CSE 141: Introduction to Computer Architecture

Pat Pannuto, UC San Diego

ppannuto@ucsd.edu

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

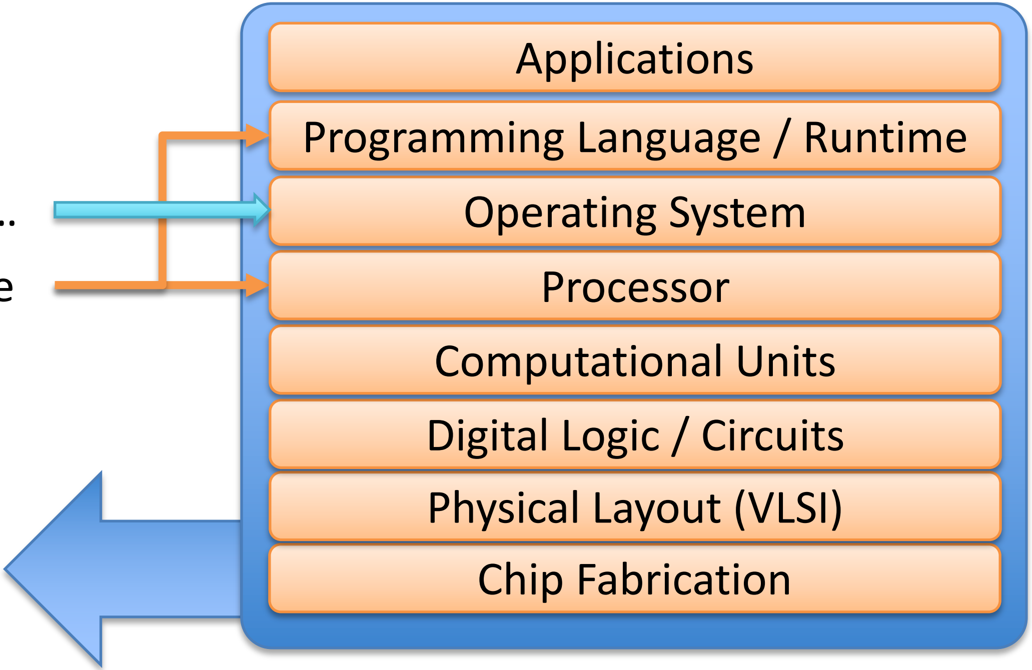
Heydar Aliyev Center in Baku, designed by Zaha

Graphics courtesy Wikipedia

Computer science is all about abstractions

The person who designs this…

…only needs to worry about these



Applications

Programming Language / Runtime

Operating System

Processor

Computational Units

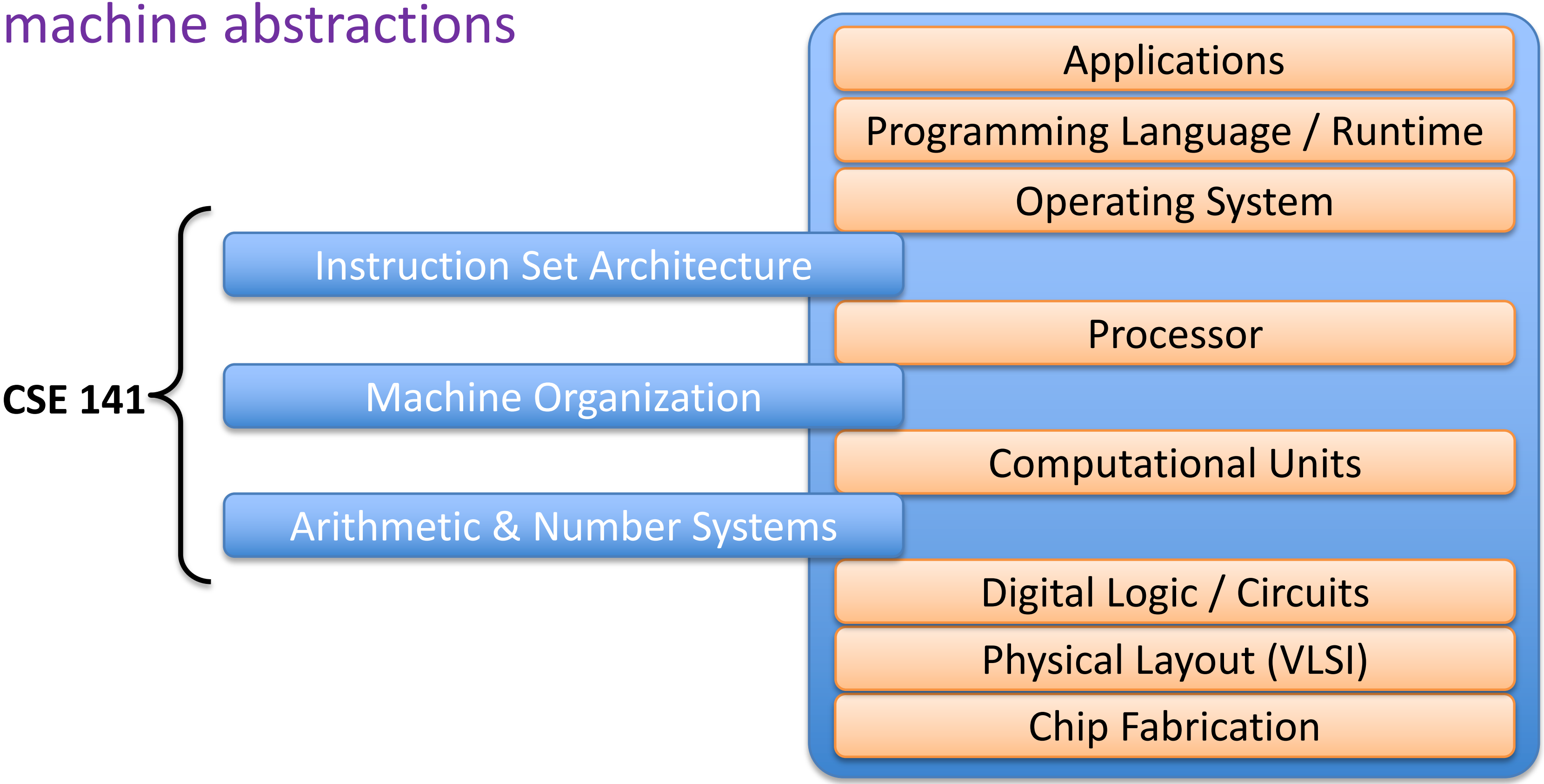
Digital Logic / Circuits

Physical Layout (VLSI)

Chip Fabrication

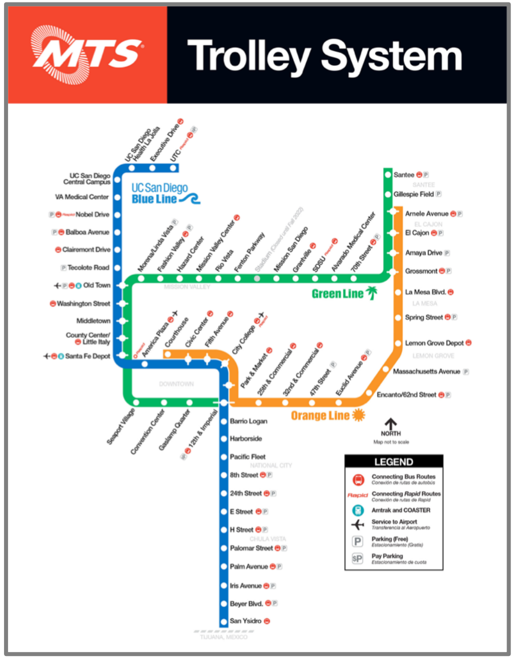
But who **designs** the big picture?

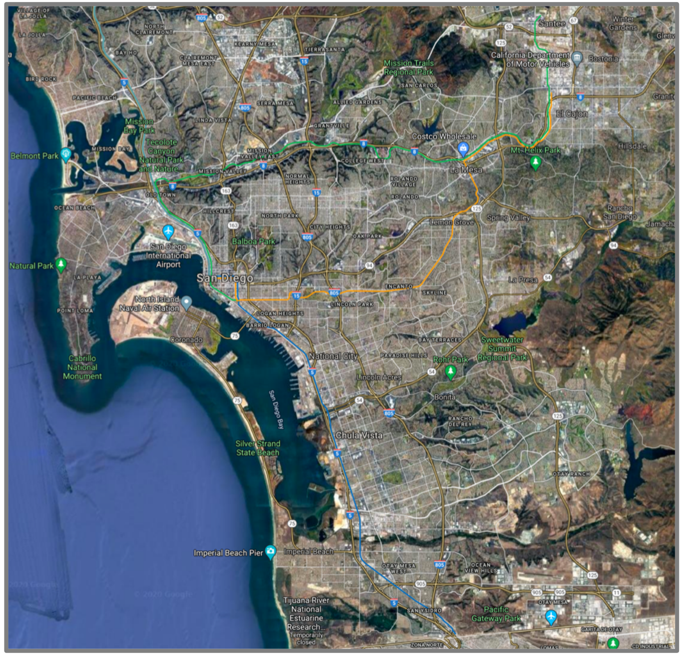
Computer architects look at the system as a whole and design machine abstractions



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

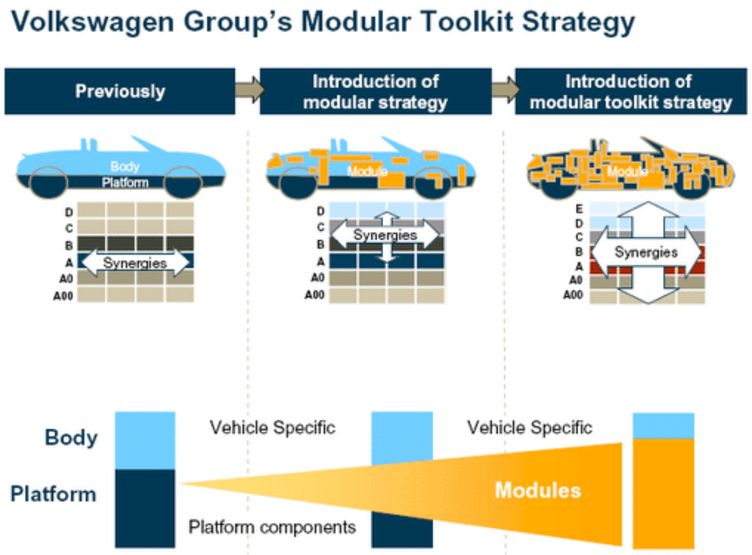
- Which of these maps is easier to use to plan a trolley trip?



