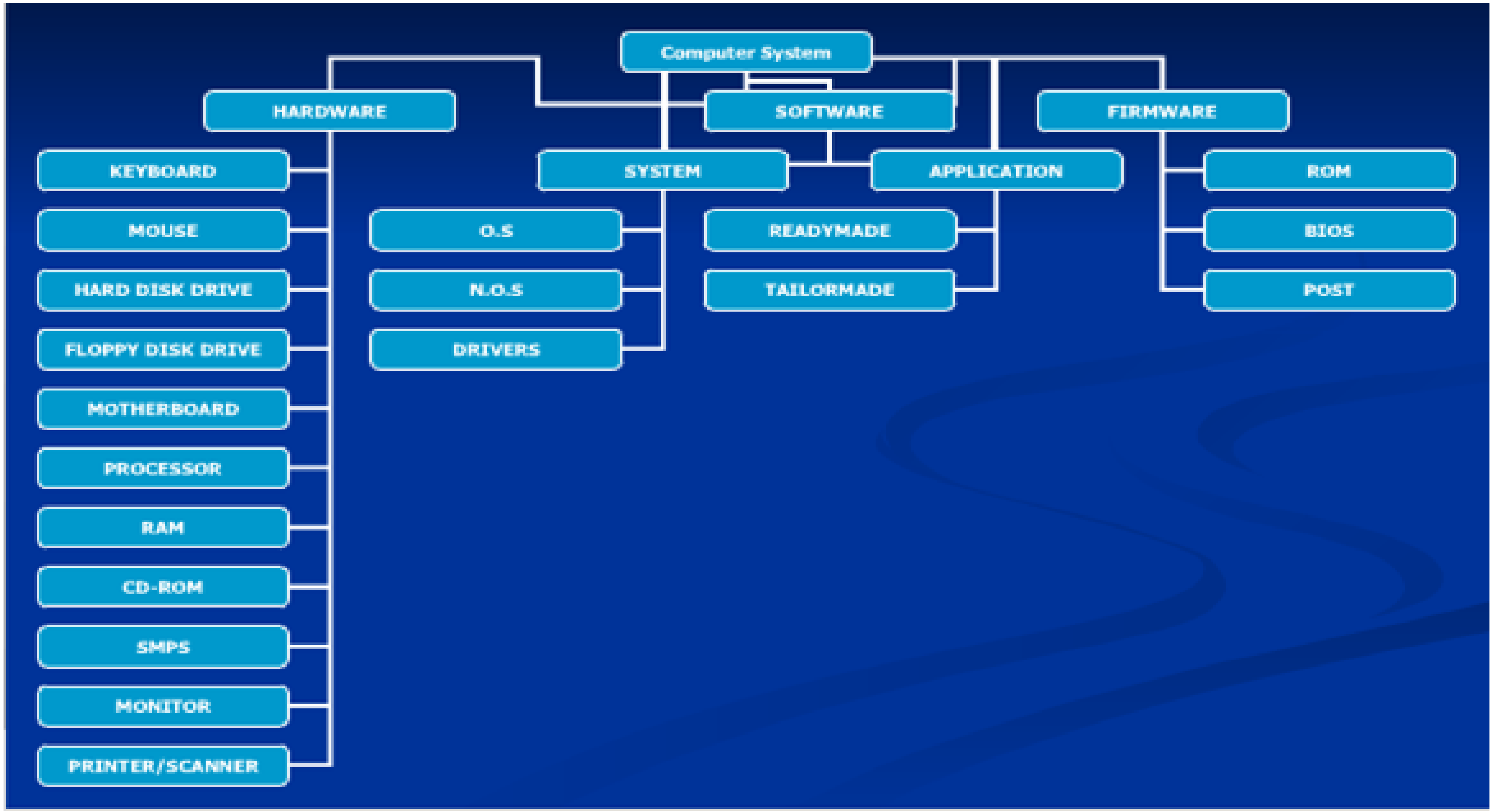
Hardware
Software
Concept of Operating System

Dr. Shweta Sharma
Assistant Professor
Dept. of Computer Engineering
NIT Kurukshetra

Hardware vs. Software

Hardware and Software



Hardware and Software

- ***Hardware***: **Physical components** that form a computer system
- A computer's hardware consists of electronic devices; the parts you can see and touch

Example:

Monitor, Mouse, CPU, RAM, secondary storage devices like CD, DVD, hard disk, Printer, etc.

- ***Software*****:** Software is a **set of instructions** in the form of programs, which control the sequence of operations (tasks)

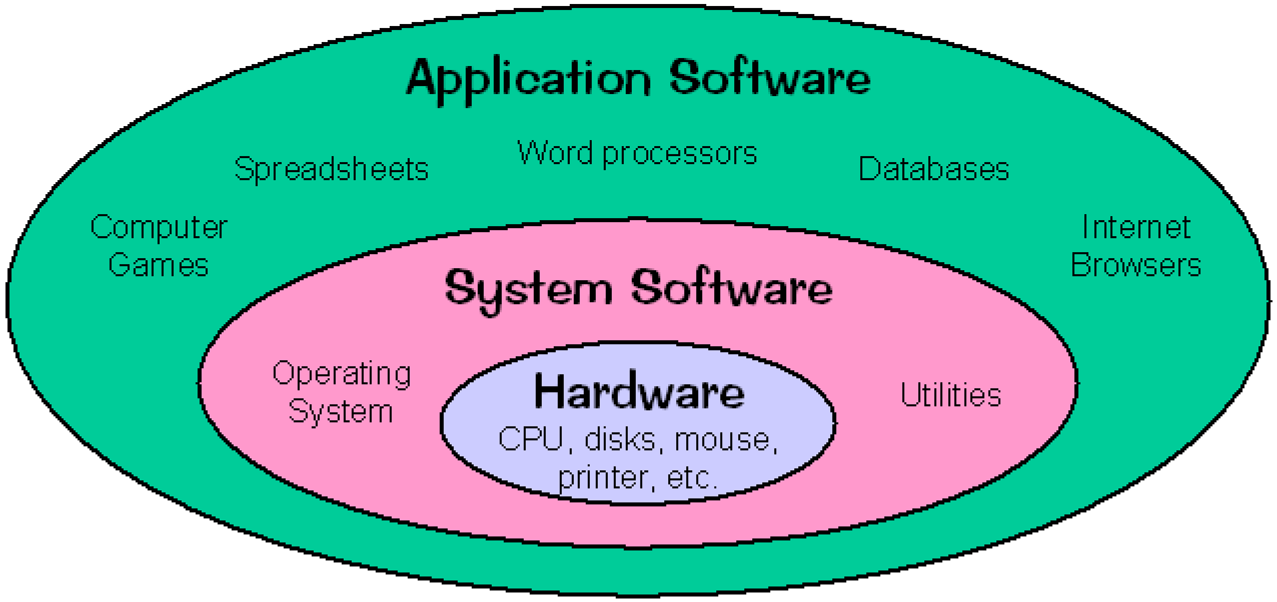
- ***Firmware*** are **programs that are permanently written and stored in memory**

