CS 110
Computer Architecture I
Introduction

Instructors:
Siting Liu and Yuan Xiao
Course website: https://faculty.sist.shanghaitech.edu.cn/liust/courses/CS110.html
School of Information Science and Technology (SIST)
ShanghaiTech University

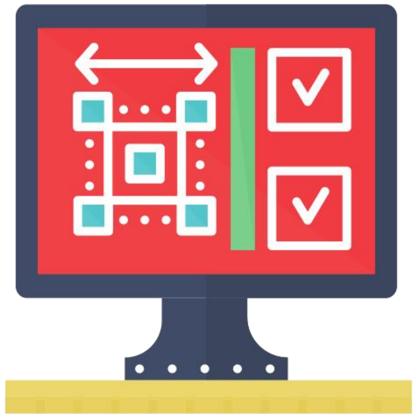
2026/3/3

Outline

- History and backgrounds
  - Recent trends
- Course content
- Course rules/elements
- Great ideas in computer architecture

What is a computer?

- A computer is a machine that can be programmed to carry out sequences of arithmetic or logical operations (computation) automatically. Modern digital electronic computers can perform generic sets of operations known as programs. These programs enable computers to perform a wide range of tasks.
- A computer system is a nominally complete computer that includes the hardware, operating system (main software), and peripheral equipment needed and used for full operation. This term may also refer to a group of computers that are linked and function together, such as a computer network or computer cluster.



From Wikipedia

Computer History

- Early computers (calculators)

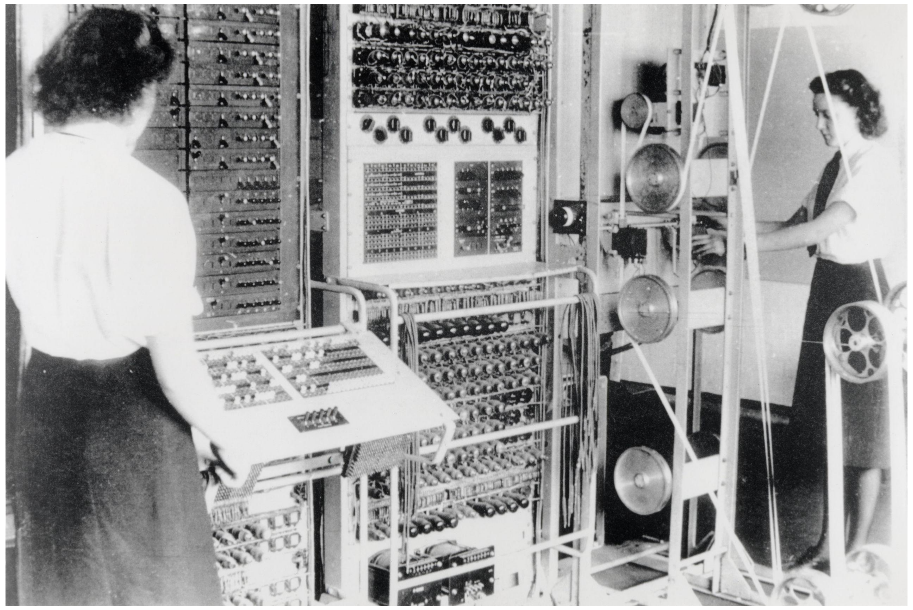
Chinese abacus

By Loadmaster (David R. Tribble). This image was made by Loadmaster (David R. Tribble). CC BY-SA 3.0.

Automatic mechanical calculator (1820)

By Jitze Couperus from Los Altos Hills, California, USA. Uploaded by oxyman, CC BY 2.0.

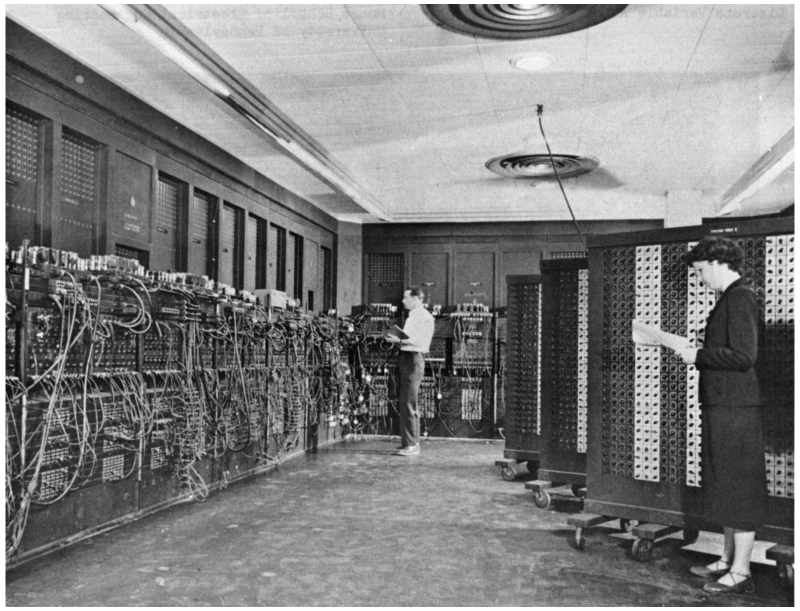
Colossus (WWII, 1943-1945)



This file is from the collections of The National Archives (United Kingdom), catalogued under document record FO850/234.

- Programmed by manually setting plugs and switches
- Vacuum tubes

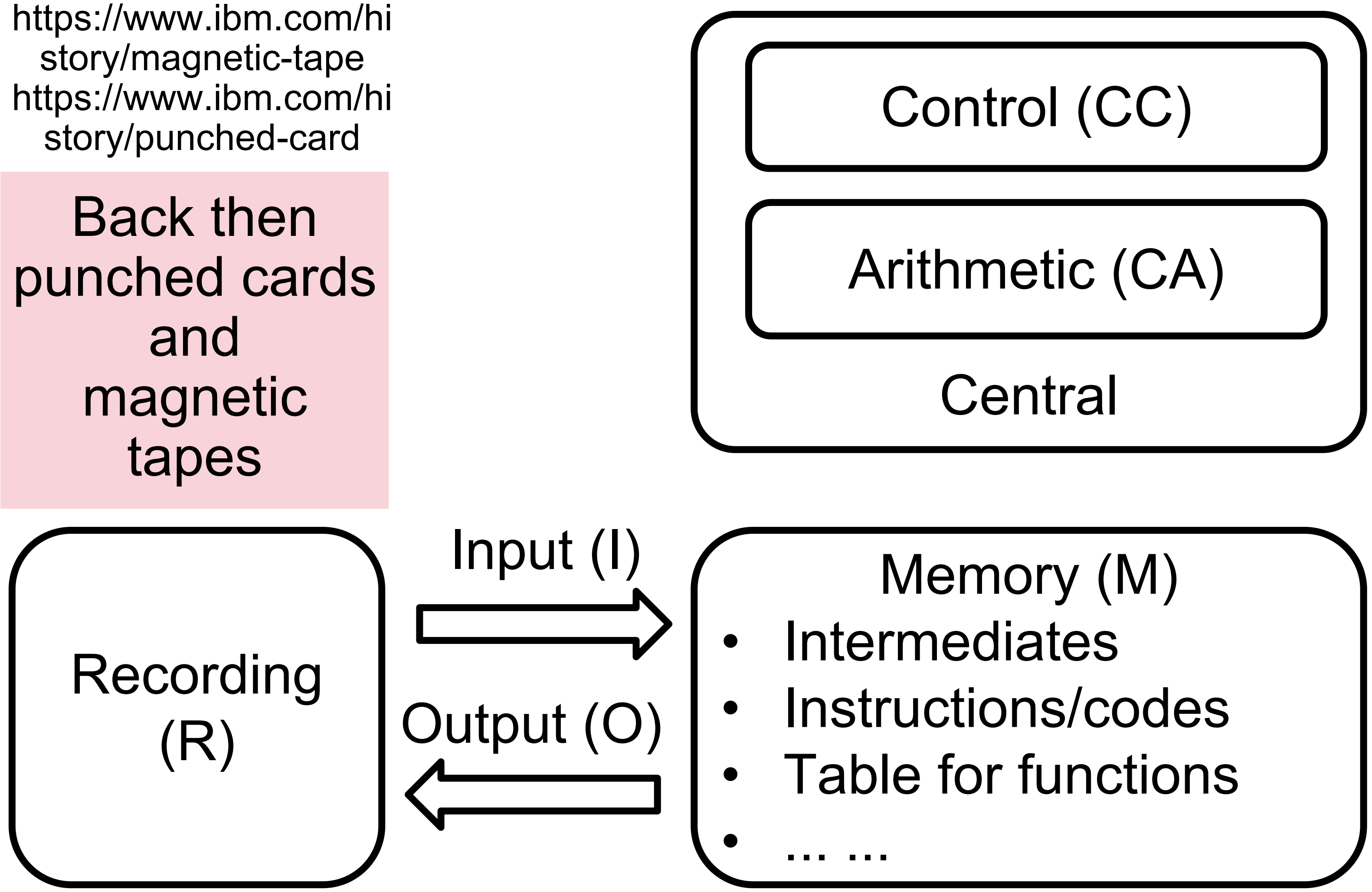
ENIAC (1945-1946)



- 5000 additions or subtractions per second
- Modules to multiply, divide, and square root
- 30 tons, 200 kW, 18,000 vacuum tubes

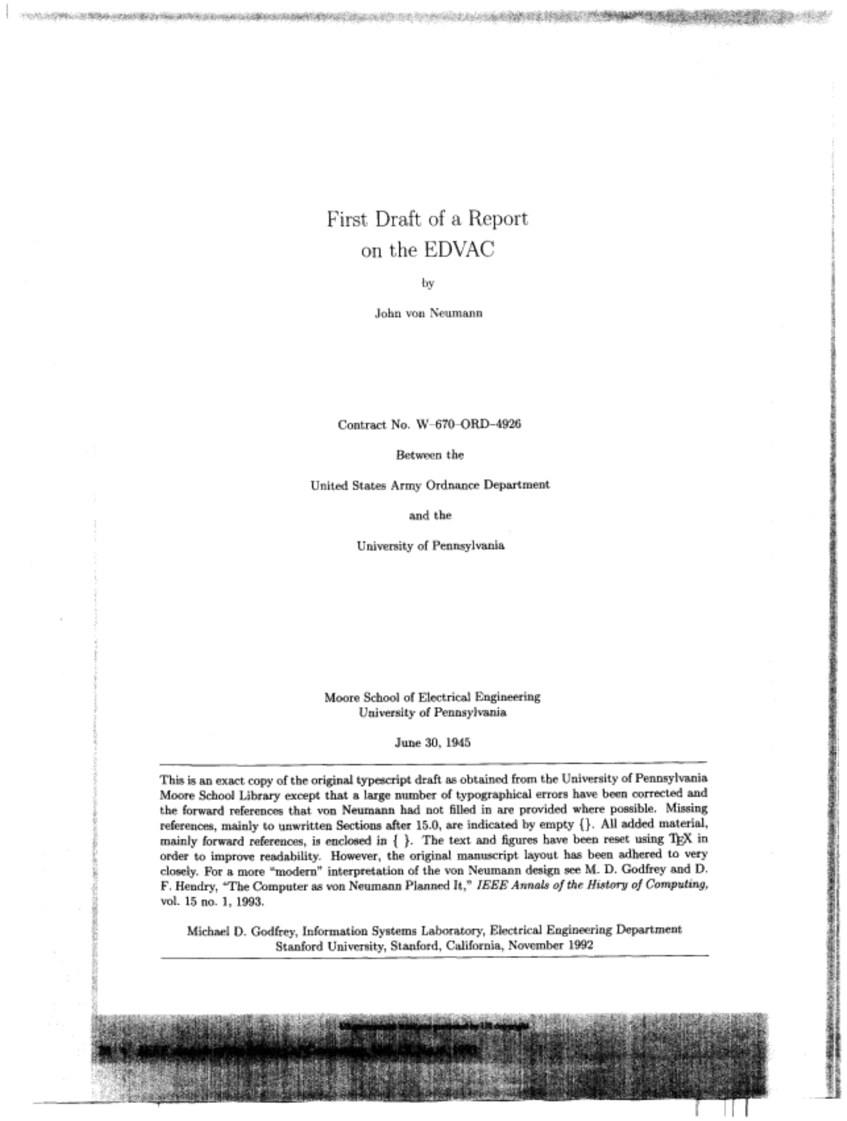
Computer History

- Evolution of theory
  - Alan Turing proposed theoretical basis for the stored-program computer (1936)
  - *The First Draft of a Report on the EDVAC* by Von Neumann (1945)
  - Konrad Zuse suggested and implemented similar ideas (Harvard architecture)



https://www.ibm.com/history/magnetic-tape
https://www.ibm.com/history/punched-card

Back then punched cards and magnetic tapes



First Draft of a Report
on the EDVAC

by

John von Neumann

Contract No. W–670–ORD–4926

Between the

United States Army Ordnance Department

and the

University of Pennsylvania

Moore School of Electrical Engineering
University of Pennsylvania

June 30, 1945

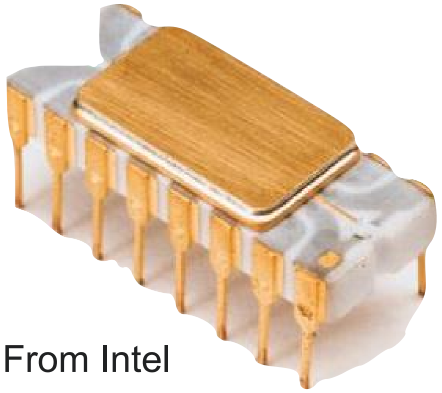
This is an exact copy of the original typescript draft as obtained from the University of Pennsylvania Moore School Library except that a large number of typographical errors have been corrected and the forward references that von Neumann had not filled in are provided where possible. Missing references, mainly to unwritten Sections after 15.0, are indicated by empty {}. All added material, mainly forward references, is enclosed in { }. The text and figures have been reset using TEX in order to improve readability. However, the original manuscript layout has been adhered to very closely. For a more "modern" interpretation of the von Neumann design see M. D. Godfrey and D. F. Hendry, "The Computer as von Neumann Planned It," *IEEE Annals of the History of Computing*, vol. 15 no. 1, 1993.

Michael D. Godfrey, Information Systems Laboratory, Electrical Engineering Department
Stanford University, Stanford, California, November 1992

Computer History

- Evolution of theory
  - Alan Turing proposed theoretical basis for the stored-program computer (1936)
  - *The First Draft of a Report on the EDVAC* by Von Neumann (1945)
  - Konrad Zuse suggested and implemented similar ideas (Harvard architecture)
- Evolution of hardware & devices
  - Random-access digital storage (Williams tube, 1947)
  - Metal-oxide-silicon field-effect transistor (MOSFET, 1970-ish)
  - Integrated circuits (ICs) and then VLSI
  - Too many to list …
- Microprocessor CPU (Intel 4004, 1971)

The first delivery of a fully operational 4004 was in March 1971 to Busicom for its 141-PF printing calculator engineering prototype (now displayed in the Computer History Museum in Mountain View, California).



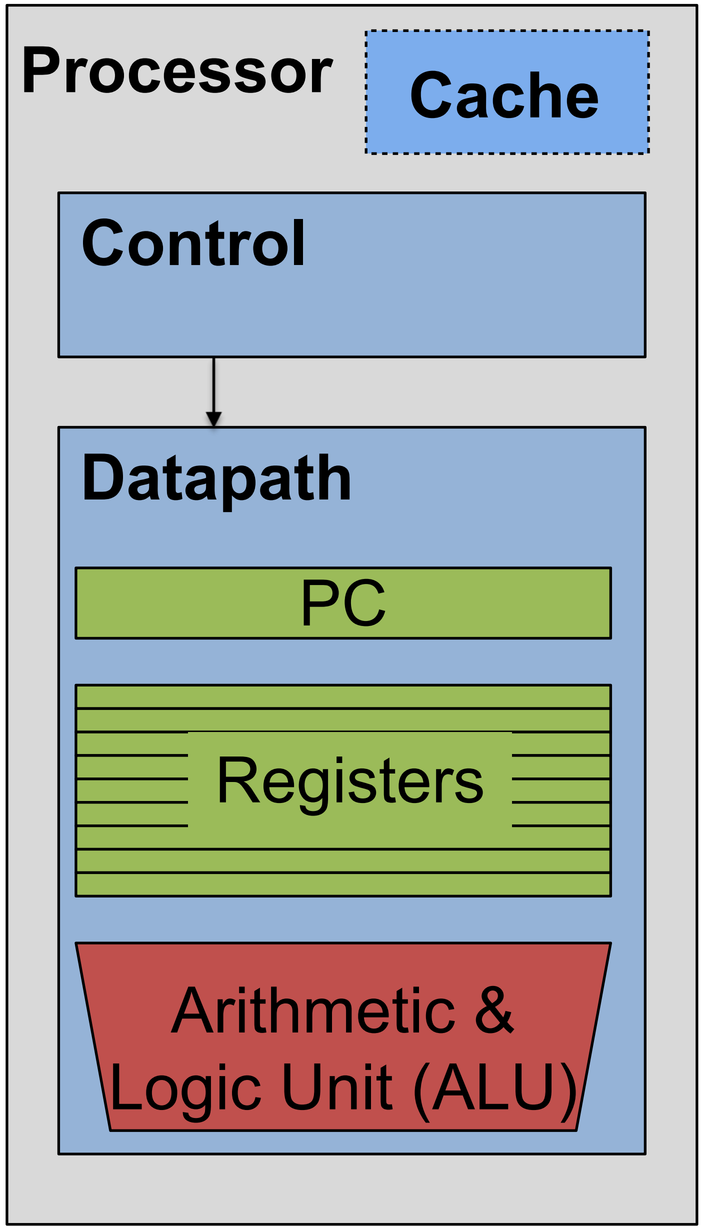
From Intel

Modern Computers

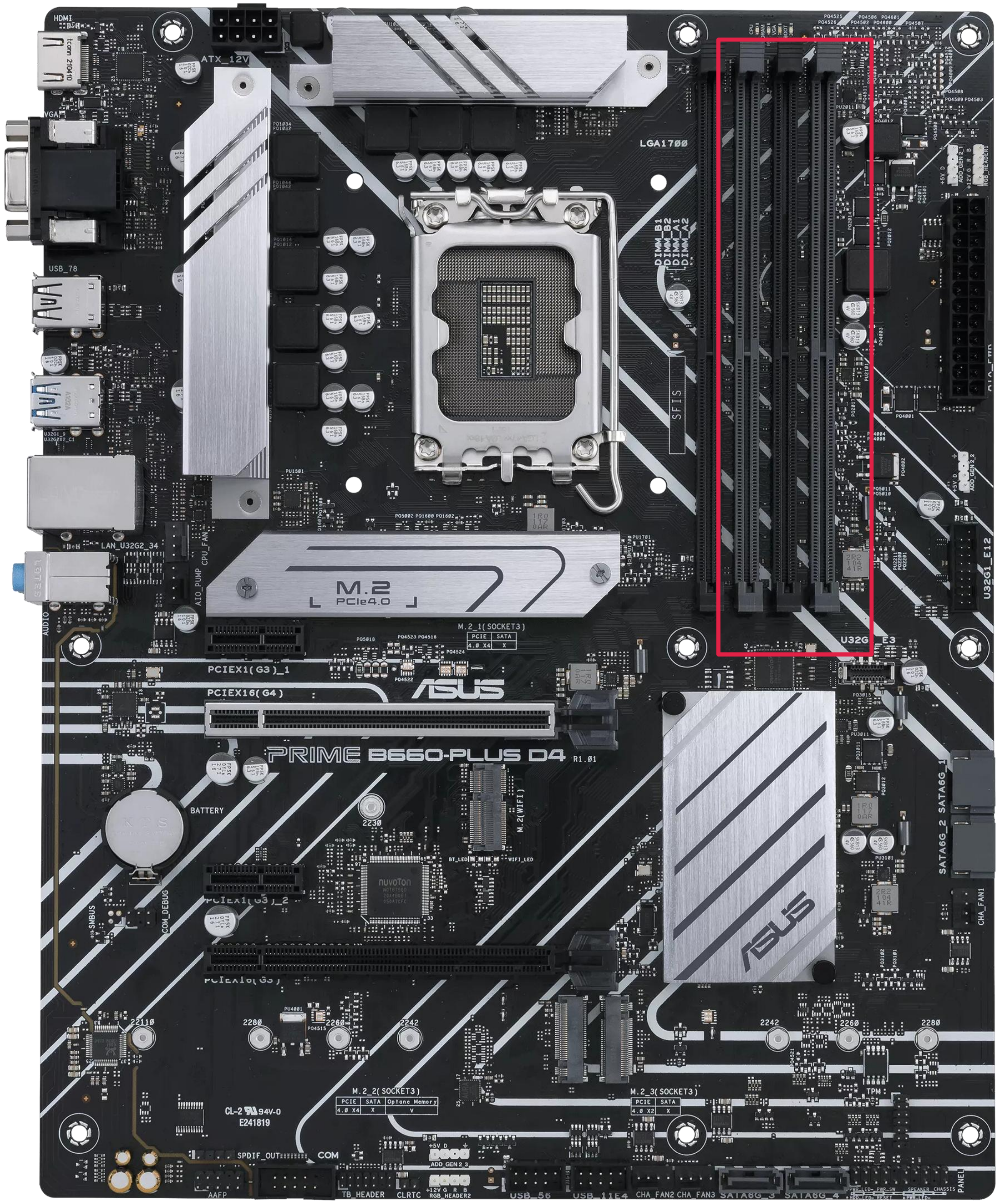
Intel i7 12700

https://maj191.com/product/intel-core-i7-12700-3-6ghz-cpu-25mb-cache-lga1700-tray/

