SLIIT ACADEMY

Higher Diploma in Information Technology
Year 1, Semester 1

Introduction to Programming(C++)

Lecture 01:An Overview of Programming Languages

Intended Learning Outcomes

On the Completion of this lecture student will be able to learn ,

LO1:Understand the use of programming languages

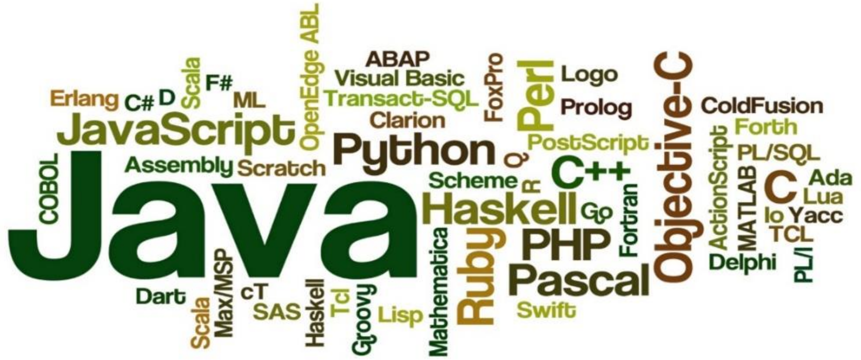
LO2:Discuss different generations of computer languages.

LO3:Discuss the role of translators

LO4:Comparison of different levels/generations of languages.

What is a Programming Language?

- A programming language is a notational system for describing computation in machine-readable and human-readable form.
- Programming languages are artificial languages created to tell the computer what to do.



Programming Language Elements

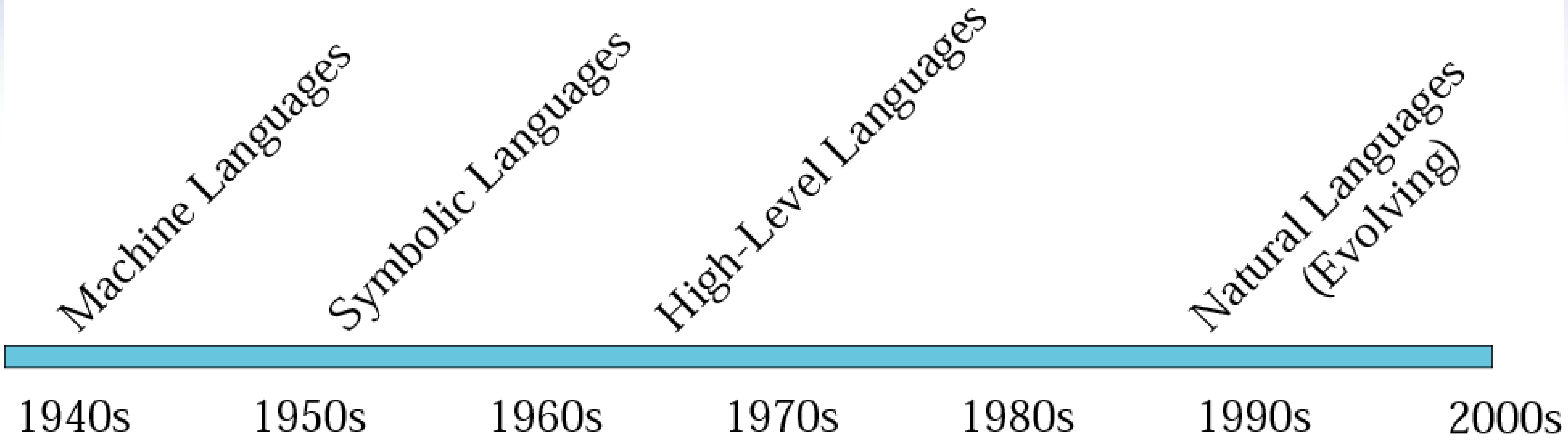
- **Vocabulary:** Set of all of the words and symbols in the language.
- **Syntax:** The form or structure of the expressions ,statements and program units. (Rules of combining statements)

  Example : Use of Keywords , Operators

- **Semantics:** The meaning of the expressions ,statements and program units. (Rules for interpreting statements)

