COMPUTER ORGANIZATION

Lecture 1
Introduction

2025 Spring

baiyh@sustech.edu.cn

Outline

- **Why to Learn Computer Organization**
- Evolution of Computer Architecture
- Concept of Computer and Manufacturing Process
- Great Ideas in Computer Architecture
- Why is Computer Architecture Exciting Today?

Why to Learn Computer Organization?

- Embarrassing if you are a student in CS and can’t make sense of the following terms: DRAM, SRAM, pipelining, cache hierarchies, I/O, virtual memory, …
- Embarrassing if you are a student in CS and can’t decide which processor to buy: 3 GHz P4 or 2.5 GHz Athlon (this course helps us reason about performance/power), …
- First step for chip designers, compiler/ OS writers
- Knowledge of the hardware will help you write better programs

Must a Programmer Care about Hardware?

- Must know how to reason about program performance and energy
- CPU Performance: if we understand how CPU process data, we can enhance the computation efficiency
- Memory management: if we understand how/where data is placed, we can help ensure that relevant data is nearby
- Thread management: if we understand how threads interact, we can write smarter multi-threaded programs
- I/O management

Prerequisites

- Binary numbers
- Read and write basic C/Java programs
- Understand the steps in compiling and executing a program
- Digital Circuit, Logic design:
  - Logical equations, schematic diagrams
  - Combinational vs. sequential logic
  - Finite state machines (FSMs)

What you will learn?

- Major content
  - Basic parts of a computer (processor, memory, disk, etc.)
  - Principles of computer architecture: CPU datapath and control unit design
  - Assembly language programming in RISC-V
  - Memory hierarchies and design
  - I/O organization and design
- Course goals
  - To learn the organizational structures that determine the capabilities and performance of computer systems
  - To understand the interactions between the computer's architecture and its software
  - To understand cost performance trade-offs

Key Topics

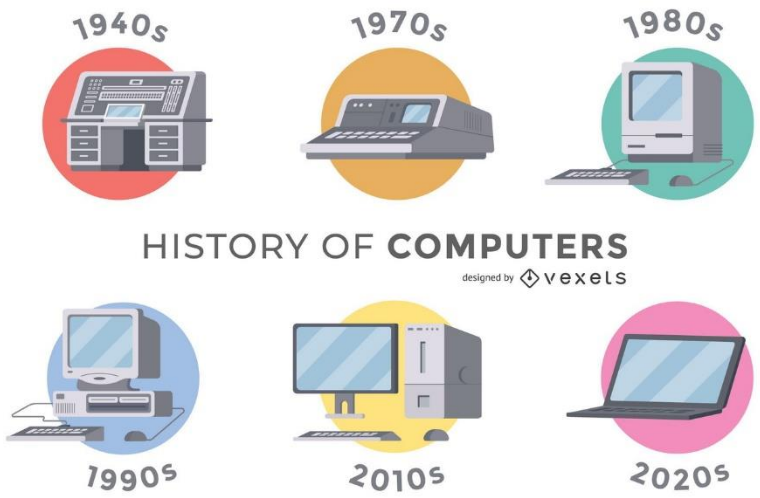
- Introduction (Chapter 1)
  - Basic terms
  - Moore’s Law, power wall
  - Core ideas in computer architecture
- Processors (Chapter 2-4)
  - Assembly language (Chapter 2)
  - Computer arithmetic (Chapter 3)
  - Pipelining (Chapter 4)
- Memory (Chapter 5)
- Parallel Processors (Chapter 6)

Outline

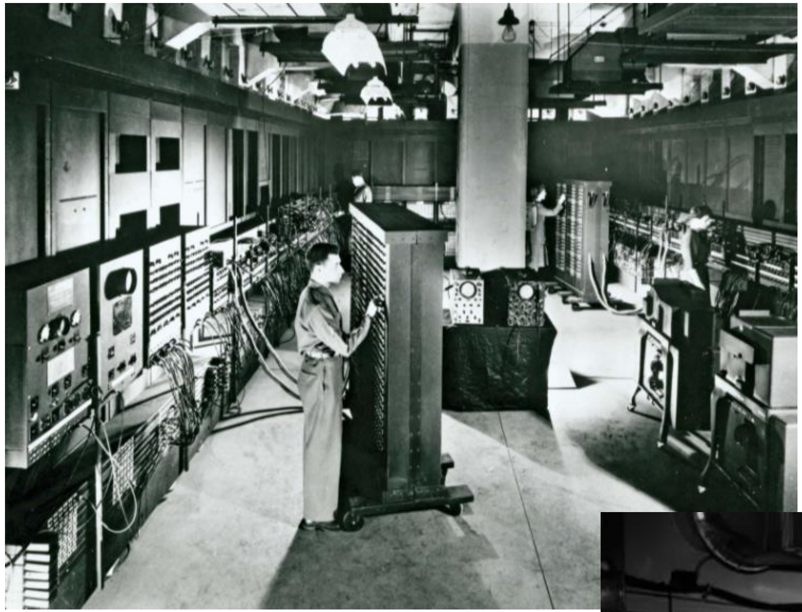
- Why to Learn Computer Organization
- **Evolution of Computer Architecture**
- Concept of Computer and Manufacturing Process
- Great Ideas in Computer Architecture
- Why is Computer Architecture Exciting Today?

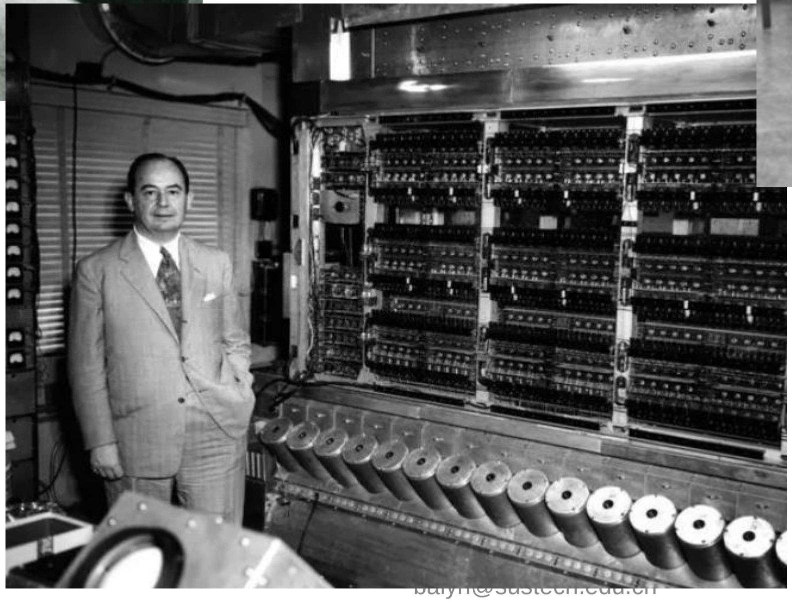
The Evolution of Computers

- First Generation (1940s - 1950s)
  - Vacuum Tubes
- Second Generation (1950s - 1960s)
  - Transistors
- Third Generation (1960s - 1970s)
  - Integrated Circuits
- Fourth/Now Generation (1970s - Present)
  - Microprocessors
  - Artificial Intelligence



