The Memory Hierarchy

Registers, caching, RAM, disks, swapping, networks…

The Memory Bottleneck

Registers are very, very fast:

Can be read/written as part of a single CPU cycle.

Main memory is up to **100x slower**!

Consider this code for doubling the value stored at address `1A0`:

```
Load 1A0
Add 1A0
Store 1A0
```

CPU has to wait with next instruction until memory has sent the data

But we’re using the data from 1A0 multiple times!

Solution: Caching

A **cache** sits between the registers and the RAM:

- Additional memory in the CPU
- Faster than RAM, slower than registers
- Smaller than RAM, larger than registers
- Keeps recently loaded values close to the CPU

Caching: Implementation

- Fully implemented in hardware
- “Transparent” for the program:

  `Load 1A0` loads from cache if possible, from RAM otherwise!

  `Store 1A0` may store only to cache (for later transfer to RAM), or both to cache and RAM, or only to RAM

Caching tricks

- Programs often access consecutive addresses, e.g.
  - searching through some text from start to finish
  - making all pixels in an image brighter
  - playing video or audio data
- When you `Load` a value that’s not yet in the cache, the cache will already load the next few values as well
- When cache is full, the “oldest” values are thrown out
  - Less likely to need those again

What makes caching tricky

- “Cache misses” can make a program go 100x slower!
- Performance of code
  - may be difficult to predict
  - may depend heavily on CPU cache architecture
- High-performance code may have to be written with cache in mind
- Remember: programs themselves are stored in RAM (and therefore also cached)

Dealing with Big Data

What to do when you don’t have enough RAM?

Swapping

- Write unused parts of RAM to disk
- Read back when required
- Almost opposite of caching
- Implemented by combination of hardware and operating system (*virtual memory*, more later)

Modern Memory Hierarchy

Memory	Speed	Size	Connection to CPU
Registers	Very fast	A few words	Inside CPU
L1 Cache	Fast (3-5x slower)	~100 KByte	Inside CPU
L2, L3 Cache	Fast (25-50x slower)	~1-8MBytes	Inside or close to CPU
RAM	Medium (50-100x slower)	Several GBytes	Main bus
Hard Disk, SSD	Slow (1000x slower)	Terabytes	External (SATA, USB)
Network	Very slow (10000x-infinity…)	?	External (PCIe, USB)

Summary

- Fast memory access is critical for code performance
- Accessing RAM is up to 100x slower than registers
- Cache keeps values around for fast access
    - Recently loaded values
    - Values close in RAM to recently used values
- Modern processors unthinkable without cache
- Swapping is opposite of caching: use disk to simulate more RAM than available

EOF