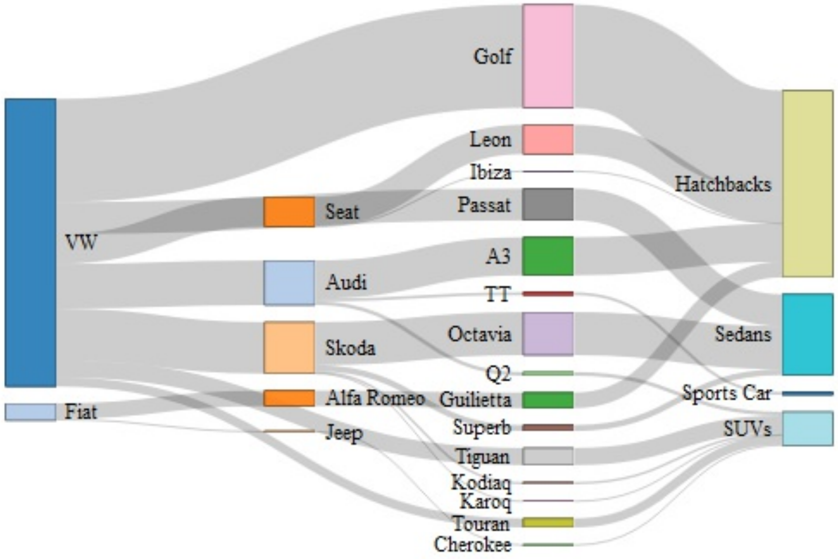

La Jolla
Mission Trails Regional Park
Santee
California Department of Motor Vehicles
El Cajon
Costco Wholesale
Mt. Helix Park
La Mesa
Belmont Park
Mission Bay Park
Tecolote Canyon Natural Park and Nature
Lemon Grove
Spring Valley
Balboa Park
San Diego International Airport
Natural Park
San Diego
North Island Naval Air Station
Coronado
Cabrillo National Monument
National City
Sweetwater Summit Regional Park
Bonita
Chula Vista
Silver Strand State Beach
Imperial Beach Pier
Imperial Beach
Tijuana River National Estuarine Research

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



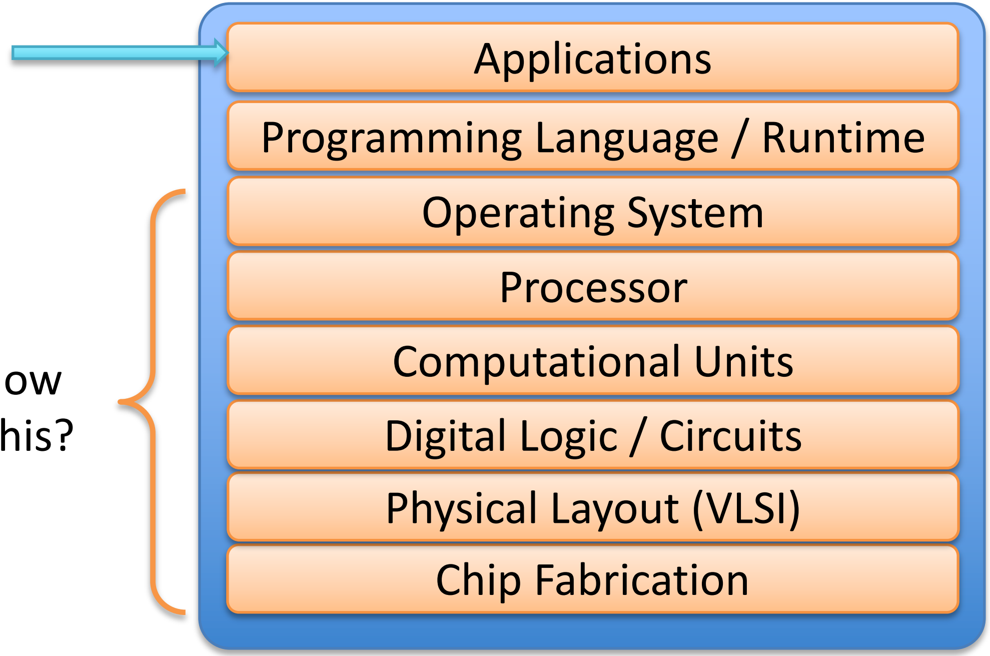
Modular Car Body Design and Optimization by an Implicit Parameterization Technique via SFE CONCEPT
Fabien Duddeck, Hans Zimmer



https://www.reddit.com/r/dataisbeautiful/comments/8m15g9/automobile_platform_sharing_work_in_progress/

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

If I only want to build these…



Applications

Programming Language / Runtime

Operating System

Processor

Computational Units

Digital Logic / Circuits

Physical Layout (VLSI)

Chip Fabrication

…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”

Course Administrivia

- Instructor
  - Pat Pannuto
- Support Infrastructure & Tools:
  - Piazza for Q&A
  - PrairieLearn for Homework
  - PrairieTest for Quizzes

CSE 141 TAs

Gabe Marcano
Email: gmarcano@ucsd.edu
Office hours:

- Monday 4:30-6:30pm in CSE 2123
- Friday 2-4pm on Zoom

Jen Switzer
Email: jfswitze@ucsd.edu
Office hours:

- Wednesdays 2-4pm in CSE 2123
- Fridays 10am-noon on Zoom

Discussions

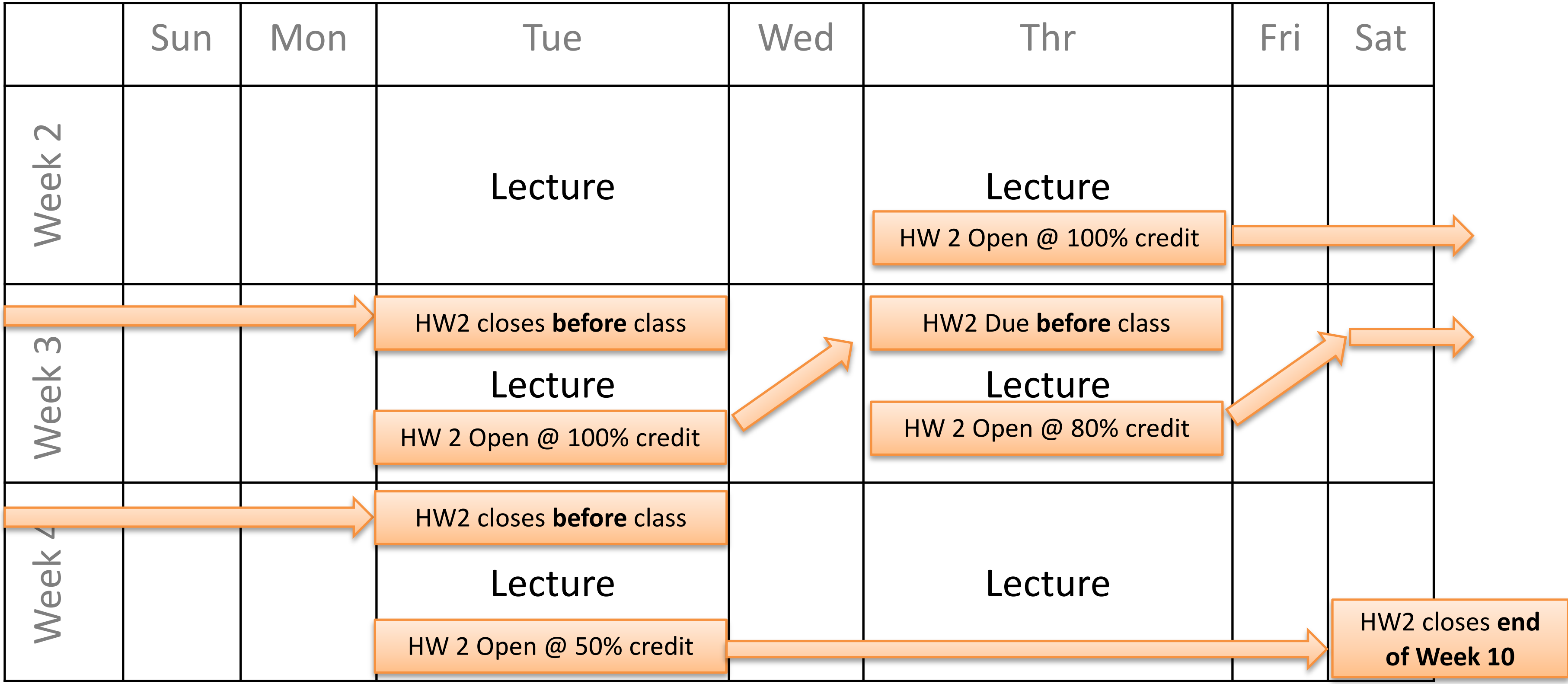
- More / different explanation of lecture concepts
- Interactive practice problems

- *(No discussion next week, will start week 2)*

Assessments & Workload

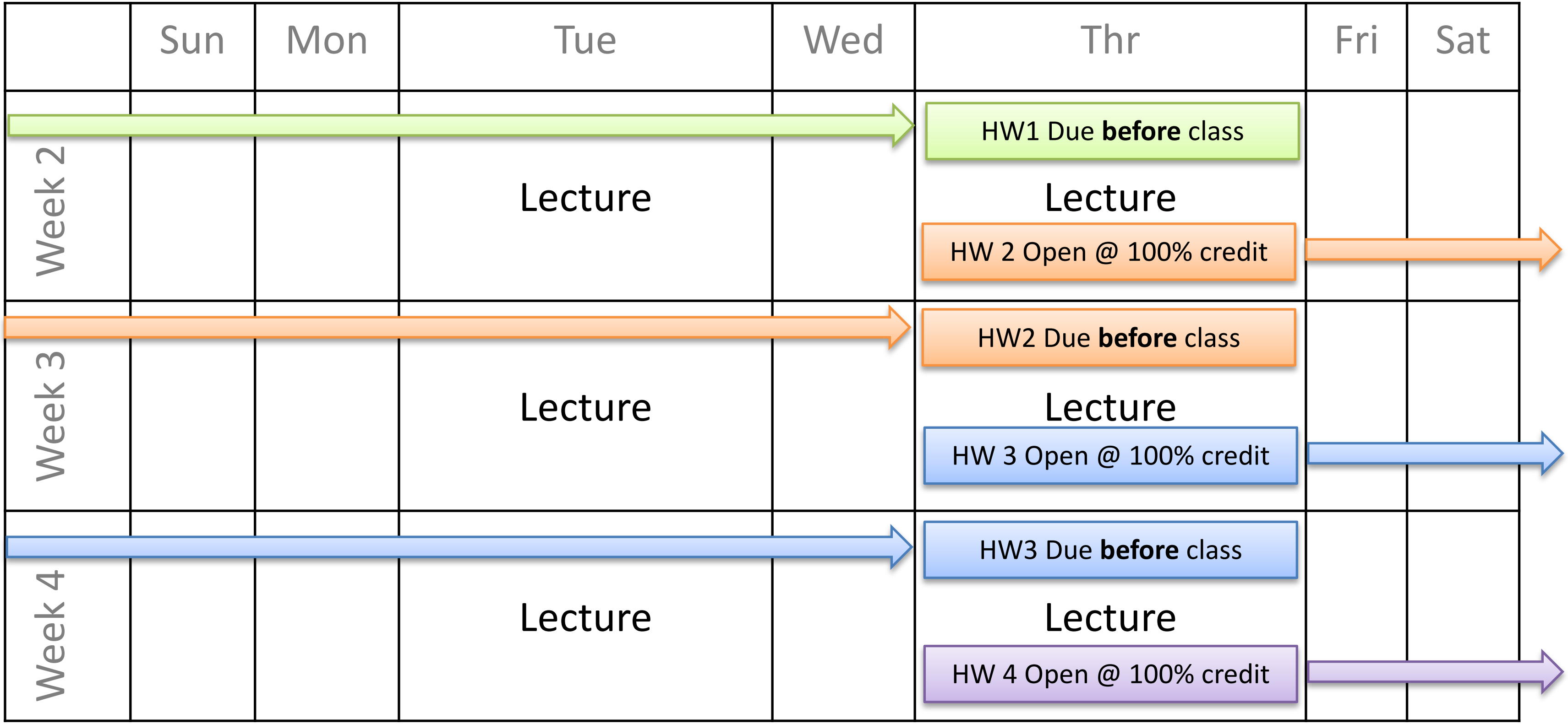
- Grading
  - Homework: 35%
  - Quizzes: 65%
    - Inclusive over the term — biased to later material

