

# CSE 141: Introduction to Computer Architecture

*We will start ~15:40 today, but promptly at 15:30 in the future*

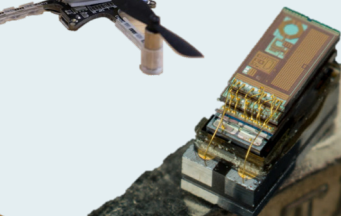
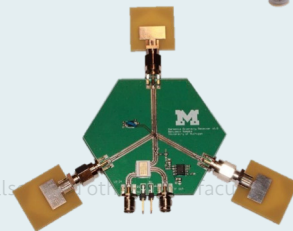
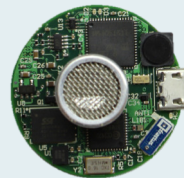
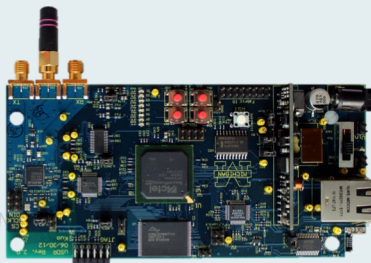
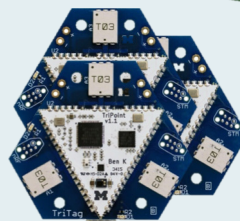
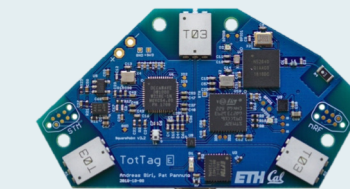
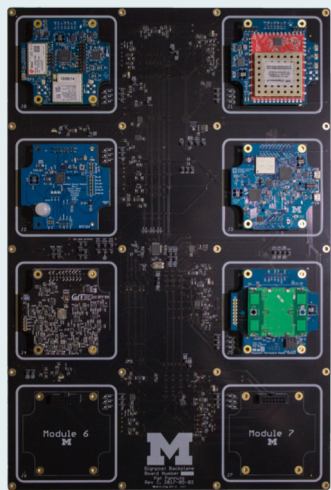
Pat Pannuto, UC San Diego

[ppannuto@ucsd.edu](mailto:ppannuto@ucsd.edu)

OInK  
Tock

Human  
Perception

Camera  
Perception



# What is Computer Architecture and where does it fit in Computer (Science) Engineering?

- One view: what is an Architect and how do they fit in the creation of buildings?

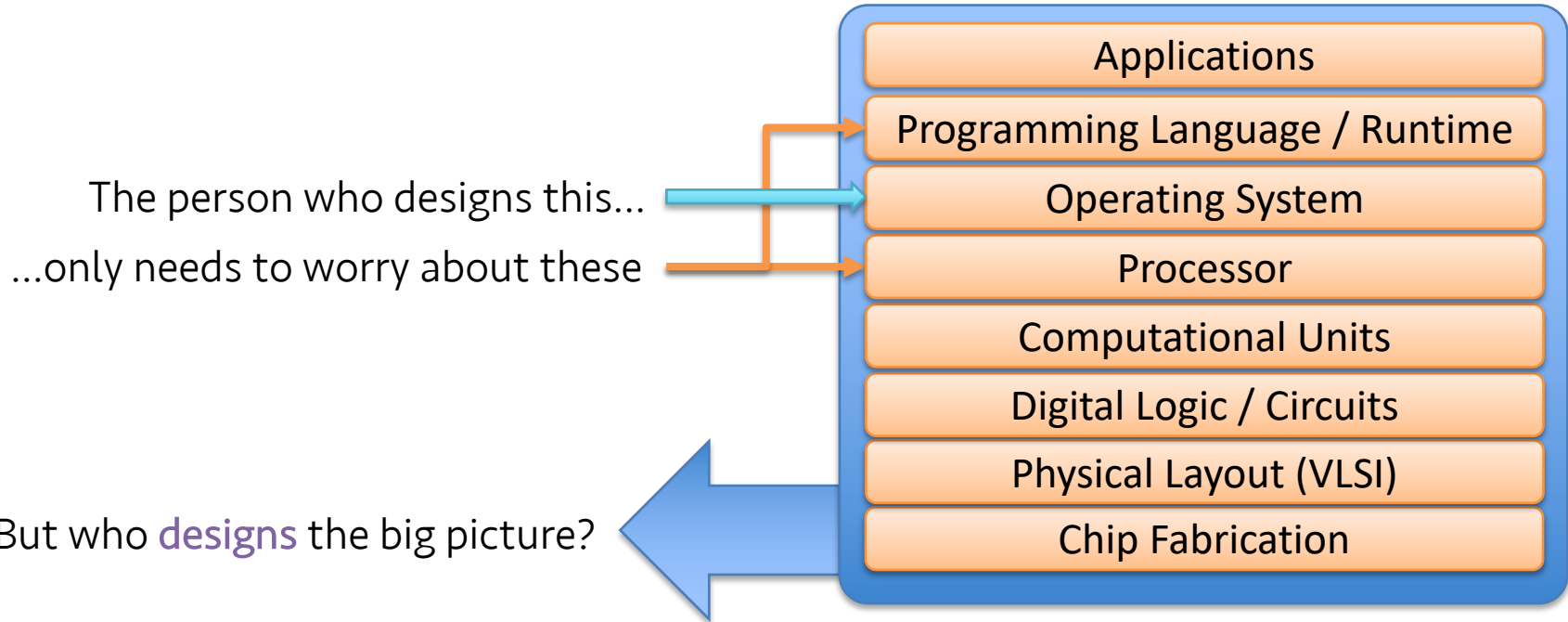


Zaha Hadid

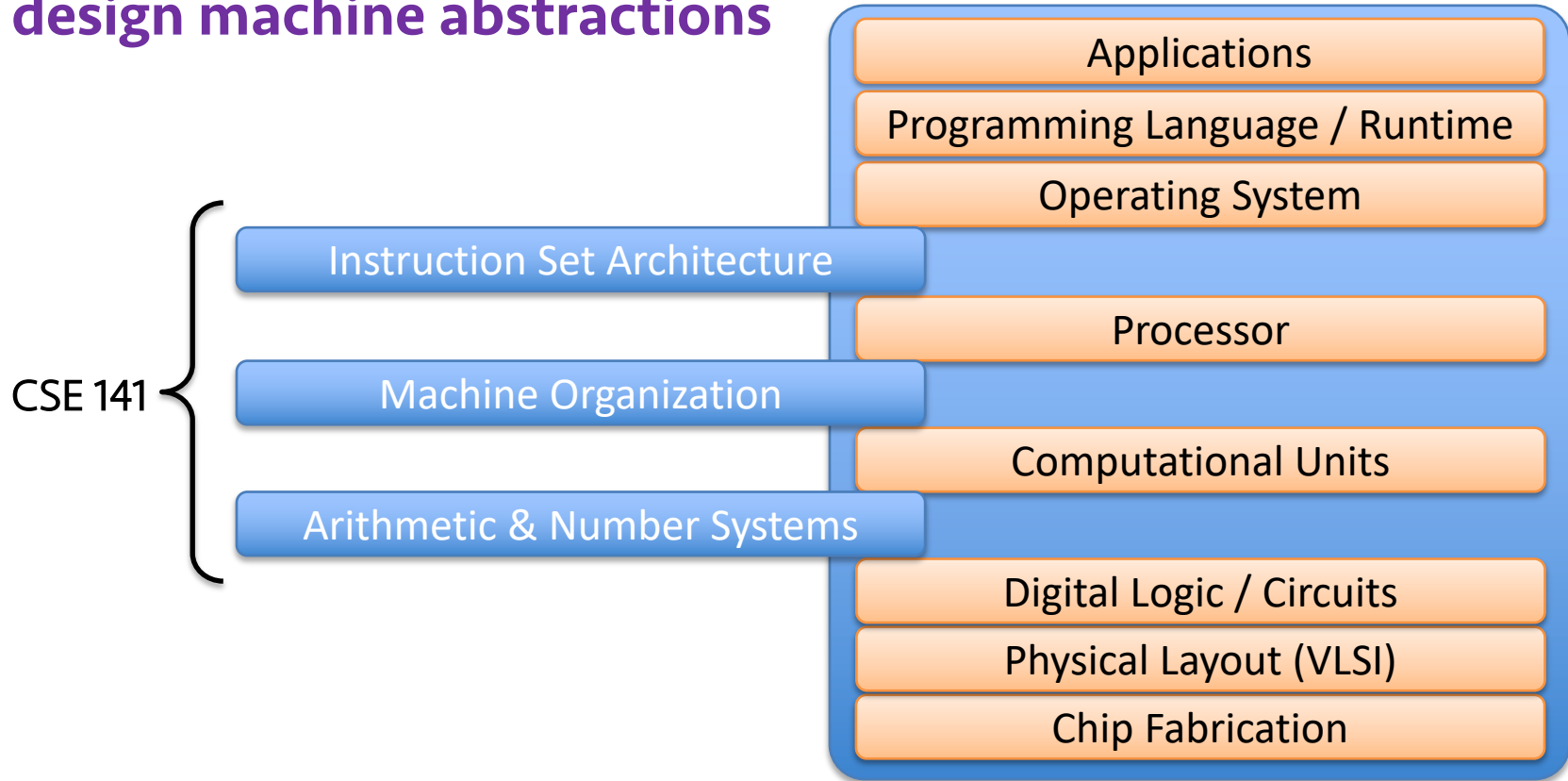


Port Authority Building in Antwerp, **designed** by Zaha

# Computer science is all about abstractions

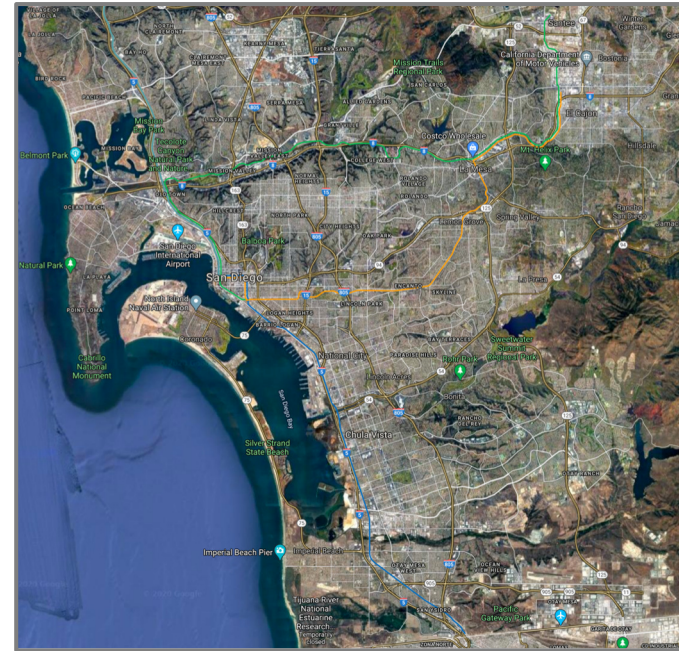


# Computer architects look at the system as a whole and design machine abstractions



# Good abstractions make it easier to focus on reasoning about one part of a large, complex system

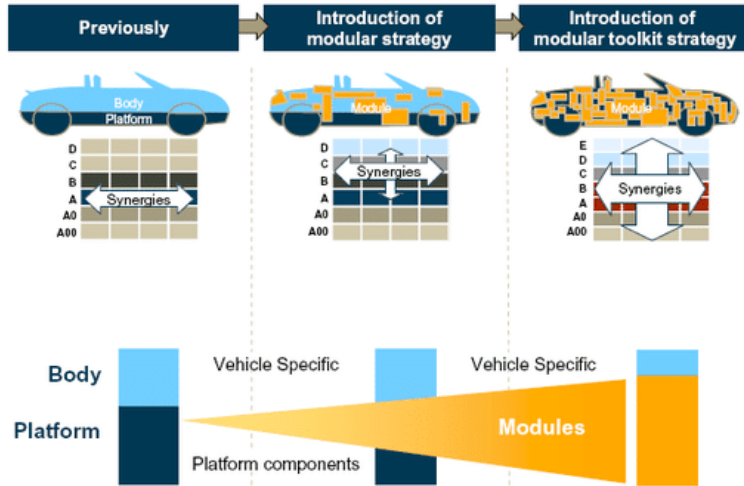
- Which of these maps is easier to use to plan a trolley trip?



