Memory Hierarchy
Where Memory really fits in

Reading from memory

1

Load operation: `movq A, %rax`

2

Load operation: `movq A, %rax`
Disks consist of **platters**, each with two **surfaces**.

- Each surface consists of concentric rings called **tracks**.
- Each track consists of **sectors** separated by **gaps**.
Capacity = (# bytes/sector) \times (\text{avg. # sectors/track}) \times (\# tracks/surface) \times (\# surfaces/platter) \times (\# platters/disk)

Example:
- 512 bytes/sector
- 300 sectors/track (on average)
- 20,000 tracks/surface
- 2 surfaces/platter
- 5 platters/disk

\[
\text{Capacity} = 512 \times 300 \times 20000 \times 2 \times 5 = 30,720,000,000 = 30.72 \text{ GB}
\]

\[\overline{T_{io}} = \overline{T_{seek}} + \overline{T_{rotate}} + \overline{T_{transfer}}\]
- **Given:**
  - Rotational rate = 7,200 RPM
  - Average seek time = 9 ms.
  - Avg # sectors/track = 400.

- **Derived:**
  - $T_{avg \ rotation} = \frac{1}{2} \times \left(\frac{60 \ secs}{7200 \ RPM}\right) \times 1000 \ ms/sec = 4 \ ms$.
  - $T_{avg \ transfer} = \frac{60}{7200} \ RPM \times \frac{1}{400} \ secs/track \times 1000 \ ms/sec = 0.02 \ ms$
  - $T_{access} = 9 \ ms + 4 \ ms + 0.02 \ ms$

- **Important points:**
  - Access time dominated by seek time and rotational latency.
  - First bit in a sector is the most expensive, the rest are free.
  - SRAM access time is about 4 ns/doubleword, DRAM about 60 ns
    - Disk is about 40,000 times slower than SRAM,
    - 2,500 times slower then DRAM.
CPU initiates a disk read by writing a command, logical block number, and destination memory address to a port (address) associated with disk controller.

Disk controller reads the sector and performs a direct memory access (DMA) transfer into main memory.

When the DMA transfer completes, the disk controller notifies the CPU with an interrupt (i.e., asserts a special "interrupt" pin on the CPU).
on to locality discussion
- **Cache**: A smaller, faster storage device that acts as a staging area for a subset of the data in a larger, slower device.

- **Fundamental idea of a memory hierarchy**:
  - For each k, the faster, smaller device at level k serves as a cache for the larger, slower device at level k+1.

- **Why do memory hierarchies work?**
  - Because of locality, programs tend to access the data at level k more often than they access the data at level k+1.
  - Thus, the storage at level k+1 can be slower, and thus larger and cheaper per bit.

- **Big Idea**: The memory hierarchy creates a large pool of storage that costs as much as the cheap storage near the bottom, but that serves data to programs at the rate of the fast storage near the top.
Cache

Data is copied in block-sized transfer units

Memory

Larger, slower, cheaper memory viewed as partitioned into “blocks”

Smaller, faster, more expensive memory caches a subset of the blocks

Request: 14

Cache

Request: 12

Cache

Request: 12

Block b is stored in cache

• Placement policy: determines where b goes
• Replacement policy: determines which block gets evicted (victim)