https://www.ocinside.de/review/intel_core_i7_12700k/3/



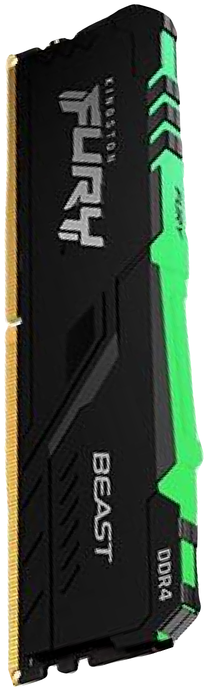
PCB



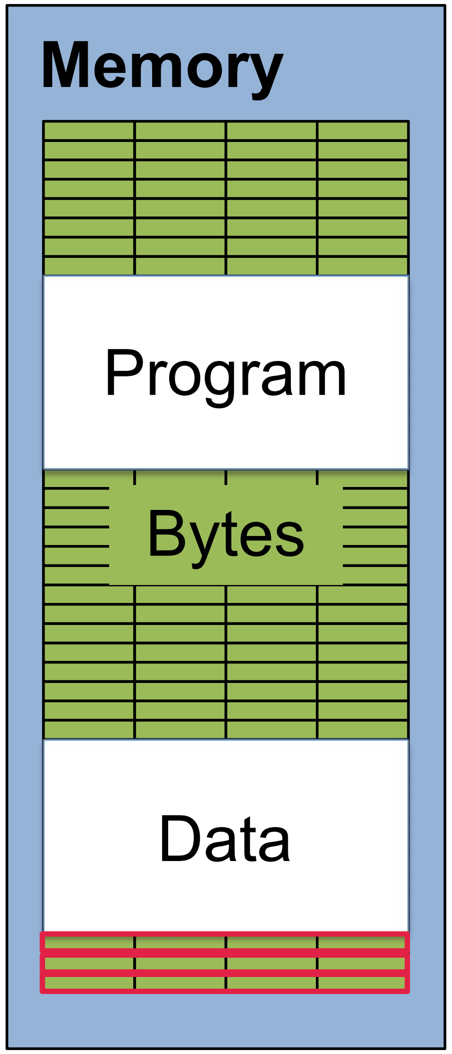
https://www.asus.com/motherboards-components/motherboards/prime/prime-b660-plus-d4/

DIMM

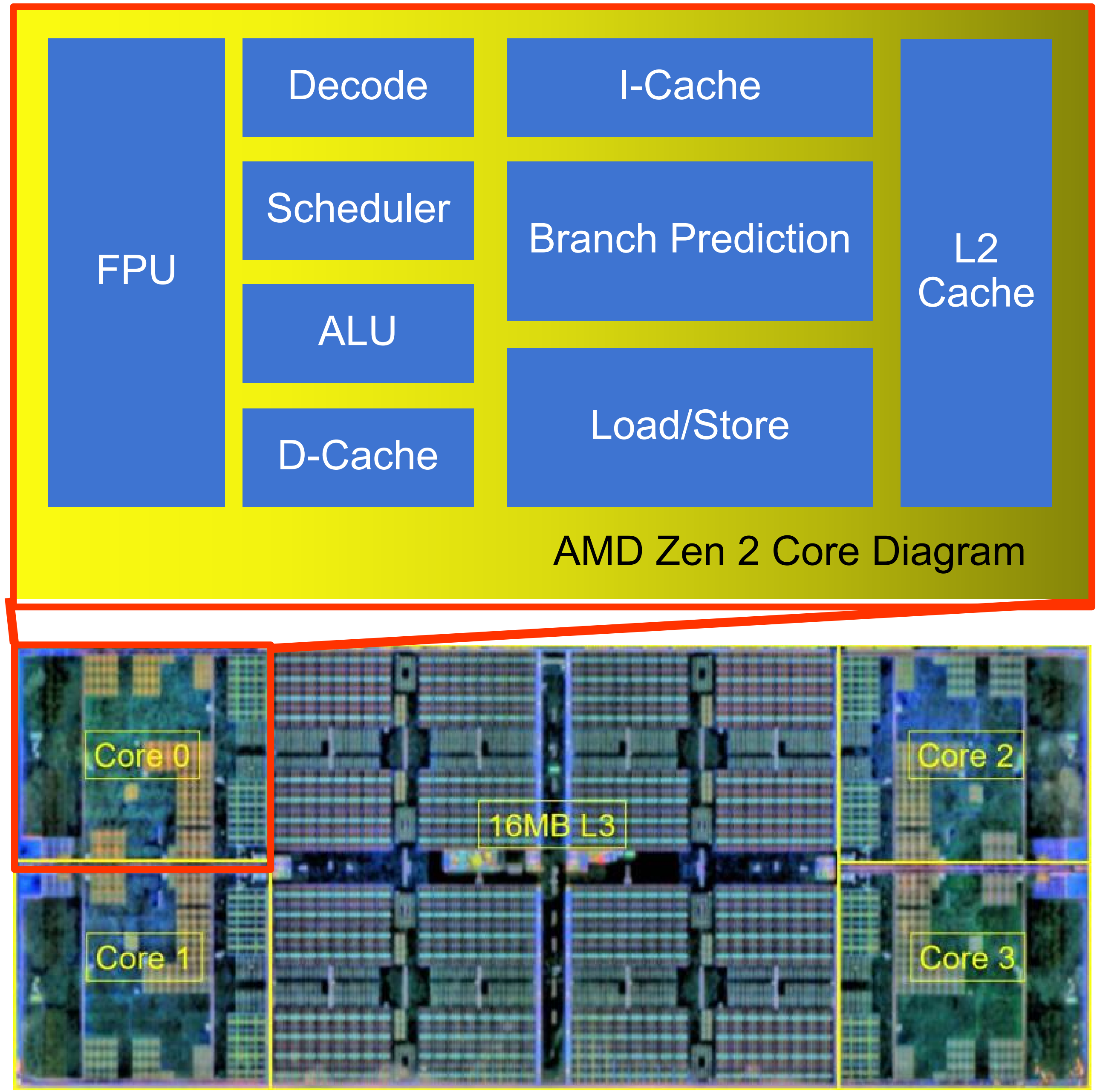
DDR4
5066 MHz



SDRAM



Inside the CPU



FPU

Decode

Scheduler

ALU

D-Cache

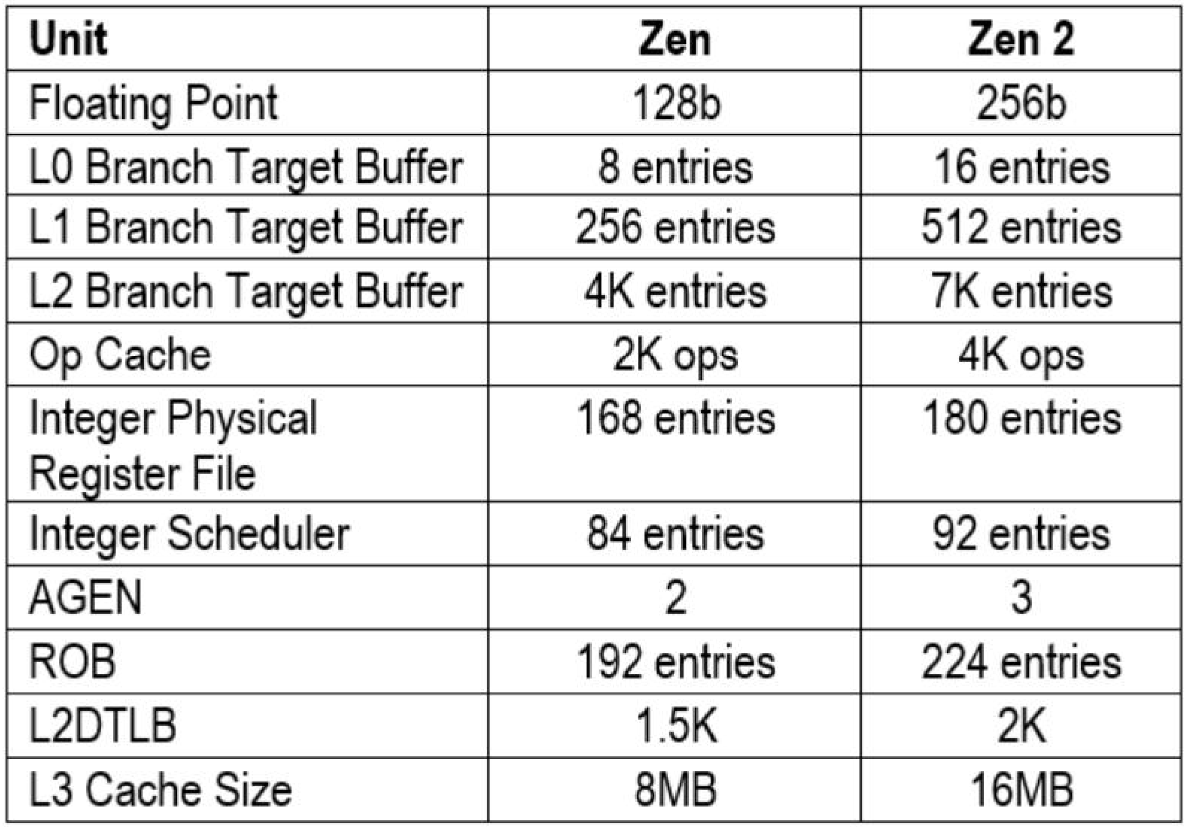
I-Cache

Branch Prediction

Load/Store

L2 Cache

AMD Zen 2 Core Diagram



Unit	Zen	Zen 2
Floating Point	128b	256b
L0 Branch Target Buffer	8 entries	16 entries
L1 Branch Target Buffer	256 entries	512 entries
L2 Branch Target Buffer	4K entries	7K entries
Op Cache	2K ops	4K ops
Integer Physical Register File	168 entries	180 entries
Integer Scheduler	84 entries	92 entries
AGEN	2	3
ROB	192 entries	224 entries
L2DTLB	1.5K	2K
L3 Cache Size	8MB	16MB

7.83 mm^2 per core

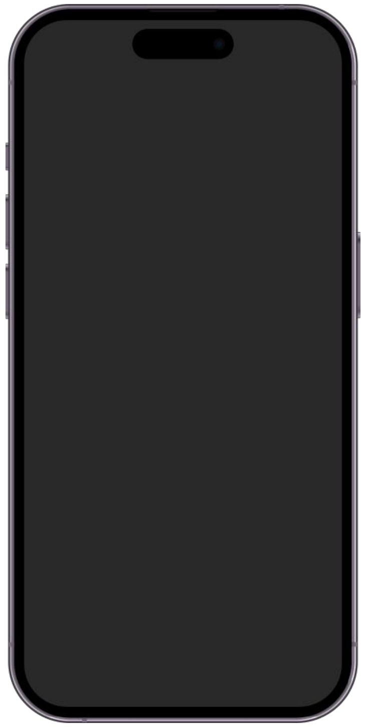
[1] T. Singh et al., "2.1 Zen 2: The AMD 7nm Energy-Efficient High-Performance x86-64 Microprocessor Core," IEEE International Solid- State Circuits Conference - (ISSCC), 2020, pp. 42-44.

Modern Computer Systems

From Huawei website

Inside an Intel Core i7 CPU

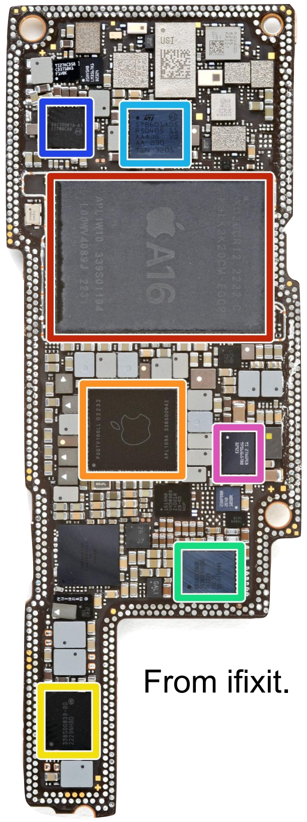
- 4 cores
- Up to 5.00 GHz
- Max 64 GB memory
- Include Intel Iris Xe Graphics (GPU)
- Windows/Linux (for desktop)



Inside an Apple A16 processor (SoC)

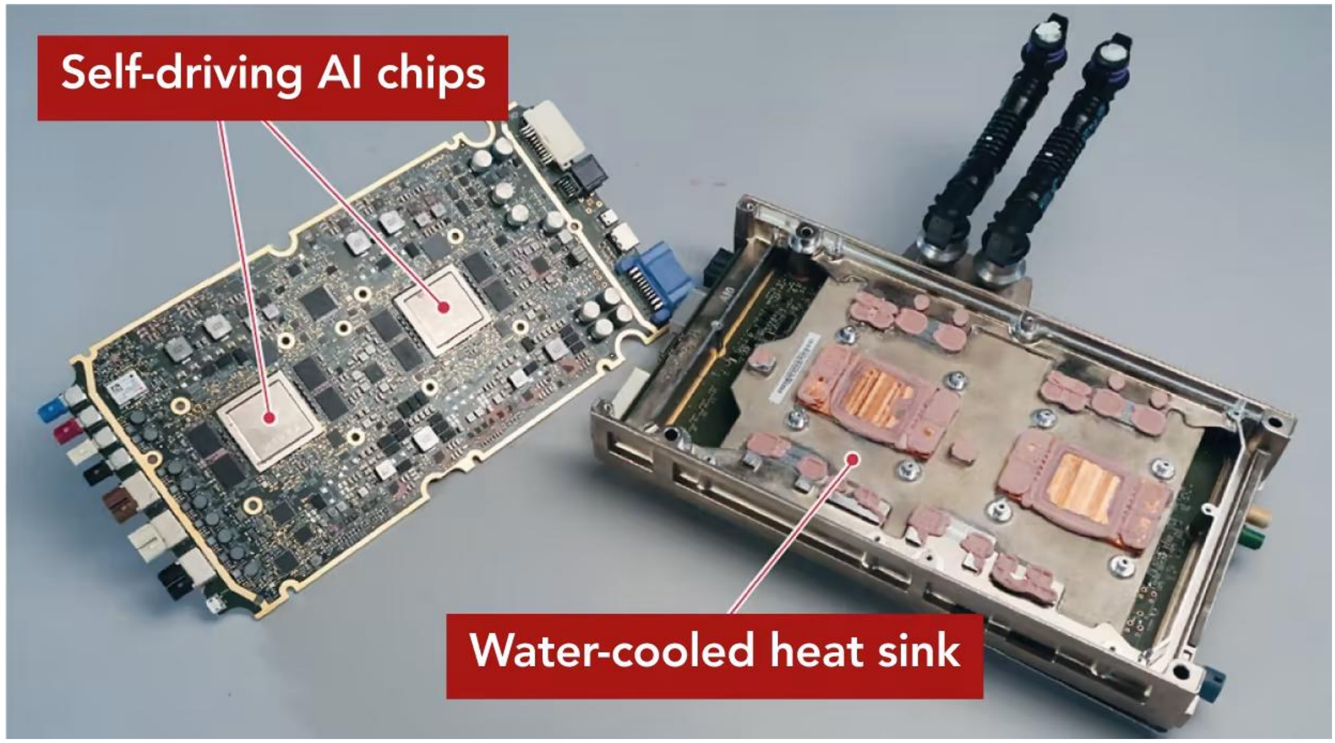
- 6 cores
- Up to 3.46 GHz
- 6 GB memory
- Include Apple-designed GPU, image processor, 16-core neural engine, etc.
- 17 TOPS on neural engine
- iOS

From Wikipedia



From ifixit.

Modern Computer (Embedded) Systems



FSD computer, inside a FSD chip (SoC)

- 12 CPU cores, 2.2 GHz
- Up to 8 GB memory
- Include a Mali GPU @ 1 GHz, 2 neural processing units @ 2 GHz, etc.
- GPU 600 GFLOPS, NPU 36.86 TOPS
- Specialized software

Main screen (infotainment) computer

- Ubuntu-based OS
- AMD Ryzen & GPU/Intel Atom

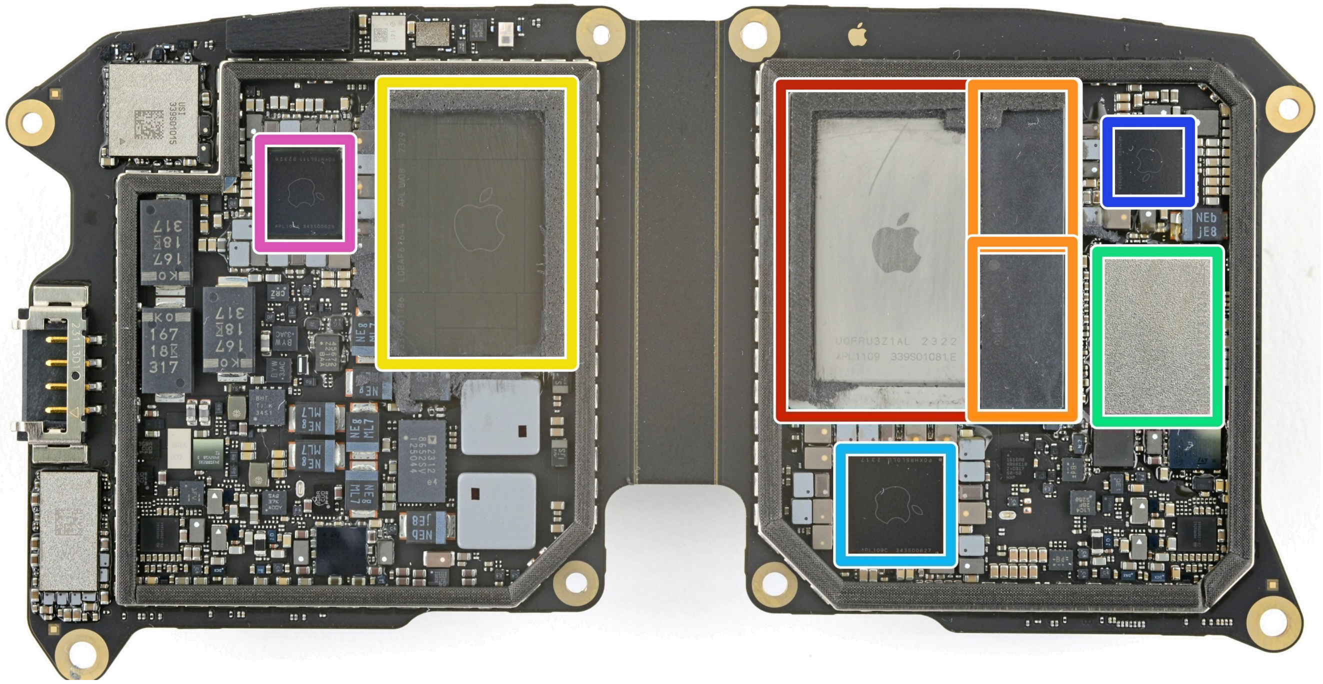
From Wikichip.