This class has a very regular “cadence”, in steady-state:



	Sun	Mon	Tue	Wed	Thr	Fri	Sat
Week 2			Lecture		Lecture HW 2 Open @ 100% credit		
Week 3			HW2 closes **before** class Lecture HW 2 Open @ 100% credit		HW2 Due **before** class Lecture HW 2 Open @ 80% credit		
Week 4			HW2 closes **before** class Lecture HW 2 Open @ 50% credit		Lecture		HW2 closes **end of Week 10**

Avoid falling behind, another assignment comes quick!

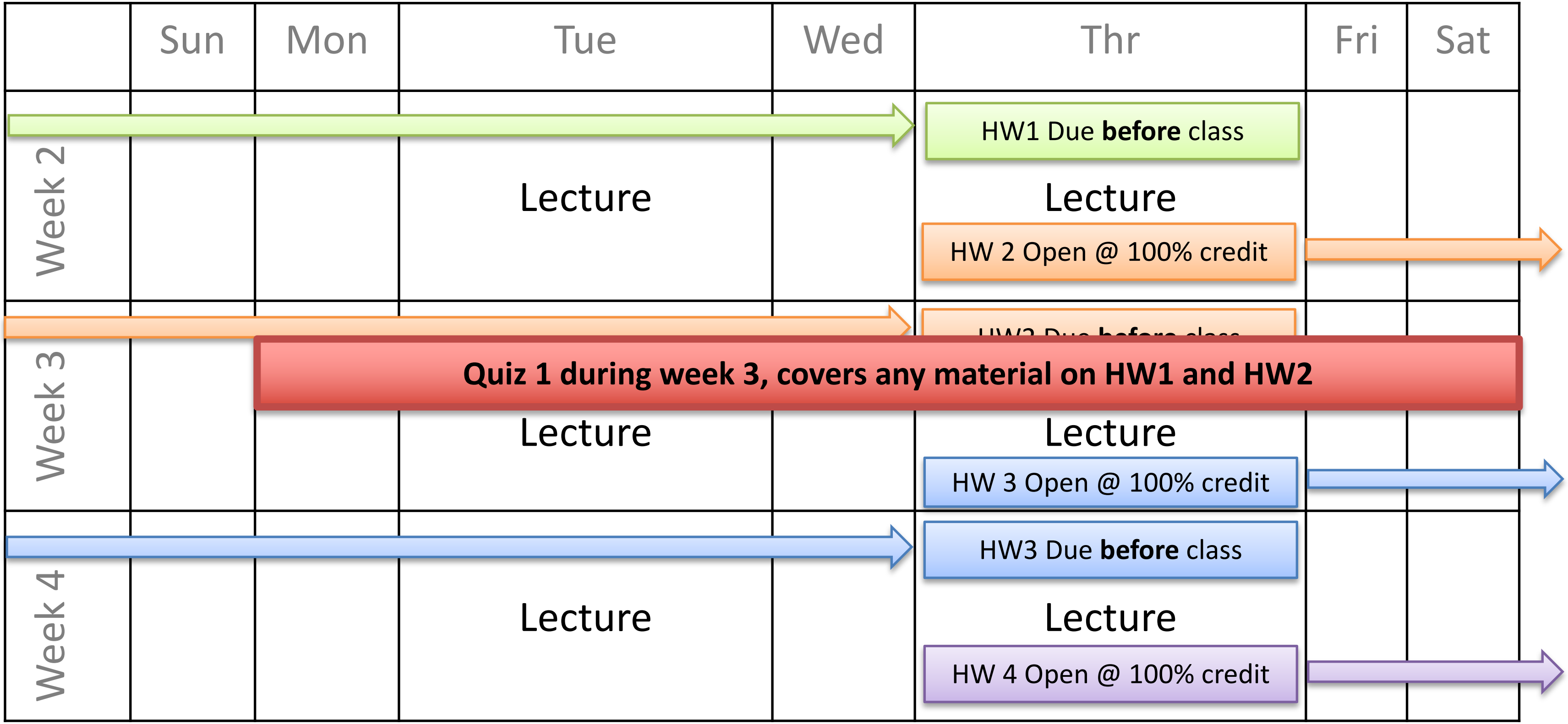


	Sun	Mon	Tue	Wed	Thr	Fri	Sat
Week 2			Lecture		HW1 Due **before** class Lecture HW 2 Open @ 100% credit		
Week 3			Lecture		HW2 Due **before** class Lecture HW 3 Open @ 100% credit		
Week 4			Lecture		HW3 Due **before** class Lecture HW 4 Open @ 100% credit		

Quizzes are outside of class at CBTF — You must schedule!

	Sun	Mon	Tue	Wed	Thr	Fri	Sat
Week 2			Lecture		Lecture		
Week 3		**Quiz 1 during week 3**	Lecture		Lecture		
Week 4			Lecture		Lecture		

Quizzes cover any material you have seen on HW



	Sun	Mon	Tue	Wed	Thr	Fri	Sat
Week 2			Lecture		HW1 Due **before** class Lecture HW 2 Open @ 100% credit		
Week 3			Lecture		HW2 Due **before** class Lecture HW 3 Open @ 100% credit		
Week 4			Lecture		HW3 Due **before** class Lecture HW 4 Open @ 100% credit		

Quiz 1 during week 3, covers any material on HW1 and HW2

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 quizzes "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)
  - Individual elements may be “internally” curved

Academic Integrity

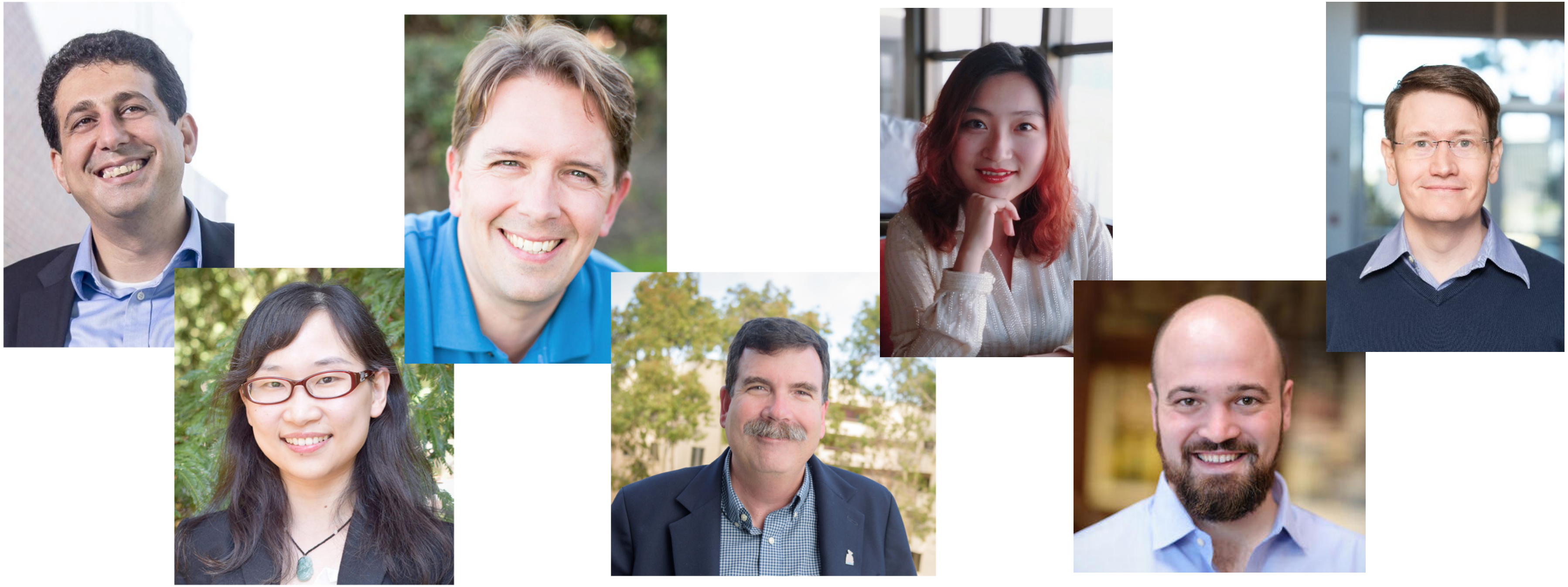
- Cheating will be taken very seriously
- Examples
  - Not cheating:
    - Discussing homework in groups, with **your hands on your own keyboard, doing your own question variants yourself**
    - Looking at lectures, practice problems & solutions, etc from “other 141’s”
  - Cheating:
    - Getting a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