Old School Computers





New School Computers

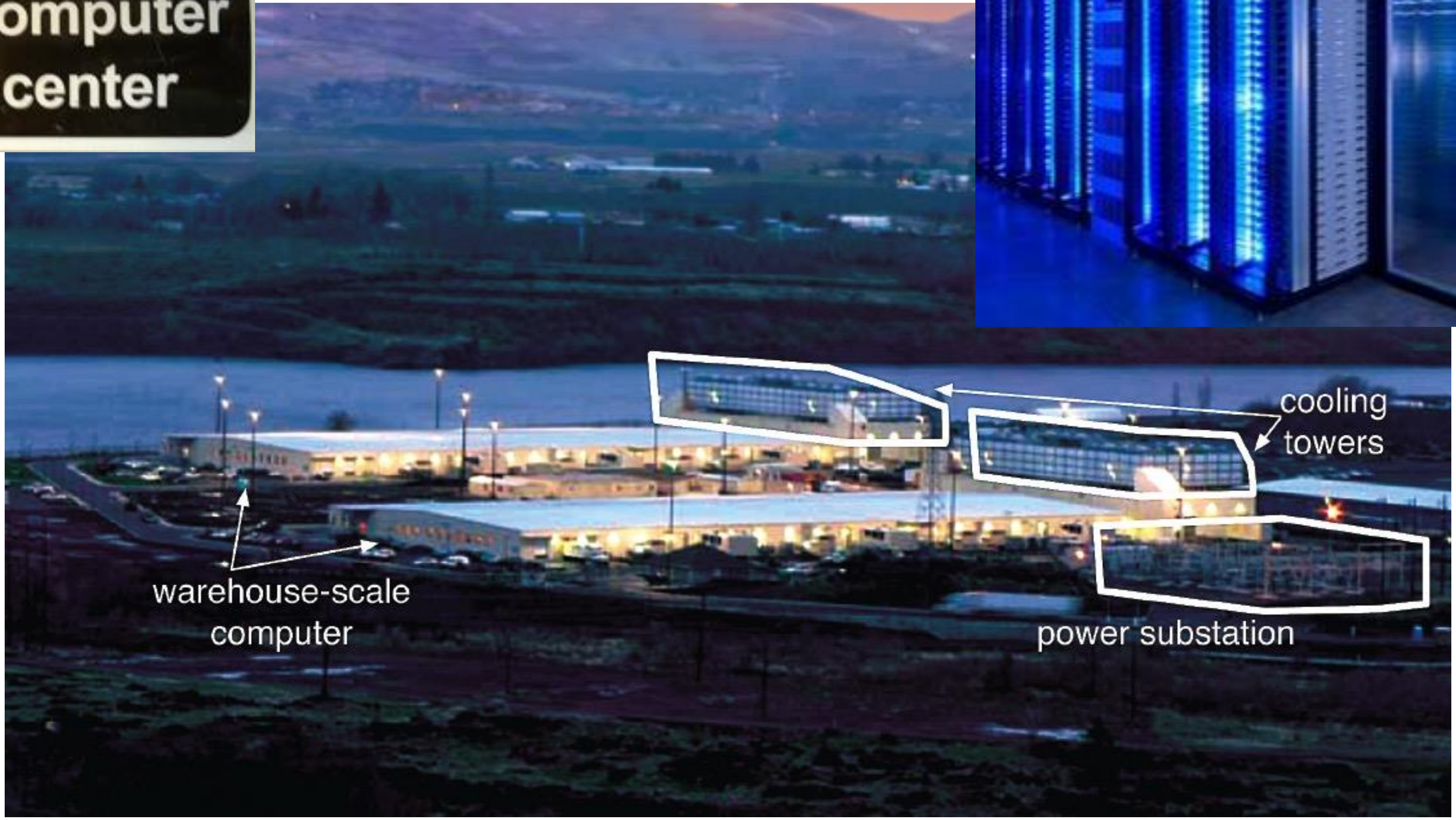
From the Small…

New School Computers

To the very small...

New School Computers

To the big…



Classes of Computers

- Personal computers
  - General purpose, variety of software
  - Subject to cost/performance tradeoff
- Server computers
  - Network based
  - High capacity, performance, reliability
  - Range from small servers to building sized
- Supercomputers
  - High-end scientific and engineering calculations
  - Highest capability but represent a small fraction of the overall computer market
  - https://www.top500.org/lists/top500
- Embedded computers
  - Hidden as components of systems
  - Stringent power/performance/cost constraints

Classes of Computers in The PostPC Era

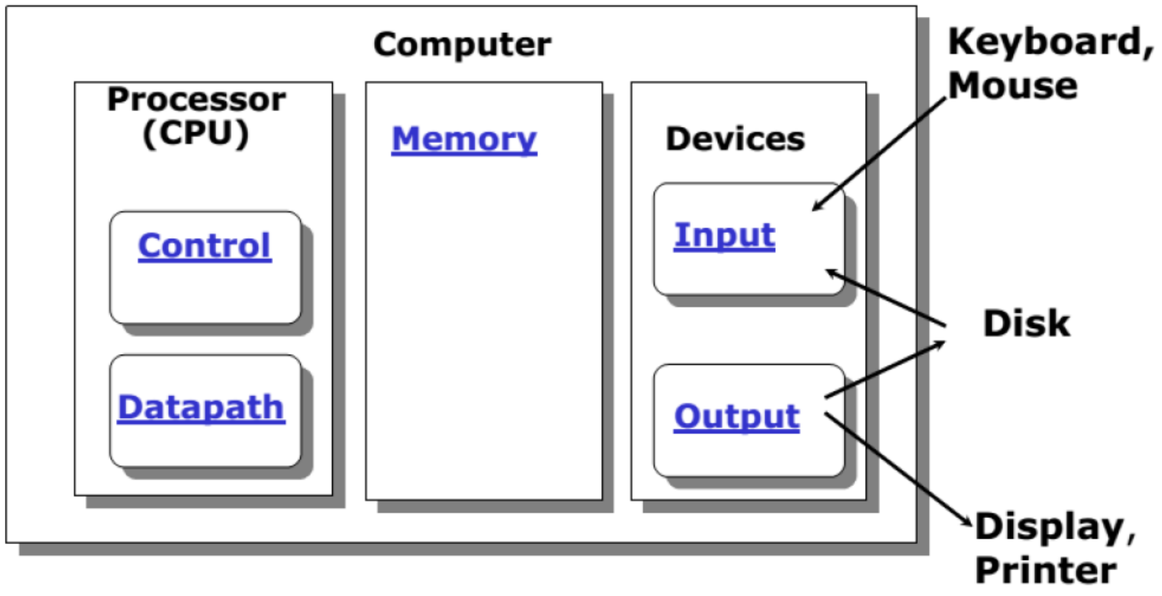
- Personal Mobile Device (PMD)
  - Battery operated
  - Connects to the Internet
  - Hundreds of dollars
  - Internet of Things
- Cloud computing
  - Warehouse Scale Computers (WSC)
  - Software as a Service (SaaS)
  - Data Centers
- New Architecture for AI Era
  - More Suitable for deep learning
  - GPU, TPU, NPU
  - AI Chips

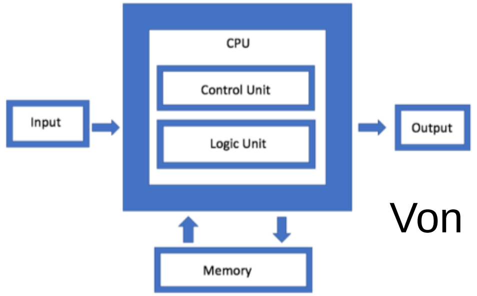
Outline

- Why to Learn Computer Organization
- Evolution of Computer Architecture
- **Concept of Computer and Manufacturing Process**
- Great Ideas in Computer Architecture
- Why is Computer Architecture Exciting Today?

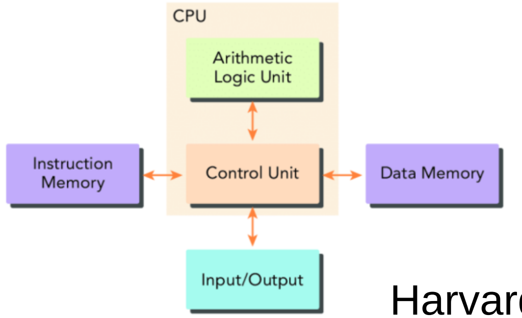
Components of a Computer

- Same components for all kinds of computer:
  - Input, Output Device, Memory, Processor: (Control, Datapath/ALU)
  - Von Neumann Architecture vs. Harvard Architecture





