CHARTS:SYMBOLIC REPRESENTATION


Start or end of the program or flowchart
A magnetic tape
A magnetic disk
Computational steps or processing function of a program
Input entry or output display operation
Connects remote portions of the flowchart not on the same page
Flow lines
A decision making and branching operation that has two alternatives
Add comments or furnish clarifications
Connects remote parts of the flowchart on the same page
Display

FLOW-CHART ADVANTAGES

- *Communication*
- *Effective analysis*
- *Proper documentation*
- *Efficient coding*
- *Proper debugging*
- *Efficient program maintenance*

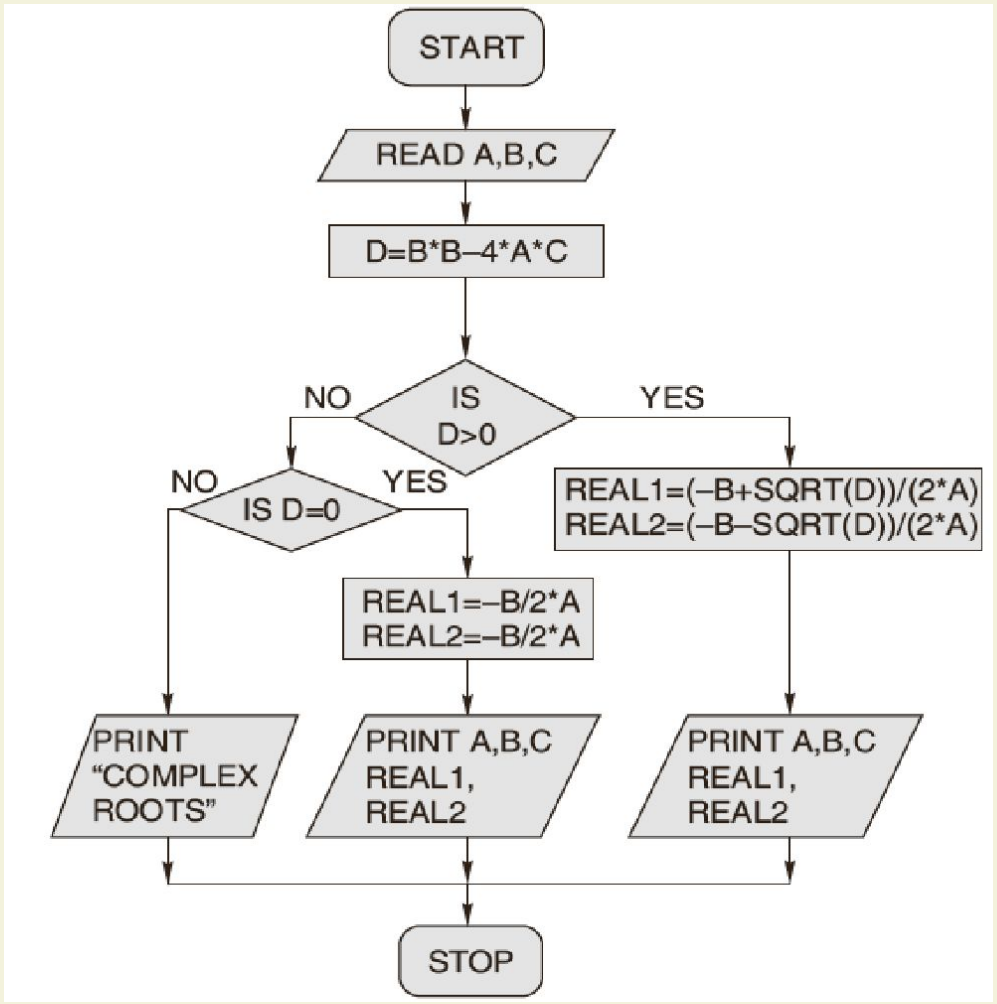
FLOW-CHARTS LIMITATIONS

- *Complex logic*
- *Alterations and modifications*
- *Reproduction*
- *Loss of objective*