OS and NOS

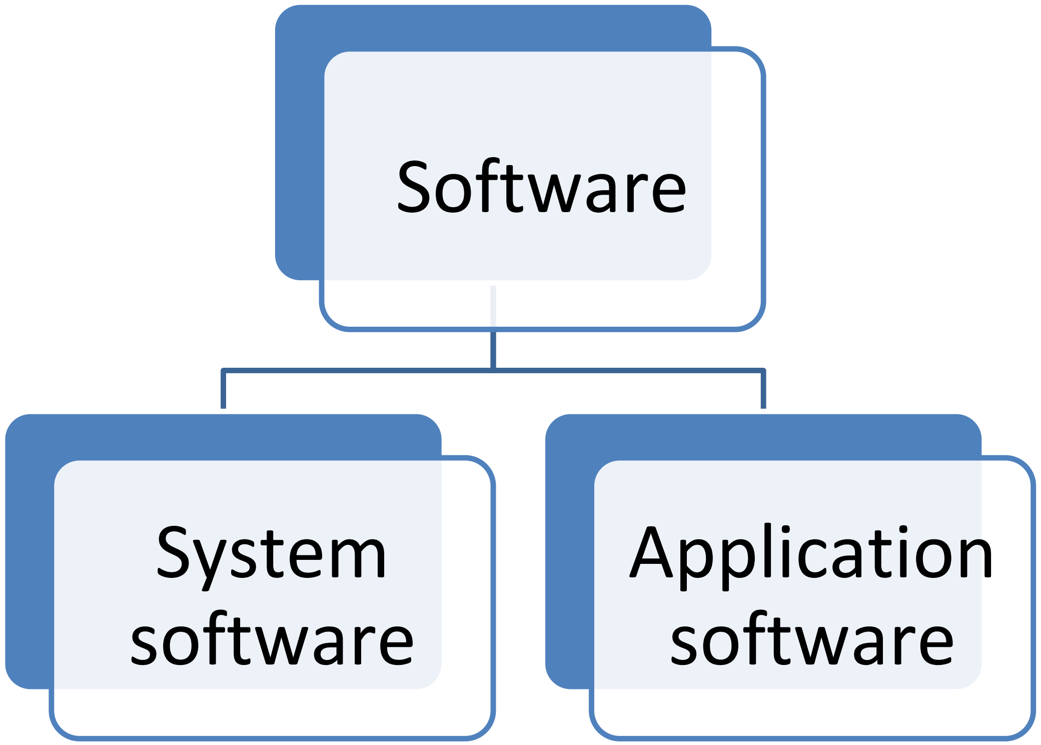
- **Operating System:** An OS manages a computer's internal functions, such as hardware interaction, file management, and running applications. It also has features like process management, which creates and deletes processes, and memory management, which allocates and de-allocates memory space to programs

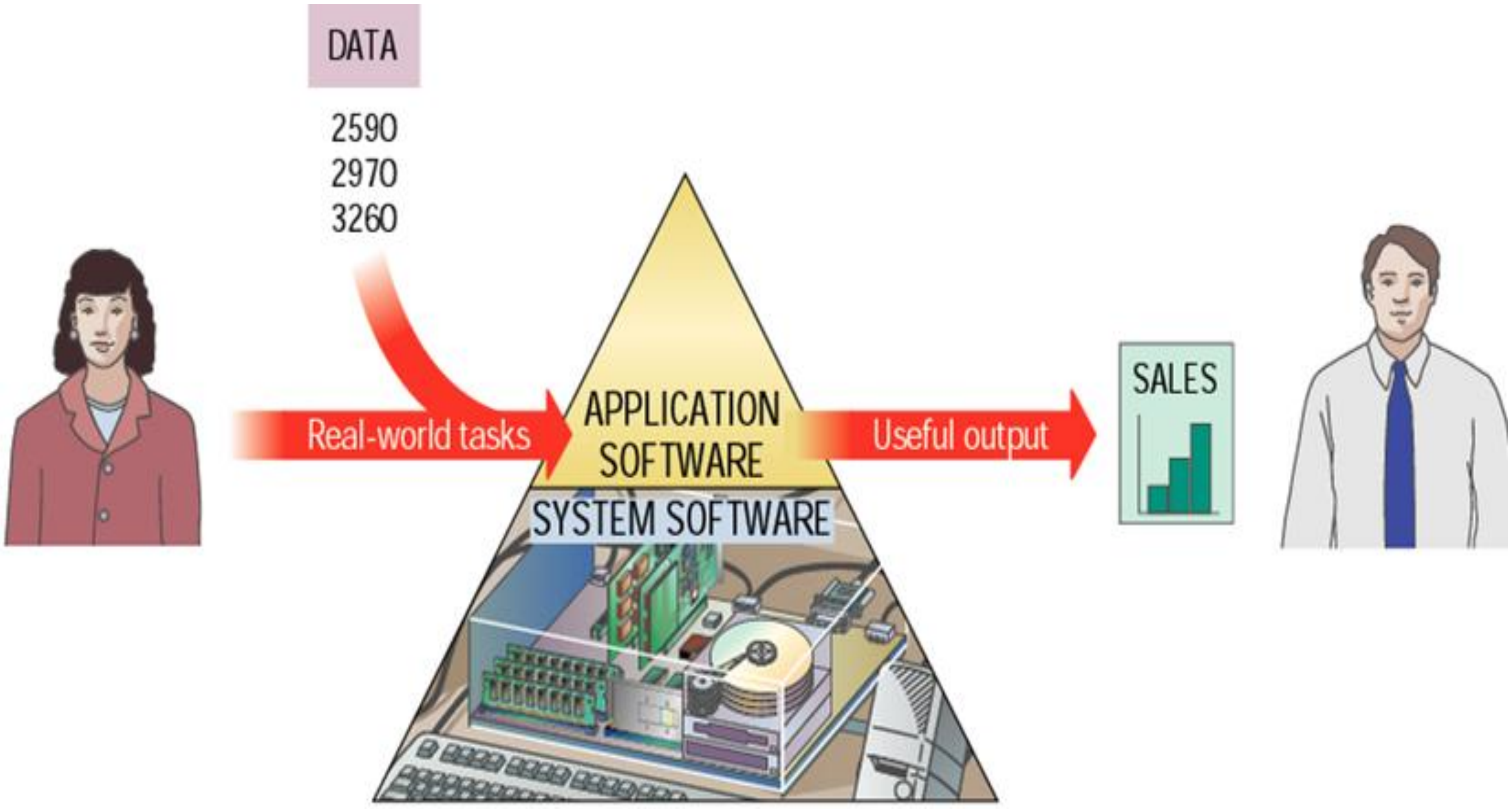
- **Network Operating System:** An NOS manages resources across a network and connects devices. It can support devices like PCs, workstations, printers, file servers, and databases, and allows them to communicate and share resources