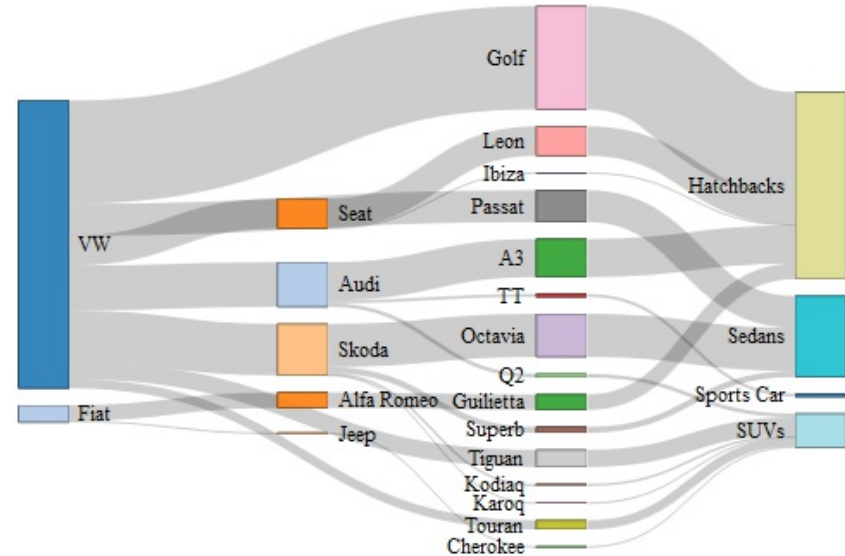
# Good abstractions make it easier to focus on reasoning about one part of a large, complex system

- Modularization is fundamental to design in many domains

## Volkswagen Group's Modular Toolkit Strategy



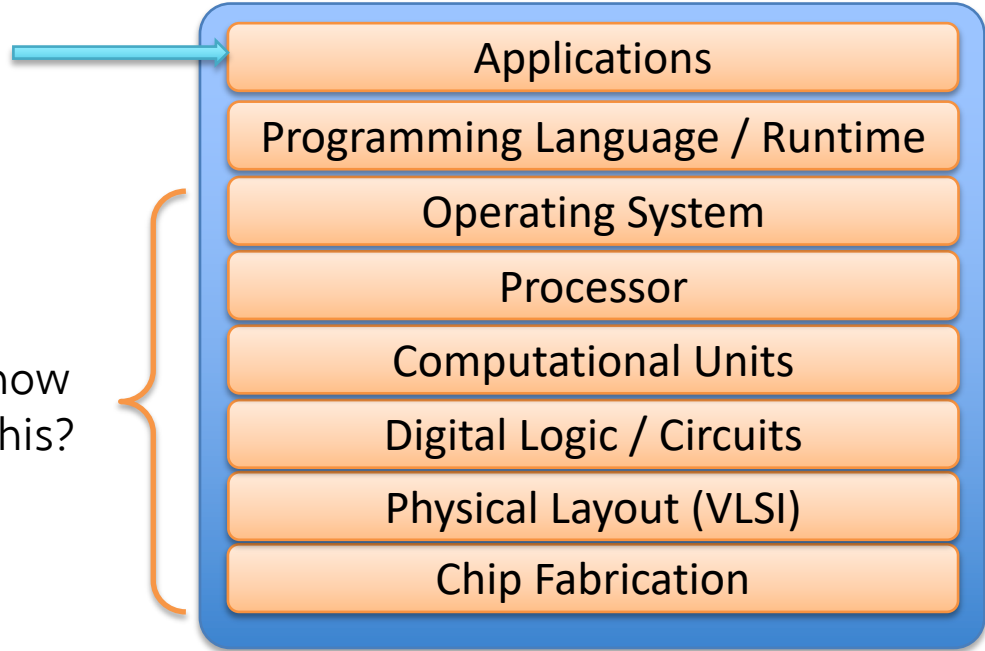
*Modular Car Body Design and Optimization by an Implicit Parameterization Technique via SFE CONCEPT  
Fabien Duddeck, Hans Zimmer*



[https://www.reddit.com/r/dataisbeautiful/comments/8m15g9/automobile\\_platform\\_sharing\\_work\\_in\\_progress/](https://www.reddit.com/r/dataisbeautiful/comments/8m15g9/automobile_platform_sharing_work_in_progress/)

# But what if I'm not going to become a computer architect?

If I only want to build these...



...why do I need to know about any of this?

# The real world is full of leaky abstractions

- Goal: Sum up all the entries of a two dimensional array
- Which of these implementations is faster?

```
int twoDarray[256][256];
int sum = 0;

for (int i=0; i<256; i++) {
    for (int j=0; j<256; j++) {
        sum += twoDarray[i][j];
    }
}
```

```
int twoDarray[256][256];
int sum = 0;

for (int i=0; i<256; i++) {
    for (int j=0; j<256; j++) {
        sum += twoDarray[j][i];
    }
}
```

*Answer: "It depends"*



# Architects look across systems to find and improve inefficiencies – always valuable, sometimes critical...

- What happens when workload changes overnight, and there's no way to buy your way out of the problem?
  - Architects help to fix it

## Microsoft Admits Supply Chain Issues Hit Cloud Server Supply

ED TARGETT EDITOR  
30TH APRIL 2020



## Data Center Hardware Shortages in the Age of COVID-19

### Coronavirus: Supply issues hit PC market

Despite strong demand, the pandemic caused problems in sourcing kit and forced the PC market to reverse three quarters of growth



By **Simon Quicke**, Microscope Editor

Published: 14 Apr 2020 10:29

Coronavirus has had an impact on PC supply chains despite the hard work of the industry to counter the negative impacts.

work, and those changes put new demands on IT that have been hard to

more people work from home and remain connected with friends and family

Late **increase** in network capacity utilization.

VMware Google and Amazon, have had outages as they rushed to add more data

Gen

But there's a basic problem afflicting everyone in the technology world. As the world searches for a light at the end of the COVID-19 tunnel, leading tech companies are working to overcome supply chain shortages of critical components for servers, storage, and networking products.

# Course Administrivia

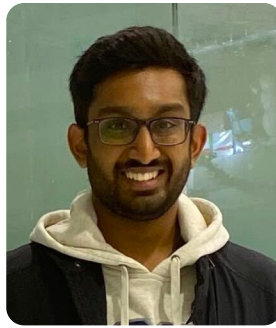


- Instructor
  - Pat Pannuto
- Support Infrastructure & Tools:
  - iClicker for participation
    - Canvas quizzes for limited make-ups
  - Piazza for Q&A
  - Gradescope for Homework
  - Exams are in-person with paper and pencil

# Course Staff

- Four amazing TAs:

- Adithya Anand



- Link Lin



- Wenshan Luo



- Rahul Polisetti

- Discussions

- Wed from 18:00 to 18:50

- More / different explanation of lecture concepts

- Interactive practice problems similar to homework/exam Q's



# Assessments & Workload

- Grading
  - 15%: Participation — **Synchronous Attendance is required**
    - Interaction with material each lecture
    - Playing for keeps starting week 2 (i.e. graded starting Tue, Apr 5)
    - 15 graded lectures → 1% / lecture
  - Homework: 30%
  - Midterm: 25%
  - Final Exam: 30%
    - (Inclusive final — biased to later material — no “midterm catchup”)
    - Final on last day of class (June 2) — Last new material May 26 — Plan for this now!

# Repeated, active engagement is key to effective learning

- Pre-class reading is your first exposure
  - 5 minutes before class is better than not at all, but 5+ hours before is much better
  - **Read actively**, try writing notes for yourself of what you understood from readings
- Lecture is not a passive activity
  - Ask (or write down) questions about what you do not understand!
  - **Use checkpoints (in-lecture questions) effectively**
- Discussions, office hours, and exercises are not passive activities
  - **Work through examples yourself** and ask the questions you have
- Homework is designed to help you solidify your understanding
- Study for tests “honestly to yourself” – **you** must engage with questions

# Class is not a competition

- My philosophy
  - I care whether you learn the material
  - The purpose of a grade is to assess how well you know the material in 141
  - The purpose of a grade is not to “rank” students
  - I am most successful if everyone in class **earns** an A
- My goal is not to curve
  - (But I reserve the right to)
  - The midterm and final may be “internally” curved

# Academic Integrity

- Cheating will be taken very seriously
- Examples
  - Not cheating:
    - Discussing homework in groups, with **your own writeup, in your own words, done on your own, later**
    - Looking at lectures, *different* practice problems & solutions, etc from “other 141’s”
  - Cheating:
    - Getting a walk-through from someone who has already done the homework
    - Looking at someone else’s completed work (even “just to check”)
    - Using solutions from the web, prior classes, or anywhere else
    - Receiving, providing, or soliciting assistance from another student during a test
- Consequences
  - Negative 100% on the assignment (including exams) where you are caught
  - Notified *after* the quarter is over by the Academic Integrity Office

Any questions on course logistics?

**NEXT IS JUST A LITTLE OF THE FIRST  
BIT OF THE COURSE MATERIAL**



# What is Computer Architecture?

Computer Architecture =

Instruction Set Architecture


+

Machine Organization

*How you talk to the machine*

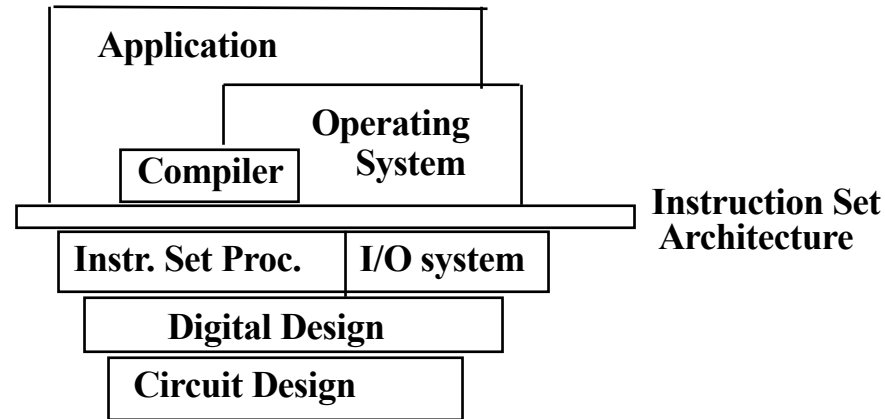


*What the machine  
hardware looks like*



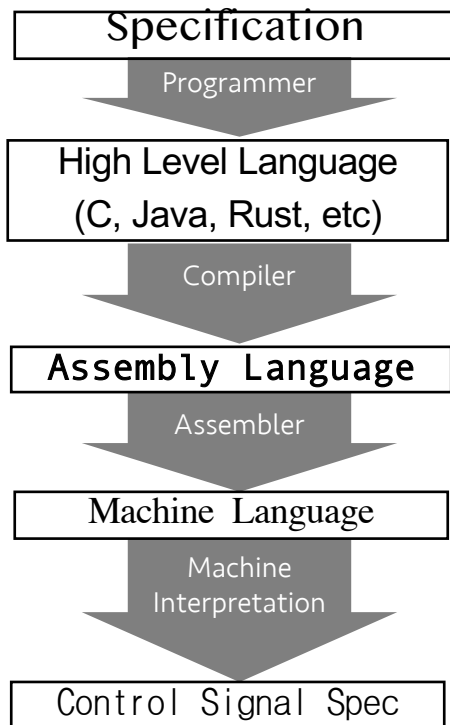
# An Instruction Set Architecture is an **abstraction** of a computational machine

- An ISA is “the agreed-upon interface between all the software that runs on the machine and the hardware that executes it.”



