# Instruction Set Architecture (ISA)

In Systems Organisation 1, we built a computing system from the bottom up, focusing on the hardware, and manipulation of bits using logic.

We will now look at the same problem from a software perspective.

## Outline

### Machine instructions and program execution

An instruction in the machine corresponds to a pathway through the machine.  
E.g. a pathway between the operands of an ADD operation and the piece of hardware that does the addition, and then from that hardware to a place to store the result.

### Addressing methods for data operands

We may want to refer to data in memory or a register, or data that doesn't yet exist when the program starts.

The addressing modes allow us to access data that's in different places in our machine.

### Assembly-language representation for instructions, data and programs

User-friendly labels and languages.

### Stacks and subroutines

A subroutine is a small sub-program, e.g. function calls. The stack allows us to do this.

Things can be computed without a stack, but it's needed practically for programming.

## Memory organisation

Memory is made up of millions of cells. Each one holds a bit, 0 or 1.

A word is a group of n bits. Word length can be 16 to 64 bits.

Memory, then, is a collection of consecutive words of the size specified by the word length.

## Word and byte encoding

A common word length is 32 bits. This word can store a 32-bit signed integer or four 8-bit bytes, e.g. ASCII characters.

For 32-bit integer encoding, bit b31 is the sign bit.

Words in memory may store data or machine instructions for a program.

Each machine instruction may require one or more consecutive words for encoding.

## Addresses for memory locations

To store or retrieve items of information, each memory location has a distinct address.

Numbers 0 to (2k - 1) are used as addresses for successive locations in the memory.

The 2k locations make up the address space.

The memory size is set by k, the number of address bits. When k = 32, we have about 4 billion locations.

## Byte addressability

Bytes are always 8 bits, but word length may range from 16 to 64 bits.

It's impractical to assign an address to each bit, so instead the memory is byte-addressable, with each byte having an address.

Byte locations have addresses 0, 1, 2, … Assuming a word length of 32 bits, word locations have addresses 0, 4, 8, …

## Big- and little-endian addressing

Two ways to assign byte addresses across words.

Big-endian assigns lower addresses to more significant (leftmost) bytes of the word. Little-endian goes the opposite way.

Commercial computers use either approach, and some can support both.

The addresses for 32-bit words are still 0, 4, 8, …

## Word alignment

Numbers of bytes per word is normally a power of 2.

Word locations have aligned addresses if they begin at byte addresses that are multiples of the number of bytes in a word.

Some computers permit unaligned addresses.