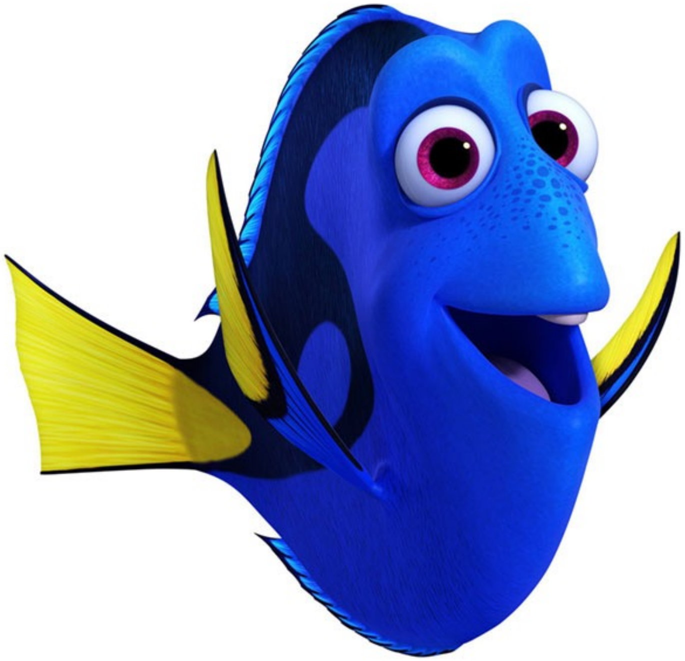
CS31: Introduction to Computer Systems

Week 9, Class 1
Storage and Memory
03/26/24

Dr. Sukrit Venkatagiri
Swarthmore College

Dive into Systems by Matthews, Newhall, and Webb

Stable Diffusion



Welcome!

Which, if any, of these storage devices is used by cloud service providers today (e.g., Google Cloud, Amazon Web Services, Dropbox)?

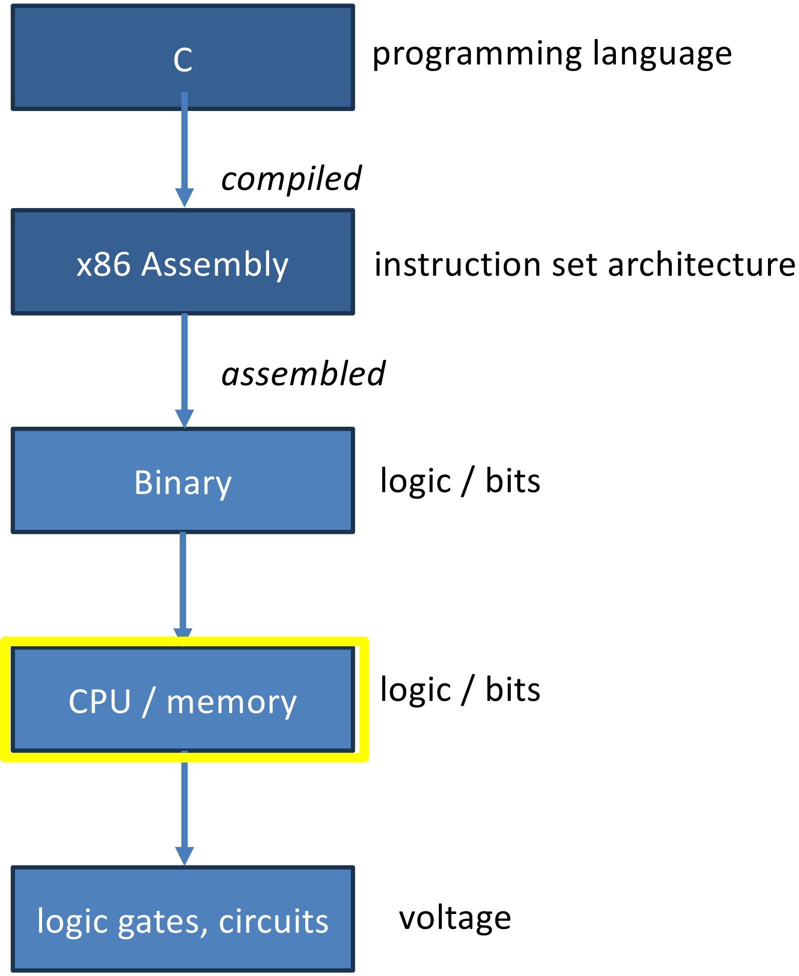
- floppy disks
- cassette tapes
- punch cards
- hard disk drives
- solid state drives

Announcements

- Updating class schedule
- Updated office hours:
  - Tuesday 2:30-4pm (same)
  - Thursday 2-3:30pm (updated, **no Wed office hours**)
- Slides available before each class for note-taking
- Space out HWs and videos (if any)
- More practice questions (ungraded)
- Second ninja session (next year)

Where are we?

Wk	Lecture	Lab
1	Intro to C	C Arrays, Sorting
2	Binary Representation, Arithmetic	Data Rep. & Conversion
3	Digital Circuits	Circuit Design
4	ISAs & Assembly Language	"
5	Pointers and Memory	Pointers and Assembly
6	Functions and the Stack	Maze Lab
7	Arrays, Structures & Pointers	"
Spring Break		
8	Storage and Memory Hierarchy	Game of Life
9	Caching	"
10	Operating System, Processing	Strings
11	Virtual Memory	Unix Shell
12	Parallel Applications, Threading	"
13	Threading	pthreads Game of Life
14	Threading	"



C — programming language

compiled

x86 Assembly — instruction set architecture

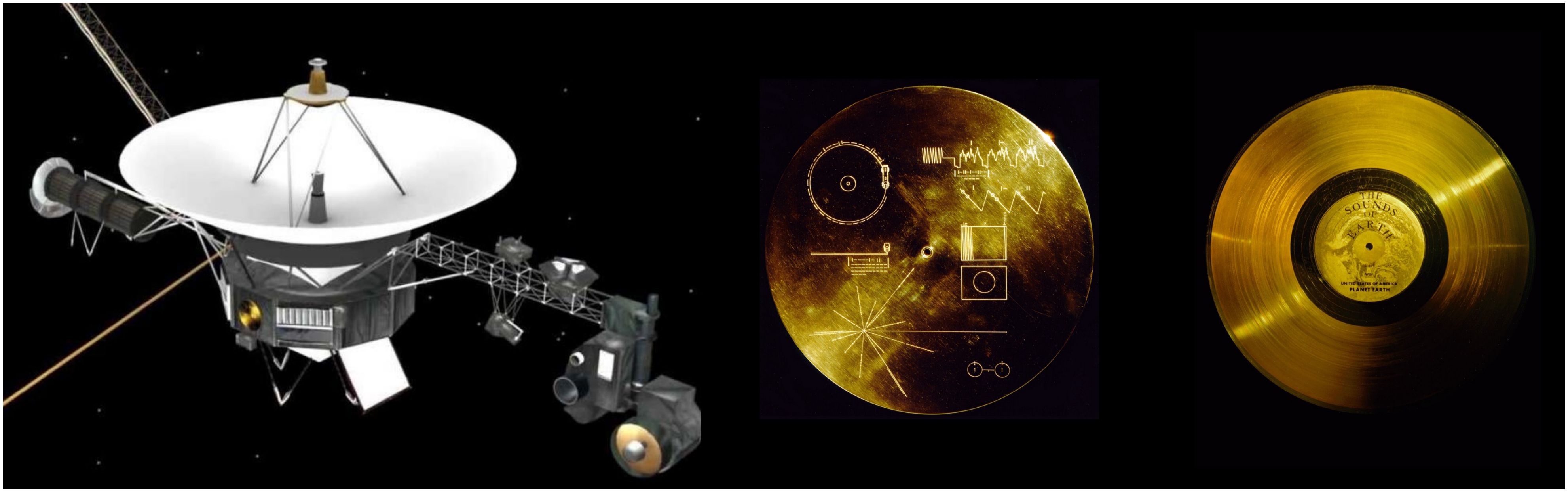
assembled

Binary — logic / bits

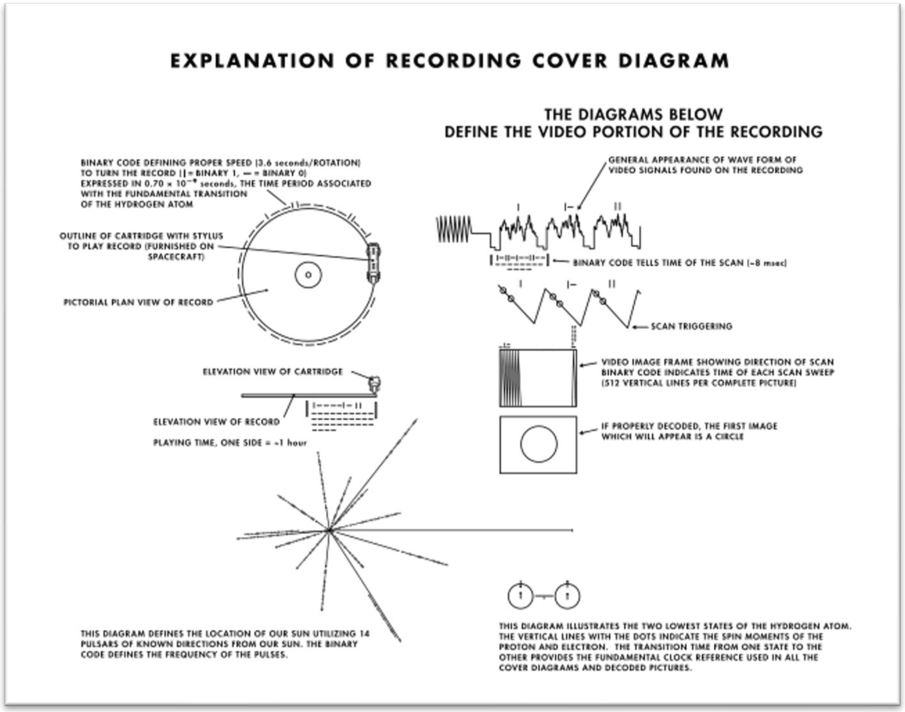
CPU / memory — logic / bits

logic gates, circuits — voltage

How do we communicate with aliens?



