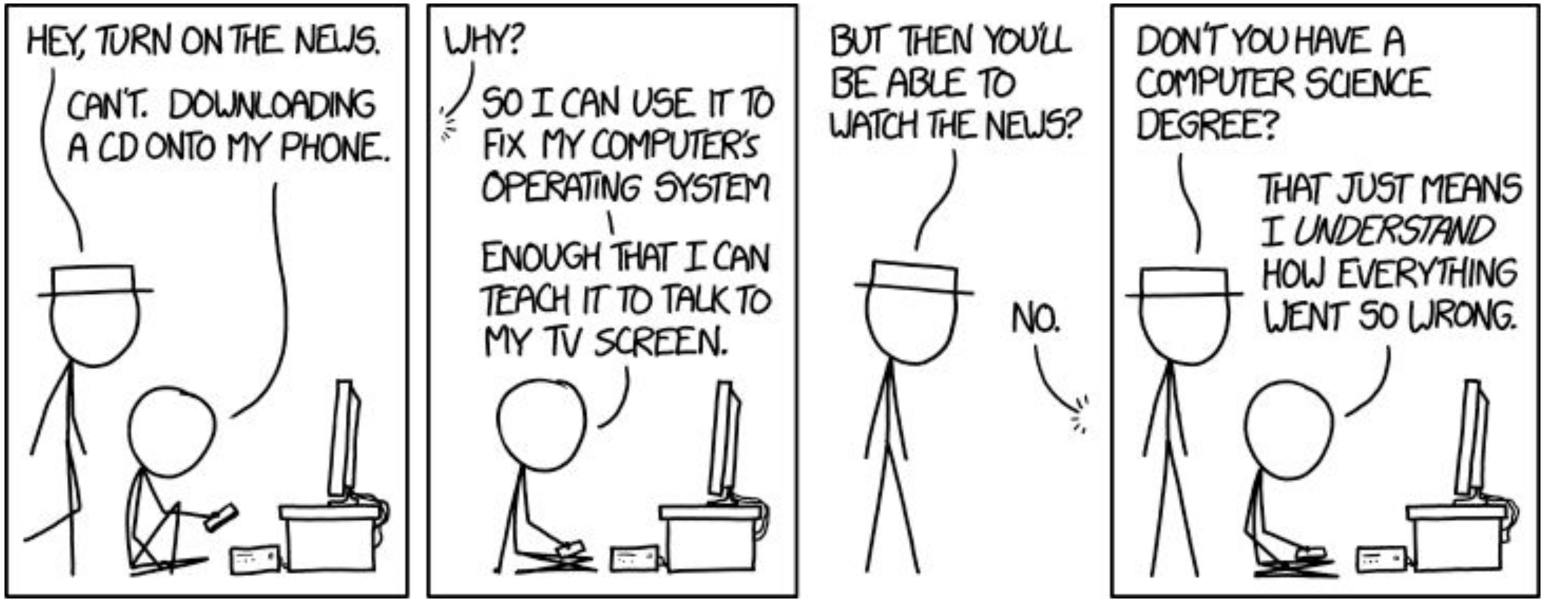
Course Wrap-Up

CSE 351 Summer 2021

Prelude No. 1, J.S. Bach

Instructor:
Mara Kirdani-Ryan

Teaching Assistants:
Kashish Aggarwal
Nick Durand
Colton Jobs
Tim Mandzyuk


HEY, TURN ON THE NEWS.
CAN'T. DOWNLOADING A CD ONTO MY PHONE.
WHY?
SO I CAN USE IT TO FIX MY COMPUTER'S OPERATING SYSTEM
ENOUGH THAT I CAN TEACH IT TO TALK TO MY TV SCREEN.
BUT THEN YOU'LL BE ABLE TO WATCH THE NEWS?
NO.
DON'T YOU HAVE A COMPUTER SCIENCE DEGREE?
THAT JUST MEANS I UNDERSTAND HOW EVERYTHING WENT SO WRONG.

https://xkcd.com/1760/

Course Evaluation Reminder Meme

- Please fill out your course evaluations!! (you should have received a couple emails with a link to the eval)

Gentle, Loving Reminders

- Unit Summary 3 due tonight!

- You're all so wonderful!
  - Imperfect,
  - Wired for struggle,
  - And worthy of love and belonging!
  - Thanks for sticking around till the end!

- Also, my OH are from 2-3pm today

Today

- End-to-end Review
  - What happens after you write your source code?
    - How code becomes a program
    - How your computer executes your code
- Victory lap and high-level concepts (key points)

C: A Low-Level High-Level Language

- C is a "hands-off" language that "exposes" more of hardware (especially memory)
  - Weakly-typed language that stresses data as bits
    - Anything can be represented with a number!
  - Unconstrained pointers can hold address of *anything*
    - And no bounds checking – buffer overflow possible!
  - Efficient by leaving everything up to the programmer
  - "C is good for two things: being beautiful and creating catastrophic 0days in memory management." (https://medium.com/message/everything-is-broken-81e5f33a24e)

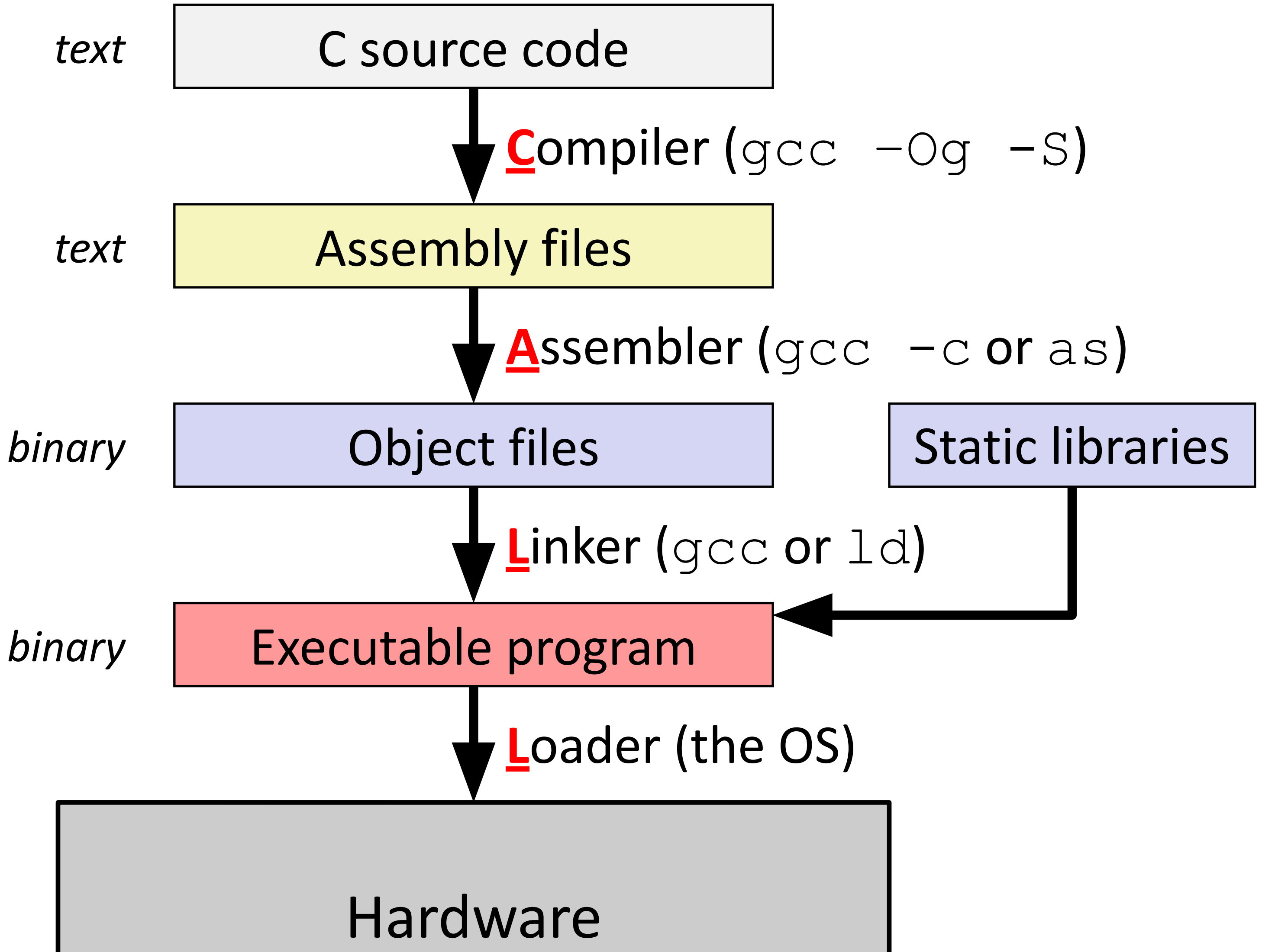
C Data Types

- C Primitive types
  - Fixed sizes and alignments
  - Characters (`char`), Integers (`short`, `int`, `long`), Floating Point (`float`, `double`)
- C Data Structures
  - Arrays – contiguous chunks of memory
    - Multidimensional arrays = still one continuous chunk, but row-major
    - Multi-level arrays = array of pointers to other arrays
  - Structs – structured group of variables
    - Struct fields are ordered according to declaration order
    - ***Internal* fragmentation:** space between members to satisfy member alignment requirements (aligned for each primitive element)
    - ***External* fragmentation:** space after last member to satisfy overall struct alignment requirement (largest primitive member)

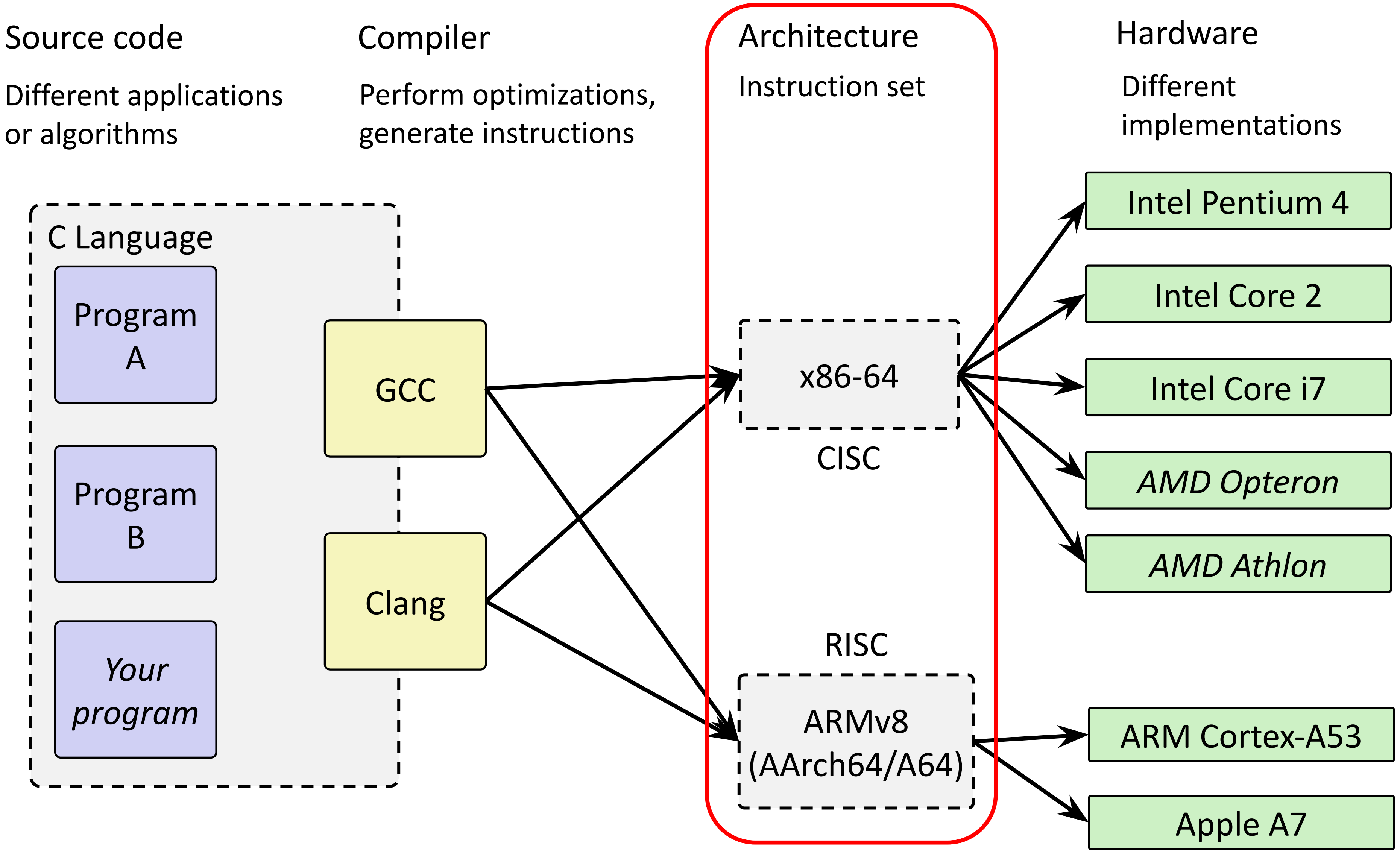
C and Memory

- Using C allowed us to examine how we store and access data in memory
  - Endianness (**only applies to memory**)
    - Is the first byte (lowest address) the least significant (little endian) or most significant (big endian)?
    - Array indices and struct fields result in calculating proper addresses to access
- Consequences of your code:
  - Affects performance (locality)
  - Affects security
- But to understand these effects better, we had to dive deeper…