Hardware and Software



Software types



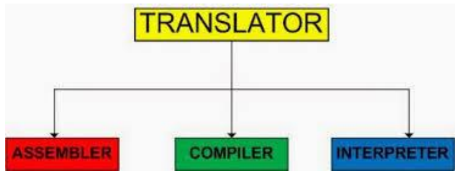

DATA
2590
2970
3260
Real-world tasks
APPLICATION SOFTWARE
SYSTEM SOFTWARE
Useful output
SALES

System software

- A program that **controls, integrates and manages the individual hardware elements of the computer system**
- These programs are **supplied by the computer manufacturer**
- The computer controlled by system software, **user cannot change the system software**

Examples

- Operating system
- Language translator
- System utility programs - Eg: antivirus software, backup software
- Device drivers – Eg: for printers, CD-ROM readers



Application software

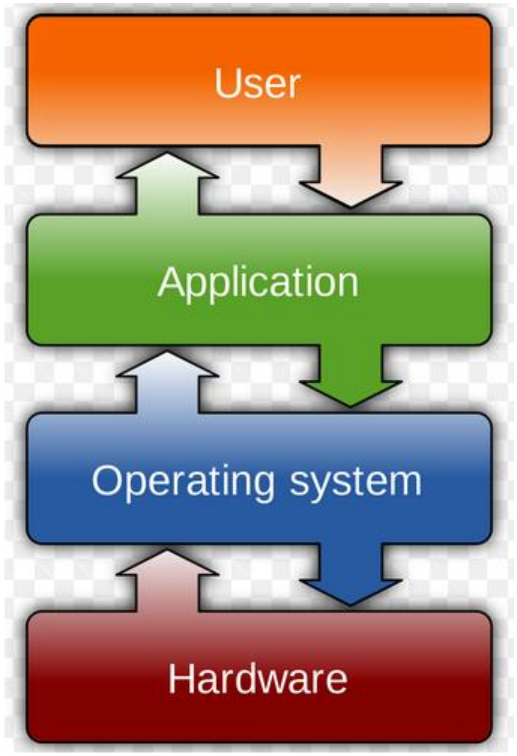
- **Set of programs necessary to carry out operations for a specific application**
- Application software is **controlled by system software**
- It **may be a single program** such as **Microsoft notepad** used for writing and editing text
- It **may also be a collection of programs** termed as Software **packages**, which is used or database management, etc.
- The most **commonly used application software** are
    - ✓ Word Processors
    - ✓ Spreadsheets
    - ✓ Image Editors
    - ✓ Database Management Systems
    - ✓ Presentation Applications
    - ✓ Games

System software vs. Application software

S.no	System software	Application software
1	It is a program that controls, integrates and manages the individual hardware elements of the computer system	It is the set of programs necessary to carry out operations for a specific application
2	These programs are supplied by the computer manufacturer	These programs can be purchased from the seller
3	The computer controlled by system software, user cannot change the system software	Application software is controlled by system software
4	Examples ➢ Operating system ➢ Language translator ➢ System utility programs ➢ Device drivers	Examples ➢ Word processor ➢ Spread sheets ➢ Presentation software

Concept of Operating System

SOFTWARE REQUIREMENTS



Operating System is an interface between the user of the computer and the hardware

OPERATING SYSTEM

- An **operating system** is the **most important software** that runs on a computer
- It manages the computer's **memory** and **processes**, as well as all of its **software** and **hardware**
- It also allows to **communicate** with the computer without knowing how to speak the computer's language

OPERATING SYSTEM

- Most of the time, there are several different computer programs running at the same time, and they all need to access computer's **central processing unit (CPU)**, **memory**, and **storage**. The OS coordinates all of this to make sure each program gets what it needs
- The kernel, or system core, ensures the smooth running of the OS within a computer and is the interface between the software and the hardware. It is used in all devices with an operating system, for example, computers, laptops, smartphones, smart watches, etc.