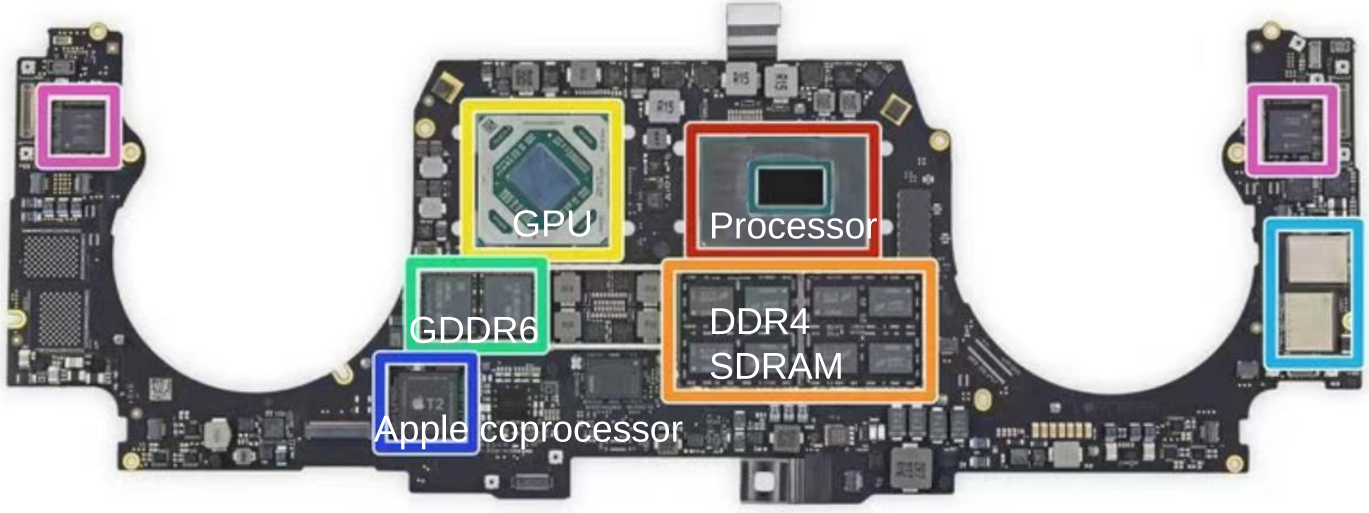
Von Neumann



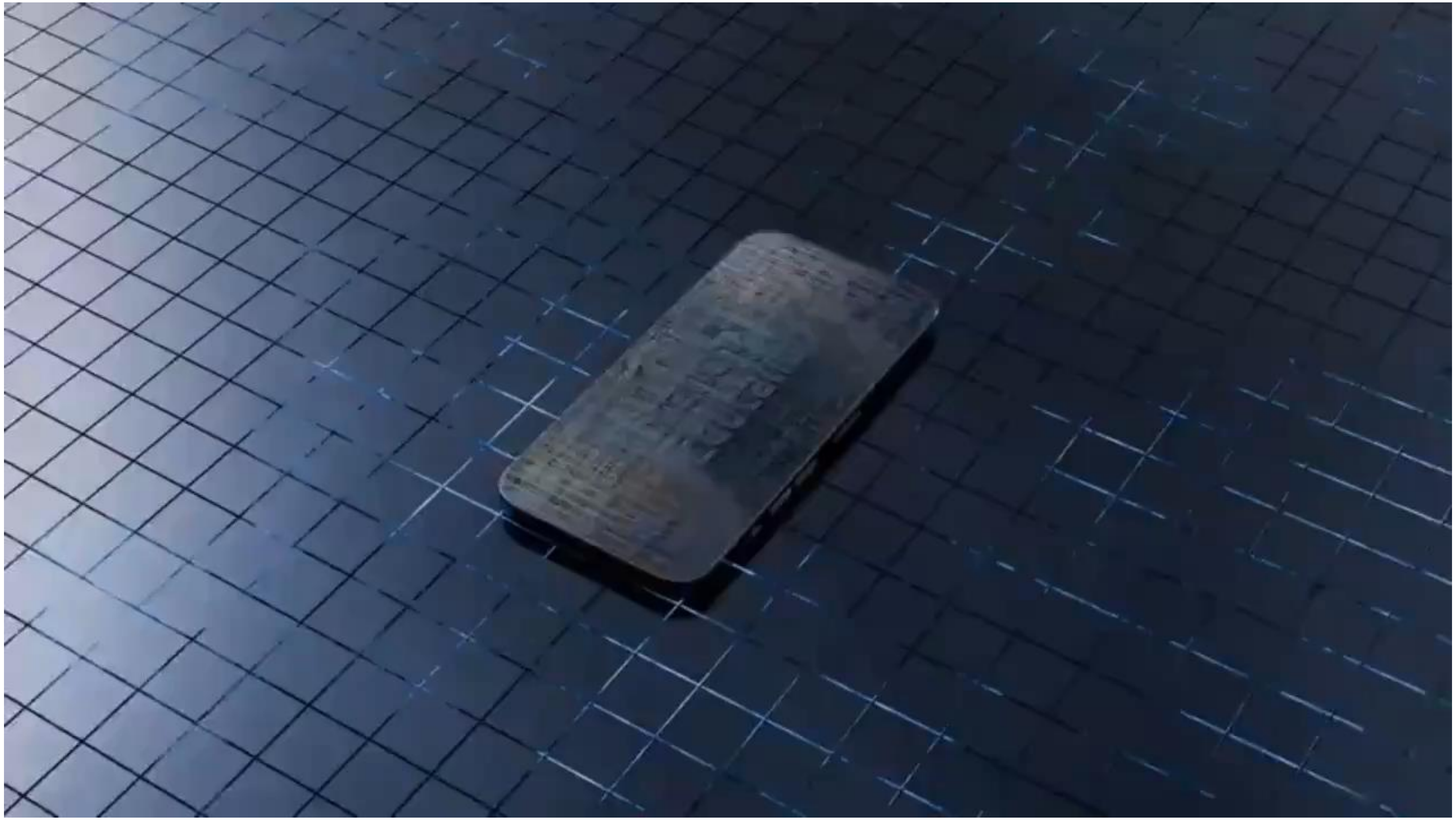
Harvard

Teardown of MacBook

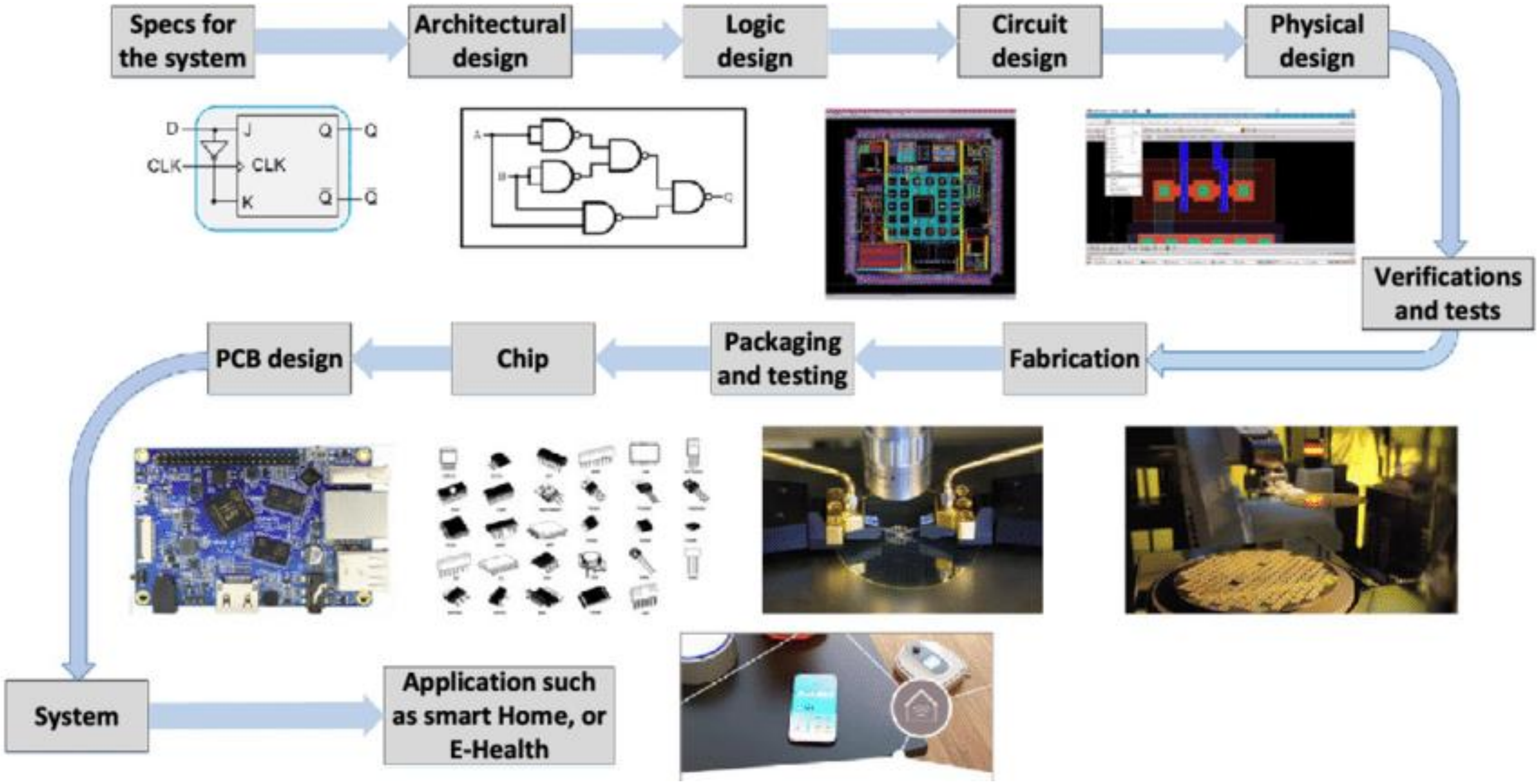
- 16" LED-backlit IPS Retina display
- Keyboard and Touch Bar
- 2.6 GHz 6-core Intel Core i7
- 16 GB of 2666 MHz DDR4 SDRAM
- 512 GB SSD
- 100 Watt-hour battery
- Speaker and microphone