We’ll take a short break here…

AND THEN SOME MODERN HIGHLIGHTS FROM HERE AT UCSD

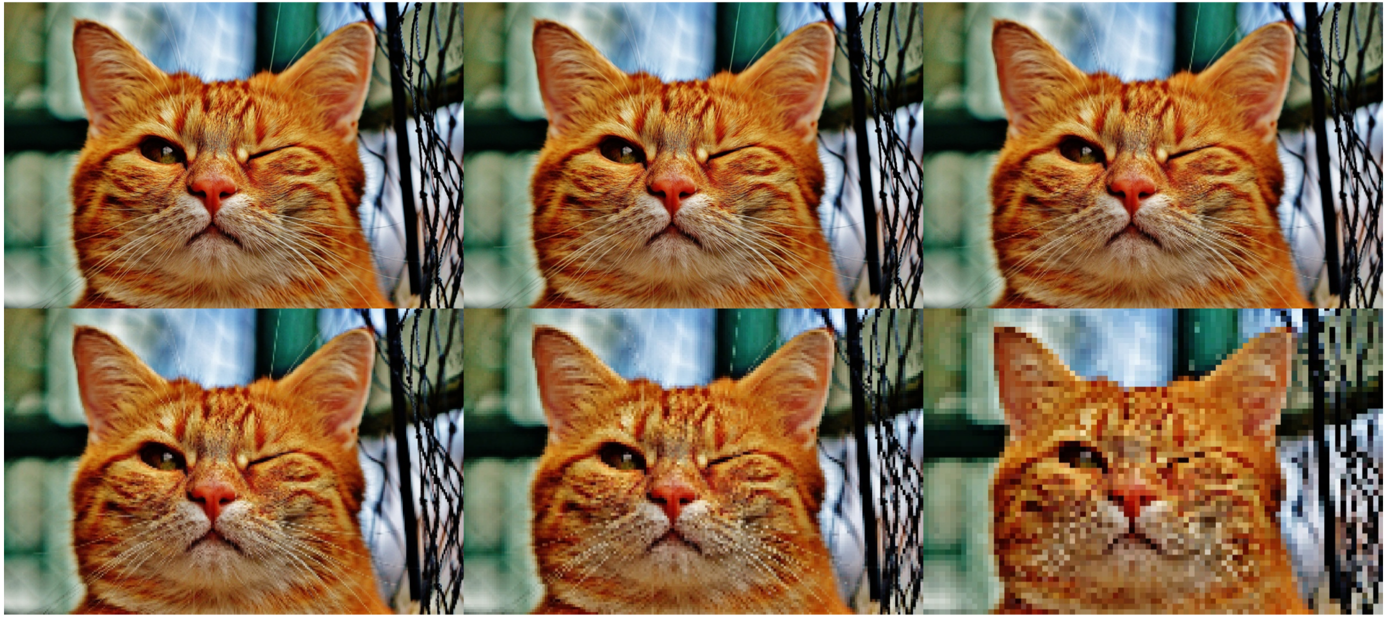
I want to highlight the kinds of cool stuff that architects ***do***

- UCSD has an amazing team of architecture faculty

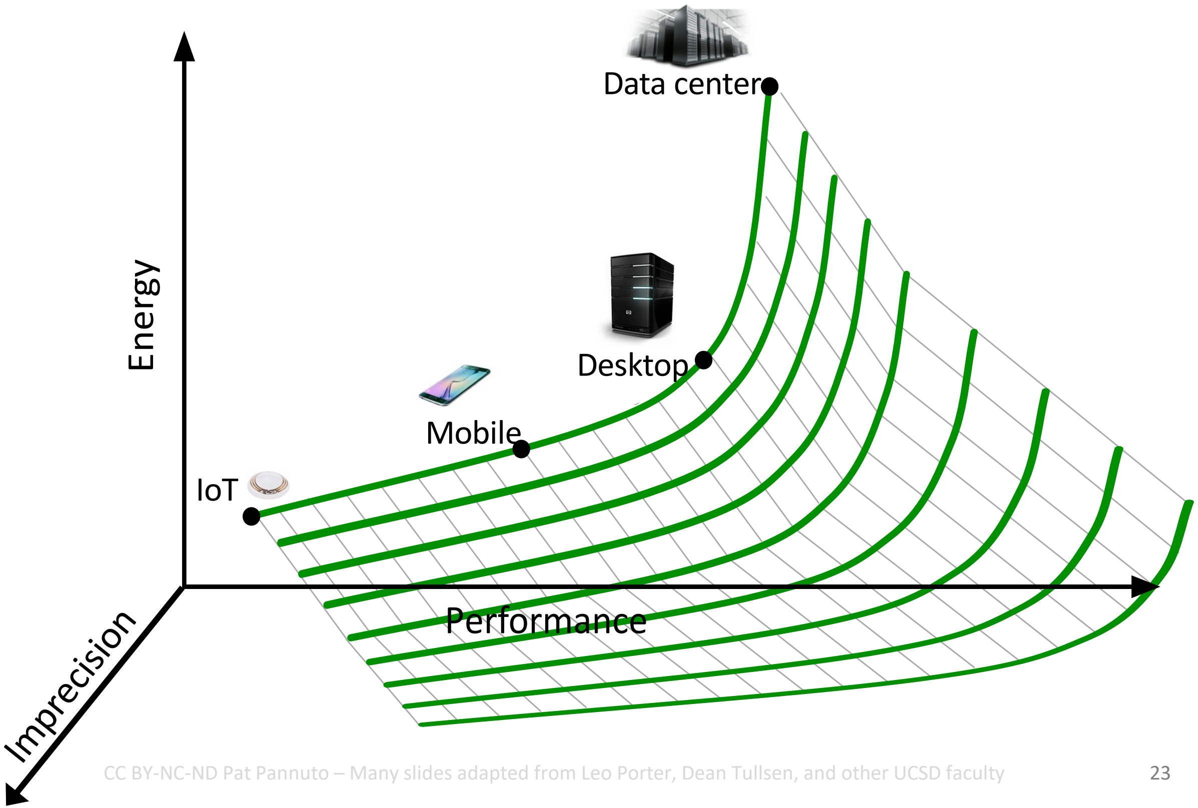


One wild idea: “Approximate Computing”

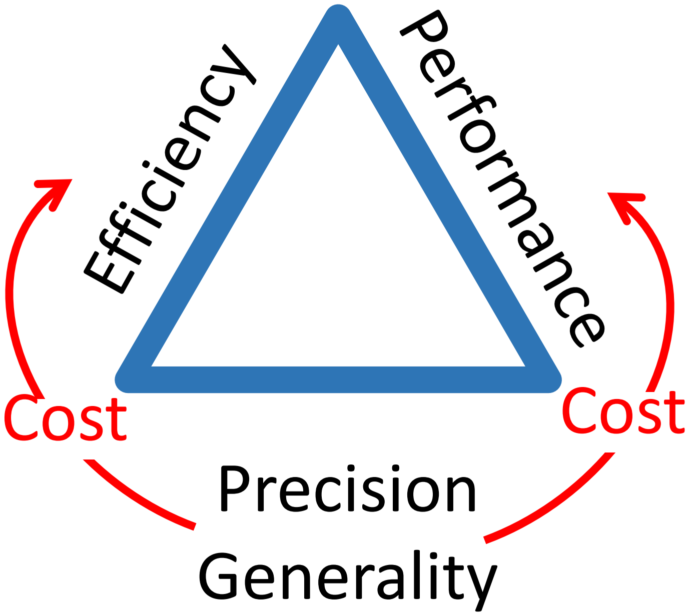
- Aka, what if 1 + 1 doesn’t *always* equal *exactly* 2?