FLOWCHART EXAMPLE

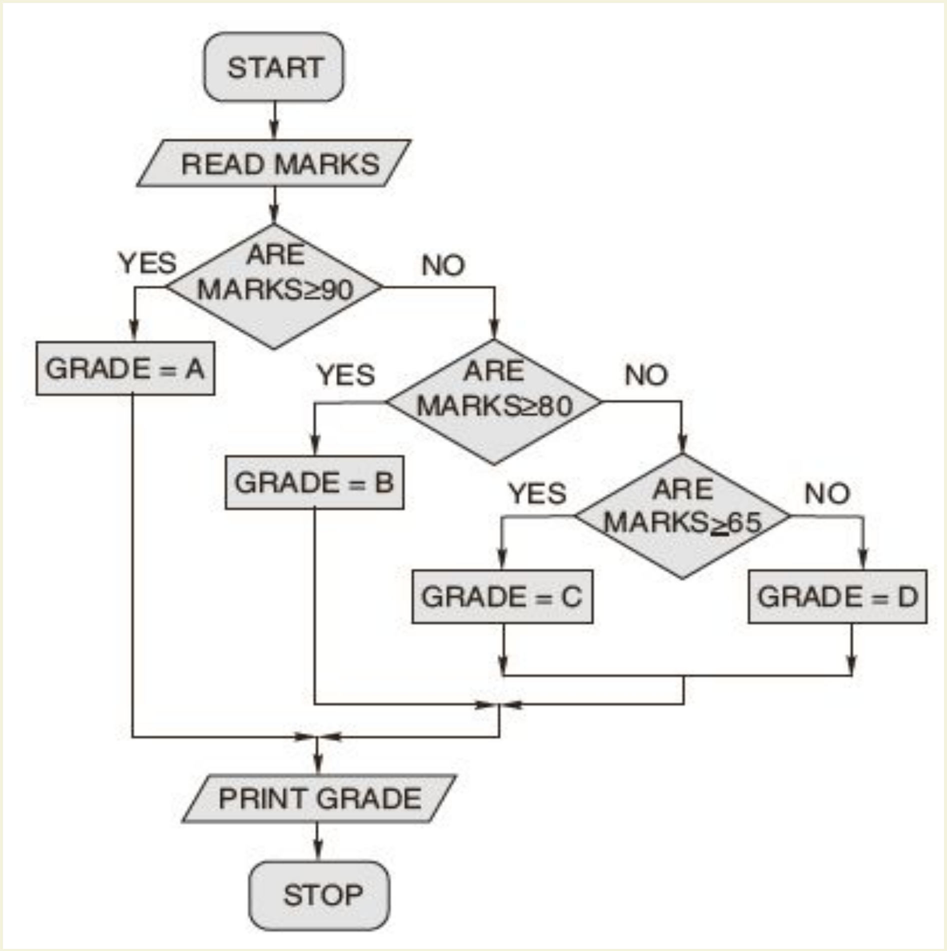
- **Problem:**

  Draw a flowchart to find the roots of a quadratic equation.



FLOW-CHART EXAMPLE

- **Problem:**
  - Prepare a flowchart to read the marks of a student and classify them into different grades. If the marks secured are greater than or equal to 90, the student is awarded Grade *A; if they are greater* than or equal to 80 but less than 90, Grade *B is awarded; if they* are greater than or equal to 65 but less than 80, Grade *C is* awarded; otherwise Grade *D is awarded.*



ALGORITHM : DESIGNING STATEGY

1. Identify the outputs needed.
2. Identify the input variables available.
3. Identify the major decisions and conditions.
4. Identify the processes required to transform inputs into required outputs.
5. Identify the environment available.

TOP DOWN DEVELOPMENT STEP

- The top-down development phase plans out the way the solution has to be done by breaking it into smaller modules and establishing a logical connection among them.
- ***Stepwise refinement :***

a. Work out each and every detail for each small piece of manageable solution procedure.

b. Decompose any solution procedure into further smaller pieces and iterate until the desired level of detail is achieved.

CONT.

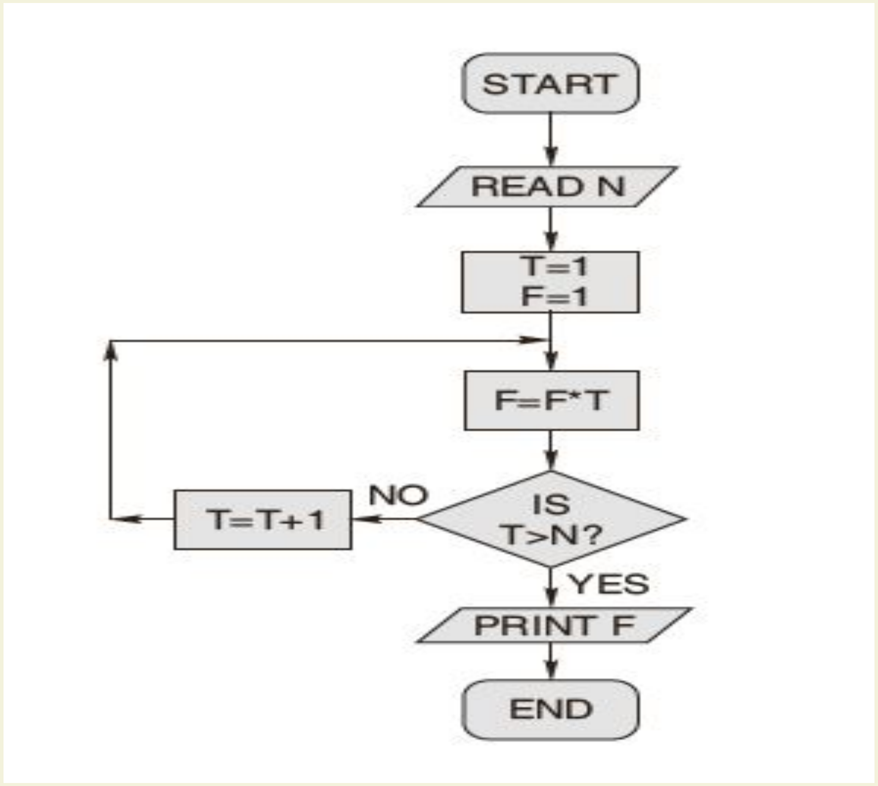
c. Group processes together which have some commonality.

d. Group variables together which have some appropriate commonality.

e. Test each small procedure for its detail and correctness and its interfacing with the other small procedures.

TRACING AN ALGORITHM TO DEPICT LOGIC

- An algorithm can be traced by verifying every procedure one by one to determine and confirm the corresponding result that is to be obtained.
- Example:



CONVERSION

- **Specification for Converting Algorithms into Programs:**

  The general procedure to convert an algorithm into a program is to code the algorithm using a suitable programming language, check the program code by employing the desk-check method and finally evaluate and modify the program, if needed.