Chapter 1

Day 25

1.1 Data Storage

Example 1.1 (Hardware used for Data Storage)
→ **Data storage technologies can be broken into the following categories:**

Disks

1. Hard Disk (HDD)
2. Solid State Drive (SSD)

Differences Between the two disk types:

- SDD is faster and smaller than HDD
- SDD is also more expensive than HDD

Random Access Memory (RAM):

1. Dynamic RAM (DRAM)
2. Static RAM (SRAM)

Registers, Such as:

- rax
- rbx
- rcx
- rdx
- rsi (*etc*)

→ **In terms of storage access time, the following is true:**

- Disks are the slowest in terms of access time.
- SSD is marginally faster than disks.
- DRAM is faster than SSD by a large margin, but still slower than registers (CPU)

Definition 1.1: CPU-Memory gap

Although the CPU is the fastest component in a computer in terms of access time, it is still progressing in terms of access speed much faster than other types of data storage.

The problem we should be concerned about it the gap between the CPU and the memory creates idle time for the CPU as is it significantly faster than memory.

Need something to bridge the gap between the CPU and the memory (*hint: Caching*).

→ **Caching vs. Memory**

- Caching is smaller, faster, and more expensive; It caches a subset of the blocks of memory
- Memory is larger, slower, and cheaper; It is partitioned into blocks.

1.2 Caching

Definition 1.2: Caching

A smaller, faster storage to speed up larger, slower storage.

Note:-
SRAM is used for hardware caches, and DRAM is used for main memory.

→ **Main Topics in Caching:**

1. Cache Operation
2. Cache Design
3. Cache Effectiveness

1.2.1 Cache Operation

Example 1.2 (Cache Operation)
Uses fixed size data blocks. The size of the block is a parameter set for each system. The "block" is the size of unit transfer between the cache and memory (An entire block will be copied from memory to cache even if you only referenced a small amount of that memory)

Note:-
→ *Cache Block Size Comparison*
Smaller blocks would lead to more blocks in general in the cache.
However, larger blocks would lead to fewer cache misses.

For example, you might access memory from some address in the middle of a block, and then load it into the cache in the hopes of getting "cache hits" later on.

In this example, we are accessing both the cache and the memory, but the time it takes to access the cache is much faster than the time it takes to access the memory, so moving to cache is nearly negligible.

In the case of a cache "miss", will perform a cache eviction.

Definition 1.3: Cache Eviction

Changing something stored in the cache to something else stored in memory.

1.2.2 Cache Design

→ Design Goals for Caches:

- Maximize cache hit rate
- Minimize complexity and cost

Definition 1.4: Write-Miss

Types of write-misses:

1. write-allocate cache (bring updated block into cache)
2. no-write-allocate cache (leave cache alone)

Example 1.3 (Write-through Cache)
→ Write-Miss
On a write-hit, will update cache and memory immediately.

→ Write-Miss
Uses the no-write-allocate policy.

Example 1.4 (Write-back Cache)
→ Write-Hit
On a write-hit, will IGNORE memory, and only update the cache.

When evicting a block, if it a block that has been changed, then it will be different than the block saved in memory, so will update memory on eviction.

This will be preferable in the scenario that a cache is updated numerous times in a row, as it will not update memory on each access.

Deferring memory updates to a later time.

→ Write-Miss
Uses the write-allocate policy.

→ **Cache Placement Policy:**

- Direct-Mapped Cache: Each memory block is mapped to 1 code block, so each cache is tied to a certain part of the memory (given a piece of memory, it has an associated cache).

   1. easy to know if memory is in the cache
   2. easy to know which cache to use for some memory
   3. easy to implement
   4. If multiple pieces of memory are often being accessed, but are associated to the same cache, a *Conflict Miss* occurs.

- Fully-Associative Cache
- Set-Associative Cache