INSTRUCTION SET ARCHITECTURE (ISA):

An Instruction Set Architecture (ISA) is **part of the abstract model of a computer that defines how the CPU is controlled by the software**. The ISA acts as an interface between the hardware and the software, specifying both what the processor is capable of doing as well as how it gets done.

Two types of instruction set architectures are

The two main categories of instruction set architectures, **CISC (such as Intel'sx86 series) and RISC (suchas ARM and MIPS)**,

The ISA of a processor can be described using 5 catagories:

Operand Storage in the CPU

Number of explicit named operandsOperand location

Operations

Type and size of operands

The 3 most common types of ISAs are:

1. ***Stack*** - The operands are implicitly on top of the stack.
2. ***Accumulator*** - One operand is implicitly the accumulator.
3. *General Purpose Register (GPR)* - All operands are explicitly mentioned,they are either registers or memory locations

Stack	**Accumulator**	**GPR**
PUSH A	LOAD A	LOAD R1,A
PUSH B	ADD B	ADD R1,B
ADD	STORE C	STORE R1,C
POP C	-	-

The i8086 has many instructions that use implicit operands although it has a general register set. The i8051 is another example, it has 4 banks of GPRs but most instructions must have the A register as one of its operands.

Stack

Advantages: Simple Model of expression evaluation (reverse polish). Short instructions.

Disadvantages: A stack can't be randomly accessed This makesit hard to generate eficient code. The stack itself is accessed every operation and becomes a bottleneck.

Accumulator

Advantages: Short instructions.

Disadvantages: The accumulator is only temporary storage so memory traffic isthe highest for this approach.

GPR

Advantages: Makes code generation easy. Data can be stored for long periods inregisters.

Disadvantages: All operands must be named leading to longer instructions.

Reduced Instruction Set Computer (RISC):

RISC stands for Reduced Instruction Set Computer. The ISA is composed ofinstructions that all have exactly the same size, usually 32 bits. Thus they can be pre-fetched and pipelined successfully. All ALU instructions have 3 operands which are only registers. The only memory access is through explicit LOAD/STORE instructions.

Thus C = A + B will be assembled as:

```
LOAD  R1,A
LOAD  R2,B
ADD
R3,R1,R2
STORE C,R3
```

Although it takes 4 instructions we can reuse the values in the registers.

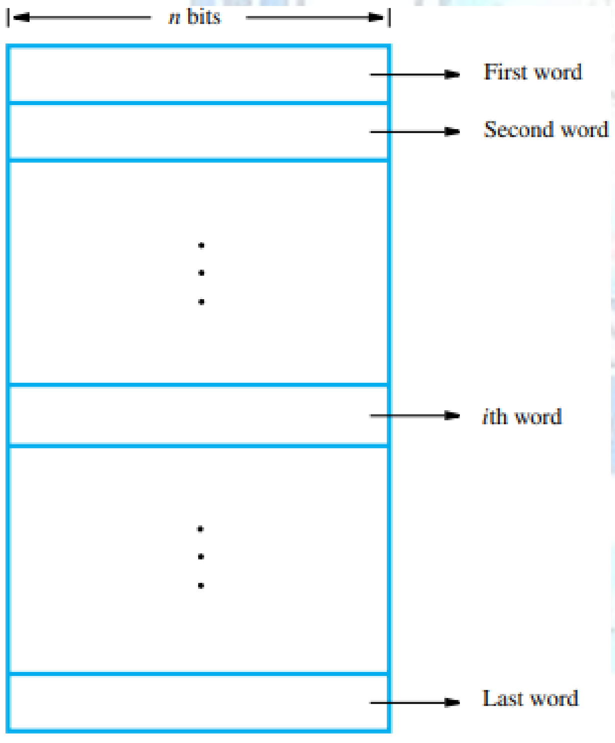
Complex Instruction Set Architecture (CISC) :