  Example : Order of Precedence

Evolution of Computer Languages

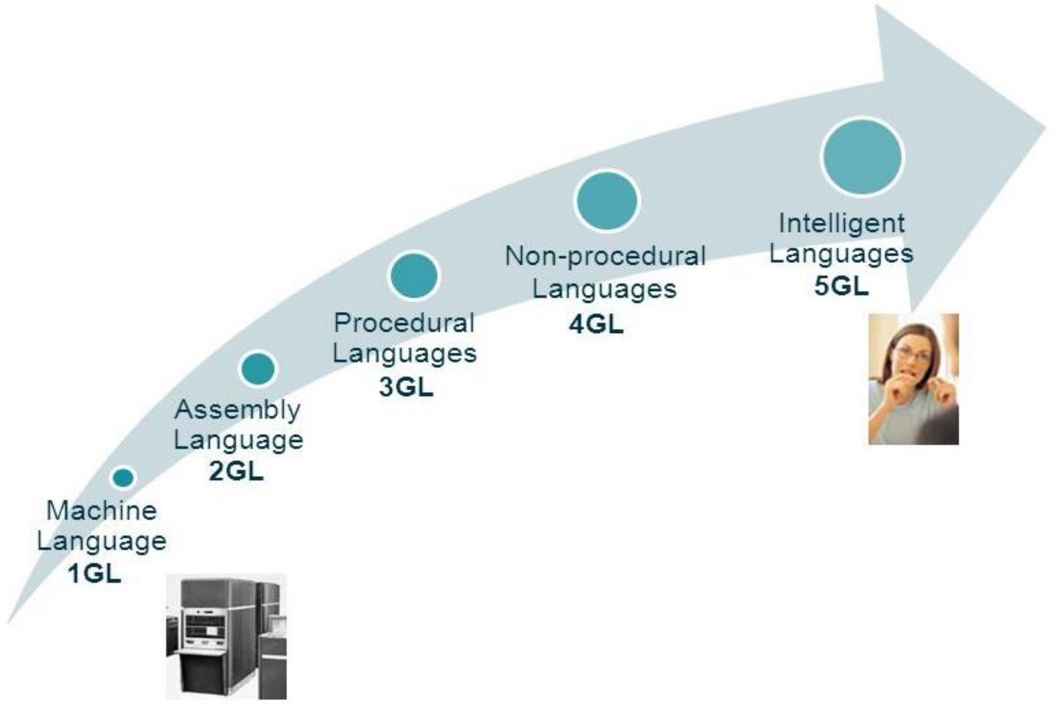


Classification of Programming Languages

- Languages are grouped into **generations**

  Based on when they were first conceived.

- There are five such generations.



Classification of Programming Languages

- Languages are grouped by **levels**

  Based on how much they are close to the language the computer itself uses, called machine language.

- Classified as :
  - Low level
  - High-level
  - Very High level

Machine Language- 1st Generation

- **Machine languages** are the most basic type of computer languages, consisting of strings of numbers the computer's hardware can use.
- Different types of hardware use different machine code.

Example: IBM computers use different machine language than Apple computers.

Machine Language =Instruction set + Rules about how the instruction can be combined

A Machine Language program to add two numbers and store the sum in a third location.

000001	1000
111111	1001
101010	1002

Assembly Language- 2nd Generation

- **Assembly languages** are only somewhat easier to work with than machine languages.
- To create programs in assembly language, developers use **cryptic English-like phrases** to represent strings of numbers.
- Each numeric instruction is assigned a short name (called a mnemonic) that is easier to remember than a number.

Example : 28 – LOAD   69 - ADD

```
LDA    x
ADD    Y
STM    Z
```

- Assembly Language Program can assign names to memory locations

Example: Instead of the memory address 1000, use the word SALARY.

