PER SCHOLAS

300.1.1 Introduction to Computers and Programs

Introduction to Computer and Program Language

Learning Objectives:

This presentation is meant to provide basic skills knowledge and skill for understanding important concepts that you need as a programmer and developer.

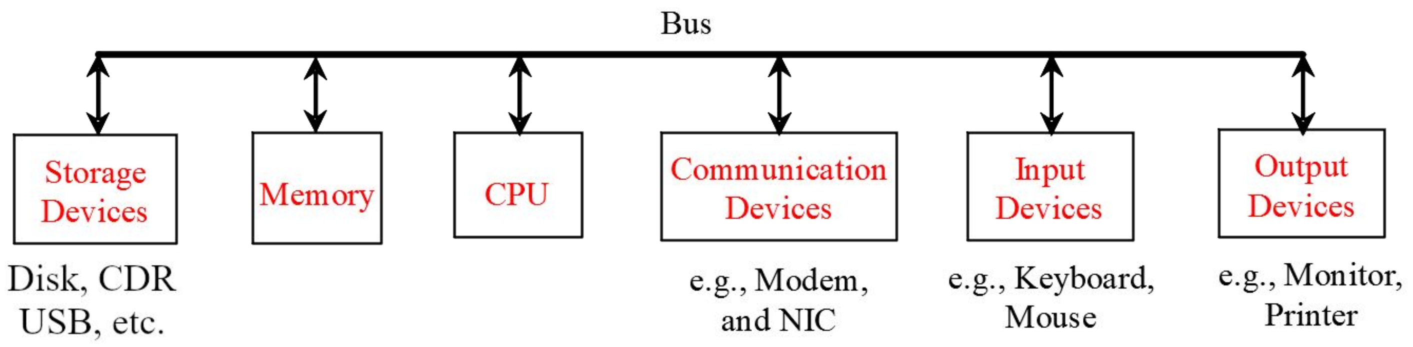
By the end of this presentation, learners will be able to:

- Describe the computer and its major components.
- Explain the programming languages and programming language classifications.
- Describe compiler and interpreter.
- Define RAM and ROM.

Table of contents

Computers

- A computer consists of a central processing unit (CPU), memory, hard disk, motherboard, Operating System and Power supply unit.
- Other components and accessories are input devices, output devices, or communication devices.

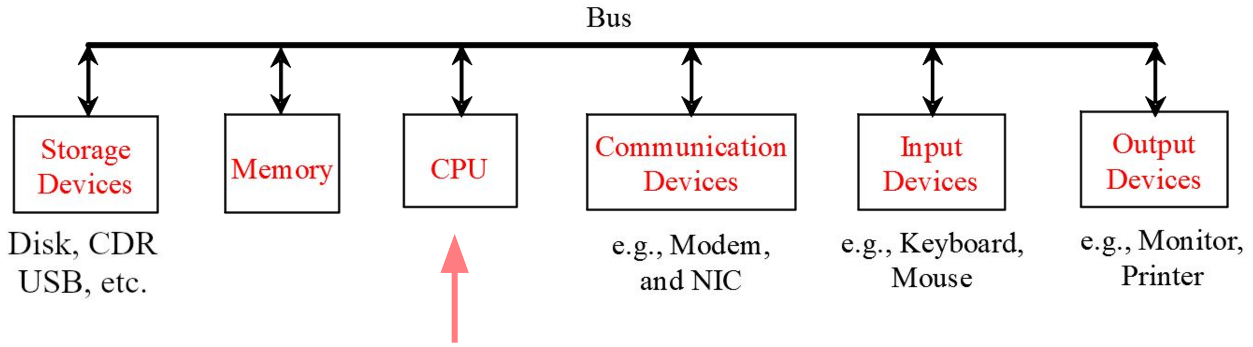


Computer Numbering Systems

- **Binary number:** Computers use binary numbers internally because storage devices, such as the internal memory and disk are made to store **0s** and **1s.**
- A number or a text inside a computer is stored as a sequence of **0s** and **1s.**
- Each 0 or 1 is called a bit, short for binary digit.
- The binary number system has two digits, 0 and 1.
- Binary numbers are not intuitive, since we use decimal numbers in our daily life.
- When you write a number like 20 in a program, it is assumed to be a decimal number.
- The digits in the decimal number system are 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9.
- Binary numbers tend to be very long and cumbersome.
- Hexadecimal numbers are often used to abbreviate binary numbers. The hexadecimal number system has 16 digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, and F. The letters A, B, C, D, E, and F correspond to the decimal numbers 10, 11, 12, 13, 14, and 15.
- Internally, computer software is used to convert decimal numbers into binary numbers, and vice versa.

