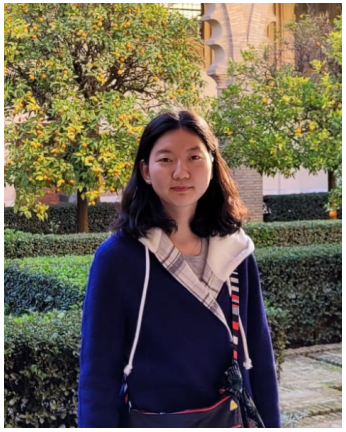
COS/ECE 375:
Computer Architecture & Organization

Welcome!

Introductions…

- Prof. Margaret Martonosi
  - W. M. Addy ‘82 University Professor. Dept. of Computer Science
  - At Princeton since 1994.
  - Active in Computer Architecture research
  - Did foundational work in power-efficient computer architecture
  - Currently laying the groundwork for QC architectures
  - Will try to weave in these first-hand experiences. Feel free to ask more during class or office hours!
- Co-Teaching with Prof. David August
  - He’ll introduce himself in Lecture 4

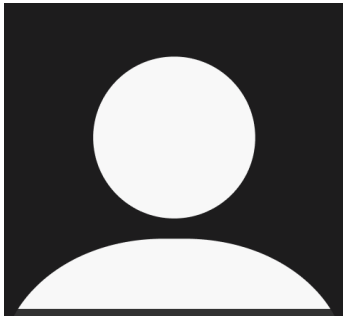
375 Graduate TAs



Yebin Chon

Rohan Prabhakar

Polly Ren



AZ Zhao

Yiyou Chen

Phyllis Wang

Maki Yu

Undergraduate Course Assistants (UCAs) coming soon!

Course Goals

By the end, we want you to feel comfortable thinking through questions like these:

- What happens under the covers when you double-click or press return to run a program?
- How do application trends drive hardware designs? What hardware features have emerged to support AI etc?
- How do hardware technology trends affect applications?
- What are the most widely-used systems and hardware techniques for improving computer performance?
- We hear a lot about computer performance, but how about how design choices shape power dissipation, security, etc?

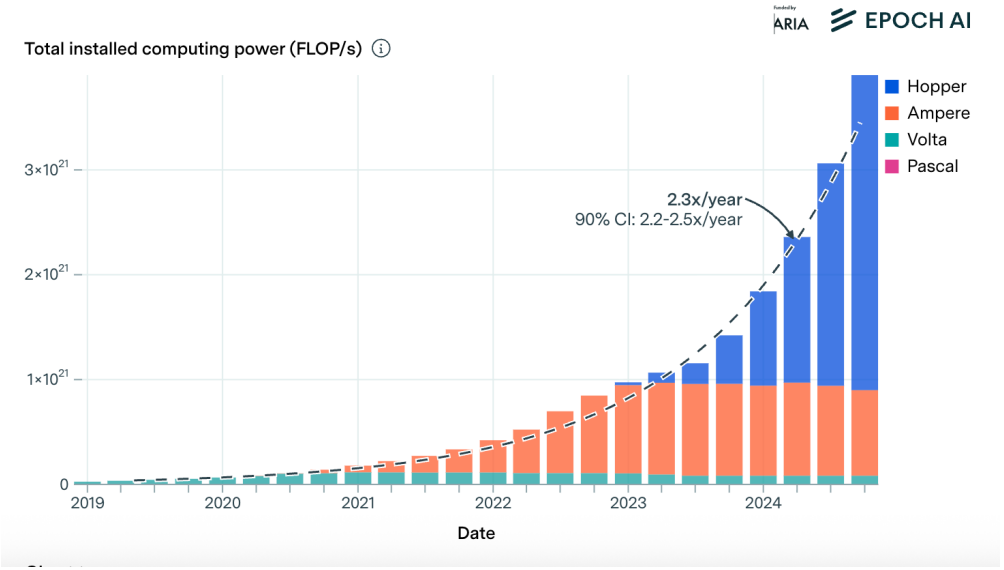
Application Trends

- Applications trends shaping hardware designs +
- Hardware capabilities shaping what applications can do

https://epoch.ai/data-insights/nvidia-chip-production

The stock of computing power from NVIDIA chips is doubling every 10 months

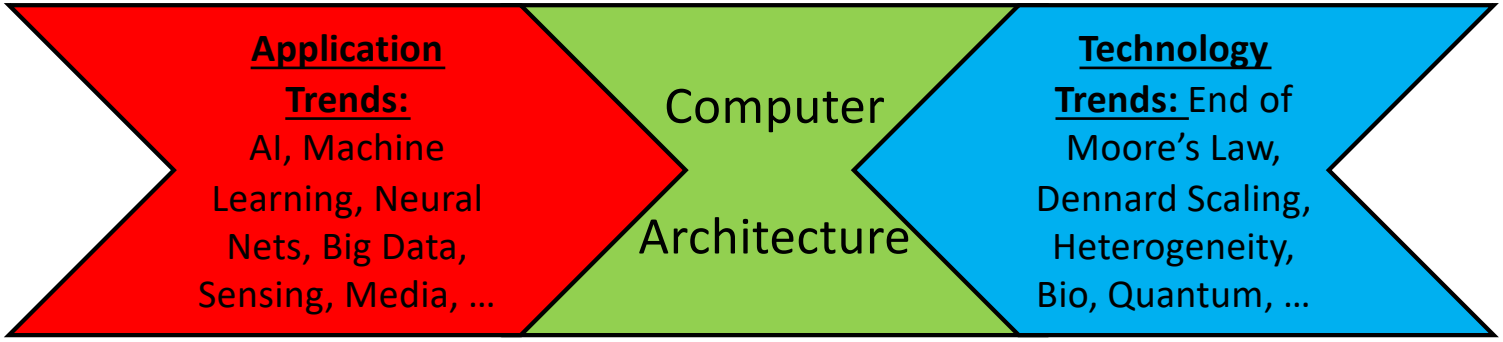
Total available computing power from NVIDIA chips has grown by approximately 2.3x per year since 2019, enabling the training of ever-larger models. The Hopper generation of NVIDIA AI chips currently accounts for 77% of the total computing power across all of their AI hardware. At this pace of growth, older generations tend to contribute less than half of cumulative compute around 4 years after their introduction.



