The difference between RUNNING, EXECUTING, AND COMPILING PROGRAM

Running a program means loading program binary image from secondary storage to main memory. When program run first time it's process control block created(PCB) and exist until program terminated. A program in running state called a process.

Execution of program means it's running and consume CPU cycle. because most of the time running program always not utilize CPU because I/O handling, or multitasking os doesn't assign CPU for process to permanently.

Compiling a program means convert a high level language into machine level language. Since machine understand own language and it's difficult to learn and program a machine language so we write program in high level language and using compiler we convert into machine level language (Assembler and Linker also involved in converting high level language into machine level language).

**High and Low Level Languages**

A computer program is a list of instructions that enable a computer to perform a specific task.

Computer programs can be written in high and [low level languages](https://www.computerscience.gcse.guru/glossary/low-level-languages), depending on the task and the hardware being used.

So, what’s the difference between high level language and low level language?

**High Level Languages**

When we think about computer programmers, we are probably thinking about people who write in [high-level](https://www.computerscience.gcse.guru/glossary/high-level-languages) programming languages.

[High level languages](https://www.computerscience.gcse.guru/glossary/high-level-languages) are written in a form that is close to our human language, enabling to programmer to just focus on the problem being solved.

No particular knowledge of the hardware is needed as [high level languages](https://www.computerscience.gcse.guru/glossary/high-level-languages) create programs that are portable and not tied to a particular computer or microchip.

These programmer friendly languages are called ‘high level’ as they are far removed from the [machine code](https://www.computerscience.gcse.guru/glossary/machine-code) instructions understood by the computer.

Examples include: C++, Java, Pascal, Python, Visual Basic.

**Advantages**

* Easier to modify as it uses English like statements
* Easier/faster to write code as it uses English like statements
* Easier to debug during development due to English like statements
* Portable code – not designed to run on just one type of machine

**Low Level Languages**

[Low level languages](https://www.computerscience.gcse.guru/glossary/low-level-languages) are used to write programs that relate to the specific architecture and hardware of a particular type of computer.

They are closer to the native language of a computer (binary), making them harder for programmers to understand.

**Examples of low level language:**

* [Assembly Language](https://www.computerscience.gcse.guru/glossary/assembly-language)
* [Machine Code](https://www.computerscience.gcse.guru/glossary/machine-code)

**Assembly Language**

Few programmers write programs in low level [assembly language](https://www.computerscience.gcse.guru/glossary/assembly-language), but it is still used for developing code for specialist hardware, such as device drivers.

It is easy distinguishable from a high level language as it contains few recognisable human words but plenty of mnemonic code.

**Advantages**

* Can make use of special hardware or special machine-dependent instructions (e.g. on the specific chip)
* Translated program requires less memory
* Write code that can be executed faster
* Total control over the code
* Can work directly on memory locations

**Machine Code**

Programmers rarely write in machine code (binary) as it is difficult to understand.