Memory (I)

Overview
- We’ve known that memory is a critical component of modern computers that is to store data and program.
- When you buy a computer, memory is also an important factor you would like to consider, as it impacts the speed, capacity and price of a computer.

Characteristics of Memory Systems
- Location
  - Depending on whether memory is internal or external to a computer.
  - Internal memory is often equated with main memory.
    - Other forms of internal memory: registers (provide temporarily storage for processor), cache
    - External memory consists of peripheral storage devices that are accessible to the processor via I/O controllers.

- Capacity Unit
  - External memory capacity is typically expressed in terms of bytes.
  - Capacity for internal memory is typically expressed in terms of bytes (1 Byte = 8 bits) or words.
    - Words is the natural data size for a processor, typically based on processor’s data bus width (i.e. the width of an integer or an instruction). In another word, a word refers to how many bits the CPU can process at a time.
    - The size of a word varies from machine to machine. New machines have words of 32 or 64 bits. Old ones had 8 or 16 bit words.
    - In some systems, the addressable unit is the word. However, most modern computers allow addressing at the byte level. This says a 16-bit word processor holds two addresses per word if the computer is byte addressable. The relationship between the length in bits (A) of an address and the number (N) of addressable units is $2^A = N$.
  - Example:
    - Consider a real computer with 512 MB of RAM. How many bits are needed to form the 512 M addresses? $[2^n = 512 \text{ M, so } n = 29]$ 
    - So such a computer should have an address bus of 29 wires in order to send address all at once. But that is not the usual size of an address bus. Why? What should be the address bus size? [32 bits to allow for addition of more memory]
- **Unit of transfer**
  - The number of bits read out of or written into memory at a time.
  - **Internal memory**: usually governed by data bus width, i.e., a word
  - **External memory**: usually measured in terms of blocks. A block is much larger than a word.

- **Method of access**
  - Based on the hardware implementation of the storage device
  - **Sequential access**
    - Start at the beginning and read through in order
    - Access time depends on location of data and previous location (Access time: the time between “requesting” a data and getting it)
    - *Example: tape*
  - **Direct access**
    - Individual blocks have unique address
    - Access is by jumping to vicinity then performing a sequential search (like find your car in parking lot, you know the region, but need to search in that region)
    - Access time depends on location of data within “block” and previous location
    - *Example: hard disk*
  - **Random access**
    - Individual addresses identify locations exactly
    - Access time is consistent across all locations and is independent of previous access (time between putting the address on bus and getting data)
    - *Example: RAM*
  - **Associative access**
    - Addressing information must be stored with data in a general data location
    - A specific data element is located by comparing desired address with address portion elements in the general data location
    - Access time is independent of location or previous access (time it takes to search through address information associated with data to determine “hit.” Done with hardware/logic and is predictable.)
    - *Example: cache*

---

**Big Endian vs. Little Endian**
- We know data are stored in memory. But not all memories store data in the same way.
- Memory can be thought as one large array containing bytes
- Using “address” to refer to the array location
- Each address store one element, which is typically one byte (byte-addressable)
- To store 32-bit integer, like 5, we need 4 bytes so 4 slots of RAM array of bytes. Should the rightmost byte 0000 0101 be at the first or fourth byte of the 4 bytes?
  - **Big Endian**: stores the leftmost byte in the lowest numerical byte address, so 0000 0101 is in the 4th byte. 5 is stored as 0000 0000 0000 0000 0000 0000 0000 0101 in memory.
  - **Little Endian**: stores the rightmost byte in the lowest numerical byte address, so 0000 0101 is in the 1st byte. 5 is stored as 0000 0101 0000 0000 0000 0000 0000 0000 in memory.
- Note that Intel uses little-endian but just about everyone else use big-endian
- Can create problems if transferring data between machines in binary over a network or on a disk.
- New CPUs can handle both using a switch to go back and forth; Network protocol (ANSI coupled with Basic Encoding Rules) can handle the data transmission between machines.

---

**Memory Hierarchy (I)**

- A typical hierarchy

![Image of memory hierarchy](image)

- As one goes down the hierarchy, the following occur:
  (a) Decreasing cost per bit
  (b) Increasing capacity
  (c) Increasing access time (the time between “requesting” data and getting it)

- To design an efficient memory, we want memory as fast as possible. However, it’s inappropriate to only use fast memory. We need to consider the trade-offs among three key characteristics:
• Amount: Software will ALWAYS fill available space
• Speed: Memory should be able to keep up with the processor (as the processor is executing instructions, we would not want it to have to pause eating for instructions or operands.)
• Cost: Whatever the market will bear

- To balance these three characteristics, we can leverage the memory hierarchy.
  • Use larger, cheaper, slower memories to supplement smaller, more expensive, faster memories.
  • To make the memory hierarchy system be able to keep up with the processor, a key condition is required.
    - Decreasing frequency of access of the memory by the processor.
  - The basis for the validity of the condition is a principle known as locality of reference.