The main idea is that a single instruction will do all loading, evaluating, and storing operations just like a multiplication command will do stuff like loading data, evaluating,and storing it, hence it's complex

Memory Locations and Addresses

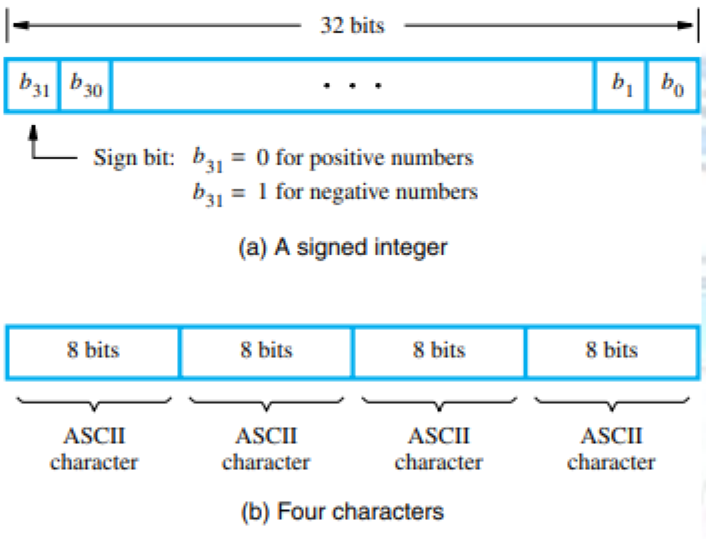
The memory consists of many millions of storage cells, each of which can store a bit of information having the value 0 or 1. Because a single bit represents a very small amount of information, bits are seldom handled individually.

The usual approach is to deal with them in groups of fixed size. For this purpose, the memory is organized so that agroup of n bits can be stored or retrieved in a single, basic operation. Each group of n bits is referred to as a word of information, and n is called the word length. The memory of a computer can be schematically represented as a collection of words.



Memory words.

Modern computers have word lengths that typically range from 16 to 64 bits. If the word length of a computer is 32 bits, a single word can store a 32-bit signed number or four ASCII-encoded characters, each occupying 8 bits, as shown in Figure



Examples of encoded information in a 32-bit word.

A unit of 8 bits is called a byte. Machine instructions may require one or more wordsfor their representation.

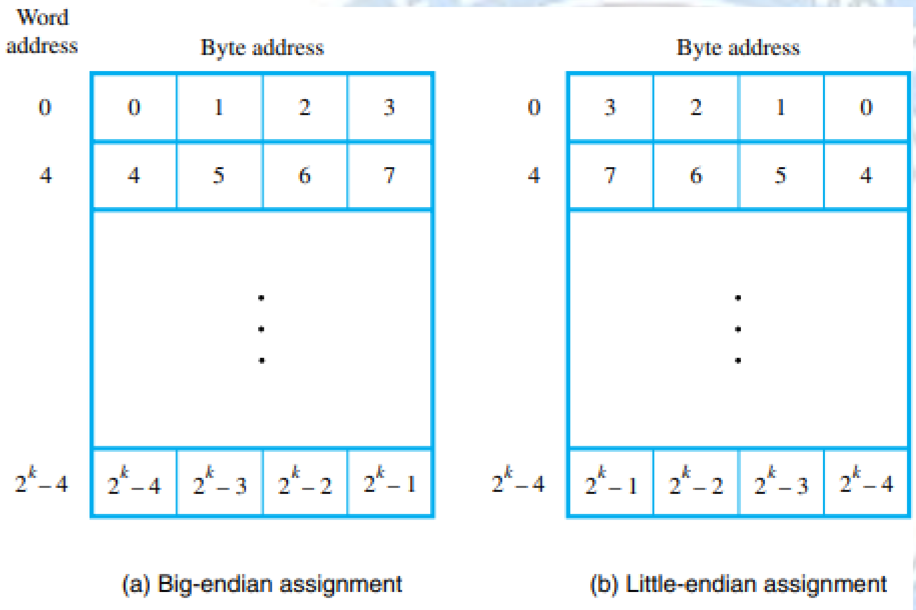
After we have described instructions at the assembly-language level. Accessing the memory to store or retrieve a single item of information, either a word or a byte, requires distinct names or addresses for each location. It is customary to use numbers from 0 to 2k – 1, for some suitable value of k, as the addresses of successive locations in the memory. Thus, the memory can have up to 2k addressable locations. The 2k addresses constitute the address space of the computer. For example, a 24-bit address generates an address space of 224 (16,777,216) locations. This number is usually written as 16M (16 mega), where 1M is the number 220 (1,048,576). A 32-bit address creates an address space of 232 or 4G (4 giga) locations, where 1Gis 230. Other notational conventions that are commonly used are K (kilo) for the number 210 (1,024), and T (tera) for the number 240

