Memory Hierarchy
Where memory really fits in

Reading from memory

1.

2.
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) x (avg. # sectors/track) x (# tracks/surface) x (# surfaces/platter) x (# platters/disk)

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

Capacity = 512 x 300 x 20000 x 2 x 5
= 30,720,000,000
= 30.72 GB

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

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

- **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
**Example Memory Hierarchy**

- **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.
NB: Address is King

A. Interface to memory

CPU may deal in registers, but ultimately

\[
\text{mov} q <\text{name}> \quad \text{mov} q <\text{name}> \\
\text{yields an address to Bus}
\]

\[
\text{Instruction} \quad 0 - q \text{[mem][PC]}
\]

Locality: Defined relative to an address

temporal

spatial

Important for all to remember b/c Programs with good locality run faster than programs with poor locality

and it is in programmer’s control to favor locality

```c
sum = 0;
for (i = 0; i < n; i++)
    sum += a[i];
return sum;
```
Which is better for real-life, and why?

Does it really make a difference?

- **Cache memories** are small, fast SRAM-based memories managed automatically in hardware
  - Hold frequently accessed blocks of main memory
- **CPU looks first for data in cache**
- **Typical system structure:**
Block b is stored in cache

- **Placement policy:** determines where b goes
- **Replacement policy:** determines which block gets evicted (victim)
Define a cache organization as $(S, E, B)$ and need $m$: number of address bits

Address: $[ \underbrace{0}_{m-1} ]$

Derived from above:

$M = 2^m$: Maximum number of unique memory addresses

$s = \log_2(S)$: Number of bits to represent set

$b = \log_2(B)$: Number of bits to represent block offset

$C = S \times E \times B$ data bytes
1. Locate set
2. Check if any line in set has matching tag
3. Yes + line valid: hit
4. Locate data starting at offset
Example 1:
Direct mapped: One line per set
Assume: cache block size 8 bytes

S = \(2^s\) sets

Address of int:
\[
\begin{array}{c}
\text{t bits} \\
0...01 \\
100
\end{array}
\]

find set

valid? + match: assume yes = hit

Address of int:
\[
\begin{array}{c}
\text{t bits} \\
0...01 \\
100
\end{array}
\]

block offset

int (4 Bytes) is here