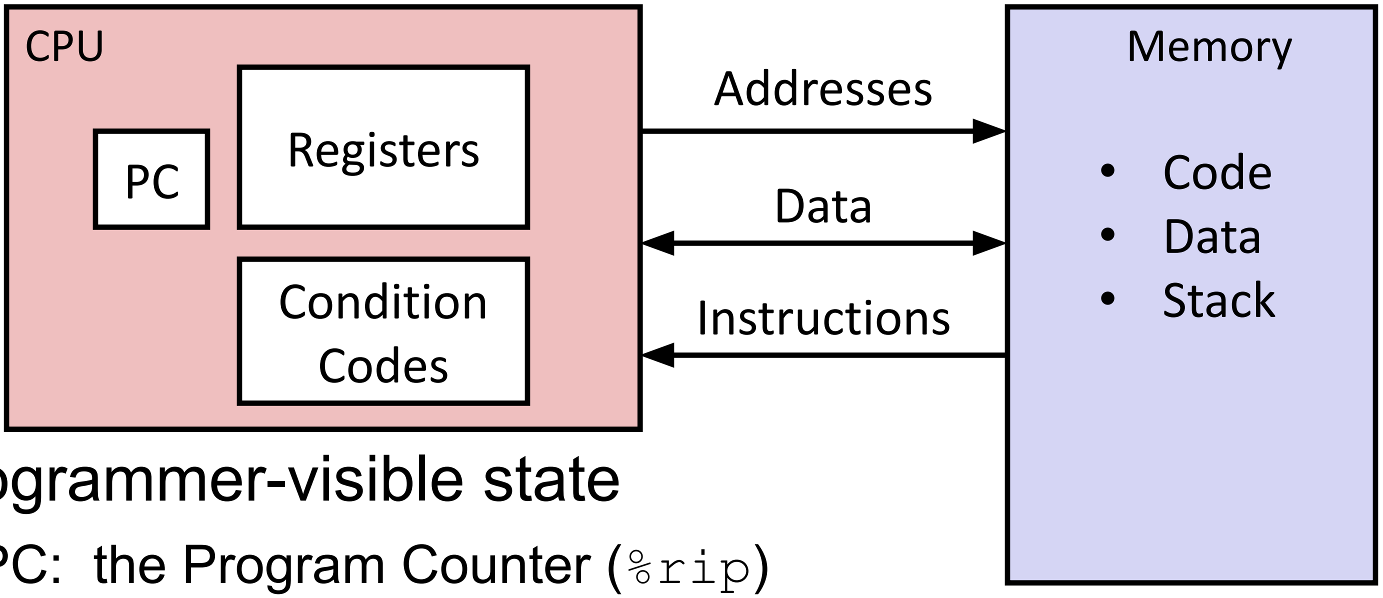
How Code Becomes a Program



Instruction Set Architecture



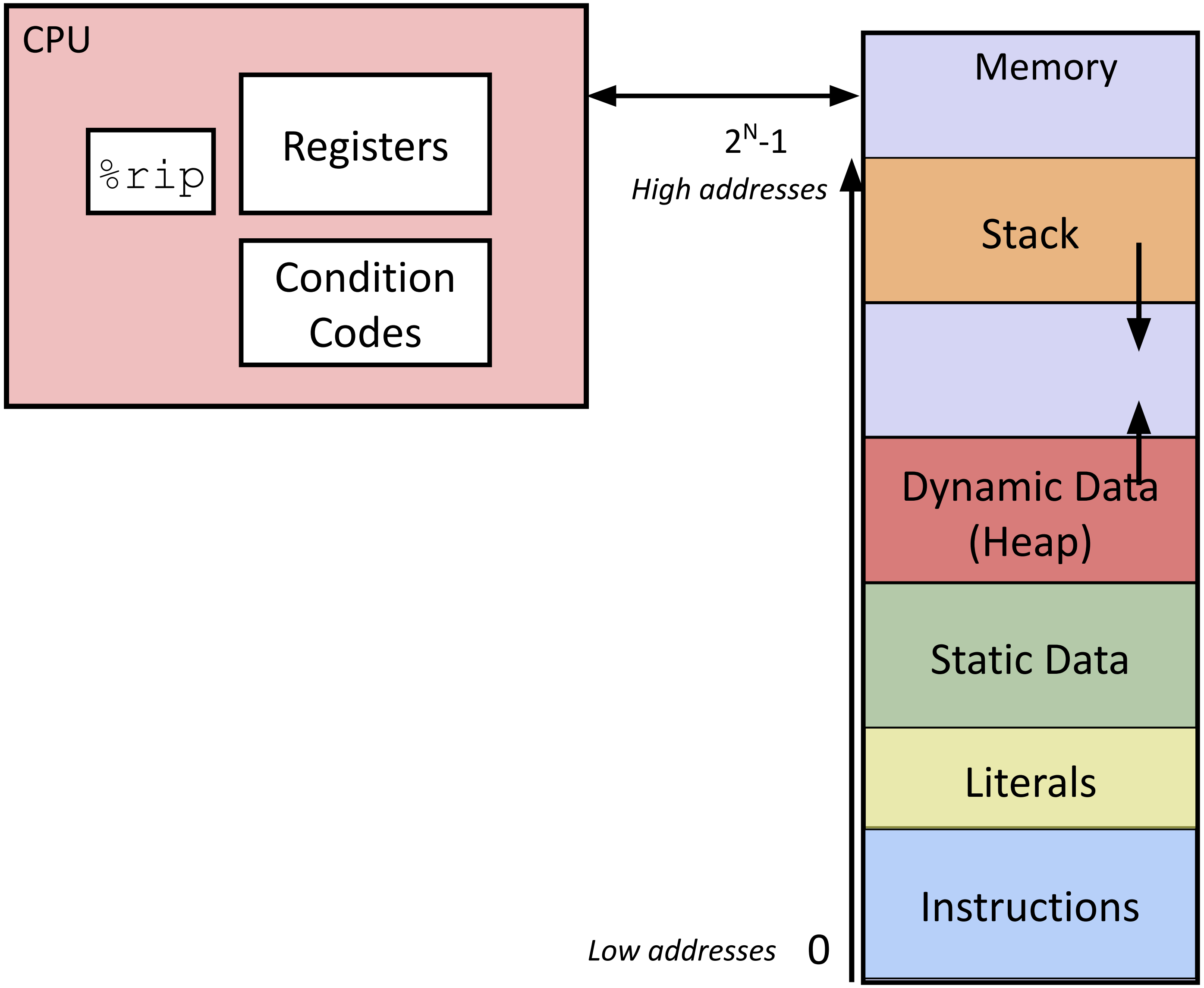
Assembly Programmer's View



- Programmer-visible state
  - PC: the Program Counter (`%rip`)
    - Address of next instruction
  - Named registers
    - Together in "register file"
    - Heavily used program data
  - Condition codes
    - Store status information about most recent arithmetic operation
    - Used for conditional branching

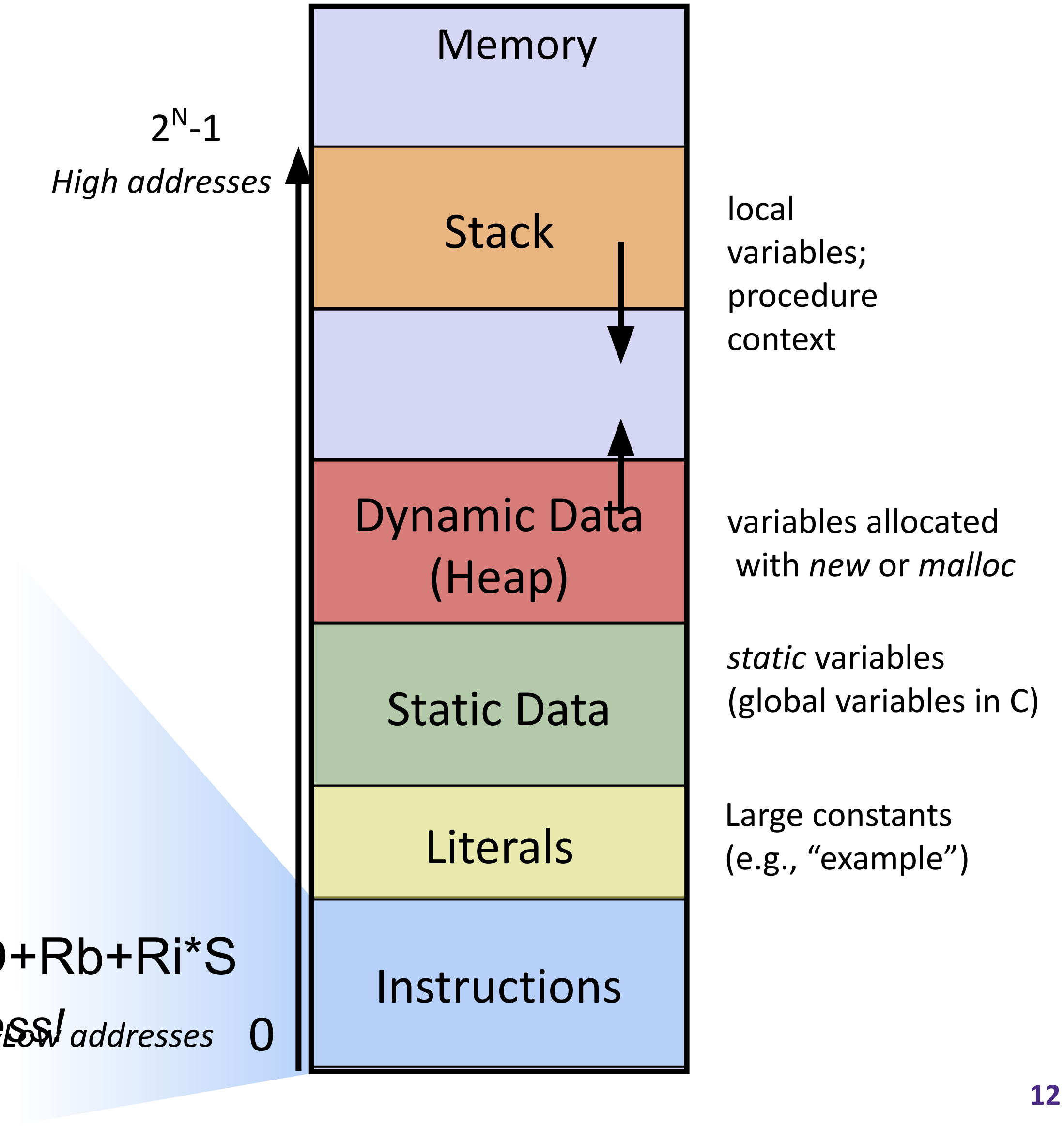
- Memory
  - Byte-addressable array
  - Huge *virtual* address space
  - *Private, all to yourself…*