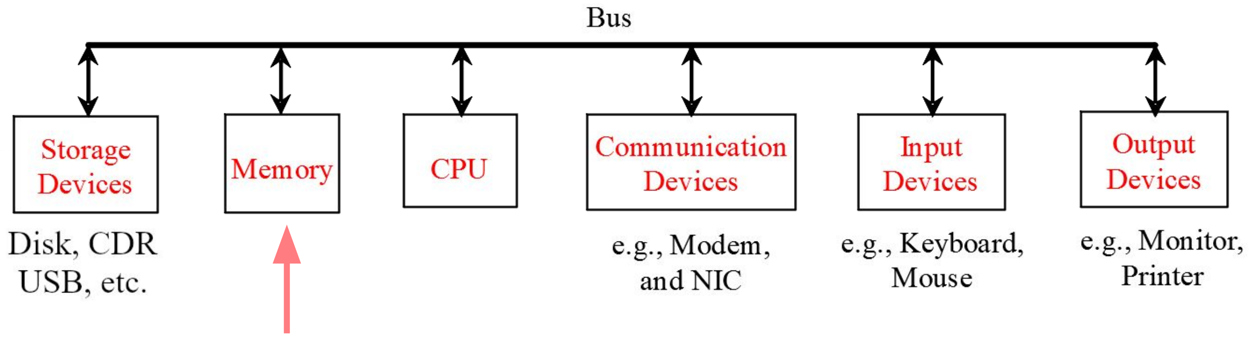
Central Processing Unit

The central processing unit (CPU) is the brain of a computer, it retrieves instructions from memory and executes them. The CPU speed is measured in megahertz (MHz), with 1 megahertz equaling 1 million pulses per second. The speed of CPUs has continuously improved. Computers, circa 2018, use an Intel i7 Processor, running at ~3 gigahertz (GHz), with 1 gigahertz equaling 1,000 MHz), and using up to 8 cores.



Memory

Memory is used to store data and program instructions for a CPU to execute. A memory unit is an ordered sequence of bytes, each holding eight bits. A program and its data must be brought into memory before it can be executed. A memory byte is never empty, but its initial content may be meaningless to your program. The current content of a memory byte is lost whenever new information is placed in it.

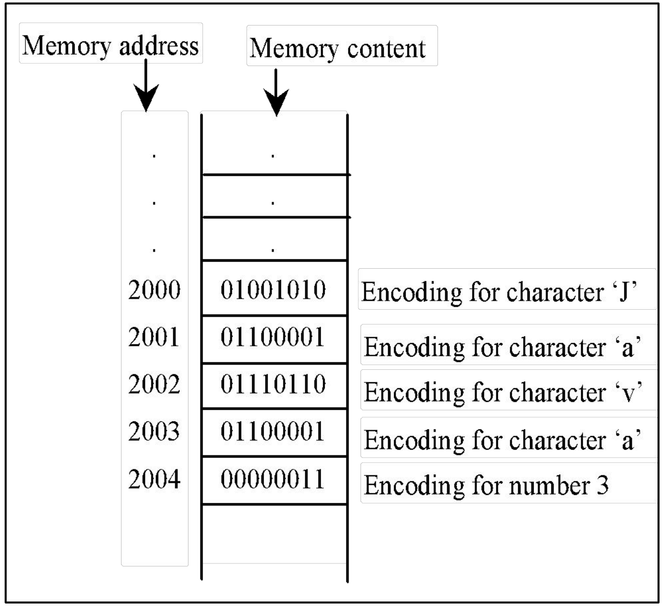


Visit the **Wiki** document for more information about memory.

How Data is Stored

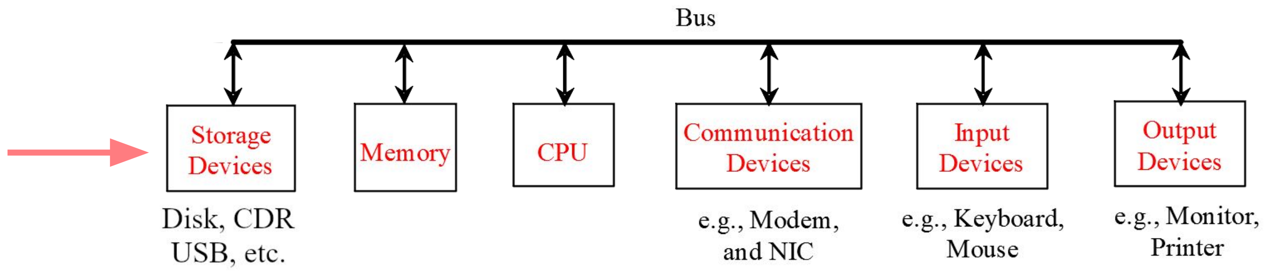
Data of various kinds, such as ***numbers, characters,*** and ***strings***, are encoded as a series of bits (zeros and ones). Computers use **zeros(0)** and **ones(1)** because digital devices have two stable states, which are referred to as *zero* and *one* by convention.

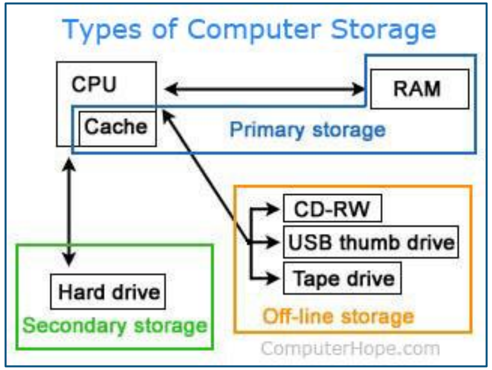
Programmers need not be concerned about the encoding and decoding of data, which is performed automatically by the system based on the encoding scheme. The encoding scheme varies. For example, character 'J' is represented by 01001010 in one byte. A small number, such as three, can be stored in a single byte. If the computer needs to store a large number that cannot fit into a single byte, it uses a number of adjacent bytes. No two data can share or split a same byte. A byte is the minimum storage unit.



