

# Assembly language introduction

## Revision of hexadecimal

4096	256	16	1
1	0	0	6

$$\hookrightarrow 1006_{16} = \underline{\hspace{2cm}}_{10}$$

$$AB_{16} = \underline{\hspace{2cm}}_{10}$$

Something to think about :

is the following a list of numbers (in hex) ?  
or a computer program ? or both ?

1006

3007

2008

7000

0000

0000

0009

000C

we will come back to this later (compare with assembled version of SimpleAdd.was).

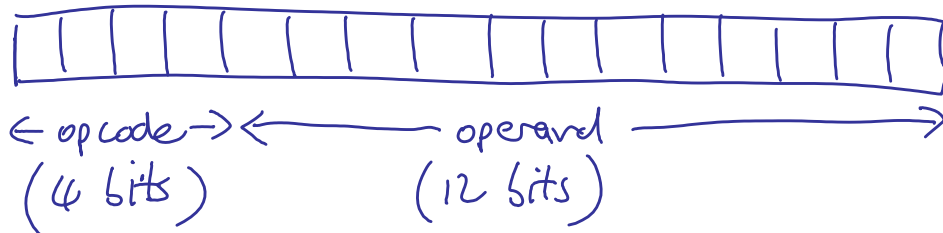
## Instruction set architecture (ISA)

- specifies the machine language of a CPU
- includes
  - the format of the instructions
  - a description of what each instruction does

Typically, any machine instruction contains

- some bits that represent an opcode (a number identifying what operation the instruction performs)
- some bits representing the arguments (or operands) for the instruction.

e.g. on the MARIE architecture introduced below, the format is:



The opcode for ADD is 3 (or 0011 in 4-bit binary) and ADD takes a single operand that is a 12-bit address. So the instruction ADD 27 is:

in binary: 0011 0000 0001 1011  
in hex: 3 0 1 B

Labels: "opcode" points to the first two hex digits (30), and "operand" points to the last two hex digits (1B).

The meaning of "ADD 27" is defined to be "add the value in memory location 27 to the current value in the accumulator register, and store the result in the accumulator register."

Common instructions include LOAD, STORE, JUMP, ADD, MUL.

↑  
jump to another location in the program

↑  
multiply

We study the ISA of a very simple computer called MARIE.

MARIE:

- has 16-bit words
- memory is word-addressable, and contains 4K words (i.e. 4096 words).
- instructions consist of a 4-bit opcode and a single 12-bit operand

[Demos: use simulator to see LOAD, STORE, ADD, JUMP

- SimpleAdd.mas and SimpleAddJump.mas show these
- show all relevant registers on whiteboard, trace contents

]

Activity: type in and execute the following program:

```
load 0  
store 5  
halt
```

- can you explain the result? What does this remind us about the key insight behind the von Neumann architecture?

Answer: instructions and data are indistinguishable  
- both are stored in memory. Any memory location can be regarded as instruction, data, or both.

Exercise:

a) Convert LOAD 03E into (i) hex  
(ii) binary

b) Convert the machine language instruction hex 4105 into assembly.

Solutions a) i) 103E  
ii) 0001 0000 0011 1110  
b) SUBT 105

Instructions often require several micro-operations.

The micro-operations needed for a particular instruction are written in register transfer language (RTL).

See textbook section 4.8.4 for examples.

The details of the fetch-decode-execute cycle are given in book figure 4.11

Minimal: modify SimpleAdd.mas to add 5 numbers of your choosing and store the result.

Challenge exercise: Write out the RTL for a hypothetical instruction that doubles the contents of a given memory location