(MCA-III) FUNDAMENTAL OF COMPUTERS & EMERGING TECHNOLOGY

Unit - 1

Introduction to Computer

➤ Definition :- A computer is an electronic device that accepts data as input, processes it according to a set of instructions (program), stores the data and results, and produces output in a useful form.

➤ Computer Hardware :- Computer hardware refers to the physical components of a computer system — the parts you can see and touch.

Examples:
- Input Devices : Keyboard, mouse, scanner
- Output Devices : Monitor, Printer, speaker
- Storage Devices : Hard Disk, Pen Drive
- Processing Unit : CPU, motherboard, RAM

➤ Computer Software :- Software is a collection of programs, instructions, and data that tell the computer how to perform specific tasks. It is intangible — you cannot touch it.

Examples : Windows, MS Office, chrome, Python, etc.

➤ Characteristics of Computer :-

(i) Speed (ii) Accuracy (iii) Automation
(iv) Storage capacity (v) Versatility (vi) Diligence
(vii) Communication (viii) Multitasking (ix) Reliability
(x) Programmable

(c) Basic Operations of a Computer: A computer can perform five basic operations that together form the Information Processing Cycle (IPC)

Input → Processing → Storage → Output → Control

(i) Input Operation – The process of entering raw data and instructions into the computer system for processing.
- Purpose: To provide data to the computer
- Devices Used: Keyboard, Mouse, Scanner, etc.

(ii) Processing Operation – The computer's CPU processes the input data using a set of instructions to generate meaningful information.

- Main Components of CPU:
  (a) ALU (Arithmetic Logic Unit)
  (b) CU (Control Unit)
  (c) Registers (Store temporary data during processing)

(iii) Storage Operation – Used to store data, instructions and results for immediate or future use.

- Types: (a) Primary Storage: RAM, ROM
  (b) Secondary Storage: Hard Disk, Pen Drive

(iv) Output Operation – The process of displaying or producing the processed information in a usable form.

- Devices Used: Monitor, Printer, Speaker, etc.
- Output can be in visual, printed or audio form.

(v) Control Operation — The control unit (CU) manages and directs all other components of the computer. It ensures that input, processing, storage, and output operations occur in proper sequence.

Components

- Hardware Components: Hardware components are the physical parts of a computer that are tangible means that we can see and touch. "All the operations of a computer are performed using these hardware components under the control of software."

Think of Hardware as the body of a computer, providing the structural foundation, while software acts as the brain, instructing it on what to do.

Hardware components are broadly classified into:

- Input Devices
- Output Devices
- Processing Devices
- Storage Devices
- Communication Devices

Input Devices : Input devices are hardware components used to enter data and instructions into the computer system for processing.

Functions :
- Convert user data into machine-readable form.
- Send data to the CPU for processing.

Common Input Devices :
- Keyboard - used for typing text, nos, and commands.
- Mouse - Pointing and clicking to interact with the computer.
- Scanner - converts physical documents into digital format
- Microphone - Input audio or voice commands.
- Camera - Captures images and videos for processing.

↳ Output Devices :- Output devices are hardware components that present the processed information to the user in a human-understandable form.

Functions : • Convert machine-processed data into readable or perceivable form.

Common Output Devices :

- Monitor - Display text, images, and videos on screen
- Printer - Produces hard copies of documents
- Speaker - Produces audio output like music or alerts
- Projector - Displays information on a large screen for presentations.

↳ Processing Devices :- Processing devices are hardware components that process data according to instructions provided by software.

Main Component : CPU (central Processing Unit) - the brain of the computer

Parts of CPU :

- ALU (Arithmetic logic Unit)
- Control Unit (CU)
- Registers

➤ Storage Devices :- Storage devices are hardware components used to store data, programs, and results, either temporarily or permanently

Types of Storage :

(a) Primary storage (Volatile / Temporary)
- RAM (Random Access Memory) :- stores data temporarily while the computer is running.
- ROM (Read Only Memory) :- stores permanent instructions like BIOS.

b) Secondary Storage (Permanent)
- Hard Disk Drive (HDD) - long term storage for files and programs
- Solid State Drive (SSD) - Faster permanent storage
- Pendrive / USB Drive / CD / DVD - Portable storge

Function : Memory stores data, programs, and results for immediate or future use.