Program’s View


CPU
%rip
Registers
Condition Codes
Memory
2^N-1
High addresses
Stack
local variables; procedure context
Dynamic Data (Heap)
variables allocated with new or malloc
Static Data
static variables (global variables in C)
Literals
Large constants (e.g., “example”)
Instructions
Low addresses
0

Program’s View

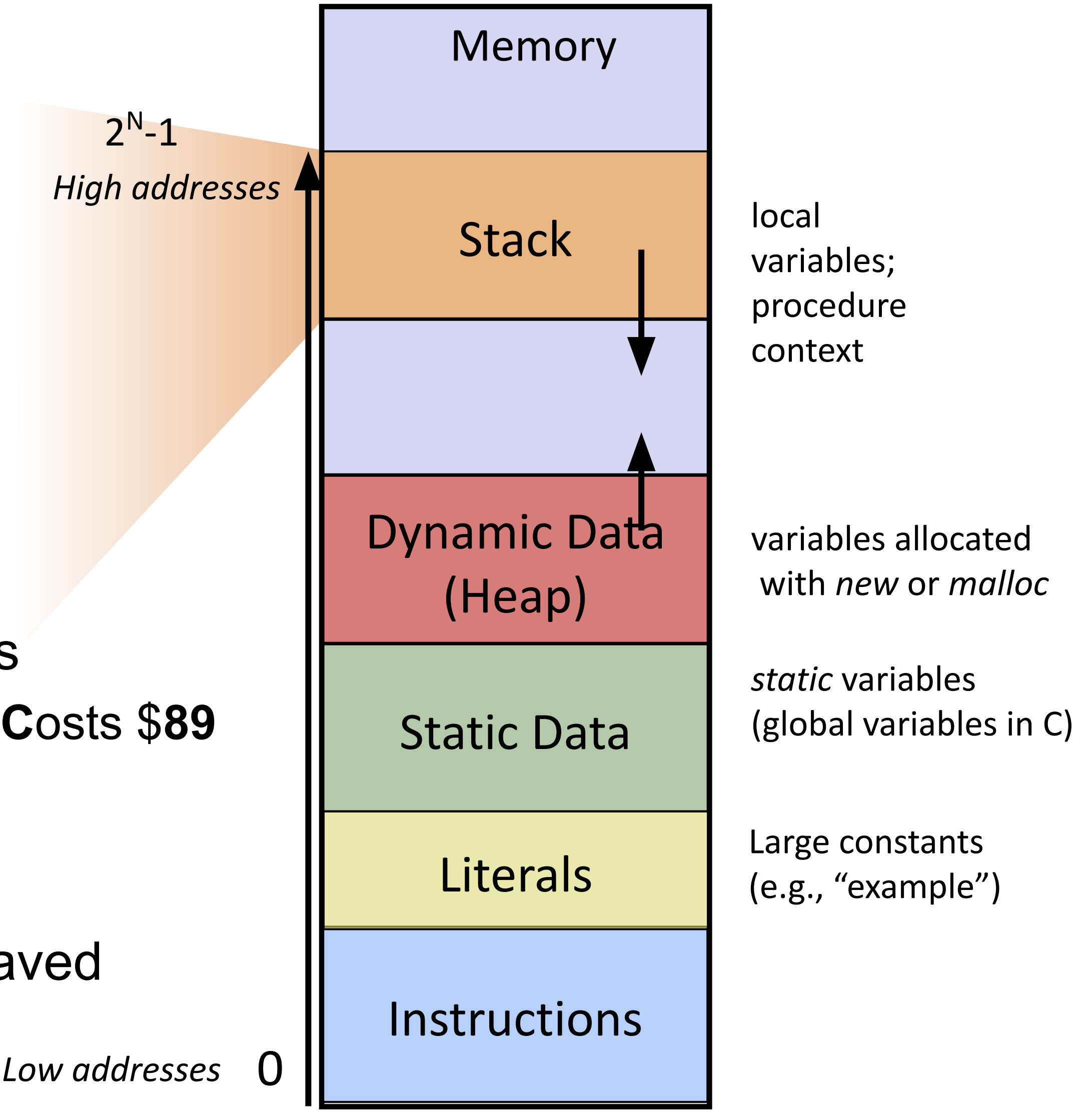
- Instructions
  - Data movement
    - `mov, movz, movz`
    - `push, pop`
  - Arithmetic
    - `add, sub, imul`
  - Control flow
    - `cmp, test`
    - `jmp, je, jgt, ...`
    - `call, ret`
- Operand types
  - Literal: `$8`
  - Register: `%rdi, %al`
  - Memory: D(Rb,Ri,S) = D+Rb+Ri*S
    - `lea`: *not a memory access!*



Memory
2^N-1
High addresses
Stack
local variables; procedure context
Dynamic Data (Heap)
variables allocated with new or malloc
Static Data
static variables (global variables in C)
Literals
Large constants (e.g., “example”)
Instructions
Low addresses
0

Program's View

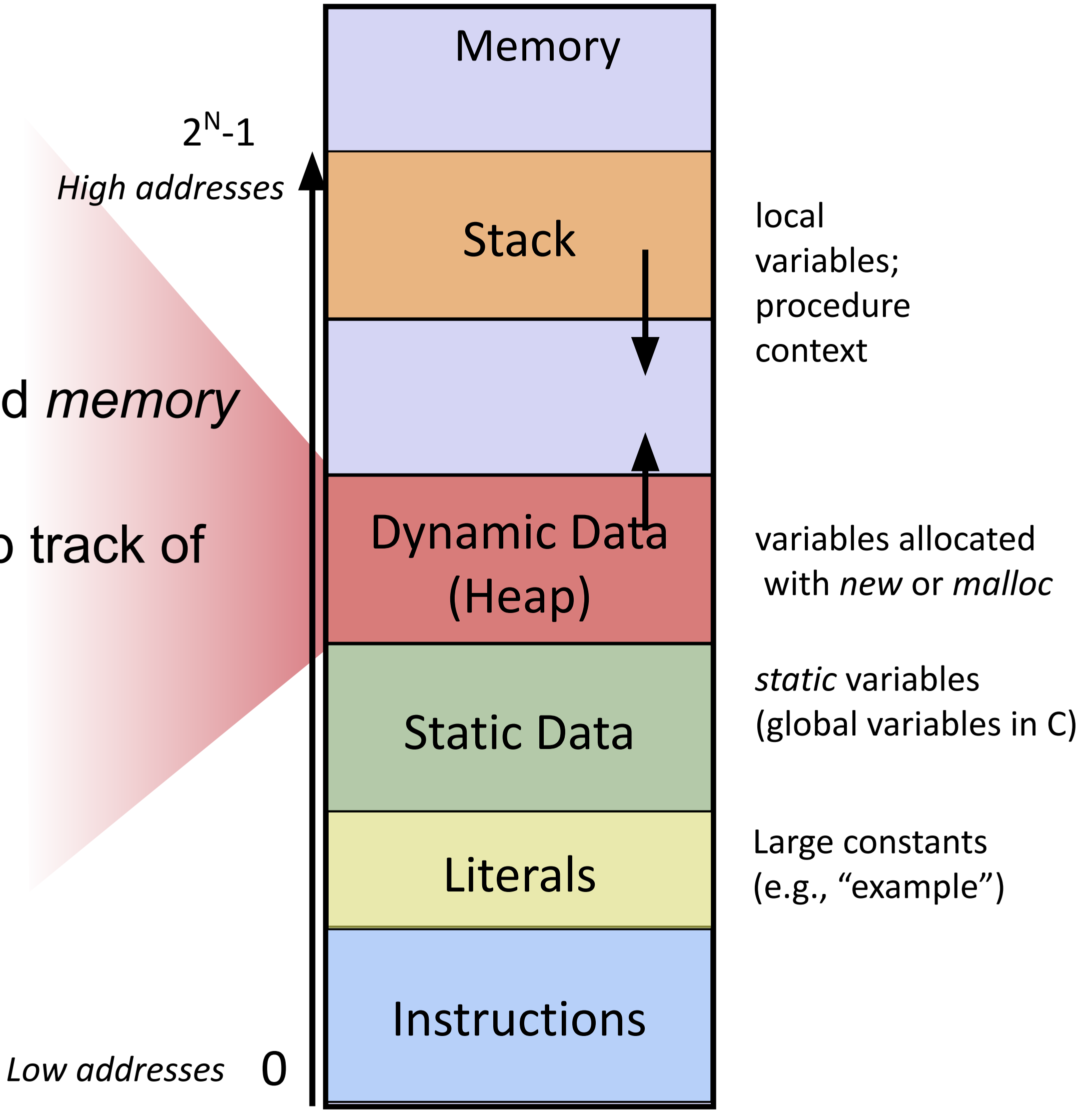
- Procedures
  - Essential abstraction
  - Recursion…
- Stack discipline
  - Stack frame per call
  - Local variables
- Calling convention
  - How to pass arguments
    - **D**ee's **S**nazzy **D**omain **C**osts $**89**
  - How to return data
  - Return address
  - Caller-saved / callee-saved registers



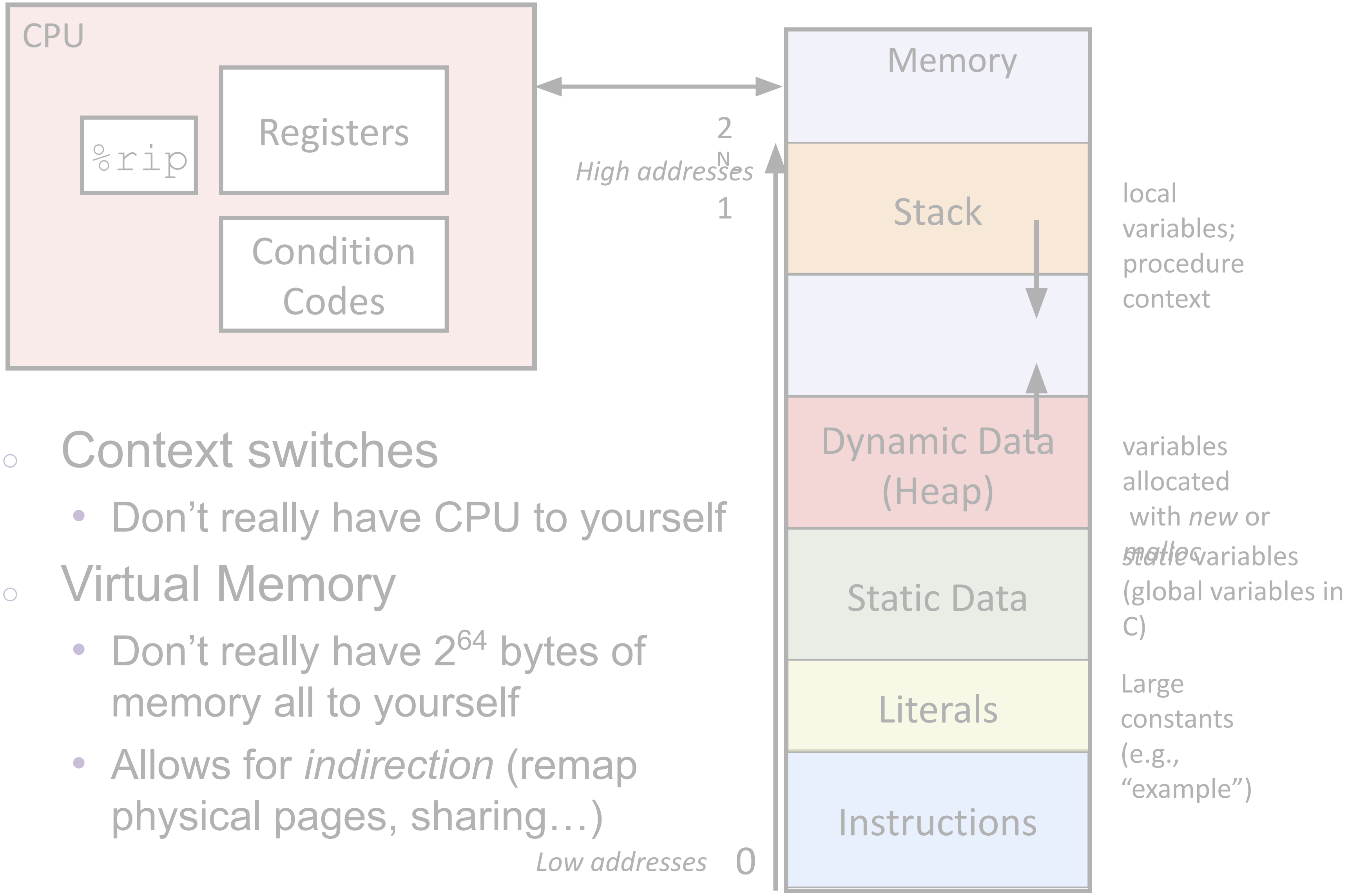
2^N-1
High addresses
Low addresses
0
Memory
Stack
Dynamic Data (Heap)
Static Data
Literals
Instructions
local variables; procedure context
variables allocated with *new* or *malloc*
static variables (global variables in C)
Large constants (e.g., "example")