From Nikkei-xTech, Ingineerix & Dongchedi.
https://www.youtube.com/watch?v=ODdIRr5RzI8
https://www.dongchedi.com/article/7122294255204712972

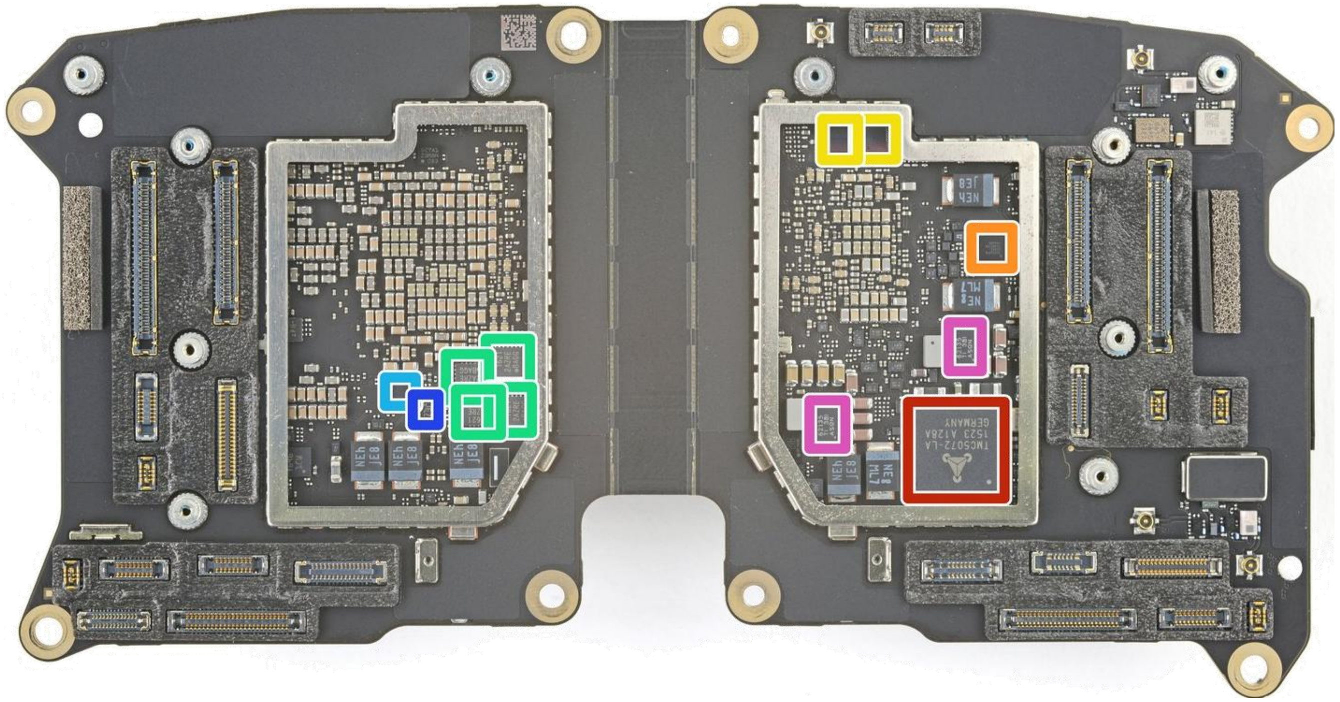
Modern Computer (Embedded) Systems

From Apple.

Modern Computer (Embedded) Systems



- Apple APL1109/339S01081E M2 octa-core applications processor & graphics processing unit & neural engine
- Micron MT62F1G64D8WT-031 XT:B 8 GB LPDDR5 SDRAM memory
- Apple APL1W08/339S01186 R1 sensor coprocessor
- Kioxia K5A4RC2097 256 GB NAND flash

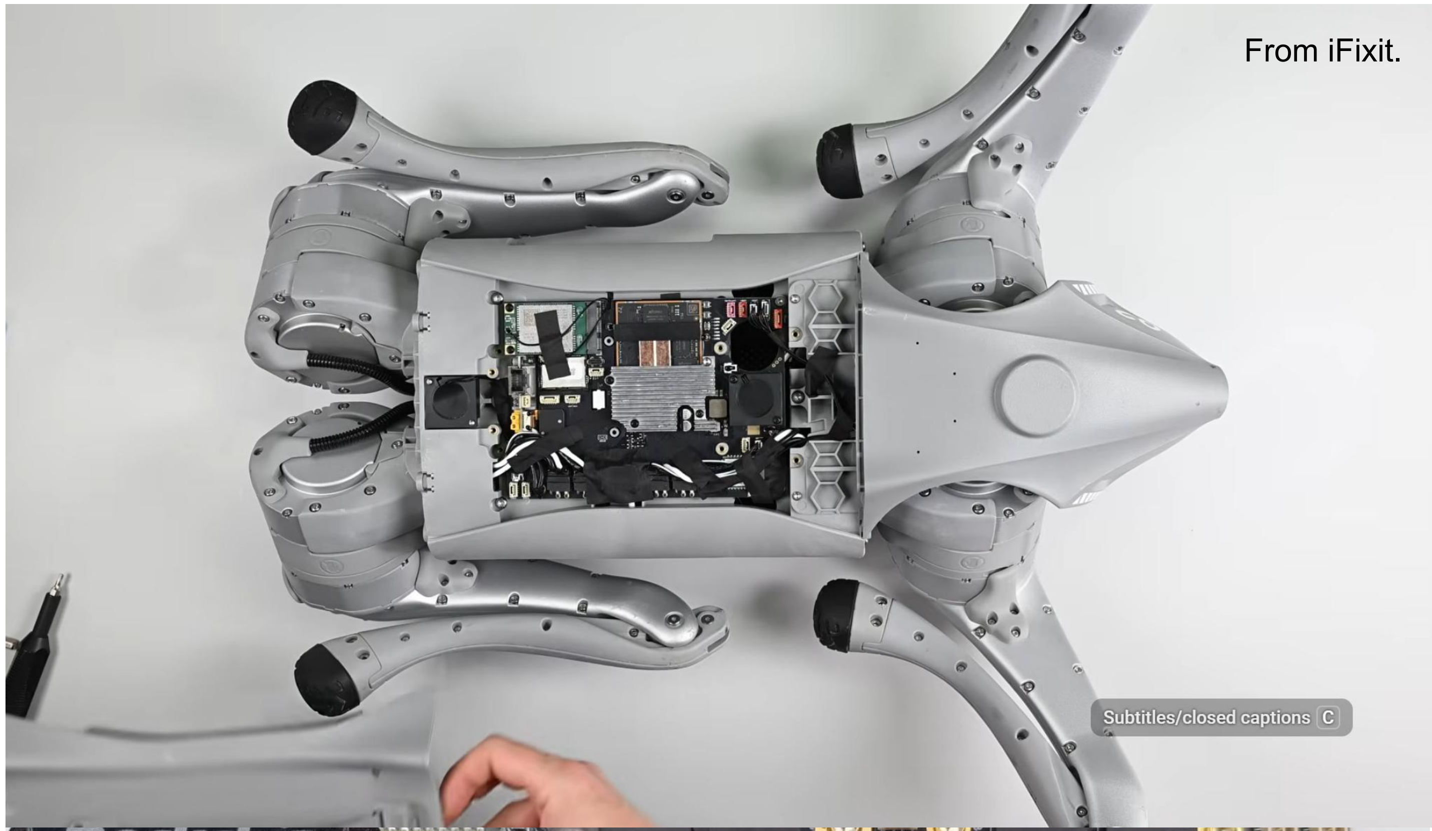


- Analog Devices TMC5072 dual 2-phase stepper motor driver
- Lattice Semiconductor ICE5LP4K iCE40 Ultra FPGA
- Likely Cirrus Logic CS46L11 audio codec

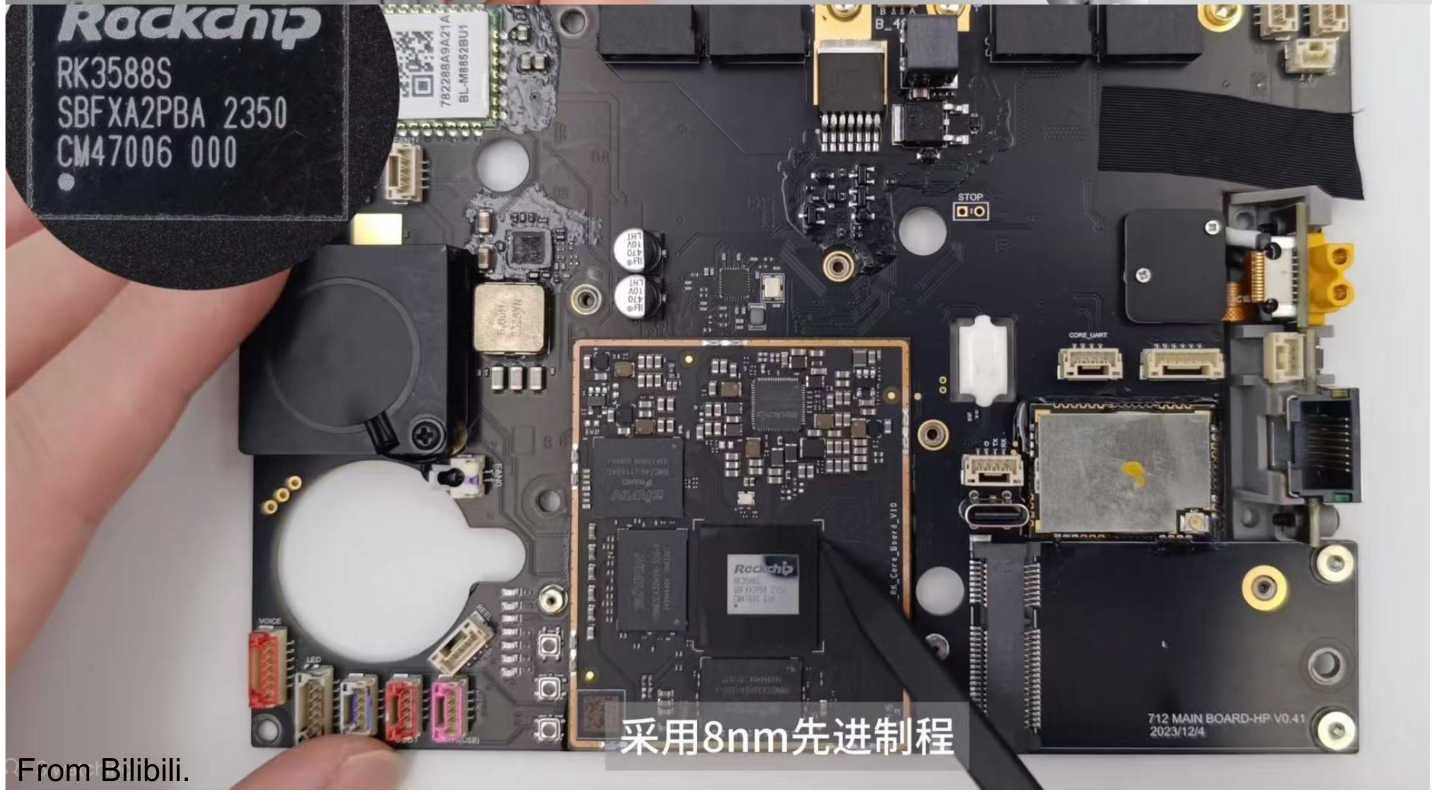
From iFixit.

Modern Computer (Embedded) Systems

From Unitree.



From iFixit.

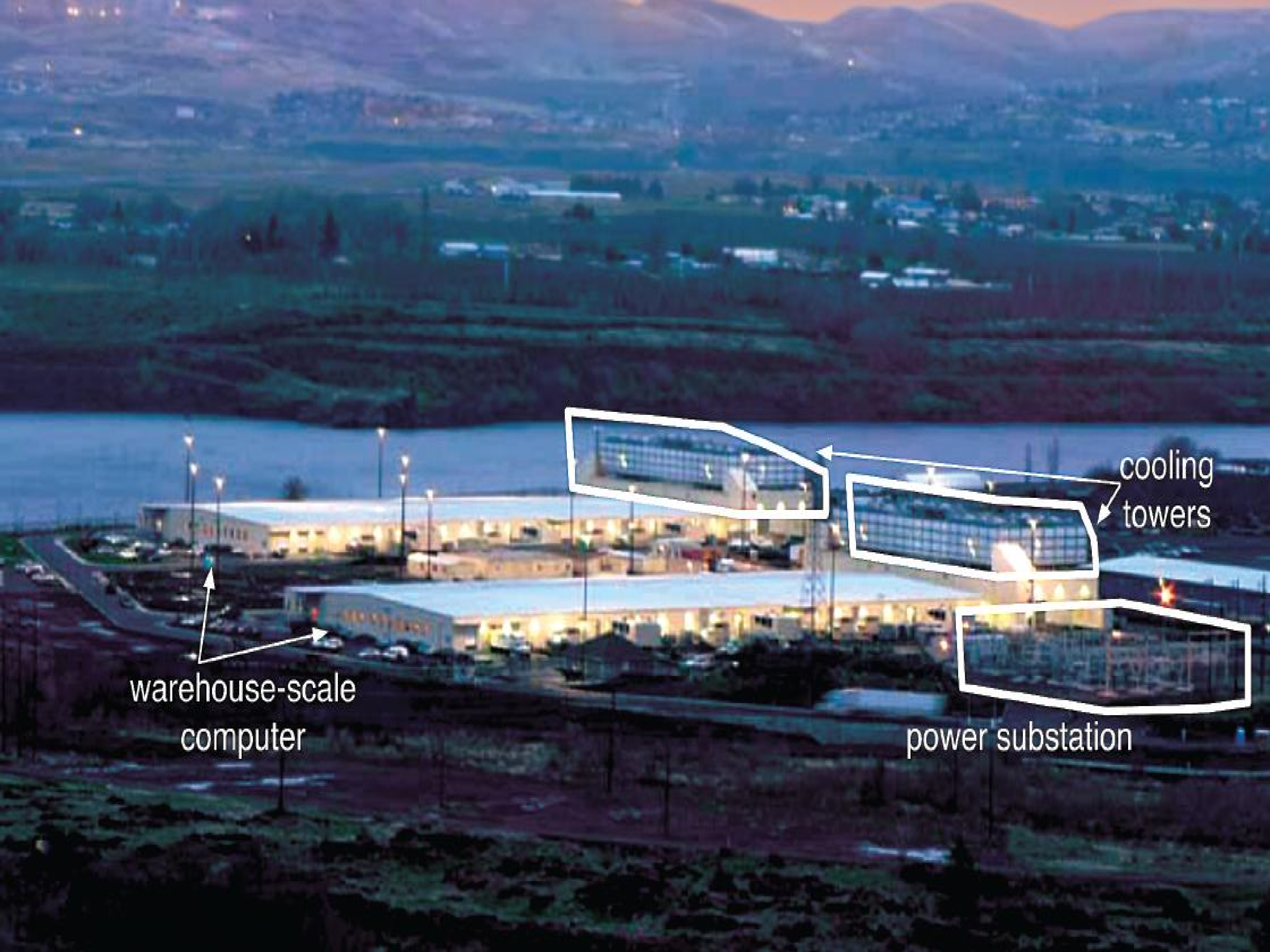


From Bilibili.

Modern Computer Systems

SIST computing center

- Hundreds/thousands of servers
- Each has one or multiple CPUs
- Mostly runs Linux OS


cooling
towers
warehouse-scale
computer
power substation

Hardware Revolution (Cloud Services)

- Heterogeneous

Providers	CPU	GPU	FPGA*	ASIC (DSA)
Alibaba	X86/ARM/RISC-V	Nvidia/AMD	Altera/AMD	AliNPU
AWS (Amazon)	X86/Graviton (ARM)	Nvidia/AMD	AMD	Trainium
Azure (MS)	X86	Nvidia	Altera	N/A
Baidu	X86	Nvidia	AMD	Kunlun
Google	X86	Nvidia	N/A	TPU
Huawei	X86/Kunpeng (ARM)	Nvidia & Ascend	AMD	Ascend
Tencent	X86	Nvidia & Xinghai	AMD	Enflame (燧原)

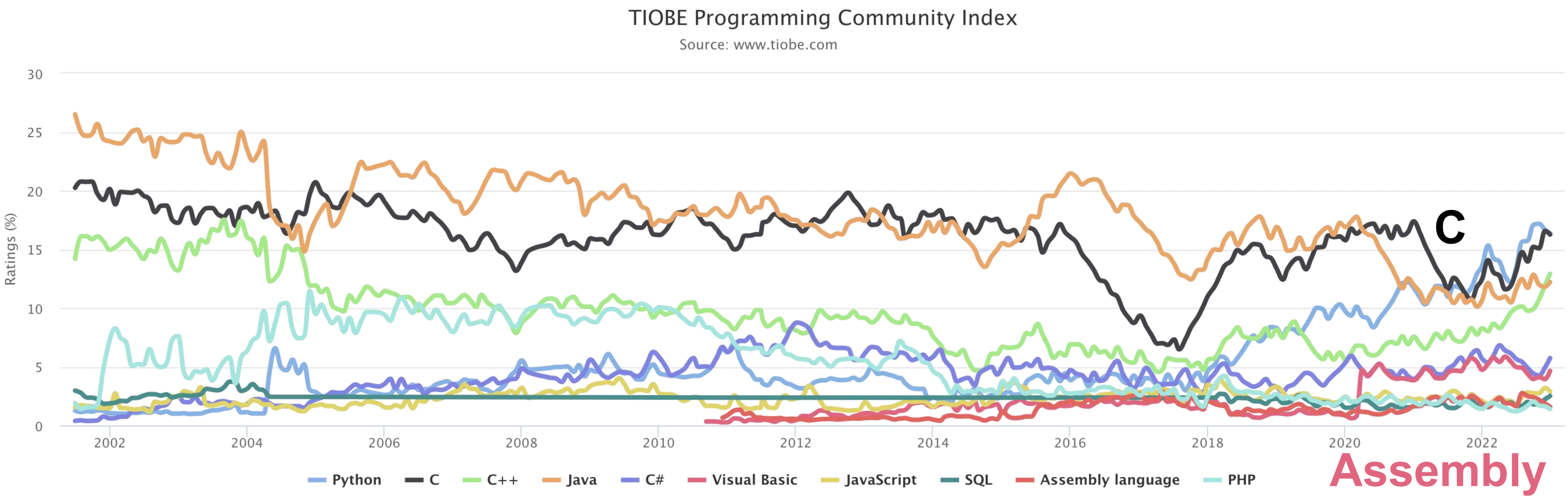
*AMD: formerly Xilinx

Computer Hardware Evolution

- Stronger (process complex information, e.g., AI)
  - High performance, high energy efficiency, etc.
- Easier to use (touch screen, voice control, etc.)
- Diversity (edge/cloud, various architectures)
- Heterogeneous (CPU/GPU/ASIC/FPGA)
- All developed based on computer architecture theory

Programming (Software)

- Write program (as CS students/programmers)


TIOBE Programming Community Index

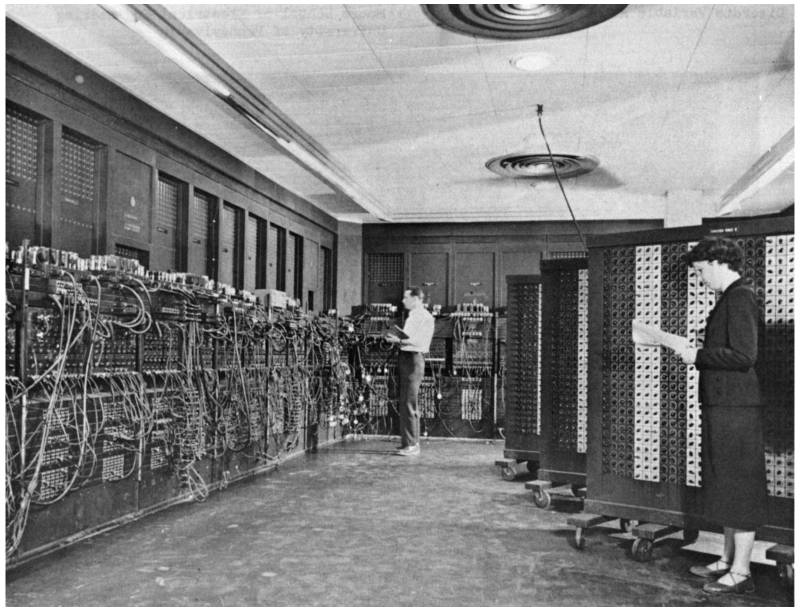
Source: www.tiobe.com

C

Assembly

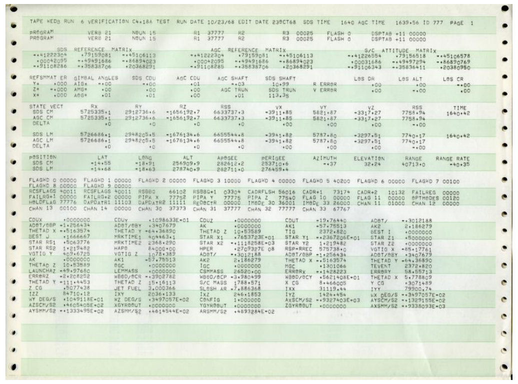
Programming Evolution

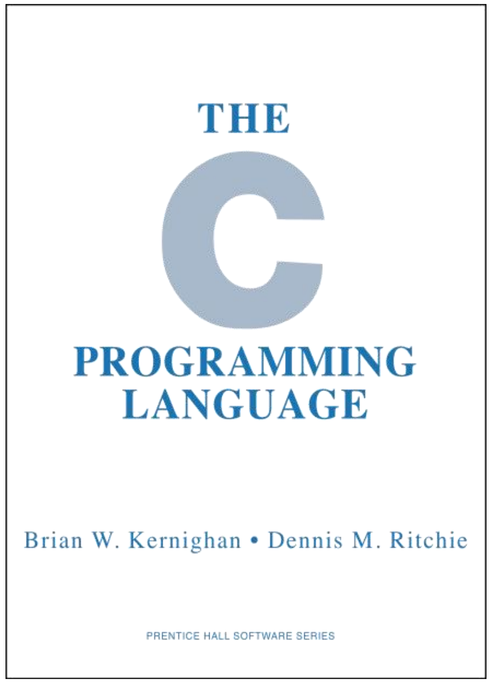
Ada Lovelace,
concept of program



ENIAC, switches and
relays

Margaret Hamilton with the
(assembly) code she wrote.