TYPES OF OPERATING SYSTEMS

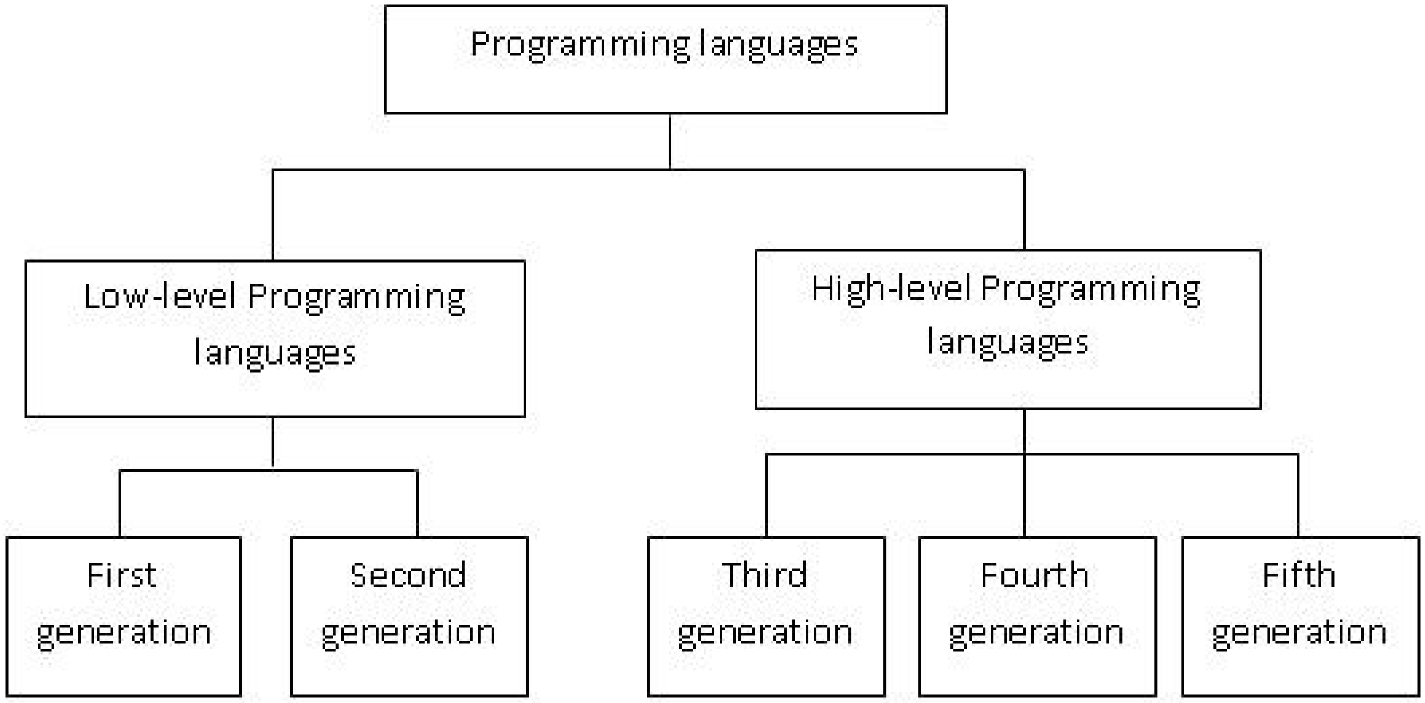
- Operating systems usually come **pre-loaded** on any computer you buy. Most people use the operating system that comes with their computer, but it's possible to upgrade or even change operating systems. The three most common operating systems for personal computers are **Microsoft Windows**, **macOS**, and **Linux**
- Modern operating systems use a **graphical user interface**, or **GUI**. A GUI lets you use your mouse to click **icons**, **buttons**, and **menus**, and everything is clearly displayed on the screen using a combination of **graphics** and **text**

Programming Language

- A programming language is an artificial language that can be used to control the behavior of a machine, particularly a computer
- Programming languages, like human languages, are defined through the use of syntactic and semantic rules, to determine structure and meaning respectively

https://www.cs.mcgill.ca/~rwest/wikispeedia/wpcd/wp/p/Programming_language.htm

Categories of Programming Languages



Evolution of Programming Languages

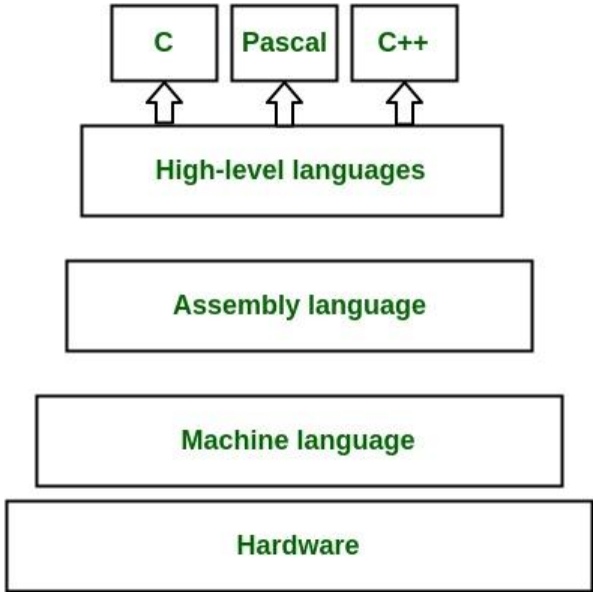
- **First Generation =** low-level languages like machine language
- **Second Generation =** low-level assembly languages used in kernels and hardware drives
- **Third Generation =** high-level languages like C, C++, Java, Visual Basic, and JavaScript