Embracing imprecision allows for major gains in performance and energy



Rethinking the abstractions



Efficiency

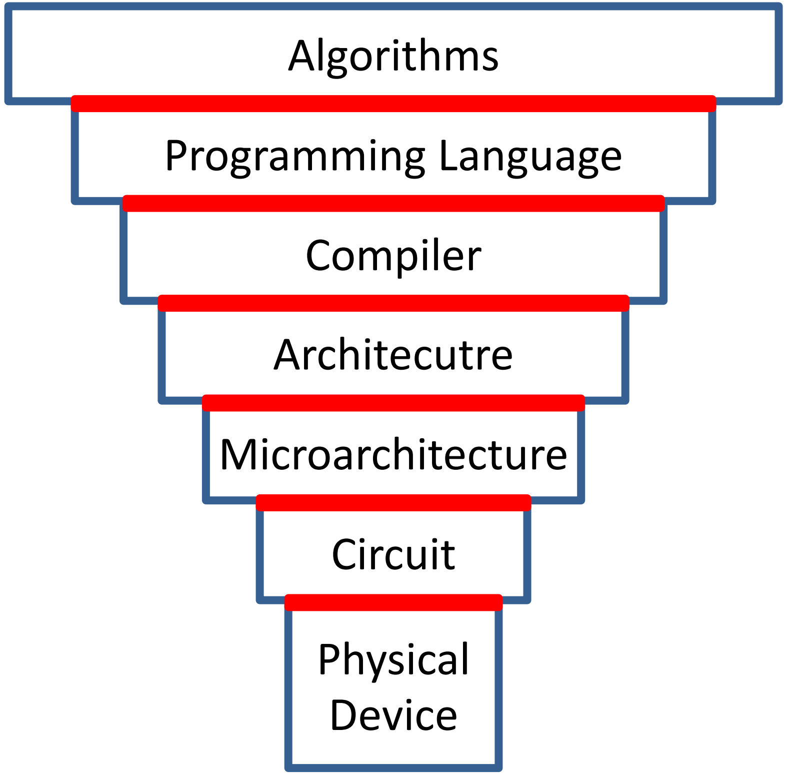
Performance

Cost

Cost

Precision

Generality



Algorithms

Programming Language

Compiler

Architecutre

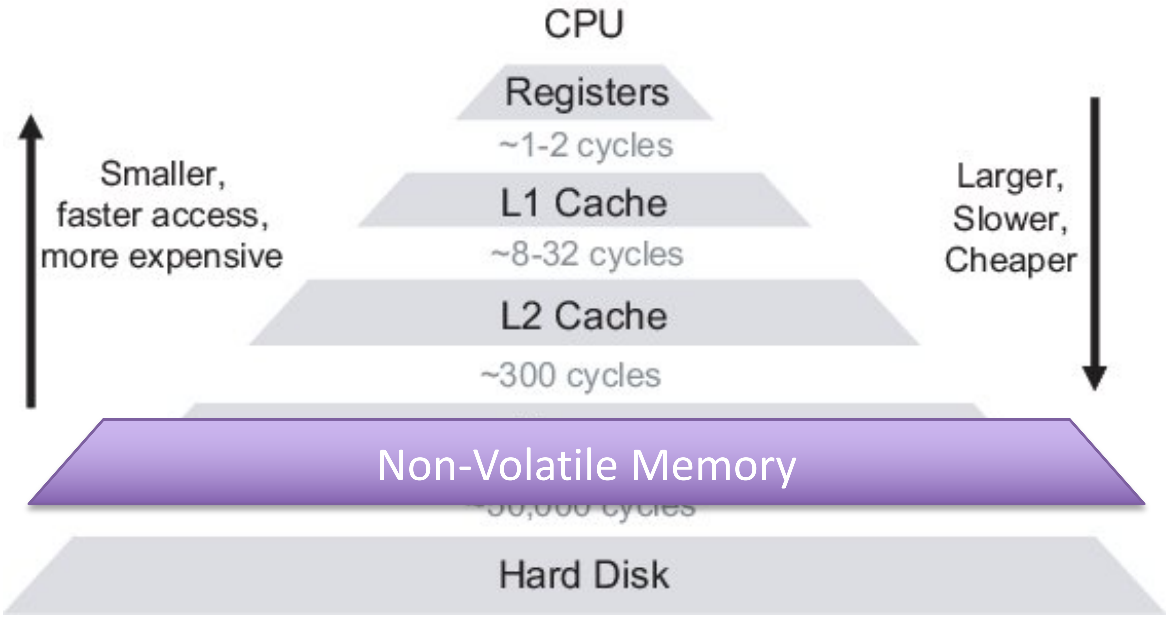
Microarchitecture

Circuit

Physical Device

Memory, Storage, Software, and Architecture in the NVSL

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



CPU

Registers

~1-2 cycles

L1 Cache

~8-32 cycles

L2 Cache

~300 cycles

Non-Volatile Memory

~50,000 cycles

Hard Disk

Smaller, faster access, more expensive

Larger, Slower, Cheaper

Chellappa, Srinivas & Franchetti, Franz & Püschel, Markus. (2007). How to Write Fast Numerical Code: A Small Introduction.

Applications

Tools

Libraries

Languages

Operating Systems

Distributed Systems

Hardware

MARS

Willow

NOVA

Orion

Ziggurat

SubZero

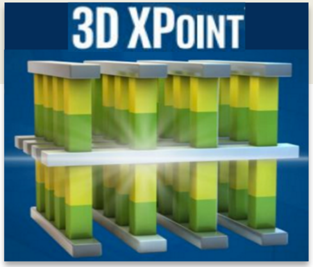
NV-Heaps

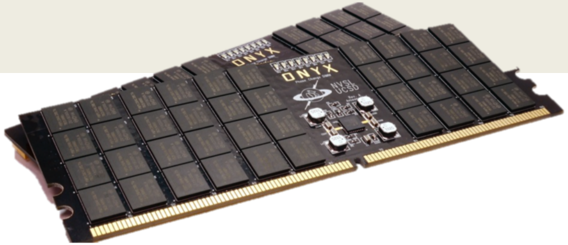
Pangolin

Pronto

Moneta

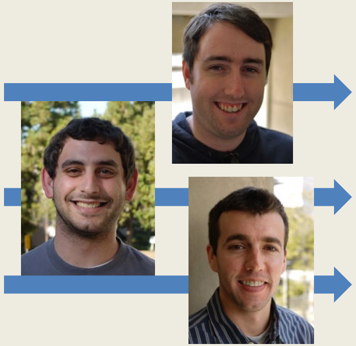
QuickSAN





NVSL Students Lead Industry

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

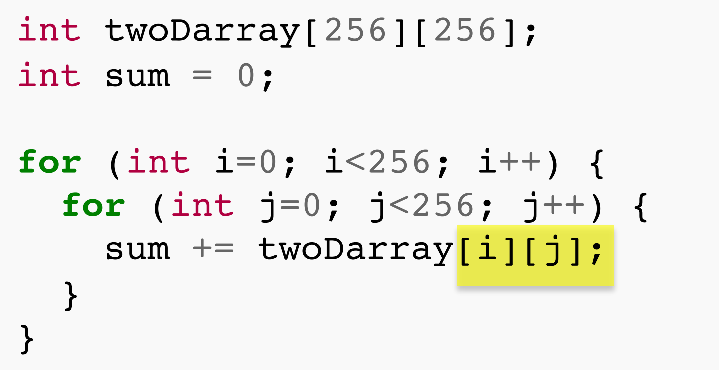


- Industry Built
  - NVMe (2011)
  - NVMe over Fabrics (2016)
  - Optane (2016)
  - PMDK (~2014)

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

Leaky abstractions are not always just performance problems…

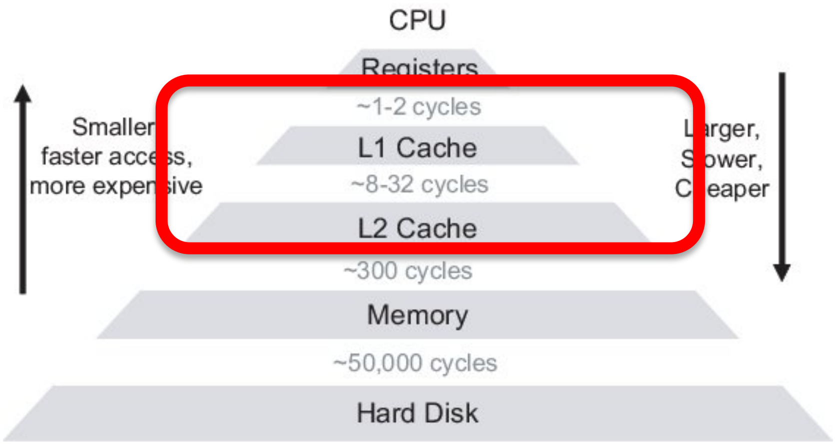
- This loop behaved differently because of how caches work



