MEMORY

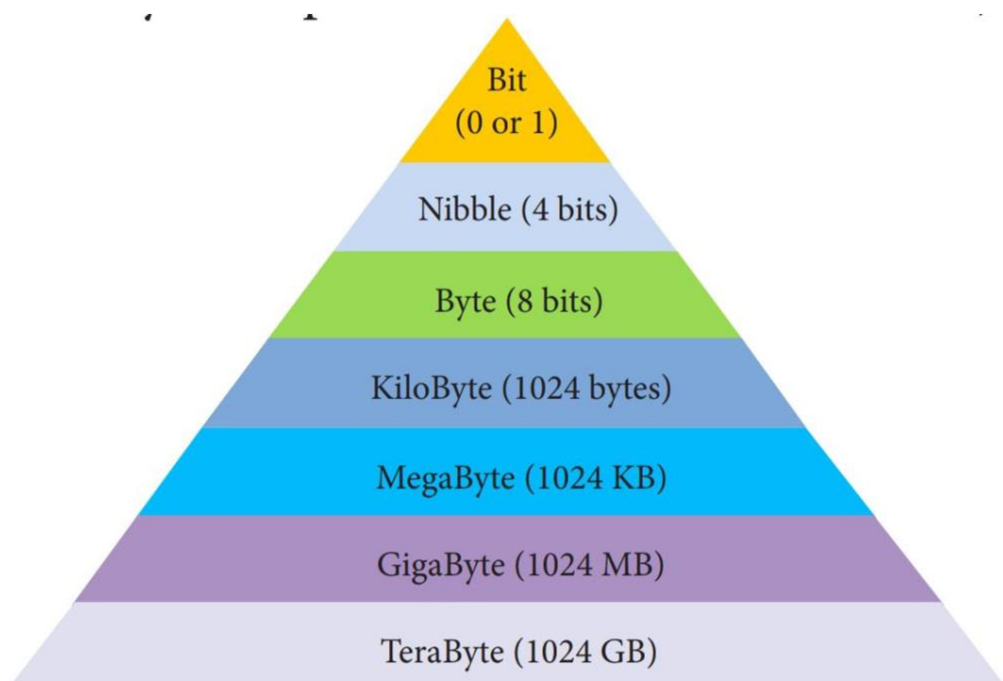
1. MEMORY

Memory devices are digital systems that store data either temporarily or for a long term. Digital computers to hard disks have built-in memory devices that can store the data of users or manufacturers. The data either be in the form of control programs or programs that boot the system. Hence, to store such a huge amount of data the memory devices must have enormous capacity. The challenge is to build memory devices that have large capacities but are cost-effective. The memory devices must be capable of storing both permanent data and instantaneous data.

Memories are made up of registers. Each register in the memory is one storage location. The storage location is also called a memory location. Memory locations are identified using Address

A memory unit consists of data lines, address selection lines, and control lines that specify the direction of transfer.

Units of Memory:



Name	Abbr.	Size
Kilo	K	2^10 = 1,024
Mega	M	2^20 = 1,048,576
Giga	G	2^30 = 1,073,741,824
Tera	T	2^40 = 1,099,511,627,776
Peta	P	2^50 = 1,125,899,906,842,624
Exa	E	2^60 = 1,152,921,504,606,846,976
Zetta	Z	2^70 = 1,180,591,620,717,411,303,424
Yotta	Y	2^80 = 1,208,925,819,614,629,174,706,173

Types of Computer Memory

Cache memory. This temporary storage area, known as a cache, is more readily available to the processor than the computer's main memory source. It is also called CPU memory because it is typically integrated directly into the CPU chip or placed on a separate chip with a bus interconnect with the CPU.

RAM. It is one of the parts of the Main memory, also famously known as Read Write Memory. Random Access memory is present on the motherboard and the computer's data is temporarily stored in RAM. As the name says, RAM can help in both Read and write.

D RAM (Dynamic RAM): D RAM uses capacitors and transistors and stores the data as a charge on the capacitors. They contain thousands of memory cells. It needs refreshing of charge on capacitor after a few milliseconds. This memory is slower than S RAM.