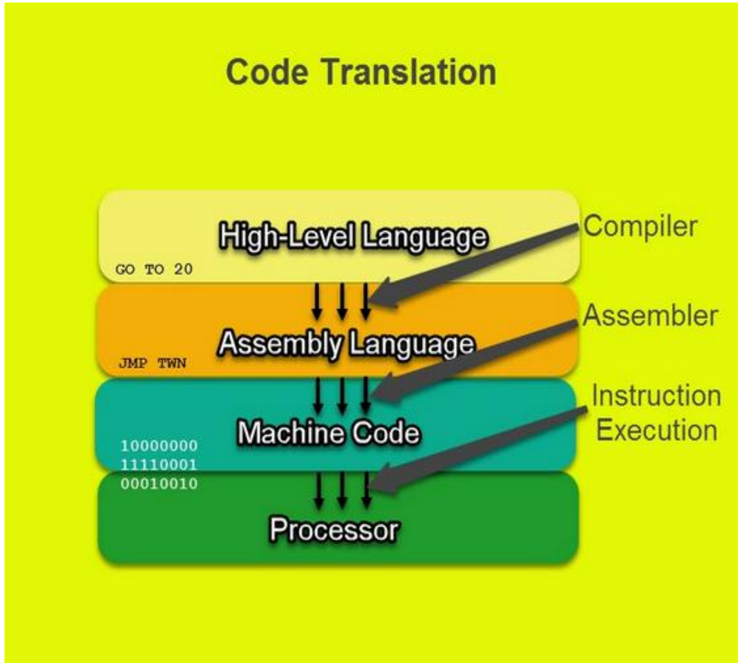
Evolution of Programming Languages

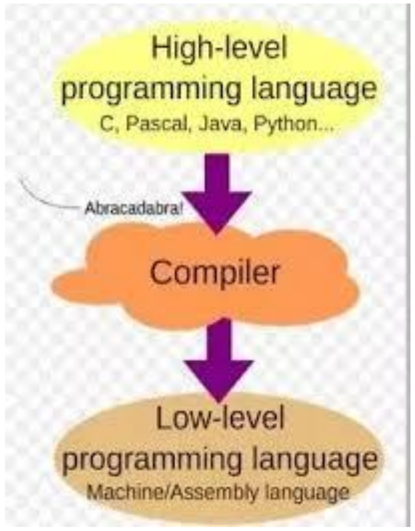
- **Fourth Generation =** These are languages that consist of statements that are similar to statements in the human language. These are used mainly in database programming and scripting. Examples of these languages include Perl, Python, Ruby, SQL, MatLab (Matrix Laboratory)

- **Fifth Generation =** Programming languages that have visual tools to develop a program. It is based on the concept of Artificial Intelligence. Examples of fifth-generation languages include Mercury, OPS5, and Prolog, LISP

EVOLUTION OF LANGUAGES






High-level programming language
C, Pascal, Java, Python...
Abracadabra!
Compiler
Low-level programming language
Machine/Assembly language

LOW-LEVEL LANGUAGES

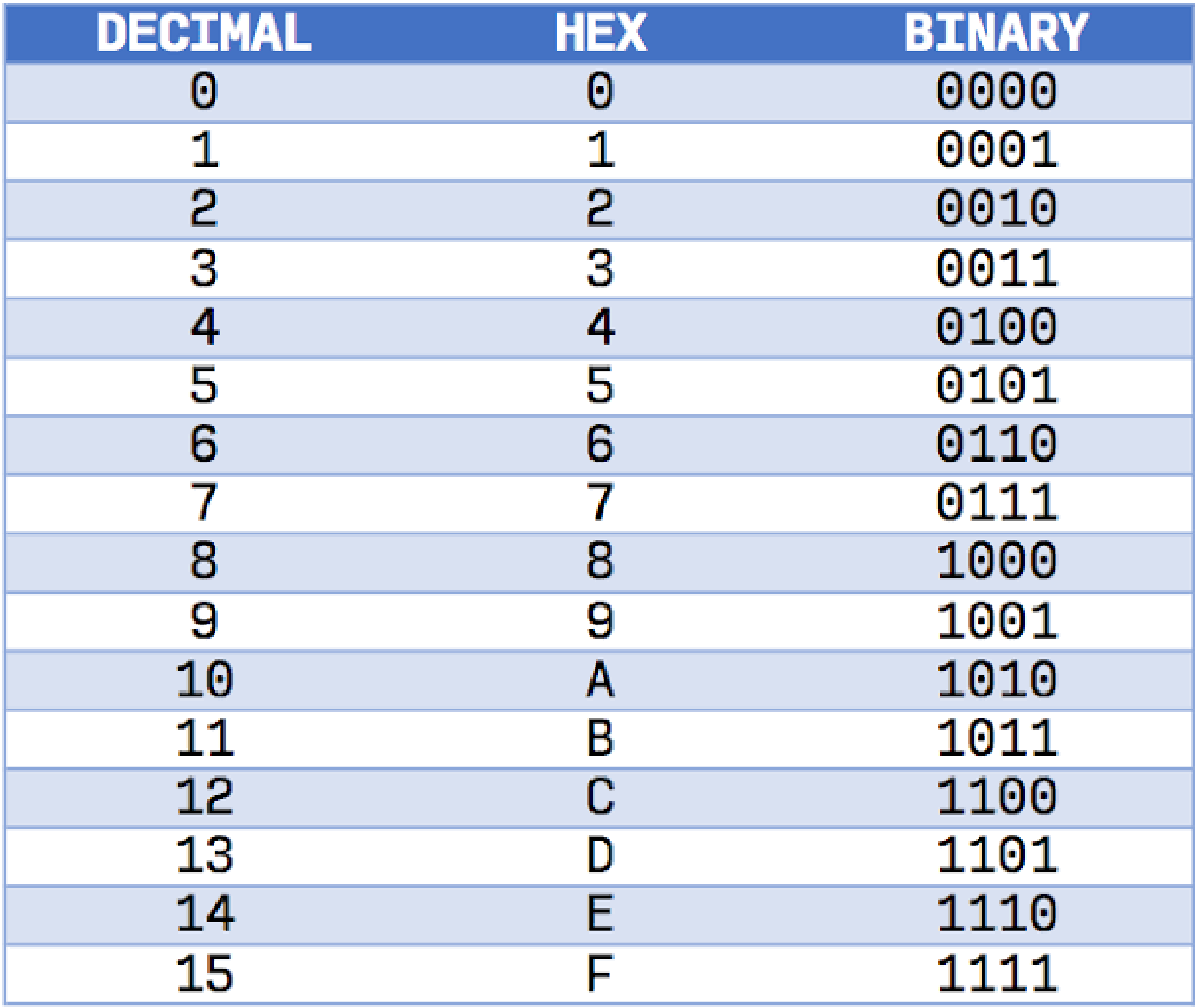
- **Low-level languages** are languages that sit close to the computer's **instruction set**. An instruction set is the set of instructions that the processor understands.
- A low-level language may also be referred to as a **computer's native language**
- It is very close to writing actual machine instructions, and it deals with a computer's hardware components and constraints
- In contrast to high-level language that are used for developing software, low-level code is not human-readable, and it is often cryptic.

Two types of low-level language are:

- **Machine code/language**
- **Assembly language**

MACHINE LANGUAGE

- Machine code is the set of instructions that a **CPU** understands directly and can act upon. A **program** written in machine code would consist of only 0s and 1s - **binary**. This is very difficult to write and **debug**. Even a very simple program could have thousands of 0s and 1s in it.
- All programs and programming languages eventually generate or run programs in machine language.
- Data is only represented with the help of binary format (0s and 1s). Machine language is normally displayed in **hexadecimal form** so that it is a little bit easier to read.



