



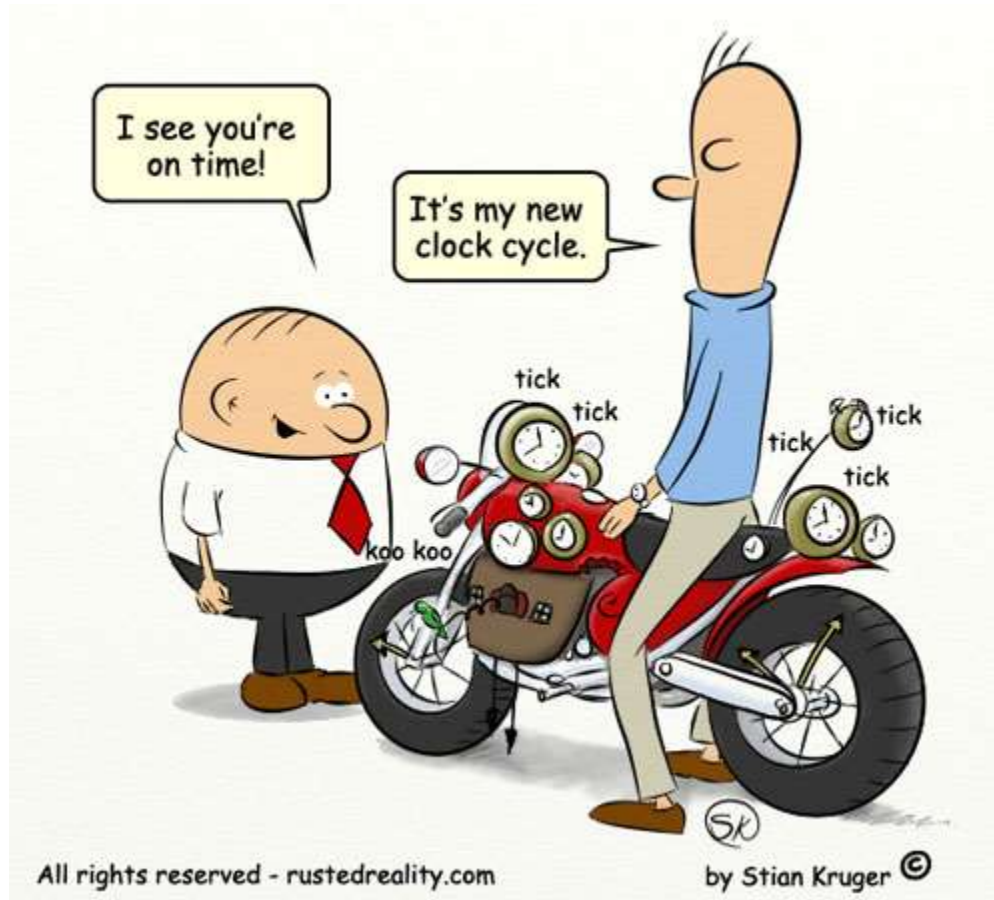
# CS230: Digital Logic Design and Computer Architecture

## Lecture 17: Empirical evaluation and Memory Hierarchy

<https://www.cse.iitb.ac.in/~biswa/courses/CS230/autumn23/main.html>

<https://www.cse.iitb.ac.in/~biswa/>

# Empirical Evaluation



Benchmarks

Metrics

Simulators

Latency and bandwidth

# Performance

- Latency (execution/response time): time to finish one task. It is additive ( $\text{Performance} = 1/\text{latency}$ )
- Throughput (bandwidth): number of tasks/unit time. It is not additive



- *Latency lags bandwidth, Bandwidth hurts latency,  
Read: <https://cacm.acm.org/magazines/2004/10/6401-latency-lags-bandwidth/fulltext>*

# Latency vs bandwidth

Latency vs Bandwidth, How they affect each other?

Latency helps bandwidth but not vice versa.

