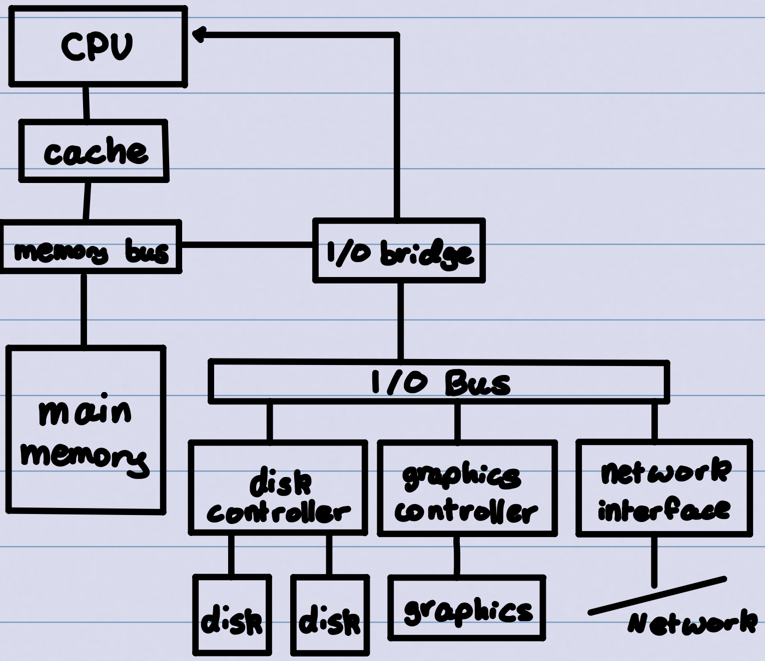
# Lecture 1 - 4th Sept 2024

Instruction Set Architecture: how hardware executes software

- What is a computer?
  - ↳ just a machine that does whatever the software tells it to do.
  - ↳ computer > digital circuits > logic gates > transistors
  - ↳ software is just a series of instructions.
  - ↳ The five classic components of a computer:
    - CPU: Central Arithmetic (ALU) and Central Control
    - Memory System
    - Input and output

↳ But, here's a more realistic view:



# Architecture vs Microarchitecture

↳ **Processor architecture:**

- Functional appearance to software (ISA)
  in this case, compilers!
  - Exactly what instructions does it have?
  - Number of memory / storage locations it has
  - Interface!

↳ **Processor microarchitecture:**

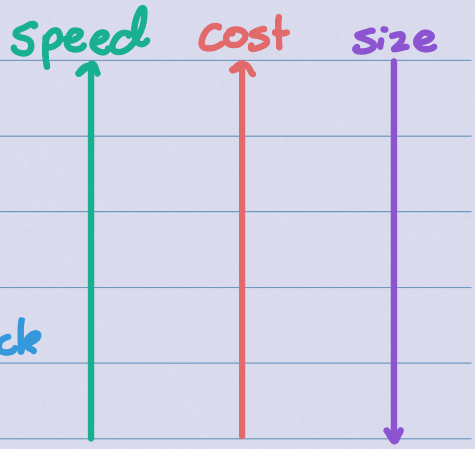
- Logical Structure that implements the architecture
  - Number of functional units, interconnection, control
  - Size of the caches
  - Not visible to the software
  - Implementation!

## Memory Hierarchy:

- CPU
- CPU Cache

→ processor

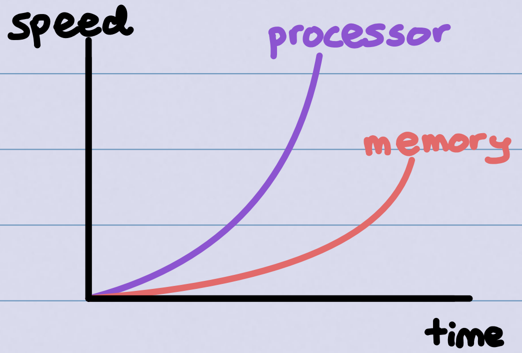
- Physical Memory → RAM
- Solid State Memory → SSD / USB stick
- Virtual Memory → hard drive



- **Moore's Law:** every 18-24 months,
  - 2x transistors on same chip area
  - 2x processor speed
  - 2x memory capacity
  - ½ energy / power consumption.

↳ technically not a law, but it's a self-fulfilling prophecy.

**Memory Wall:** every 2 years,
- 2x Instructions / second
- 2x memory capacity
- 1.1x memory latency



↳ growing disparity between processor and memory performance!
∴ significant effort in reducing / hiding memory latency.

**Latency:** time from start to finish (aka response time).

**Throughput:** number of tasks completed per time unit (aka bandwidth)
↳ throughput can exploit parallelism, but latency cannot!

- Improving the latency of a component always improves the overall system throughput.

$$\text{Speedup} = \frac{\text{Performance}(x)}{\text{Performance}(y)}.$$

"x is "speedup" times faster than y"

↳ can be broken down into $\frac{\text{throughput}(x)}{\text{throughput}(y)}$ or $\underbrace{\frac{\text{latency}(y)}{\text{latency}(x)}}$.

don't forget - latency will likely be < 1, so need to invert fraction!

## Iron Law of Performance:

$$\text{CPU Time: } \frac{\cancel{\text{instructions}}}{\text{program}} \cdot \underbrace{\frac{\cancel{\text{cycles}}}{\cancel{\text{instruction}}}}_{\text{aka CPI}} \cdot \frac{\text{seconds}}{\cancel{\text{cycle}}}$$

IPC is inverse - instructions per cycle