What is Computer Architecture

- ***Instruction Set Architecture*** ~"contract" between hardware and software, kind of like an API between software and hardware.
  - Everything software needs to know in order to build a correct compiler/OS.
  - Everything hardware needs to know in order to build a correct implementation.
- ***Implementation*** ~ physical hardware realization of an architecture.
  - Many potential implementations per arch: e.g. x86
- This class will cover both architectures and their implementations. Above us is software. Below us, are the VLSI circuits and devices from which our organizational features are built.
- This class: Start with uniprocessor CPUs, but move outward into multiprocessors, GPUs, and memory systems as well.

Computer Architecture:
Mediator between Technology & Applications



Moore's Law

1965

Cramming more components onto integrated circuits

With unit cost falling as the number of components per circuit rises, by 1975 economics may dictate squeezing as many as 65,000 components on a single silicon chip

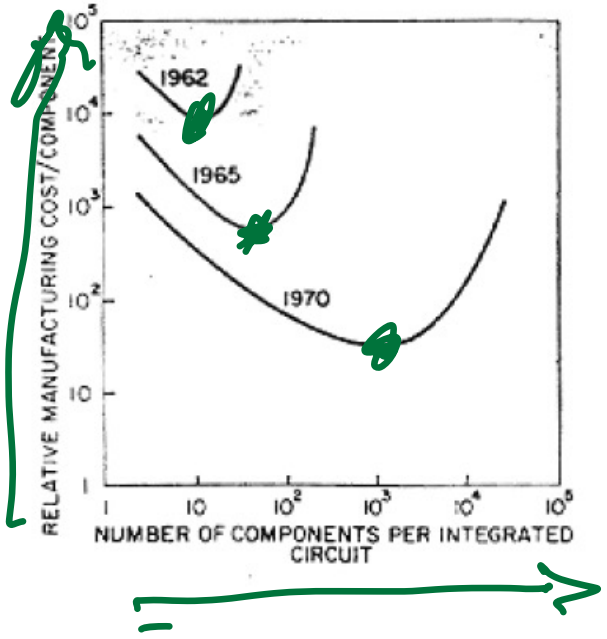
By Gordon E. Moore

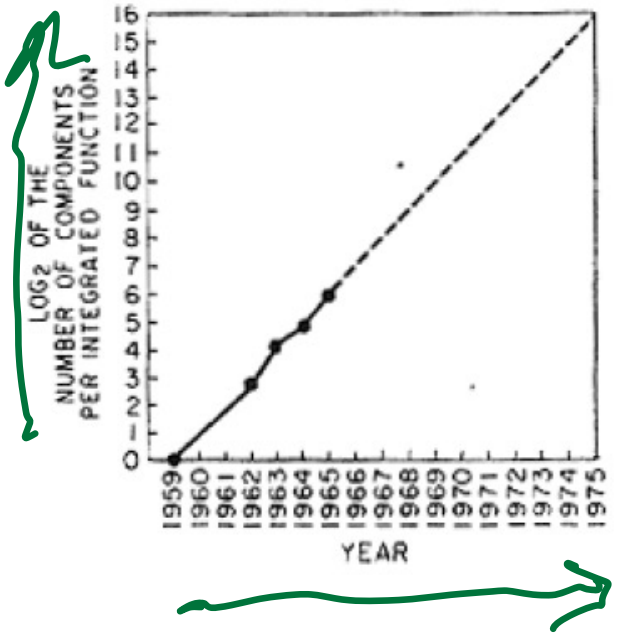
Director, Research and Development Laboratories, Fairchild Semiconductor division of Fairchild Camera and Instrument Corp.

[Photo: Gordon Moore & Robert Noyce. Wikimedia Commons]