# Computers do not speak English

## *And they do not speak C or Java or Python or Haskell (or...) either*



“Swap two array elements.”

```
int temp = array[index];  
array[index] = array[index + 1];  
array[index + 1] = temp;
```

```
lw $15, 0($2)  
lw $16, 4($2)  
sw $16, 0($2)  
sw $15, 4($2)
```

```
10001100011000100000000000000000  
1000110011110010000000000000100  
10101100111100100000000000000000  
1010110001100010000000000000100
```

```
ALUOP[0:3] <= InstReg[9:11] & MASK
```

# Poll Q: If you had to put the ISA somewhere in this stack, would you say it sits between...

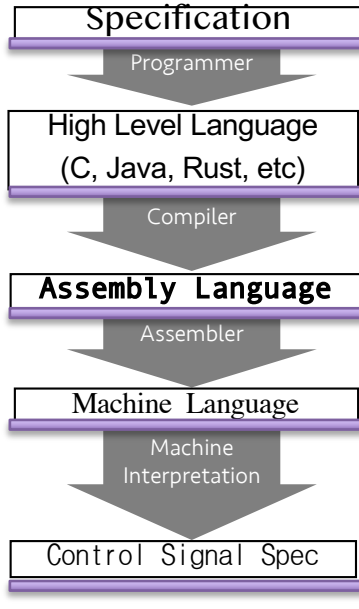
**A:** Specification and HLL

**B:** HLL and assembly

**C:** Assembly and machine language

**D:** Machine language & control signals

**E:** ISAs define control signals



“Swap two array elements.”

```
int temp = array[index];  
array[index] = array[index + 1];  
array[index + 1] = temp;
```

```
lw $15, 0($2)  
lw $16, 4($2)  
sw $16, 0($2)  
sw $15, 4($2)
```

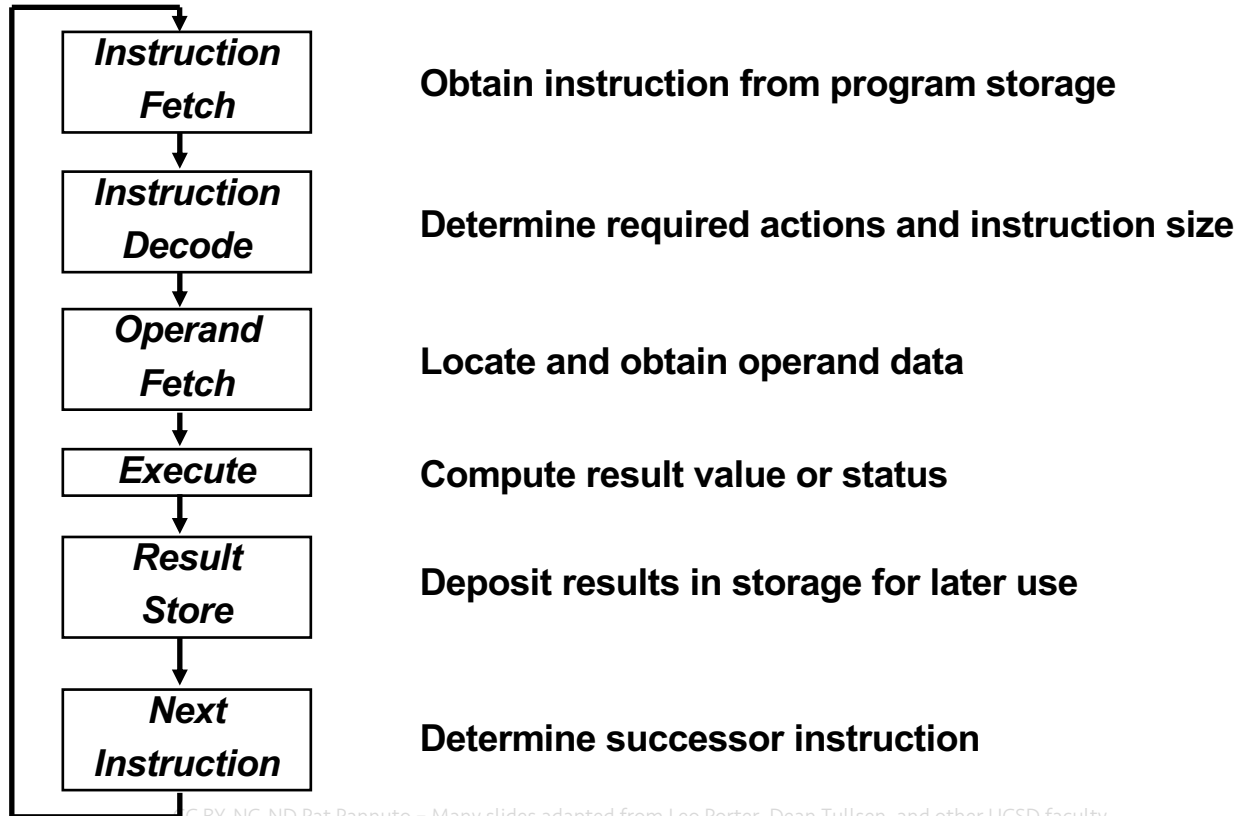
```
10001100011000100000000000000000  
10001100111100100000000000000100  
10101100111100100000000000000000  
10101100011000100000000000000100
```

```
ALUOP[0:3] <= InstReg[9:11] & MASK
```

# The Instruction Set Architecture

- that part of the architecture that is visible to the programmer
  - available instructions (“opcodes”)
  - number and types of registers
  - instruction formats
  - storage access, addressing modes
  - exceptional conditions

# The Instruction Execution Cycle



# A brief preview of some machine organization concepts:

## *Cycle*

- The smallest unit of time in a processor



**macOS Catalina**  
Version 10.15.6

iMac (Retina 5K, 27-inch, 2017)

Processor 4.2 GHz Quad-Core Intel Core i7

Memory 40 GB 2400 MHz DDR4

Startup Disk Macintosh HD

Graphics Radeon Pro 580 8 GB



**macOS Catalina**  
Version 10.15.7

MacBook Pro (13-inch, 2018, Four Thunderbolt 3 Ports)

Processor 2.7 GHz Quad-Core Intel Core i7

Memory 16 GB 2133 MHz LPDDR3

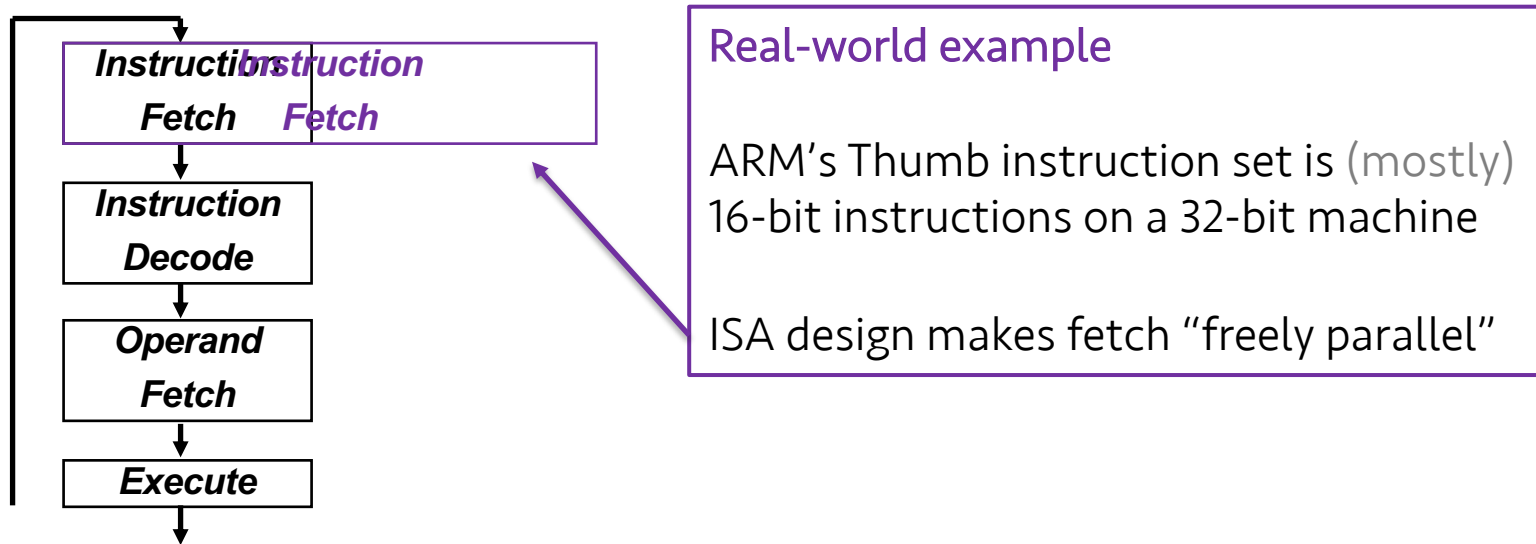
Startup Disk APPLE SSD AP1024M Media

Graphics Intel Iris Plus Graphics 655 1536 MB

# A brief preview of some machine organization concepts:

## *Parallelism*

- The ability to do more than one thing at once

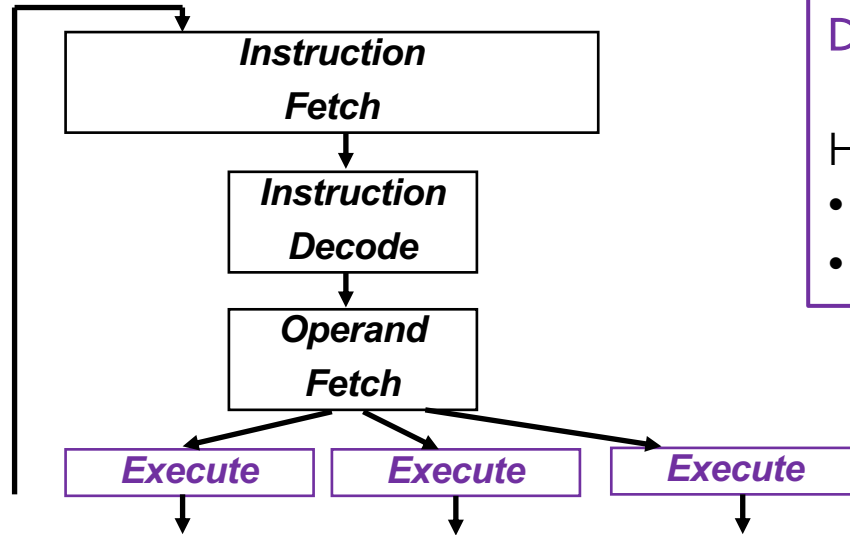




# A brief preview of some machine organization concepts:

## *Superscalar Processor*

- Can execute more than one instruction per cycle



Duplication is easy but expensive...

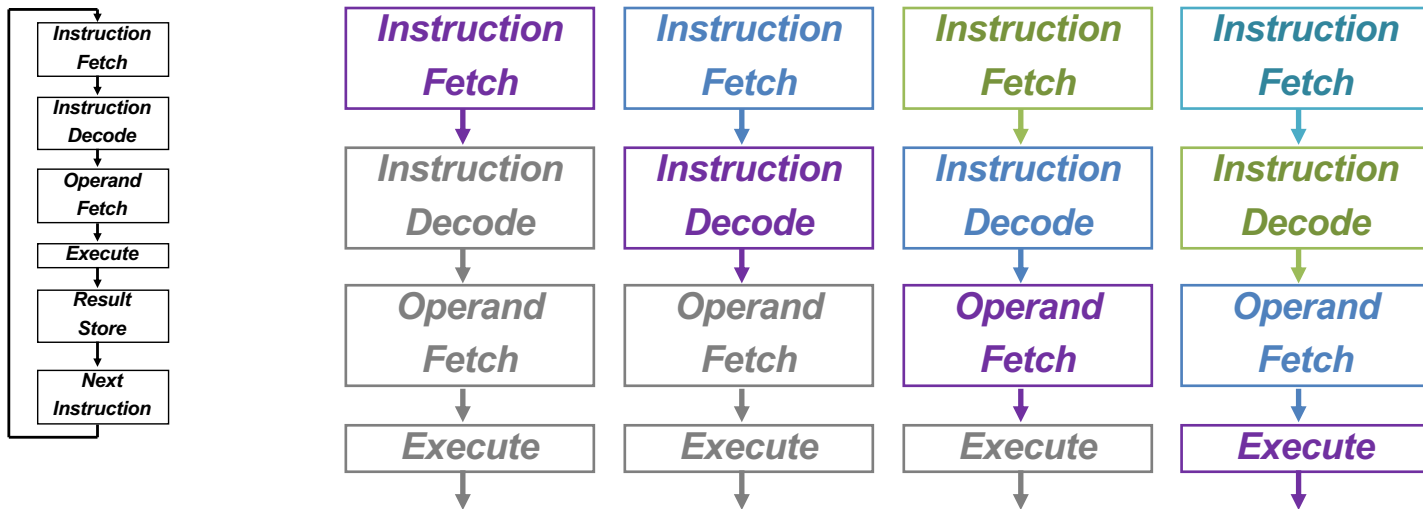
How to do parallelism well?