Storage Devices

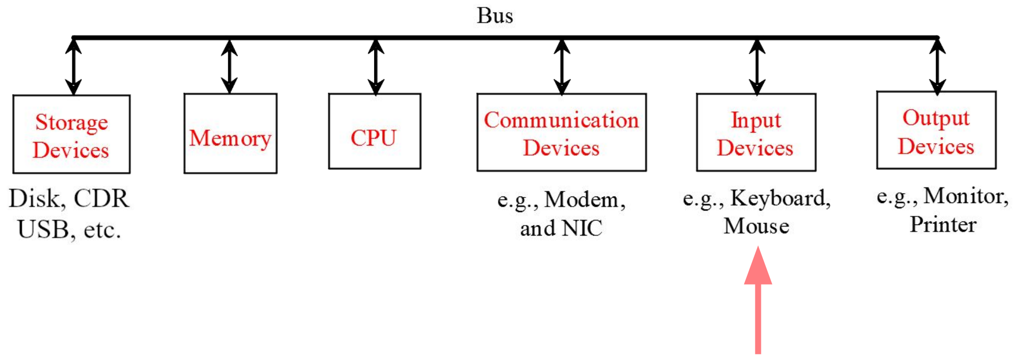
- Memory is volatile, and information in memory is lost when the power is off.
- Programs and data are permanently stored on storage devices and are moved to memory when the computer actually uses them.
- Few Storage device are Disk drives(hard disks), Cloud storage, USB flash drive, and CD drives*(hardly used)*.





Input Devices

- ❖ Any information or data sent to a computer for processing is considered “input.”
- ❖ Input is sent to a computer using an input device.
- ❖ Devices commonly used to provide input include:
  - ➢ Keyboard
  - ➢ Mouse
  - ➢ Microphone (audio input or voice input)
  - ➢ Webcam
  - ➢ Touchpad
  - ➢ Touch screen
  - ➢ Graphics Tablet
  - ➢ Scanner
  - ➢ Switch

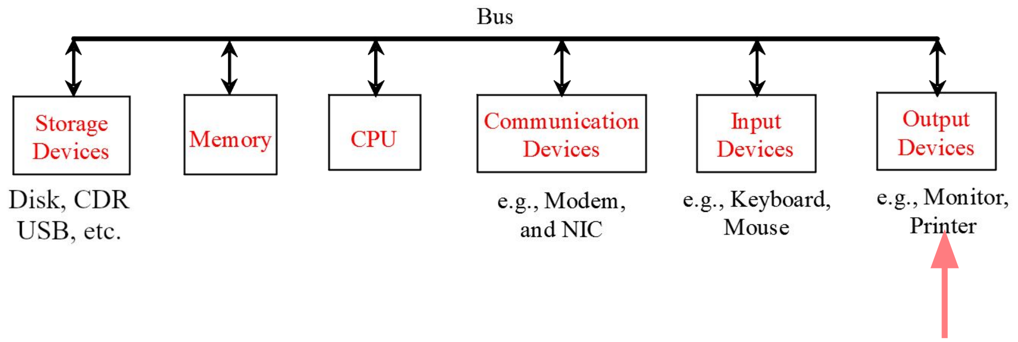


Output Devices and Types

Information processed by and sent out from a computer or other electronic device is considered "output." An example of output is anything viewed on your computer monitor screen, such as the ***words you type on your keyboard***. Without some type of output that a human can see, feel, or hear, a human could not interact with the computer.

In the picture, the bottom half shows data sent from a computer to a printer. When the printer finishes printing its hard copy, the paper is considered a form of *output*.

- There are four basic types of output: ***audio, graphics, text, and video***.



Communication Devices

- A communication device is a **hardware device** capable of transmitting a signal (analog or digital) over the telephone or any other communication wire. The signal can also be transmitted using a wireless device.
- A *regular modem* uses a phone line and can transfer data at a speed of up to 56,000 bps (bits per second).
- A *DSL* (digital subscriber line) also uses a phone line and can transfer data at a speed that is 20x faster than a regular modem.
- A *cable modem* uses the TV cable line, which is maintained by the cable company. A cable modem is as fast as a DSL.
- A *network interface card* (*NIC*) is a device used to connect a computer to a local area network (LAN).
- The *local area network* (*LAN)* is commonly used in business, universities, and government organizations.

Programs

- Computer *programs*, known as ***software***, are instructions to the computer.

- You tell a computer what to do through programs. Without programs, a computer is an empty machine. Computers do not understand the human language; therefore, you need to use computer language to communicate with them.

- Programs are written using **programming languages**.

How a Program Works?

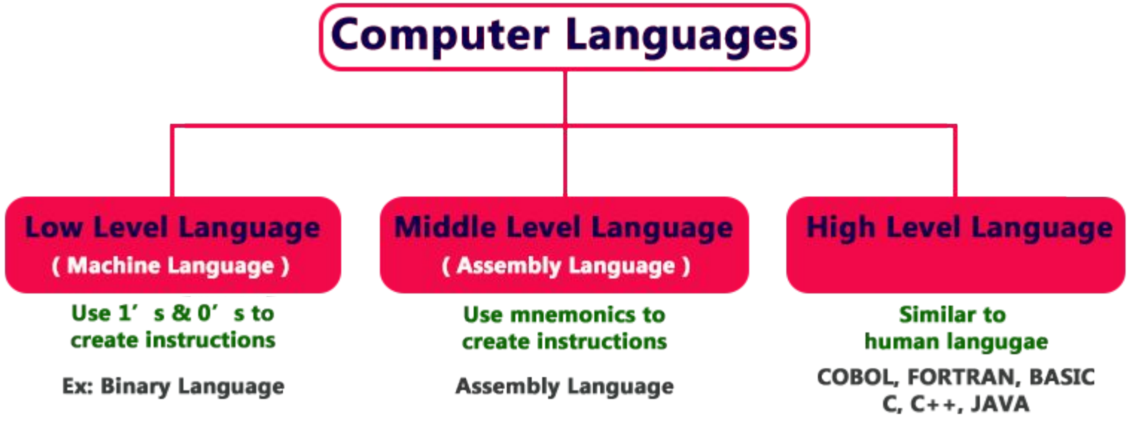
A computer’s CPU can only understand ***instructions*** that are written in ***machine language***. Because people find it very difficult to write entire programs in machine language; other programming languages have been invented.