DECIMAL	HEX	BINARY
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

EXAMPLE OF MACHINE LANGUAGE

```
01001000   01100101   01101100   01101100
01101111   00100000   01010111   01101111
01110010    01101100    01100100
```

It means Hello World

MACHINE vs. BINARY

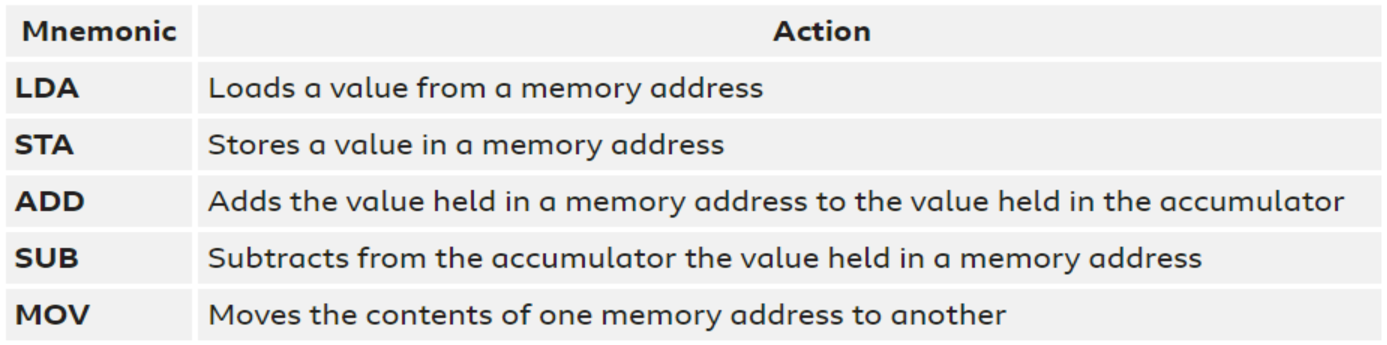
- Machine code can also be expressed in hex-format (hexadecimal) - a number system with base 16

- Let say you have the binary sequence 1001111000001010 - it can easily be converted to hex by grouping in blocks - each block consisting of four bits

- 1001 1110 0000 1010 => 9 14 0 10 which in hex becomes: 9E0A

ASSEMBLY LANGUAGE

Assembly language sits between machine code and **high-level language** in terms of ease of use. While high-level languages use **statements** to form instructions, assembly language uses **mnemonics** - short abbreviations. Each mnemonic directly corresponds with a machine code instruction. Here are some examples of mnemonics:



Mnemonic	Action
LDA	Loads a value from a memory address
STA	Stores a value in a memory address
ADD	Adds the value held in a memory address to the value held in the accumulator
SUB	Subtracts from the accumulator the value held in a memory address
MOV	Moves the contents of one memory address to another

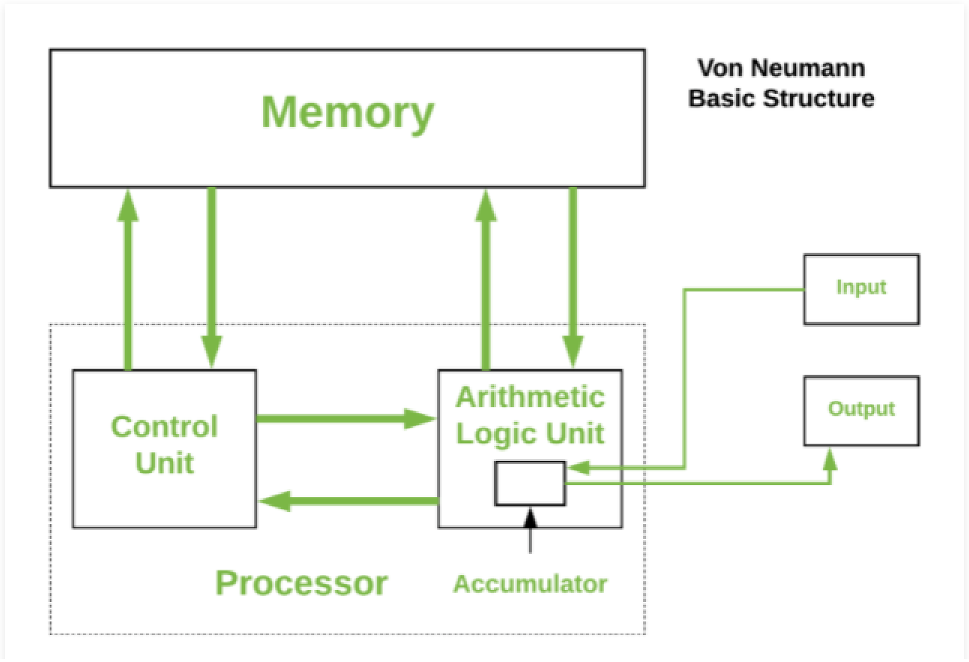
ASSEMBLY LANGUAGE

- In assembly language, programmers write programs as a series of mnemonics. Mnemonics are much easier to understand and debug than machine code, giving programmers a simpler way of directly controlling a computer
- Writing in mnemonics is easy for programmers because they are usually brief representations of the actual commands. They are quicker to write than machine code, and it is easier to spot mistakes