Semiconductor manufacturing process chain

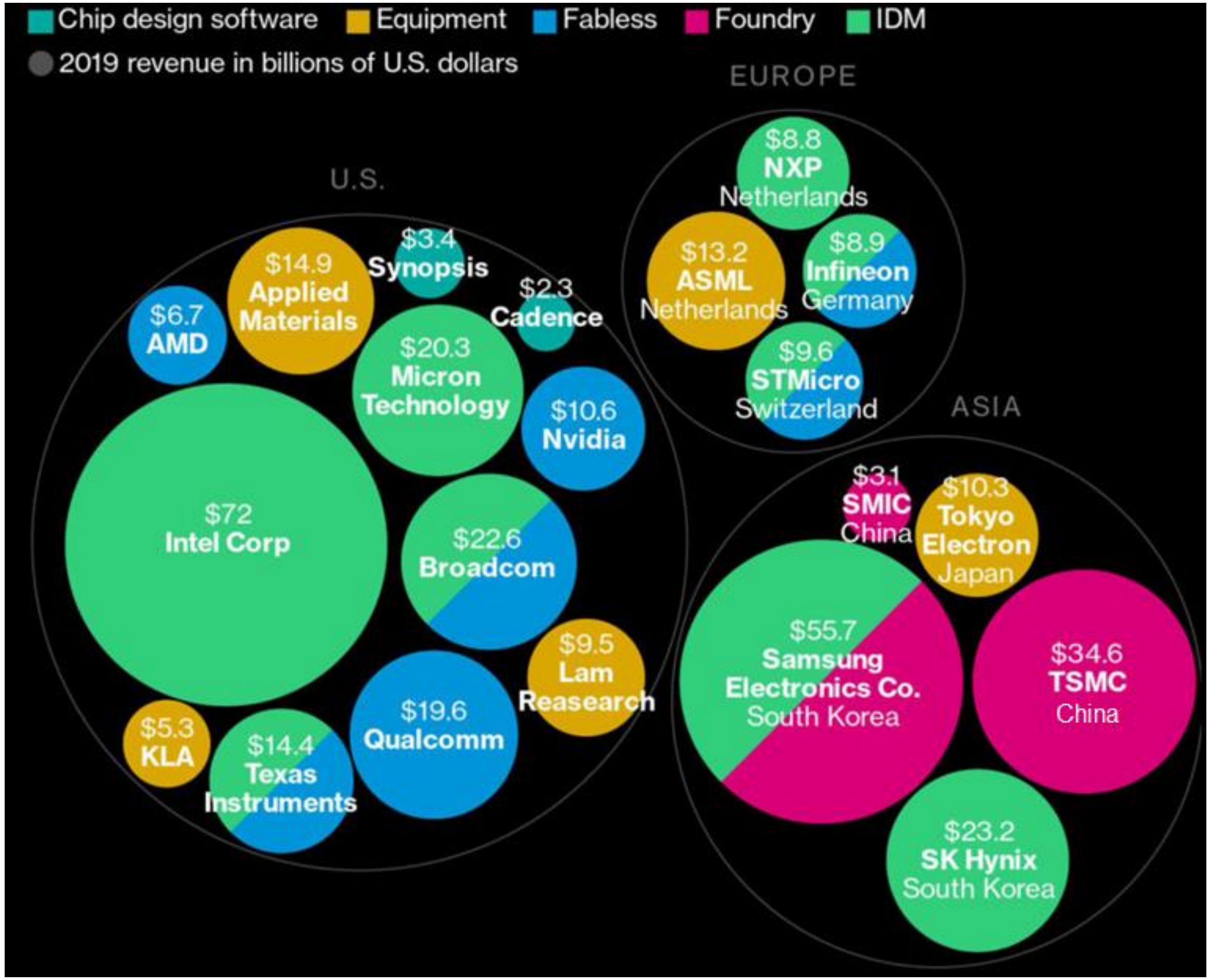


Semiconductor manufacturing process chain



Chip Industry

- Key players in chip industry
  - Intel
  - AMD
  - Qualcomm
  - Samsung
  - TSMC
  - Broadcom
  - Nvidia
  - ASML
  - …



IDM: Integrated Design and Manufacture

Machine Organization

Software

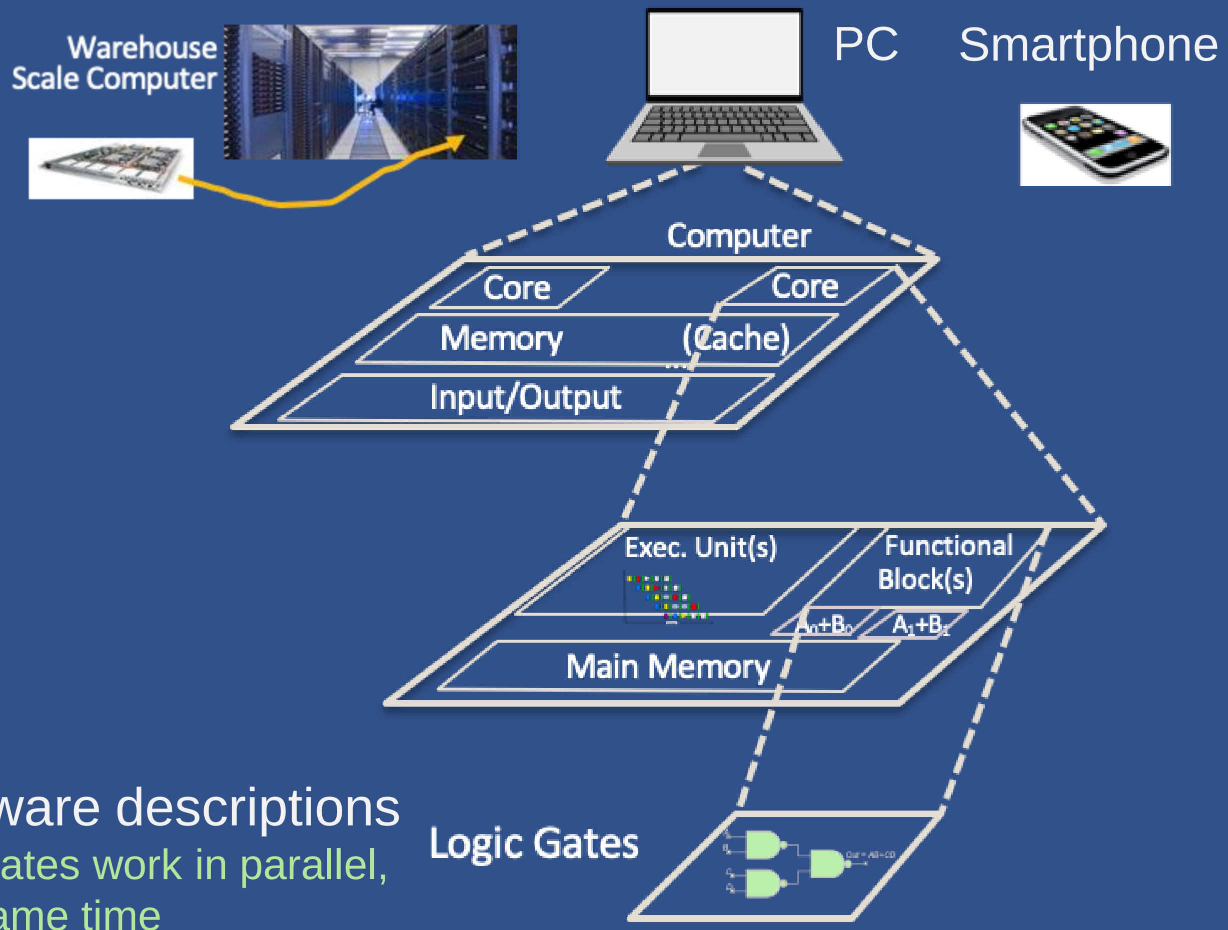
Parallel Requests
Assigned to computer
e.g., Search “Cats”