Programming Languages

- Programming language theory is a subfield of computer science that deals with the design, implementation, analysis, characterization, and classification of programming languages.
- Although many languages share similarities, each has its own syntax. Once a programmer learns the languages rules, syntax, and structure, they write the source code in a text editor or IDE. Then the programmer often compiles the code into machine language that can be understood by the computer. Scripting languages, which do not require a compiler, use an interpreter to execute the script.
- **Computer languages** and **programming languages:** The term computer language is sometimes used **interchangeably** with programming language.

Programming Languages Classification

Over the years, programming languages have evolved from **low-Level** to **high-Level** languages. In the earliest days of computers, only Binary Language was used to write programs. The computer languages are classified as follows:



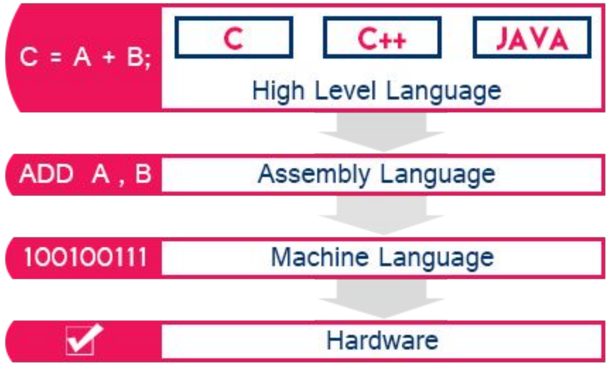


Image source : btechsmartclass.com

Overview of Machine Language

- Machine language is the only language that the computer understands; it a **low-level language** made up of binary numbers or bits that a computer can understand. It is also known as ***machine code*** or ***object code.***
- Machine language is a set of primitive instructions built into every computer. The instructions are in the form of binary code; therefore, you have to enter binary codes for various instructions. Programming with native machine language is a tedious process. Moreover, the programs are highly difficult to read and modify. For instance, to add two numbers, you might write an instruction in binary like this: `1101101010011010.`
- Example: The text "**Hello World**" would be written in the machine language as:

```
01101000 01100101 01101100 01101100 01101111 00100000 01110111
01101111 01110010 01101100 01100100 00001101 00001010
```

Reference link:
http://www.unit-conversion.info/texttools/convert-text-to-binary/

Visit the Wiki document for more information about Machine Language.

Overview of Assembly Language

- **Middle-level language** is a computer language in which the instructions are created using symbols such as letters, digits, and special characters. Assembly language is an example of middle-level language.
- Assembly languages were developed to make programming easy. Since the computer cannot understand assembly language, a program called an ***assembler*** is used to convert assembly language programs into machine code. For example, to add two numbers, you might write an instruction in assembly code like this:

ADDF3 R1, R2, R3

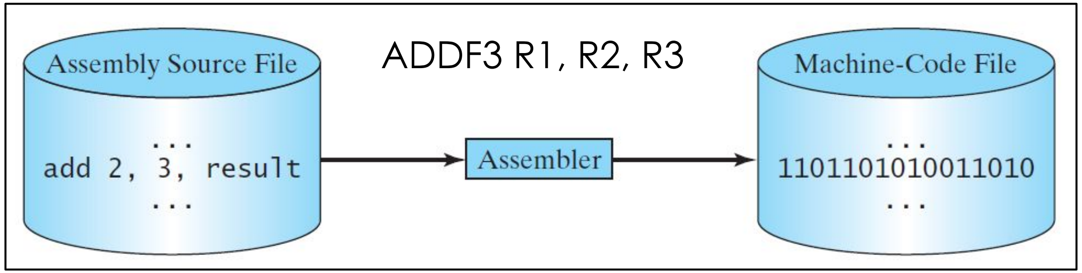


Image source : btechsmartclass.com

- Assembly language is a **human-only** language that is not understood by computers. As a result, it acts as a *link* between high-level programming languages and machine languages.

Overview of High-Level Language

- High-level languages are English-like and programs that are easy to learn. For example, the following is a high-level language statement that computes the area of a circle with a radius of 5:

```
area = 5 * 5 * 3.1415;
```

- High-level language is easy to understand for the user, but the computer cannot understand it. For the computer to understand it, it needs to be converted into the low-level language. We use Compiler or Interpreter to convert high-level language to low-level language.

Compiling and Executing a High-Level Program

A compiler translates a high-level language program into a separate machine language program. The machine language program can then be executed any time it is needed.