DRAM  
latency



More # Accesses  
~DRAM Bandwidth



Bandwidth usually hurts latency

Queues -  
Bandwidth



Increases latency



# For the curious ones

*Bandwidth problems can be cured with money.  
Latency problems are harder because the speed of light is  
fixed – you can't bribe God*

[https://www.youtube.com/watch?list=PL2LuePcZTMh\\_MzNHqZWNdvWdAnAThHCKK&v=lfqgpuH10uc&feature=emb\\_logo](https://www.youtube.com/watch?list=PL2LuePcZTMh_MzNHqZWNdvWdAnAThHCKK&v=lfqgpuH10uc&feature=emb_logo)

[https://www.youtube.com/watch?v=GNK-67JUH7M&list=PL2LuePcZTMh\\_MzNHqZWNdvWdAnAThHCKK&index=2](https://www.youtube.com/watch?v=GNK-67JUH7M&list=PL2LuePcZTMh_MzNHqZWNdvWdAnAThHCKK&index=2)

[https://www.youtube.com/watch?v=5CxpoGwCxKU&list=PL2LuePcZTMh\\_MzNHqZWNdvWdAnAThHCKK&index=3](https://www.youtube.com/watch?v=5CxpoGwCxKU&list=PL2LuePcZTMh_MzNHqZWNdvWdAnAThHCKK&index=3)

Some of the major bottlenecks: latency and bandwidth bottlenecks

# Amdahl's Law (common case fast)



$$\text{Speedup}_{\text{overall}} = \frac{\text{Execution Time}_{\text{old}}}{\text{Execution Time}_{\text{new}}}$$

1

$$= \frac{(1 - \text{Fraction}_{\text{enhanced}}) + \frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}}}{1}$$




# Amdahl's Law

Which one will provide better overall speedup?

- A. Small speedup on the large fraction of execution time.
- B. Large speedup on the small fraction of execution time.
- C. Does not matter.

Depends on the difference between small and large. Mostly it is A.

# Evaluation

- To compare Processor A with Processor B by running programs
- How many programs? 
- The programs that you care. 
- What if I want to build a new one (processor, caches, DRAM) ? 



# World of Benchmarks

- SPEC CPU 2017 (<https://www.spec.org/cpu2017/>)

The **SPEC CPU® 2017** benchmark package contains SPEC's **next-generation, industry-standardized, CPU intensive** suites for measuring and comparing **compute intensive performance, stressing a system's processor, memory subsystem and compiler.**

**SPECspeed:** used for comparing time for a computer to complete single tasks

**SPECrate:** measure the throughput or work per unit of time.

# World of Benchmarks

CloudSuite (<https://www.cloudsuite.ch/>)

CloudSuite is a benchmark suite for **cloud services**. The benchmarks are based on real-world software stacks and represent real-world setups.

PARSEC (<https://parsec.cs.princeton.edu/>)

Benchmark suite composed of **multithreaded** programs. The suite focuses on emerging workloads and was designed to be representative of next-generation shared-memory programs for chip-multiprocessors.

# World of Benchmarks

MobileBench

(<https://mobilebench.engineering.asu.edu/>)

comprising a selection of representative **smart phone applications**.

Many more application domain specific: Graph processing, ML perf,

# Pitfalls of Benchmarks

Benchmark not representative of all

Your workload is I/O bound → SPEC CPU is  
useless

Benchmark is too old

Need to be periodically refreshed

# Non-benchmarks

- Application kernels: A small code fragment or part of the program
- Synthetic benchmark : Not part of any real program!!
- Micro-benchmark
- *OK! So, I will create a chip and then evaluate these benchmarks*

# World of Simulators

- Functional Simulator: Used to **verify the correct** execution of the program. Can not be used for performance evaluation.
- Performance simulators:
  - (i) Trace-driven: ChampSim  
(<https://github.com/ChampSim/ChampSim>)
  - (ii) Execution-driven: gem5, Multi2sim

Functional simulator is part of the performance simulators.