How do we communicate with aliens?



EXPLANATION OF RECORDING COVER DIAGRAM

BINARY CODE DEFINING PROPER SPEED (3.6 seconds/ROTATION) TO TURN THE RECORD (| = BINARY 1, — = BINARY 0) EXPRESSED IN 0.70×10^{-9} seconds, THE TIME PERIOD ASSOCIATED WITH THE FUNDAMENTAL TRANSITION OF THE HYDROGEN ATOM

OUTLINE OF CARTRIDGE WITH STYLUS TO PLAY RECORD (FURNISHED ON SPACECRAFT)

PICTORIAL PLAN VIEW OF RECORD

ELEVATION VIEW OF CARTRIDGE

ELEVATION VIEW OF RECORD

PLAYING TIME, ONE SIDE = ~1 hour

THIS DIAGRAM DEFINES THE LOCATION OF OUR SUN UTILIZING 14 PULSARS OF KNOWN DIRECTIONS FROM OUR SUN. THE BINARY CODE DEFINES THE FREQUENCY OF THE PULSES.

THE DIAGRAMS BELOW DEFINE THE VIDEO PORTION OF THE RECORDING

GENERAL APPEARANCE OF WAVE FORM OF VIDEO SIGNALS FOUND ON THE RECORDING

BINARY CODE TELLS TIME OF THE SCAN (~8 msec)

SCAN TRIGGERING

VIDEO IMAGE FRAME SHOWING DIRECTION OF SCAN BINARY CODE INDICATES TIME OF EACH SCAN SWEEP (512 VERTICAL LINES PER COMPLETE PICTURE)

IF PROPERLY DECODED, THE FIRST IMAGE WHICH WILL APPEAR IS A CIRCLE

THIS DIAGRAM ILLUSTRATES THE TWO LOWEST STATES OF THE HYDROGEN ATOM. THE VERTICAL LINES WITH THE DOTS INDICATE THE SPIN MOMENTS OF THE PROTON AND ELECTRON. THE TRANSITION TIME FROM ONE STATE TO THE OTHER PROVIDES THE FUNDAMENTAL CLOCK REFERENCE USED IN ALL THE COVER DIAGRAMS AND DECODED PICTURES.

Reading Quiz

The fact that not all memory technologies have similar size and performance characteristics leads to the idea of a ___.

A. Memory Disparity

B. Memory Hierarchy

C. Memory Imbalance

D. Memory Sadness

The idea that we typically access the same data over and over is known as…

A. caching

B. data repetition

C. iterative data usage

D. locality

Compared to the speed of main memory (DRAM), accessing a disk is…

A. blazing fast (~100,000x faster)

B. somewhat faster (100x faster)

C. about the same (1x)

D. somewhat slower (100x slower)

E. excruciatingly slow (~100,000x slower)

Transition

- First half of course: hardware focus
  - How the hardware is constructed
  - How the hardware works
  - How to interact with hardware / ISA

- Up next: performance and software systems
  - Memory performance
  - Operating systems
  - Standard libraries (strings, threads, etc.)

Today

- Types of memory
  - Primary and secondary
- Memory hierarchy
- Abstraction through cache
  - Prediction and locality
- Multi-core CPUs (brief)

Efficiency

- How to efficiently run programs

- Good algorithms are critical…

- **BUT**: many systems concerns to account for too!
  - The memory hierarchy and its effect on program performance
  - OS abstractions for running programs efficiently
  - Support for parallel programming

Efficiency

- How to efficiently run programs

- Good algorithms are critical…

- Many systems concerns to account for too!
  - The memory hierarchy and its effect on program performance
  - OS abstractions for running programs efficiently
  - Support for parallel programming

Today

- Types of memory
  - Primary and secondary
- Memory hierarchy
- Abstraction through cache
  - Prediction and locality
- Multi-core CPUs (brief)

Suppose you’re designing a new computer architecture. Which type of memory would you use? Why?

A. low-capacity (~1 MB), fast, expensive

B. medium-capacity (a few GB), medium-speed, moderate cost

C. high-capacity (100’s of GB), slow, cheap

D. something else (it must exist)

Classifying Memory

- Broadly, two types of memory:
  1. Primary storage: CPU instructions can access any location at any time (assuming OS permission)
  2. Secondary storage: CPU can’t access this directly

Random Access Memory (RAM)

- Any location can be accessed directly by CPU
  - Volatile Storage: lose power → lose contents
- Static RAM (SRAM)
  - Latch-Based Memory (e.g. RS latch), 1 bit per latch
  - Faster and more expensive than DRAM
    - "On chip": Registers, Caches
- Dynamic RAM (DRAM)
  - Capacitor-Based Memory, 1 bit per capacitor
    - "Main memory": Not part of CPU

Memory Technologies

- Static RAM (SRAM)
  - 0.5ns – 2.5ns, $2000 – $5000 per GB

- Dynamic RAM (DRAM)
  - 50ns – 100ns, $20 – $75 per GB
    (Main memory, "RAM")

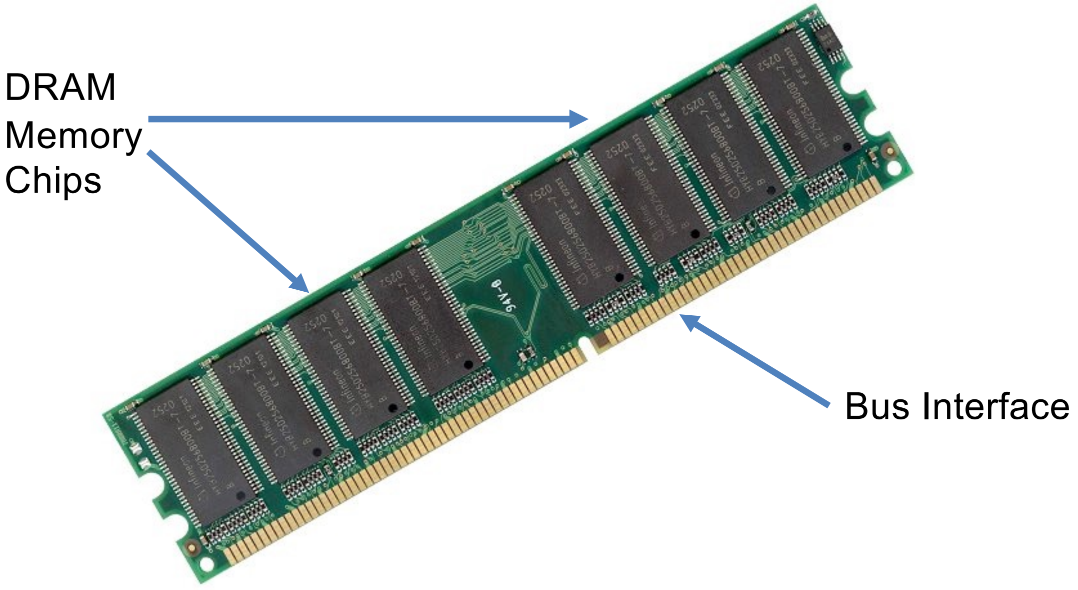
We've talked a lot about registers (SRAM) and we'll cover caches (SRAM) soon. Let's look at main memory (DRAM) now.

Dynamic Random Access Memory (DRAM)

Capacitor based:

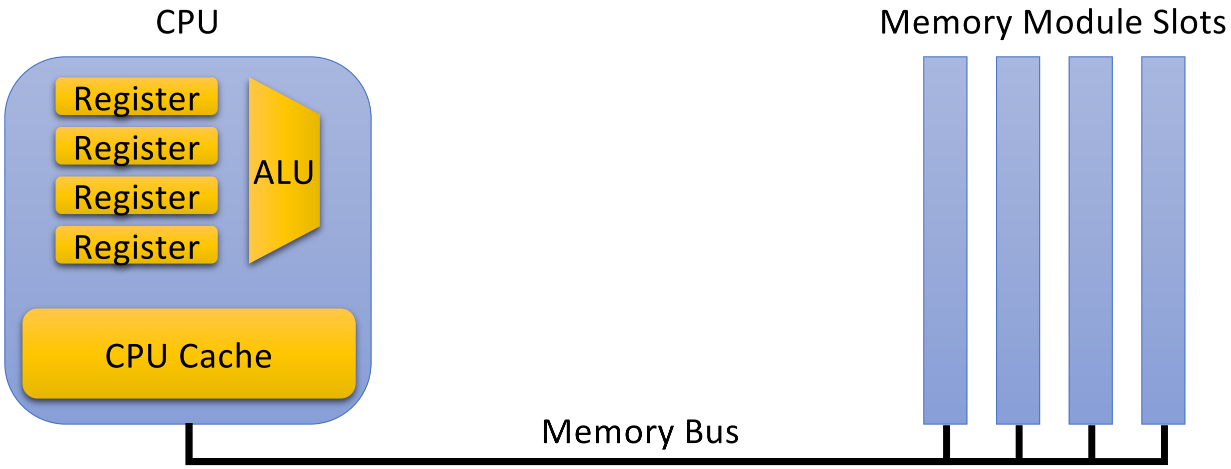
- cheaper and slower than SRAM
- capacitors are leaky (lose charge over time)
- Dynamic: value needs to be refreshed (every 10-100ms)

Example: DIMM (Dual In-line Memory Module):



Connecting CPU and Memory

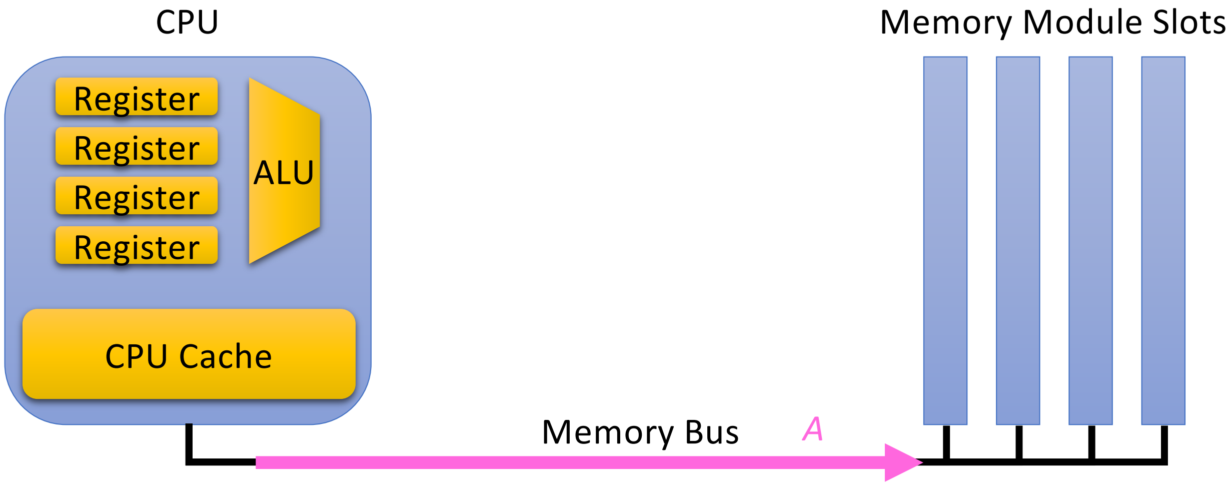
- Components are connected by a bus:
  - A bus is a collection of parallel wires that carry address, data, and control signals.
  - Buses are typically shared by multiple devices.