Moore's Law

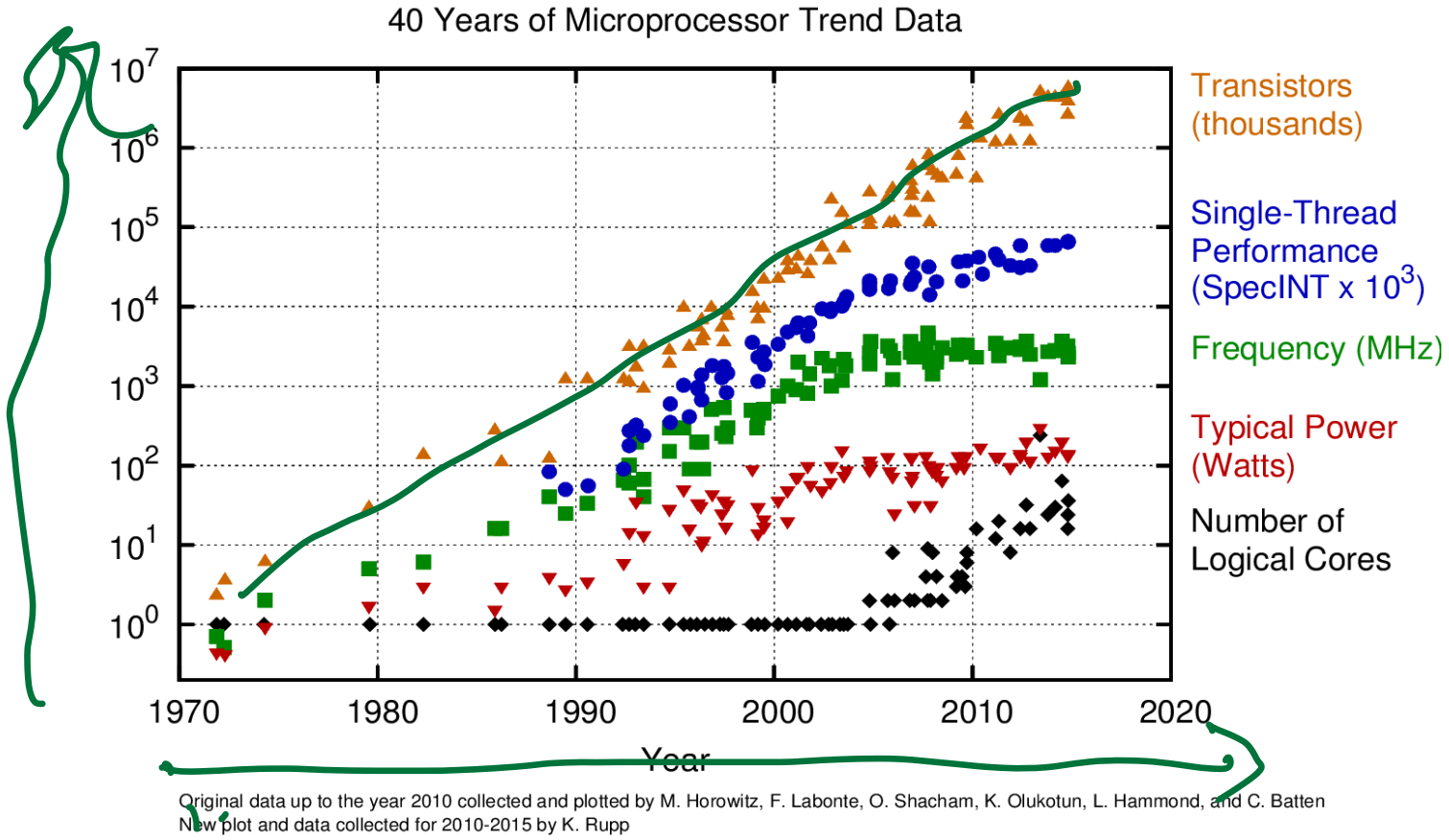
1965

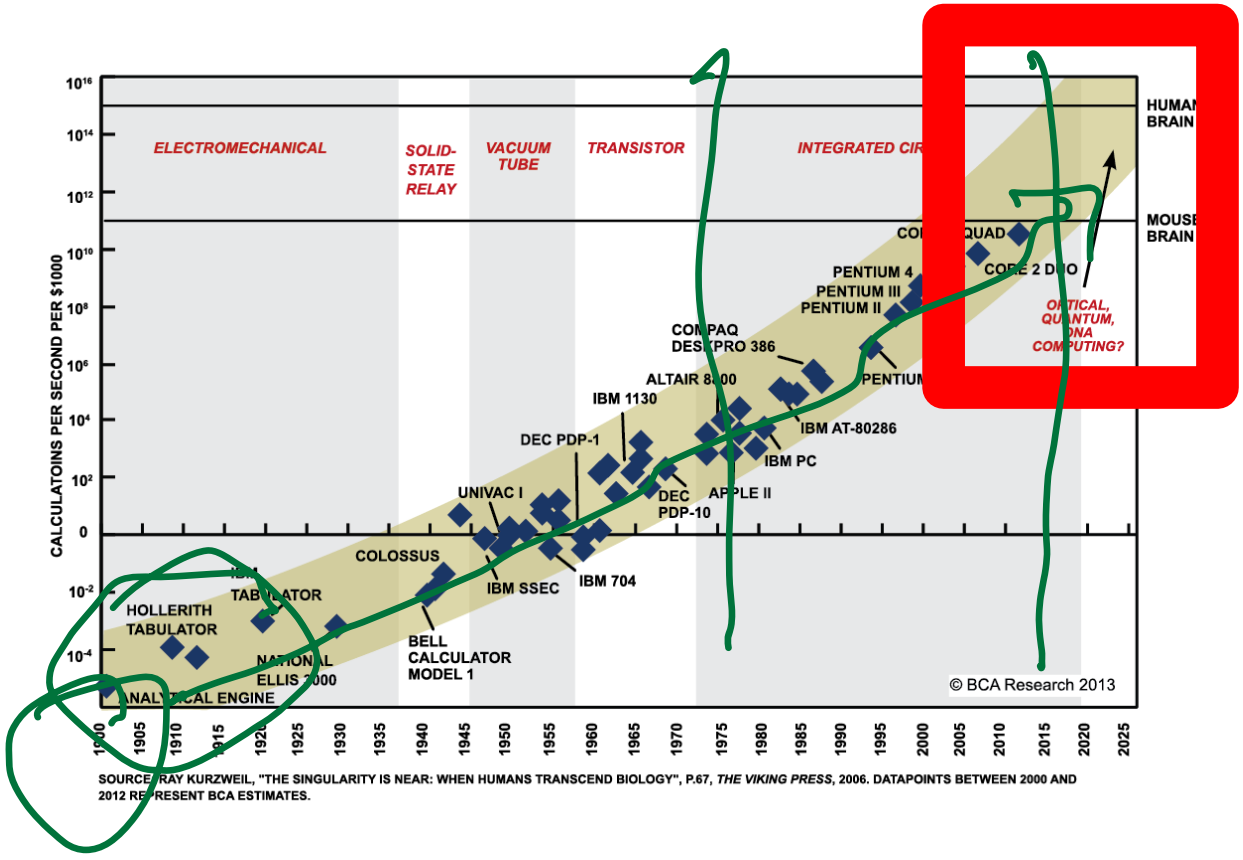




[Photo: Gordon Moore & Robert Noyce. Wikimedia Commons]