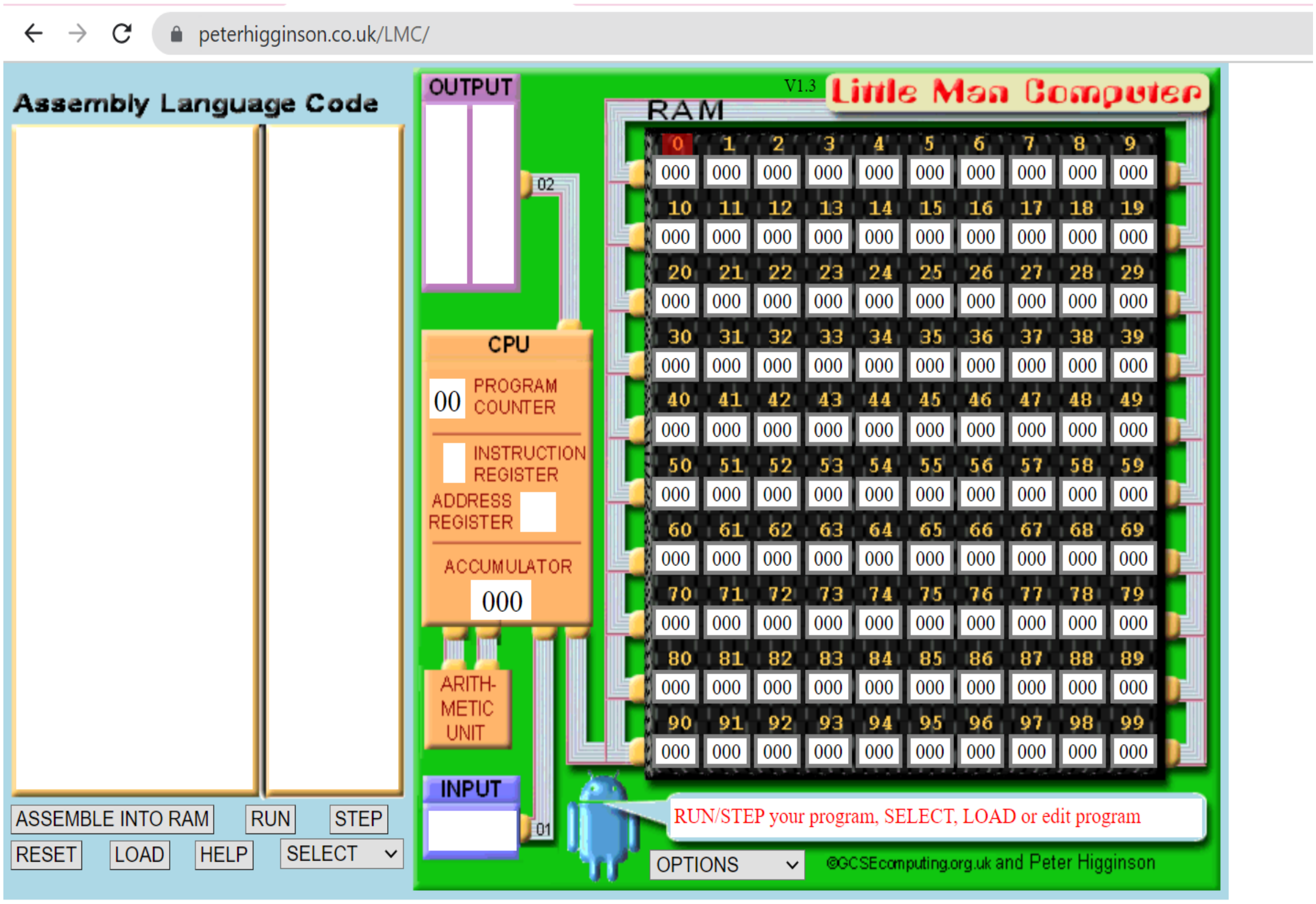
ASSEMBLY LANGUAGE

- Little Man Computer (LMC) is a simulation of a very basic processor using **Von Neumann architecture**. It uses an example of simple assembly language that contains a limited set of mnemonic instructions which can be used to program simple assembly programs. LMC is freely available on the Internet for students to use
- The machine languages need no translators. It is because they are already present in machine-understandable form
- Assembly languages need translators (also known as assemblers) for converting the mnemonics into a machine-understandable form

Von Neumann Basic Structure



https://peterhigginson.co.uk/LMC/



0	1	2	3	4	5	6	7	8	9
000	000	000	000	000	000	000	000	000	000
10	11	12	13	14	15	16	17	18	19
000	000	000	000	000	000	000	000	000	000
20	21	22	23	24	25	26	27	28	29
000	000	000	000	000	000	000	000	000	000
30	31	32	33	34	35	36	37	38	39
000	000	000	000	000	000	000	000	000	000
40	41	42	43	44	45	46	47	48	49
000	000	000	000	000	000	000	000	000	000
50	51	52	53	54	55	56	57	58	59
000	000	000	000	000	000	000	000	000	000
60	61	62	63	64	65	66	67	68	69
000	000	000	000	000	000	000	000	000	000
70	71	72	73	74	75	76	77	78	79
000	000	000	000	000	000	000	000	000	000
80	81	82	83	84	85	86	87	88	89
000	000	000	000	000	000	000	000	000	000
90	91	92	93	94	95	96	97	98	99
000	000	000	000	000	000	000	000	000	000