THE

C

PROGRAMMING
LANGUAGE

Brian W. Kernighan • Dennis M. Ritchie

PRENTICE HALL SOFTWARE SERIES

General purpose,
procedural programming
language
C

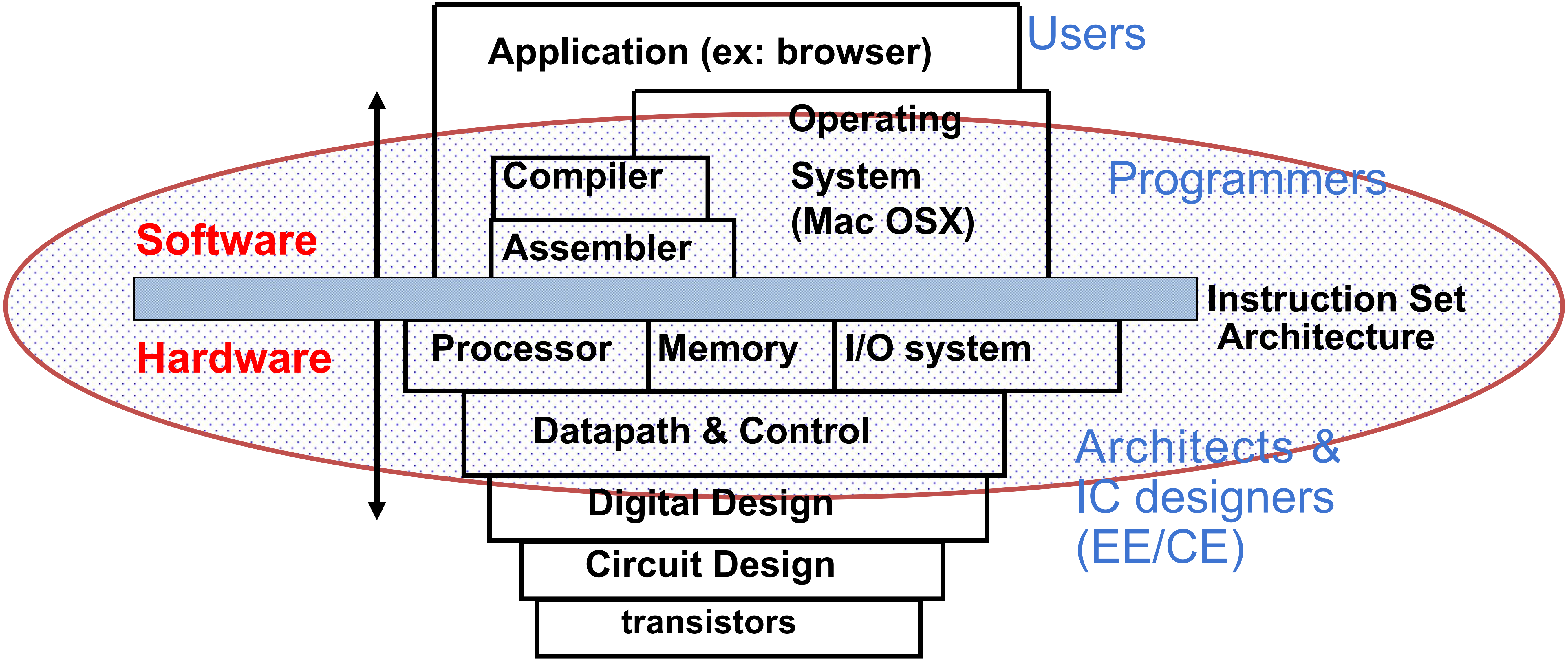
Swift
Java
Go
LabVIEW
MATLAB
PHP
Go
CUDA
tcl
Pytorch
MATLAB
Go
tcl
SQL
Swift
CUDA
Java
VerilogHDL
tcl
VisualBasic
Swift
Pytorch
Pytorch
VerilogHDL
CUDA
PHP
LabVIEW
SQL
PHP
CUDA
VisualBasic
MATLAB

"It is our job to create computing technology such that nobody has to program" by Jensen Huang.

Domain or
"hardware" specific

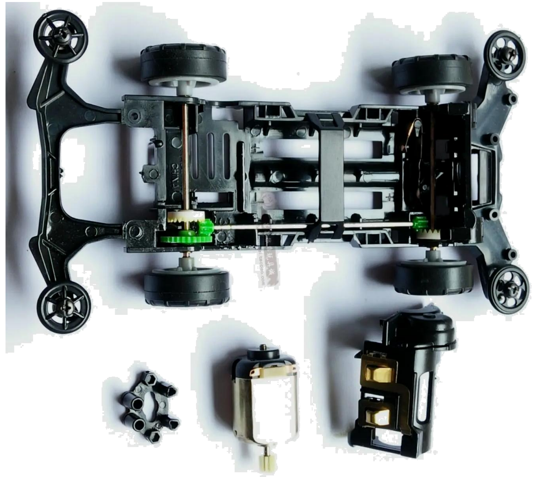
CS 110 is TO EXPLAIN HOW COMPUTER WORKS (FUNDAMENTALS)

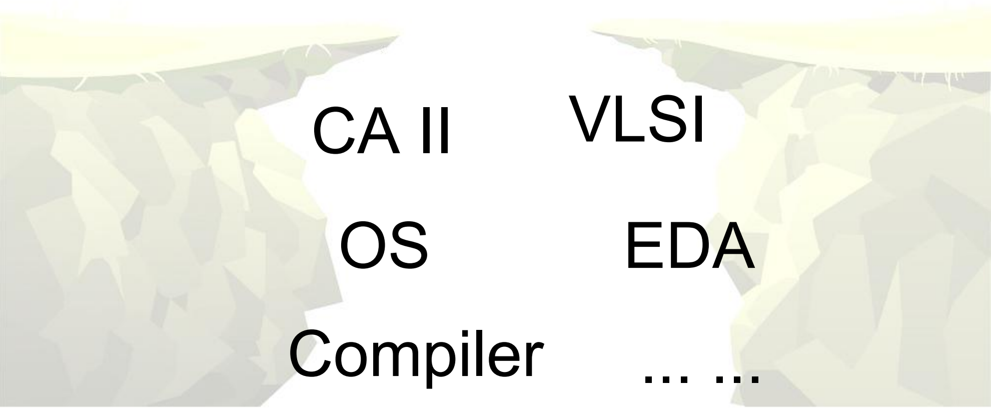
- Abstraction of computers



CS 110 is TO EXPLAIN HOW COMPUTER WORKS (FUNDAMENTALS)

- It is about the hardware-software interface
  - How is the high-level language translated to machine code
  - How to build the hardware (digital circuit) to understand the machine code and execute corresponding instructions (CPU/MCU design)





CA II

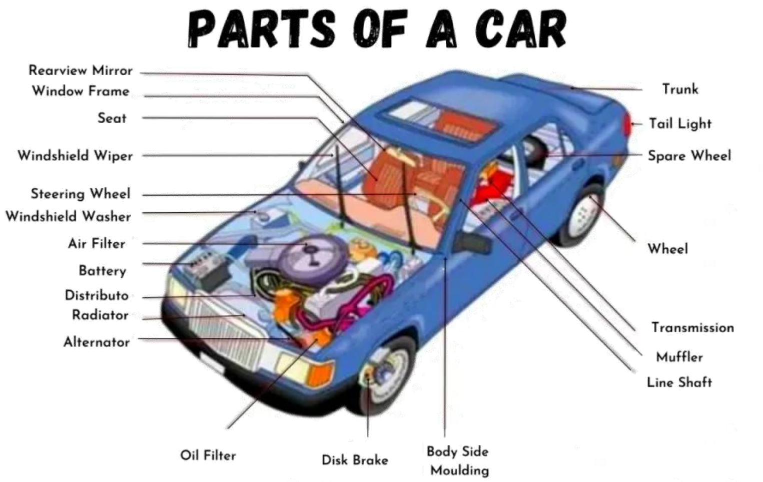
VLSI

OS

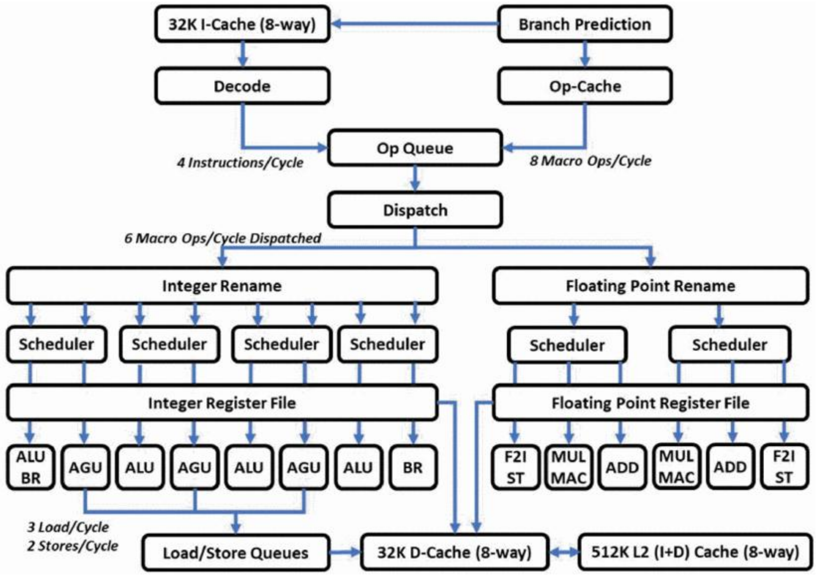
EDA

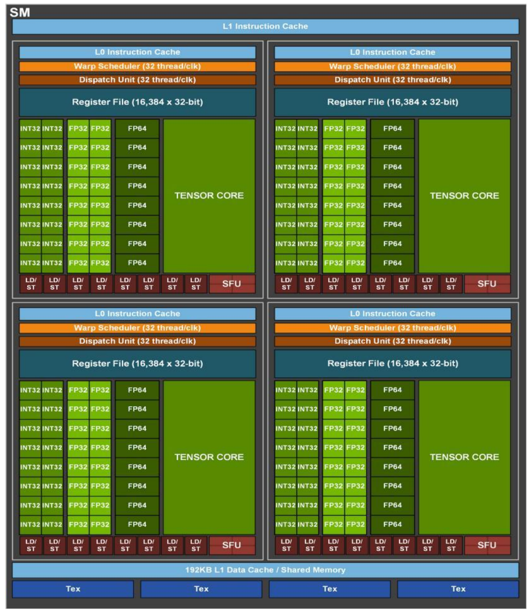
Compiler

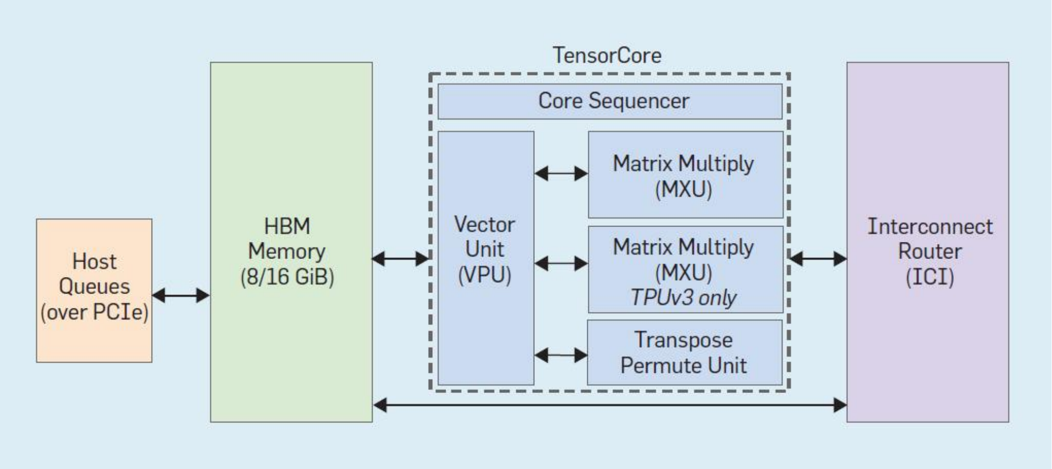
... ...



https://www.partsofacarengine.com/parts-of-a-car/




SM
L1 Instruction Cache
L0 Instruction Cache
Warp Scheduler (32 thread/clk)
Dispatch Unit (32 thread/clk)
Register File (16,384 x 32-bit)
INT32
FP32
FP64
TENSOR CORE
LD/ST
SFU
192KB L1 Data Cache / Shared Memory
Tex



CS 110 is TO EXPLAIN HOW COMPUTER WORKS (FUNDAMENTALS)

- It is about the hardware-software interface
  - How is the high-level language translated to machine code
  - How to build the hardware (digital circuit) to understand the machine code and execute corresponding instructions (CPU/MCU design)
- We will learn
  - Information representation
  - C review and memory management
  - CALL (compiler, assembler, linker & loader, basics)
  - ISA & assembly (RISC-V as example)
  - Digital circuit design

CS 110 is TO TRAIN PROFESSIONALS

- It is about the hardware-software interface
  - How is the high-level language translated to machine code
  - How to build the hardware (digital circuit) understand the machine code and execute corresponding instructions (CPU/MCU design)
  - Evaluate tradeoffs of different designs and ideas
- How to improve the performance of computers (parallelism/memory hierarchy)
- What does the programmer need to know to achieve the highest possible performance

CS 110 is **NOT** really about Programming

- Programming language as Tools
  - **C** is close to the underlying hardware, unlike languages like Python, Java!
    - Allows us to talk about key hardware features in higher level terms
    - Allows programmer to explicitly harness underlying hardware parallelism for higher performance and power efficiency
  - We will also learn **assembly**
    - Allows us to talk about the hardware in lower level terms
    - Uses **RISC-V** assembly/ISA as an example

deepseek

CUDA®

Parallel Thread eXecution **PTX ISA**

https://www.deepseek.com/
https://docs.nvidia.com/cuda/parallel-thread-execution/index.html (over 700 pages)

CS 110 is TO EXPLAIN HOW COMPUTER WORKS (FUNDAMENTALS)

- Other than the CPU core & main memory, more to consider
  - I/O
  - Operating system/virtual memory
  - Reliability and security issues
  - Large-scale or warehouse-scale computing (WSC)
  - Heterogenuous computing
  - ... ...

Course Contents

- How hardware & program orchestrate in a computer?
  - Software side:
    - Information representation
    - C review and memory management
    - Assembly (RISC-V ISA)
    - Compiler, assembler, linker & loader (in general)
  - CPU hardware design & How CPU hardware works (RISC-V ISA)
    - Logic & synchronous digital systems
    - Functional Units, FSM & Datapath
    - Pipeline, superscalar (Parallelism 1, Instruction-level parallelism)
    - Multi-issue
- Memory systems (Cache, Main memory)
- Parallelism 2 (SIMD, TLP, Sync & OpenMP, data/thread-level parallelism)
- Topic studies (OS, interrupts, I/O, virtual memory, FPGA, warehouse-scale computing, DMA, external memory, fault-tolerance, security, etc.)

Why These Materials?

- Because you are here for a Computer Science degree!

- Regardless of your future direction, learning the principles of digital design & computer architecture will be useful to
  - design better hardware (computer engineering, IC design)
  - design better software (as programmer)
  - design better systems (as architects)
  - make better tradeoffs in design (required for all CS-related tasks)
  - understand why computers behave the way they do
  - solve problems better
  - think “in parallel”
  - etc. (too many to list)

From Onur Mutlu’s course: Digital design and computer architecture

Learning Goals

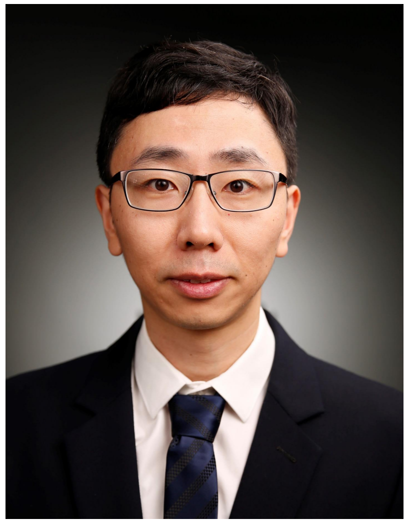
- Become a CS professional!
- Know how the hardware works in DEPTH (CPU, memory, I/O, etc.)
- Be prepared for further CS/EE course studies

Outline

- History and backgrounds
  - Recent trends
- Course content
- Course info., rules & elements
- Great ideas in computer architecture

Meet the Teaching Team!

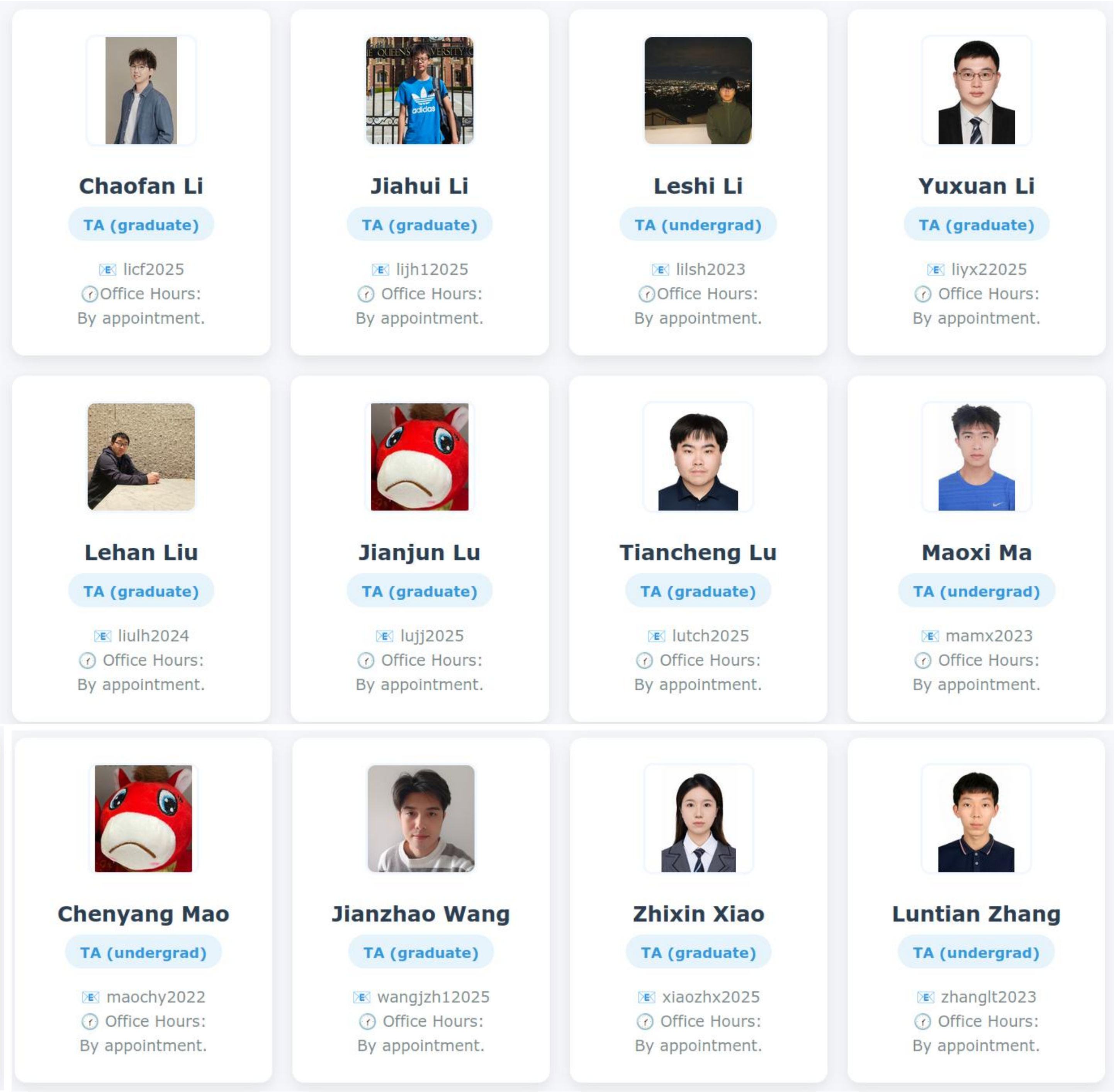
- Greetings!