- Second half of this class
- CSE148

# A brief preview of some machine organization concepts:

## *Pipelining*

- Overlapping parts of a large task to increase throughput without decreasing latency
  - Key insight: The less work you do in one step, the faster each step can finish

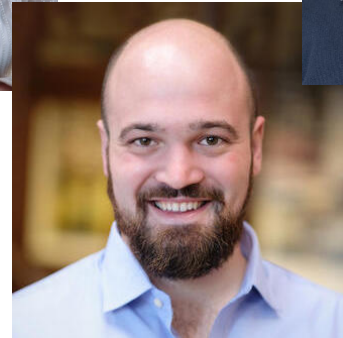
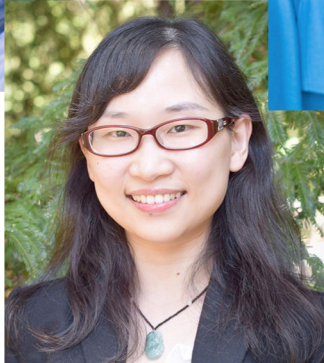
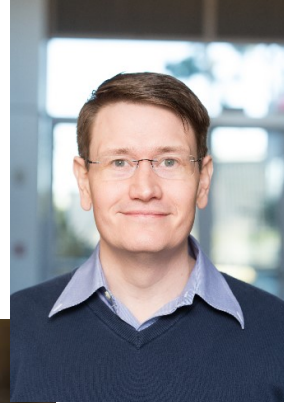


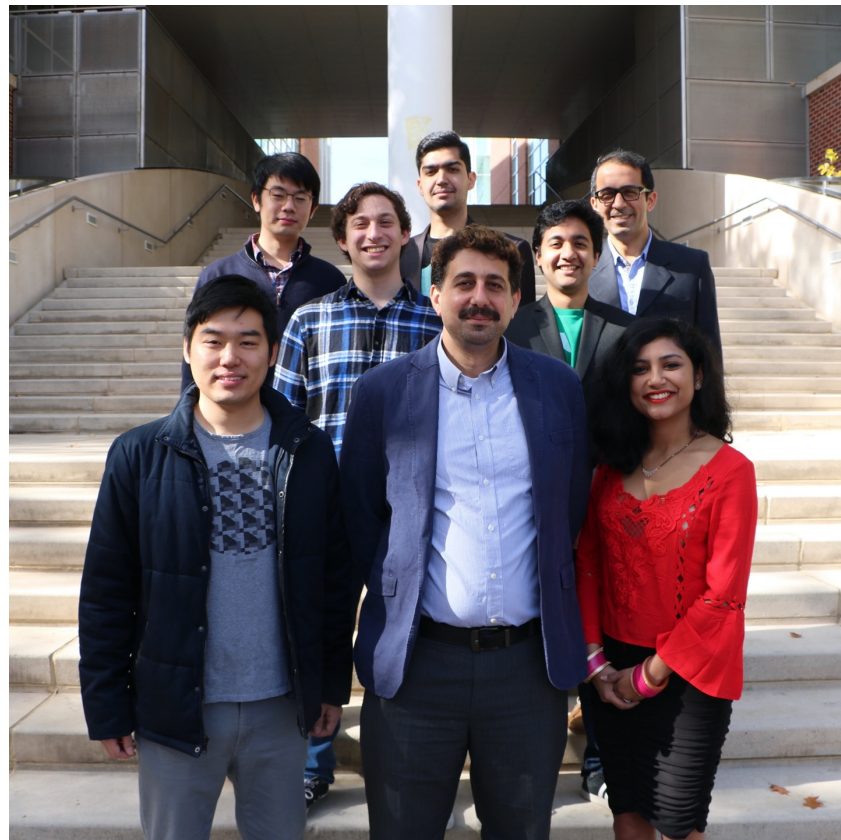
We'll take a short break here...

# **AND THEN SOME MODERN HIGHLIGHTS FROM HERE AT UCSD**

# But for the rest of today, I want to highlight the kinds of cool stuff that architects *do*

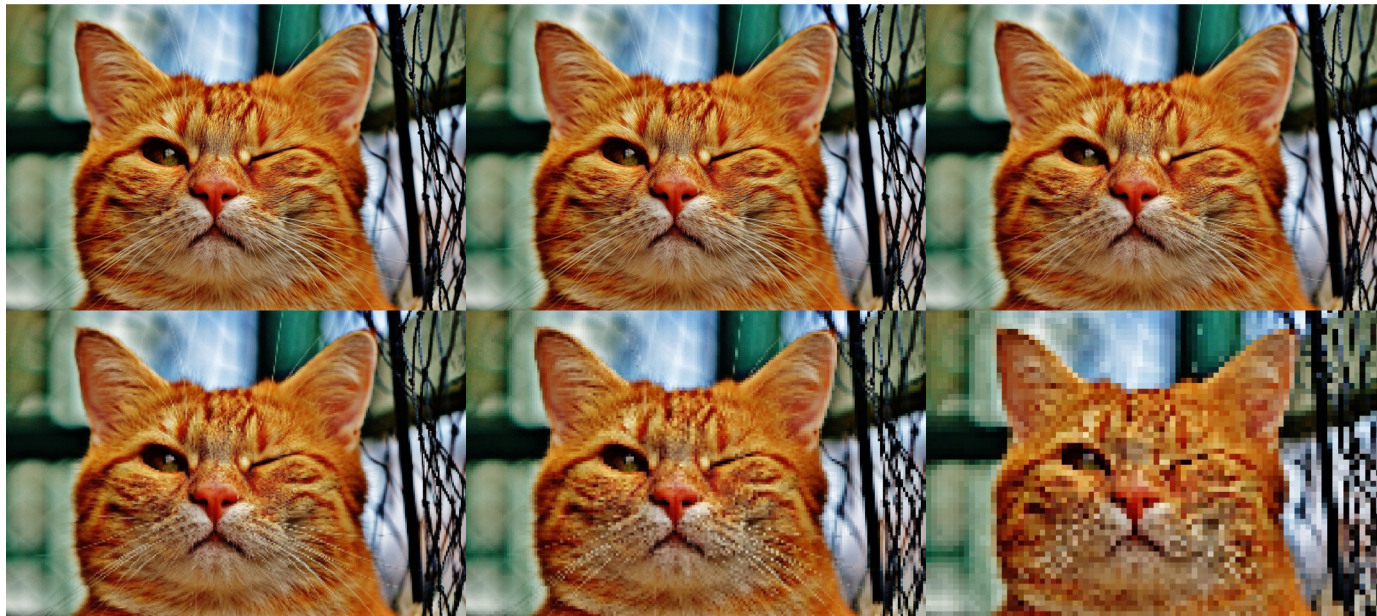
- UCSD has an amazing team of architecture faculty



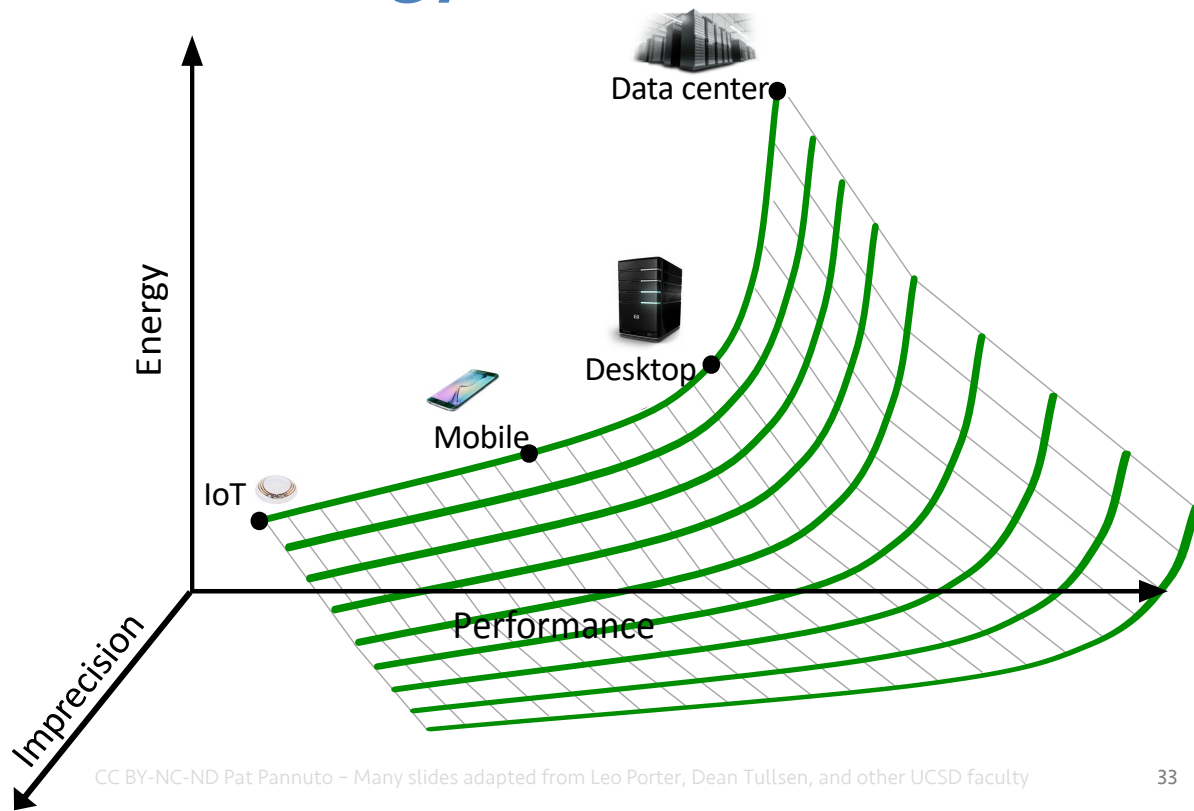


# One wild idea: “Approximate Computing”

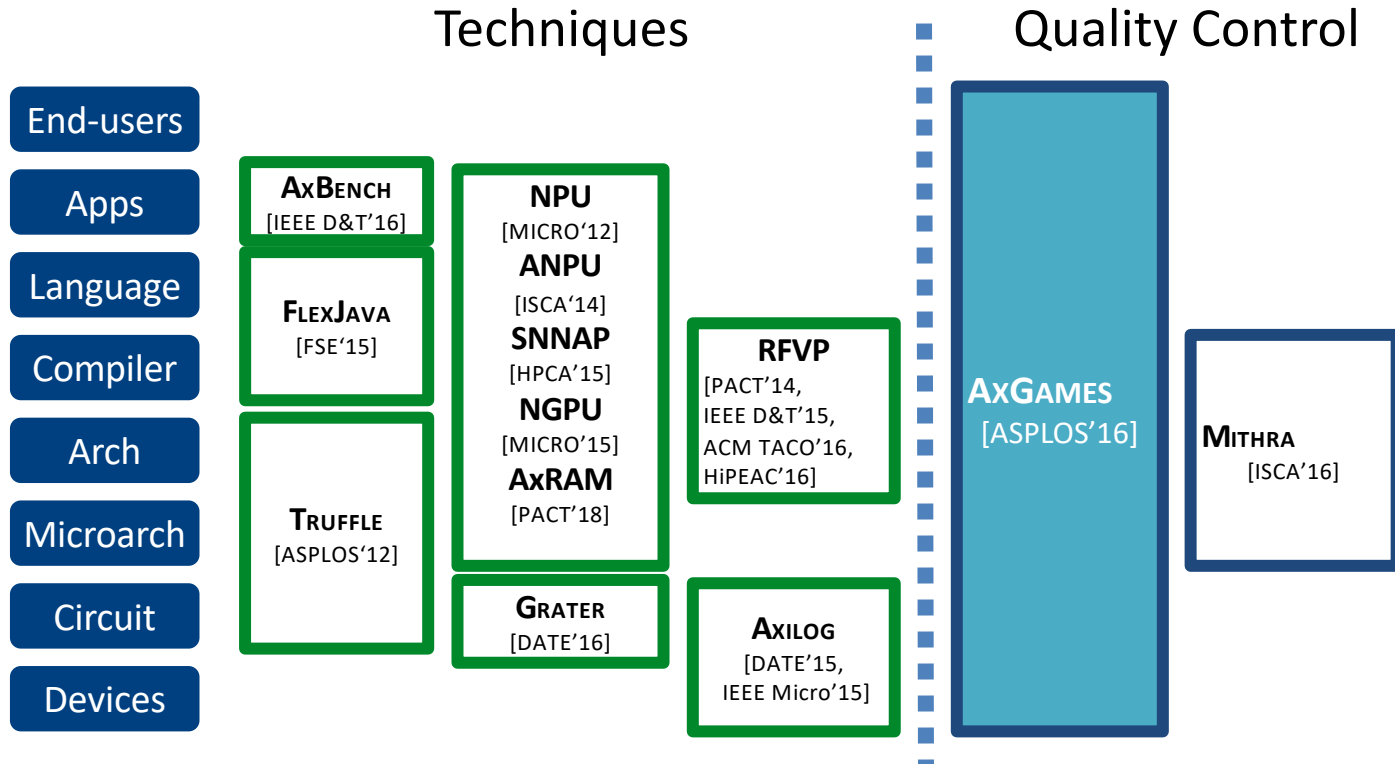
- Aka, what if  $1 + 1$  doesn't *always* equal *exactly* 2?