# Evaluation Continued

Pick a *relevant* benchmark suite

Measure IPC of each program

Summarize the performance using:

Arithmetic Mean (AM)

Geometric Mean (GM)

Harmonic Mean (HM)

*Which one to  
choose?*

# Example

	IMTEL	ABM	AND
App. one	10	20	30
App. two	20	30	40
App. three	30	40	10

Which machine performs better over IMTEL and why?



Contd.

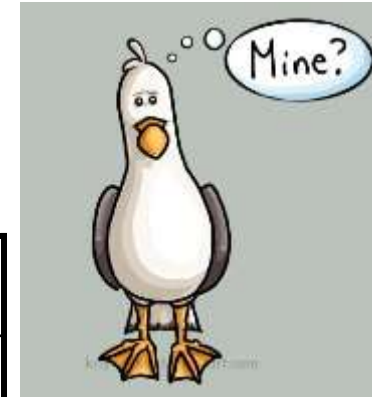
	ABM	AND
App. one	2	3
App. two	1.5	2
App. three	1.3	0.3
A.M.	1.60	1.76
G.M.	1.57	1.21
H.M.	1.54	0.72



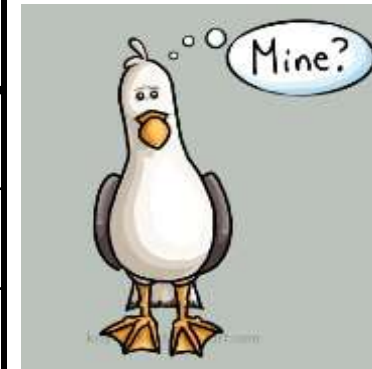
# AM on ratios (use always GM on ratios)

	X	Y
App. 1	1	100
App. 2	1000	10

Normalized to X	X	Y
App. 1	1	100
App. 2	1	0.01
AM	1	50.005
Normalized to Y	X	Y
App. 1	0.01	1
App. 2	100	1
AM	50.005	1