Siting Liu
<liust>

Yuan Xiao
<xiaoyuan>

- TAs



Chaofan Li
TA (graduate)
✉ licf2025
🕒Office Hours:
By appointment.

Jiahui Li
TA (graduate)
✉ lijh12025
🕒 Office Hours:
By appointment.

Leshi Li
TA (undergrad)
✉ lilsh2023
🕒Office Hours:
By appointment.

Yuxuan Li
TA (graduate)
✉ liyx22025
🕒 Office Hours:
By appointment.

Lehan Liu
TA (graduate)
✉ liulh2024
🕒 Office Hours:
By appointment.

Jianjun Lu
TA (graduate)
✉ lujj2025
🕒 Office Hours:
By appointment.

Tiancheng Lu
TA (graduate)
✉ lutch2025
🕒 Office Hours:
By appointment.

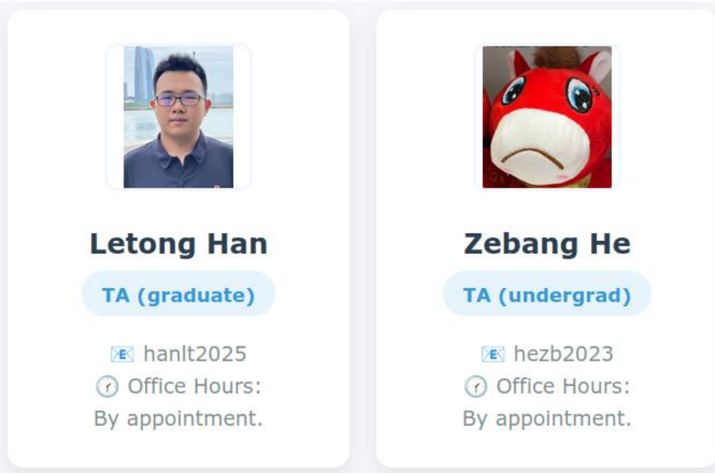
Maoxi Ma
TA (undergrad)
✉ mamx2023
🕒 Office Hours:
By appointment.

Chenyang Mao
TA (undergrad)
✉ maochy2022
🕒 Office Hours:
By appointment.

Jianzhao Wang
TA (graduate)
✉ wangjzh12025
🕒 Office Hours:
By appointment.

Zhixin Xiao
TA (graduate)
✉ xiaozhx2025
🕒 Office Hours:
By appointment.

Luntian Zhang
TA (undergrad)
✉ zhanglt2023
🕒 Office Hours:
By appointment.



Letong Han
TA (graduate)
✉ hanlt2025
🕒 Office Hours:
By appointment.

Zebang He
TA (undergrad)
✉ hezb2023
🕒 Office Hours:
By appointment.

Course Info

CS110				
Lecture	Tue. & Thur.	8:15-9:55	Teaching Center 303 SIST 1D107	Siting Liu Yuan Xiao (Parallel sessions)
Lab sessions	(CS110P)			
Lab 1	Mon.	18:00-19:40	SIST 1B-106	TA: Zebang He
Lab 2	Mon.	18:00-19:40	SIST 1B-108	TA: Letong Han
Lab 3	Mon.	18:00-19:40	SIST 1B-110	TA: Leshi Li
Lab 4	Tue.	19:50-21:30	SIST 1B-106	TA: Tianchen Lu
Lab 5	Tue.	19:50-21:30	SIST 1B-108	TA: Chenyang Mao
Lab 6	Thu.	19:50-21:30	SIST 1B-106	TA: Maoxi Ma
Lab 7	Thu.	19:50-21:30	SIST 1B-108	TA: Luntian Zhang
Lab 8	Thu.	19:50-21:30	SIST 1B-110	TA: Chaofan Li
Discussion	Mon.	19:50-21:30	Teaching Center 301	TBD (recording will be provided afterwards)

***Lab & Discussion starts next week.**

Lab Sessions

- Labs: indepedent works on labs and projects!
  - Labs start next week
- Need Ubuntu Linux for the labs!
  - Install natively (dual boot)! (STFW)
  - Virtual Machine works fine
  - If in doubt: Ubuntu 24.04
  - Many labs/homeworks/projects may work with Mac, but no guarantee, no support. Use Linux!
  - **No Windows Subsystem for Linux (WSL)!!!**

Discussion Sessions

- Time (Starting the 2nd week)
  - Monday 19:50-21:30 Teaching center 301
- Content: The TAs will give a presentation about the topics being covered in the lectures, followed by Q&A.
- Discussions often cover topics from the lectures AND the project (tools, linux, etc.)
- Participation is strongly encouraged.
- Recording will be released after each discussion session.

Office Hours

- Meet a TA in person and ask questions/get help with HW or projects
- **Before going to office hour:**
  - Is you question already answered on Piazza?
  - If the question is suitable (i.e. not giving away answers), **prefer to ask on piazza**
- Office hours of TAs will be based on appointment
- Office hours with the Profs. are also possible – write an email first to make an appointment.

Computer Architecture

- **The most important** course this semester

  6 credit points in total

  4 credit lecture (CS110); 2 credit projects (CS110P)

  - Your first CS only course
  - Spend a lot of time on:

    Textbook reading before class;

    HWs and projects;

    Lab preparation;

    Mid-terms and final;

- Start **early** on all the assignments!

CS110 vs CS110P

- Project course CS110P and CA I course are graded separately =>
  - Different grades in both are likely
  - It's possible to fail one but not the other
- The scheduled time for CS110P is used for the lab sessions, which are part of the course!
- CS110P is for the projects and labs.
- Announcements for the projects are held in CS110 lectures **AND** posted on Piazza.
- You can get help the projects on Piazza and from the TAs during your lab sessions, discussion sessions and the office hours.

Course Info

- **Course webpage**: https://faculty.sist.shanghaitech.edu.cn/liust/courses/CS110.html
- Acknowledgement: UC Berkeley’s CS61c: https://cs61c.org/; CS152/252A; 一生一芯: https://ysyx.oscc.cc/
- Textbooks: Average 15 pages of reading/week
  - Patterson & Hennessey, Computer Organization and Design RISC-V edition!
  - Kernighan & Ritchie, The C Programming Language, 2nd Edition
  - RTFM: C & RISC-V manual
- Piazza:
  - Every announcement, discussion, clarification happens there
- Materials this year are similar to previous years, but there might be differences!
- Gradescope:
  - Assignment submission and marking

CS110 Grading

- Homework (HW): 40%
  - 4 programming HW: 6% each (tentative)
  - 3 paper HW: 5% each (tentative)
  - HW1: 1%
- Lab (individual): 5%
- Exams: 53%
  - Midterm I: 14%
  - Midterm II: 14%
  - Final: 25%
- Participation: 2%
  - Based on participation (reading and writing posts) on Piazza, performance in assignments/projects & contributions to the class, etc.

CS110P Grading

- **Big Change !!!**
  - P/NP starting this year.
  - Project 1.1 17%
  - Project 1.2 17%
  - Project 2.1 16%
  - Project 2.1 16%
  - Project 3 17%
  - Project 4 17%
  - Total 100%

- Discuss & ask questions after each class
  - General topics can be discussed outside of class thread
- Ask general questions about HW/projects
- Announcements will be posted on piazza only!
  - Check email & Piazza at least once a day!
- Participation is recorded & goes into grade!
- **https://piazza.com/class/mjc8n24zks34gg#**

gradescope

by Turnitin

- Gradescope for code submission (can be linked with github, instructions will be provided on piazza), text-based HW and exams
- https://www.gradescope.com/courses/1243230
- Access code: PK3GX6
- HW1 is available on gradescope, **due Mar. 10th**

Late Policy … **No** Slip-Days!

- All assignments/labs/HWs/projects due at **11:59:59 PM** (unless specified otherwise)
- Only the last valid activated submission is evaluated.
- **No extention! Deadline is firm.**
- Start **EARLY** on all your assignments!

Policy on Assignments and Independent Work

- ALL LABS/HOMEWORKS/ASSIGNMENTS, AND PROJECTS WILL BE DONE INDEPENDENTLY
- With the exception of laboratories and assignments that explicitly permit you to work in groups, all homework are to be YOUR work and your work ALONE.
- PARTNER TEAMS MAY NOT WORK WITH OTHER PARTNER TEAMS
- You can discuss your assignments with other students, and credit will be assigned to students who help others by answering questions on Piazza (participation), but we expect that what you hand in is yours.
- Level of detail allowed to discuss with other students: Concepts (Material taught in the class/in the text book)! Pseudocode or code is NOT allowed!
- Use the Office Hours of the TA and the Profs. if you need help with your homework/project!
- Rather submit an incomplete homework with maybe 0 points than risking an F!
- It is NOT acceptable to copy solutions/code from other sources (other students/web/**AI-generated**).
- You can never look at homework/ project code not by you/ your team!
- You cannot give your code to anybody else –> secure your computer when not around it
- It is NOT acceptable to copy (or start your) solutions from the Web/other students/AI-generated.
- It is NOT acceptable to use PUBLIC github archives (giving your answers away)
- It is NOT acceptable to give anyone other than your project partner access to your gitlab!
- Protect your code and password! Do not allow other students peeping at your screen.
- We have tools and methods, developed over many years, for detecting this. You WILL be caught, and the penalties WILL be severe.
- At the minimum F in the course, and a letter to your university record documenting the incidence of cheating.
- **Both Giver and Receiver are equally culpable and suffer equal penalties**

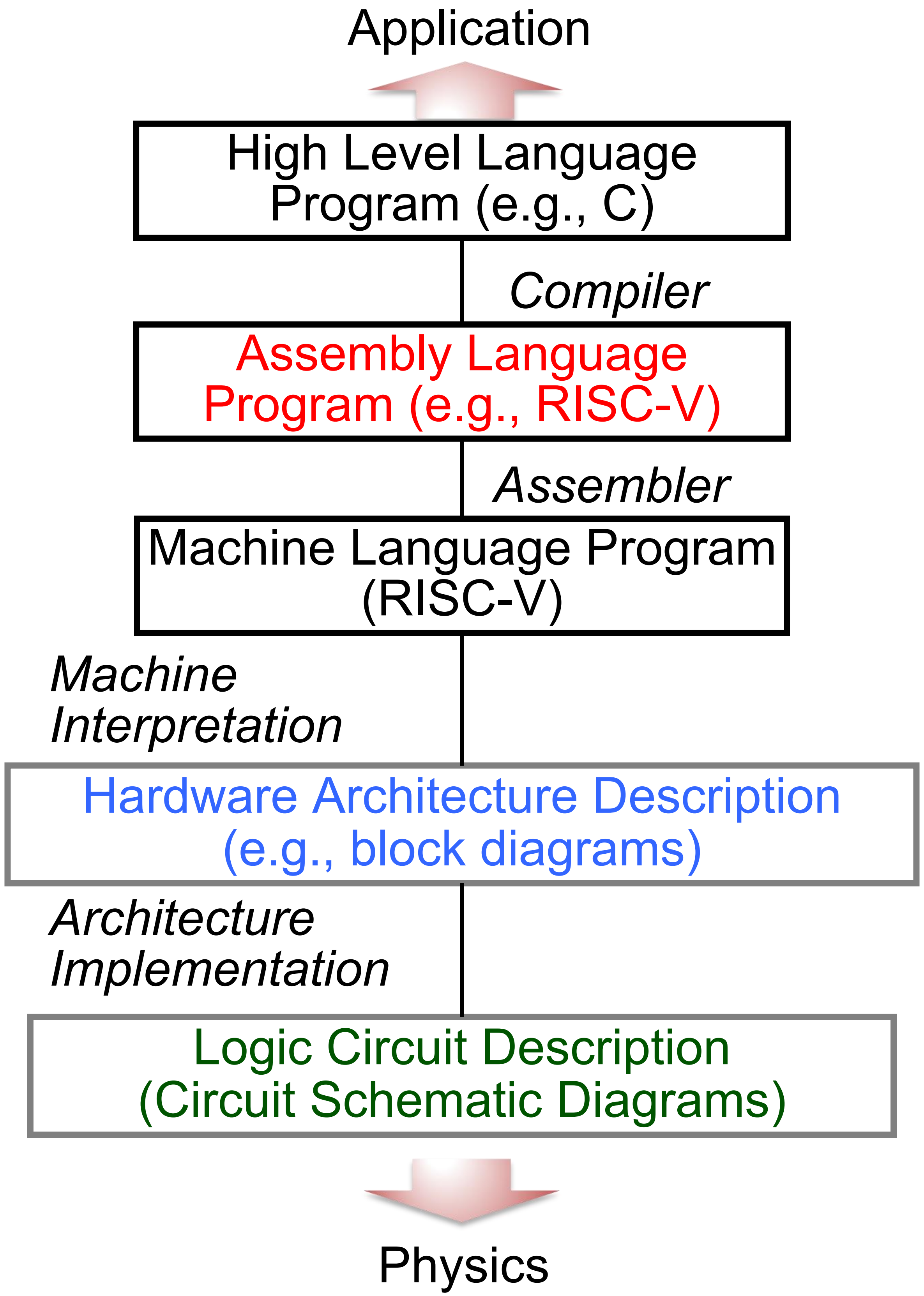
Outline

- History and backgrounds
  - Recent trends
- Course content
- Course info., rules & elements
- Great ideas in computer architecture

Great Ideas in Computer Architecture

- Abstraction (layers of representation/interpretation)
- Moore’s law (designing through trends)
- Make the common case fast
- Principle of locality (memory hierarchy)
- Parallelism (pipeline as a special case)
- Performance measurement & improvement
- Dependability via redundancy

Great Idea 1: Abstraction (Levels of representation/Interpretation)



Application

High Level Language Program (e.g., C)

Compiler

Assembly Language Program (e.g., RISC-V)

Assembler

Machine Language Program (RISC-V)

Machine Interpretation

Hardware Architecture Description (e.g., block diagrams)

Architecture Implementation

Logic Circuit Description (Circuit Schematic Diagrams)

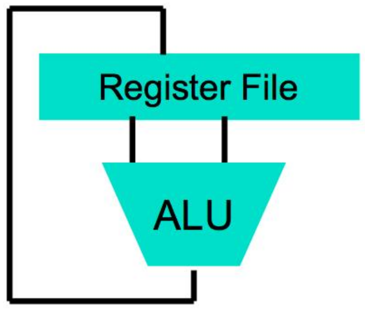
Physics

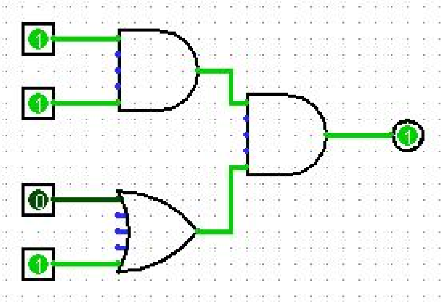
```
temp = v[k];
v[k] = v[k+1];
v[k+1] = temp;
```

```
lw    t0, 0(s2)
lw    t1, 4(s2)
sw    t1, 0(s2)
sw    t0, 4(s2)
```

Anything can be represented as a *number*, i.e., data or instructions

```
0000 1001 1100 0110 1010 1111 0101 1000
1010 1111 0101 1000 0000 1001 1100 0110
1100 0110 1010 1111 0101 1000 0000 1001
0101 1000 0000 1001 1100 0110 1010 1111
```





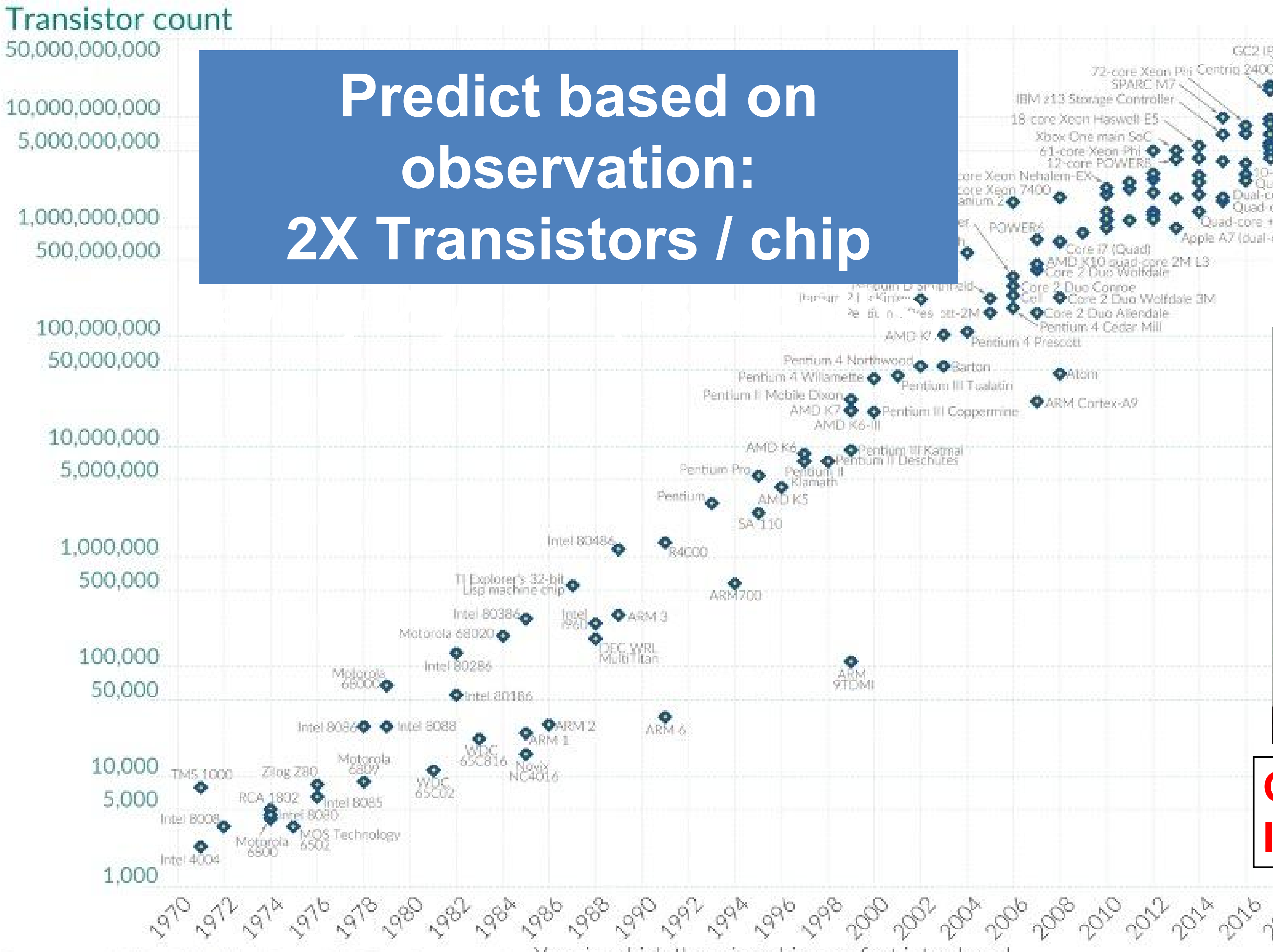
2. Moore's Law?