Decades of Moore's Law scaling





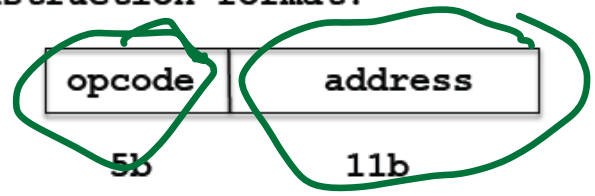
1900

Performance Scaling over the Decades

Disruptive Moment:
What is next?
Quantum, Optical, Biological, …

EDSAC Architecture Specification

16-bit wordsize
2 Kword Memory "M"
1 Accumulator "Acc", a 16-bit register
Instruction format:



Instruction Set

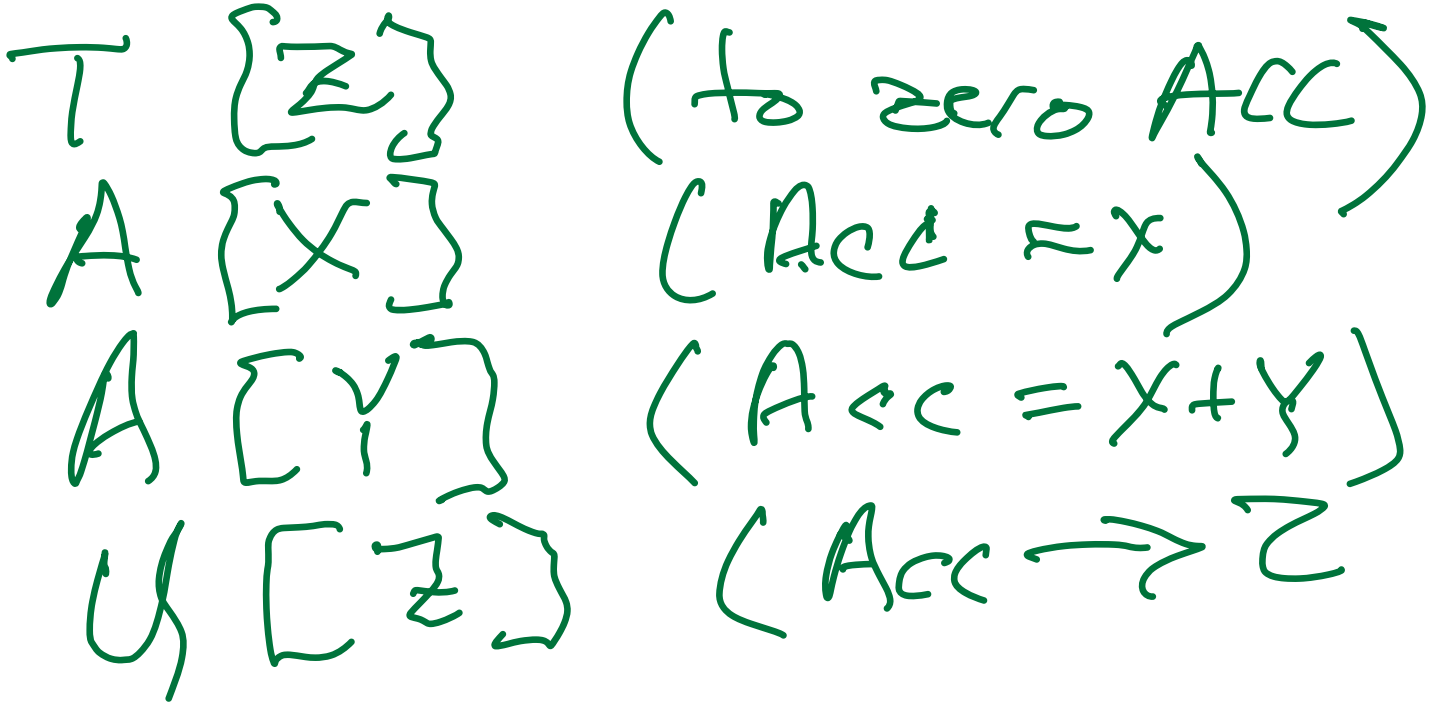
Name	Mnemonic	Function
Add	A n	Acc = Acc + M[n]
Subtract	S n	Acc = Acc - M[n]
Transfer	T n	M[n] = Acc; Acc = 0
Update	U n	M[n] = Acc
Goto	G n	Go to n if Acc < 0

(and a few more)