Program Example

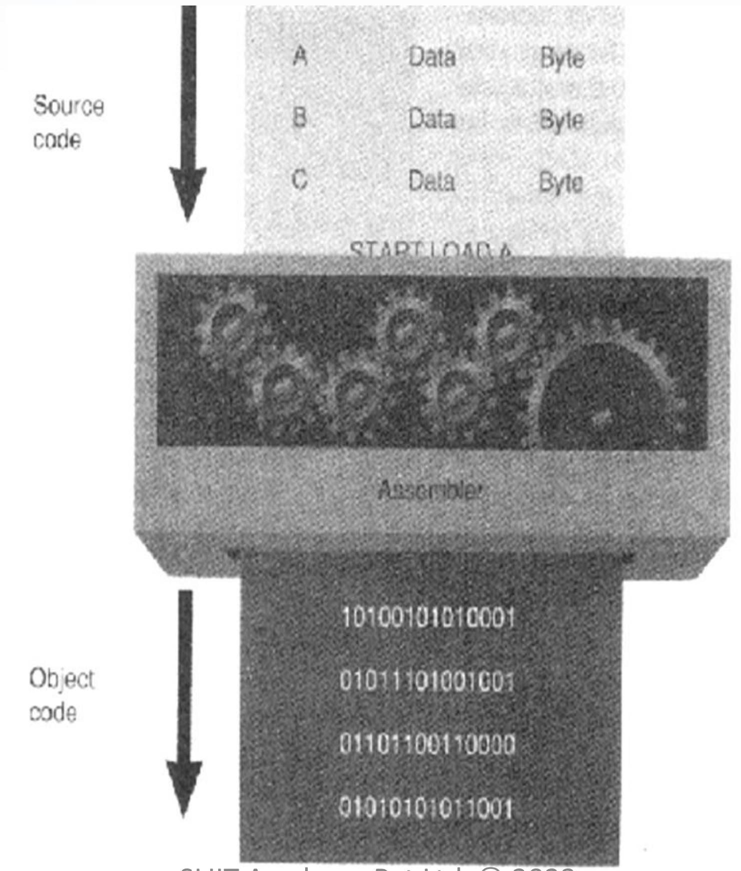
Machine language	Assembly language
1st generation	*2nd generation*

```
156C        LD R5, Price
166D        LD R6, ShippingCharge
5056        ADDI R0, R5 R6
30CE        ST R0, TotalCost
C000        HLT
```

Assembler

- A program called an assembler translates Assembly Language into machine language.