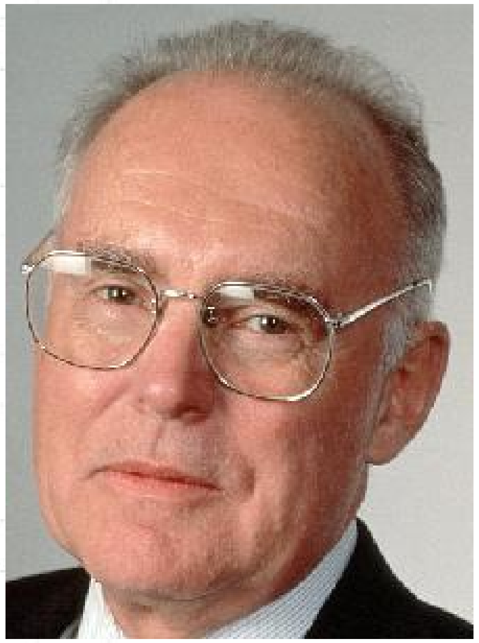
Moore's Law: The number of transistors on microchips doubles every two years

Moore's law describes the empirical regularity that the number of transistors on integrated circuits doubles approximately every two years. This advancement is important for other aspects of technological progress in computing – such as processing speed or the price of computers.

Our World in Data

Predict based on observation:
2X Transistors / chip





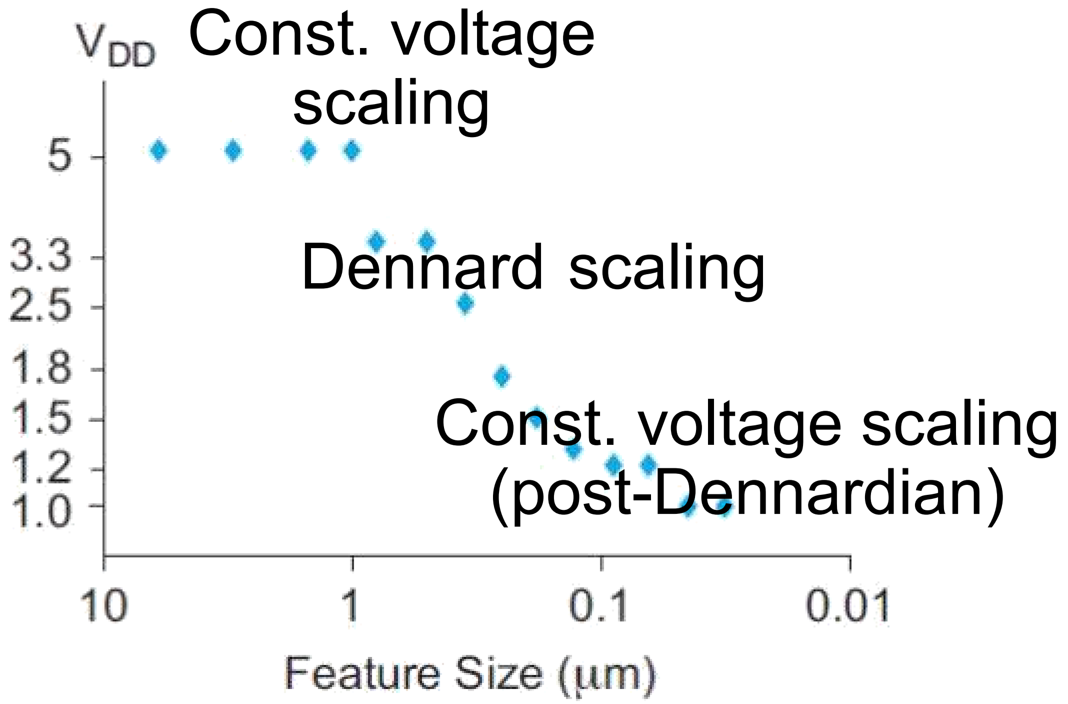
Gordon Moore
Intel Cofounder

Data source: Wikipedia (wikipedia.org/wiki/Transistor_count)

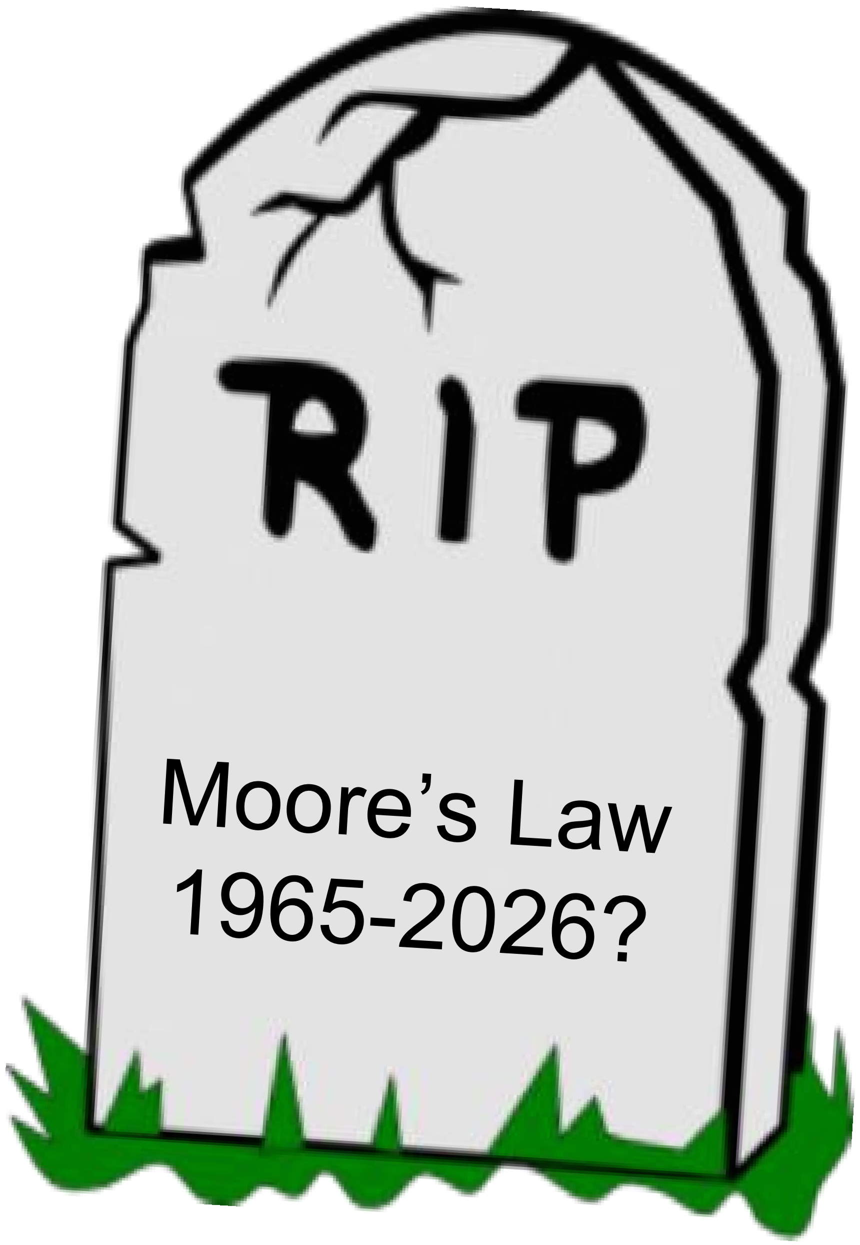
OurWorldinData.org – Research and data to make progress against the world's largest problems.

Interesting Times

- Dennard scaling (TDP)

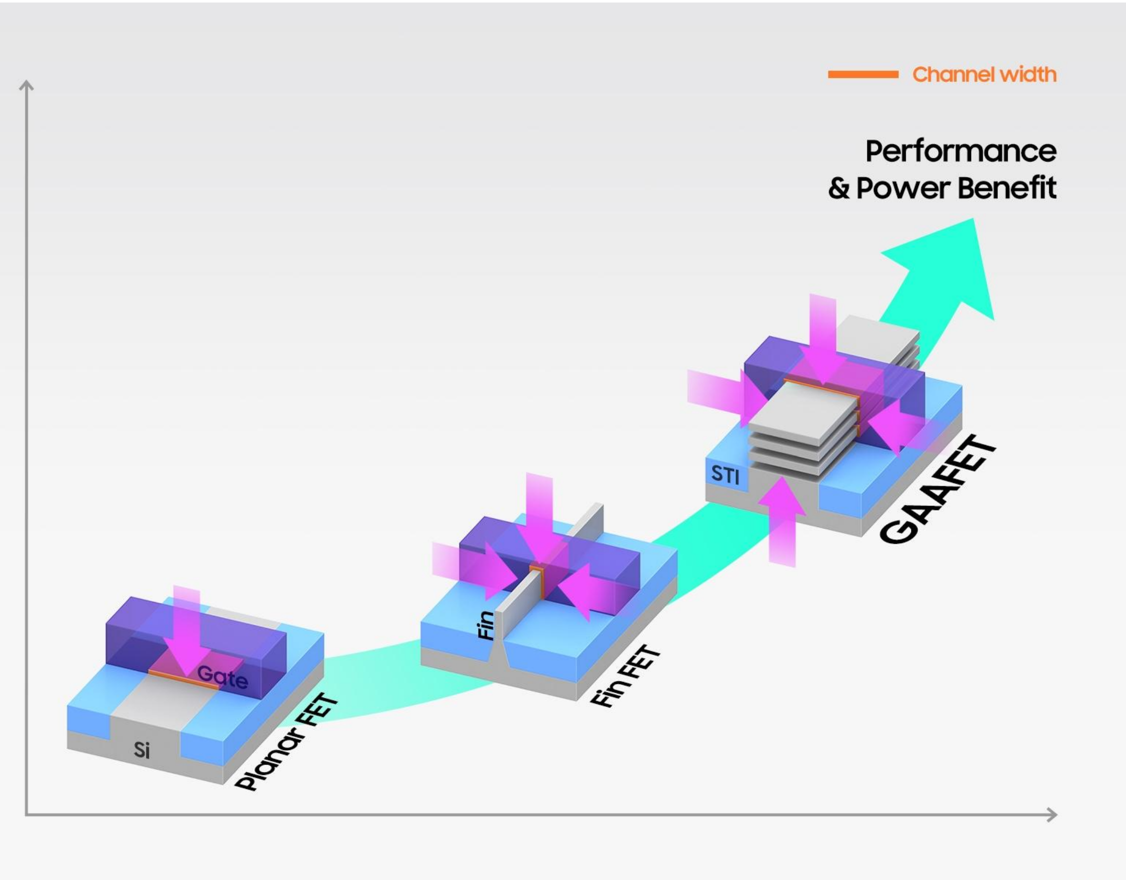


- Moore's Law relied on the cost of transistors scaling down as technology scaled to smaller and smaller feature sizes.
- BUT newest, smallest fabrication processes <5nm, might have greater cost/transistor !!!!
- So, why shrink???

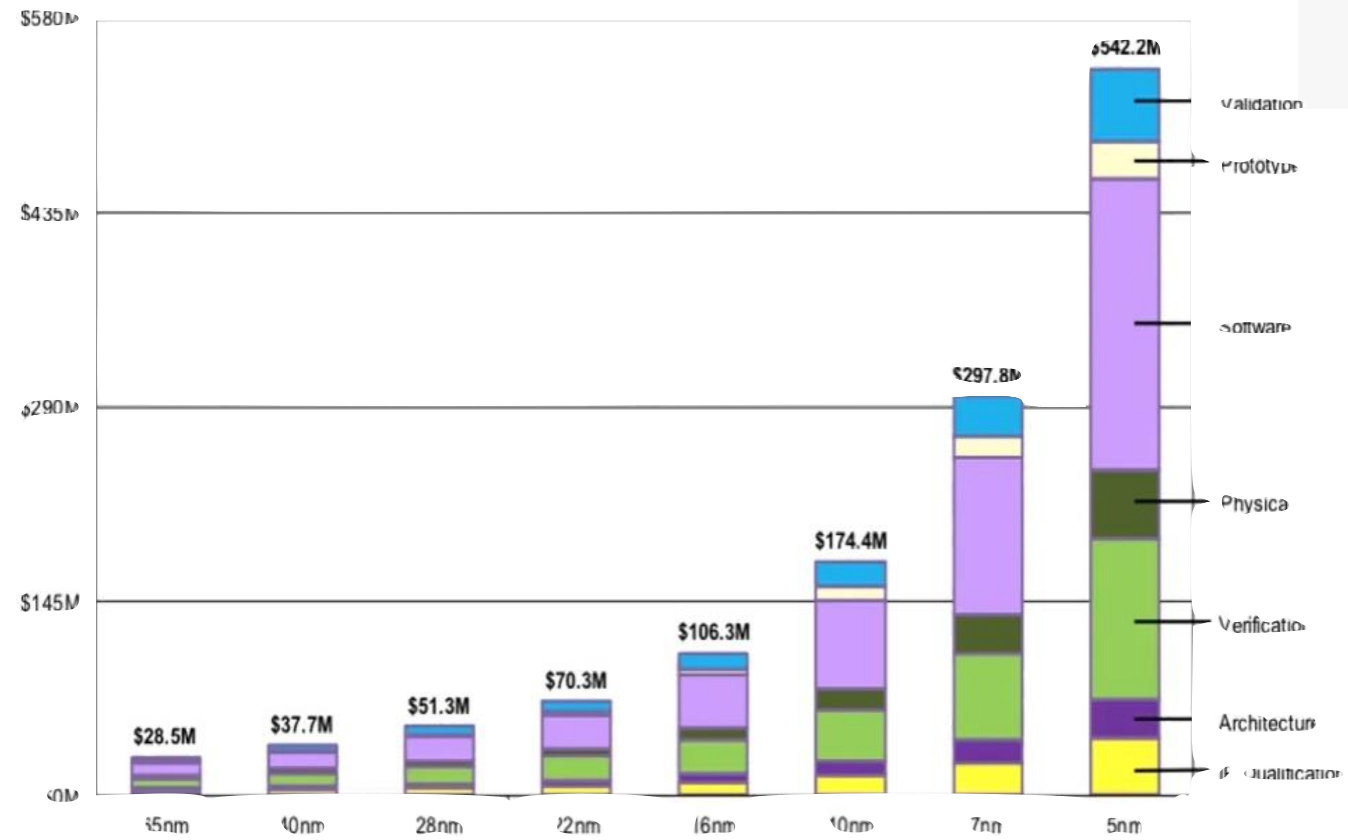


Advanced Technology Nodes

- Now: 4 nm process (A16 Bionic, TSMC)
- Future: up to 3 nm (rumors on A17)
- Other notable players: Samsung, Intel, GlobalFoundries (AMD), IBM
- China: SMIC: 14-nm production



From Samsung Semiconductor



Design cost at different technology nodes.
Data source: Handel Jones, IBS, 2018.

3. Make Common Case Faster: Amdahl's Law

A rule stating that the performance enhancement possible with a given improvement is limited by the amount that the improved feature is used. It is a quantitative version of the law of diminishing returns.

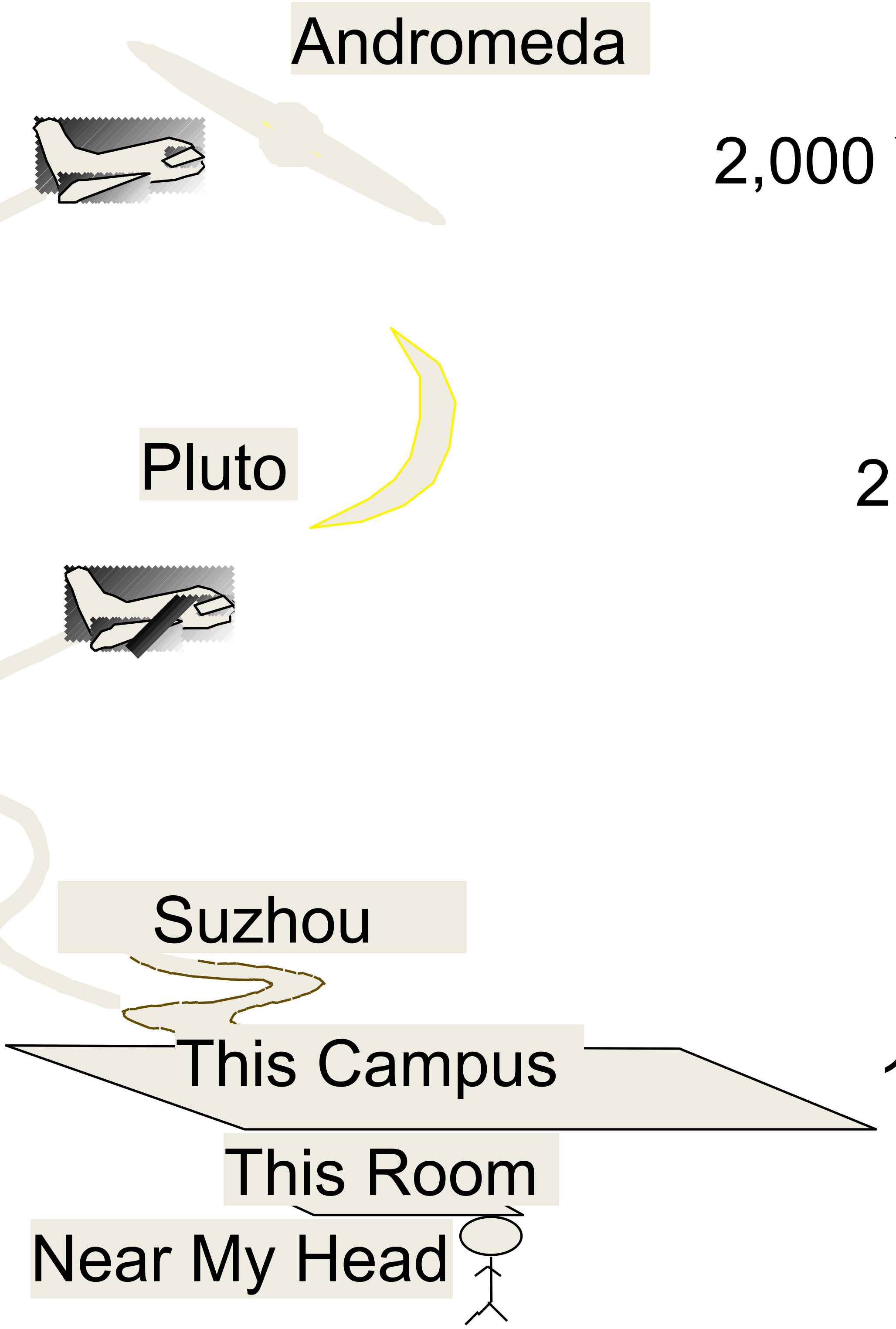
Gene Amdahl
Computer Pioneer

Execution time after improvement

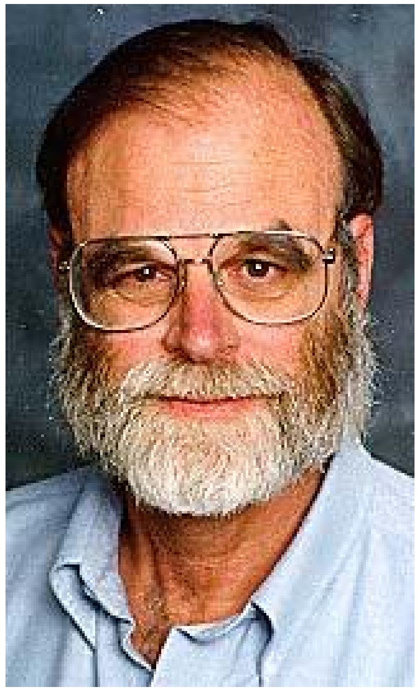
$$= \frac{\text{Execution time affected by improvement}}{\text{Amount of improvement}} + \text{Execution time not affected}$$

(+ Overhead for implementing the improvement)

Jim Gray's Storage Latency Analogy: How Far Away is the Data?