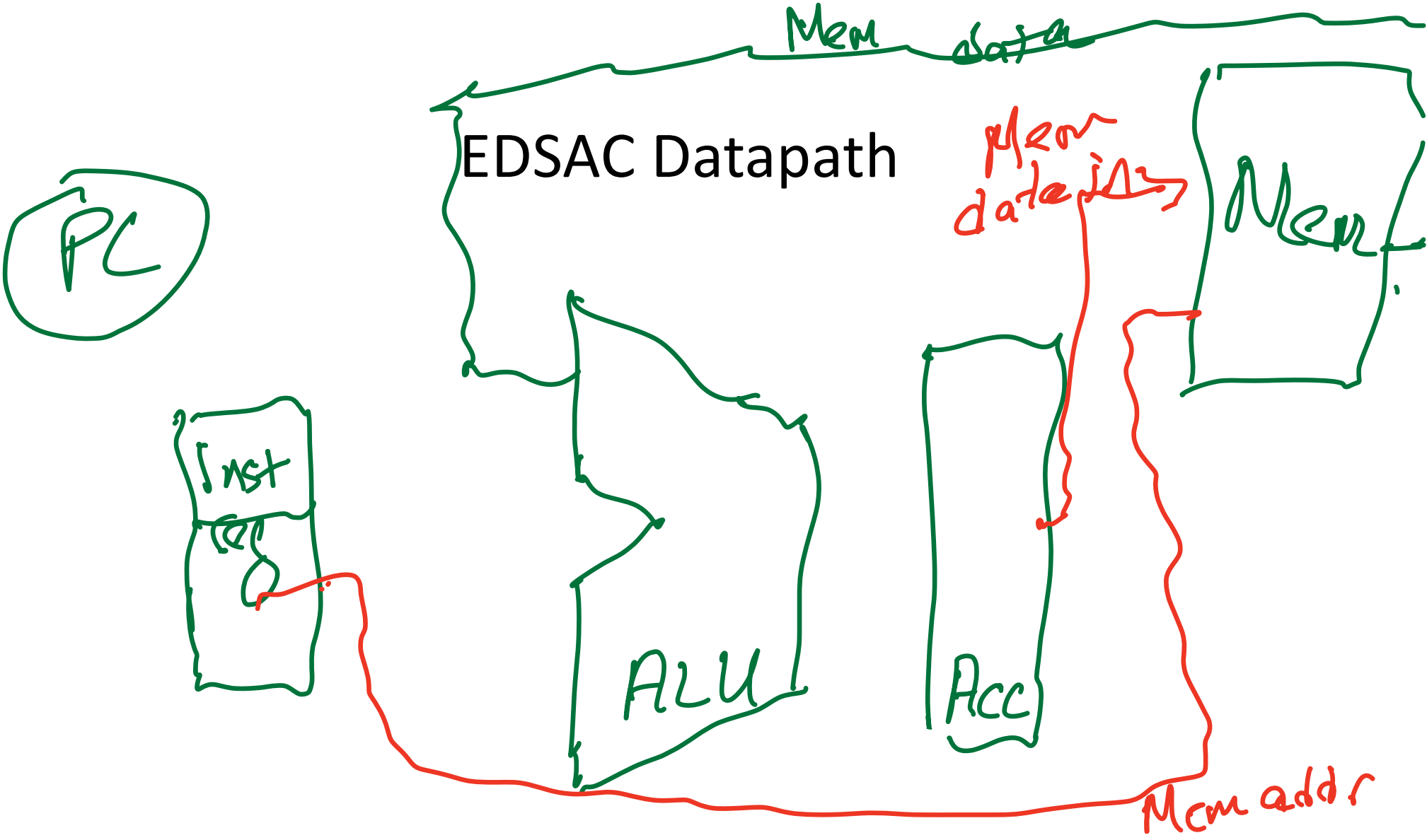
$Z = X + Y$ (where Z, X, Y are in Memory)