➤ Communication Devices :- Communication devices are hardware components that enable computers to communicate with other system or networks.

Functions : • Send and receive data over networks or the internet.

Example : • Modem
• NIC
• Wi-Fi Dolapter

Software Components :- Software components are the intangible parts of a computer system — programs, instructions, and data — that tell the hardware how to perform tasks.
Unlike hardware, software cannot be touched, but it is essential for the computer to function.

Software is broadly divided into three main types : (i) System Software
(ii) Application Software
(iii) Programming Software

System Software :- System software is designed to manage and control the hardware and provide a platform for running application software.

Functions :
- Controls computer hardware
- Provides a user interface
- Manage files and system resources

Examples :-
- Operating Systems (OS): Windows, Linux, macOS Controls all system processes.
- Utility Programs : Antivirus, Disk Cleanup, Backup tools. Help maintain the computer and improve performance.

Application Software :- Application software is designed to perform specific tasks for the user.

Functions : Helps the user perform tasks like browsing, writing, calculating etc.

Examples :- Office Application, Web Browser, Media Players etc.

↳ Programming Software :- Programming software provides tools for developers to write, test and debug programs.

Functions :
- Helps in creating new software and applications
- Converts human-readable code into machine code.

Example :
- Programming languages
- IDEs
- Compilers & Interpreters

Computer languages

↳ Definition of Computer languages :- Computer languages are formal languages used to communicate instructions to a computer.
They allow humans to write programs that the computer can understand and execute.

Computers only understand binary code (0s and 1s) computer languages provide a way for humans to write instructions in a more understand-able form, which is then translated into machine code.

Types of Computer Languages : Computer languages are broadly classified into three categories.

(i) Machine language : It is the lowest-level language and consists of binary code (0s and 1s) that the computers directly understands.
Ex : 10101010 10010010

(ii) Assembly language : Assembly language is a low-level language that uses mnemonics instead of binary code. Easier for humans to read than machine language.
Ex : MOV A, B (Move data from B to A)

(iii) High-level language : HLL are closer to human language & easier to read, write, and understand.

Requires a compiler or interpreter to convert code into machine language.

Ex : Python, C, C++, Java, etc.

(iv) Fourth-Generation languages (4GL) : These are even higher-level languages designed to be closer to natural language and simplify programming.

Focused on database queries, report generation and business applications.

Ex : SQL, MATLAB, Oracle Reports

(e) Concept of Compiler :- A compiler is a special program that translates the entire "source code" written in a high-level programming language into "machine code" or object code before execution.

It allows humans to write programs in high-level languages (like C, C++, Java) that are easier to understand.
Computer can only understand machine language, so the compiler performs the translation.

- Characteristics of a Compiler :

(i) Translates high-level language code into machine language.

(ii) Produces an executable file that can be run on the computer.

(iii) Performs syntax checking and error detection during compilation.

(iv) The translation is done once, and the program can be executed multiple times.