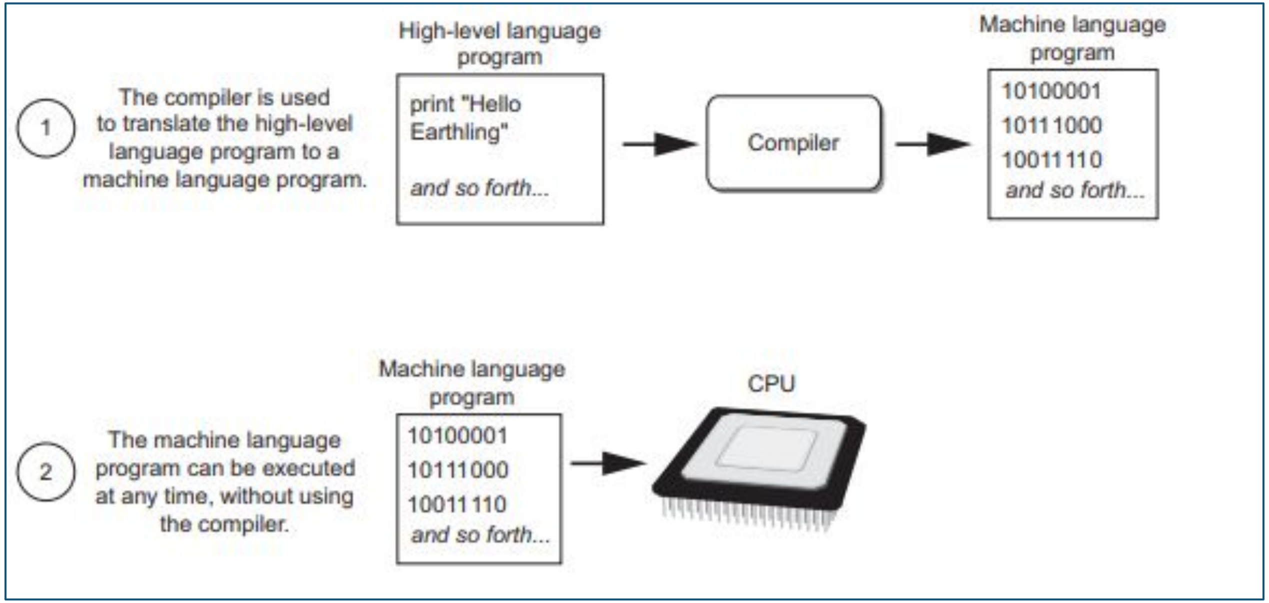
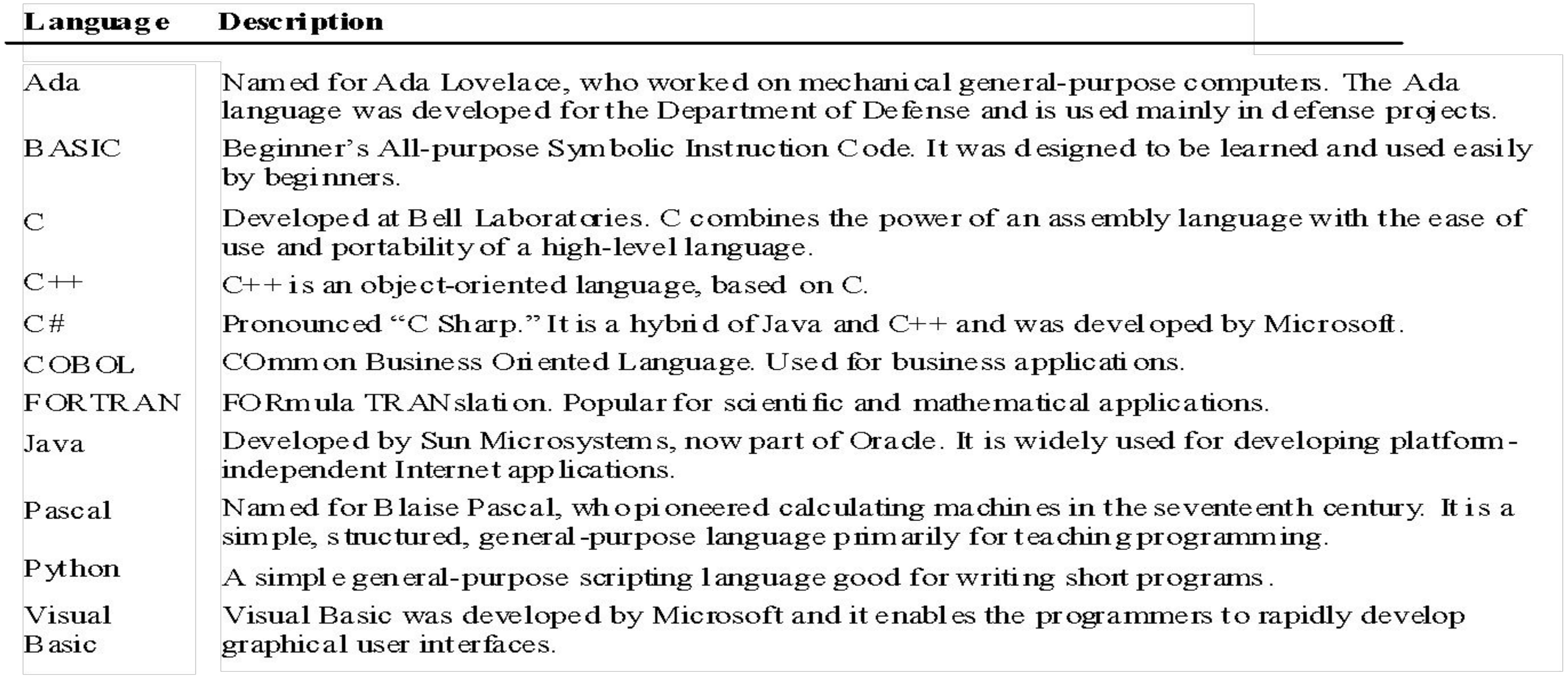

1
The compiler is used to translate the high-level language program to a machine language program.
High-level language program
print "Hello Earthling"
and so forth...
Compiler
Machine language program
10100001
10111000
10011110
and so forth...
2
The machine language program can be executed at any time, without using the compiler.
Machine language program
10100001
10111000
10011110
and so forth...
CPU