Program's View

- Heap data
  - Variable size
  - Variable lifetime
- Allocator
  - Balance *throughput* and *memory utilization*
  - Data structures to keep track of free blocks

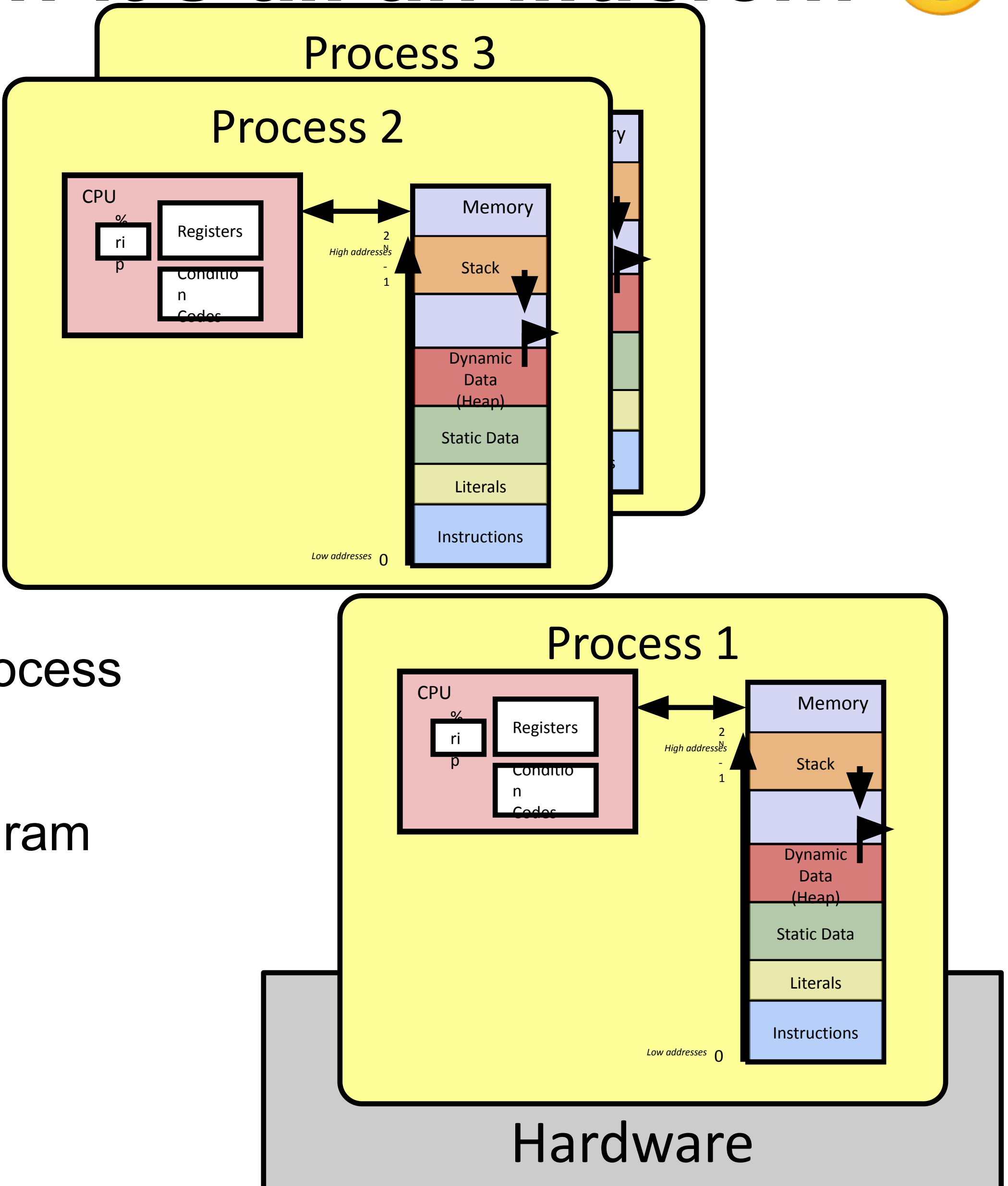

Memory
2^N-1
High addresses
Stack
local variables; procedure context
Dynamic Data (Heap)
variables allocated with new or malloc
Static Data
static variables (global variables in C)
Literals
Large constants (e.g., "example")
Instructions
Low addresses
0

But remember… it's all an *illusion*! 😮


CPU
%rip
Registers
Condition Codes
Memory
$2^N - 1$
High addresses
Stack
Dynamic Data (Heap)
Static Data
Literals
Instructions
Low addresses
0
local variables; procedure context
variables allocated with *new* or *malloc*
static variables (global variables in C)
Large constants (e.g., "example")

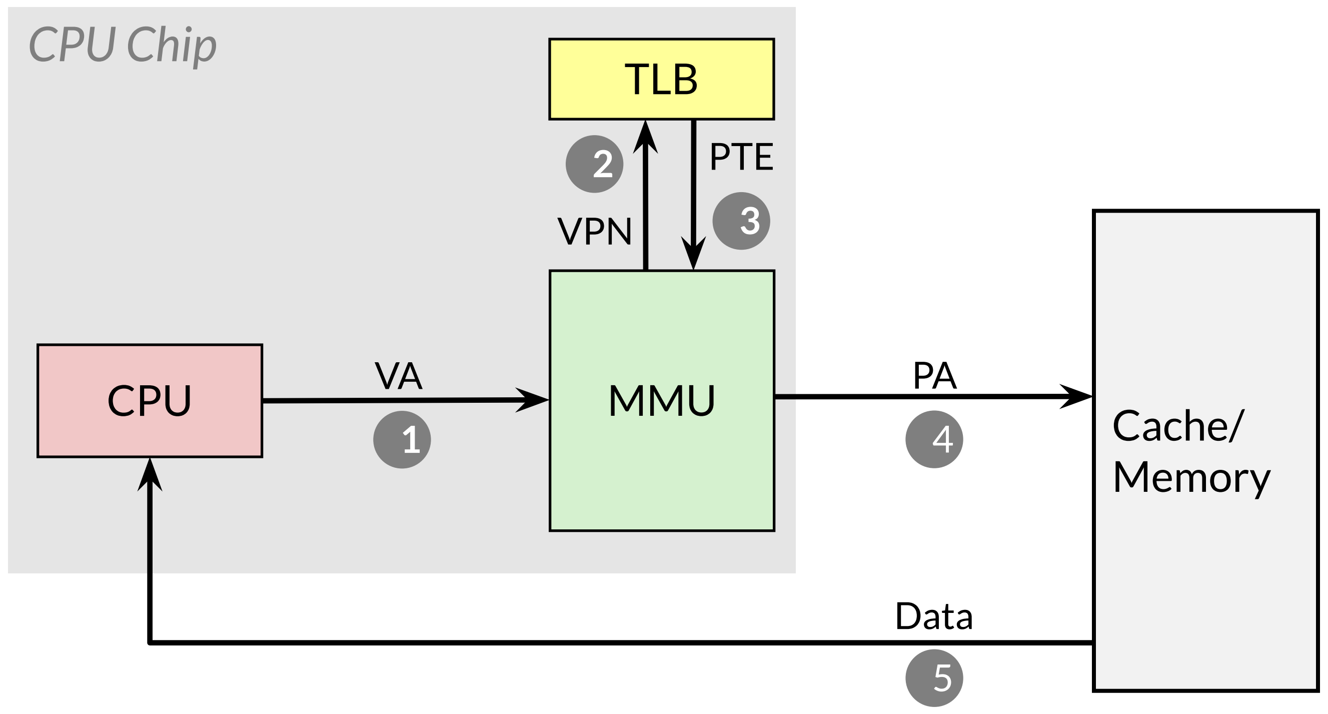
- Context switches
  - Don't really have CPU to yourself
- Virtual Memory
  - Don't really have 2^{64} bytes of memory all to yourself
  - Allows for *indirection* (remap physical pages, sharing…)

But remember… it's all an *illusion*! 😮



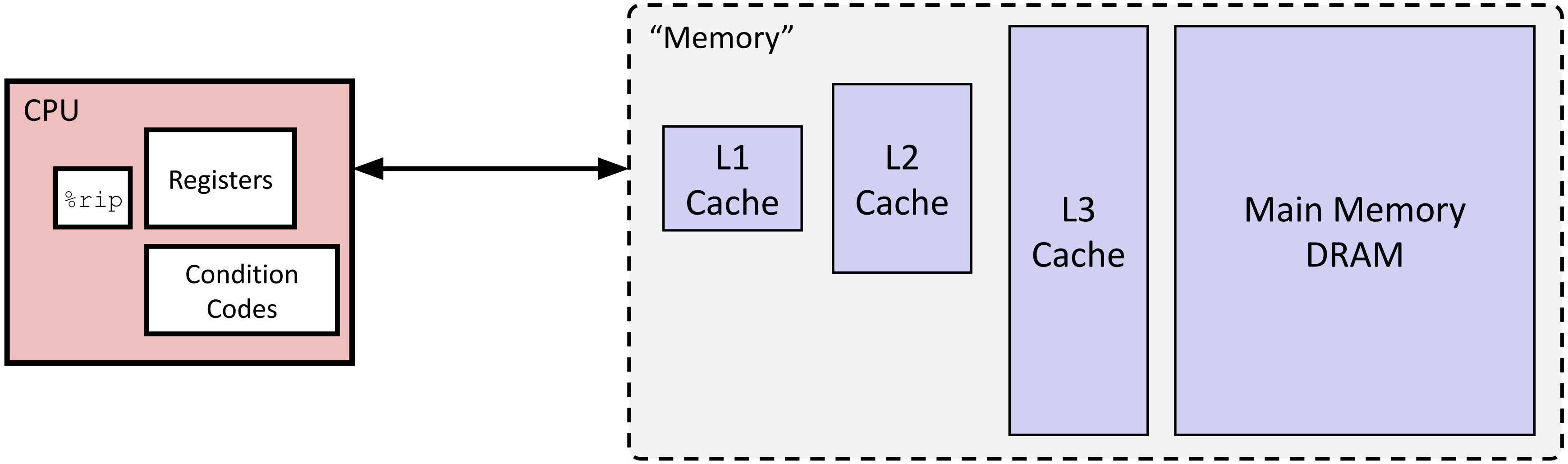
- `fork`
  - Creates copy of the process
- `execv`
  - Replace with new program