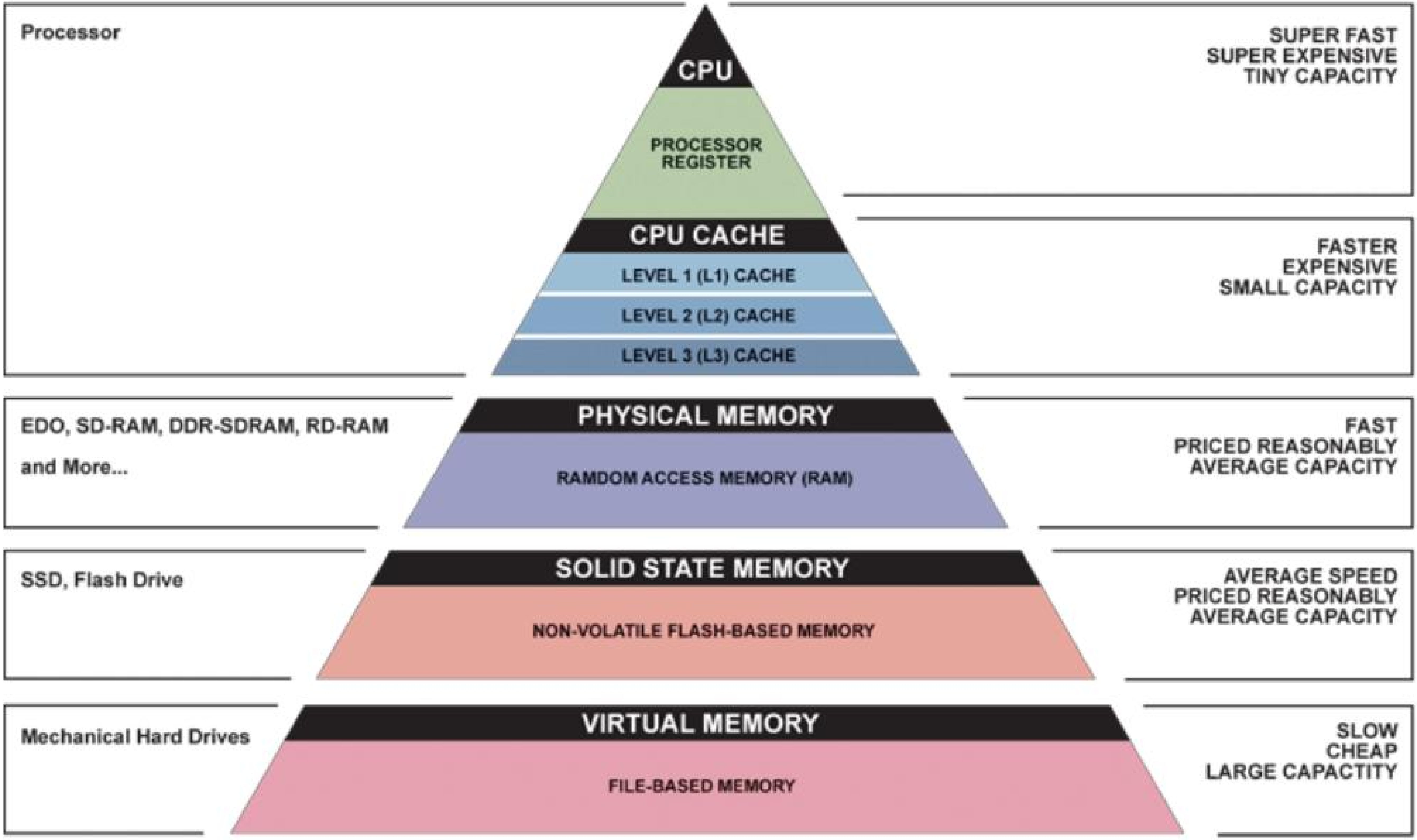


10^9	Tape	Andromeda	2,000 Years
10^6	HardDisk	Pluto	2 Years
100	Main Memory	Suzhou	1.5 hr
10	On Board Cache	This Campus	10 min
2	On Chip Cache	This Room	
1	Registers	Near My Head	1 min
(ns)			

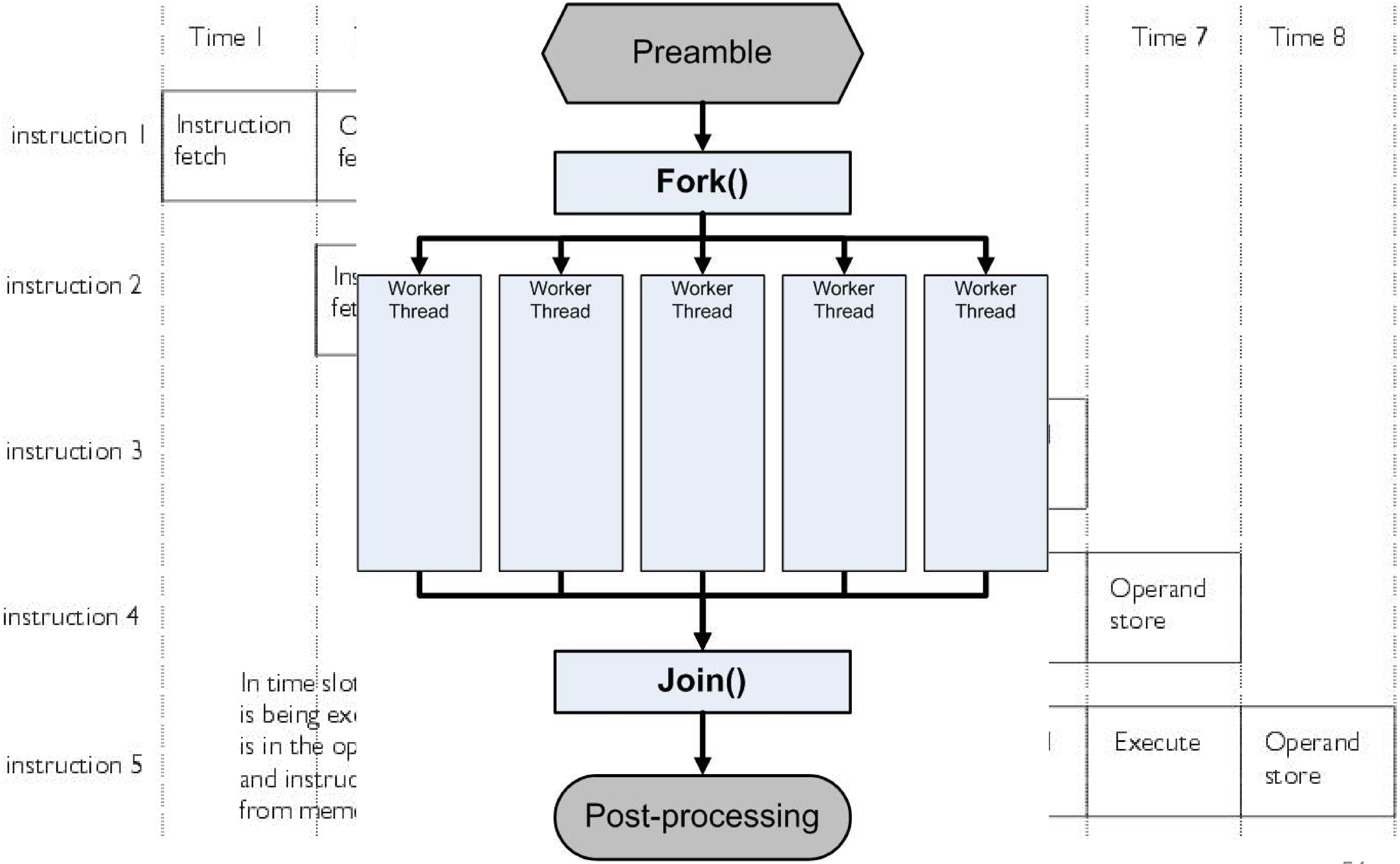


Jim Gray
Turing Award

Great Idea 4: Principle of Locality/ Memory Hierarchy



Great Idea 5: Parallelism/Pipeline



Caveat: Amdahl's Law

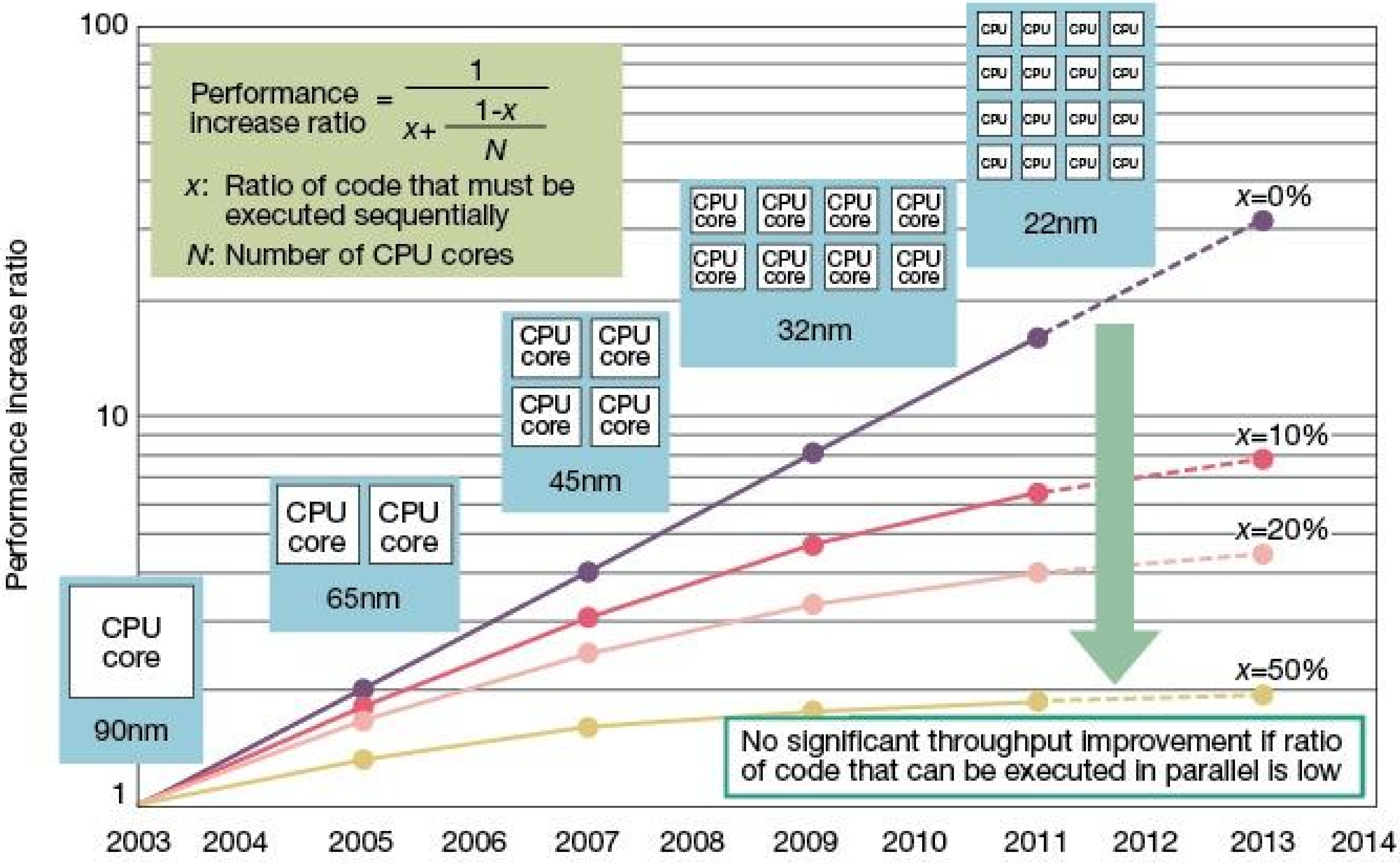


Fig 3 Amdahl's Law an Obstacle to Improved Performance Performance will not rise in the same proportion as the increase in CPU cores. Performance gains are limited by the ratio of software processing that must be executed sequentially. Amdahl's Law is a major obstacle in boosting multicore microprocessor performance. Diagram assumes no overhead in parallel processing. Years shown for design rules based on Intel planned and actual technology. Core count assumed to double for each rule generation.

Gene Amdahl
Computer Pioneer

Great Idea 6: Performance Measurement & Improvement

- Latency
  - How long to set the problem up. It is all about time to finish.
- Power/energy consumption
  - How much energy consumed to perform certain tasks.
- Benchmark (SPEC 2006, https://www.top500.org/, LINPACK)

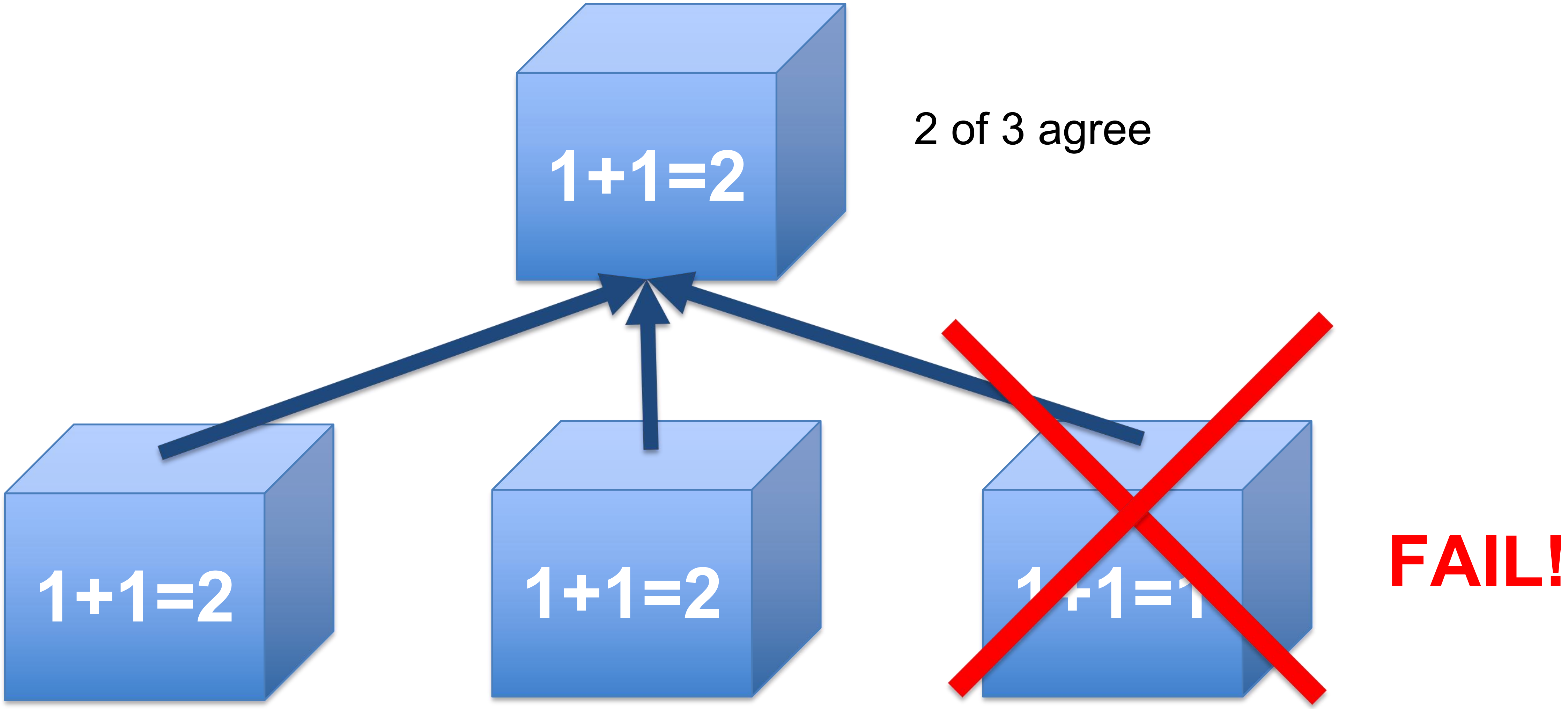
Coping with Failures

- 4 disks/server, 50,000 servers
  - Assume 4% annual failure rate
- On average, how often does a disk fail?
  - 1 / month
  - 1 / week
  - 1 / day
  - 1 / hour

50,000 x 4 = 200,000 disks
200,000 x 4% = 8000 disks fail
365 days x 24 hours = 8760 hours

Great Idea 7: Dependability via Redundancy

- Redundancy so that a failing piece doesn't make the whole system fails



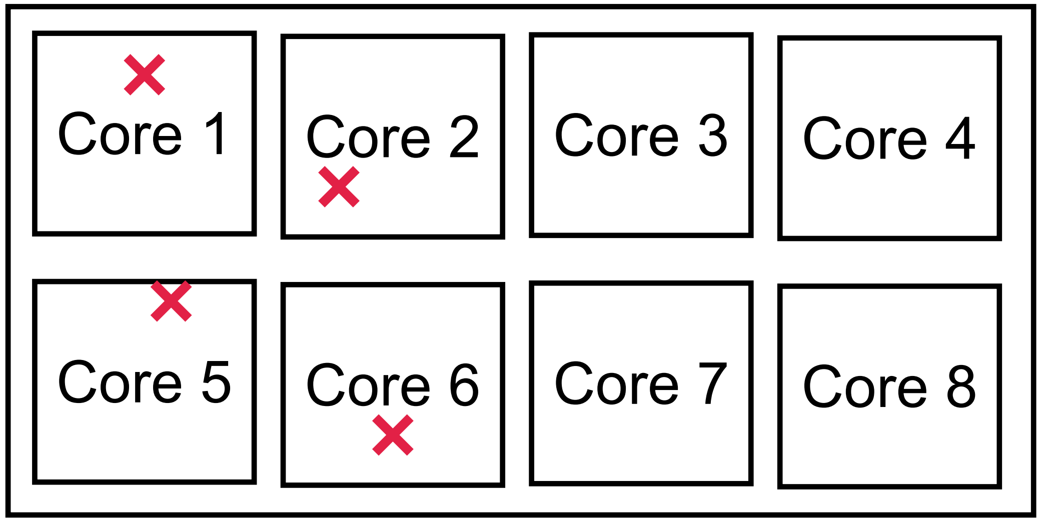
Increasing transistor density reduces the cost of redundancy

Great Idea 7: Dependability via Redundancy

- Applies to everything from datacenter to storage to memory to instructors
  - Redundant datacenters so that can lose 1 datacenter but Internet service stays online
  - Redundant disks so that can lose 1 disk but not lose data (Redundant Arrays of Independent Disks/RAID)
  - Redundant memory bits so that can lose some bits but not the data (Error Correcting Code/ECC Memory)

Great Idea 7: Dependability via Redundancy

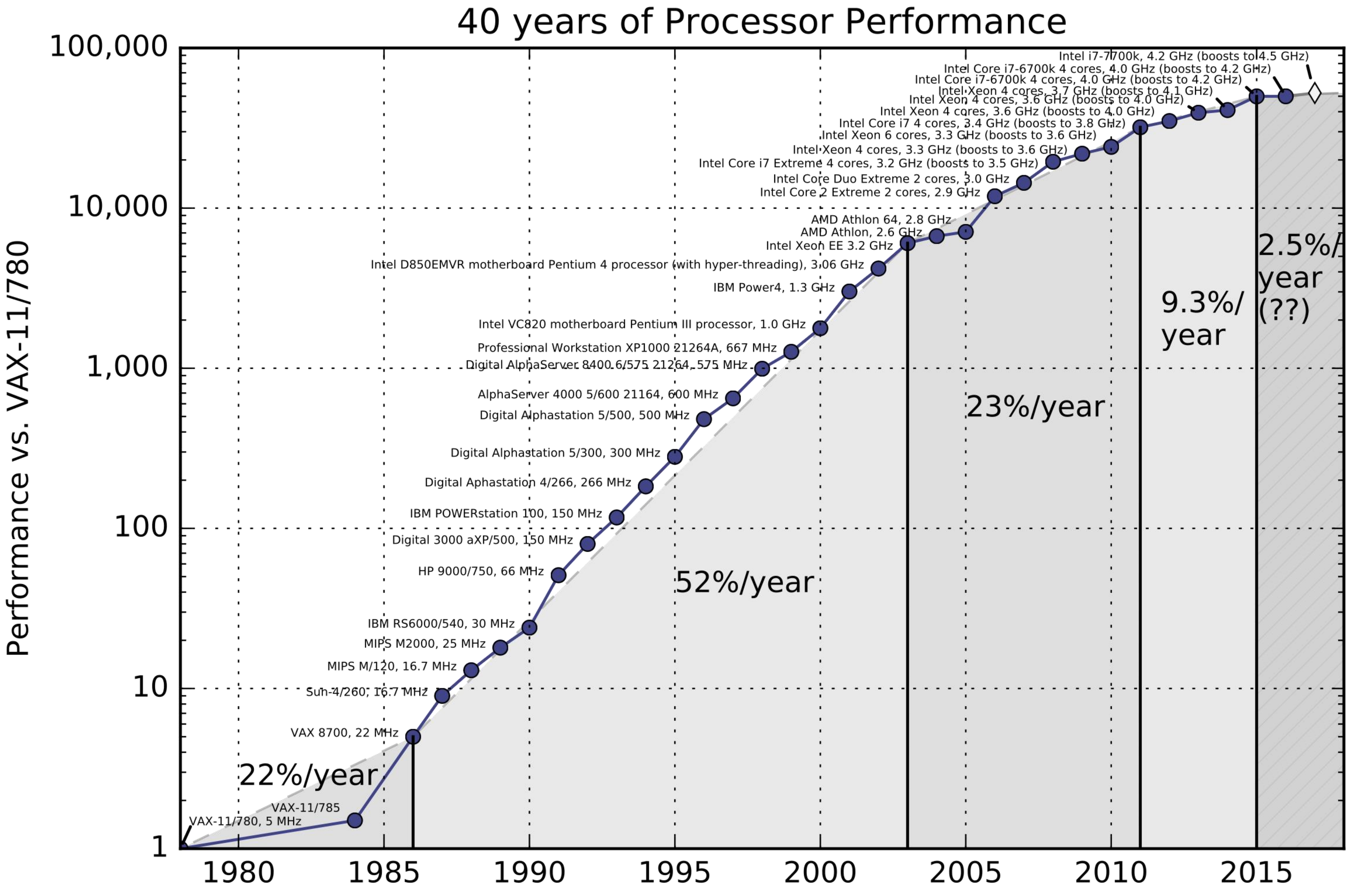
- Redundancy improves yield
  - Considering a CPU with 10^9 transistors
  - Each transistor has a probability 10^{-9} to be failure during production