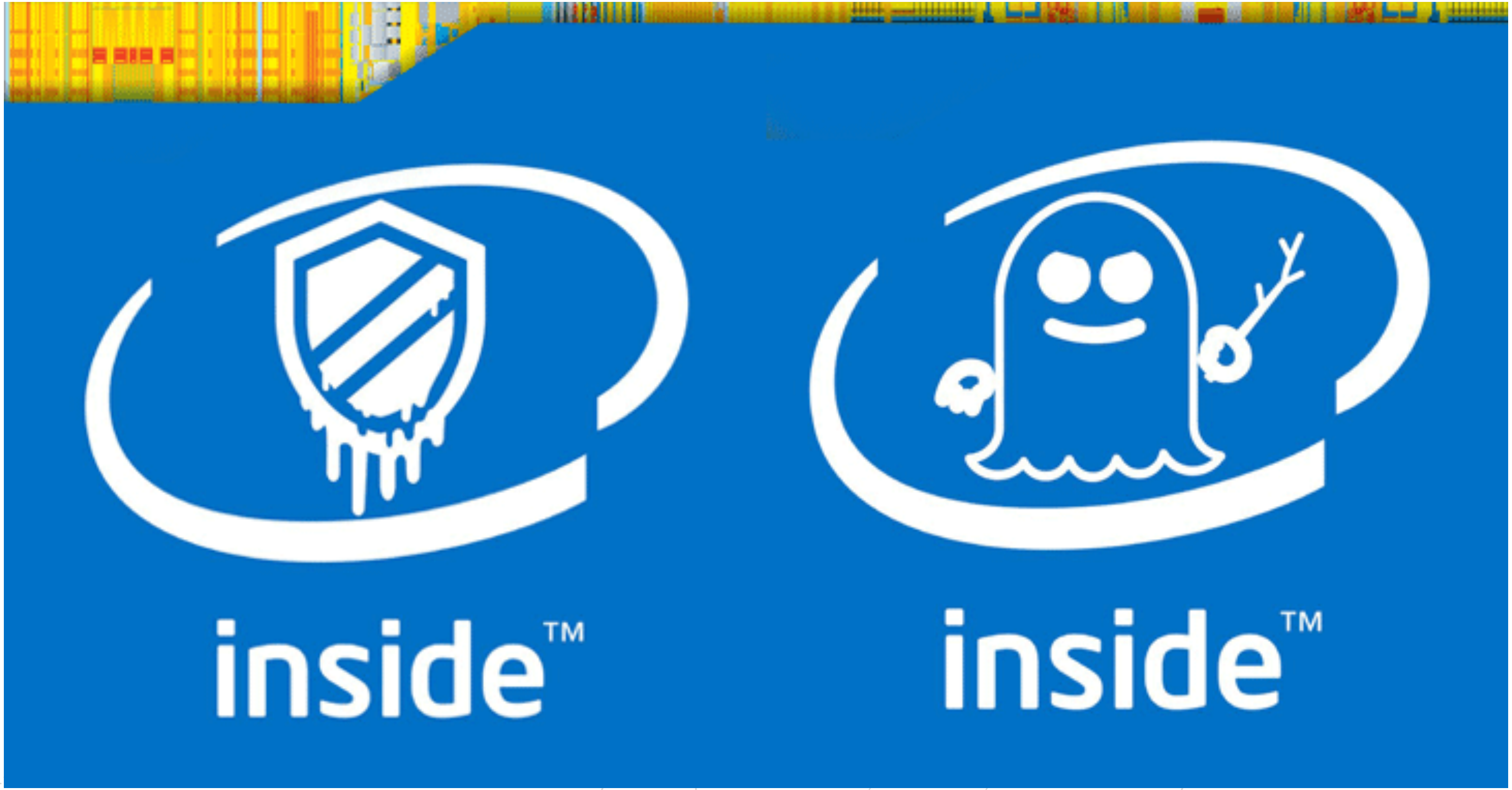
```
int twoDarray[256][256];
int sum = 0;

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

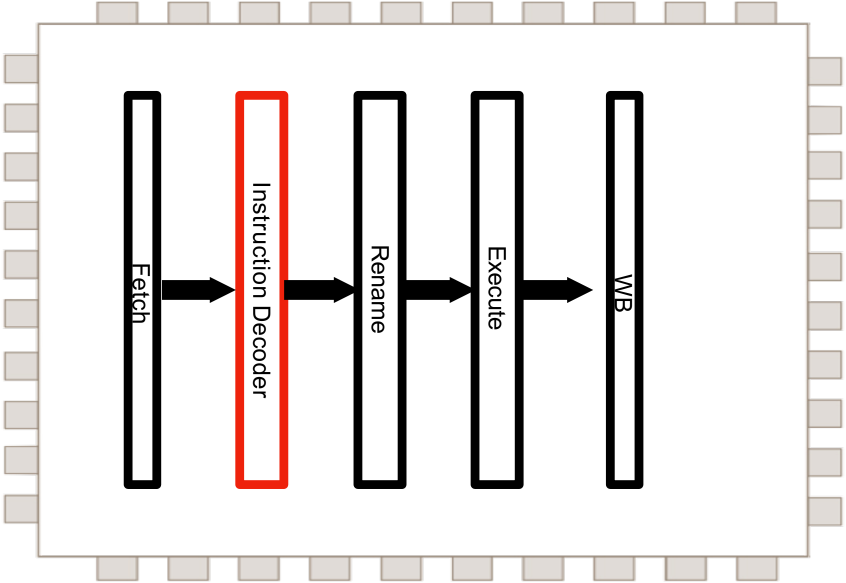


Leaky abstractions can be security threats!



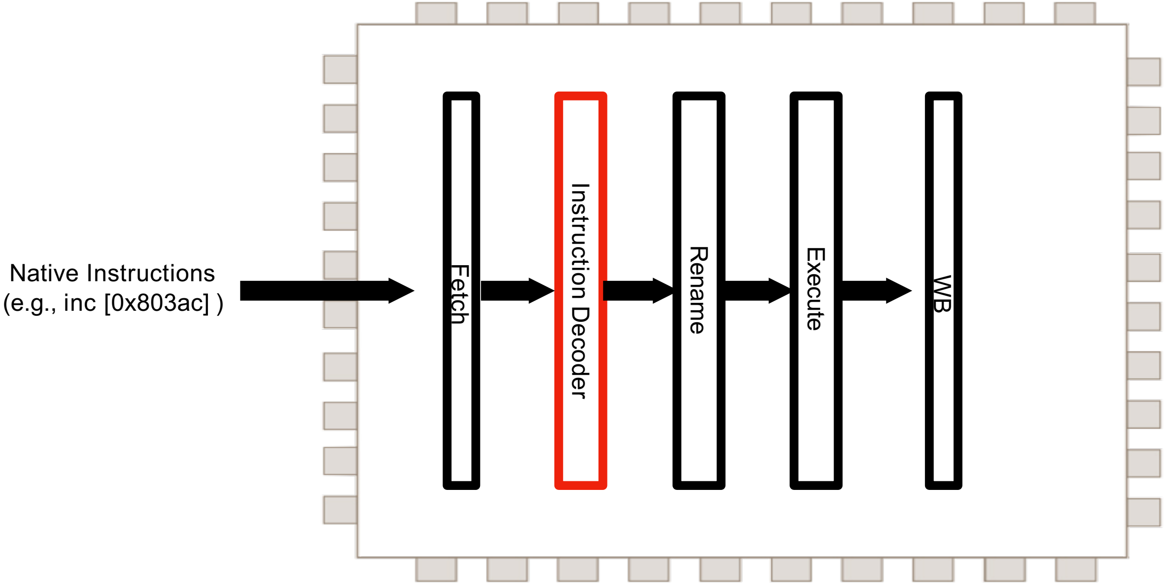
Mobilizing the Micro-Ops

Exploiting Translated ISAs



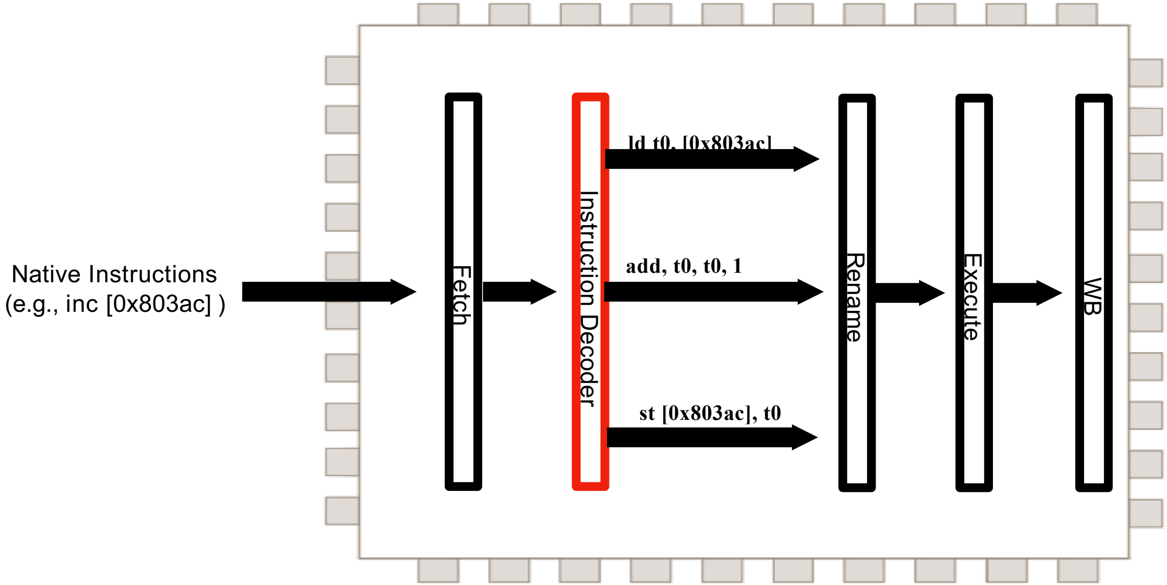
Mobilizing the Micro-Ops

Exploiting Translated ISAs



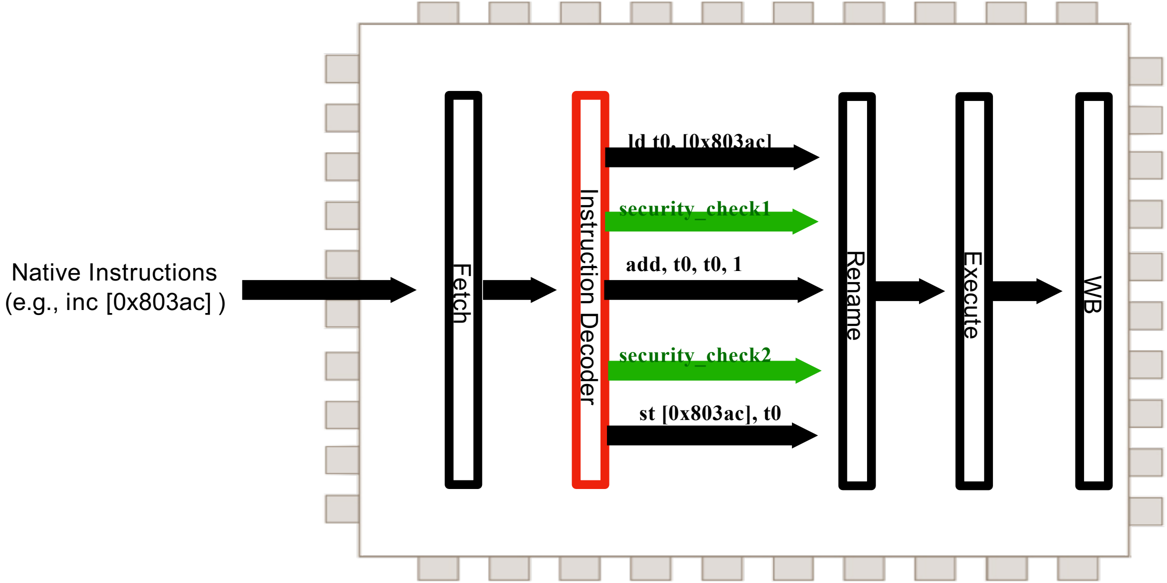
Mobilizing the Micro-Ops

Exploiting Translated ISAs



Mobilizing the Micro-Ops

Exploiting Translated ISAs



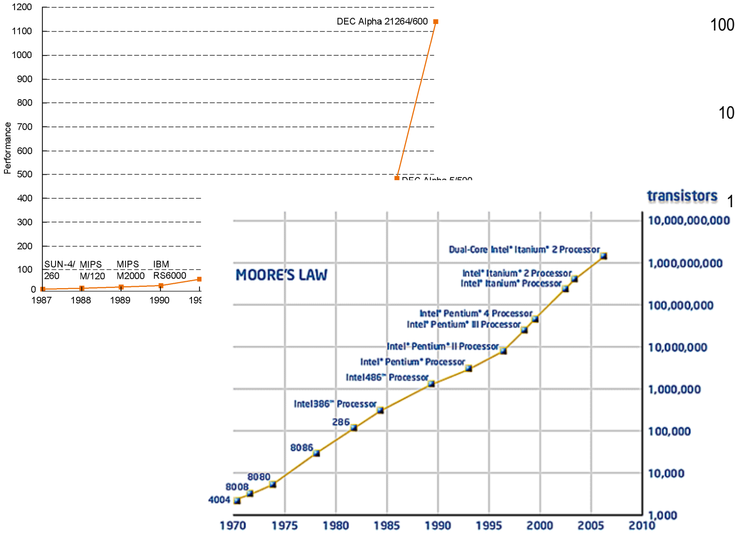
Context Sensitive Decoding fixes a leaky abstraction

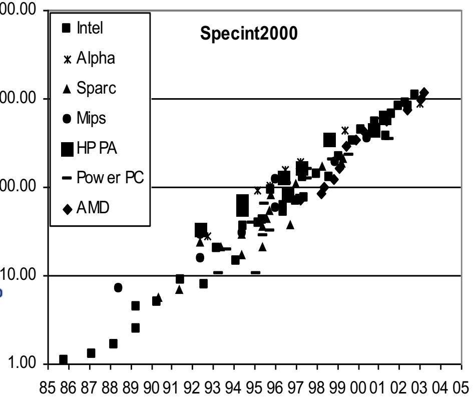
- Eliminating cache side channels via cache obfuscation
- Energy and Performance optimization via selective devectorization
  - ISCA 2018
  - IEEE Micro Top Picks in Computer Architecture

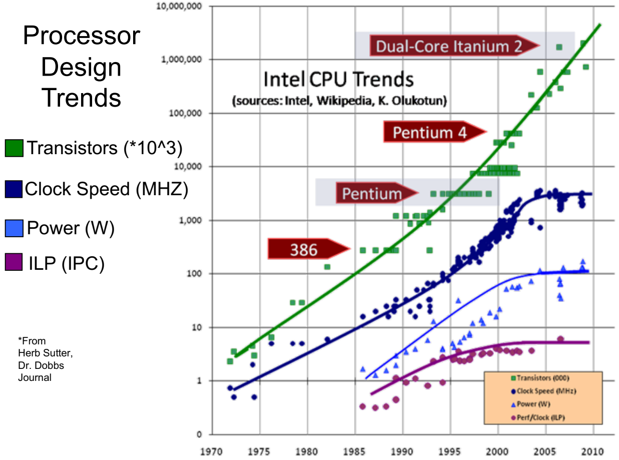
- Spectre mitigation via targeted insertion of fence micro-ops (Context Sensitive Fencing)
  - ASPLOS 2019

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

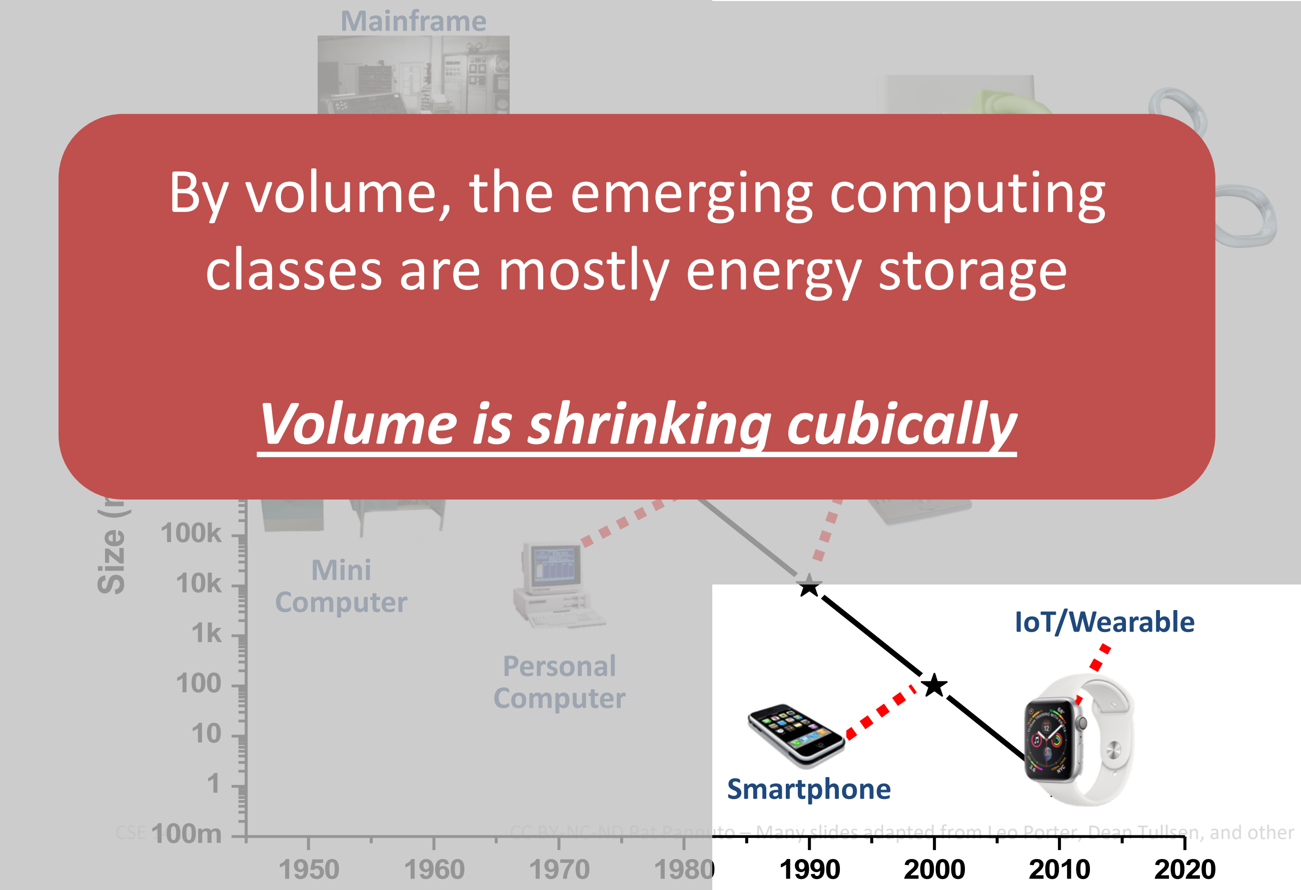