How A Memory Read Works

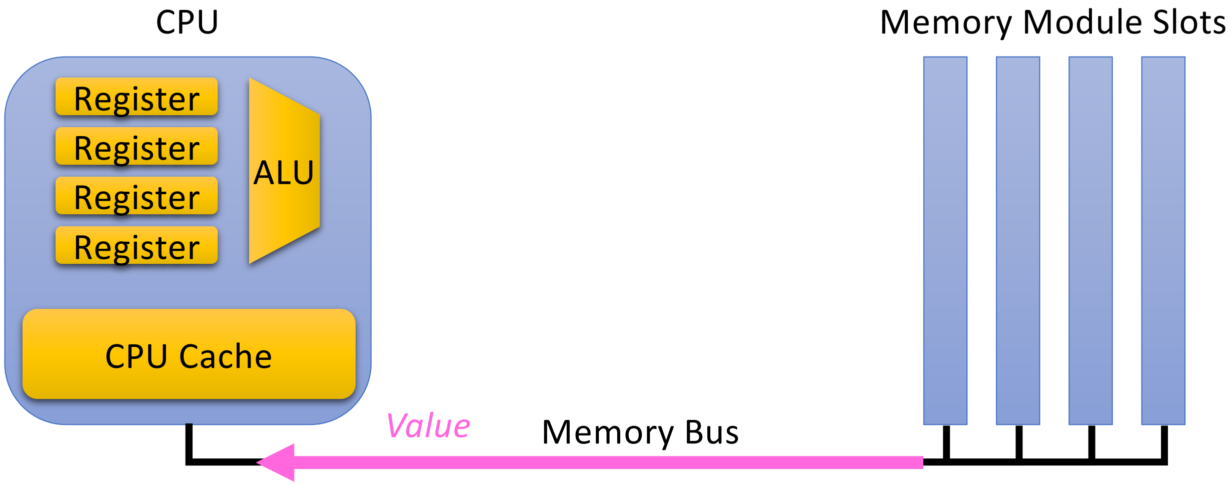
(1) CPU places address A on the memory bus.

Load operation: `mov (Address A), %rax`



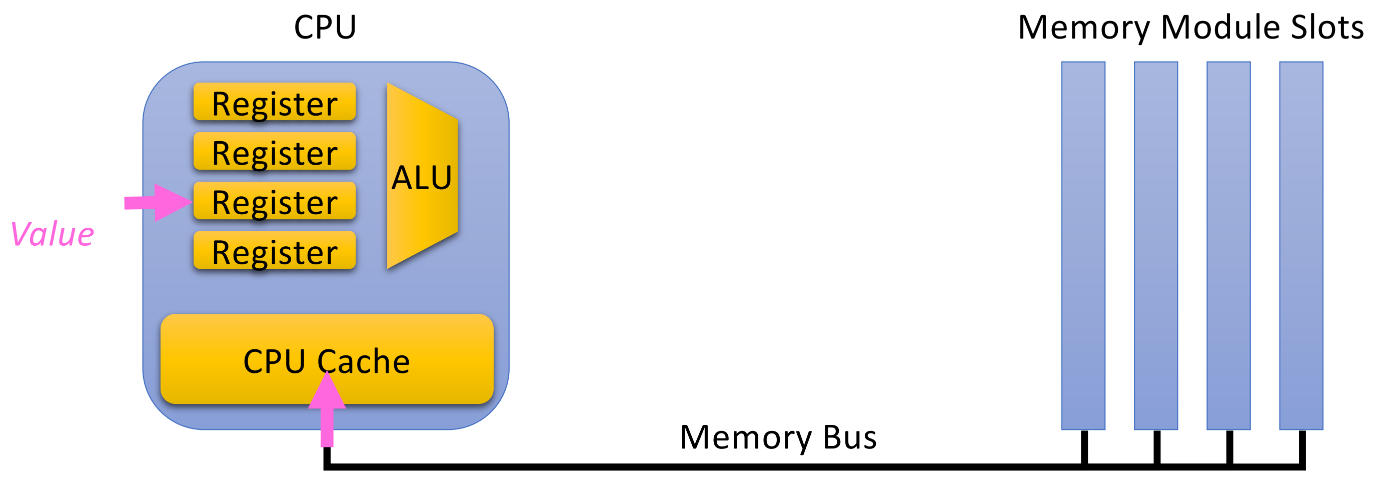
Read (cont.)

(2) Main Memory reads address A from memory, fetches value at that address and puts it on the bus



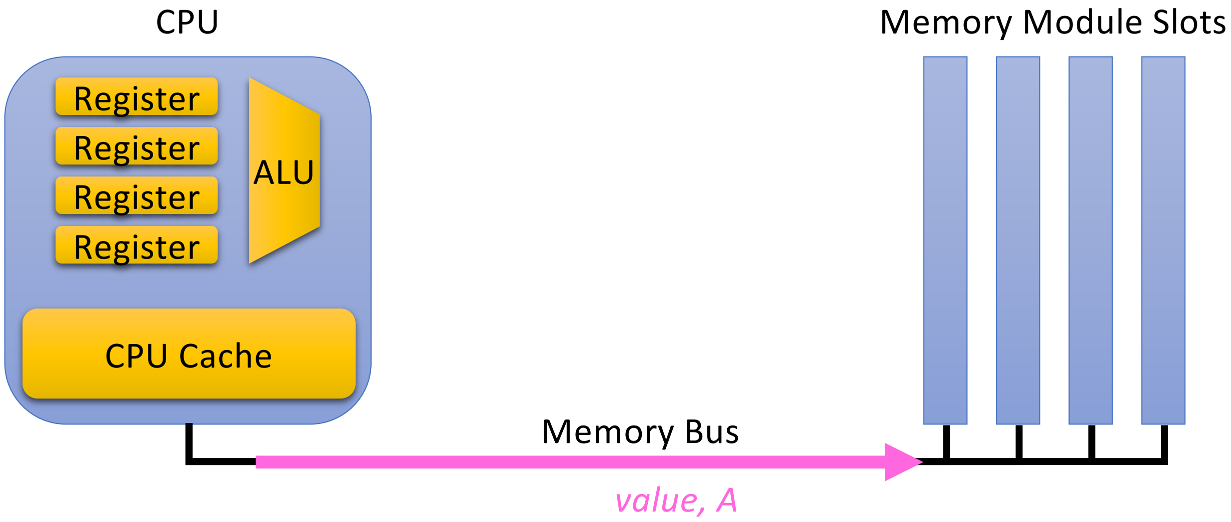
Read (cont.)

(3) CPU reads value from the bus, and copies it into register rax, a copy also goes into the on-chip cache memory



Write

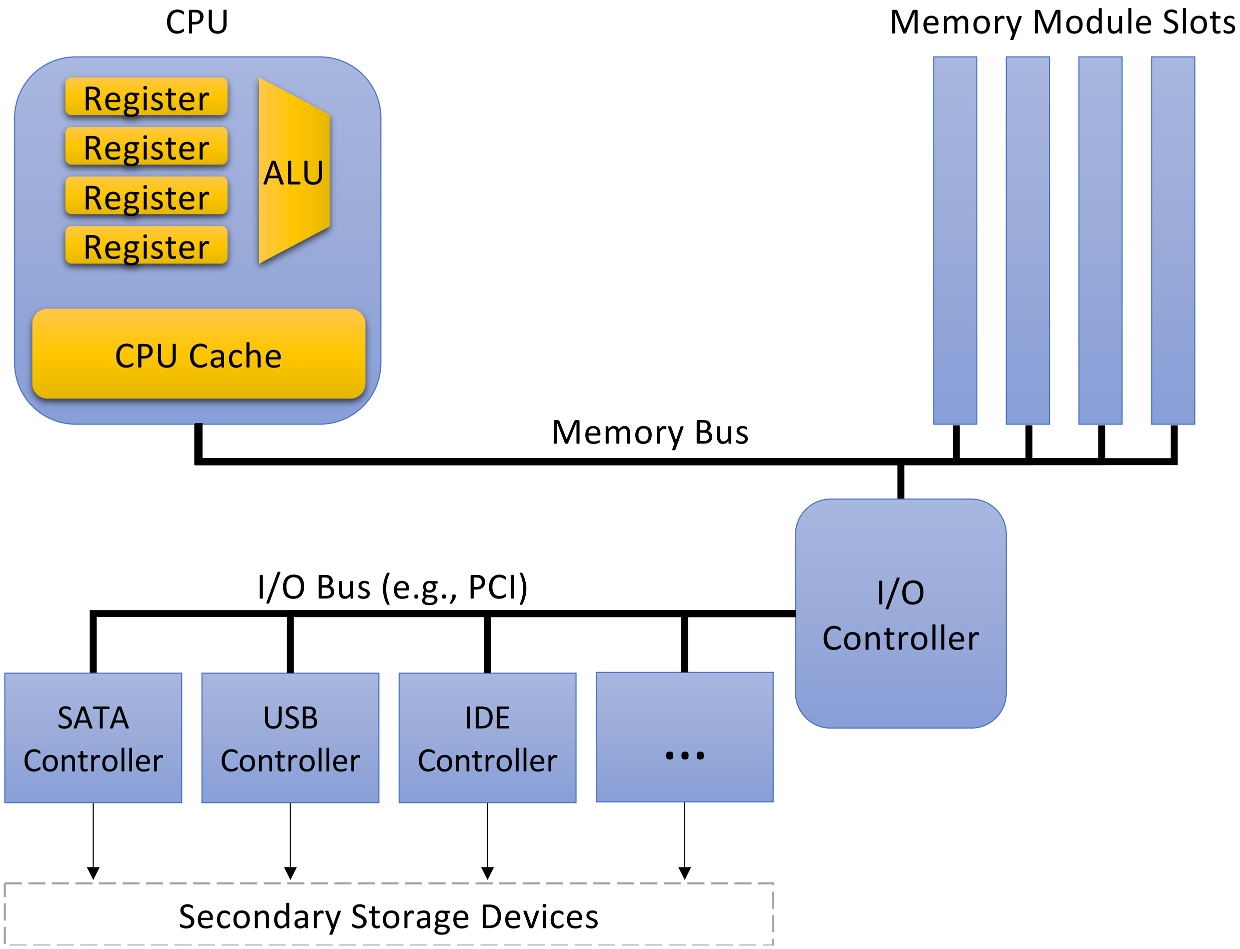
1. CPU writes A to bus, memory reads it
2. CPU writes value to bus, memory reads it
3. Memory stores value at address A