Y is 50 times faster than X



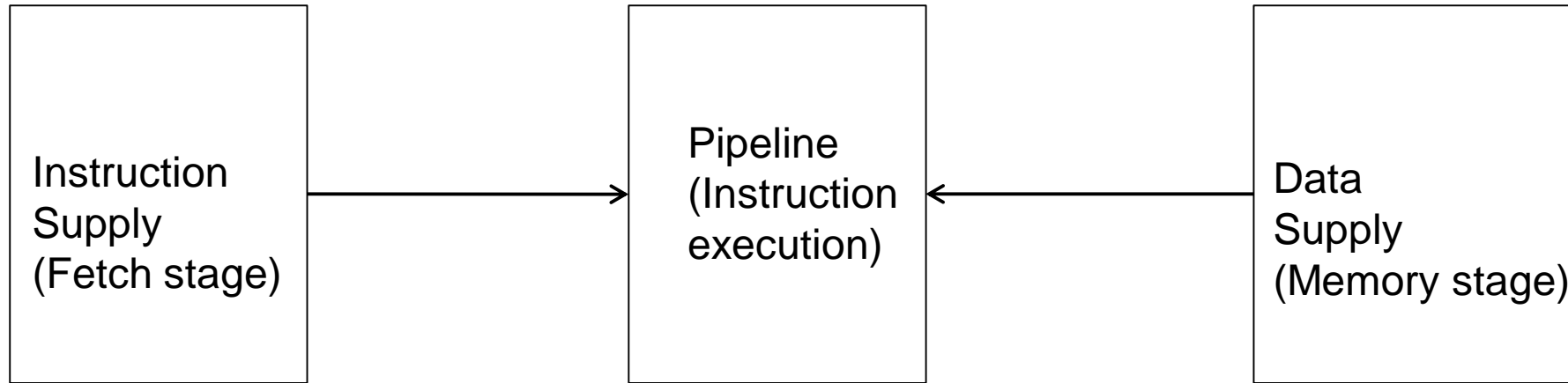
X is 50 times faster than Y



# Time for Memory Hierarchy

---

# The Ideal World

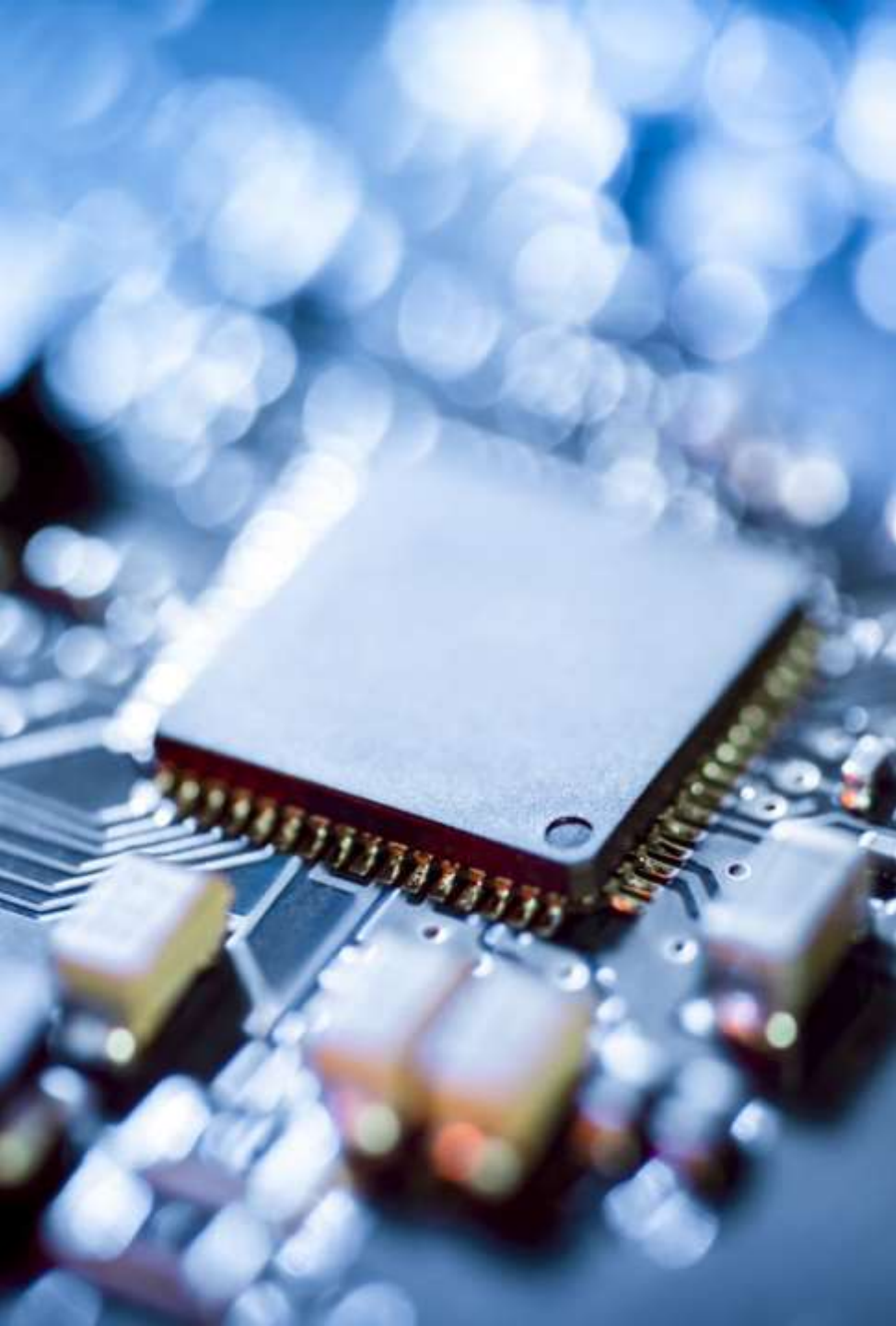


- Zero-cycle latency
- Infinite capacity
- Perfect control flow

- Zero-cycle latency
- Infinite capacity
- Infinite bandwidth

# World of Memory Hierarchy: Why?

## Before that What is memory?

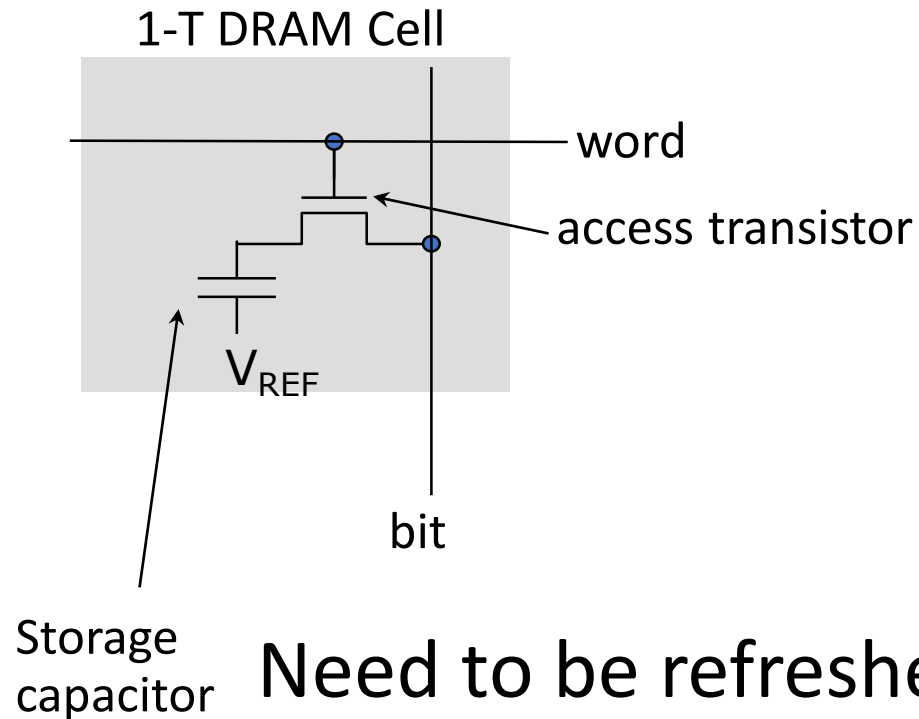


# Semiconductor Memory

---

- Semiconductor memory began to be competitive in early 1970s
  - Intel formed to exploit market for semiconductor memory
  - Early semiconductor memory was Static RAM (SRAM). SRAM cell internals similar to a latch (cross-coupled inverters).
- First commercial Dynamic RAM (DRAM) was Intel 1103
  - 1Kbit of storage on single chip
  - charge on a capacitor used to hold value

# One transistor DRAM



Denser

Value kept in one cell is as per the charge in the capacitor

Need to be refreshed periodically to maintain the charge

So dynamic RAM (DRAM)



# DRAM

- Dynamic random-access memory
- Capacitor charge state indicates stored value
  - Whether the capacitor is charged or discharged indicates storage of 1 or 0
  - 1 capacitor
  - 1 access transistor
- Capacitor leaks
  - DRAM cell loses charge over time
  - DRAM cell needs to be refreshed



SRAMs (6T to 8T)

Static RAMs. No need of refresh as no capacitor.