# Embracing imprecision allows for major gains in performance and energy

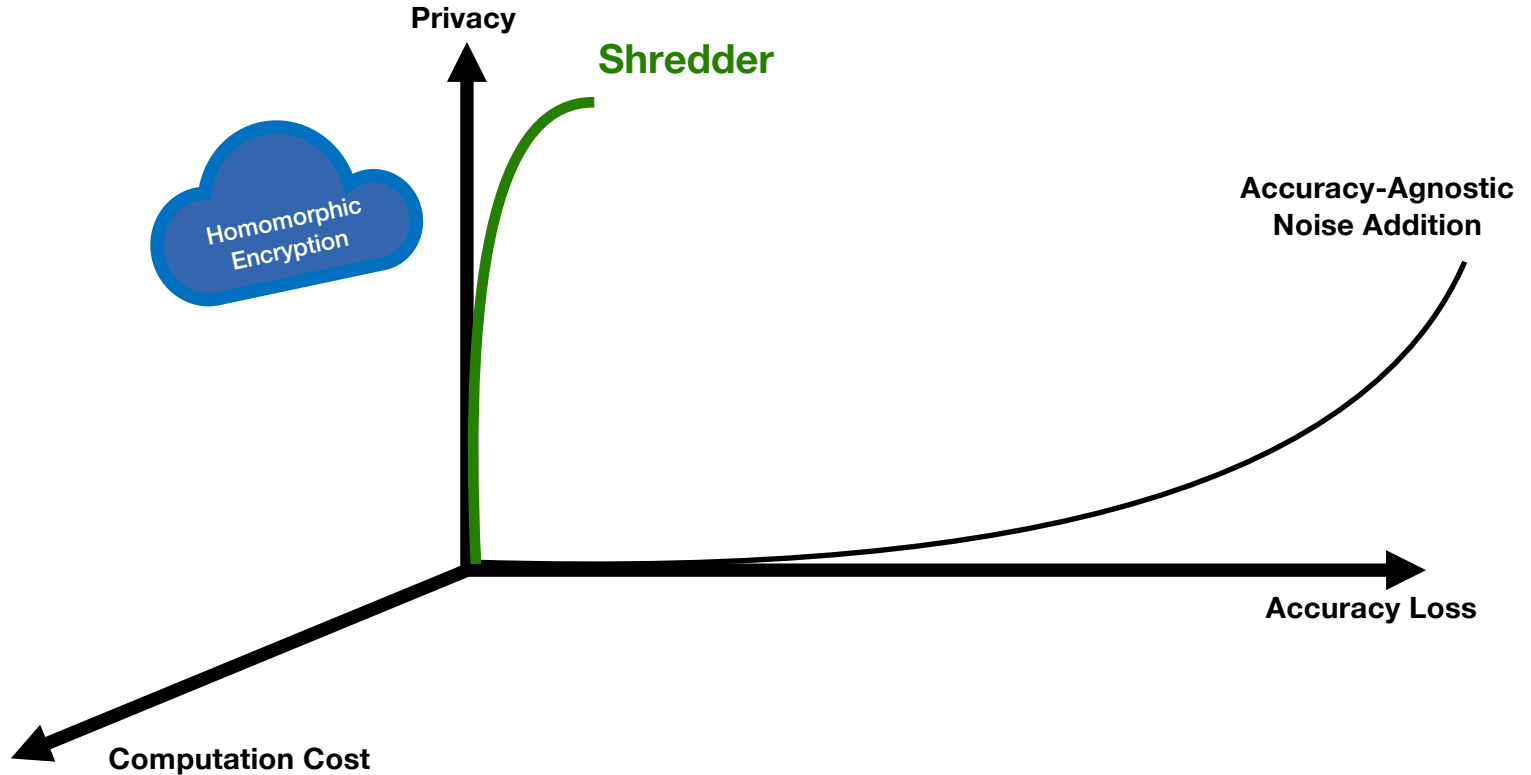


# A cross-stack approach to enable approximation

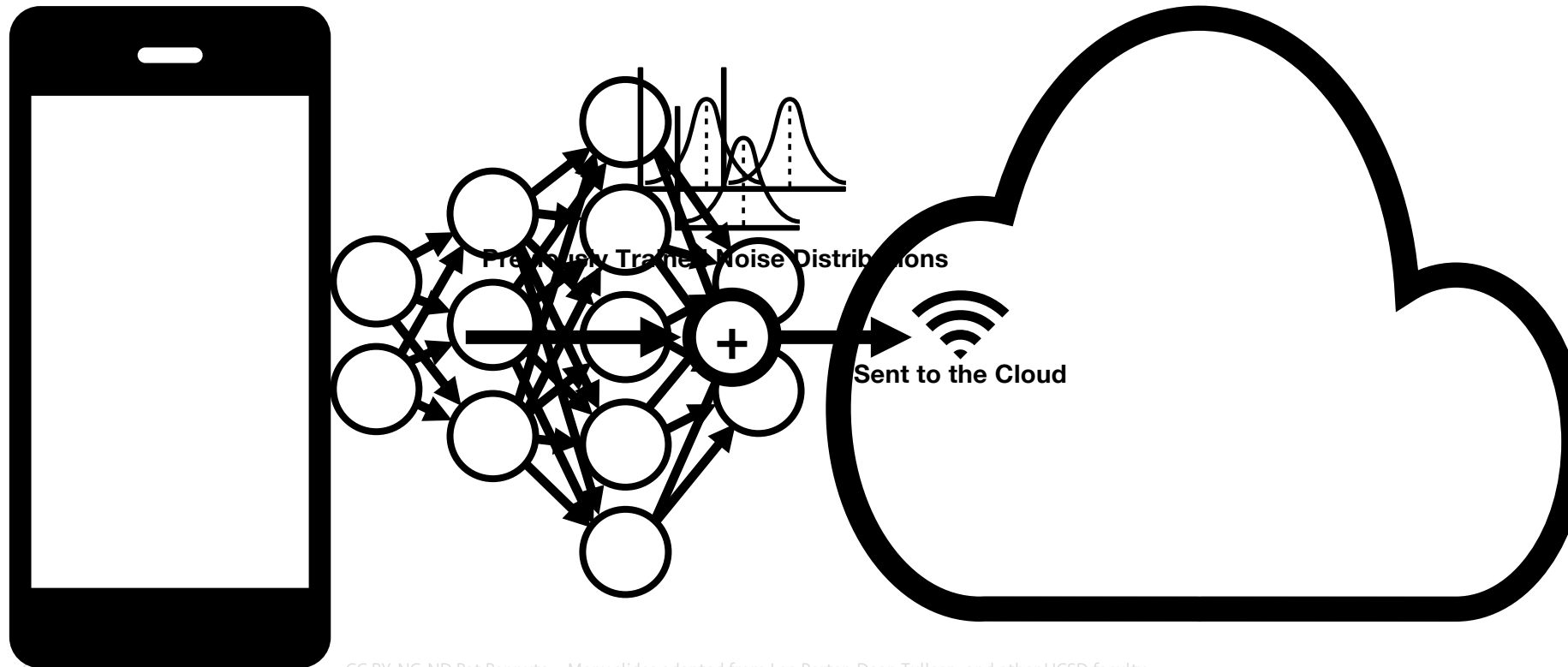




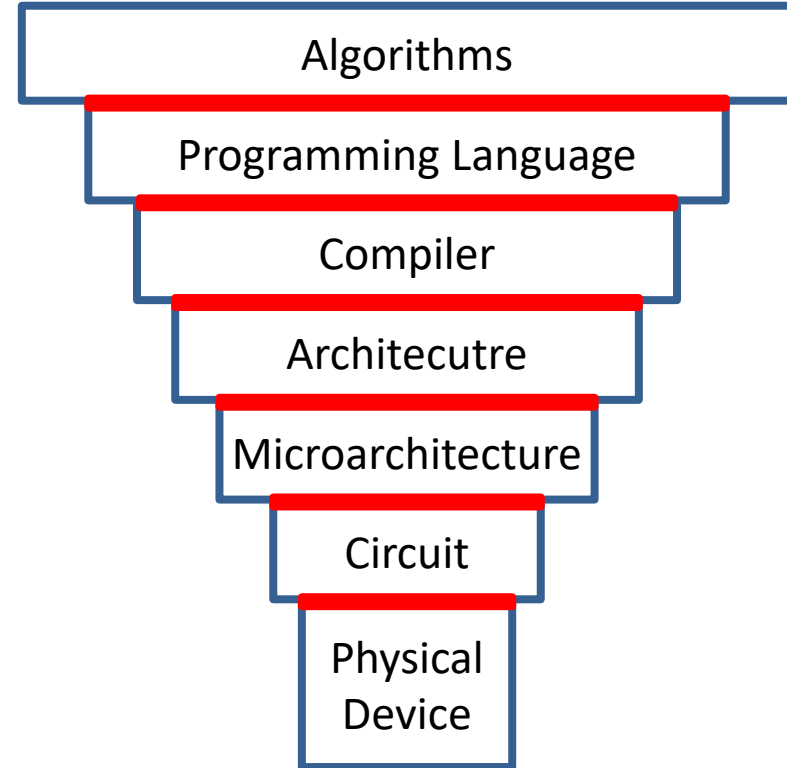
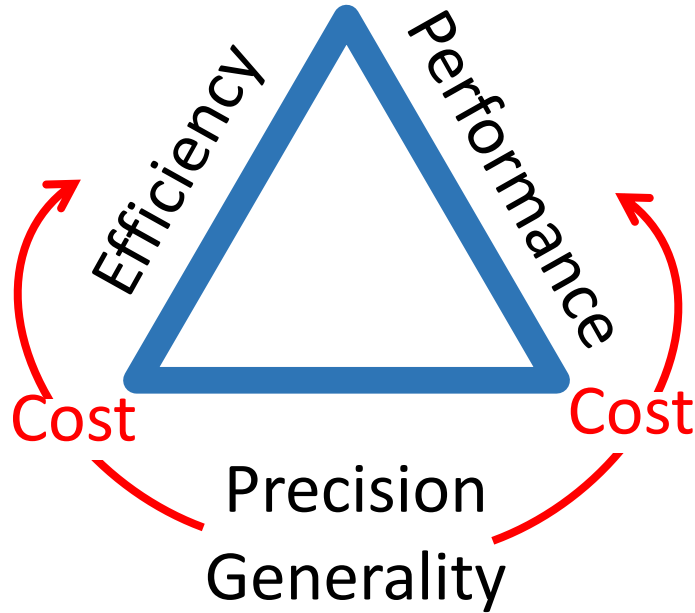
# Privacy Preserving Techniques for Inference



# Execution Model



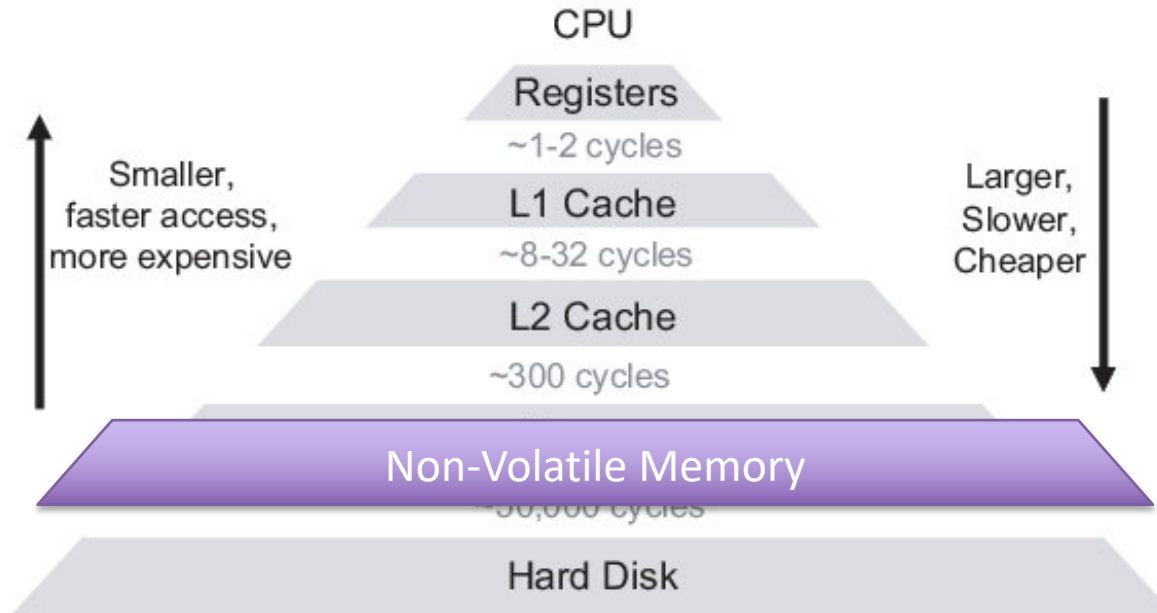
# Rethinking the abstractions



# Memory, Storage, Software, and Architecture in the NVSL



# This is a slide you will encounter in many CE/CSE classes...



# Applications

MARS

Willow

NOVA

Orion

Ziggurat

SubZero

NV-Heaps

Pangolin

Pronto

Tools

Libraries

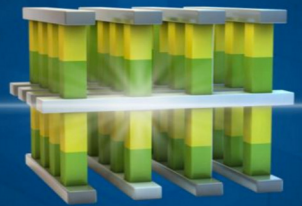
Stacks

Operating Systems

Moneta

QuickSAN

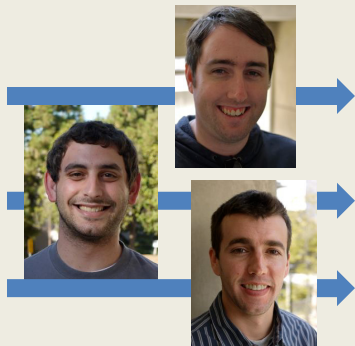
3D XPOINT



# NVSL Students Lead Industry

- We Built

- Opt. SSD interface (2009)
- Direct, remote SSD (2013)
- First PCM SSD (2011)
- PMEM prog. tools (2011)



- Industry Built

- NVMe (2011)
- NVMe over Fabrics (2016)
- Optane (2016)
- PMDK (~2014)

# Mobilizing the Micro-Ops: Exploiting **Context Sensitive Decoding** for Security and Energy Efficiency





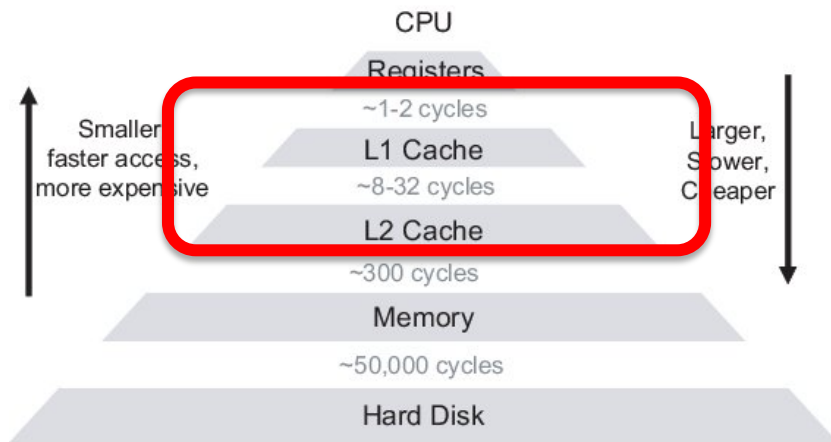
# Leaky abstractions are not always just performance problems...