Parallel Threads
Assigned to core e.g., Lookup, Ads

Parallel Instructions
>1 instruction @ one time
e.g., 5 pipelined instructions

Parallel Data
>1 data item @ one time
e.g., Add of 4 pairs of words

Hardware



Hardware descriptions
All gates work in parallel, at same time

Outline

- Why to Learn Computer Organization
- Evolution of Computer Architecture
- Concept of Computer and Manufacturing Process
- **Great Ideas in Computer Architecture**
- Why is Computer Architecture Exciting Today?

Great Ideas in Computer Architecture

1. Abstraction (Layers of Representation/Interpretation)
2. Moore’s Law
3. Principle of Locality/Memory Hierarchy
4. Parallelism
5. Performance Measurement & Improvement
6. Dependability via Redundancy

Great Idea #1: Abstraction (Levels of Representation/Interpretation)

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

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

Compiler

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

```
lw    x3, 0(x10)
lw    x4, 4(x10)
sw    x4, 0(x10)
sw    x3, 4(x10)
```

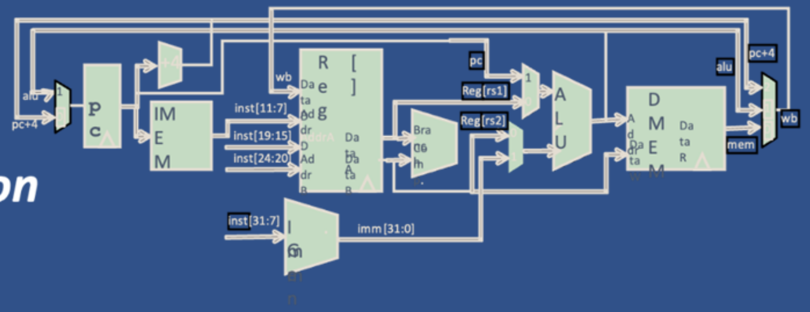
Assembler

Machine Language Program (RISC-V)

```
1000 1101 1110 0010 0000 0000 0000 0000
1000 1110 0001 0000 0000 0000 0000 0100
1010 1110 0001 0010 0000 0000 0000 0000
1010 1101 1110 0010 0000 0000 0000 0100
```

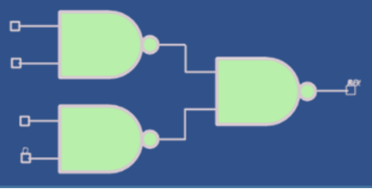
Anything can be a ***number***—data, instructions, etc.

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

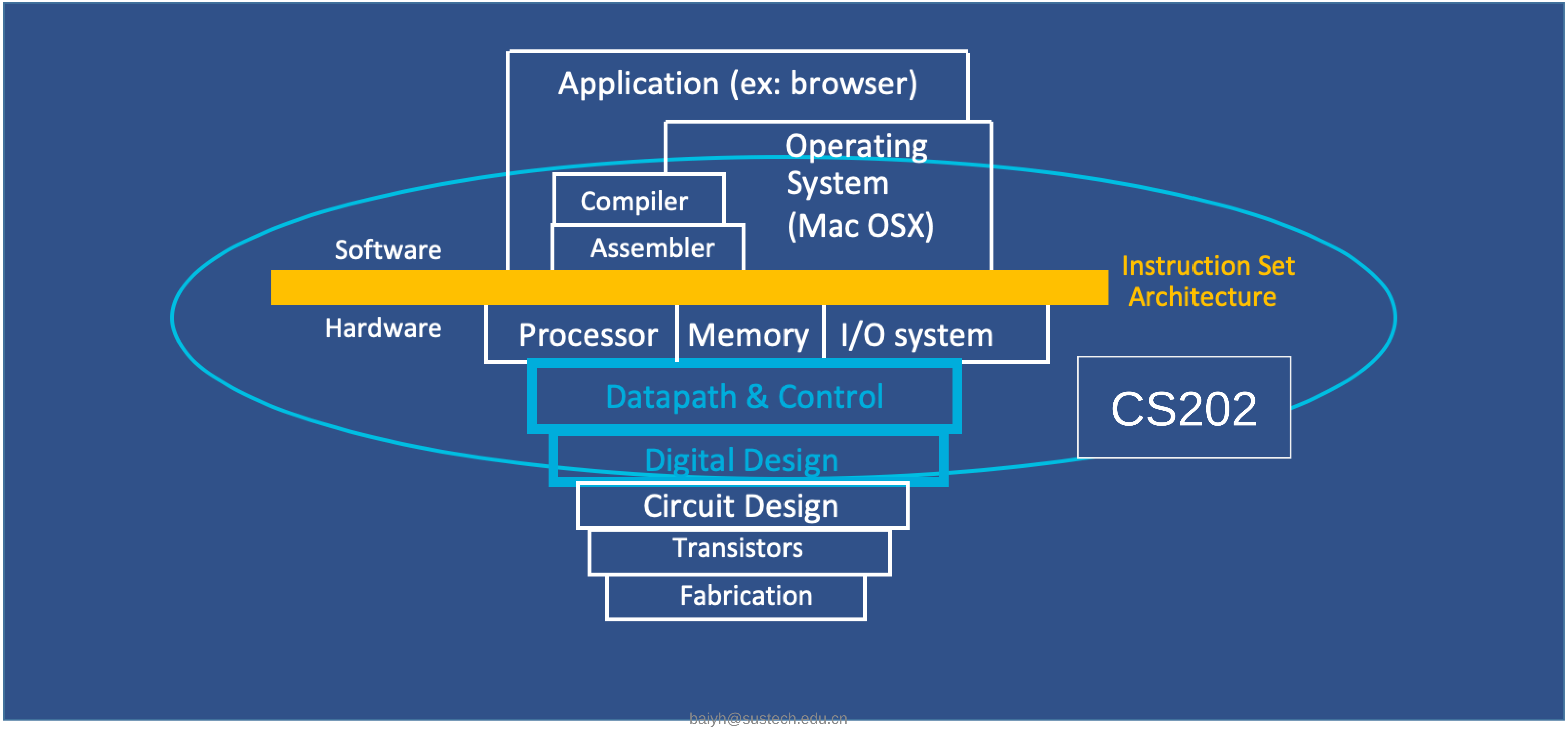


Architecture Implementation

Logic Circuit Description (Circuit Schematic Diagrams)