Faster access (no capacitor)

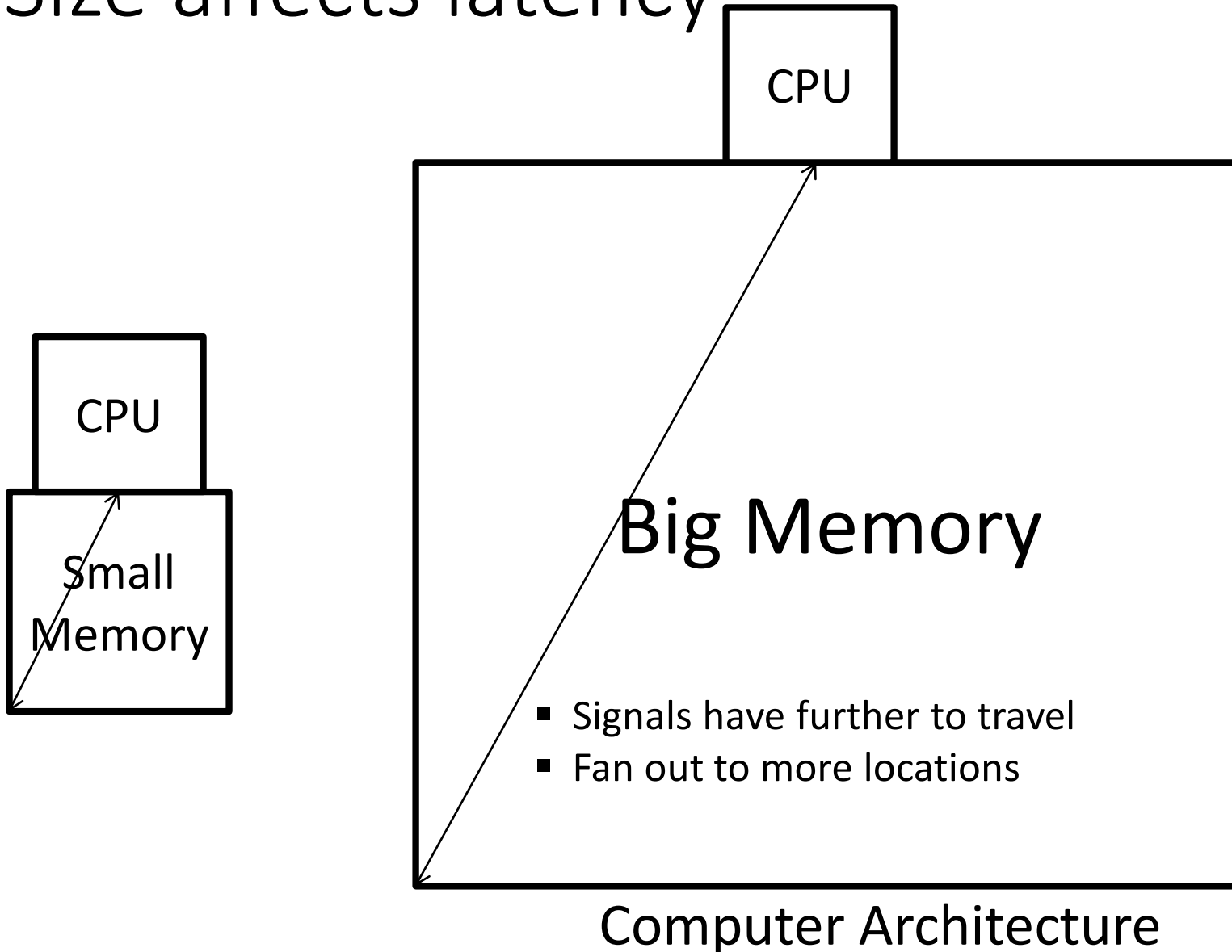
Density low as 6T: 1 bit compared to 1T: 1bit in DRAM