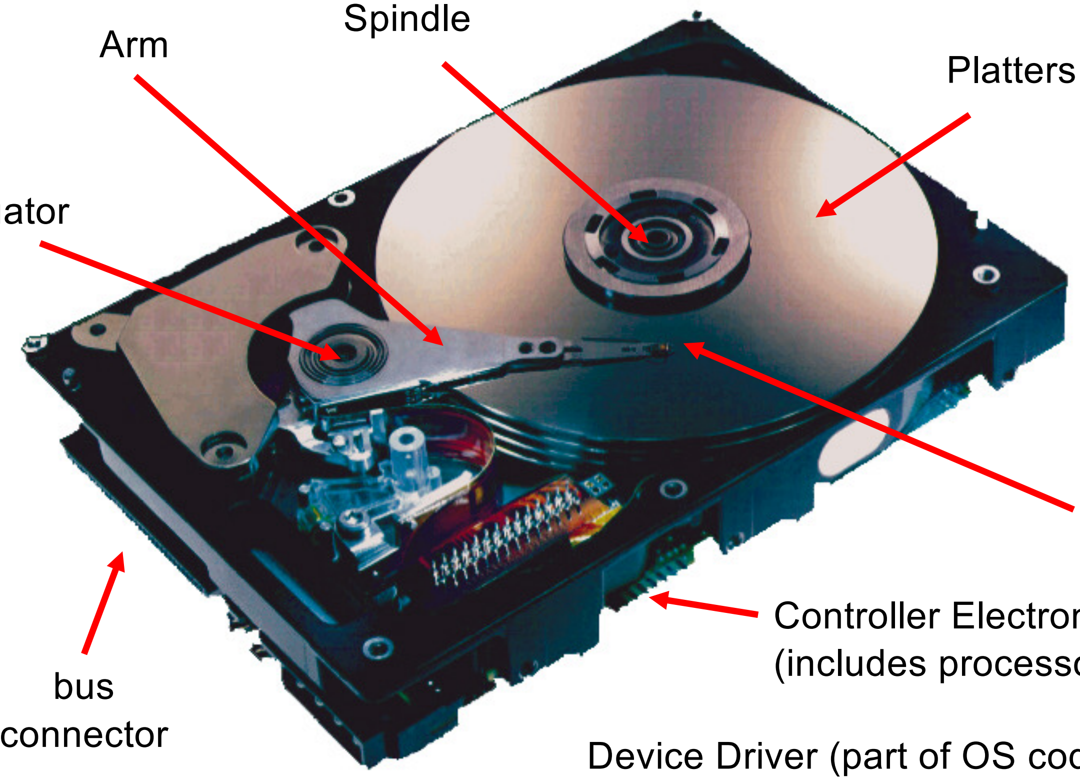
Secondary Storage

- Disk, Tape Drives, Flash Solid State Drives, …
- Non-volatile: retains data without a charge
- Instructions CANNOT directly access data on secondary storage
  - No way to specify a disk location in an instruction
  - Operating System moves data to/from memory

Secondary Storage



What’s Inside A Disk Drive?



Spindle

Arm

Platters

Data Encoded as points of magnetism on Platter surfaces

Actuator

R/W head

Controller Electronics
(includes processor & memory)

bus
connector

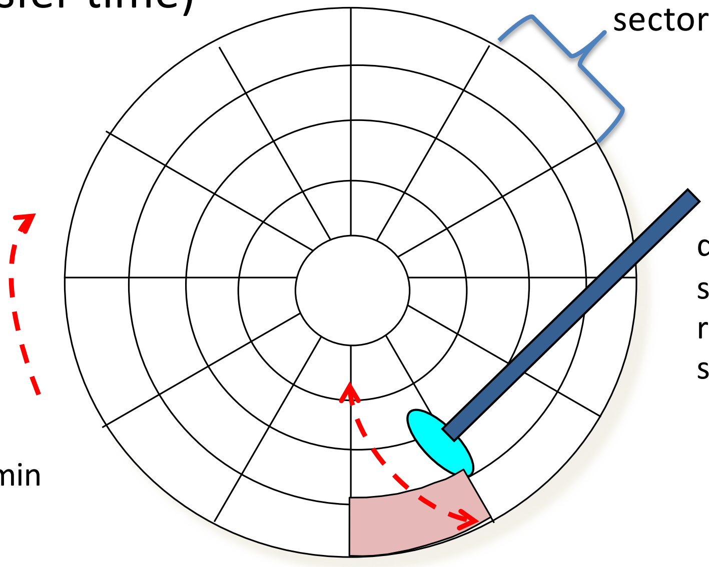
Device Driver (part of OS code)
interacts with Controller to R/W to disk

Image from Seagate Technology

Reading and Writing to Disk

Data blocks located in some sector of some srack on some surface

1. Disk Arm moves to correct track (seek time)
2. Wait for sector spins under R/W head (rotational latency)
3. As sector spins under head, data are Read or Written (transfer time)



sector

disk arm sweeps across surface to position read/write head over a specific track.

disk surface spins at a fixed rotational rate ~7200 rotations/min

Memory Technology

- Static RAM (SRAM)
  - 0.5ns – 2.5ns, $2000 – $5000 per GB

Like walking:

down the hall

- Dynamic RAM (DRAM)
  - 50ns – 100ns, $20 – $75 per GB

across campus

Solid-state disks (flash): 100 us – 1 ms, $2 - $10 per GB

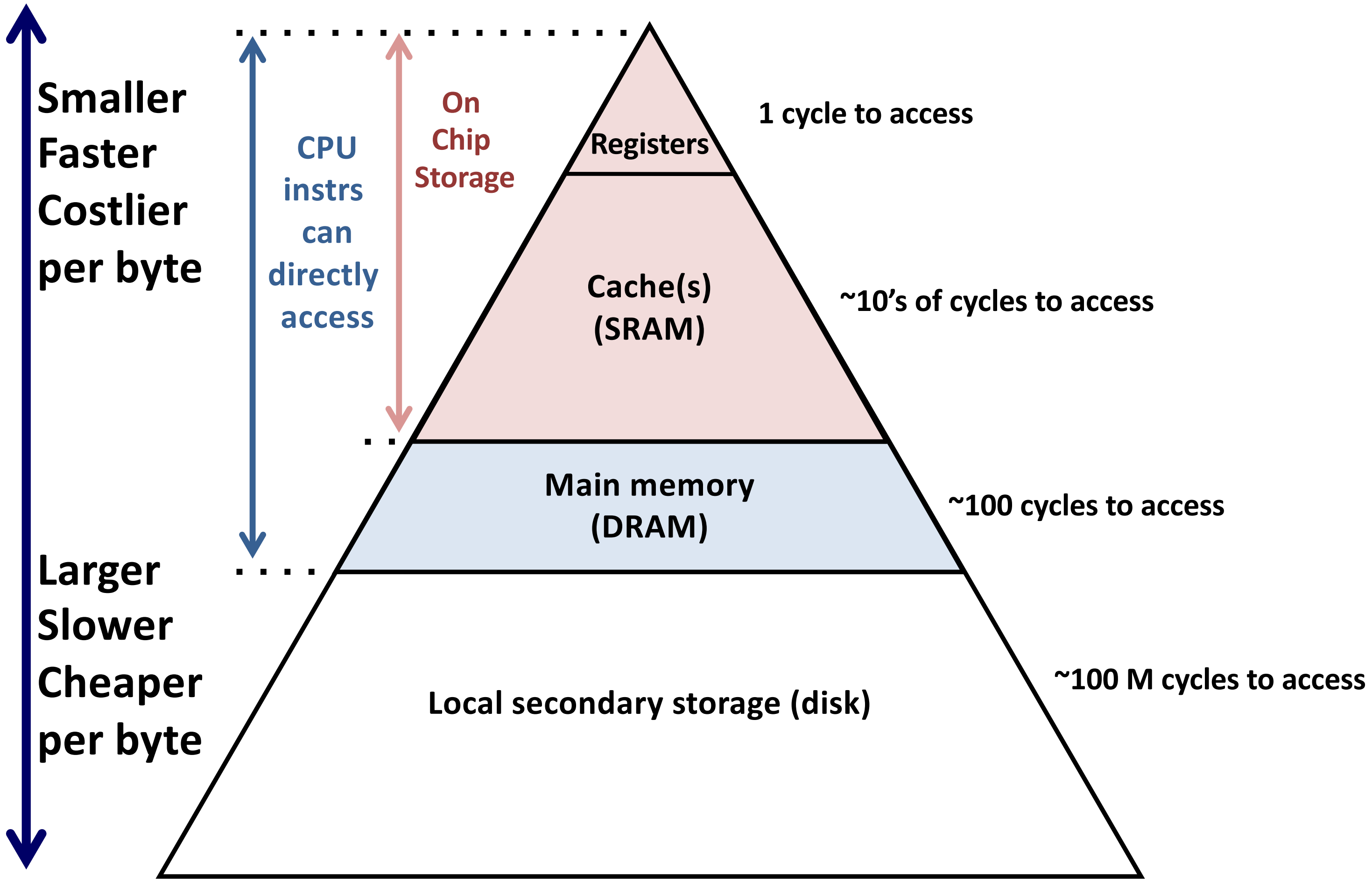
(to Cleveland / Indianapolis)

- Magnetic disk
  - 5ms – 15ms, $0.20 – $2 per GB

to Seattle

1 ms = 1,000,000 ns

The Memory Hierarchy



What is the best place to store 1GB of data? Why?