Processor Design Trends

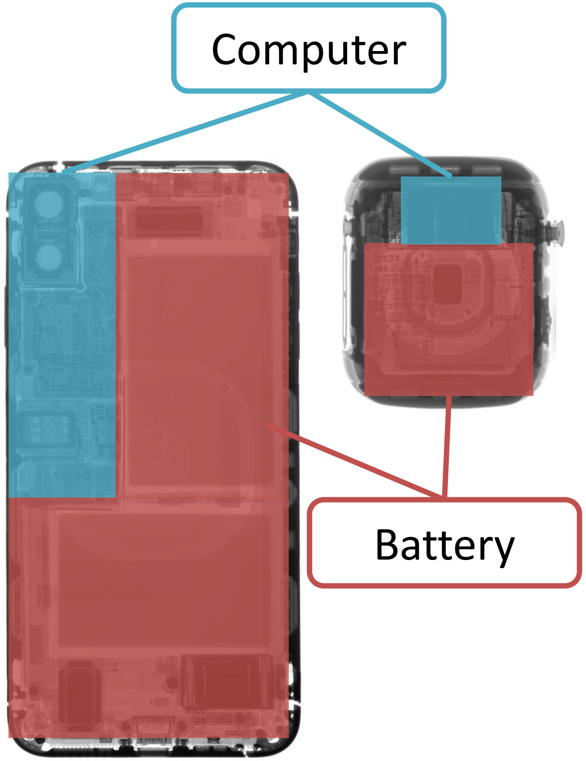
*From Herb Sutter, Dr. Dobbs Journal

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

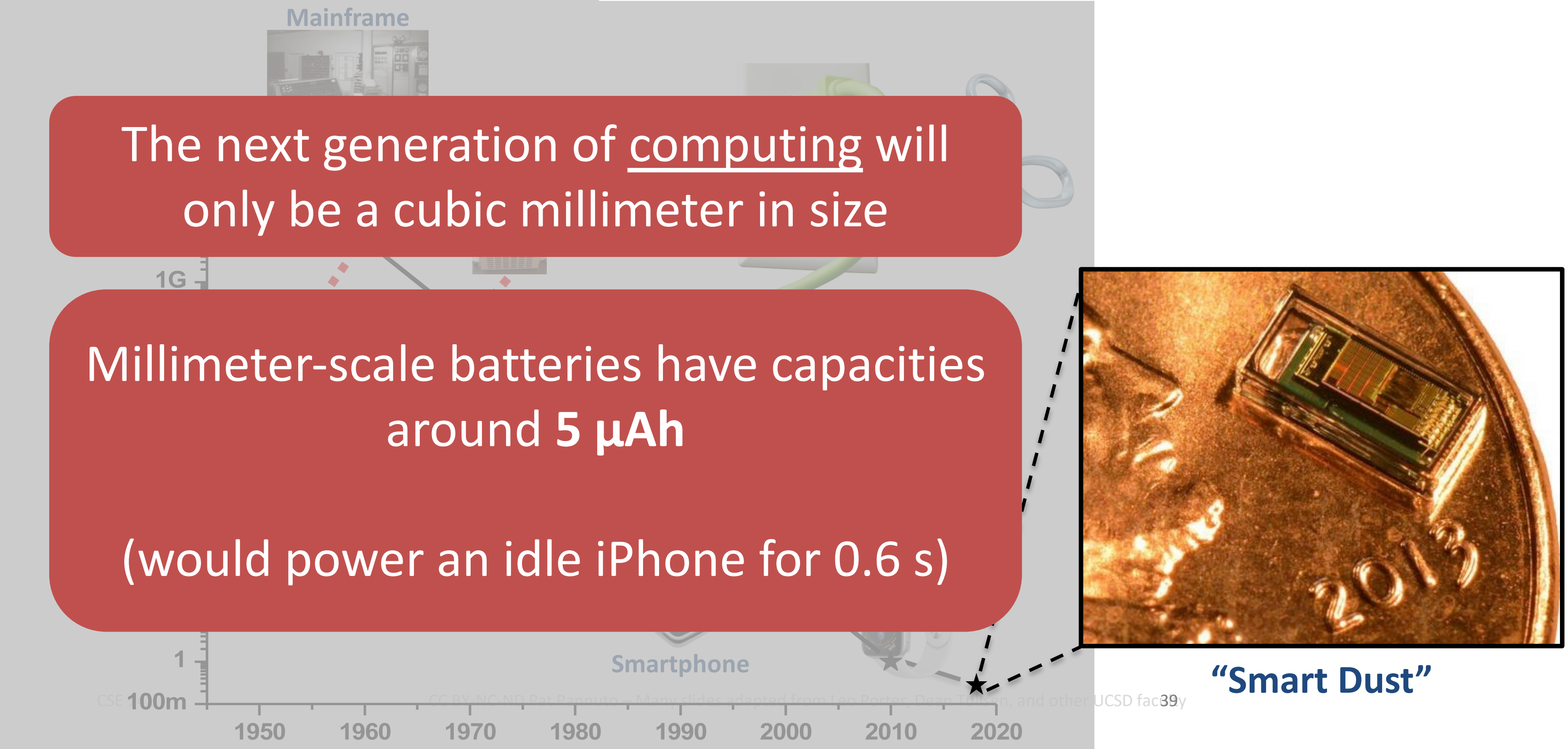


By volume, the emerging computing classes are mostly energy storage

Volume is shrinking cubically



Computational platforms will continue to scale



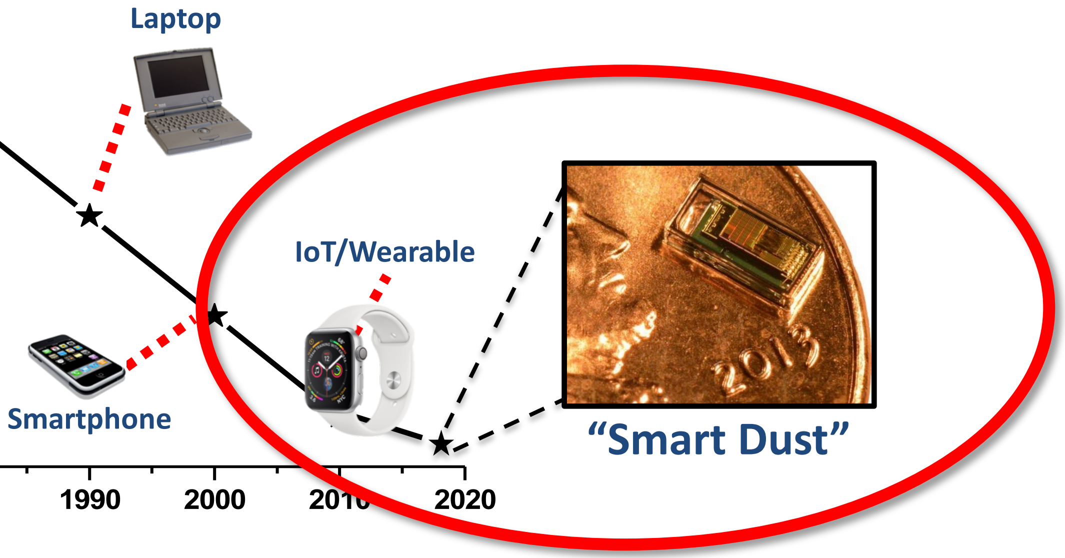
The next generation of computing will only be a cubic millimeter in size

Millimeter-scale batteries have capacities around **5 µAh**

(would power an idle iPhone for 0.6 s)

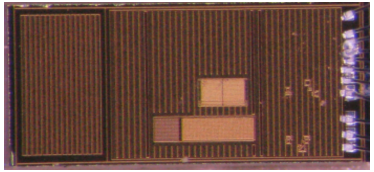
“Smart Dust”

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

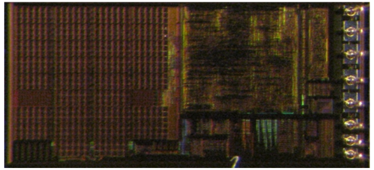


How must traditional paradigms change, adapt, or re-invent for the new computing classes?

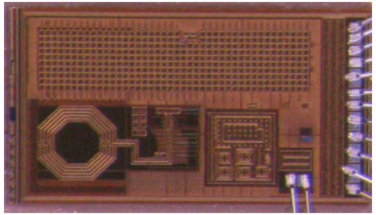
One of the first challenges was re-thinking how we put together computers



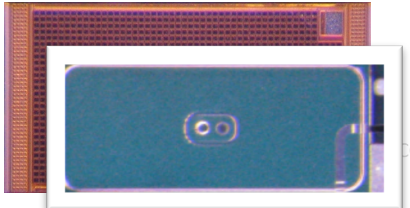
Temperature Sensor
~10 pW standby, < 1 µW active



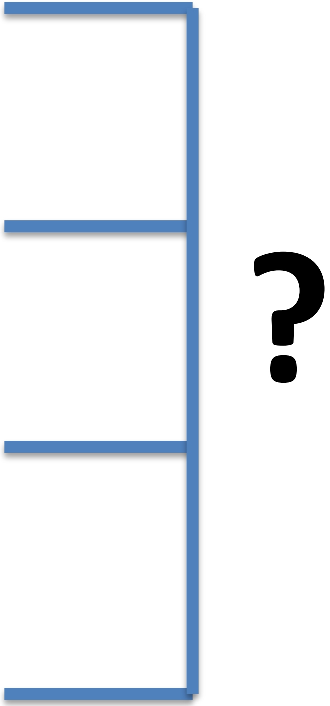
CPU
~1 nW standby, ~5 µW active



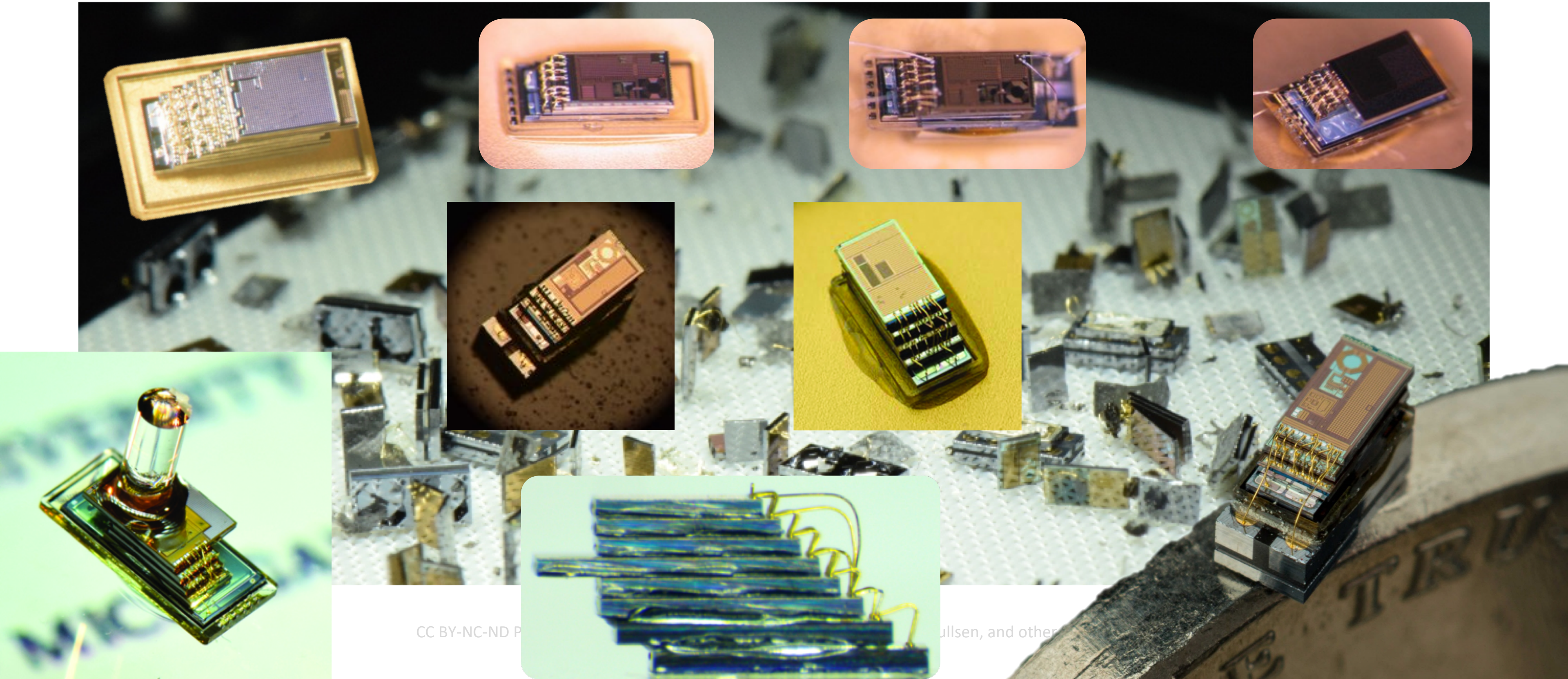