- This loop behaved differently because of how caches work

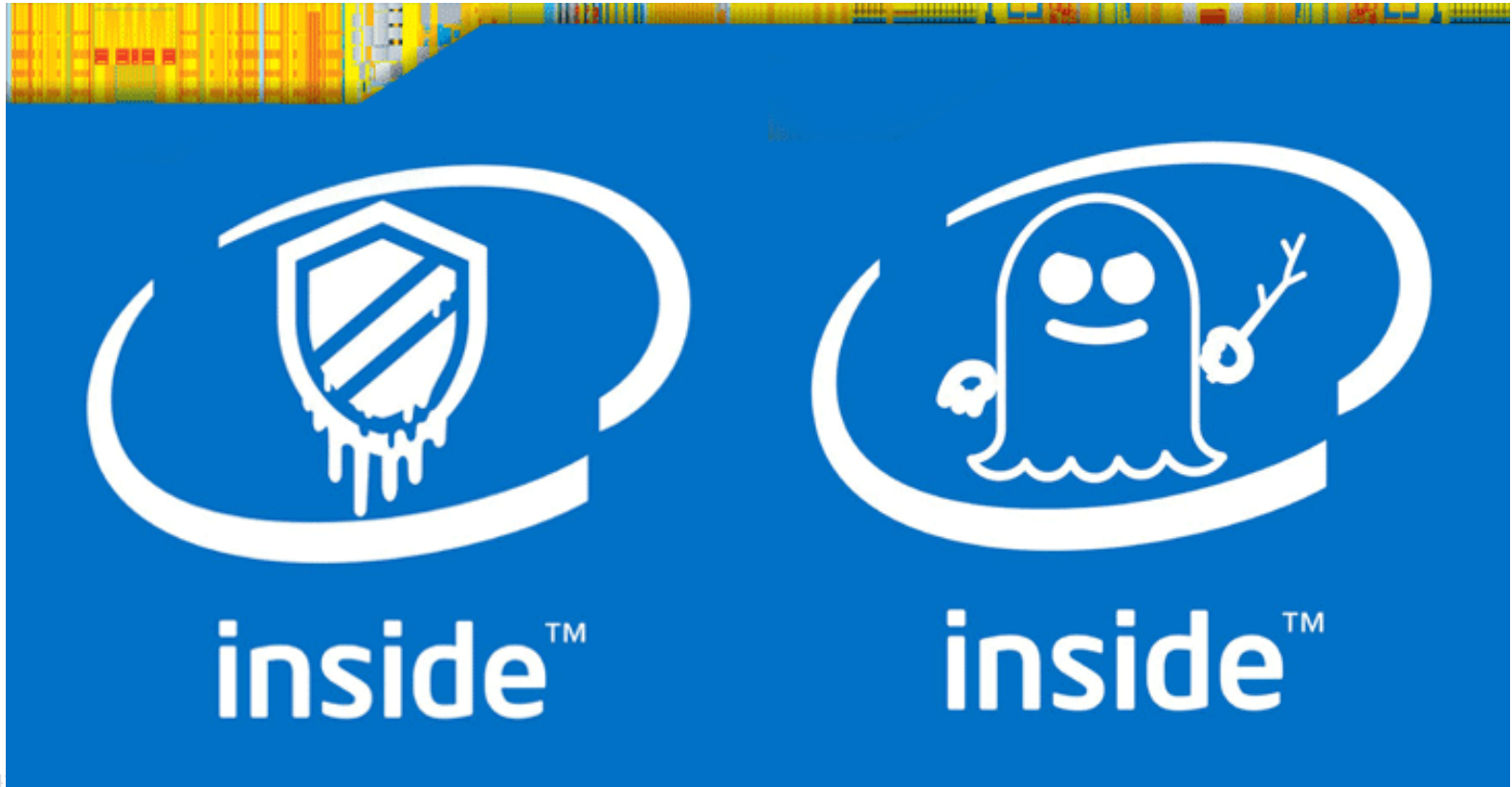
```
int twoDarray[256][256];
int sum = 0;

for (int i=0; i<256; i++) {
    for (int j=0; j<256; j++) {
        sum += twoDarray[i][j];
    }
}
```