Virtual Memory



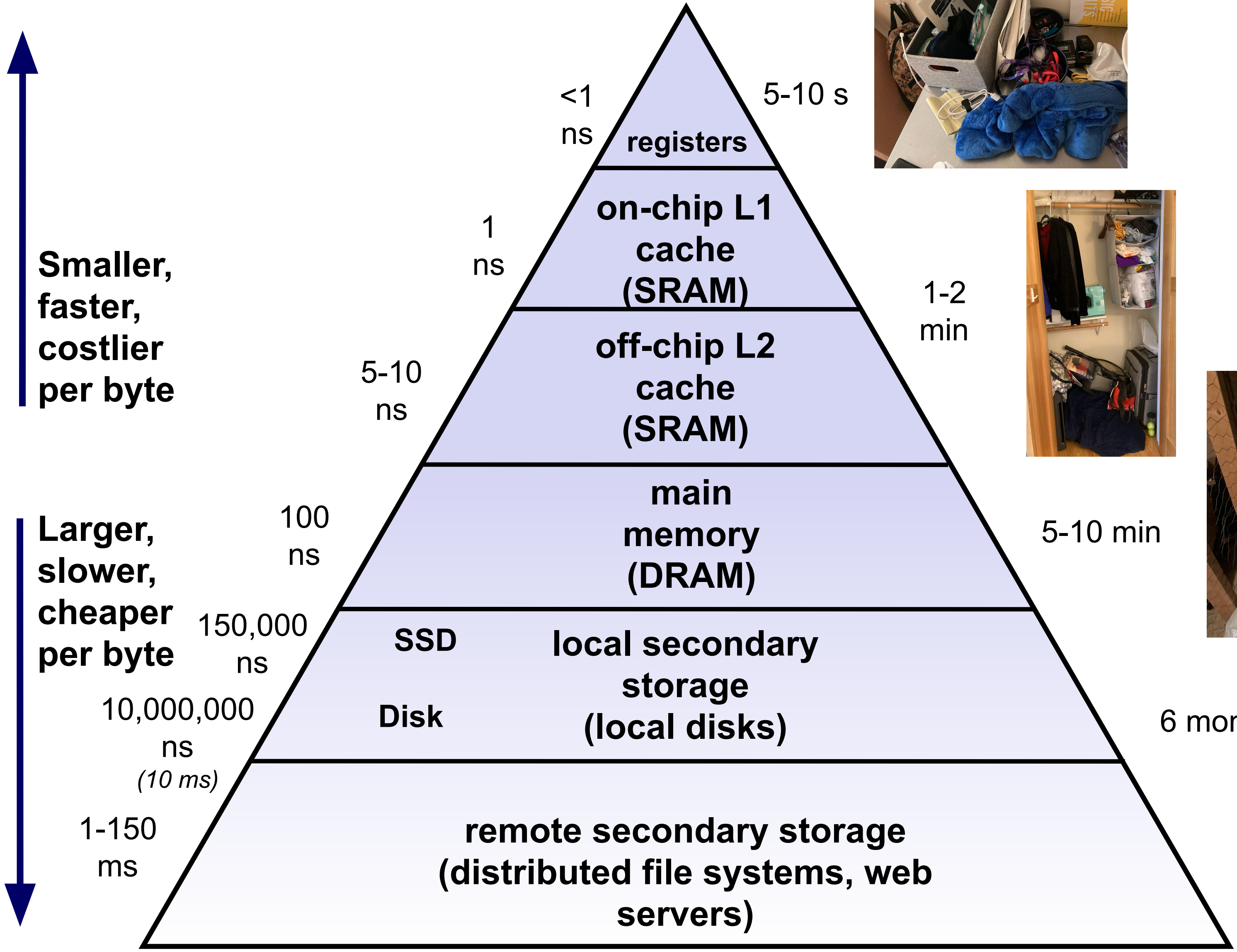
- Address Translation
  - Every memory access must first be converted from virtual to physical
  - *Indirection:* just change the address mapping when switching processes
  - Luckily, TLB (and page size) makes it pretty fast

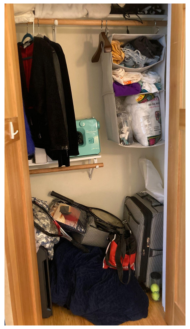
But Memory is Also a Lie! 😮



- *Illusion* of one flat array of bytes
  - But *caches* invisibly make accesses to physical addresses faster!
- Caches
  - **Associativity** tradeoff with miss rate and access time
  - **Block size** tradeoff with spatial and temporal locality
  - **Cache size** tradeoff with miss rate and cost

Memory Hierarchy





Course Themes

Course Perspective

- CSE 351 teaches about computer systems
  - Traditionally, how they *are*, rather than asking “why”
  - Understand ideology to name it later
    - Future courses, future employers
- The House of Computing
  - Some structures need remodeling!
  - Inaccessible, hostile, exclusionary
  - We can’t just burn it down -- what should we change?

Encoding

- All digital systems encode everything as 0s & 1s
  - The 0 and 1 are really two different voltage ranges in the wires
  - Or magnetic positions on a disc, or hole depths on a DVD,
- “Everything” includes:
  - Numbers – integers and floating point
  - Characters – the building blocks of strings
  - Instructions – the directives to the CPU that make up a program
  - Pointers – addresses of data objects stored away in memory
- Encodings are stored in a computer system
  - In registers, caches, memories, disks, etc.
- They all need addresses (a way to locate)
  - Find a new place to put a new item
  - Reclaim the place in memory when data no longer needed

Theme 1: Encoding

- What's so true that it isn't even questioned?
  - What's *ideological*?
- How is that ideology encoded into tools?
  - *We shape our tools, and thereafter, our tools shape us*
  - "Reification" -- making the abstract concrete
- Computing is a non-neutral tool, built by a non-neutral society!
  - We've always had values!
  - We've *always* built our tools around them

- What values are in CS?

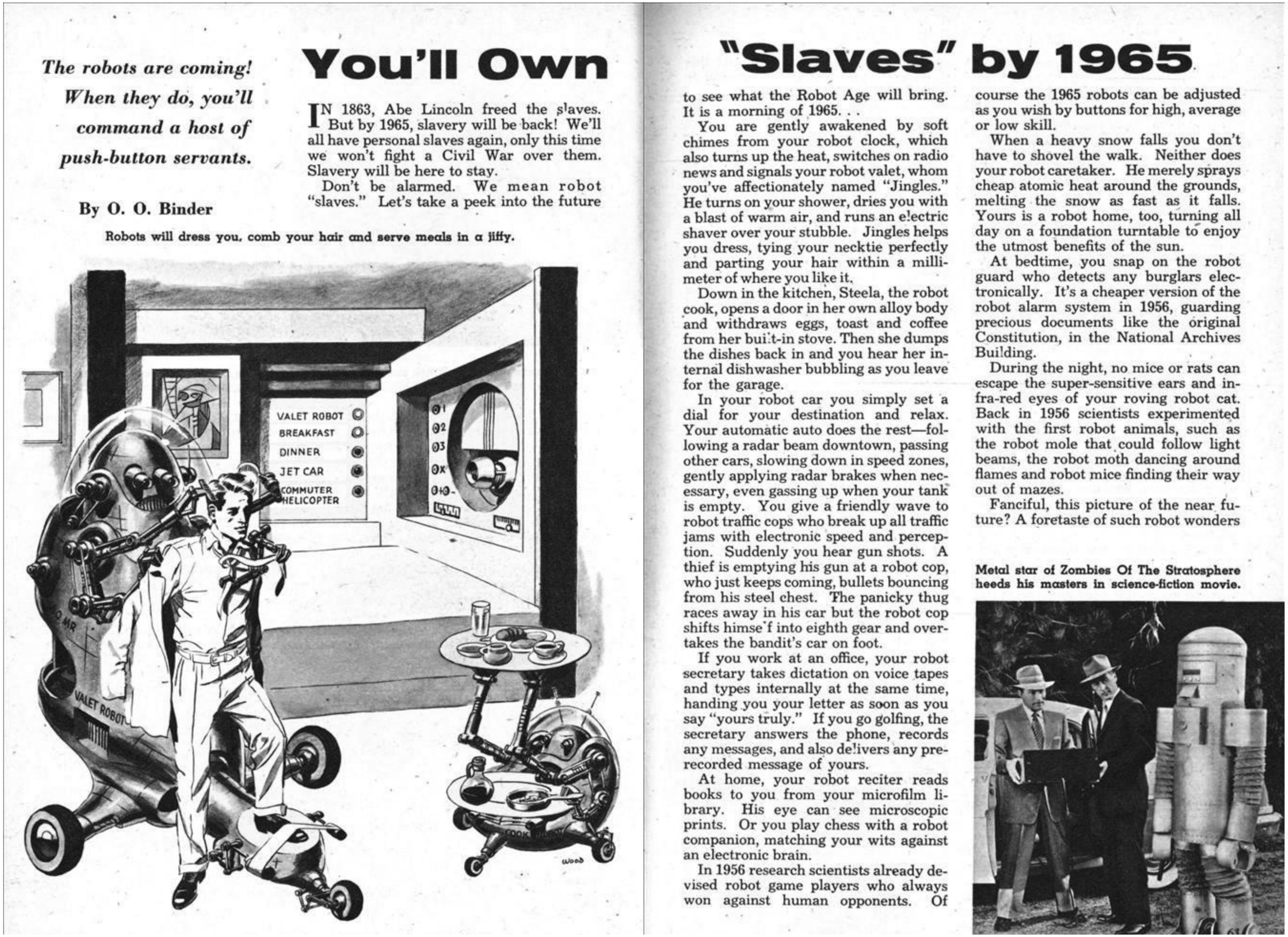
Computing in the US

- **Computer**: one who computes
- Observatory calculations @ Harvard (1870s)

Human Computers at NACA, Credit: NASA

Human Computers at JPL, Credit: JPL

Computer Operators



The robots are coming! When they do, you'll command a host of push-button servants.

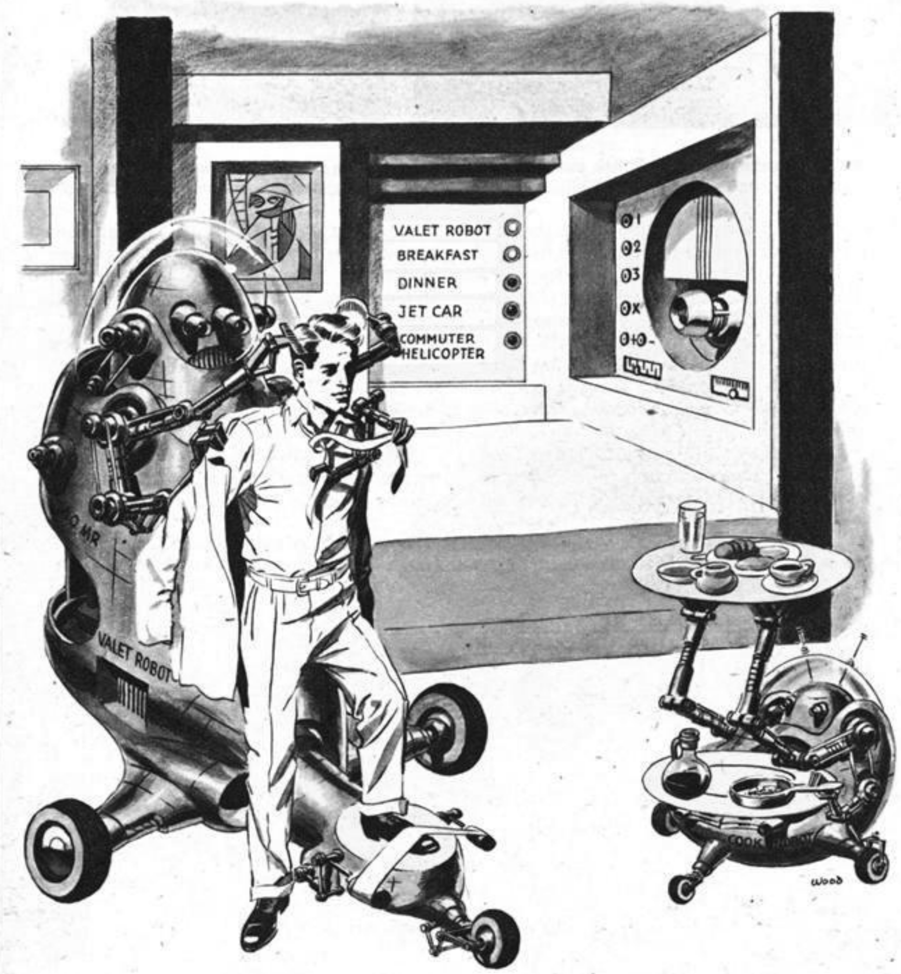
By O. O. Binder

You'll Own "Slaves" by 1965

IN 1863, Abe Lincoln freed the slaves. But by 1965, slavery will be back! We'll all have personal slaves again, only this time we won't fight a Civil War over them. Slavery will be here to stay.

Don't be alarmed. We mean robot "slaves." Let's take a peek into the future to see what the Robot Age will bring. It is a morning of 1965. . .