Radio
~10 pW standby, ~10 µW active



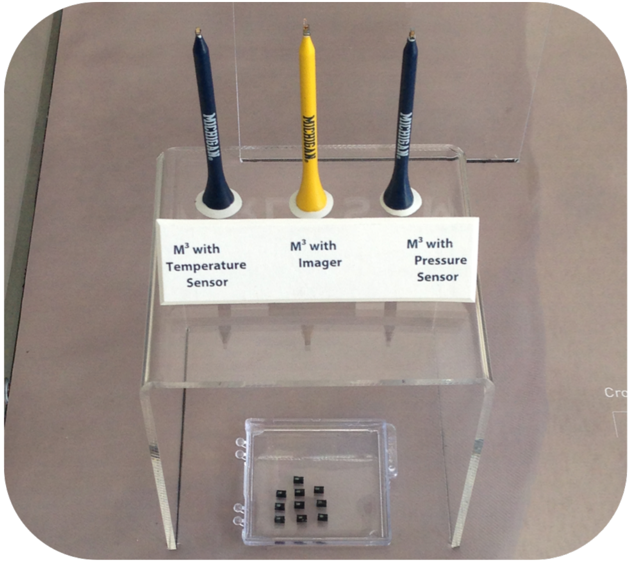
Energy Harvesting & Storage
1~10 nW indoors
2~10 µAh capacity

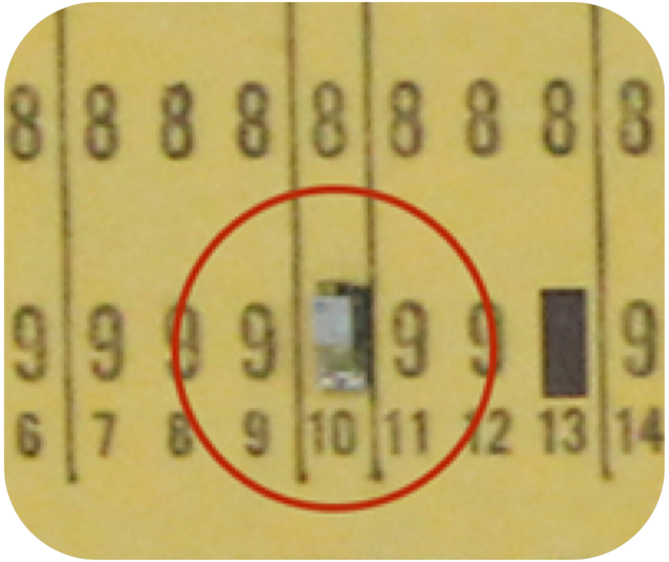


MBus enabled the of development of dozens of millimeter-scale motes as part of the Michigan Micro Mote (M3) project



Check out the "World's Smallest Computer" exhibit at Silicon Valley's Computer History Museum!

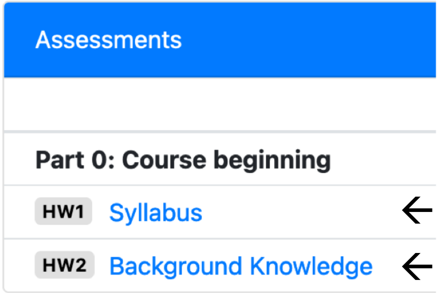




Next up: Instruction Set Architectures (ISAs)

- Reading:
  - Skim 1.1 [7 pages]
  - Read 1.2, 1.3 [6.5 pages]
- Okay if not until Oct 2:
  - Skim 2.1-2.2 [5 pages]
  - Read 2.3-2.5 [16 pages]
  - Skim 2.10 [10 pages]

Out now! Administrative “HW”



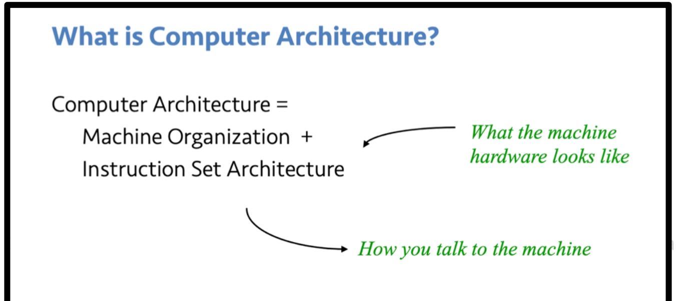
Assessments

Part 0: Course beginning

HW1 Syllabus ← Due: Thr, 14:00

HW2 Background Knowledge ← Due: Thr, 14:00

Should be quick — don’t delay, do today!



What is Computer Architecture?

Computer Architecture =
Machine Organization +
Instruction Set Architecture

What the machine hardware looks like

How you talk to the machine

ny slides adapted from Leo Porter, Dean Tullsen, and other UCSD faculty