A. CPU registers

B. main memory (RAM)

C. secondary storage (e.g., disk drive)

D. cloud storage (e.g., Google Drive)

What is the best place to store 1GB of data… That you want to back up? That you want to access really fast? That you want to store affordably?

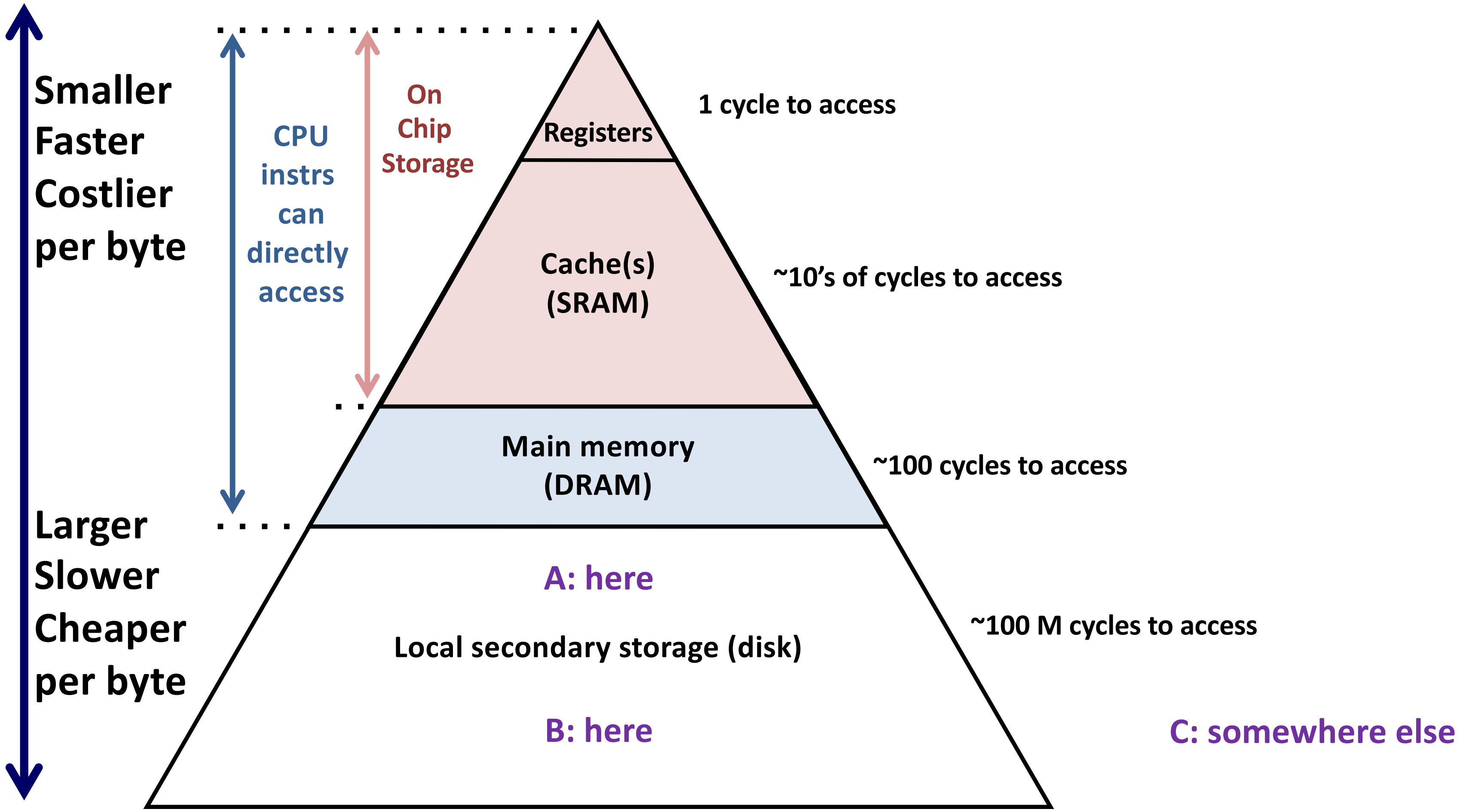
A. CPU registers

B. main memory (RAM)

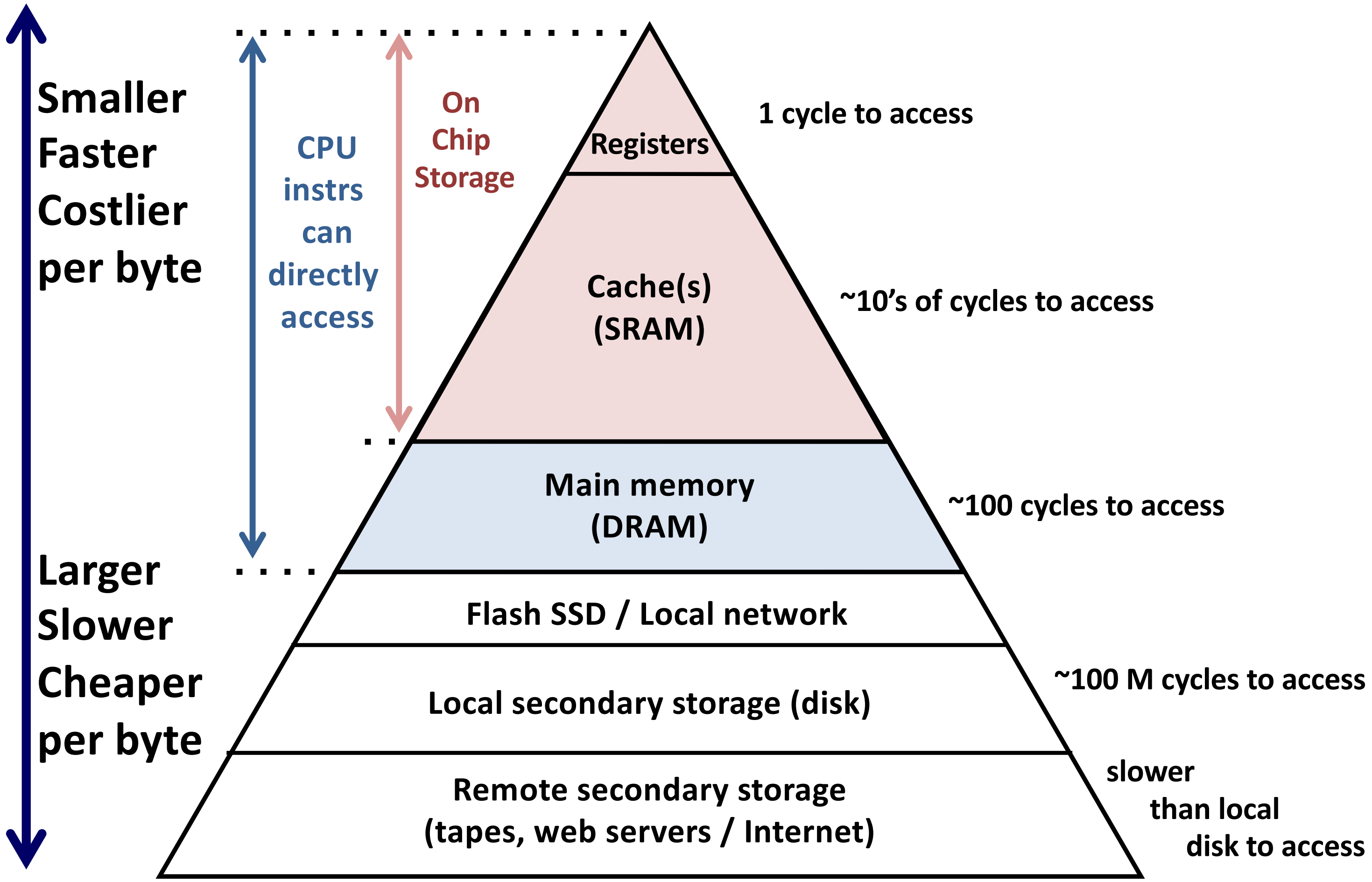
C. secondary storage (e.g., disk drive)

D. cloud storage (e.g., Google Drive)

Where does accessing the network belong?



The Memory Hierarchy



Abstraction Goal

- Reality: There is no one type of memory to rule them all!

- Abstraction: hide the complex/undesirable details of reality.

- Illusion: We have the speed of SRAM, with the capacity of disk, at reasonable cost.