Architects added “hidden” caches:  
faster, intermediate memories

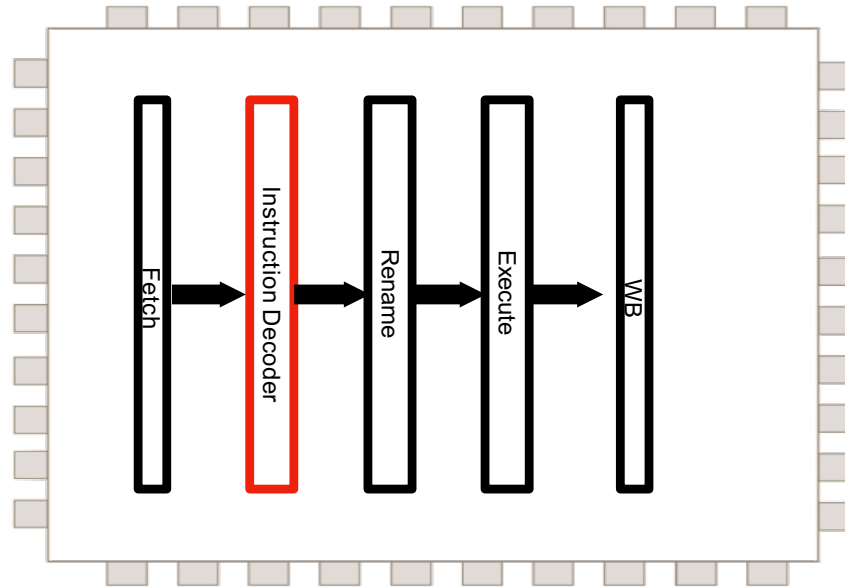


# Leaky abstractions can be security threats!



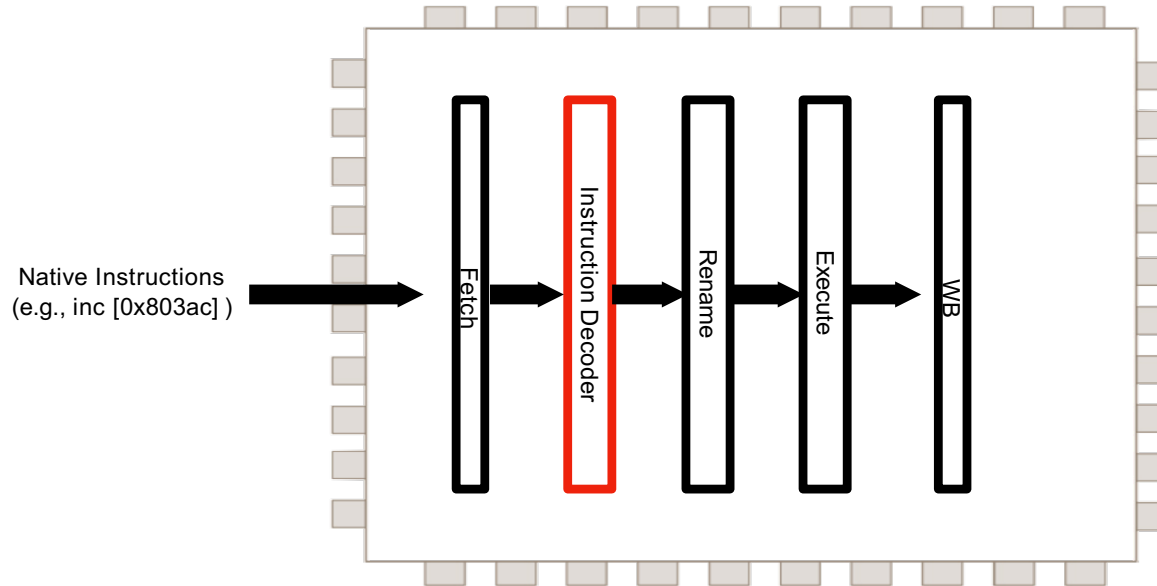
# Mobilizing the Micro-Ops

## Exploiting Translated ISAs



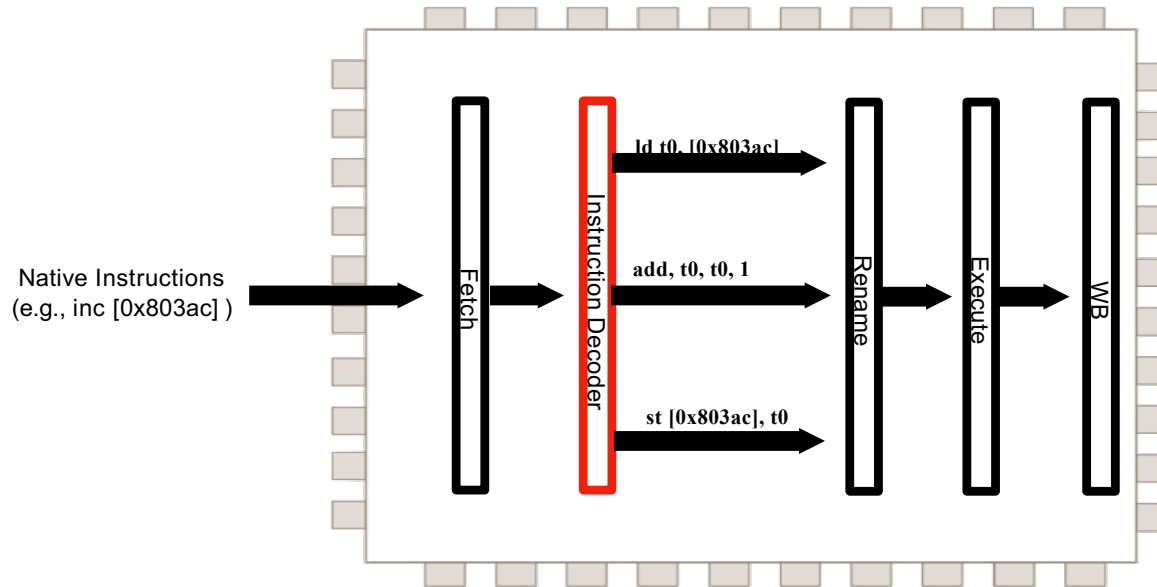
# Mobilizing the Micro-Ops

## Exploiting Translated ISAs



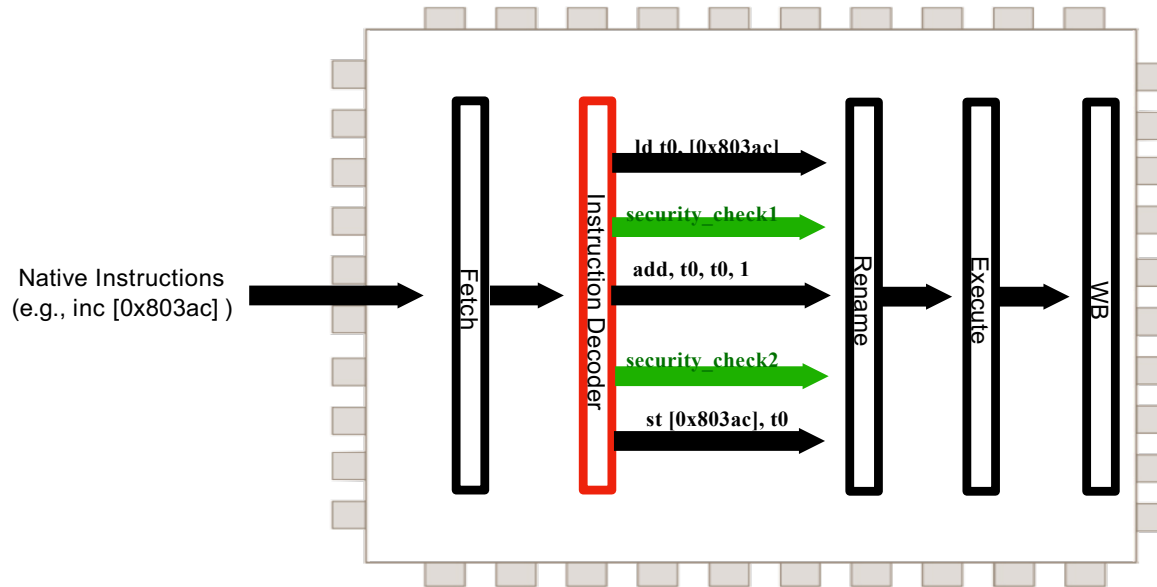
# Mobilizing the Micro-Ops

## Exploiting Translated ISAs



# Mobilizing the Micro-Ops

## Exploiting Translated ISAs

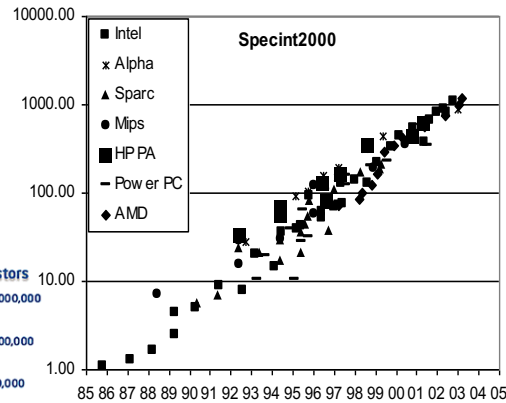
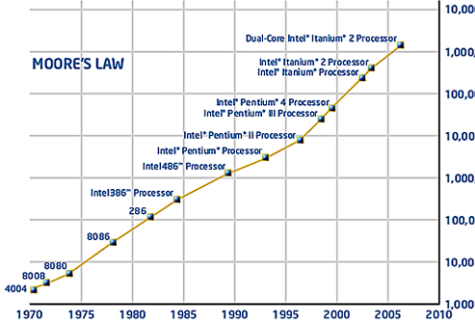
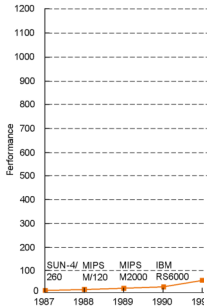


# Context Sensitive Decoding fixes a leaky abstraction

- Eliminating cache side channels via cache obfuscation
- Energy and Performance optimization via selective devectorization
  - ISCA 2018
  - IEEE Micro Top Picks in Computer Architecture
- Spectre mitigation via targeted insertion of fence micro-ops (**Context Sensitive Fencing**)
  - ASPLOS 2019

# Performance was king, until we unplugged computers

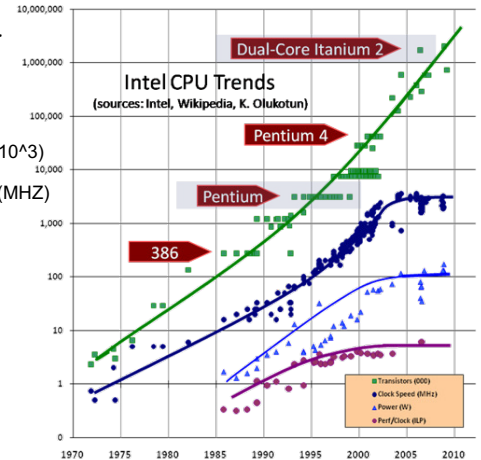
- A lot of “classic” architecture research is makes sure graphs continue to go up and to the right



## Processor Design Trends

- Transistors ( $\times 10^3$ )
- Clock Speed (MHZ)
- Power (W)
- ILP (IPC)

\*From Herb Sutter, Dr. Dobbs Journal

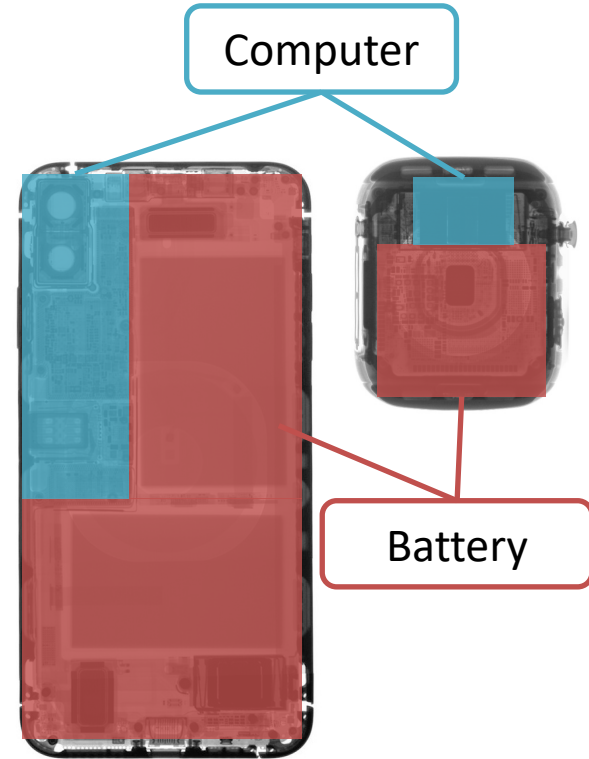
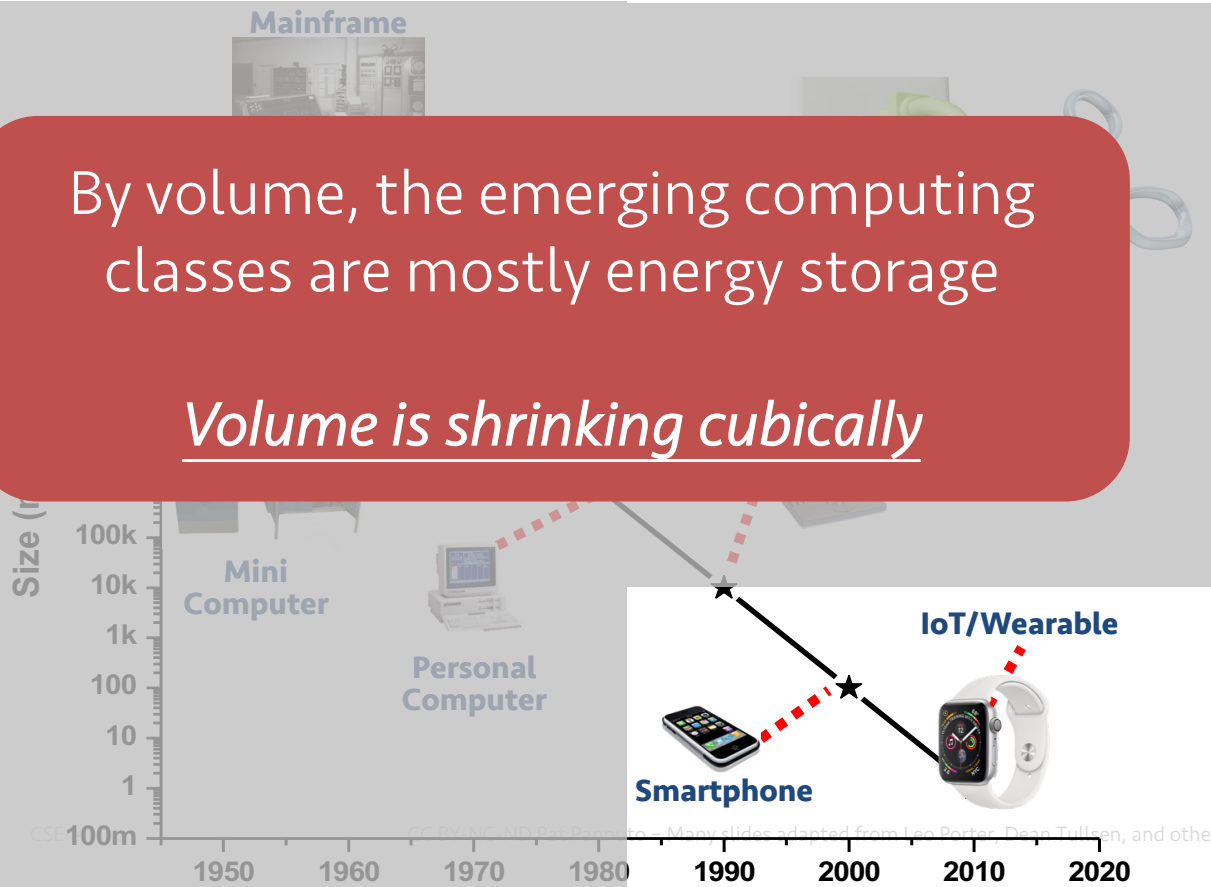




# I spend my time on graphs that go down and to the right

By volume, the emerging computing classes are mostly energy storage

Volume is shrinking cubically

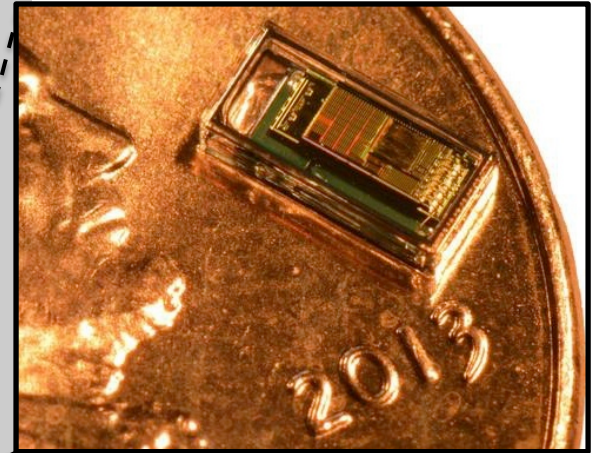


# Computational platforms will continue to scale

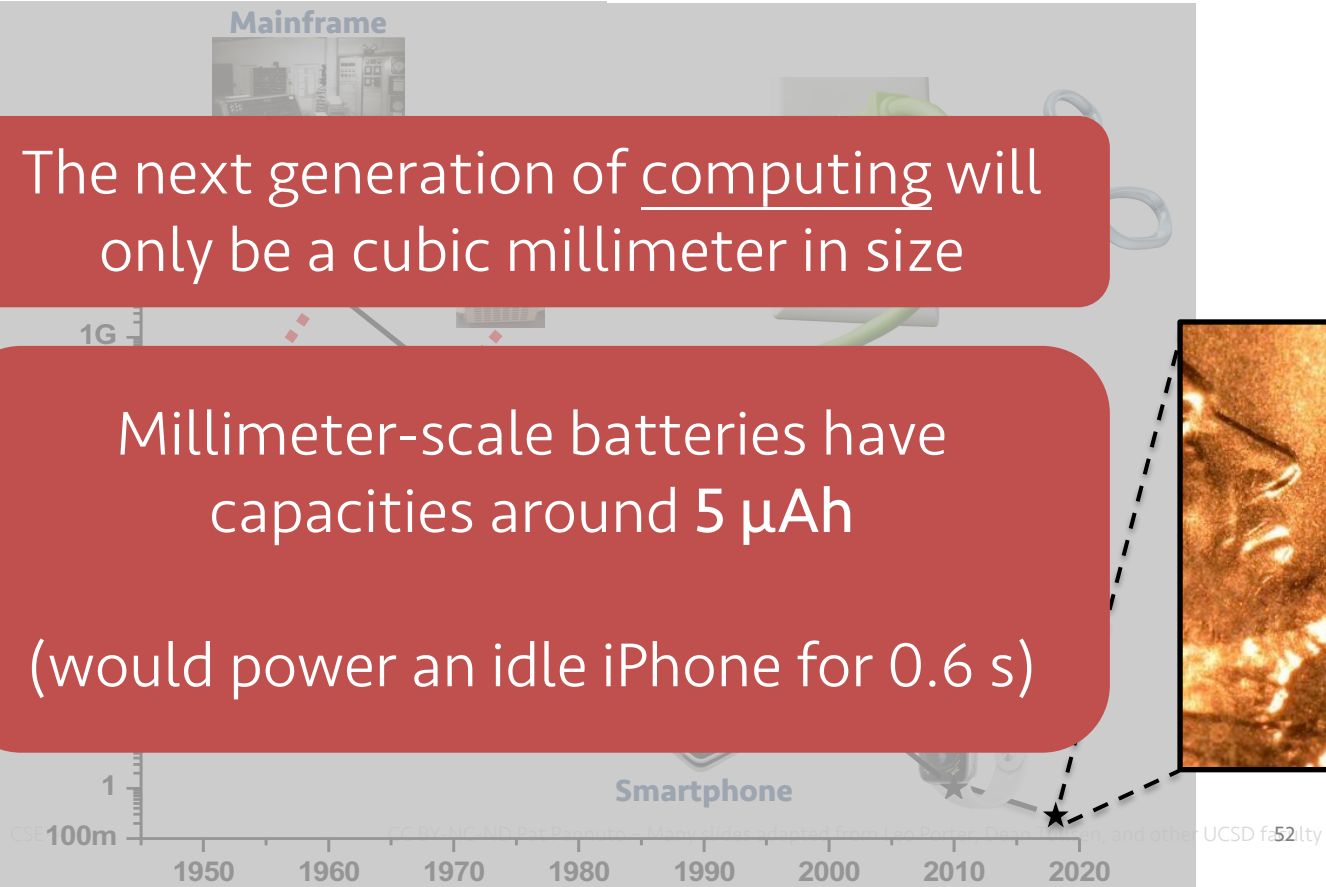
The next generation of computing will only be a cubic millimeter in size