Image source : btechsmartclass.com

Popular High-Level Languages



Language	Description
Ada	Named for Ada Lovelace, who worked on mechanical general-purpose computers. The Ada language was developed for the Department of Defense and is used mainly in defense projects.
BASIC	Beginner's All-purpose Symbolic Instruction Code. It was designed to be learned and used easily by beginners.
C	Developed at Bell Laboratories. C combines the power of an assembly language with the ease of use and portability of a high-level language.
C++	C++ is an object-oriented language, based on C.
C#	Pronounced "C Sharp." It is a hybrid of Java and C++ and was developed by Microsoft.
COBOL	COmmon Business Oriented Language. Used for business applications.
FORTRAN	FORmula TRANslation. Popular for scientific and mathematical applications.
Java	Developed by Sun Microsystems, now part of Oracle. It is widely used for developing platform-independent Internet applications.
Pascal	Named for Blaise Pascal, who pioneered calculating machines in the seventeenth century. It is a simple, structured, general-purpose language primarily for teaching programming.
Python	A simple general-purpose scripting language good for writing short programs.
Visual Basic	Visual Basic was developed by Microsoft and it enables the programmers to rapidly develop graphical user interfaces.

How Programs Execute in Computer

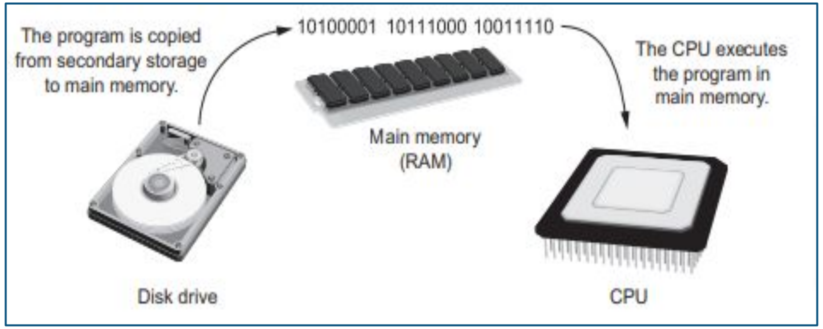
A program is copied into the main memory, and then executes.

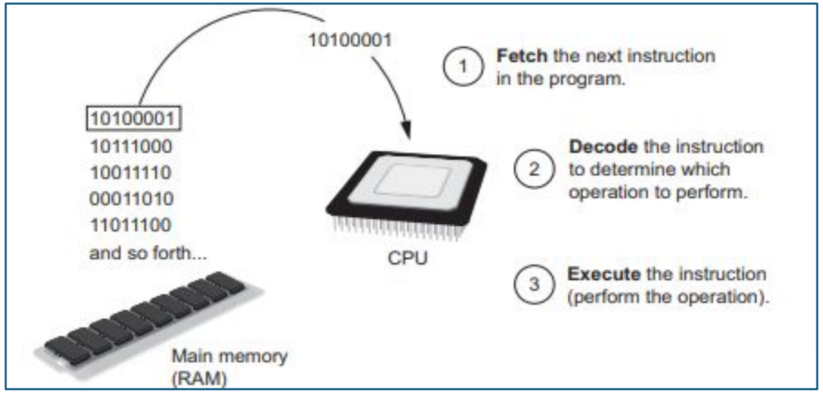
When a CPU executes the instructions in a program, it is engaged in a process that is known as the **fetch-decode-execute cycle**. This cycle, which consists of three steps, and is repeated for each instruction in the program. The steps are:

1. Fetch: A program is a long sequence of machine language instructions. The first step of the cycle is to fetch (or read) the next instruction from memory into the CPU.

2. Decode: A machine language instruction is a binary number that represents a command that tells the CPU to perform an operation. In this step, the CPU decodes the instruction that was just fetched from memory to determine which operation it should perform.

3. Execute: The last step in the cycle is to execute, or perform the operation.


The program is copied from secondary storage to main memory.
10100001 10111000 10011110
The CPU executes the program in main memory.
Main memory (RAM)
Disk drive
CPU


10100001
10100001
10111000
10011110
00011010
11011100
and so forth...
CPU
Main memory (RAM)
1 Fetch the next instruction in the program.
2 Decode the instruction to determine which operation to perform.
3 Execute the instruction (perform the operation).

Interpreting and Compiling Source Code

A program written in a high-level language is called a ***source program*** or ***source code***. Because a computer cannot understand a source program, the source program must be translated into **machine code** for execution. The translation can be done using an ***Interpreter*** or a ***Compiler,*** which are programming tools.

Interpreting Source Code

- An interpreter reads one statement from the source code, translates it into machine code or virtual machine code, and then executes it right away, as shown in the image below. Note that a statement from the source code may be translated into several machine instructions.
- Interpreter executes the source statements line by line during the execution.

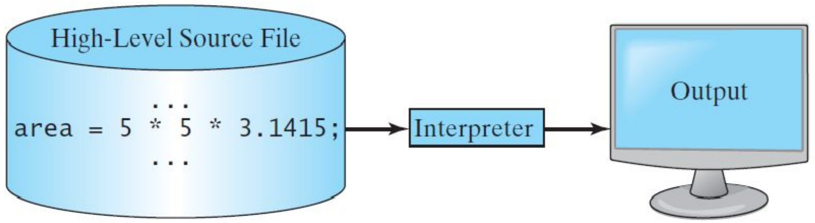


Image source : btechsmartclass.com

Interpreter does not require the linking of files or the generation of machine code.

Compiling Source Code

- A ***Compiler*** translates the entire source code into a machine-code file, and the machine-code file is then **executed**, as shown in the image below.