Byte Addressability :

A byte is always 8 bits, but the word length typically ranges from 16 to 64 bits.It is impractical to assign distinct addresses to individual bit locations in the memory. The most practical assignment is to have successive addresses refer to successive byte locations in the memory.

This is the assignment used inmost modern computers. The term byte-addressable memory is used for this assignment. Byte locations have addresses 0, 1, 2,. Thus, if the word length ofthe machine is 32 bits, successive words are located at addresses 0, 4, 8,. , with each word consisting of four bytes.

There are two ways that byte addresses can be assigned across words **big-endian** and **Little endian**


Word address
Byte address
Byte address
0
4
2^k-4
0 1 2 3
4 5 6 7
2^k-4 2^k-3 2^k-2 2^k-1
3 2 1 0
7 6 5 4
2^k-1 2^k-2 2^k-3 2^k-4
(a) Big-endian assignment
(b) Little-endian assignment

Byte and word addressing.

The name big-endian is used when lower byte addresses are used for the more significant bytes (the leftmostbytes) of the word.

The name little-endian is used for the opposite ordering, where the lower byte addresses are used for the less significant bytes (the rightmost bytes) of the word. The words "more significant" and "less significant" are used in relation to the weights (powers of 2) assigned to bits when the word represents a number. Both little- endian and big-endian assignments are used in commercial machines. In both cases, byte addresses 0, 4, 8,..., are taken as the addresses of successive words in the memory of a computer with a 32-bit word length. These are the addresses used when accessing the memory to store or retrieve a word.

Memory Operations

Both program instructions and data operands are stored in the memory. To execute an instruction, the processor control circuits must cause the word (or words) containing the instruction to be transferred from the memory to theprocessor. Operands and results must also be moved between the memory and the processor.

Thus, two basic operations involving the memory are needed, , namely Read andWrite.

Read Operation:

The Read operation transfers a copy of the contents of a specific memory location to the processor. The memory contents remain unchanged. To start a Read operation, the processor sends the address of the desired location to the memory and requests that its contents be read. The memory reads the data stored at that address and sends them to the processor.

Write Operation:

The Write operation transfers an item of information from the processor to a specific memory location, overwriting the former contents of that location. To initiate a Write operation, the processor sends the address of the desired locationto the memory, together with the data to be written into that location. The memory then uses the address and data to perform the write.

Instructions and Instruction Sequencing

The tasks carried out by a computer program consist of a sequence of small steps, such as adding two numbers, testing for a particular condition, reading a character from the keyboard, or sending a character to be displayed on a display screen.

A computer must have instructions capable of performing four types of operations:

- Data transfers between the memory and the processor registers
- Arithmetic and logic operations on data
- Program sequencing and control
- I/O transfers

We begin by discussing instructions for the first two types of operations. To facilitate the discussion, we first need some notation

Register Transfer Notation

We need to describe the transfer of information from one location in a computer to another. Possible locations that may be involved in such transfers are memory locations, processor registers, or registers in the I/O subsystem.

Assembly-Language Notation:

To represent machine instructions and programs we use assembly – Language notation .