Old Conventional Wisdom

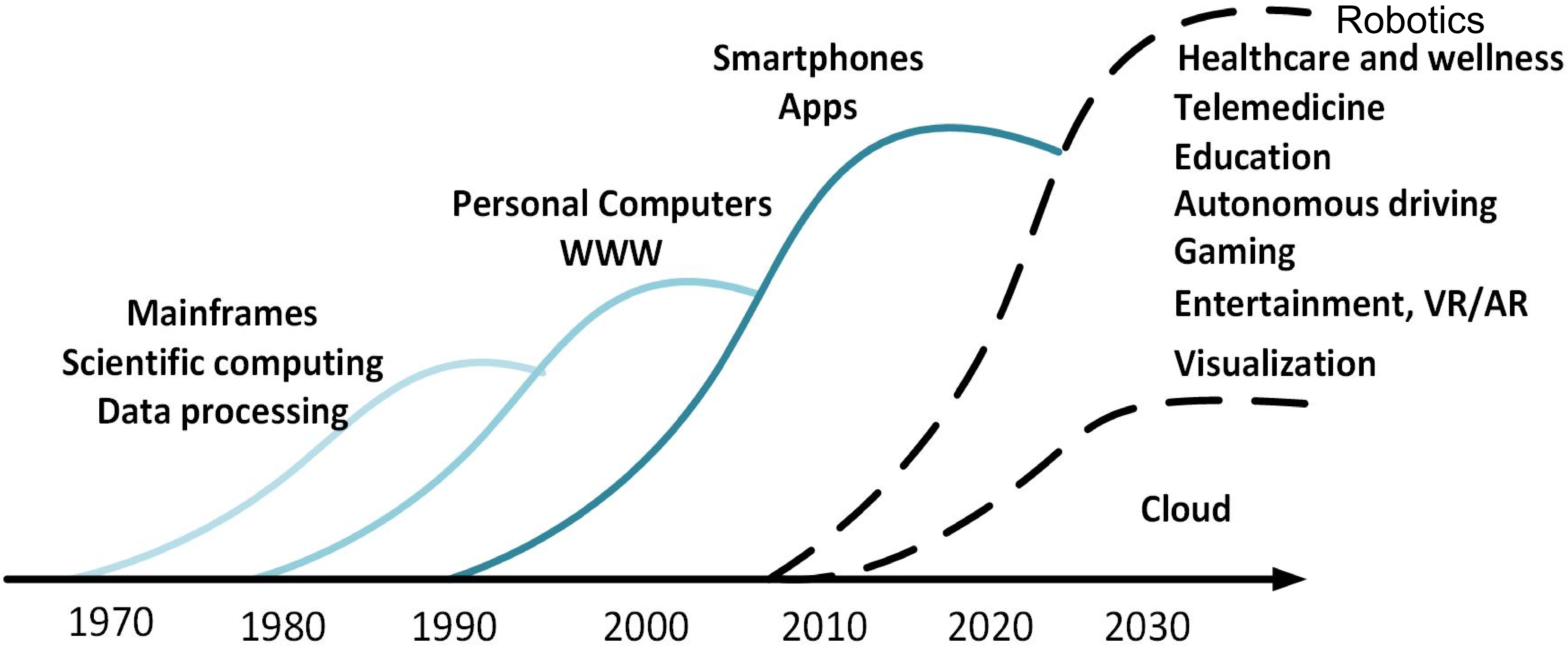
- Moore’s Law + Dennard Scaling = faster, cheaper, lower-power general-purpose computers each year
- In glory days, 1%/week performance improvement!
- Dumb to compete by designing specialized computers
- By time you’ve finished design, next generation of general-purpose will beat you

Why is Architecture Exciting Today?



Hennessey & Patterson, 2017

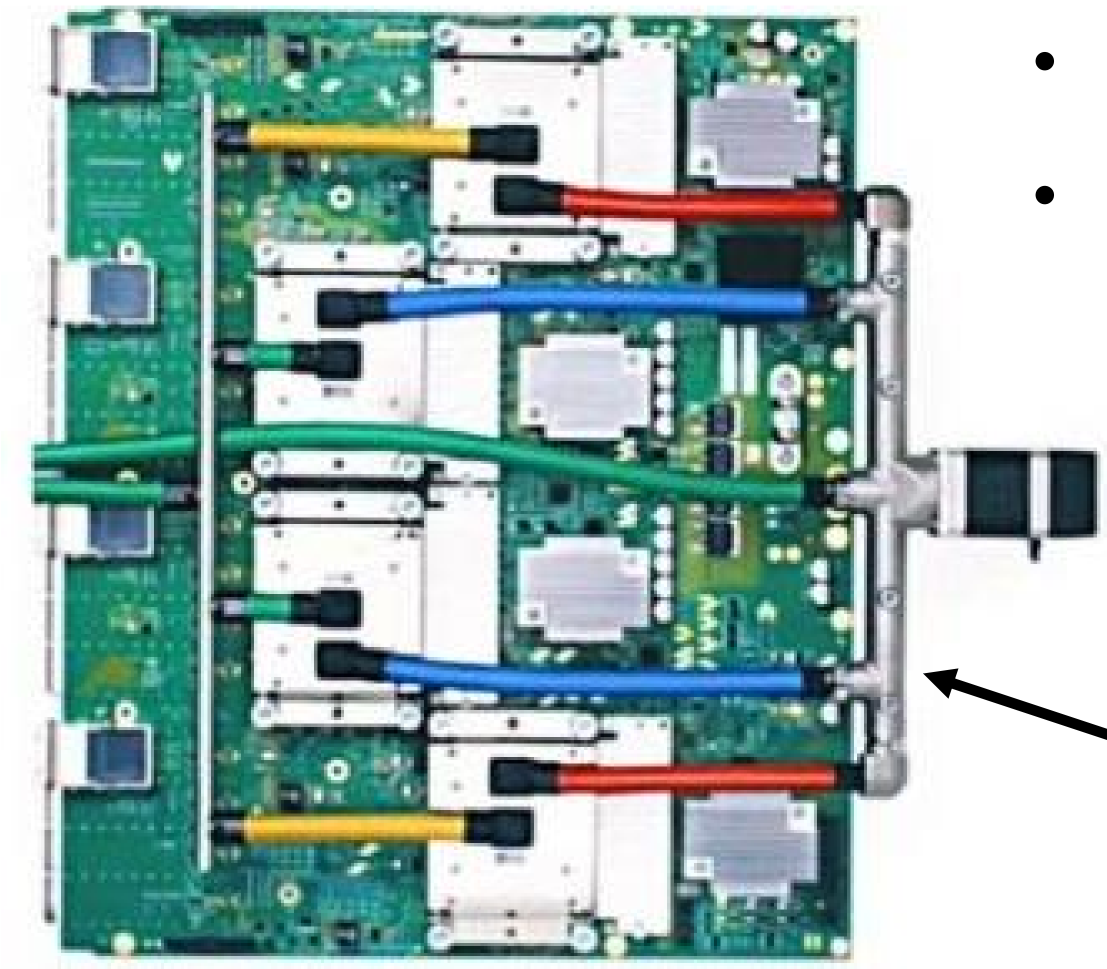
Why is Architecture Exciting Today?



- Number of deployed devices continues growing, but no single killer app
  - Diversification of needs, architectures

New Conventional Wisdom

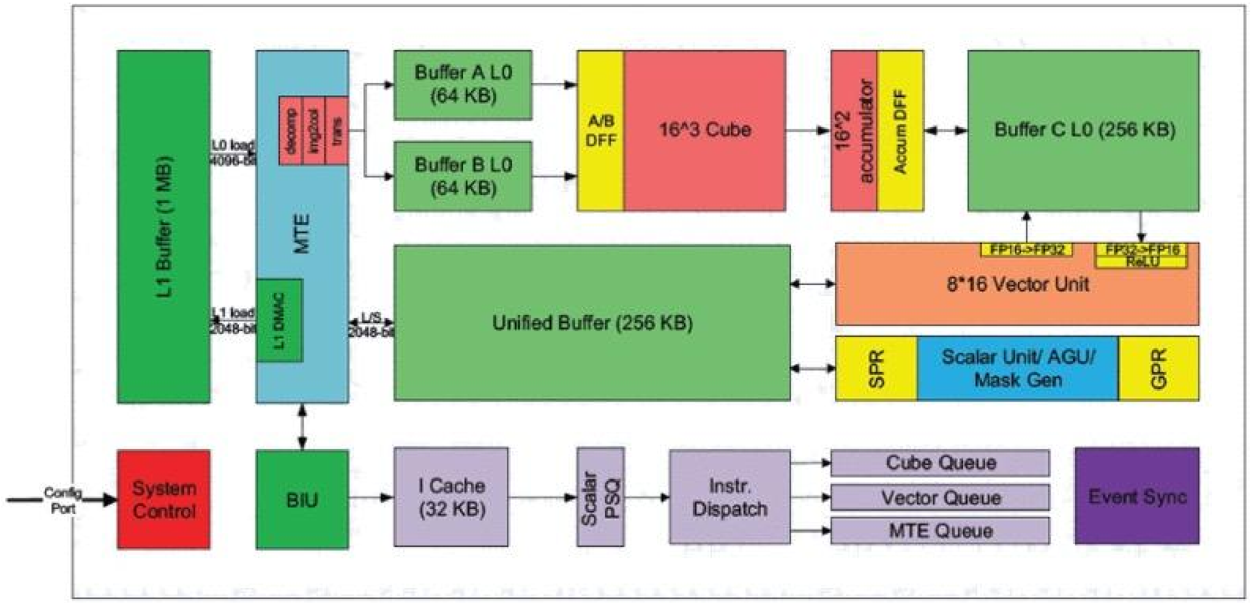
- Modern CPUs massively parallel
- Many specialized chips (heterogenuous)



Google TPUv4
Specialized Engine for NN training
Deployed in cloud
180+ TFLOPS/chip

Water Cooling!

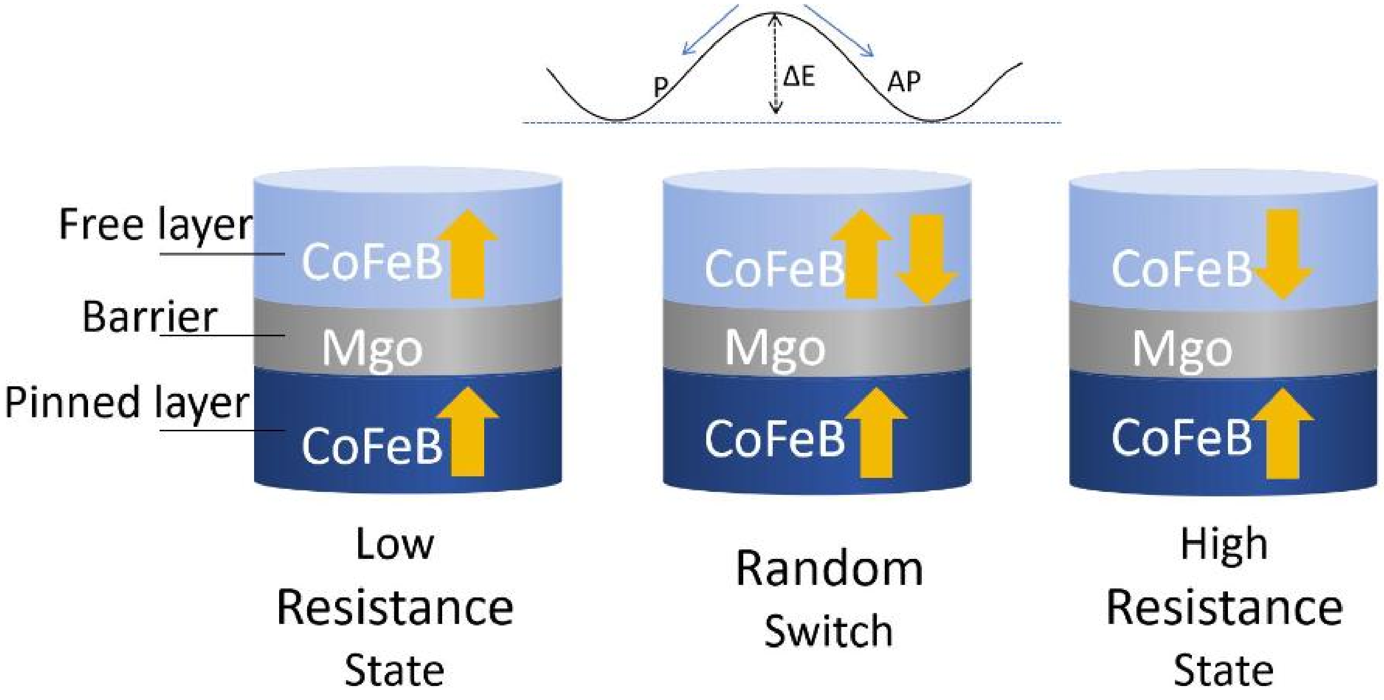
Domain-specific architectures



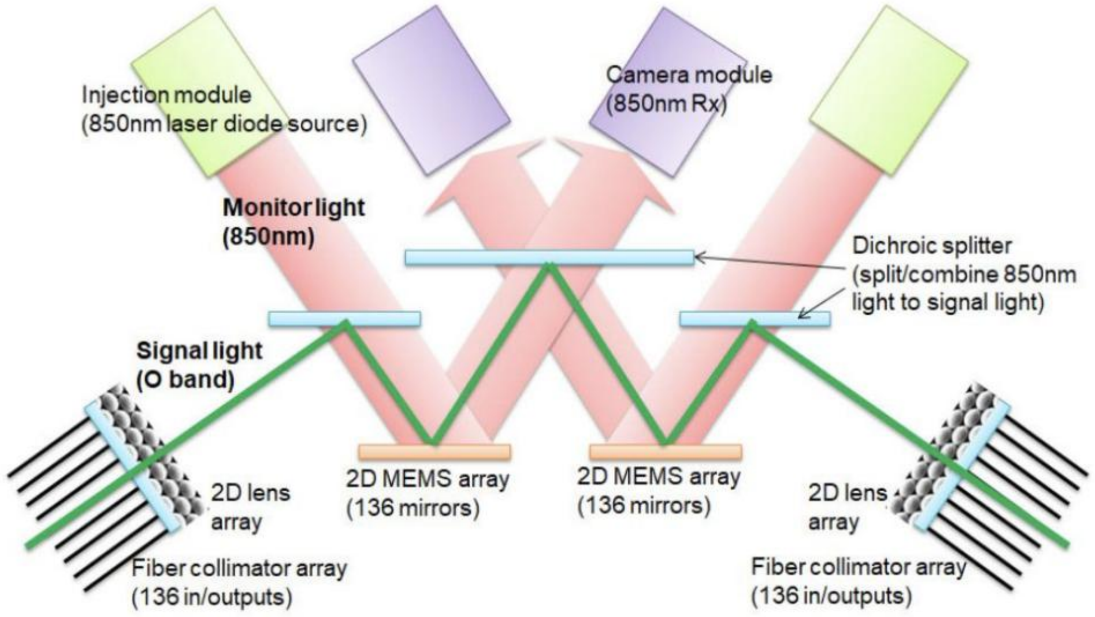
Huawei edge device
Specialized for NN inference
11 TFLOPS FP16; 22 TOPS INT8

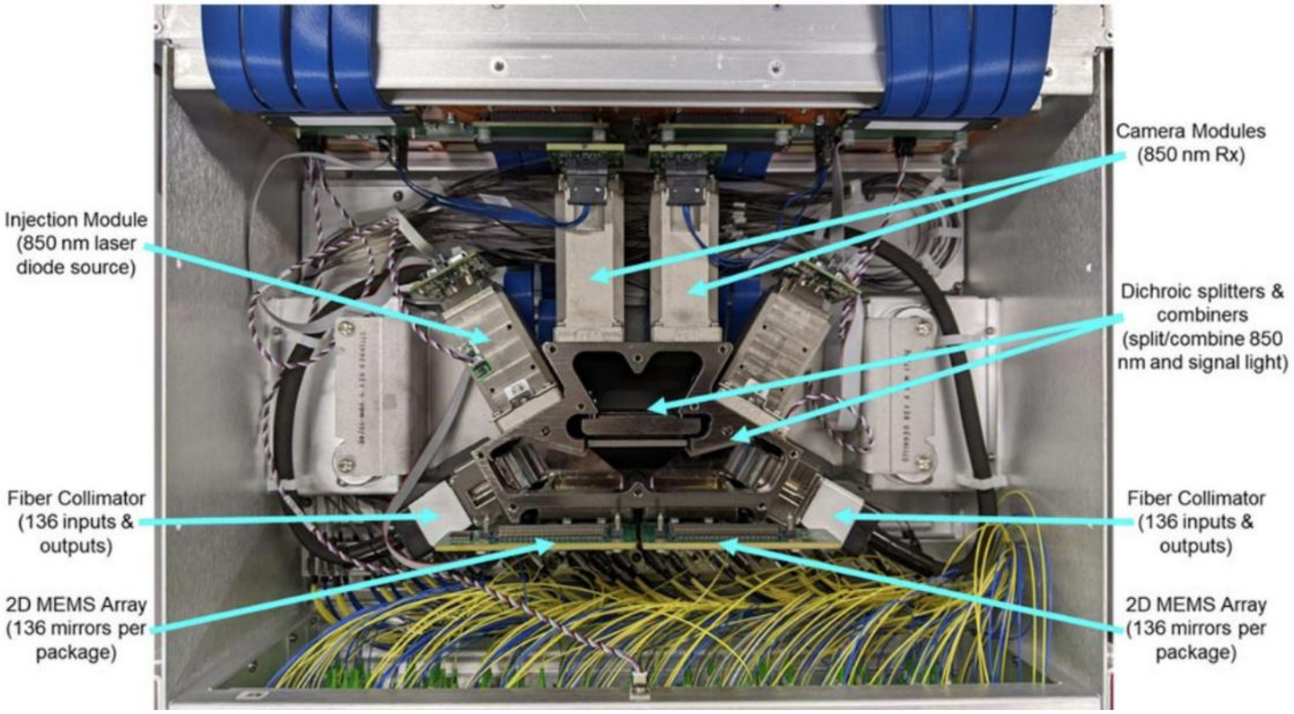
New Conventional Wisdom

- Emerging technologies and devices, e.g.



MTJ (Magnetic Tunnel Junction)
MRAM
Compute-in-memory




Injection Module
(850 nm laser diode source)
Camera Modules
(850 nm Rx)
Dichroic splitters & combiners
(split/combine 850 nm and signal light)
Fiber Collimator
(136 inputs & outputs)
Fiber Collimator
(136 inputs & outputs)
2D MEMS Array
(136 mirrors per package)
2D MEMS Array
(136 mirrors per package)

Optically Reconfigurable Interconnect inside Google AI infrastructure

Summary

- Great ideas in CA
  - Abstraction (Layers of Representation / Interpretation)
  - Moore’s Law (designing through trends)
  - Make the common case fast (Amdahl’s Law)
  - Principle of locality (memory hierarchy)
  - Parallelism (pipeline as special case)
  - Performance measurement and improvement
  - Dependability via redundancy