PROCESSOR INSTRUCTION SET

Instruction set

The patterns of 0s & 1s that a particular processor recognises as commands, along with their associated meanings. Each processor will have its own. Instruction sets are classified as either *RISC* or *CISC* (Reduced or Complex).

Machine Code Instruction ←→ Assembly Code Opcode

The instructions are coded in binary (Machine Code), they are typically divided in: operation code (op-code) and operand part.

Assembly language exists because it is easier to write for humans, then an assembler program will translate in machine code.

High-level languages are translated into machine code before they are executed.

Operation Code	**Operand Part**
Machine code	
0001 0001	0000 1111
Assembly language	
Add (direct)	15

Opcode: an operation code or instruction used in assembly language. Opcodes are sub-divided as Operation & Addressing mode.

Operand: a value or memory address that forms part of an assembly language instruction.

Addressing mode: the way in which the operand is interpreted. It can be:

- *Direct Address:* operand is a Memory address (in assembly language just the value)
- *Immediate Address:* operand is a Data Value (in assembly language a # before the value)

Assemble language

A way of programming using *mnemonics*, short codes that are used as instructions in programming (e.g. LDR = Load into Register, ADD).

Example:

Machine Code	**Using AQA's - Assembler equivalent**	
0001 0000 0000 0011	LOAD #3	Load 3 into acc
1000 0000 0000 1101	STORE 13	Store acc in address 13
0001 0000 0000 0110	LOAD #6	Load 6 into acc
0100 0000 0000 1101	ADD 13	Add data in address 13 to acc
1000 0000 0000 1110	STORE 14	Store acc in address 14

Assembly language invented to enable us humans to read/write code, 1-1 mapping between an assembler instruction & a machine instruction

Types of Operation Codes

- *Data processing*
  - Arithmetic operations: op. within an instruction set that perform basic maths, such as add and subtract
  - Shift Instructions: op. within an instruction set that move bits within a register
  - Logical operations: op. within an instruction set that move bits within an operand
- *Data transfer:* op. within an instruction set that move data around between the registers and memory
- *Control*
  - Branch operations: op. within an instruction set that allow you to move from one part the program to another