Robots will dress you, comb your hair and serve meals in a jiffy.



You are gently awakened by soft chimes from your robot clock, which also turns up the heat, switches on radio news and signals your robot valet, whom you've affectionately named "Jingles." He turns on your shower, dries you with a blast of warm air, and runs an electric shaver over your stubble. Jingles helps you dress, tying your necktie perfectly and parting your hair within a millimeter of where you like it.

Down in the kitchen, Steela, the robot cook, opens a door in her own alloy body and withdraws eggs, toast and coffee from her built-in stove. Then she dumps the dishes back in and you hear her internal dishwasher bubbling as you leave for the garage.

In your robot car you simply set a dial for your destination and relax. Your automatic auto does the rest—following a radar beam downtown, passing other cars, slowing down in speed zones, gently applying radar brakes when necessary, even gassing up when your tank is empty. You give a friendly wave to robot traffic cops who break up all traffic jams with electronic speed and perception. Suddenly you hear gun shots. A thief is emptying his gun at a robot cop, who just keeps coming, bullets bouncing from his steel chest. The panicky thug races away in his car but the robot cop shifts himself into eighth gear and overtakes the bandit's car on foot.

If you work at an office, your robot secretary takes dictation on voice tapes and types internally at the same time, handing you your letter as soon as you say "yours truly." If you go golfing, the secretary answers the phone, records any messages, and also delivers any prerecorded message of yours.

At home, your robot reciter reads books to you from your microfilm library. His eye can see microscopic prints. Or you play chess with a robot companion, matching your wits against an electronic brain.

In 1956 research scientists already devised robot game players who always won against human opponents. Of course the 1965 robots can be adjusted as you wish by buttons for high, average or low skill.

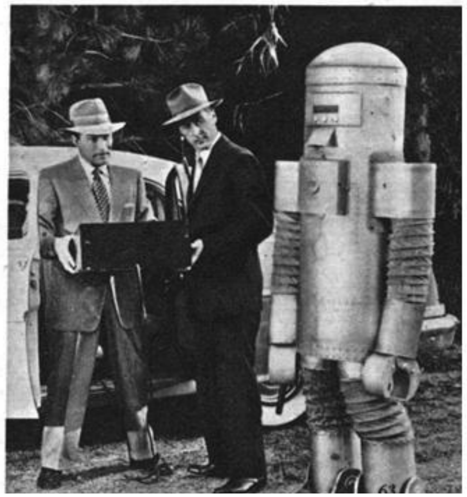
When a heavy snow falls you don't have to shovel the walk. Neither does your robot caretaker. He merely sprays cheap atomic heat around the grounds, melting the snow as fast as it falls. Yours is a robot home, too, turning all day on a foundation turntable to enjoy the utmost benefits of the sun.

At bedtime, you snap on the robot guard who detects any burglars electronically. It's a cheaper version of the robot alarm system in 1956, guarding precious documents like the original Constitution, in the National Archives Building.

During the night, no mice or rats can escape the super-sensitive ears and infra-red eyes of your roving robot cat. Back in 1956 scientists experimented with the first robot animals, such as the robot mole that could follow light beams, the robot moth dancing around flames and robot mice finding their way out of mazes.

Fanciful, this picture of the near future? A foretaste of such robot wonders

Metal star of Zombies Of The Stratosphere heeds his masters in science-fiction movie.



Modern Hardware: Historic Relics

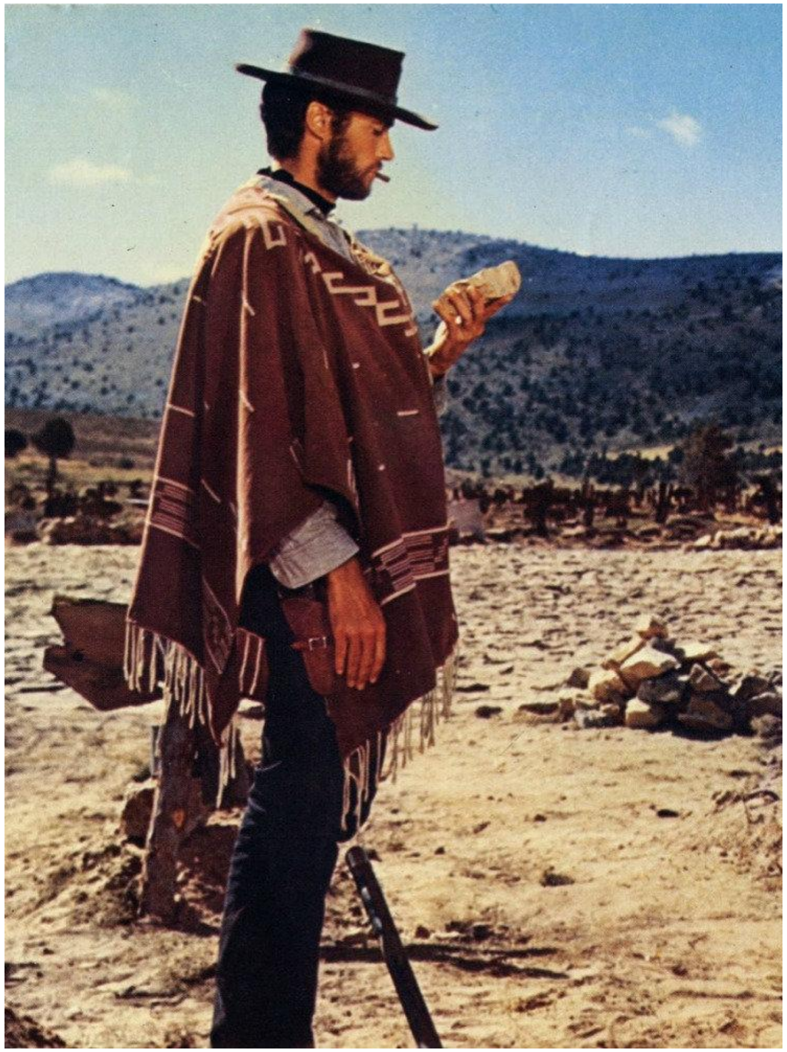
1. "Boring, repetitive work" should be automated
   - Two narratives, both true: Automation/Augmentation
   - Consistently eliminating jobs of marginalized folks
2. Boring, repetitive work is "robot work"
   - Robot work should be performed by robots
     - *"Robot work" anything unvalued by the powerful, once including computing, programming*
   - If the task can't be automated, use people
     - Frequently, this ends up being marginalized people
3. Augmentation is highly valued, and exclusive
   - "Boring, repetitive work" is more available

- We'll get to earlier computational machines later!

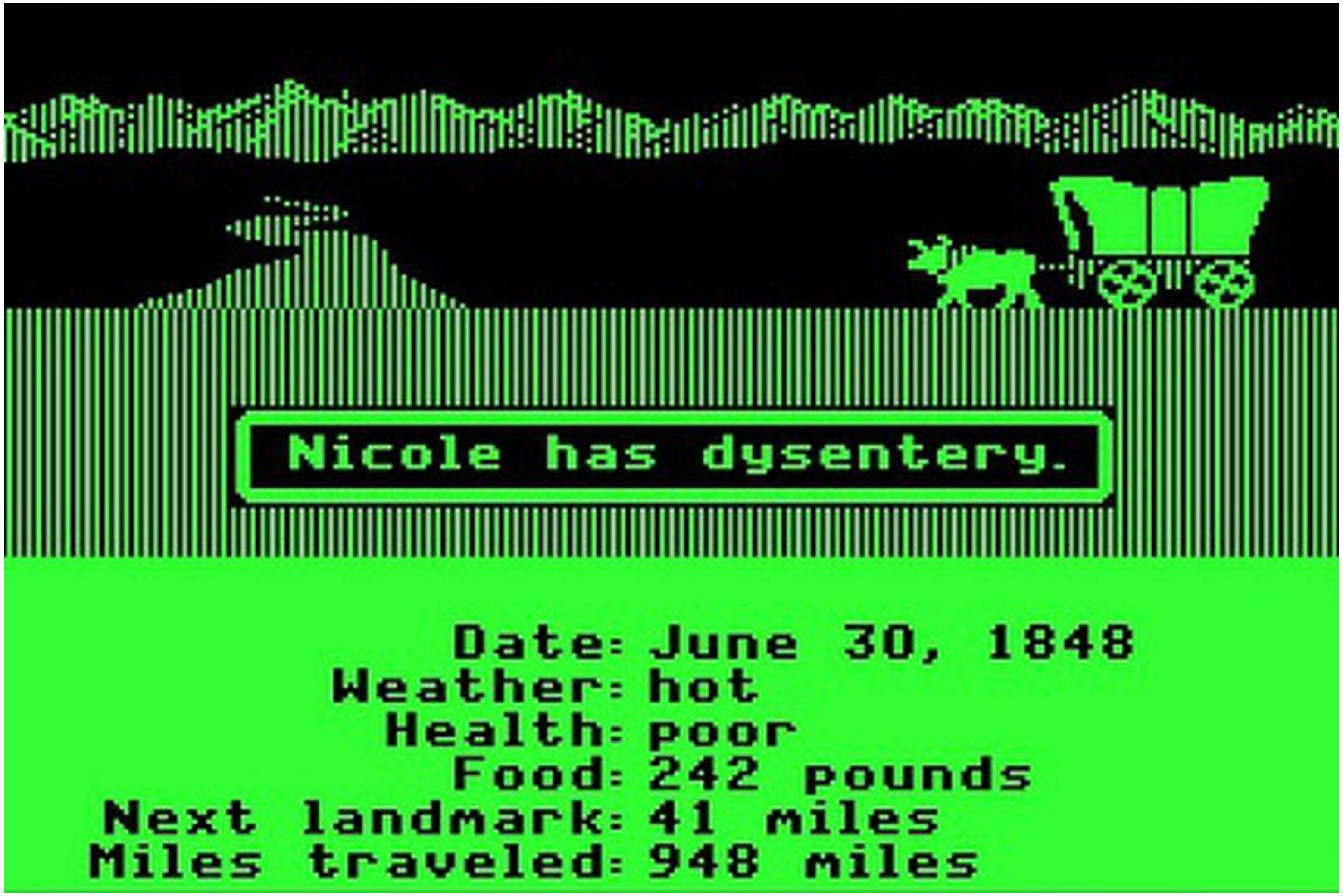
Principles of C, viewed today

- **Minimalist**:
  - *“Since C is relatively small, it can be described in small space, and learned quickly.”*
  - "Only the bare essentials”
- **Rugged**:
  - “Close to the *hardware*”
  - “Shows what’s *really happening*”
- **Individualistic**
  - “No one to help you”
  - “You’re on your own”
  - “I know what I’m doing, get out of my way”