Simple Program: How would you…

Compute Z = X + Y, where Z, X, Y are in Memory

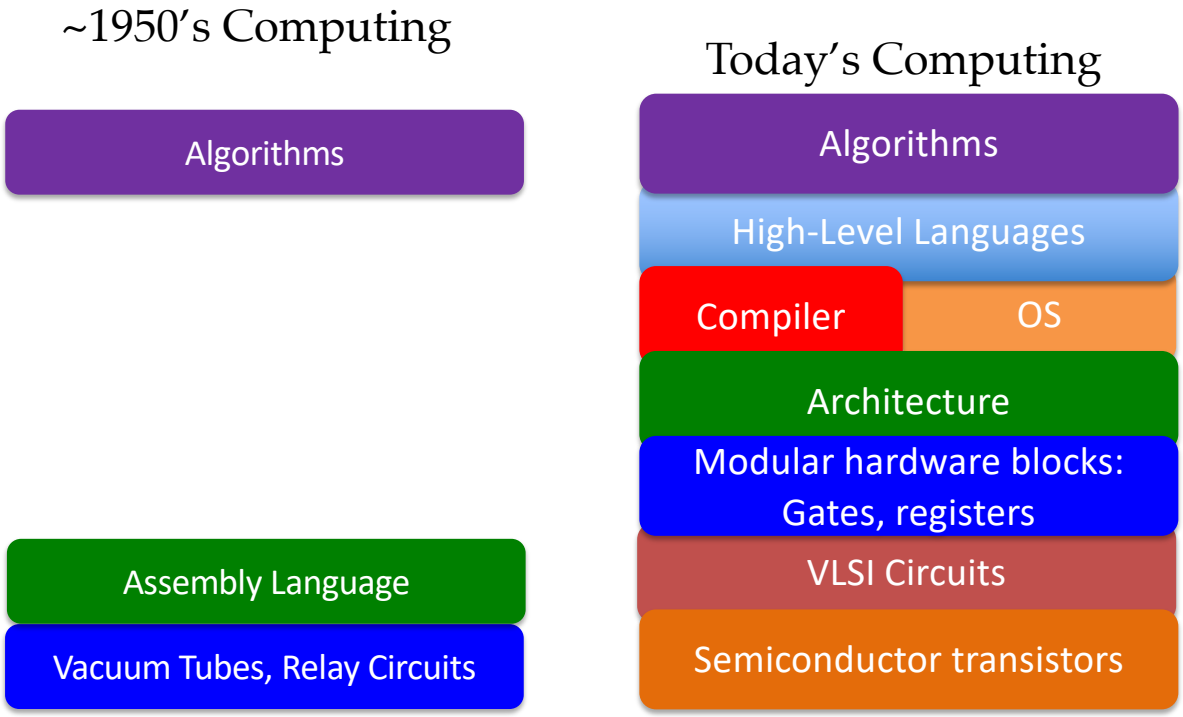


EDSAC Datapath

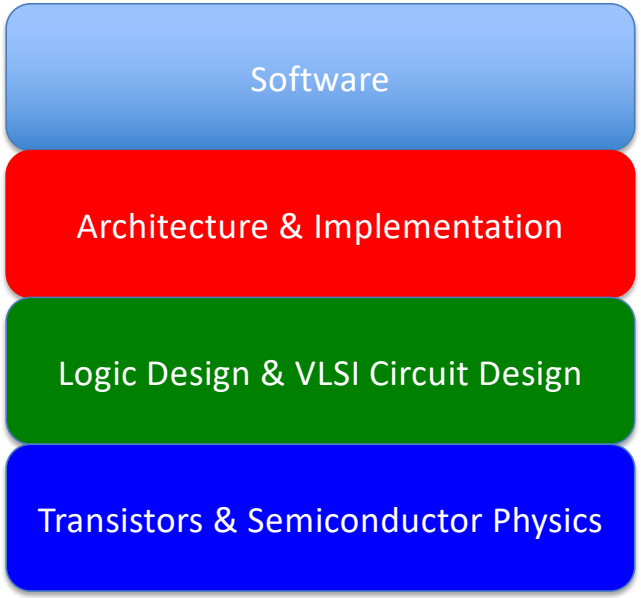


Glossary

375: Showing how the pieces connect and contribute!



How the classes fit together…



COS 217, 316, 320, 333.

This class!

Not this class!
(ECE206, ECE462)

Really Really
not this class!
(ECE341)

Logistics and Syllabus

- All on Canvas…
- First Assignment is due Fri Sept 5 at 6pm, but it is short, I promise!
- 8 Assignments total – roughly weekly – except no assignments on week of midterm or project deadline etc.
- 2 project modules. First due Oct 3 – more on that Tues Sept 9.

Architecture vs. Implementation

1964:
While Moore's Law was being sketched, "Computer Architecture" was coined…

G. M. Amdahl
G. A. Blaauw
F. P. Brooks, Jr.,

Architecture of the IBM System/360

Abstract: The architecture* of the newly announced IBM System/360 features four innovations:

1. An approach to storage which permits and exploits very large capacities, hierarchies of speeds, read-only storage for microprogram control, flexible storage protection, and simple program relocation.

2. An input/output system offering new degrees of concurrent operation, compatible channel operation, data rates approaching 5,000,000 characters/second, integrated design of hardware and software, a new low-cost, multiple-channel package sharing main-frame hardware, new provisions for device status information, and a standard channel interface between central processing unit and input/output devices.

3. A truly general-purpose machine organization offering new supervisory facilities, powerful logical pro-

a line of six models having a per-

tionale for the main features of the
ompatibility among central process-
scientific, real-time, and logical in-
he Appendices.

> *The term *architecture* is used here to describe the attributes of a system as seen by the programmer, i.e., the conceptual structure and functional behavior, as distinct from the organization of the data flow and controls, the logical design, and the physical implementation.

Introduction

The design philosophies of the new general-purpose machine organization for the IBM System/360 are discussed in this paper.† In addition to showing the architecture* of the new family of data processing systems, we point out the various engineering problems encountered in attaining to make the system design compatible, at the program bit level, for large and small models. The compatibility was planned not only to models of any size but also to their applications—scientific, commercial, real-time, and

The section that follows describes the objectives of the new system design, i.e., that it serve as a base for new technologies and applications, that it be general-purpose, efficient, and strictly program compatible in all models. The remainder of the paper is devoted to the design problems faced, the alternatives considered, and the decisions made for data format, data and instruction codes, storage assignments, and input/output controls.

Design objectives

The new architecture builds upon but differs from the designs that have gradually evolved since 1950. The evolution of the computer had included, besides major technological improvements, several important systems concepts and developments:

87

IBM JOURNAL • APRIL 1964

term *architecture* is used here to describe the attributes of a s seen by the programmer, i.e., the conceptual structure and l behavior, as distinct from the organization of the data flow rols, the logical design, and the physical implementation. itional details concerning the architecture, engineering design, sing, and application of the IBM System/360 will appear in a articles in the *IBM Systems Journal*.

= The value of HW/SW abstraction…

Architecture = Abstraction

- Separate implementation from specification
  - INTERFACE: specify the provided services.
  - IMPLEMENTATION: provide code or HW for operations.
  - CLIENT: code or HW that uses services.
- Examples?

Why take 375?

- Abstraction is great but
  - Behavior of software depends on adhering to hardware realities

Example #1: Precision

```
def sum(a, b):
    return (a + b)

a = float(input('Enter 1st number: '))
b = float(input('Enter 2nd number: '))

print(f'Sum of {a} and {b} is {sum(a, b)}')
```

For a entered as 0.1 and b entered as 0.2,
Sum: $0.30000000000000004 ?

- Why isn't it 0.3? And how does this finite precision affect the answers your large-scale code gives?