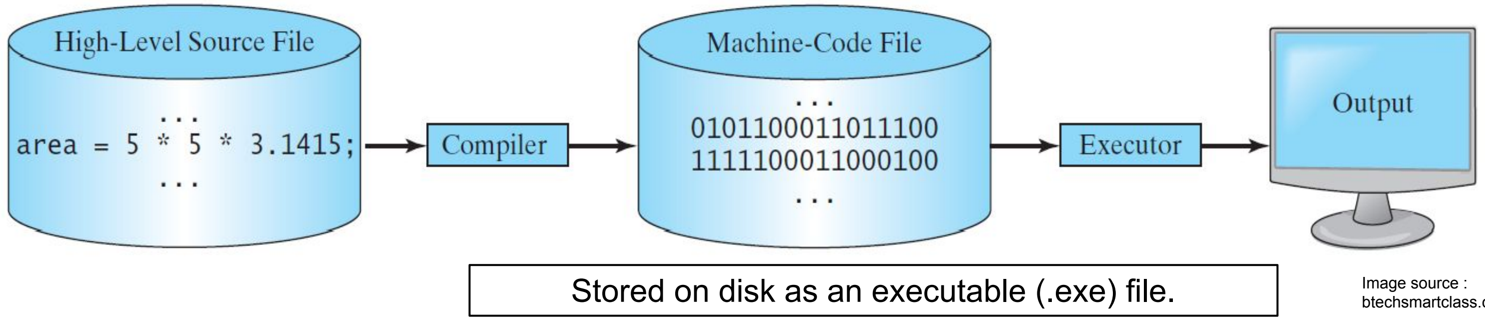


Image source : btechsmartclass.com

- The Compiler analyzes all of the language statements and throws an error when it finds something incorrect.
- If there is zero errors, the compiler converts the source code into machine code, and links various code files into a runnable program (exe).

Operations of Compiler

The important tasks executed by the compiler include:

- Breaking down the source program into *chunks,* and applying grammatical structure to each one.
- Enabling programmers to create the symbol table and the target program desired after the intermediate representation.
- Helping to compile source code and detect errors.
- Organizing and saving all codes and variables.
- Supporting separate compilation.
- Reading the full program, analyzing it, and translating it into a semantically similar language.
- Depending on the type of machine, converting source code to object code.

Compiler vs. Interpreter

Parameter	Compiler	Interpreter
Machine Code	Stores machine language on the disk in the form of machine code.	Does not save the machine language at all.
Generation of Program	Generates an output program in the exe format.	Does not generate an output program, which means that it evaluates the source program every time during individual execution.
Execution	Separates the program execution from the compilation.	Execution of the program is one of the steps of the Interpretation process; therefore, it can be performed line-by-line.
Optimization of Code	Capable of seeing the entire code upfront; thus, it makes the codes run faster by performing plenty of optimizations.	Optimization is not very robust (sees a code line-by-line).
Execution of Error	Displays every error and warning while compiling; therefore, a program cannot be run until the errors are fixed.	Reads every statement, and then displays the errors, if any. A user must resolve these errors in order to interpret the next line.
Output	Generates intermediate machine codes.	Never generates any intermediate machine codes.
Programming Languages	Java, Scala, C#, C, C++ use Compilers.	Python, Perl, Ruby, PHP use Interpreter

Programming Paradigms

Declarative and **Imperative** programming are common programming paradigms, with key differences:

➢ Declarative programming focuses on how a program operates and what the program should accomplish.

➢ Imperative programming focuses on how the program should achieve the result.

Imperative languages:
- C
- C++
- Java
- PHP
- Ruby
- Pascal
- Python

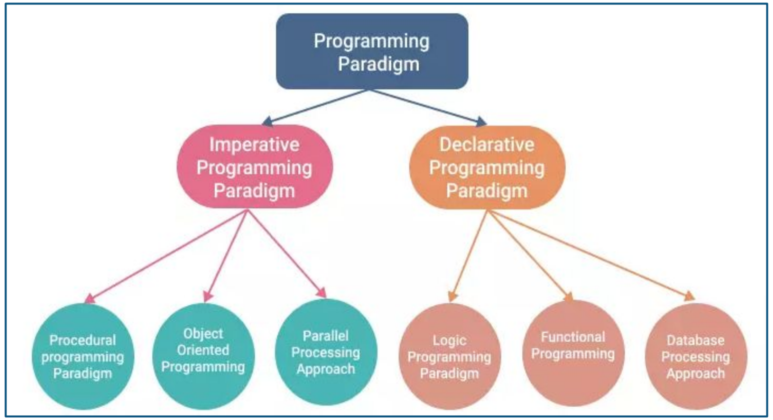

Programming Paradigm
Imperative Programming Paradigm
Declarative Programming Paradigm
Procedural programming Paradigm
Object Oriented Programming
Parallel Processing Approach
Logic Programming Paradigm
Functional Programming
Database Processing Approach

Image source : btechsmartclass.com

Declarative languages:
- Prolog
- javascript
- Scala
- Lisp
- SQL
- XQuery
- Clojure

Random Access Memory

Random Access Memory (RAM) plays a very important role in computer programming languages and software/application development industry.

- ❑ RAM is the temporary storage component (short-term memory) of a computer, and holds all of the data currently being used, including the data from the website to the movement of the mouse.

- ❑ Whenever you perform anything on your computer, you execute many processes. For example, when typing a sentence, saving a document, or jumping in a video game, your RAM is working behind the scenes. It is much easier (faster) to perform those tasks if the data is stored in your computer's RAM, where it is easily accessible, rather than on the hard drive (SSD or HDD).

- ❑ RAM is **volatile memory**, and is always connected to power so that each memory cell's electrical charge can be refilled as needed.

