Assembly Language

Assembly language is a low-level programming language

- limited set of instructions
- used to program directly
  - embedded systems
  - hardware components
- limited structures - iteration, selection, arrays, subroutines etc...
- each instruction equates to one line of machine code (one CPU instruction)

Assembly Code

Data at held MEM8921 = 6

Data held at MEM8922 = 7

```
MOVE R2, 0
MOVE R0, MEM8921
MOVE R1, MEM8922
CMP R1, 0
```

(IS R1=0? IF **NOT** DO THIS)

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

STORE MEM8923, R2
HALT
```

Note: you **don't** need to know the syntax of Assembly Code instructions, just the theory of how it works.

Assembly Language

Key point:

- each instruction equates to one line of machine code (one CPU instruction)

This is a **1:1 equivalence** (or correspondence)

It means that assembly code is quick to convert to machine code, so programs run efficiently - so it’s quicker

Assembly Language

Programs written in assembly code must still be translated into machine code

Not all forms of assembly language will work with all processors - remember, machine code is specific to the processor

Assembly Language

Why use assembly code:

- 1:1 equivalence so quick to convert and process
- suitable for simple processes in embedded systems
- efficient code - important in very small systems without lots of memory
- processor specific, so can be tailored to a specific task