- Examples of compiled languages :
  - C, C++
  - Java
  - Go, Rust

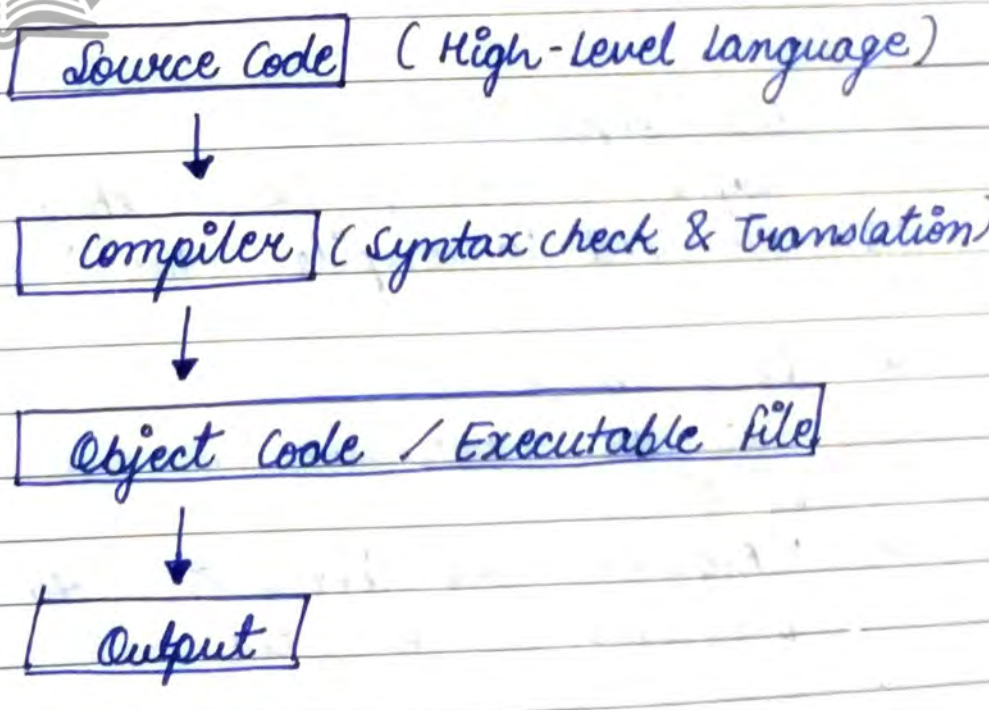
- **Working of a Compiler :**

(i) **Source Code** : The program written by the programmer in a high-level language.
Example : `int sum = a+b;`

(ii) **Compilation** : The compiler checks the syntax and semantics, then converts it into machine code.

(iii) **Object code / Executable file** : The resulting machine-level code that the computer can execute.

(iv) **Execution** : The computer runs the executable file to produce output.



- **Advantages of Compiler :**

(i) High execution speed (entire program is translated once).
(ii) Detects errors before execution
(iii) Produce standalone execution file.

(e) ▶ Interpreter :- An interpreter translates and executes a high-level language program line by line, instead of converting the entire program into machine code at once.

- Characteristics of an Interpreter :-

(i) Reads the source code line by line.

(ii) Executes each line immediately after translation.

(iii) Stops execution if it encounters an error.

(iv) Does not create a separate executable file.

(v) Useful for scripting languages or programs in development / testing.

- Advantages :

(i) Useful for testing and debugging programs

(ii) No need to compile the entire program

- Disadvantages :

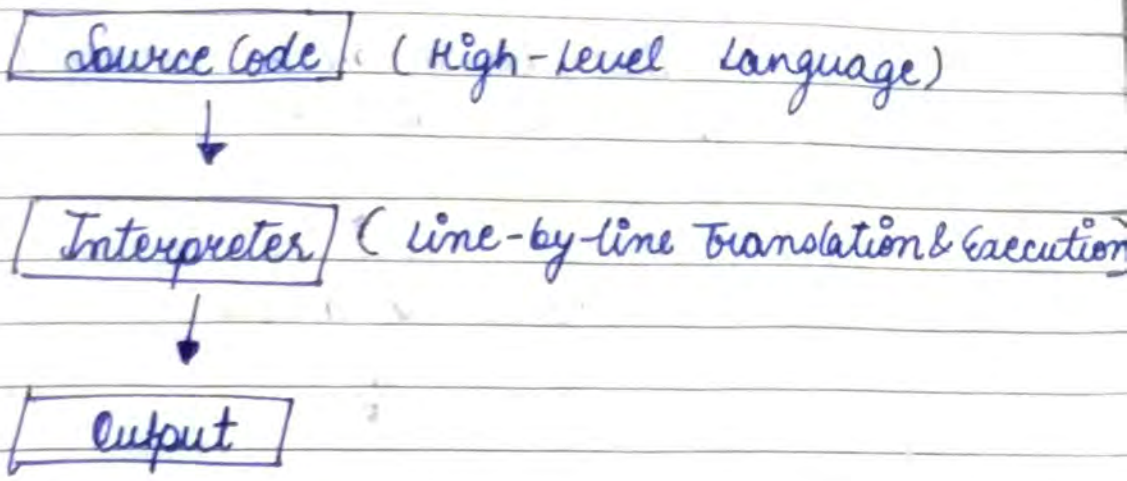
(i) Slower execution because translation occurs line by line.

ii) Requires the interpreter to run everytime the programs is executed.