Millimeter-scale batteries have capacities around  $5 \mu\text{Ah}$

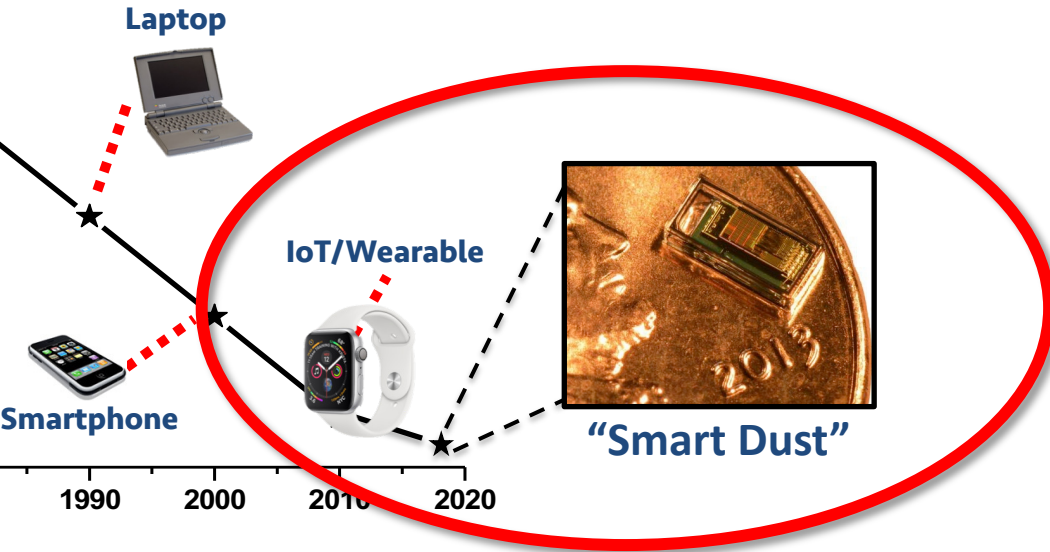
(would power an idle iPhone for 0.6 s)



**"Smart Dust"**

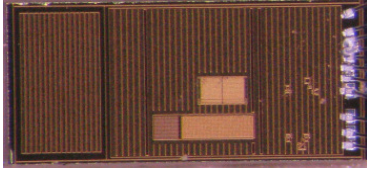


# Energy constraints will play a central role in the evolution of computing platforms



How must traditional paradigms change, adapt, or re-invent for the new computing classes?

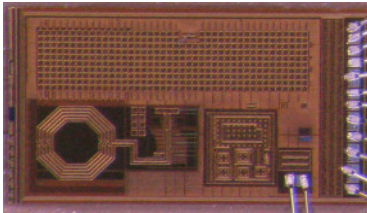
# One of the first challenges was re-thinking how we put together computers



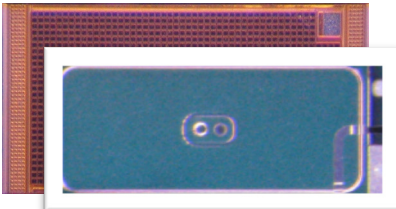
**Temperature Sensor**  
~10 pW standby, < 1  $\mu$ W active



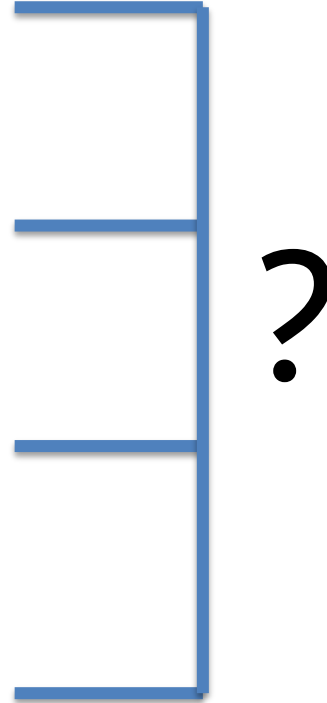
**CPU**  
~1 nW standby, ~5  $\mu$ W active



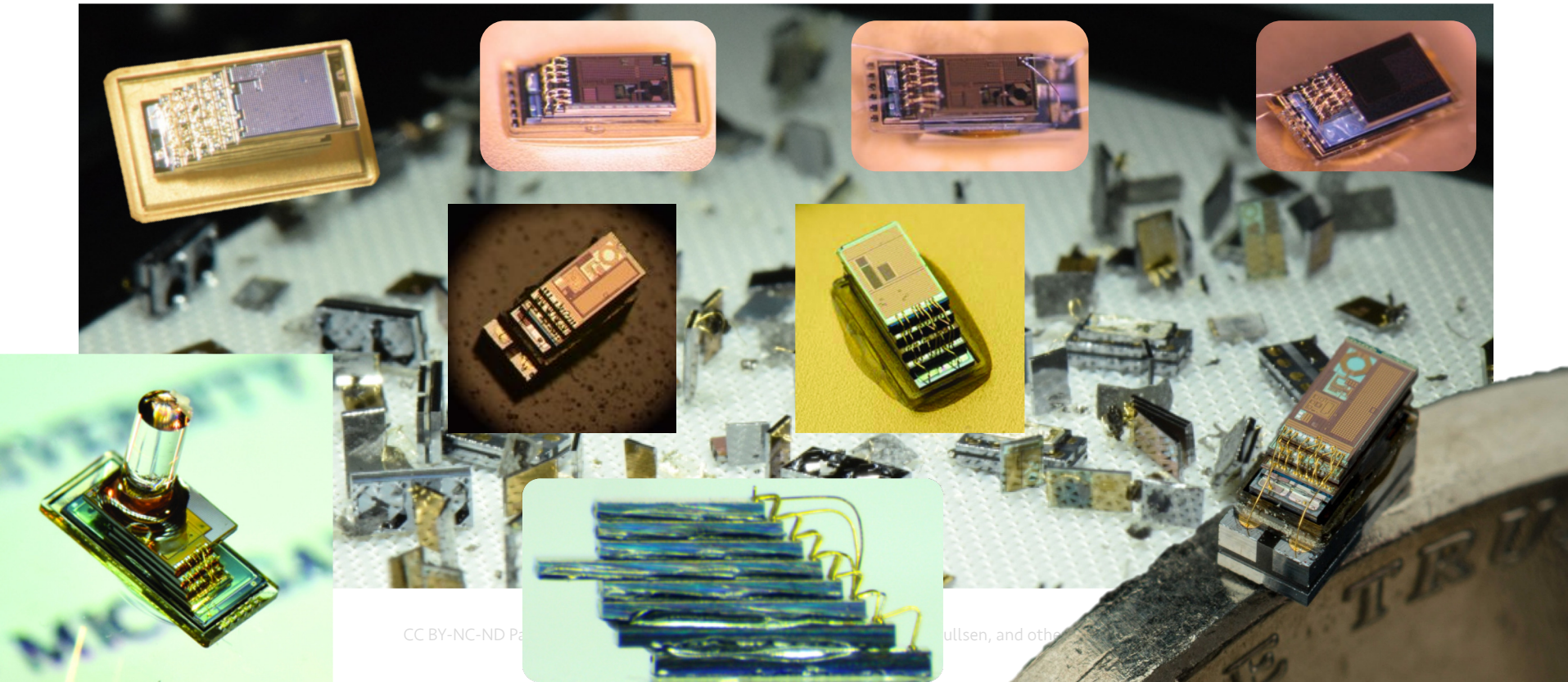
**Radio**  
~10 pW standby, ~10  $\mu$ W active



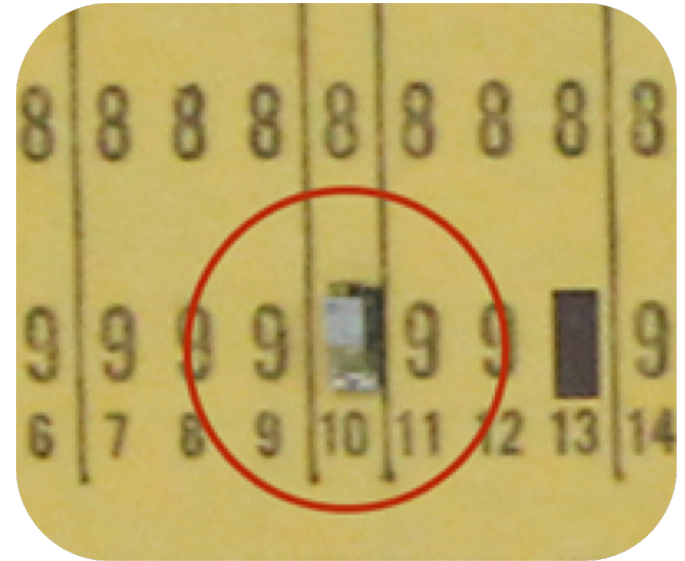
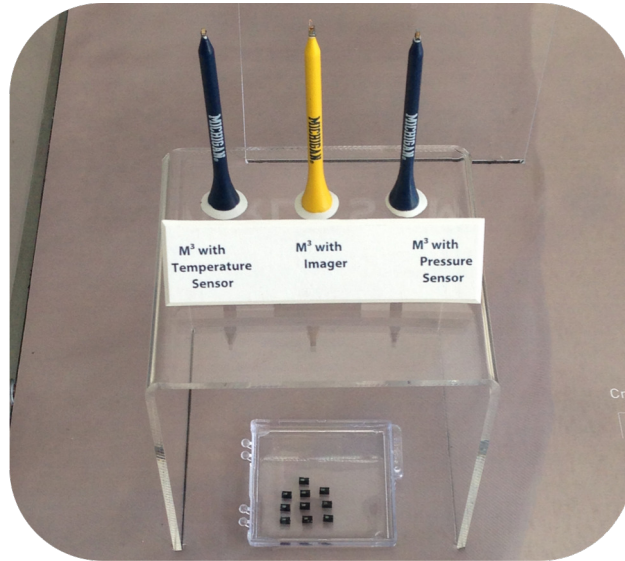
**Energy Harvesting & Storage**  
1~10 nW indoors  
2~10  $\mu$ Ah capacity



# MBus enabled the development of dozens of millimeter-scale motes as part of the Michigan Micro Mote (M3) project



# Check out the “World’s Smallest Computer” exhibit at Silicon Valley’s Computer History Museum!



# Next time: Instruction Set Architectures (ISAs)

- Reading:
  - Skim 1.1 [7 pages]
  - Read 1.2, 1.3 [6.5 pages]
- Okay if not until Apr 5:
  - Skim 2.1-2.2 [5 pages]
  - Read 2.3-2.5 [16 pages]
  - Skim 2.10 [10 pages]

**What is Computer Architecture?**

Computer Architecture =  
Machine Organization +  
Instruction Set Architecture

*What the machine hardware looks like*

*How you talk to the machine*

CSE 141 CC BY-NC-ND Pat Pannuto – Many slides adapted from Dean Tullsen

The diagram is enclosed in a black rectangular border. At the top left, the title 'What is Computer Architecture?' is written in blue. Below it, the equation 'Computer Architecture = Machine Organization + Instruction Set Architecture' is displayed. To the right of 'Machine Organization', a curved arrow points to the green italicized text 'What the machine hardware looks like'. To the right of 'Instruction Set Architecture', a curved arrow points to the green italicized text 'How you talk to the machine'. At the bottom left, 'CSE 141' is written in small grey font. At the bottom right, 'CC BY-NC-ND Pat Pannuto – Many slides adapted from Dean Tullsen' is written in small grey font.