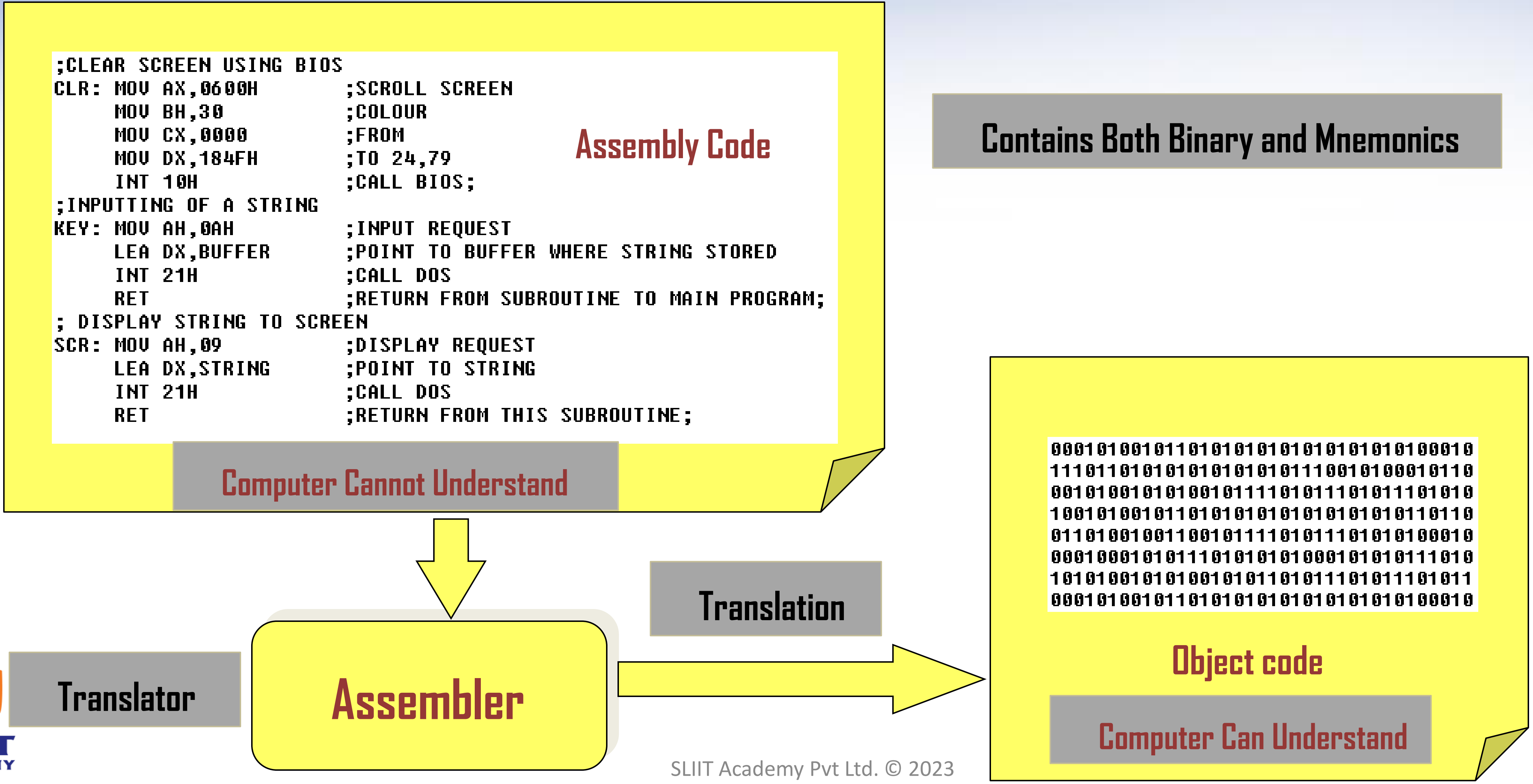
Source code



Object code

Assembly languages are specific to the type of processor.

Assembler



```
;CLEAR SCREEN USING BIOS
CLR: MOV AX,0600H       ;SCROLL SCREEN
     MOV BH,30          ;COLOUR
     MOV CX,0000        ;FROM
     MOV DX,184FH       ;TO 24,79
     INT 10H            ;CALL BIOS;
;INPUTTING OF A STRING
KEY: MOV AH,0AH         ;INPUT REQUEST
     LEA DX,BUFFER      ;POINT TO BUFFER WHERE STRING STORED
     INT 21H            ;CALL DOS
     RET                ;RETURN FROM SUBROUTINE TO MAIN PROGRAM;
; DISPLAY STRING TO SCREEN
SCR: MOV AH,09          ;DISPLAY REQUEST
     LEA DX,STRING      ;POINT TO STRING
     INT 21H            ;CALL DOS
     RET                ;RETURN FROM THIS SUBROUTINE;
```

```
0001010010110101010101010101010100010
1110110101010101010101110010100010110
0010100101010010111101011101011101010
1001010010110101010101010101010110110
0110100100110010111101011101010100010
0001000101011101010101000101010111010
1010100101010010101101011101011101011
0001010010110101010101010101010100010
```

Procedural Languages–3rd Generation

- **Third-generation languages (3GLs)** are the first to use true English-like phrasing, making them easier to use than previous languages.
- 3GLs are portable, meaning the object code created for one type of system can be translated for use on a different type of system.
- Later in the process, a separate program called a translator handles all the details.
- Independent of hardware; Can use Procedural Languages to write programs for many kinds of computers.

3rd Generation Programming Languages Contd…

- 3rd Generation programming languages are procedural.

Programmer should be specifying a sequence of instructions(procedure),which compiled into machine language before being executed.

Example:

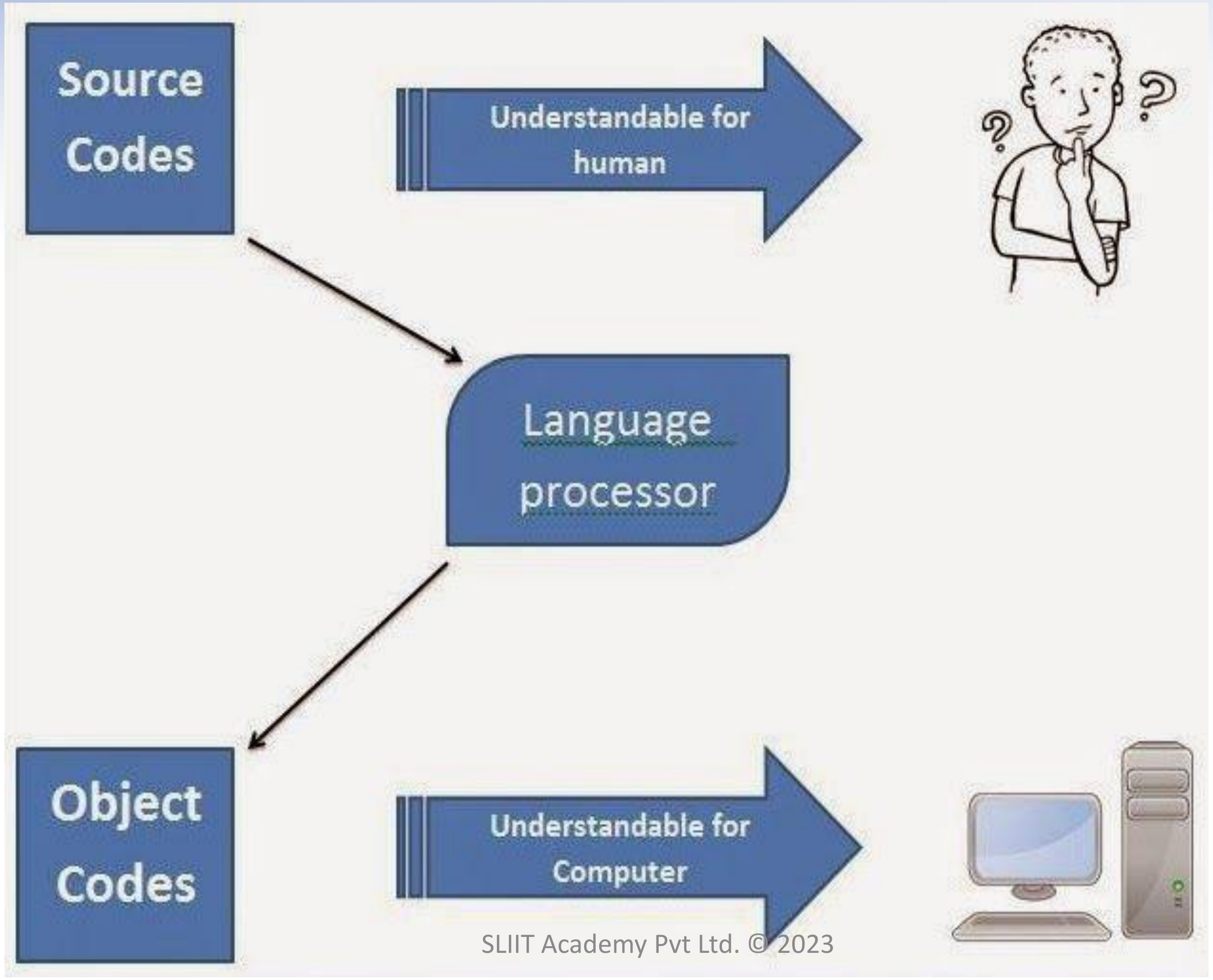
Language	Year	Primary Application
FORTRAN	1954	Science and Engineering
COBOL	1959	Business
BASIC	1965	Education
Pascal	1971	Education
C	1972	General
Visual Basic	1992	General
Java	1996	General

SLIIT ACADEMY

Language Translators

- Language translators convert programming source code into language that the computer processor understands.
- Programming source code has various structures and commands, but the computer processors understand only machine language.
- Kinds of Translators : **Assembler , Compiler, Interpreter.**
- Translator Depending on two factors:
  - **The programming language the program was created with**
  - **The processor it is being translated for.**

Role of Language Translators



Compiler

- Compilers are translators used before a program is run to produce a complete machine language program that is stored on the disk so it can be executed later.
- The entire program is first translated to machine code, and this code is then executed.
- Each language requires its own compiler.