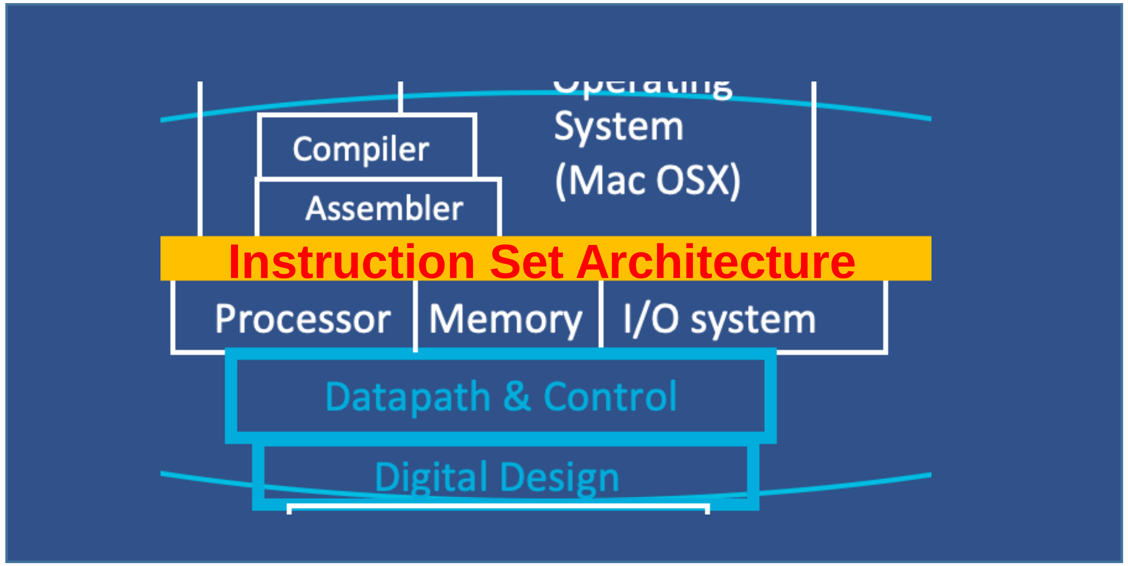


Abstractions and Instruction Set Architecture

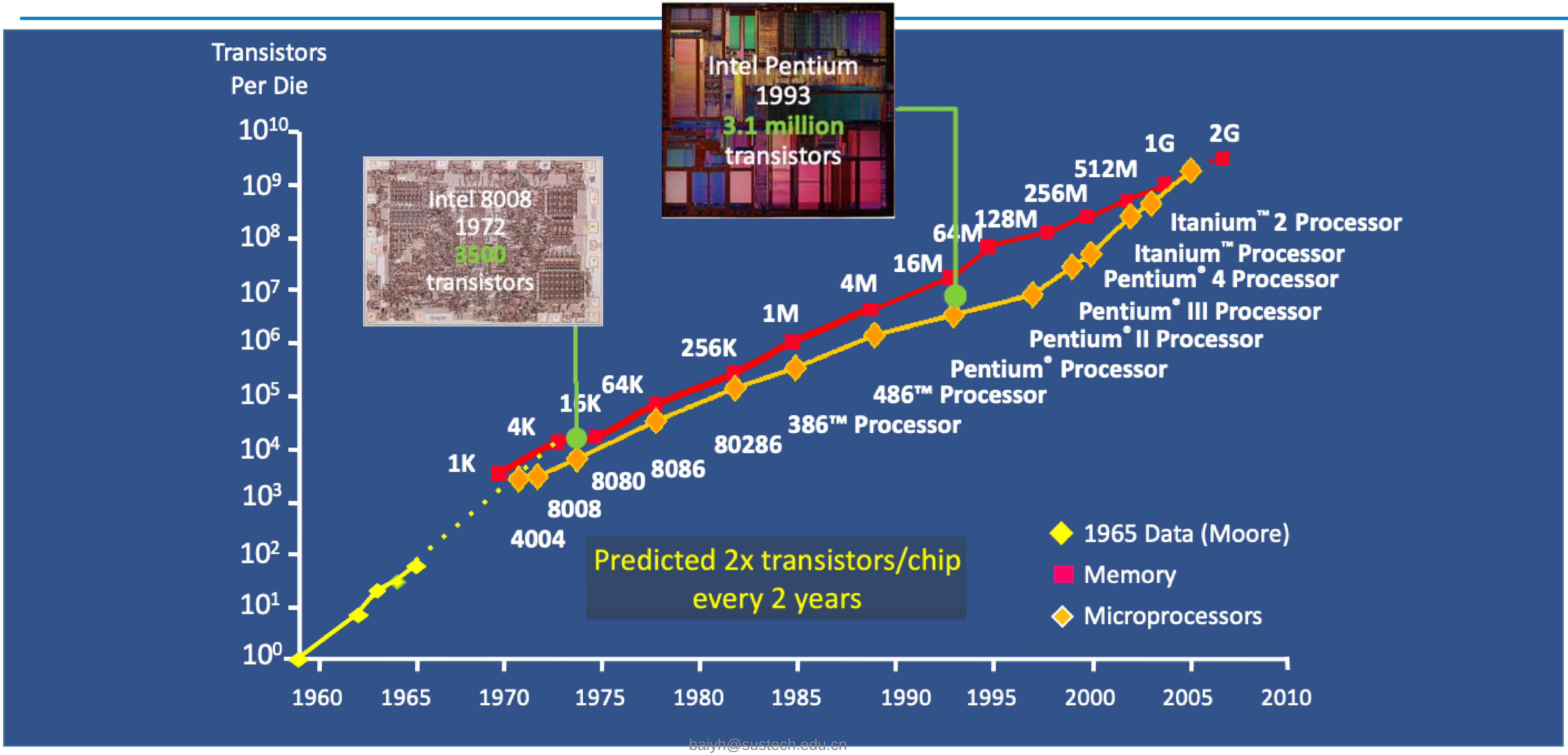


Abstractions and Instruction Set Architecture

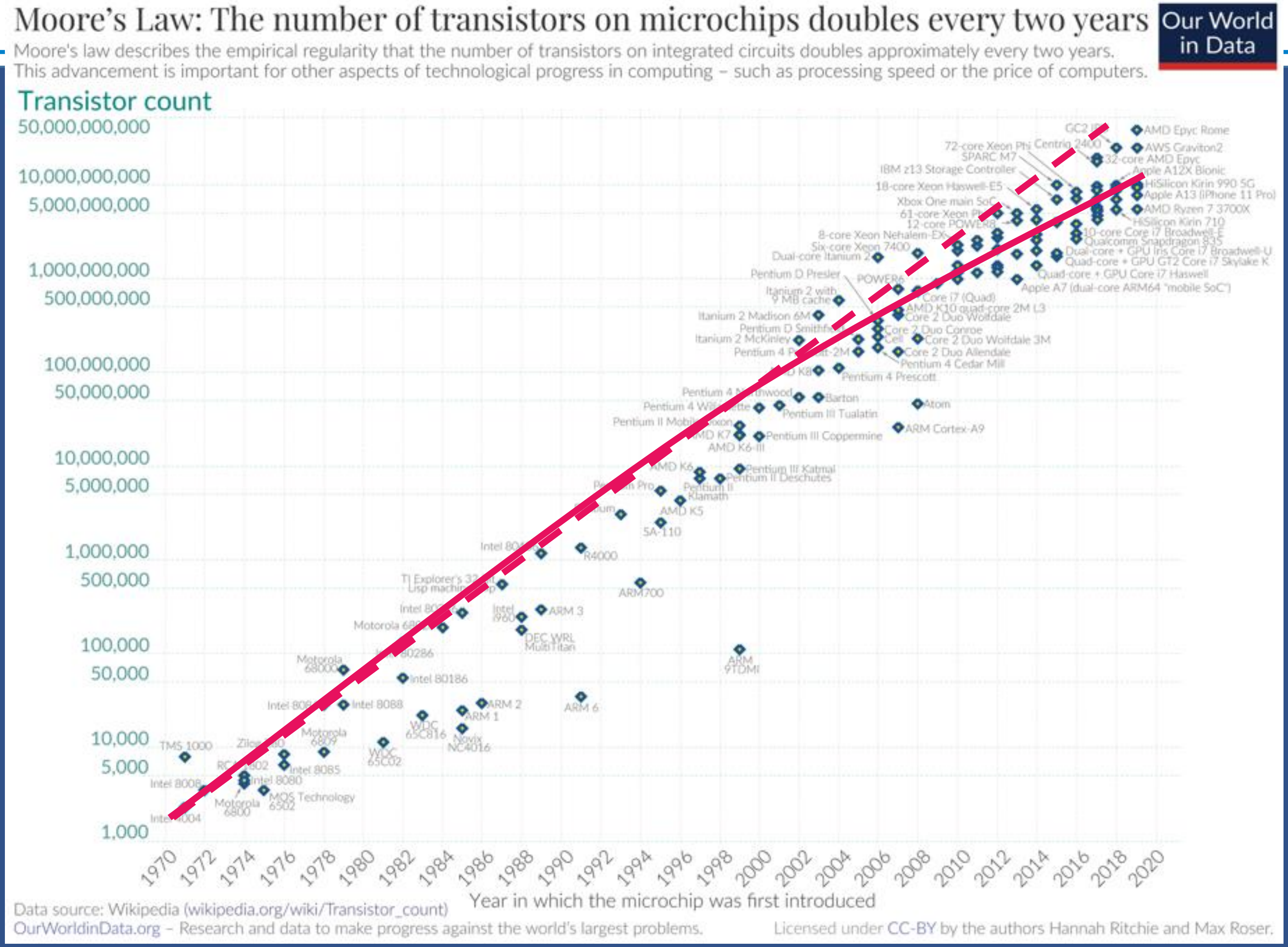
- A set of assembly language instructions (ISA) provides a link between software and hardware.
- Given an instruction set, software programmers and hardware engineers work more or less independently.
- Common types of ISA: RISC, CISC
- Examples:
  - IBM370/X86 (CISC)
  - RISC-V (RISC)
  - MIPS (RISC)
  - ARM (RISC)