Costly than DRAM

# The Problem

- **Bigger is slower**
  - SRAM, 512 Bytes, sub-nanosec
  - SRAM, KByte~MByte, ~nanosec
  - DRAM, Gigabyte, ~50 nanosec
  - Hard Disk, Terabyte, ~10 millisec
- **Faster is more expensive (dollars and chip area)**
  - SRAM, < 1000\$ per GB
  - DRAM, < 20\$ per GB
  - Hard Disk < 0.01\$ per GB
  - These sample values scale with time
- **Other technologies have their place as well**
  - Flash memory, NVRAM, MRAM etc

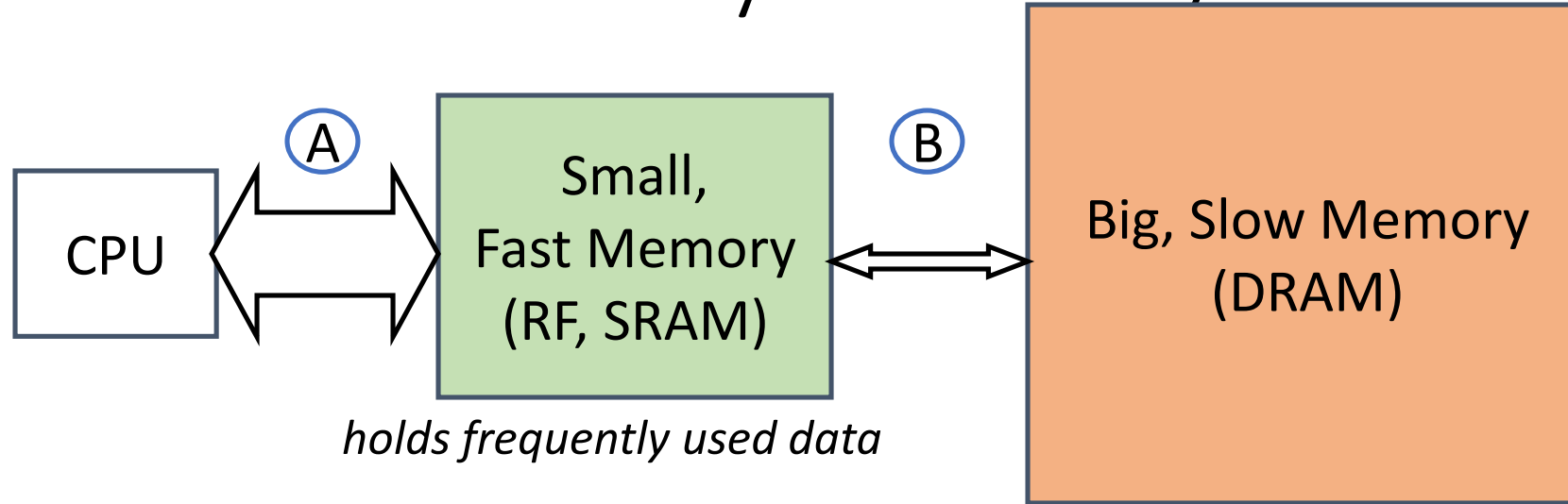
# Size affects latency



# Why Memory Hierarchy

- We want both fast and large
- But we cannot achieve both with a single level of memory
- Idea: Have multiple levels of storage (progressively bigger and slower as the levels are farther from the processor) and ensure most of the data the processor needs is kept in the fast(er) level(s)

# Welcome to Memory Hierarchy



- *capacity*: Register  $\ll$  SRAM (Cache)  $\ll$  DRAM
- *latency*: Register  $\ll$  SRAM (Cache)  $\ll$  DRAM
- *bandwidth*: on-chip  $\gg$  off-chip

On a data access:

*if data  $\in$  fast memory  $\Rightarrow$  low latency access Cache*

*if data  $\notin$  fast memory  $\Rightarrow$  high latency access DRAM*

# World with no caches

*North pole ☹️*

**Core**

32-bit Address

Data

200 to 300 cycles

Minimizing costly DRAM accesses  
is critical for performance

**Costly DRAM  
accesses ☹️**



4 GB DRAM

*South pole ☹️*