**LearnCpp**

**Chp .1**

**Introduction to programming languages**

Machine Language:

A CPU is limited to a small set of instructions known as *machine code* (machine language or an instruction set). Each instruction is composed of binary digits (0 and 1) known as bits that are translated by the CPU to perform a specific job like comparing two numbers.

Assembly Language:

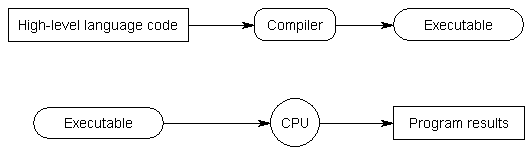
Writing machine code is a rigorous and exhausting process. Programmers had to know their binary really well! This difficulty led to the creation of Assembly language where sets of bits were identified by names rather than their numbers. This made programming easier for humans but not for computers. Computers needed to still translate assembly back into machine code. What’s important about assembly is that it is tailored to a particular CPU thus enchanting its speed and performance. Assembly written for 1 CPU will not work on another.

High-level languages:

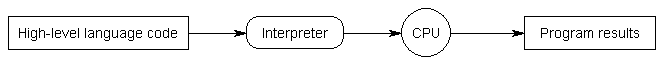
Writing assembly itself was still a tedious task because it still required many instructions for it to be written and the exclusivity to a particular CPU required the same programs to be rewritten several times. In response to this high-level language were created to act as universal translators that allowed programs that could be run by any type of CPU. These programs still needed to transform its information into a language that a CPU can understand.

Compilers vs Interpreters

A Compiler is a program that reads code and produces an executable program that a CPU can understand directly. Once it’s been through a compiler once, it never needs to go through the process again.



An Interpreter is a program that *directly* runs and executes code without compiling it (transforming it into machine code). Interpreters tend to be more flexible but less efficient than compilers because it repeats its process for every instance the program is run for.



I personally like to think as them as a white box and a black box respectively. With compilers, we can understand procedurally how a CPU knows how to run a program while with an interpreter, everything is managed through itself like a black box and it somehow creates results.