Great Idea #2: Moore's Law



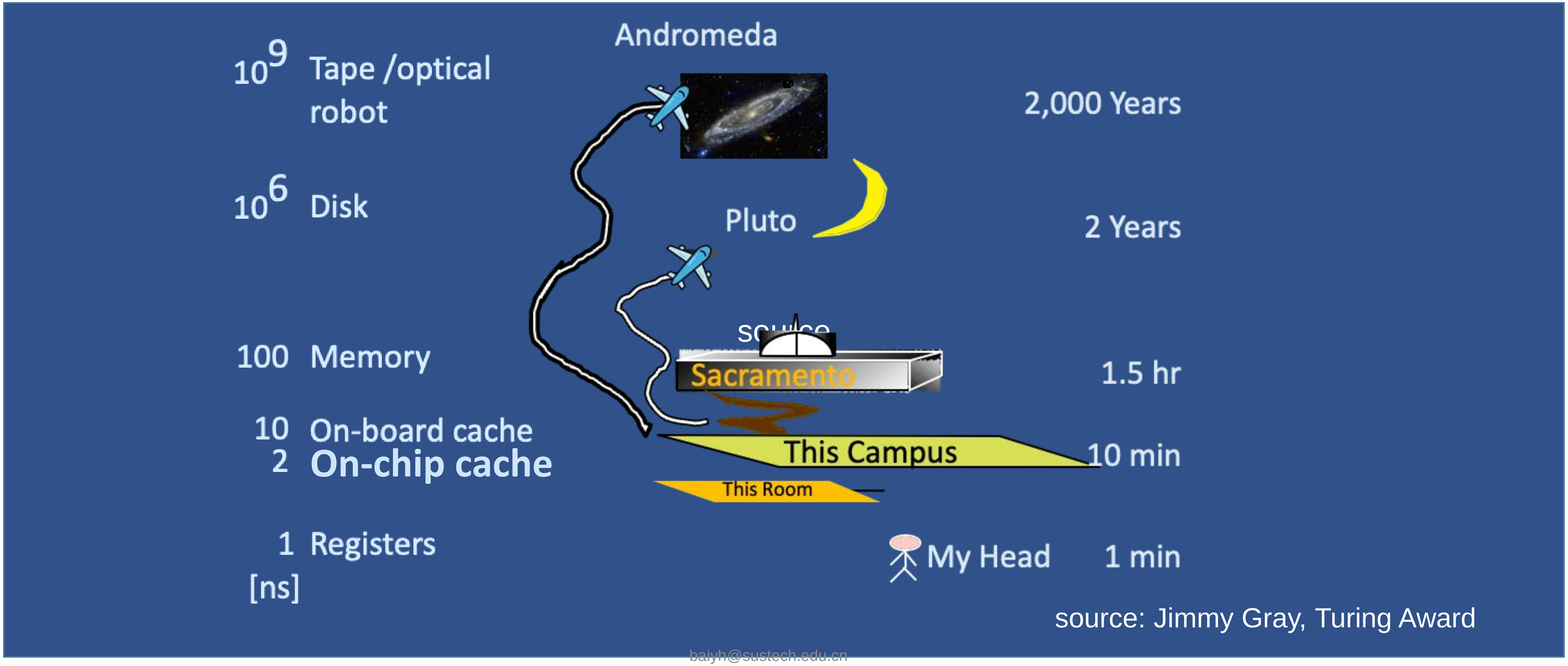
Moore's Law…?



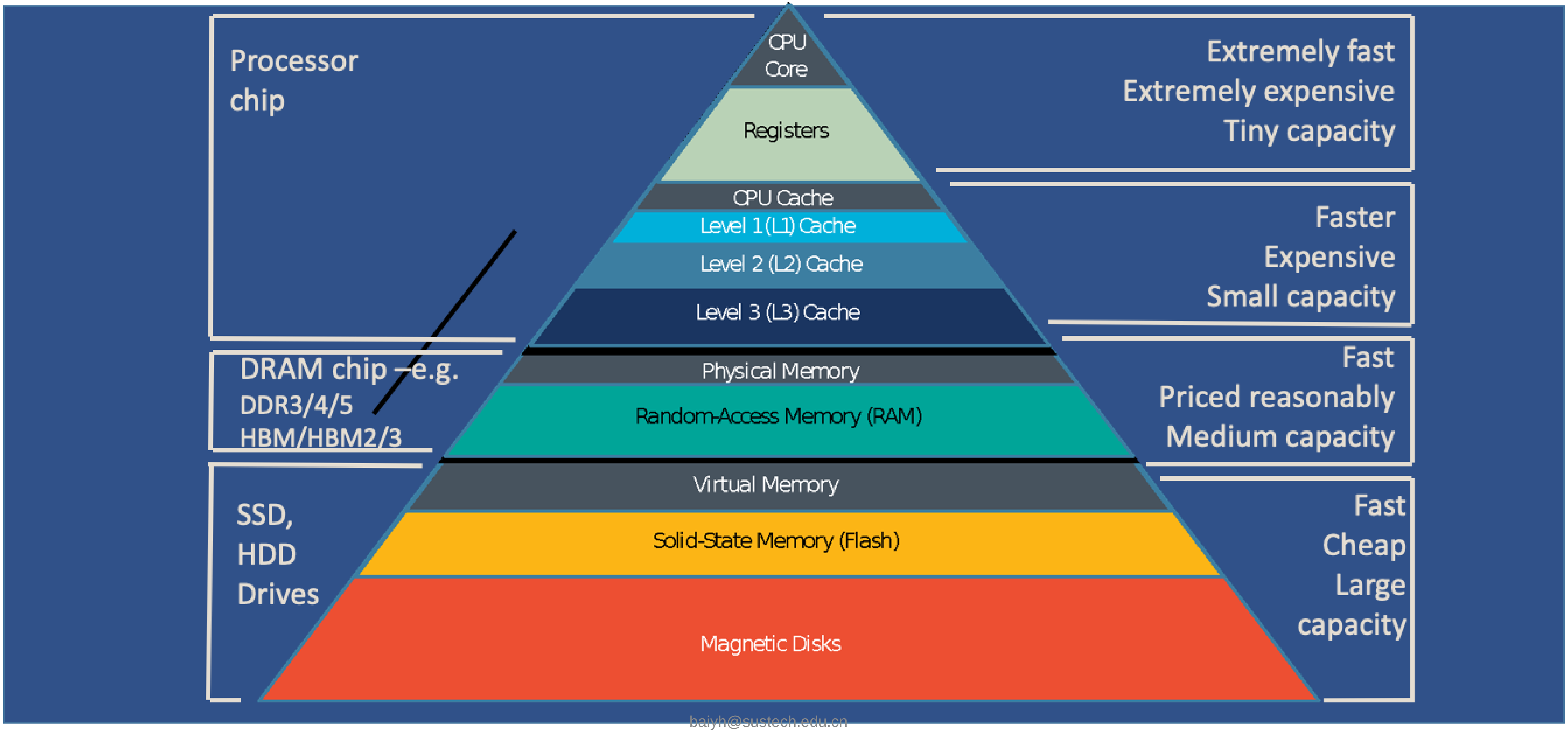
Seems to be tapering…? (more later)

Great Idea #3: Principle of Locality / Memory Hierarchy

- Storage Latency Analogy: How Far Away is the Data?



