

Memory (I)

Overview

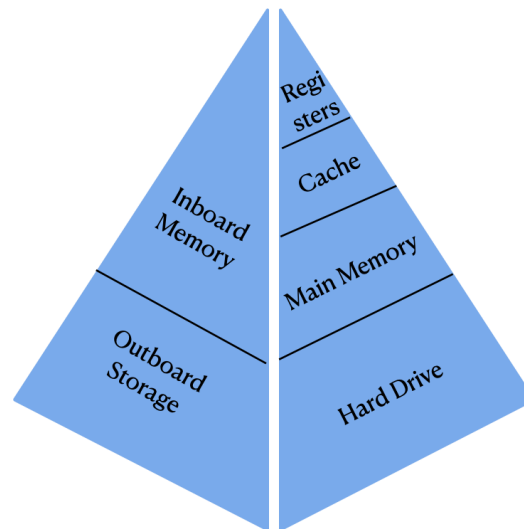
- We've known that memory is a critical component of modern computers that is to store data and program.
 - When you buy a computer, memory is also an important factor you would like to consider, as it impacts the speed, capacity and price of a computer.
 - How do we get an efficient memory system? How does the memory system of a computer work?
-

Characteristics of Memory Systems

- Location
 - Depending on the location of memory, the two types of memory are.
 - **Inboard memory** is often equated with **main memory**.
 - Other forms of internal memory: **registers** (provide temporarily storage for processor), **cache**
 - **Outboard memory** often refers to the hard drive .
- Capacity Unit
 - Two commonly used capacity units are **bytes** and **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**. 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]

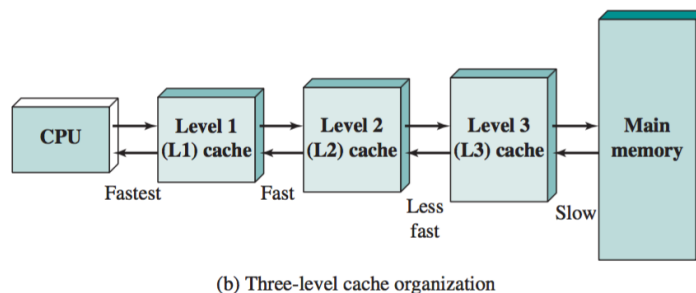
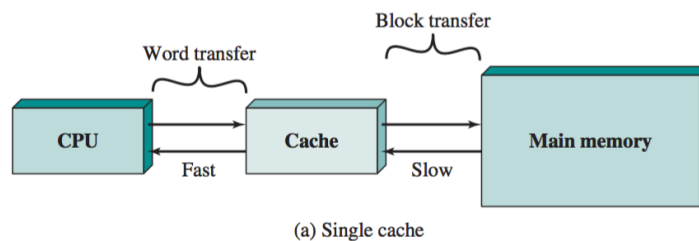
Memory Hierarchy (I)

- A typical hierarchy



- 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 waiting 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.

- **Locality of reference**
 - Due to the nature of programming, **instructions and data tend to cluster together** (loops, subroutines, and data structures)
 - Over a long period of time, clusters will change
 - Over a short period, cluster will tend to be the same
 - Accordingly, it is **possible to organize data across the hierarchy**, such that the **percentage of access to each successively lower level is substantially less** than that of the level above.
- Breaking memory into levels
 - Assume a hypothetical system has two levels of memory (e.g., IAS machine)
 - Two levels: registers + main memory
 - Level 2 should contain all instructions and data
 - Level 1 doesn't have room for everything, so when a new cluster is required, the cluster it replaces must be sent back to the level 2
 - This principle can be applied across more than two levels of memory
 - Three levels: registers + L1 cache + main memory
 - Four levels: registers + L1 cache + L2 cache + main memory
 - Five levels: registers + L1 cache + L2 cache + L3 cache + main memory



- Performance:
 - Example: Given a two-level memory, assume the access time of the faster level is 0.01 μs (microsecond, 10^{-6} s) and the access time of the slower level is 0.1 μs . If 95% of the memory accesses are found in the faster level, what the average access time might be?

$$- 0.95 * 0.01 + 0.05 * (0.01 + 0.1) = 0.0095 + 0.0055 = 0.015 \text{ us}$$

- What affects the average access time?
 - **Hit ratio**: the ratio that the accessed word is found in the faster memory
 - Level 1 access time T_1 , level 2 access time T_2 .

