Programming languages

Machine code:

```
000010 00000 00000 00000 10000 000000
100011 00011 01000 00000 00001 000100
000000 00001 00010 00110 00000 100000
```

Assembly code:

```
MOVE R2, 0
ADD R2, R0, R2
SUB R1, R1, 1
```

High-level language:

```
pWord = input(“Enter the password”).upper()
```

Programming languages

Low level languages: (assembly language & machine code)

Used in embedded systems or CPU processing

- difficult for humans to understand/write
- may only be able to be used with a limited set of systems
- efficient to run – 1:1 equivalence with CPU processes (so quick and less memory needed)
- programmer has total control over components

Programming languages

High level languages:

Such as Python, Java etc...

- can be used on many different systems
- much more powerful set of constructs (selection, repetition, arrays etc…)
- closer to our languages - easier and quicker to learn, use, debug etc...
- need to be converted to machine code – takes CPU time and less efficient to run

Programming languages

Key points:

- Assembly Language and Machine Code are low-level languages
- most programs are written in high-level languages (e.g. Python)
- everything ends up as machine code
- there are advantages to using low-level languages, but they’re harder for humans to write

Translators

All programs need to be translated to machine code

Assembler: converts assembly language to machine code

Compiler: converts HL to machine code. Deals with whole program before it runs any of it

Interpreter: converts HL to MC - translating each line as it executes it (e.g. JavaScript)

Compiler	Interpreter
Translates the whole program to produce the executable object code	Translates and executes one line at a time
Compiled program executes faster as it is already in machine code	Interpreted programs take more time to execute because each instruction is translated before it is executed
Customers do not need to have the compiler installed on their computer to run the software	Customers must have the interpreter installed on their computer and they can see the source code
Customers cannot see the actual source code when you distribute the program	Customers can see the source code and could copy it
Used for software that will be run frequently or copyright software sold to a third party	Used for program development and when program must be able to run on multiple hardware platforms

An **interpreter** is also used to translate high-level language code into machine code. An interpreter executes the program directly, translating each statement into a sequence of calls to one or more subroutines already compiled into machine code. You must have the interpreter installed on your computer in order to run the software.

A long, complex program will take considerably more time to execute if it is being interpreted. For example, if a loop is performed 10,000 times, the lines within the loop are translated 10,000 times.

The distinction between a compiler and an interpreter is not completely clear-cut, however. Some languages, such as **Java**, are compiled into an intermediate stage called **bytecode**. This can be interpreted and run on many different types of processor using an appropriate bytecode interpreter.

JavaScript, which is used in creating web pages, is interpreted; the source code is included in the web page and then interpreted in the browser.