Example: Pascal compiler and a Java compiler

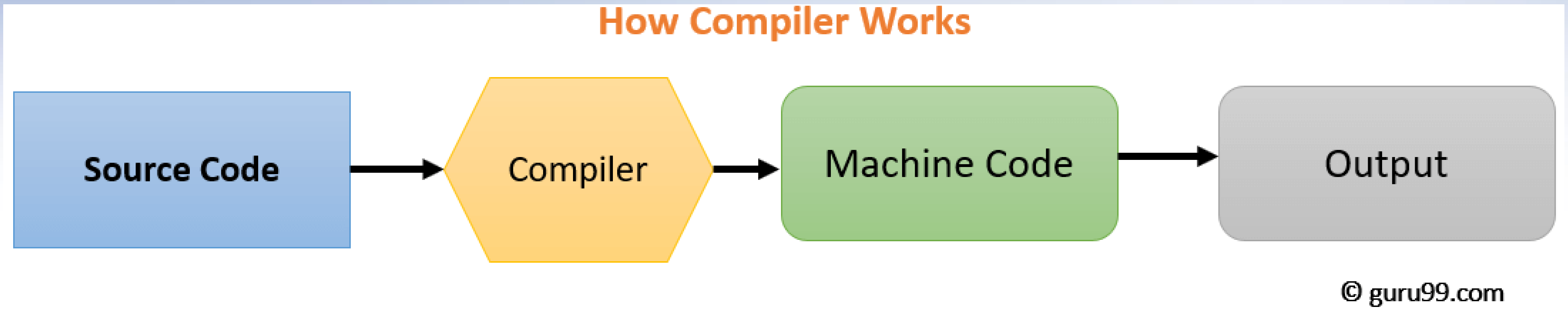
Main Functions of a Compiler

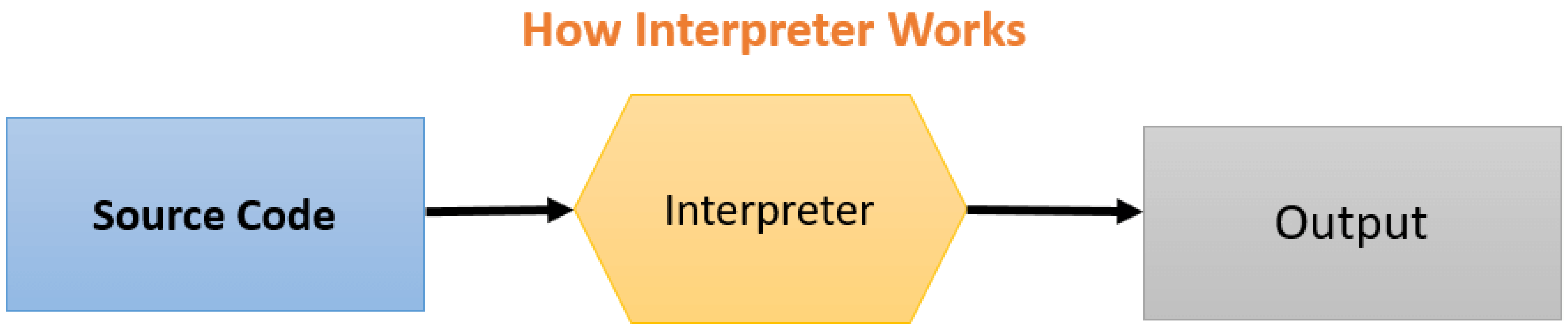
- It checks for syntax errors in a program, for statements which do not conform to the grammatical rules of the language.
- It produces a listing of the program, indicating errors, if any.
- If there are no syntax errors, it generates machine code equivalent to the given program.