Immortalized in Popular Culture

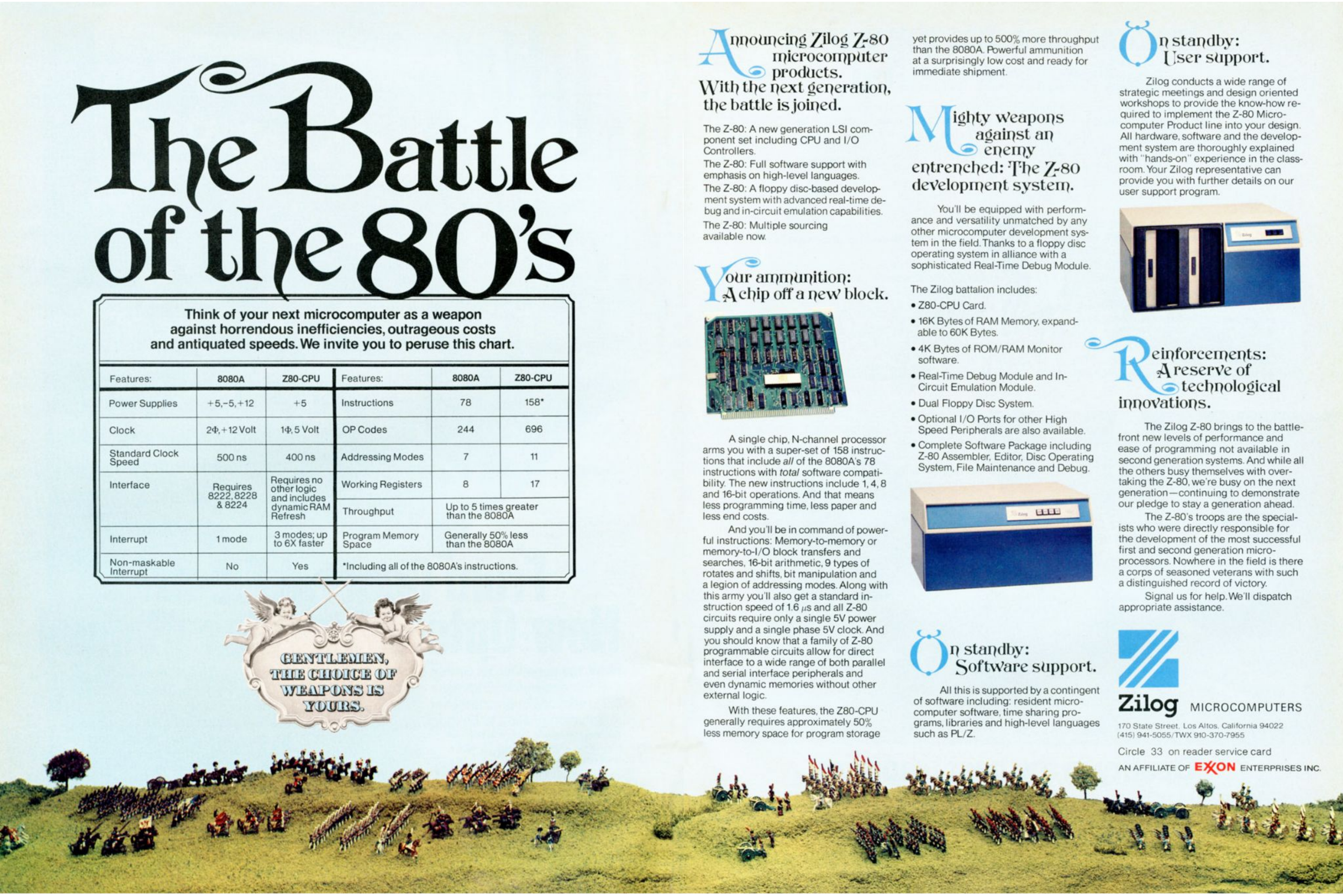


Replicated in Computing Culture



Nicole has dysentery.

```
       Date: June 30, 1848
    Weather: hot
     Health: poor
       Food: 242 pounds
Next landmark: 41 miles
Miles traveled: 948 miles
```



The Battle of the 80's

Think of your next microcomputer as a weapon against horrendous inefficiencies, outrageous costs and antiquated speeds. We invite you to peruse this chart.

Features:	8080A	Z80-CPU	Features:	8080A	Z80-CPU
Power Supplies	+5,−5,+12	+5	Instructions	78	158*
Clock	2Φ,+12 Volt	1Φ,5 Volt	OP Codes	244	696
Standard Clock Speed	500 ns	400 ns	Addressing Modes	7	11
Interface	Requires 8222,8228 & 8224	Requires no other logic and includes dynamic RAM Refresh	Working Registers	8	17
			Throughput	Up to 5 times greater than the 8080A	
Interrupt	1 mode	3 modes; up to 6X faster	Program Memory Space	Generally 50% less than the 8080A	
Non-maskable Interrupt	No	Yes	*Including all of the 8080A's instructions.		

GENTLEMEN, THE CHOICE OF WEAPONS IS YOURS.

Announcing Zilog Z-80 microcomputer products. With the next generation, the battle is joined.

The Z-80: A new generation LSI component set including CPU and I/O Controllers.

The Z-80: Full software support with emphasis on high-level languages.

The Z-80: A floppy disc-based development system with advanced real-time debug and in-circuit emulation capabilities.

The Z-80: Multiple sourcing available now.

Your ammunition: A chip off a new block.

A single chip, N-channel processor arms you with a super-set of 158 instructions that include *all* of the 8080A's 78 instructions with *total* software compatibility. The new instructions include 1, 4, 8 and 16-bit operations. And that means less programming time, less paper and less end costs.

And you'll be in command of powerful instructions: Memory-to-memory or memory-to-I/O block transfers and searches, 16-bit arithmetic, 9 types of rotates and shifts, bit manipulation and a legion of addressing modes. Along with this army you'll also get a standard instruction speed of 1.6 μs and all Z-80 circuits require only a single 5V power supply and a single phase 5V clock. And you should know that a family of Z-80 programmable circuits allow for direct interface to a wide range of both parallel and serial interface peripherals and even dynamic memories without other external logic.

With these features, the Z80-CPU generally requires approximately 50% less memory space for program storage yet provides up to 500% more throughput than the 8080A. Powerful ammunition at a surprisingly low cost and ready for immediate shipment.

Mighty weapons against an enemy entrenched: The Z-80 development system.

You'll be equipped with performance and versatility unmatched by any other microcomputer development system in the field. Thanks to a floppy disc operating system in alliance with a sophisticated Real-Time Debug Module.

The Zilog battalion includes:

- Z80-CPU Card.
- 16K Bytes of RAM Memory, expandable to 60K Bytes.
- 4K Bytes of ROM/RAM Monitor software.
- Real-Time Debug Module and In-Circuit Emulation Module.
- Dual Floppy Disc System.
- Optional I/O Ports for other High Speed Peripherals are also available.
- Complete Software Package including Z-80 Assembler, Editor, Disc Operating System, File Maintenance and Debug.

On standby: Software support.

All this is supported by a contingent of software including: resident microcomputer software, time sharing programs, libraries and high-level languages such as PL/Z.

On standby: User support.

Zilog conducts a wide range of strategic meetings and design oriented workshops to provide the know-how required to implement the Z-80 Microcomputer Product line into your design. All hardware, software and the development system are thoroughly explained with "hands-on" experience in the classroom. Your Zilog representative can provide you with further details on our user support program.

Reinforcements: A reserve of technological innovations.

The Zilog Z-80 brings to the battlefront new levels of performance and ease of programming not available in second generation systems. And while all the others busy themselves with overtaking the Z-80, we're busy on the next generation—continuing to demonstrate our pledge to stay a generation ahead.

The Z-80's troops are the specialists who were directly responsible for the development of the most successful first and second generation microprocessors. Nowhere in the field is there a corps of seasoned veterans with such a distinguished record of victory.

Signal us for help. We'll dispatch appropriate assistance.

Zilog MICROCOMPUTERS

170 State Street, Los Altos, California 94022
(415) 941-5055/TWX 910-370-7955

Circle 33 on reader service card

AN AFFILIATE OF EXXON ENTERPRISES INC.

Computing tends to value efficiency over humanity.

Along with militarism, war, conquest, individualism

Maybe we could have different values?

Peace, Antiracism, justice?

Translation

- There is a big gap between how we think about programs and data and the 0s and 1s of computers
  - Need languages to describe what we mean
  - These languages need to be translated one level at a time
- We know Java as a programming language
  - Have to work our way down to the 0s and 1s of computers
  - Try not to lose anything in translation!
  - We encountered C language, assembly language, and machine code (for the x86 family of CPU architectures)

What’s prioritized in this translation?

Regardless of what we build, the way that we define success shapes the systems we build!

Choose your metrics carefully!
There’s more to choose than “performance”!
i.e. usability, access, simplicity, agency

Metrics are a “heading”

And, most often, and ideological one.

Best to reevaluate from time to time!

Individual vs. Structure

- There's lots of examples, especially in tech
  - Privacy is a commodity
  - People should know better than to click on ads
  - "You're a bad person if you don't recycle"
  - Everyone should aim for zero-waste
  - Don't compare floats for equality
  - Remember to check array bounds in C
  - "If you can't access *x*, you shouldn't use *x*"
  - …
  - …
  - …

If someone shames you for not knowing something, and they try to take away your "CS card"…

… Ask them, "Who hurt you?"
You'll maintain the moral high ground!

You can do whatever you want, you just have to know what that is.

Choices as Information Gathering

- Careers are (hopefully) pretty long
  - Many opportunities for change!
- UW Quarters are pretty short
  - But, a great opportunity to try something different!

- Decisions are more about making uninformed choices to get more information
  - Try something! Worst case, you find out you hate it.
  - Either way, you learn more about yourself!
    - Knowledge you can carry into future decisions

Future Choices

- Lots of courses near HW/SW interface
  - Computer Architecture: what happens inside a CPU?
  - Digital Design: how do we design hardware?
  - Embedded Systems: what's different when computer systems are specialized?
  - Programming Languages, Compilers
  - Data Structures & Parallelism
  - Systems Programming:
  - Operating Systems
  - Networks
  - Security

Lots of choices

- HCI: how can we build systems that interact with people better?
- CSE's Ethics seminar, *but also*
- INFO 102: Gender & Info Technology
- INFO 350: Information Ethics & Policy
- EDUC 251: Seeking Educational Equity & Diversity

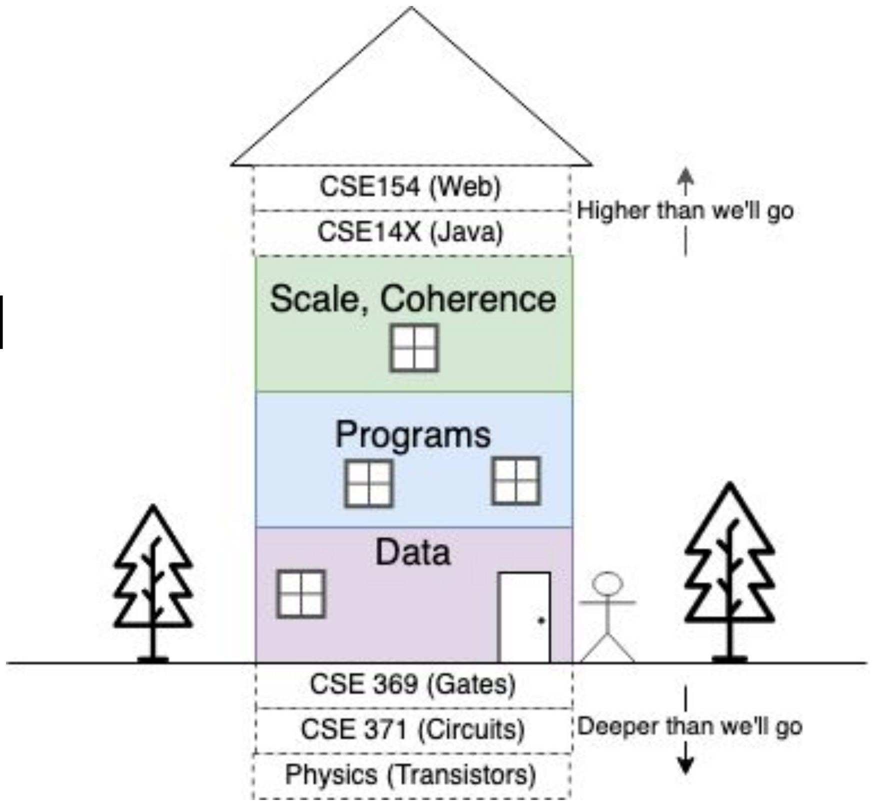
- Or, go learn about whatever else you want!
  - This is the most intellectually diverse space that you'll be in, seems like a bit of a shame to only look at CS

"More and more, the oppressors are using science and technology as unquestionably powerful instruments for their purpose, the maintenance of the oppressive order thorough manipulation and repression"

Paulo Freire, 1970

Remodeling, and you

- Things *need* to be fixed!
- Understand the foundation before you start hammering!
- Technical know-how and socio-technical understanding together!



Everyone deserves support & agency.

Everyone deserves a space that feels safe.

CS as a refuge

- It’s been the only place that’s safe for:
  - Women, whose jobs were automated by computers
  - Bullied kids that liked calculators
  - Wealthy “young geeks” (me)
  - Closeted trans kids (me)
  - Autistic people (me)
  - Queer folks trying to move to more progressive cities (SF/Seattle)
  - Those seeking social mobility, especially now

Remodeling the house of computing

- If you go in and start swinging hammers, you're going to scare some people
  - Fear sometimes manifests as aggression, especially among folks socialized as men
    - I only had access to anger for 15 years

- You might seem like an oppressor!
  - Even to those who've been oppressing others!
- There aren't many spaces where neurodiversity is valued!
  - There aren't many spaces that are safe!