Random Access Memory (continued)

As short-term memory, RAM is designed to work with small bits of data at a time. For example, when you click on a link to go to a new website, a series of capacitors and transistors (basically switches) in the RAM's circuit board turn on or off, letting your computer translate the link into the web page you see.

If you are working on a document or spreadsheet and want to save your work for later, you will move it from being stored in your RAM, where it can be found immediately into the long-term storage of your hard drive. And your computer will need to do more work to pull up data from your hard drive (longer-term memory).

When working with RAM, you can access data in any order — it is *random* access, not *sequential* access. RAM is directly connected to your computer's motherboard, allowing for the fastest speeds possible. **The more RAM you have, the better your computer will perform.**

Random Access Memory Types

There are three main types of RAM:

- **SRAM (static random access memory) -** Data is stored in transistors and uses (low) power constantly to keep data fresh. It does not need to be refreshed. SRAM is used in cache memories, such as the small files your processor (CPU) accesses all the time.
- **DRAM (dynamic random access memory) -** Data is stored in capacitors, and slowly discharges power. The gradual energy decline means that it needs to be refreshed periodically to work properly. When the power is cut, the electrical charges dissipate and the RAM is emptied of data.
- **VRAM (video random access memory) -** Immediate data sits in this space as you work with it. VRAM is similar to RAM, except that it sits on your graphics card focusing on image data. It is great for content editors and anyone looking to speed up their gaming PC because it helps load more and better graphics.

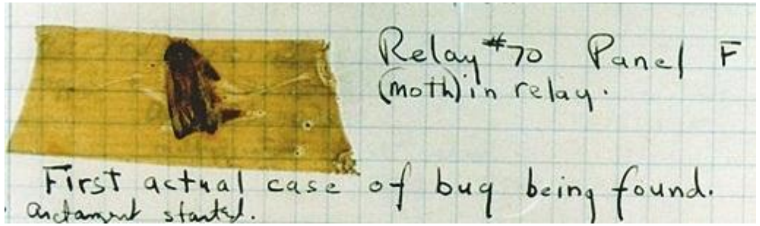
Visit the **Wiki** document for more information about **RAM Types**.

Read-Only Memory

- RAM is volatile memory that temporarily stores the files you are working on. Read-only Memory **(ROM)** is **non-volatile memory** that permanently stores instructions (information) for your computer on a **chip**. The memory does not depend on an electric current to save data; instead, data is written to individual cells using binary code.
- Non-volatile memory is used for parts of the computer that do not change, such as the initial boot-up portion of the software, or the firmware instructions that make your printer run. Turning off the computer does not have any effect on ROM. **Non-volatile memory** cannot be changed by users.
- ROM data is located directly on the motherboard (chip) and tells a computer how to work. It is the first thing your computer accesses when you turn it on. ROM data is your computer's most essential data because nothing else is accessible without it.
- ROM cannot be changed — that is why it is called "read only." RAM data, by contrast, is constantly being changed.

Debugging

No introduction to computer programming would be complete without at least mentioning **debugging, which refers to the discovery and correction of mistakes in computer programs**. The computer is doing what you instructed it to do, not what you meant it to do. The origin of the term came from a bug (a moth) found in a relay of a computer in 1947 by Admiral Grace Murray Hopper. She found out why her program was not working.



Reference:
https://en.wikipedia.org/wiki/File:First_Computer_Bug,_1945.jpg

- Debugging is a process. Debugging a program can be done in steps that match the *Scientific Method:*
  1. Observe
  2. Hypothesize.
  3. Predict.
  4. Test.

Source Code, Machine Code and Assembly Language

Click here to watch video:
https://www.youtube.com/watch?v=bgN4LBt5buU

Knowledge Check

1- Which is a series of instructions written to perform a specified task on a computer?

A. Programming
B. Programming Language
C. Computer Program
D. Computer

2 - Which is a tool that allows programmers to write commands to a computer?

A. Programming Language
B. Programmer
C. Visual Studio
D. Compiler

3 - Which refers to the classification of programming language that is understood by the computer?

A. Assembly Language
B. High-Level Language
C. Object Oriented/Event-Driven Programming Language
D. Machine Language

Knowledge Check

4 - High-level language must be processed by a __________ or a __________ before the code is executed or read by the computer.

A. Assembler; Compiler
B. Debugger; Compiler
C. Runner; Assembler
D. Compiler; Debugger

Knowledge Check - Answers

1. Computer Program.
2. Programming Language.
3. Machine Language.
4. Assembler or Compiler.

Knowledge Check

- A CPU understands instructions that are written only in what language?
- A program has to be copied into what type of memory each time the CPU executes it?
- What is assembly language?
- What do you call a program that translates a high-level language program into a separate machine-language program?
- What do you call a program that both translates and executes the instructions in a high-level language program?

Summary

A computer is a programmable device that stores, retrieves, and processes data. The term "computer" was originally given to humans (human computers) who performed numerical calculations using mechanical calculators, such as the abacus and slide rule.

- ❑ Processor - A component that executes instructions from the software and hardware.
- ❑ Memory - Temporary primary storage for data traveling between the storage and the CPU.
- ❑ Storage device (e.g., hard drive) - Slower secondary storage that permanently stores data.
- ❑ Machine language is a low-level language made up of binary numbers or *bits* that a computer can understand.
- ❑ Assembly language is an example of middle-level language.

Random Access Memory (RAM) is the temporary storage component (short-term memory) of a computer.

Read-Only Memory **(ROM)** is **non-volatile memory** that permanently stores instructions (information) for your computer on a **chip.**

References

https://www.pearsonhighered.com/assets/samplechapter/0/3/2/1/0321537114.pdf

http://guyhaas.com/bfoit/itp/Programming.html

Questions?