Example #2: Memory Misuse

```
int array[12];
int val = 0;

array[12] = 1000;
```

Either Exception or val == 1000

When writing beyond the end of an array, you get an ArrayOutOfBoundsException OR you change a different variable's value (depending on the programming language).

Example #3: Out of Memory

```
int fib(int n)
{
   return fib(n-1)+fib(n-2);
}
```

When writing a deeply recursive function, you can get a StackOverflowException or your program quits unexpectedly.

Example #4: I/O

```
public static void main(String [] args)
{
  try{
    File file = new File("output.txt");
    PrintWriter pw = new PrintWriter(file);
    pw.println("Hello World!");
    pw.close();
  }
  catch(Exception e){
    System.out.println(e);
   }
}
```

- When you can't save a file because the disk is full.

Why take 375?

- Abstraction is great but
  - Behavior of software depends on it adhering to hardware realities
  - Performance depends on software exploiting what the hardware is fast at and avoiding what it's slow at

Performance Problem #1

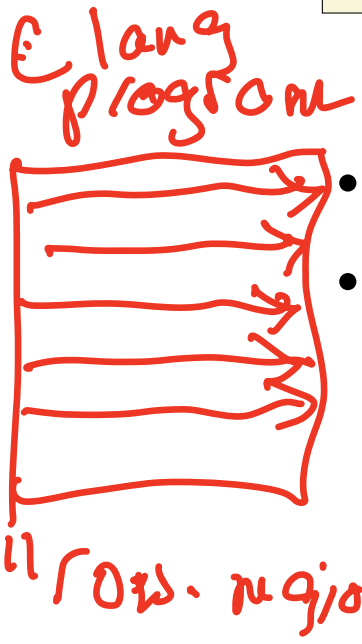
```
void copyij(int src[2048][2048],
            int dst[2048][2048])
{
  int i,j;
  for (i = 0; i < 2048; i++)
    for (j = 0; j < 2048; j++)
      dst[i][j] = src[i][j];
}
```

48ms

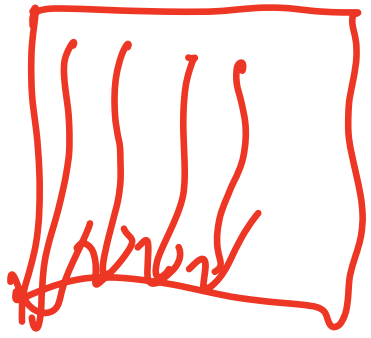
```
void copyji(int src[2048][2048],
            int dst[2048][2048])
{
  int i,j;
  for (j = 0; j < 2048; j++)
    for (i = 0; i < 2048; i++)
      dst[i][j] = src[i][j];
}
```

167ms

2.9 GHz Intel Core i5



- Hierarchical memory organization
- Performance depends on access patterns
  - Including how to step through a multi-dimensional array (row or column?)



Why take 375?

- Abstraction is great but
  - Behavior of software depends on it adhering to hardware realities
  - Performance depends on software exploiting what the hardware is fast at and avoiding what hardware is slow at
  - Hardware constructs may introduce security flaws

Computer Architecture & Security

Well-known cache side-channel attack { Flush+Reload

+

Widely-used hardware feature { Speculation

New exploit =

2 new attacks {

January 2018: Spectre & Meltdown

c|net REVIEWS NEWS VIDEO HOW TO SMART HOME CARS DEALS DOWNLOAD

SECURITY / Leer en español

Spectre and Meltdown: Details you need on those big chip flaws

Design flaws in processors from leading chipmakers could let attackers access sensitive information. How did this happen, and what's the fix?

