Programming languages

The instructions a CPU actually executes need to be in **machine code**

- series of binary values only
- very simple steps
- each processor has its own version of machine code

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

Programming languages

Problems with machine code

- difficult for humans to use - just 1s and 0s
- no real human syntax
- depends on the processor (or processor family) you have
- steps are too simple to program efficiently

But it executes v quickly as no **translation** is needed

Programming languages

Assembly code is a low level programming language

- one assembly code line = 1 machine code instruction (1:1 correspondence)
- easier for humans to use - has a basic syntax
- has to be **assembled** to convert it to machine code - using an **assembler**

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

Programming languages

Each family of processors has its own version of assembly code. This means assembly code can't necessarily be moved between processor types and might have to be re-written if the processor changes.

Each processor has **assemblers** to convert (translate) assembly code to machine code.

Some assemblers can work on multiple processors.

Programming languages

Instructions written in assembly code are very **efficient**. They are written **directly** for the processor involved and so can make the best use of hardware components.

This means that assembly code:

- uses less memory
- runs quicker

IMPORTANT

Programming languages

Traditionally assembly code was used to program **embedded systems**

This isn’t so common now - a fairly simple high-level language such as C is more commonly used now

But assembly code is still useful for programming older processors that would otherwise become obsolete

Programming languages

Machine code and assembly code are **low level programming languages**

- one line of code = one processor instruction
- work at a machine level
- difficult for humans to read/write

Programming languages

High-level programming languages are more similar to human language

- easier for humans to understand and code
- easier to debug
- can be used on a wide range of processors
- need to be translated to machine code to run using an interpreter or compiler

Programming languages

High-level languages have many advantages:

- easier for humans to code and debug
- fewer lines of code needed
- can be used on a wide range of processors
- more useful structures - iteration, selection, arrays, subroutines etc…
- easier to structure code
- more support and help and built-in functions and libraries
- easier to maintain someone else’s code