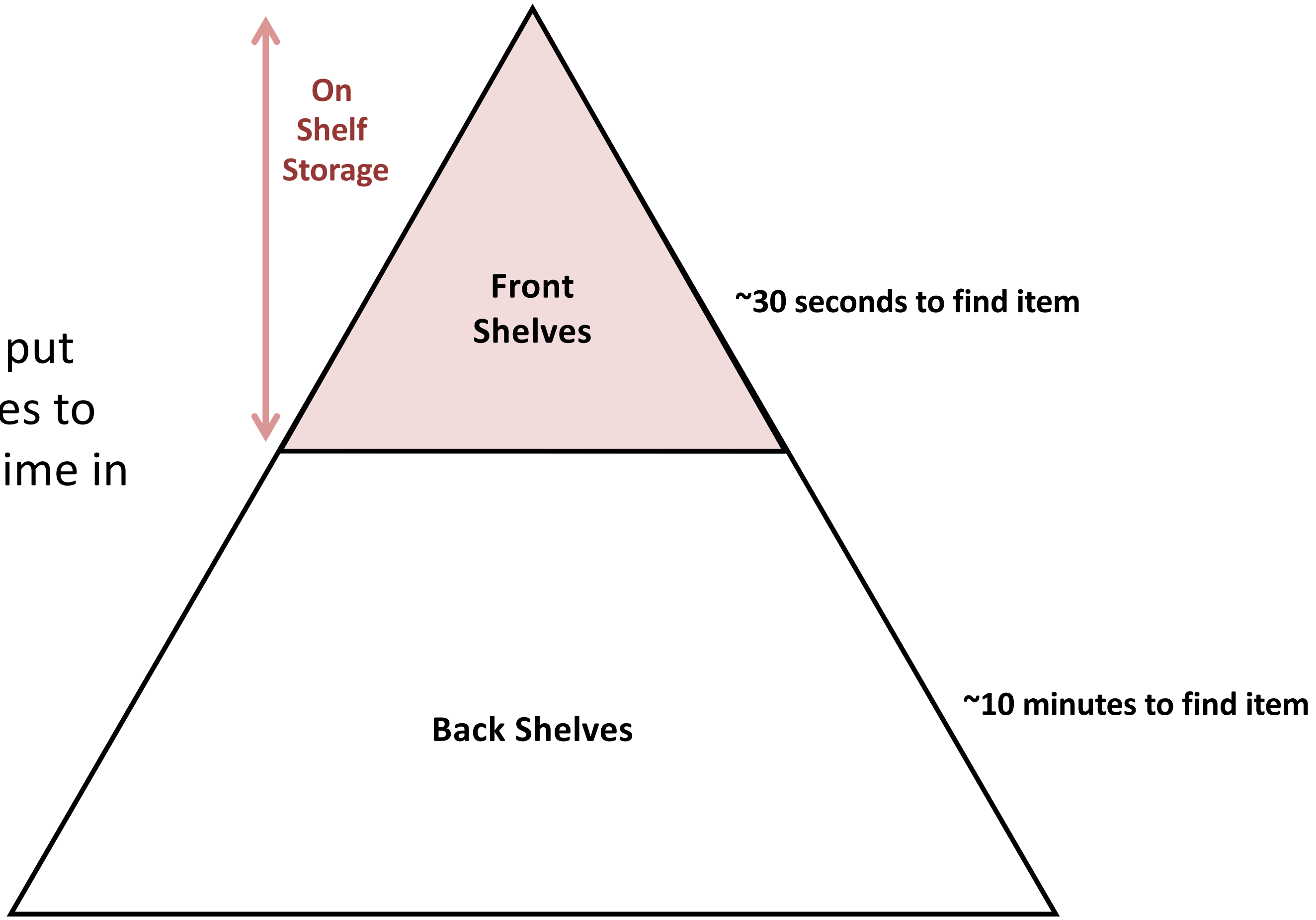
Motivating Story / Analogy

- You work at a grocery store in *The Good Place*
- Your store has dozens of aisles
  - 10-15 minutes to find something
  - Customers don't like searching…
- You have a set of shelves in the from of the store
  - < 30 seconds to find movie on front shelf

CHARACTER
DEATHS
R.I.P.
The
Good
Place

The Grocery Store Hierarchy

Goal: strategically put groceries on shelves to reduce customer time in store.



Quick vote: Which **single** item should we place on the front shelf?

A. Eggs

B. Milk

C. Oreos

D. Pumpkins

E. There’s no way for us to know

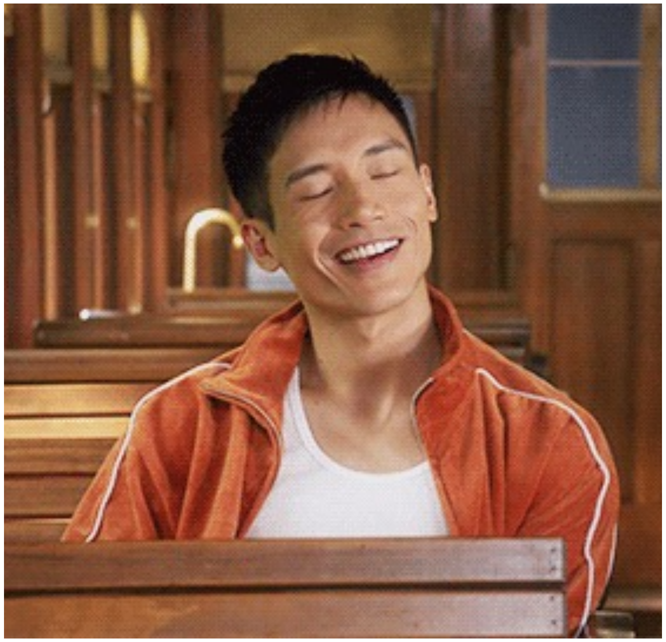
Problem: Prediction

- We can't know the future…
- So… are we out of luck?
  What might we look at to help us decide?
- The past is often a pretty good predictor…

Repeat Customer: Jason

- Has bought Oreos ten times in the last two weeks for his girlfriend

- You talk to him:
  - He just broke up with his girlfriend
  - Swears it will be the last time he buys Oreos (he’s said this the last six times)



Quick vote: Which item should we place on the shelf for tonight?

A. Eggs

B. Milk

C. Oreos

D. Pumpkins

E. There’s no way for us to know

Repeat Customer: Eleanor

- Eleanor buys pumpkin and eggs every time she goes grocery shopping

- You talk to her:
  - She loves baking pumpkin pie
  - She throws eggs out of her 30 story apartment window

Quick vote: Which **two** item should we place on the shelf for tonight?

A. Eggs and Oreos

B. Milk and Pumpkins

C. Milk and Oreos

D. Pumpkins and Eggs

E. There’s no way for us to know

Critical Concept: Locality

- Locality: we tend to repeatedly access recently accessed items, or those that are nearby.

- Temporal locality: An item that has been accessed recently is likely to be accessed again soon (Jason)

- Spatial locality: We're likely to access an item that's nearby (or related to) others we just accessed (Eleanor)

In the following code, how many examples are there of temporal / spatial locality?
Where are they?

```
int i;
int num = read_int_from_user();
int *array = create_random_array(num);
for (i = 0; i < num; i++) {
    printf(“At index %d, value: %d”, i, array[i]);
}
```

A. 1 temporal, 1 spatial
B. 1 temporal, 2 spatial
C. 2 temporal, 1 spatial
D. 2 temporal, 2 spatial
E. Some other number

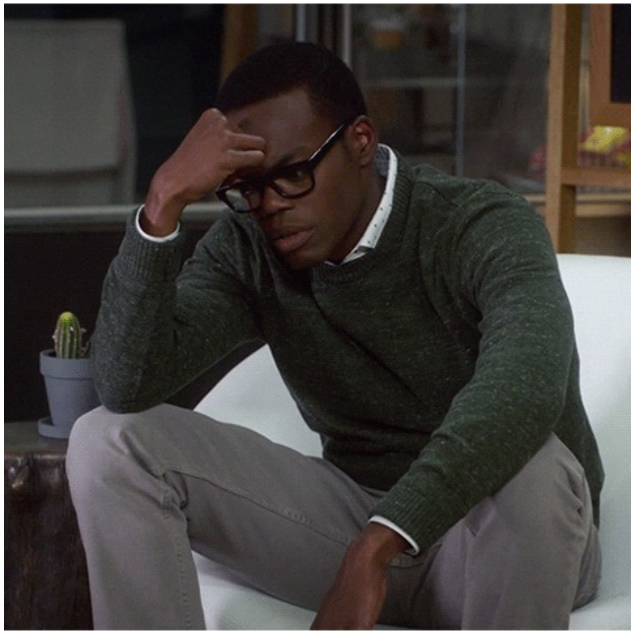
Customer: Tahani

- Tahani… doesn’t buy groceries. Pfft.

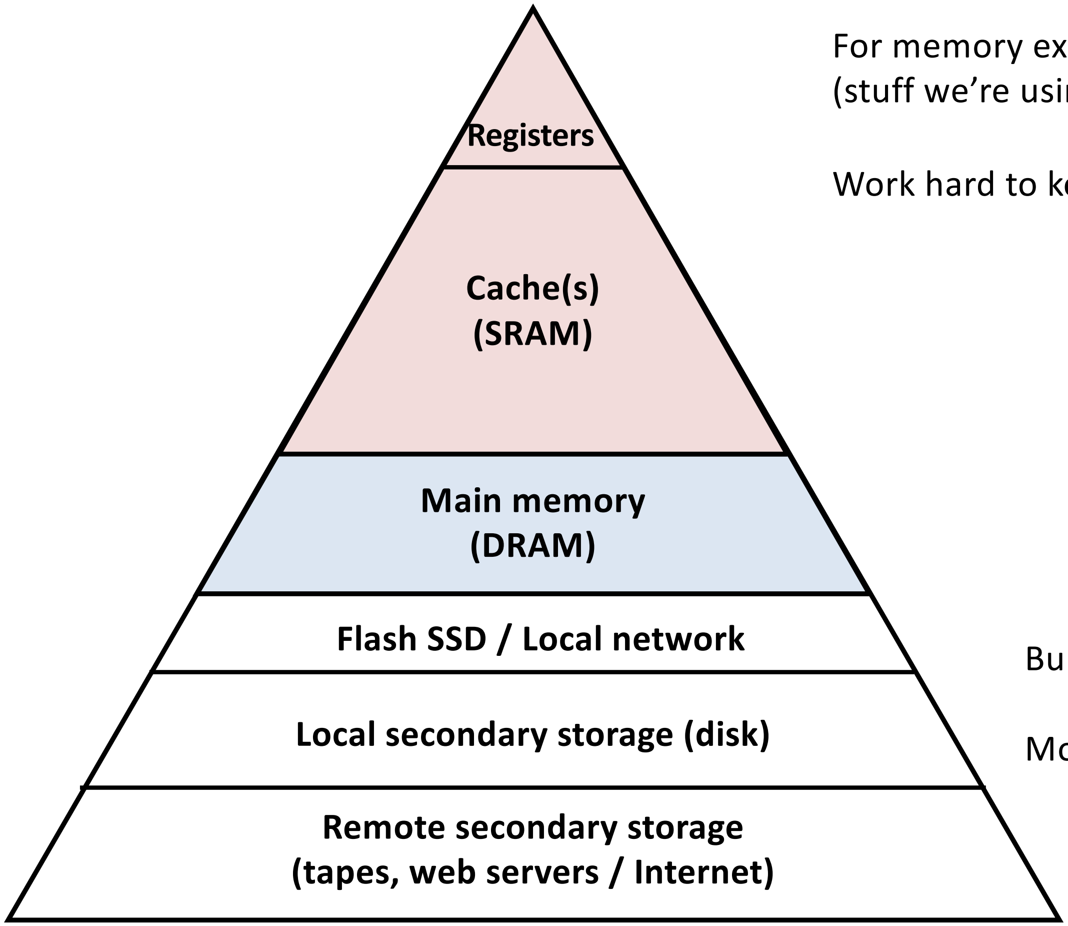
Customer: Chidi

- Chidi can never decide what to make for dinner, so he, too, doesn’t end up buying anything