Interpreter

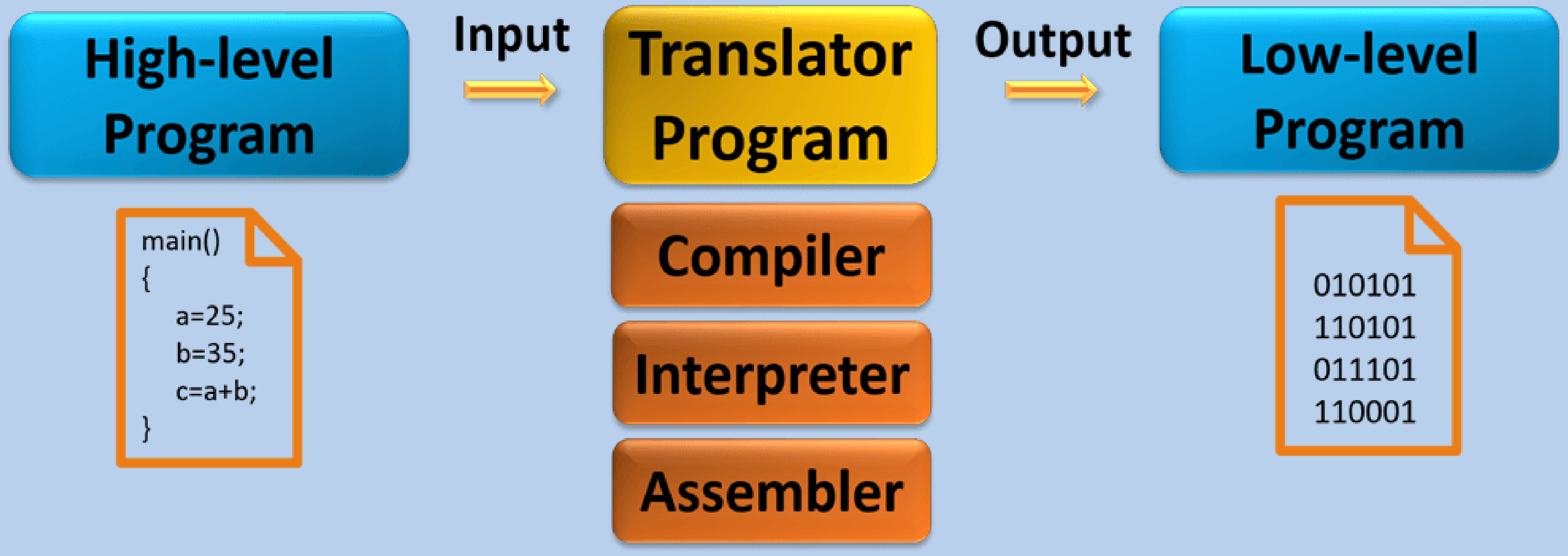
- As each statement of the high-level program is encountered it is translated and executed.
- Translation is done on the fly.
- Interpreted programs often run slower than compiled code.
- No object code is created in the process, so no executable program file is stored on the disk for later use.

Compiler and Interpreter





Translation Process



```
main()
{
    a=25;
    b=35;
    c=a+b;
}
```

```
010101
110101
011101
110001
```

4th Generation Languages

- 4GLs designed to be user friendly and interactive, and help you quickly develop an application package.
- It is known as non-procedural; they concentrate on what you want to do rather how you are going to do it.

Example:

SQL, PostScript, Relational database-oriented languages(ORACLE), VisualAge ,Maple, Mathematica , Python, Ruby

SLIIT ACADEMY

5th Generation

- 5th generation programming language is **based around solving programs using constraints given to the program**, rather than using an algorithm written by a programmer.
- Incorporates the concepts of knowledge-based systems, expert systems, inference engines, and natural language processing.

5th Generation

- In the early 90's when the 5GL came out, it was believed they would become the future of programming. However, after realizing that the most crucial step (defining constraints) still needs human intervention, the initial high expectations were lowered.
- Many constraint-based languages, logic programming languages and some of the declarative languages are identified as 5GL. **Example: Prolog, OPS5 ,Mercury , Lisp**

SLIIT ACADEMY

Language Generation and Level Summary

Language	Generation	Level
Machine Language	First	Low level
Assembly Languages (Mnemonics)	Second	Low level
Procedural Languages (FORTRAN, COBOL, C, C++, Visual Basic, Java)	Third	High level
Query Languages (SQL, Postscript)	Fourth	Very high level
Prolog, OPS5 ,Mercury , Lisp	Fifth	Very high level

SLIIT ACADEMY

Summary

- Programming languages can be categorized into levels and generations.
- The programming language must be first translated into machine language.
- Assembly language is closely related to machine language. It is translated into machine language using an assembler.
- High level languages translated into machine language using a compiler or an interpreter.
- A compiler translates the whole program into machine code and when this is finished, the program is executed. With an Interpreter, the statements are translated and executed one at a time.