RUN/STEP your program, SELECT, LOAD or edit program

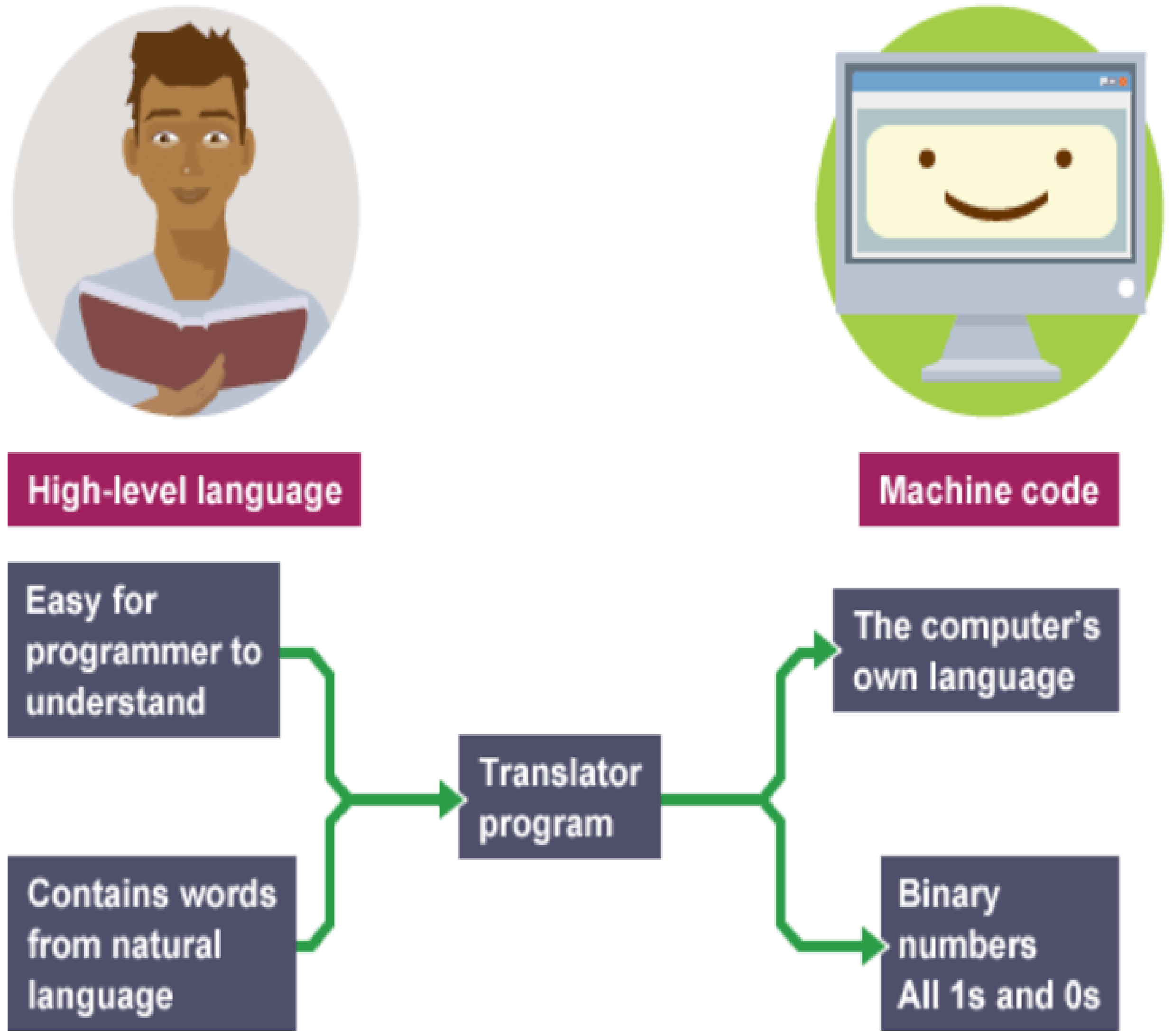
HIGH LEVEL LANGUAGE

- High-level languages allow programmers to write instructions in a language that is easier to understand than low-level languages.

- Translators are needed to translate programs written in high-level languages into the machine code that a computer understands.

HIGH LEVEL LANGUAGE

- The **instructions** that tell a computer what to do are written in **machine code**. Machine code is a series of numbers written in **binary**. Each number represents a different instruction.
- Programmers find machine code difficult to learn, program in and **debug**. As a result, the majority of programmers write **programs** in **high-level programming languages**. These languages are close to natural language - the spoken and written language of humans.
- For example, **C programming** uses 'printf', ‘if’, and 'while' **statements** - all words from the English language - to form instructions. In fact, instructions often look like abbreviated English sentences.


High-level language
Machine code
Easy for programmer to understand
Contains words from natural language
Translator program
The computer's own language
Binary numbers All 1s and 0s

HIGH LEVEL LANGUAGE

- Programmers write in high-level languages because they are easier to understand and are less complex than machine code. They allow the programmer to focus on what needs to be done, rather than on how the computer actually works.
- For example, in many high-level languages, to place a message on the screen, a programmer would use the statement 'print'. The programmer might not know how the computer actually generates the message. They just need to know how to use the 'print' statement.

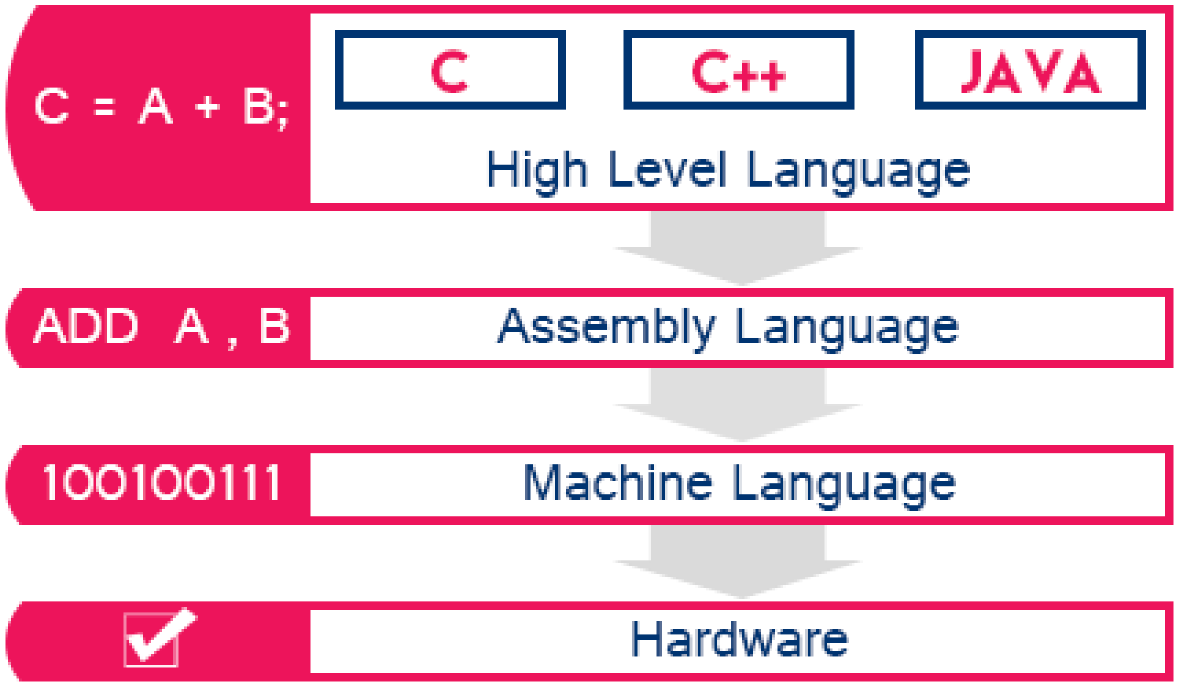
HIGH LEVEL LANGUAGE

Commonly used high-level languages

Many types of high-level language exist and are in common use today, including:

- Python
- Java
- C++
- C#
- Visual Basic
- JavaScript

Summary


C = A + B;
C
C++
JAVA
High Level Language
ADD A , B
Assembly Language
100100111
Machine Language
Hardware

Thank You!