- Difficult to predict how to stock the shelves for a customer like him



Big Picture


Registers
Cache(s)
(SRAM)
Main memory
(DRAM)
Flash SSD / Local network
Local secondary storage (disk)
Remote secondary storage
(tapes, web servers / Internet)

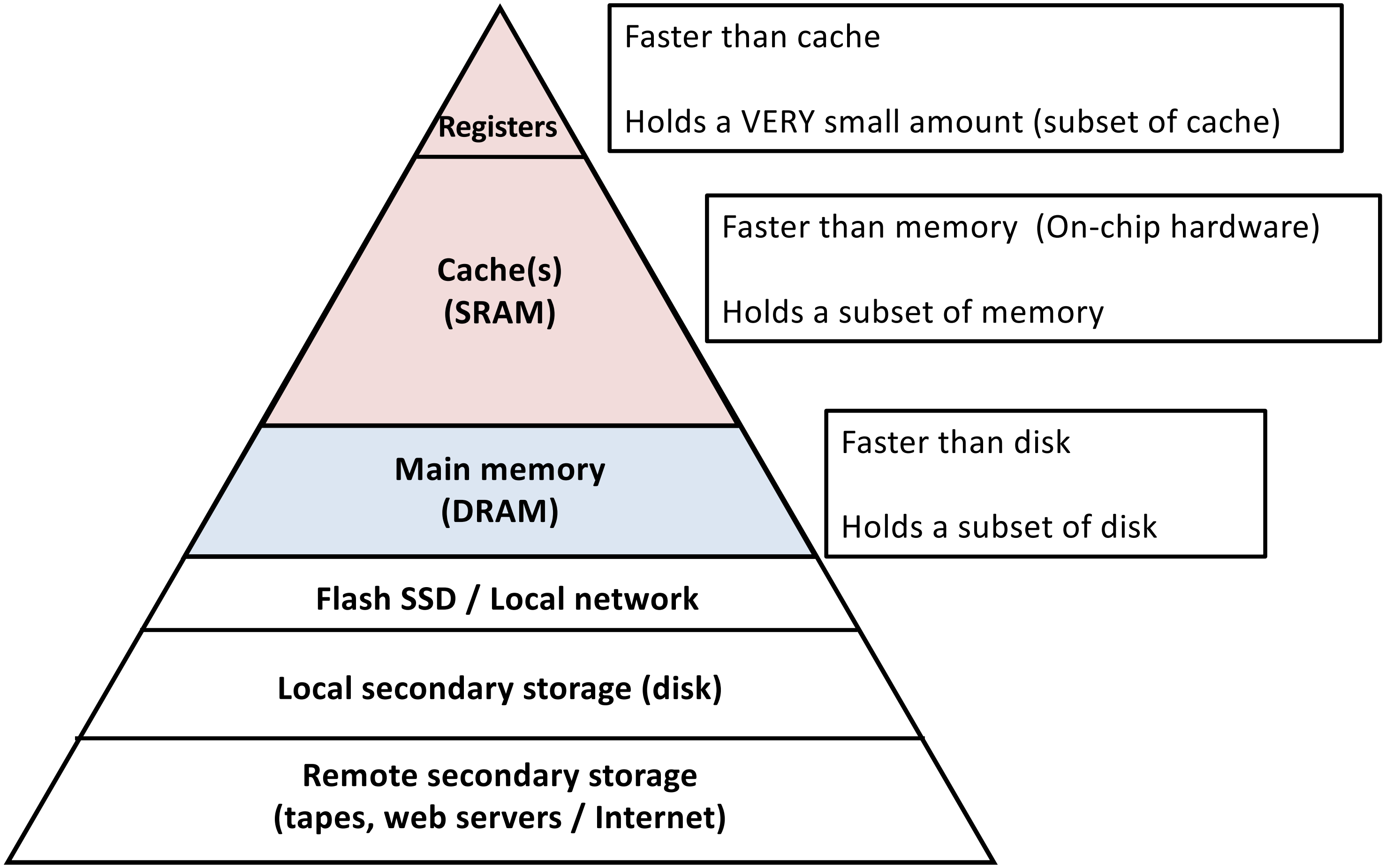
For memory exhibiting **locality**
(stuff we're using / likely to use):

Work hard to keep them up here!

Bulk storage down here

Move this up on demand

Big Picture



Registers

Cache(s)
(SRAM)

Main memory
(DRAM)

Flash SSD / Local network

Local secondary storage (disk)

Remote secondary storage
(tapes, web servers / Internet)

Faster than cache

Holds a VERY small amount (subset of cache)

Faster than memory (On-chip hardware)

Holds a subset of memory

Faster than disk

Holds a subset of disk

Cache

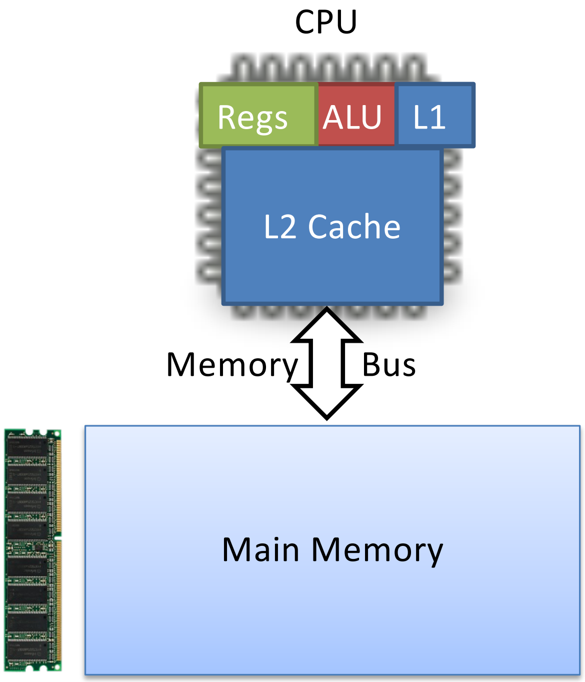
- Cache: in general, a storage location that holds a subset of a larger memory but is faster to access

When we say "cache", assume we're referring to CPU cache from now on, unless we say otherwise.

- CPU cache: an SRAM on-chip storage location that holds a subset of DRAM main memory (10-50x faster to access)

- Goal: choose the right subset, based on past locality, to achieve our abstraction

Cache Basics

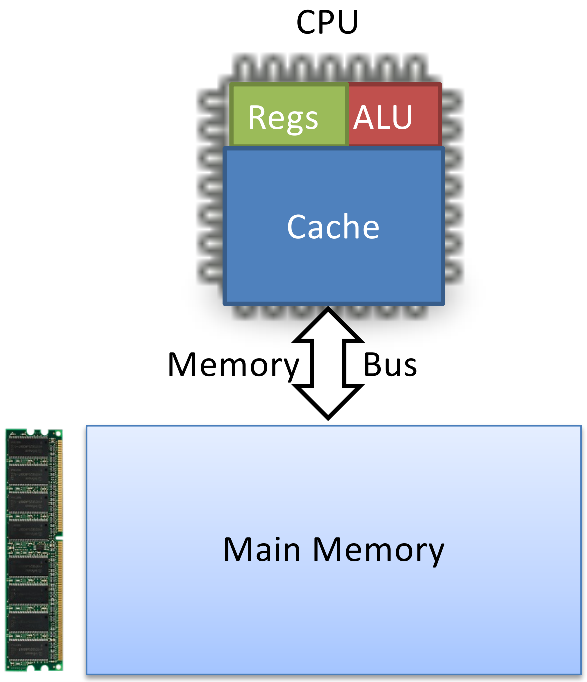


- CPU real estate dedicated to cache

- Usually two (or more) levels:
  - **L1**: smallest, fastest
  - **L2**: larger, slower

- Same rules apply:
  - **L1** subset of **L2**
  - Goes up to **L4** today

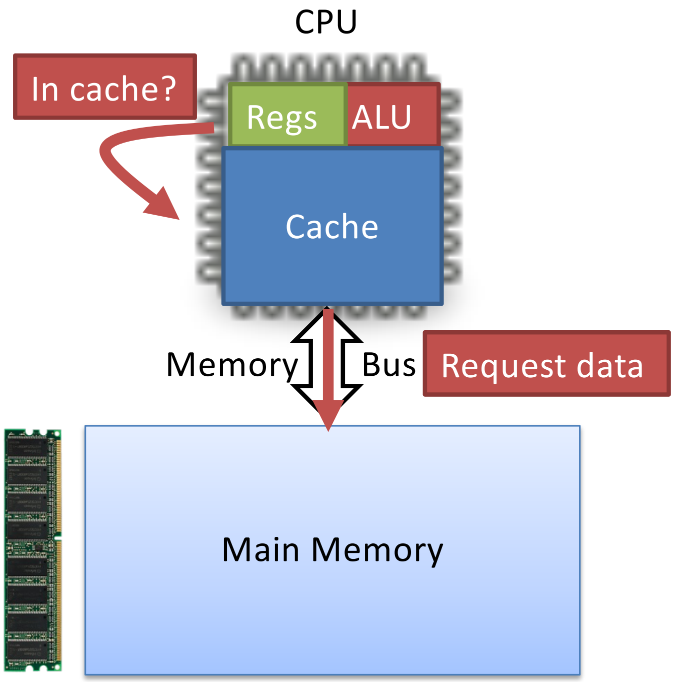
Cache Basics



Cache is a subset of main memory.
(Not to scale, memory much bigger!)