S RAM (Static RAM): S RAM uses transistors and the circuits of this memory are capable of retaining their state as long as the power is applied. This memory consists of the number of flip flops with each flip flop storing 1 bit. It has less access time and hence, it is faster.

ROM: ROM full form is Read Only Memory. ROM is a non volatile memory and it is used to store important information which is used to operate the system. We can only read the programs and data stored on it and can not modify of delete it.

MROM(Masked ROM): Hard-wired devices with a pre-programmed collection of data or instructions were the first ROMs. Masked ROMs are a type of low-cost ROM that works in this way.

PROM (Programmable Read Only Memory): This read-only memory is modifiable once by the user. The user purchases a blank PROM and uses a PROM program to put the required contents into the PROM. Its content can't be erased once written.

EPROM (Erasable Programmable Read Only Memory): EPROM is an extension to PROM where you can erase the content of ROM by exposing it to Ultraviolet rays for nearly 40 minutes.

EEPROM (Electrically Erasable Programmable Read Only Memory): Here the written contents can be erased electrically. You can delete and reprogramme EEPROM up to 10,000 times. Erasing and programming take very little time, i.e., nearly 4 -10 ms(milliseconds). Any area in an EEPROM can be wiped and programmed selectively.

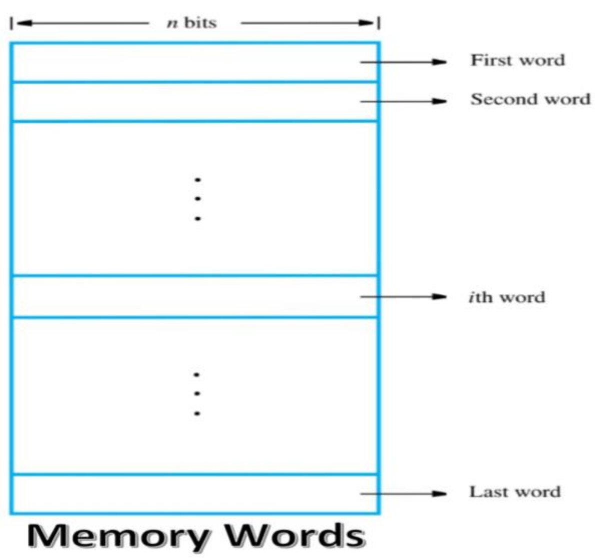
Virtual memory. A memory management technique where secondary memory can be used as if it were a part of the main memory. Virtual memory uses hardware and software to enable a computer to compensate for physical memory shortages by temporarily transferring data from RAM to disk storage.

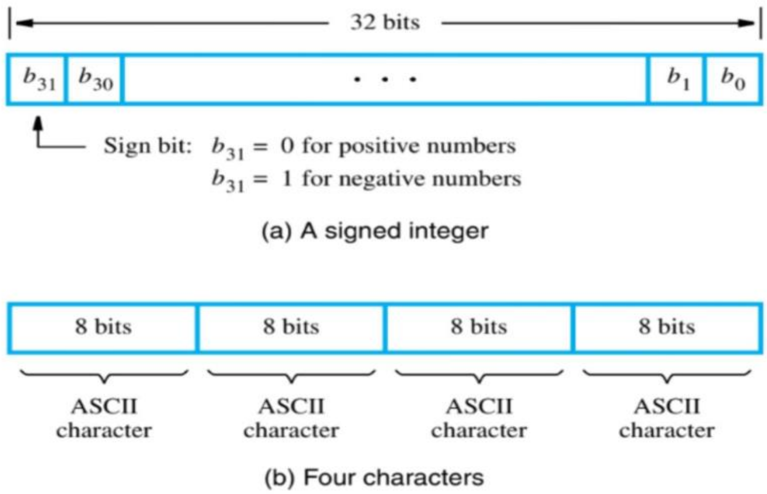
Memory Organizing

• Memory is organized into groups of n bits known as words, Where n is the word length.