- Examples of interpreted languges :-

- Python
- JavaScript
- Ruby
- BASIC

- Working of an Interpreter



(e) Assembler :- An assembler is a program that translates assembly language programs (low-level mnemonic codes) into machine code that the computer can execute.

- Characteristics of an Assembler :

(i) Converts mnemonics (like MOV, ADD) into machine instructions.
(ii) Produces object code or machine code.
(iii) Requires an assembler before the program can run on hardware.
(iv) Works at a low-level language close to hardware.

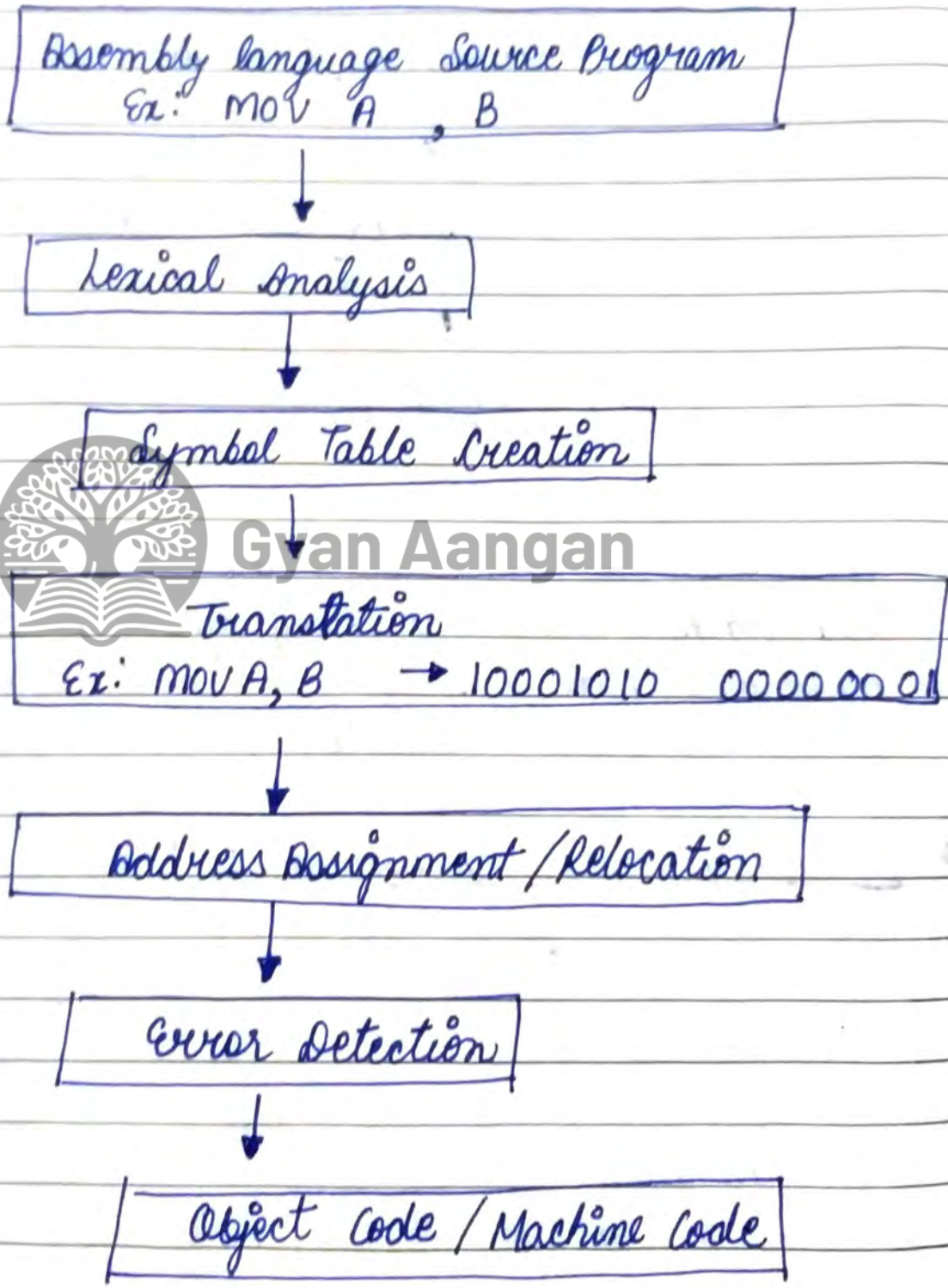
- Advantages :