- CPU real estate dedicated to cache
- Usually two levels:
  - **L1**: smallest, fastest
  - **L2**: larger, slower
- We'll assume one cache (same principles)

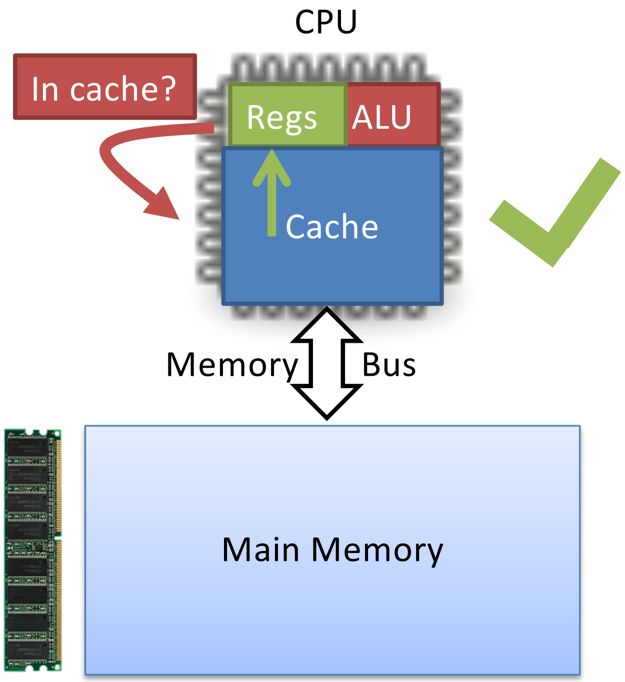
Cache Basics: Read from memory



- In parallel:
  - Issue read to memory
  - Check cache

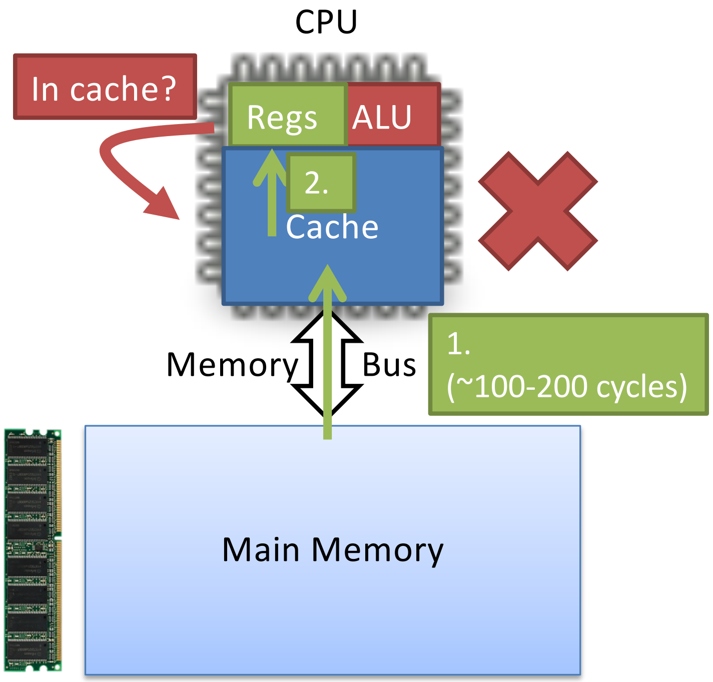
MB
BATTLESHIP
Combine Strategy And Luck
In This Exciting Naval Action Game
Ages 8 to Adult
2 Players
SET UP YOUR FLEET
CALL OUT THE SHOTS
MARK THE HITS
SINK YOUR OPPONENT'S FLEET

Cache Basics: Read from memory



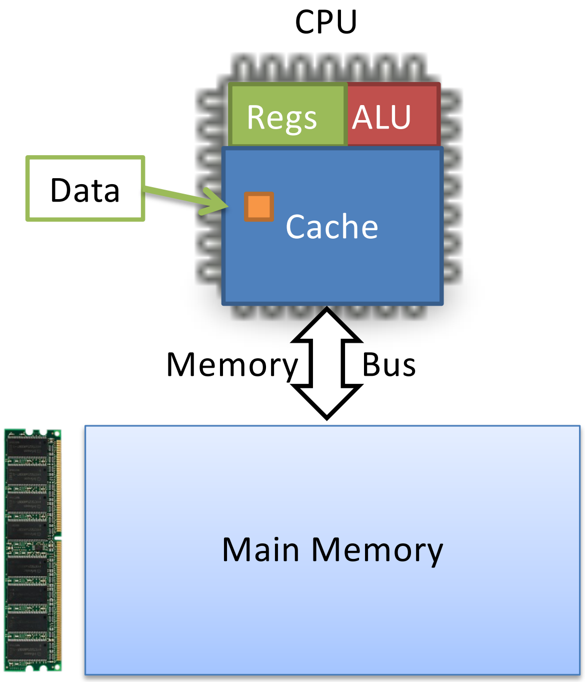
- In parallel:
  - Issue read to memory
  - Check cache
- Data in cache (hit):
  - Good, send to register
  - Cancel/ignore memory

Cache Basics: Read from memory



- In parallel:
  - Issue read to memory
  - Check cache
- Data in cache (hit):
  - Good, send to register
  - Cancel/ignore memory
- Data not in cache (miss):
  1. Load cache from memory (might need to evict data)
  2. Send to register

Cache Basics: Write to memory



- Assume data already cached
  - Otherwise, bring it in like read

1. Update cached copy

2. Update memory?

When should we copy the written data from cache to memory? Why?

A. Immediately update the data in memory when we update the cache.

B. Update the data in memory when we remove ("evict") the data from the cache.

C. Update the data in memory if the data is needed elsewhere (e.g., another core).

D. Update the data in memory at some other time. (When?)

Cache Basics: Write to memory

- Both options (write-through, write-back) viable

- write-though: write to memory immediately
  - simpler, accesses memory more often (slower)

- write-back: only write to memory on eviction
  - complex (cache inconsistent with memory)
  - potentially reduces memory accesses (faster)

Cache Basics: Write to memory

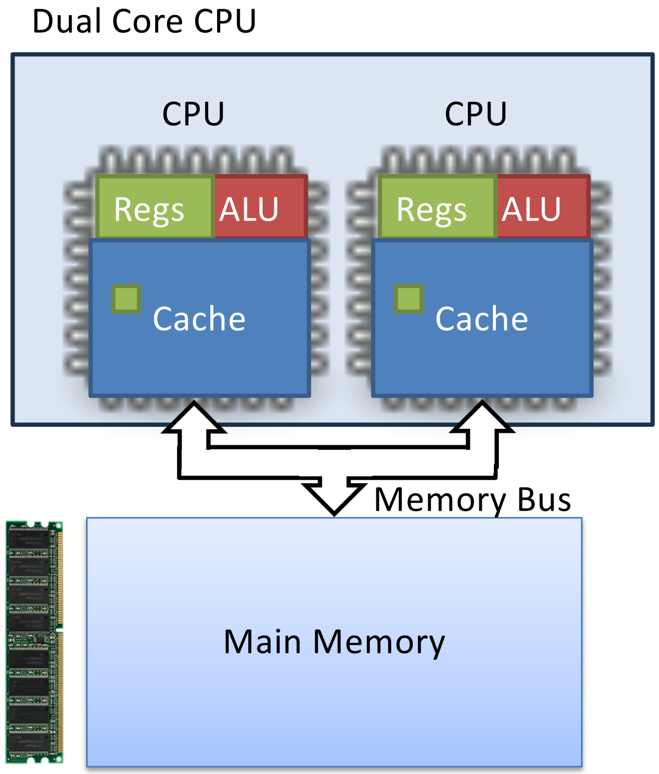
- Both options (write-through, write-back) viable

- write-though: write to memory immediately
  - simpler, accesses memory more often (slower)

- write-back: only write to memory on eviction
  - complex (cache inconsistent with memory)
  - potentially reduces memory accesses (faster)

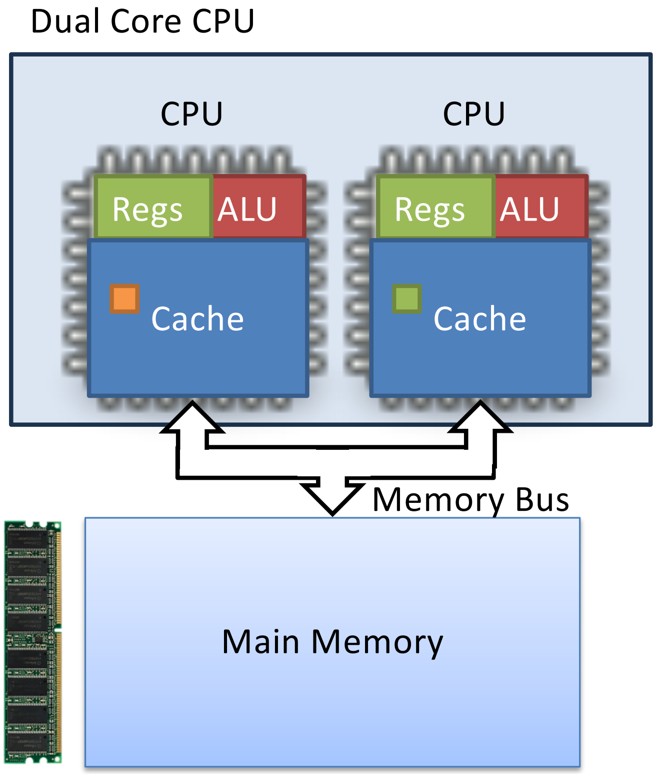
Sells better.
servers/desktops/laptops

Cache Coherence



- Keeping multiple cores’ memory consistent

Cache Coherence



- Keeping multiple cores' memory consistent
- If one core updates data
  - Copy data directly from one cache to the other.
  - Avoid (slower) memory
- Lots of hardware complexity here. We might discuss towards end of semester.

Today

- Types of memory
  - Primary and secondary
- Memory hierarchy
- Abstraction through cache
  - Prediction and locality
- Multi-core CPUs (brief)

Up next:

- Cache details
- How cache is organized
  - finding data
  - storing data
- How cached subset is chosen (eviction)