• Modern computers typically have word lengths ranging from 16 To 64 bits.

• A 32-bit word can store a 32-bit signed number or four ASCII- encoded characters, each occupying 8 bits





Memory Address:

A memory address is a unique identifier that points to a specific location in a computer's memory. It's like a street address that tells you where a particular house is located in a city. In computing, memory addresses are used to locate and access data stored in memory.

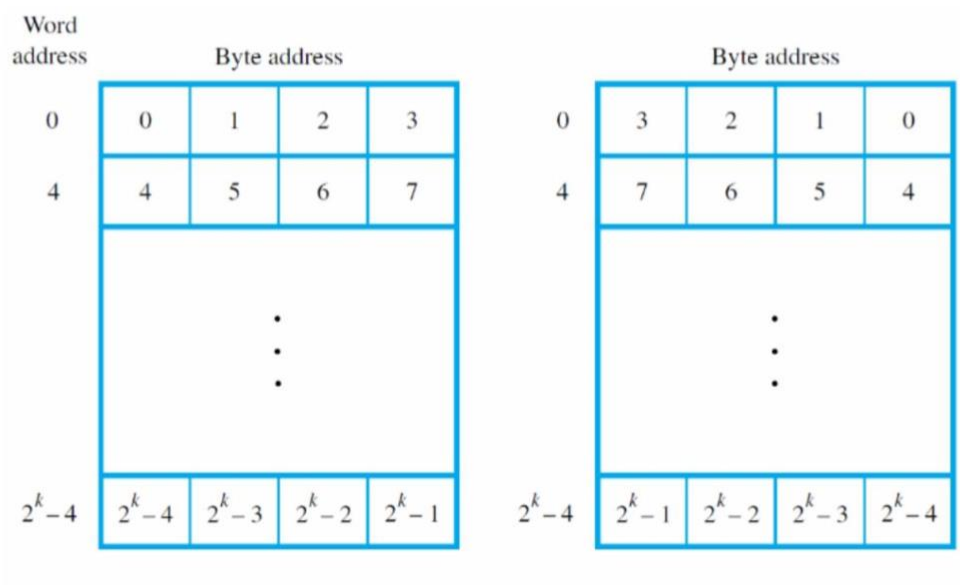
Each byte of memory in a computer has its own unique address. These addresses are typically represented in hexadecimal or binary notation. For example, in a 32-bit system, memory addresses might range from 0x00000000 to 0xFFFFFFFF.

Programs use memory addresses to read data from or write data to specific locations in memory. This allows them to manipulate variables, arrays, objects, and other data structures effectively. Understanding memory addresses is crucial for tasks such as memory management, pointer manipulation, and low-level programming.

Big endian and little endian addressing

Big Endian: In big endian format, the most significant byte (the byte with the highest value) of a multi-byte data type is stored at the lowest memory address, while the least significant byte is stored at the highest memory address. It's like writing numbers from left to right, with the most significant digit on the left.

Little Endian: In little endian format, the least significant byte is stored at the lowest memory address, and the most significant byte is stored at the highest memory address. It's like writing numbers from right to left, with the least significant digit on the left.



(a) Big-endian assignment (b) Little-endian assignment

Memory operations

Memory operations encompass a variety of tasks related to manipulating data stored in a computer's memory. Here are some common memory operations:

1. Read: This operation involves retrieving data from a specific memory location and transferring it to a processor register or another location in memory.

2. Write: Writing to memory involves storing data from a processor register or another memory location into a specific memory address.

3. Allocate: Allocating memory involves reserving a block of memory for storing data. This is commonly done dynamically during program execution using functions like malloc() in C or new operator in languages like C++ and Java.

4. Free: Freeing memory deallocates previously allocated memory, making it available for other uses. This is important to prevent memory leaks and manage memory efficiently.

5. Copy: Copying data from one memory location to another involves reading data from one location and writing it to another.

6. Move: Moving data involves transferring data from one memory location to another while potentially modifying its original location.

7. Compare: Comparing memory involves examining the contents of two memory locations to determine if they are equal, greater than, or less than each other.