(i) Faster execution because machine code is produced.
(ii) Allows precise control over hardware resources

- Disadvantages :

(i) Programming in assembly language is difficult & time taking
(ii) Not portable in different hardware architecture

- Working of an Assembler :



Q. Compiler vs Interpreter vs Assembler

Feature	Compiler	Interpreter	Assembler
Definition	Translates the entire High-level source code into machine code before execution	Translates & execute high-level source code line-by-line	translate assembly language (mnemonics) into machine code
Input	HLL (C, C++, Java)	HLL (Python, JavaScript)	Assembly Language
Output	Machine code or Executable file	Immediate execution no separate file.	Machine code or object file
Execution	After full compilation	Line by line	After assembly
Errors Detection	All errors are detected before execution	Stops execution at first error...	Errors detected during assembly
speed of Execution	Fast, as entire program is already translated	Slower, translate line by line during execution.	Fast, once assembled into machine code
File Creation	Yes, a standalone executable file	NO	Yes, object / machine code file
Usage	Large programs where efficiency matters.	For scripting, testing & debugging	Low-level hardware programming.
Example	GCC, JavaC	Python Interpreter	MASM, NASM, TASM

Problem Solving Concept

Definition of Problem Solving :- Problem Solving is the process of analyzing a problem, finding possible solutions, and implementing the most efficient one using logical and systematic steps.

In computer science, it means developing algorithms, and programs to make computers solve real-world problems.

- Steps in Problem Solving Process :-

1.) Problem Definition/Understanding - clearly define what the problem is and what the desired output should be. Identify the input, processing, and output.

2.) Problem Analysis - Break the problem into smaller parts. Determine constraints, data requirements & relationships between components.

3.) Designing the Solution - Create a plan or algorithm that describes the step-by-step solution to the problem. You can use flowcharts / pseudocodes.

4.) Coding / Implementation - Translate the designed algorithm into a programming language.

5.) Testing and Debugging

6.) Documentation

7.) Maintainance

- **Techniques used in Problem solving :**

(i) **Algorithm** - A step-by-step method to solve a specific problem.

(ii) **Flowchart** - A diagrammatic representation of algorithm using symbols.

(iii) **Pseudocode** - A plain English like representation of the logic before actual coding.

(iv) **Modularization** - Dividing a big problem into smaller, manageable sub-problems (modules)

- Algorithm : An algorithm is a finite sequence of well-defined steps that provides a solution to a particular problem.

In simple terms, Algorithm = Step-by-step instructions to solve a problem

- **Example :** Problem : Find the sum of two numbers.
  Algorithm : 1.) Start
  2.) Input two numbers a and b
  3.) Add a and b and store the result in sum
  4.) Display sum
  5.) Stop

- **Characteristics of a Good Algorithm :**

(i) **Finiteness** : The algorithm must always finish after a finite number of steps.

(ii) **Definiteness** : Each step must be clearly and unambiguously defined.

(iii) Input : An algorithm should have zero or more inputs.

(iv) Output : An algorithm should produce atleast one output

(v) Effectiveness : Each step must be simple enough to be performed exactly and infinite time.

(vi) Generality : The algorithm should be applicable to a class of problems, not just one instance.

- Types of Algorithms :

(i) Sequential Algorithm - executes steps one after another in order. Ex- Sum of nos

(ii) Conditional Algorithms - Includes decision-making steps (IF-THEN-ELSE). Ex- Finding largest of two nos.