You have unprecedented power and access as technologists.

What would you like to accomplish?
Who do you want to serve?

Ideally, better than "move fast and break things"

Please, be reflexive!

Understand, as much as you can, who you are before you build!

Some unsolicited career advice

The shortest career guide ever

- Try to know yourself first! No other option.
  - But, steps along the way help you learn more
- Jobs don't define careers, you do!
- Each job is information gathering
  - What you like, what you don't like
  - What you can stand, what you can't
  - What's exciting, what's boring

Some questions for reflection

- What do you want from your first job (or internship, or whatever)?
- Why do you want those things? Why those places?
- Why those places and not other places?

- When you justify, who’s talking? Are you sure it’s you?

Who else might be speaking?

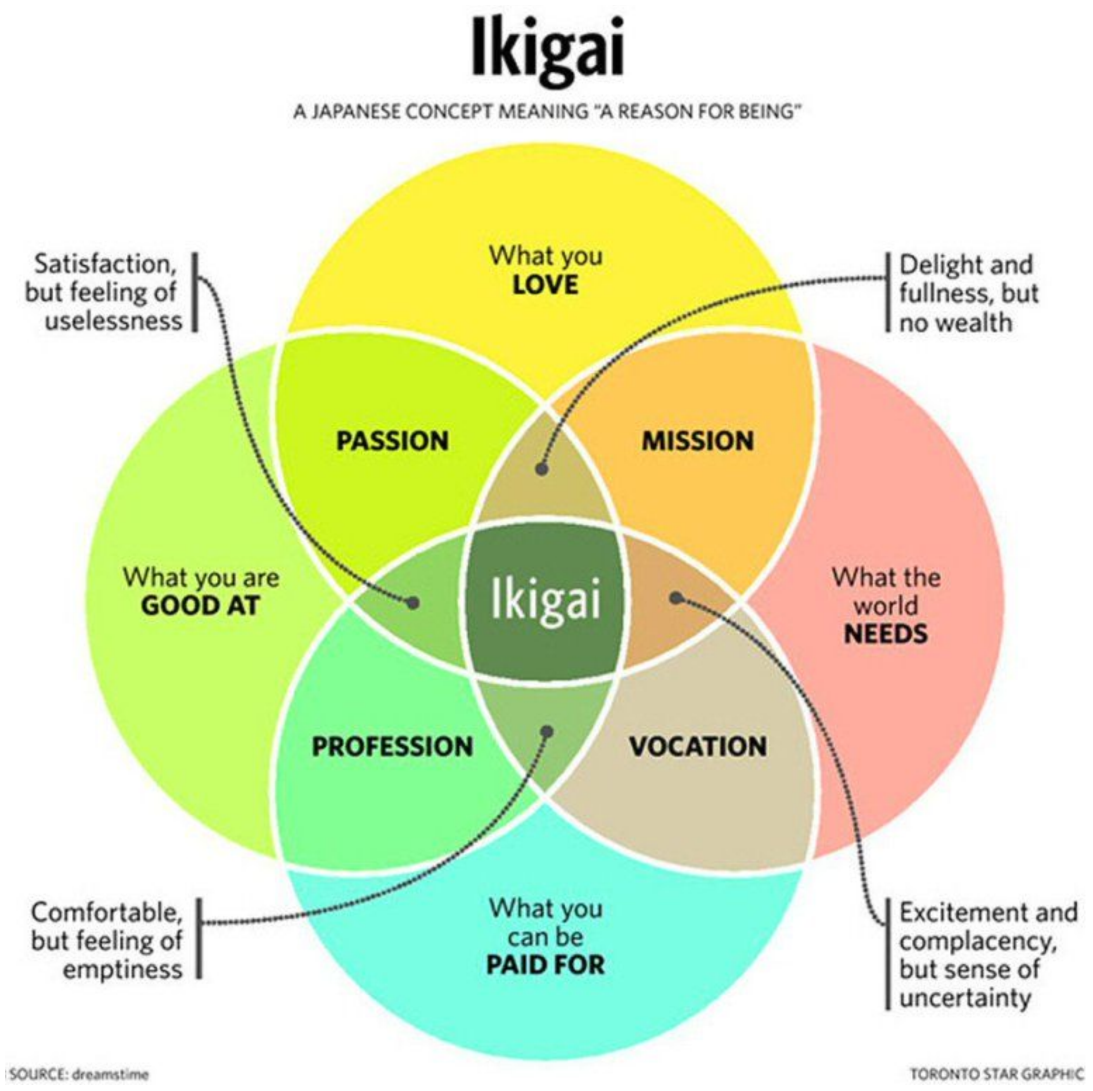
- Most career aspirations tend around “legitimate” jobs/careers in X
  - As an example, what jobs are viewed as legitimate in the Allen School? Which jobs are lauded over others? Why?
- As social humans, we get ideas of legitimacy from parents, friends, institutions, basically from groups **that we see as part of our identity**
  - For instance, if you “identify as” a computer science major, you might be beholden to what computer science major deem as legitimate jobs
- From Wenger, Bourdieu, Holland, etc: **Identifying with (and being identified by) leads one to**

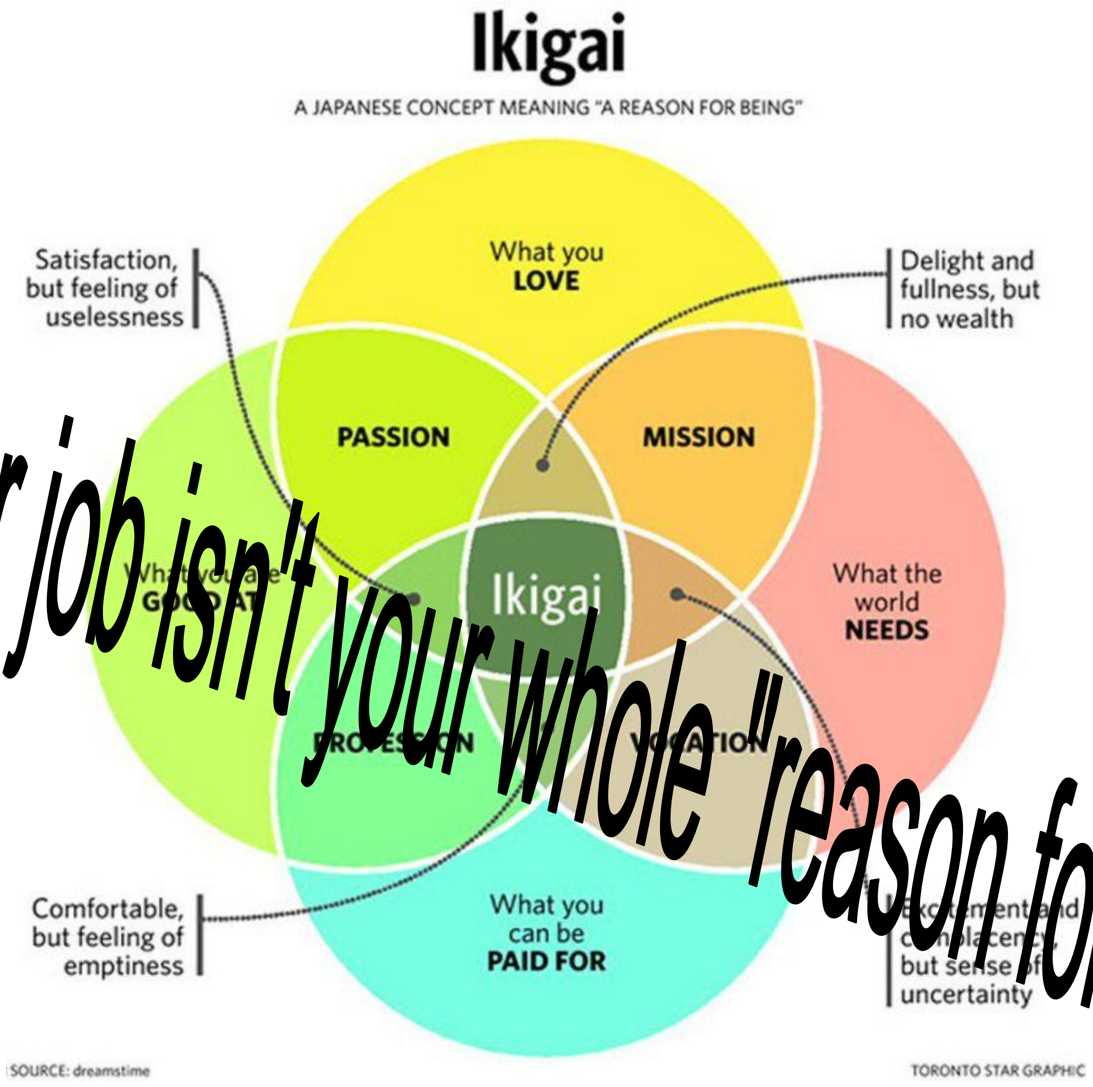
Most importantly

- If you have space, ask yourself “why”, early and often, again and again
  - This requires space to think, if you can give yourself time & space for reflection
- Most of y’all (in CS) are gunning for Big Four jobs for prestige
  - Prestige is fine, but there’s a lot of legitimacy and **not a lot of you**

Most importantly

- **Try to find justifications that feel like yours!**
    - **Reasons that are situated in your experience**
    - **Reasons that are situated in what excites you**
    - **Reasons that are situated in what you're good at**
    - **Reasons that are situated in what the world needs!**
- There isn't anything bad about working at Amazon, just know why you're doing it, and (ideally) be able to name that


Ikigai
A JAPANESE CONCEPT MEANING "A REASON FOR BEING"
What you LOVE
What you are GOOD AT
What the world NEEDS
What you can be PAID FOR
PASSION
MISSION
PROFESSION
VOCATION
Ikigai
Satisfaction, but feeling of uselessness
Delight and fullness, but no wealth
Comfortable, but feeling of emptiness
Excitement and complacency, but sense of uncertainty
SOURCE: dreamstime
TORONTO STAR GRAPHIC


Ikigai
A JAPANESE CONCEPT MEANING "A REASON FOR BEING"
What you LOVE
What you are GOOD AT
What the world NEEDS
What you can be PAID FOR
PASSION
MISSION
PROFESSION
VOCATION
Ikigai
Satisfaction, but feeling of uselessness
Delight and fullness, but no wealth
Comfortable, but feeling of emptiness
Excitement and complacency, but sense of uncertainty
SOURCE: dreamstime
TORONTO STAR GRAPHIC

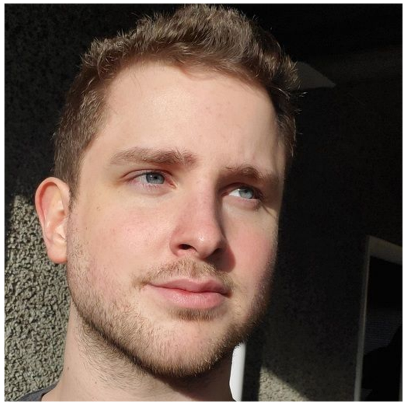
Also, your job isn't your whole "reason for being"

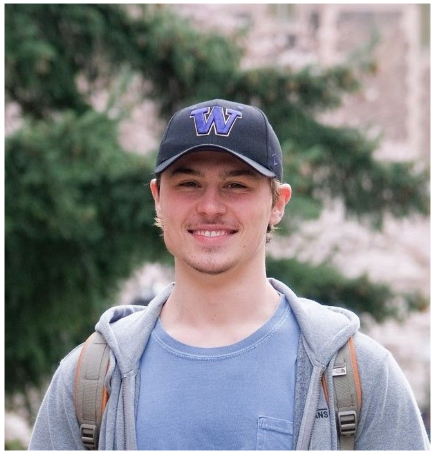
Most importantly, find people to talk to about all this!

That can be me (marakr@cs), if you'd like.

Thanks for a lovely quarter!

- Huge thanks to your awesome TAs!





- Let us know how we can be helpful!
  - Send emails, keep asking questions!

Thanks friends!

Anything you'd like to ask me?