BY LAURA HAUTALA / JANUARY 8, 2018 11:51 AM PST

Project Zero

News and updates from the Project Zero team at Google

Wednesday, January 3, 2018

Reading privileged memory with a side-channel

Posted by Jann Horn, Project Zero

WIRED Triple Meltdown: How So Many Researchers Found a 20-Year-Old Ch

SHARE

SHARE 4335

TWEET

ANDY GREENBERG SECURITY 01.07.18 02:23 PM

TRIPLE MELTDOWN: HOW SO MANY RESEARCHERS FOUND A 20-YEAR-OLD CHIP FLAW AT THE SAME TIME

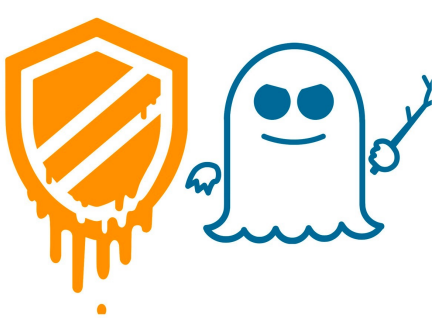
Computer Architecture & Security

Well-known cache side-channel attack { Flush+Reload

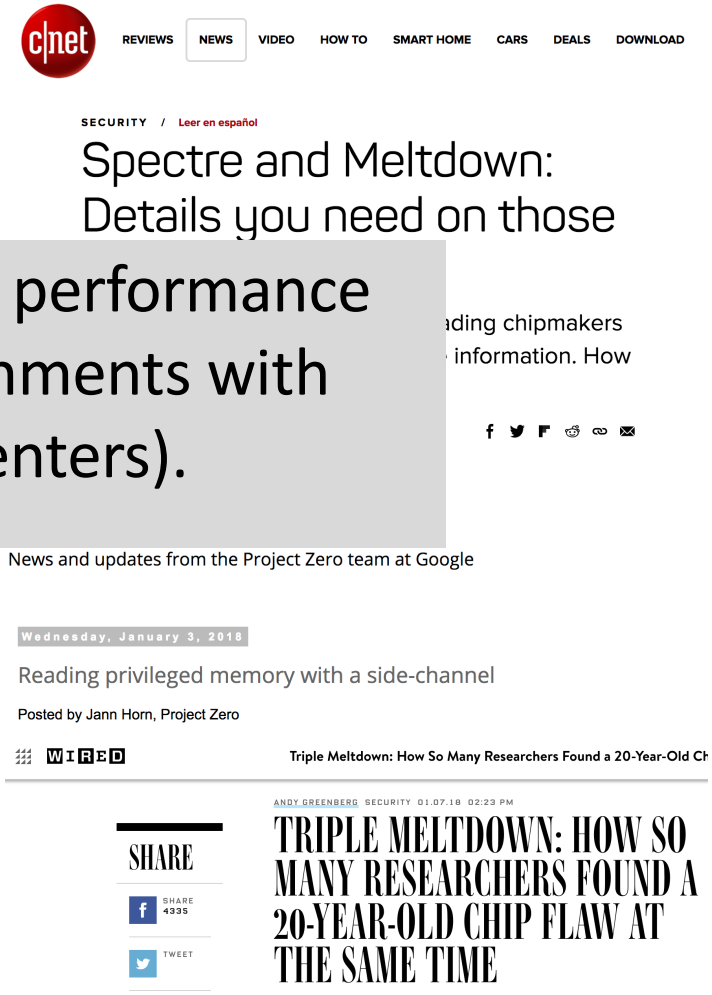
Hardware features introduced for performance become security flaws in environments with multiple users (esp. data centers).

New exploit =

2 new attacks {



January 2018: Spectre & Meltdown



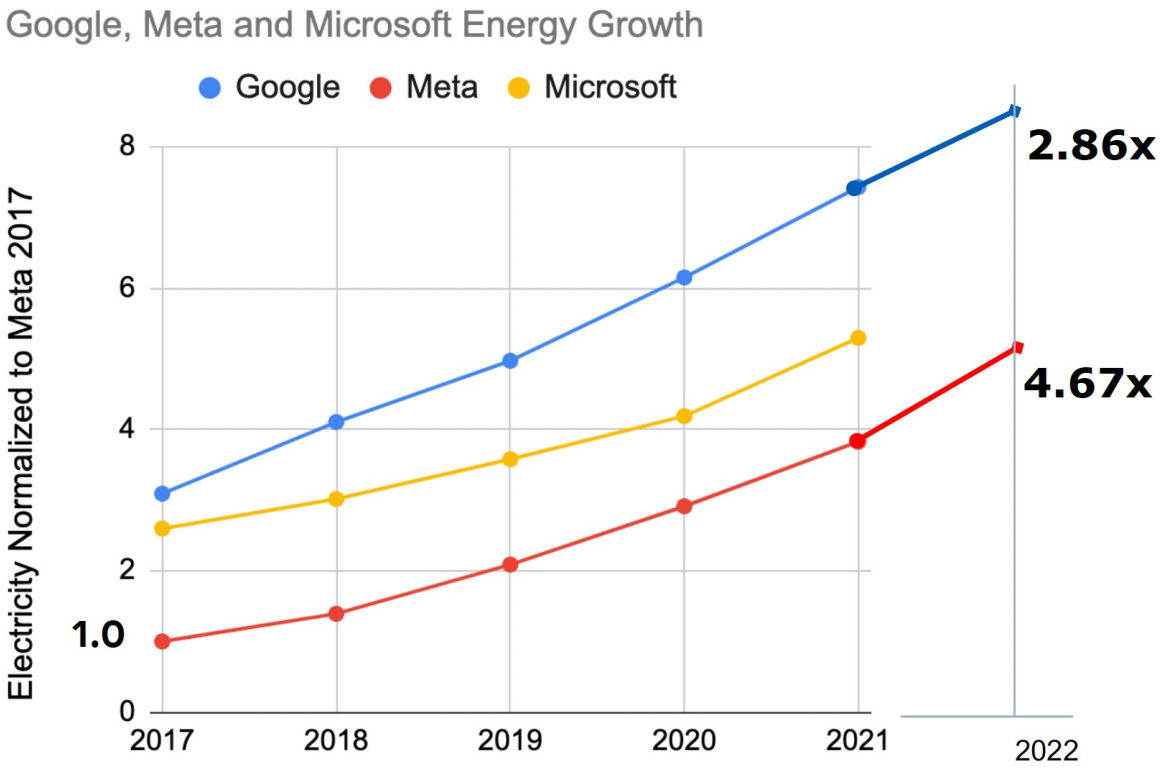
Why take 375?

- Abstraction is great but
  - Behavior of software depends on it adhering to hardware realities
  - Performance depends on software exploiting what the hardware is fast at and avoiding what hardware is slow at
  - Hardware constructs may introduce security flaws
  - Different implementation choices may dramatically affect power dissipation and energy efficiency

Warehouse-scale Computing

- Modern datacenters dissipate 50-100 Megawatts
  - Similar to a medium-sized city of ~60,000 households
- AI is driving higher usage of existing datacenters and increased pace of building new ones

Overall computing demand growth means aggregate electricity usage is growing fast



Credit: Carole-Jean Wu, META

Acknowledgements

- Fall, 2025 Course slides include materials from:
  - Margaret Martonosi and David August, Princeton
  - Doug Clark, Princeton (retired)
  - Kelly Shaw, Williams College
  - Carole-Jean Wu, META

Thank you!

Please fill out ICQ Form here!

https://tinyurl.com/Fa25ICQ.