(iii) Iterative / Looping Algorithms - Uses repetition (loops). ex - Printing 1 to 10 using loop

(iv) Recursive Algorithms - Calls itself repeatedly until a condition is met. Ex- Finding factorial.

(v) Searching Algorithms - Used to find data. ex- Linear, Binary search

(vi) Sorting Algorithm - Used to arrange data. ex- Bubble sort, Quick Sort.

- **Advantages of Algorithms:**
  (i) Easy to understand & implement
  (ii) Helps in error detection before coding
  (iii) Serves as a blueprint for writing programs

- **Representation Of Algorithm:** Algorithms are usually represented using:
  (i) Pseudocode - written in plain English statements
  (ii) Flowcharts - visual representation of steps using symbols.

- **Limitations of an Algorithm:**

(i) Requires well-defined Problem, vague or undefined problems cannot be solved
(ii) Time consuming to design
(iii) Not suitable for complex problem
(iv) Efficiency issues
(v) Limited to Logical Problems
(vi) Debugging and Testing Complexity
(vii) Space complexity
(viii) Hardware dependence

(a) Conditions in Pseudocode :- First of all let us know what is pseudo-code, so a Pseudo-code is a simplified, informal way of describing an algorithm that is easier to read and understand than actual programming code.

Now, let's define conditions in Pseudocode.

"A condition is pseudocode is a logical statement that checks whether something is true or false and allows the program to make decision accordingly."

In simpler terms, conditions controls the flow of logic in a program - they decide what to do next based on certain tests or comparisons.

- Conditional statements Used in Pseudocode :-

(i) IF statement - executes a block of statements only if a condition is True.
Syntax : IF condition THEN statement ENDIF

(ii) IF... ELSE statement - executes one block if the condition is true, otherwise another block
Syntax : IF condition THEN statement ELSE statement2
ENDIF

(iii) IF... ELSE IF... ELSE statement - used to check multiple conditions one after another

Syntax - IF condition1 THEN statement1 ELSEIF condition2 THEN statement2 ELSE statement3 ENDIF

(iv) Nested IF statement - An IF statement inside another IF statement. Used for complex decision.

Syntax : IF condition1 THEN IF condition2 THEN statement
ENDIF ENDIF

(v) CASE / SWITCH statement - select one block of statements from multiple options based on a variable's value.

- Common Relational Operators used in Conditions :

Operator	Meaning	Example
=	Equal to	IF a=b THEN
<> or !=	Not Equal to	IF a!=b THEN
>	Greater than	IF a > b THEN
<	Less than	IF a < b THEN
>=	Greater than or equal to	IF a >= b THEN
<=	Less than or equal to	IF a <=b THEN

- Example :

```
IF marks >= 40 THEN
    PRINT "Pass"
ELSE
    PRINT "Fail"
ENDIF
```

(e) Loops in Pseudocode : A loop in pseudocode is a control structure that allows a set of instructions to be repeated multiple times until a certain condition is met.

- Purpose of Loops :
  - To execute statements repeatedly
  - To save time and code
  - To automate repetitive tasks

- Types of Loops in Pseudocode :

i) WHILE Loop : repeats a block as long as a condition is TRUE. The condition is checked 'before' entering the loop.

ii) REPEAT... UNTIL Loop : repeats the block until a condition becomes true. The condition is checked after executing the loop body (runs atleast once)

iii) FOR loop - repeats a block a fixed number of times, using a counter variable.

Examples : 1.) WHILE LOOP

```
SET i = 1
WHILE i <= 5 DO
    PRINT i
    SET i = i + 1
ENDWHILE
```

2.) REPEAT... UNTIL loop

```
SET i = 1
REPEAT
    PRINT i
    SET i = i + 1
UNTIL i > 5
```

3.) FOR loop

```
FOR i = 1 TO 5 DO
    Print i
NEXT i
```

- Importance of loops :- i) Reduces code length
  ii) Increases efficiency
  iii) Helps in automating repetitive tasks.