Great Idea #3: Principle of Locality / Memory Hierarchy



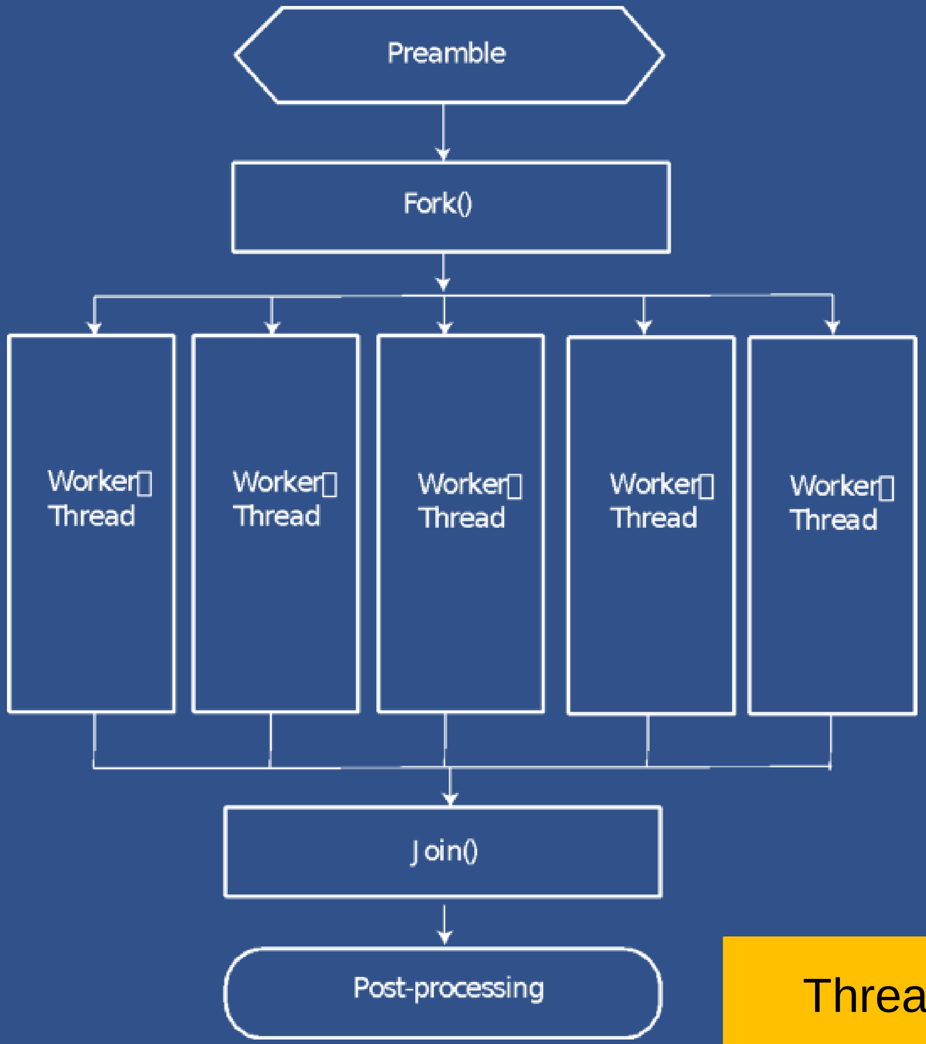
Great Idea #4: Parallelism

	Time 1	Time 2	Time 3	Time 4	Time 5	Time 6	Time 7	Time 8	Time 9
Instruction 1	Instruction Fetch	Instruction Decode	Execute	Memory Access	Write Back				
Instruction 2		Instruction Fetch	Instruction Decode	Execute	Memory Access	Write Back			
Instruction 3			Instruction Fetch	Instruction Decode	Execute	Memory Access	Write Back		
Instruction 4				Instruction Fetch	Instruction Decode	Execute	Memory Access	Write Back	
Instruction 5					Instruction Fetch	Instruction Decode	Execute	Memory Access	Write Back

In time slot 3,
Instruction 1 is being executed
Instruction 2 is being decoded
And Instruction 3 is being fetched from memory

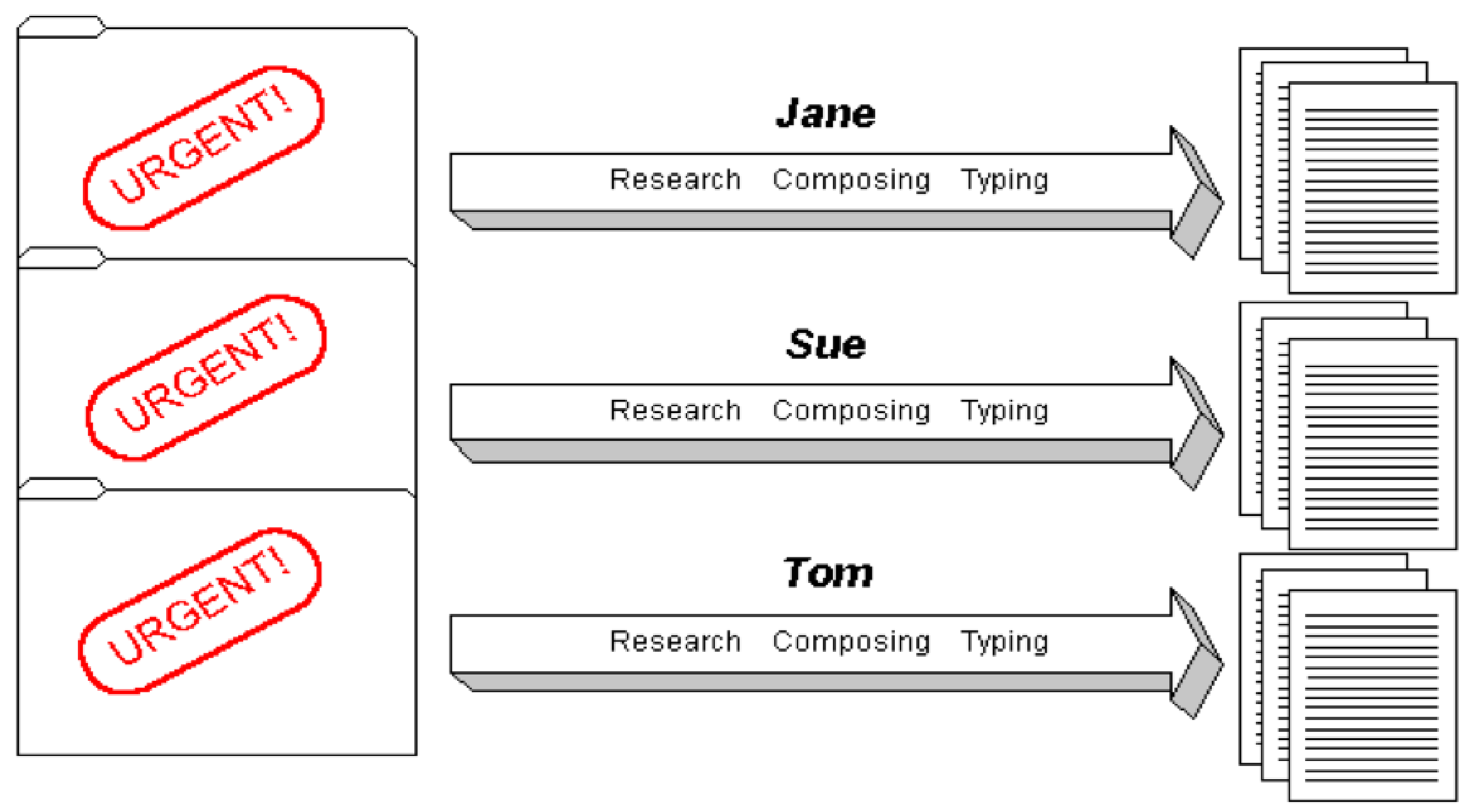
Instruction-Level Parallelism

Great Idea #4: Parallelism



Thread-Level Parallelism

Great Idea #4: Parallelism



Data-Level Parallelism

Amdahl's Law

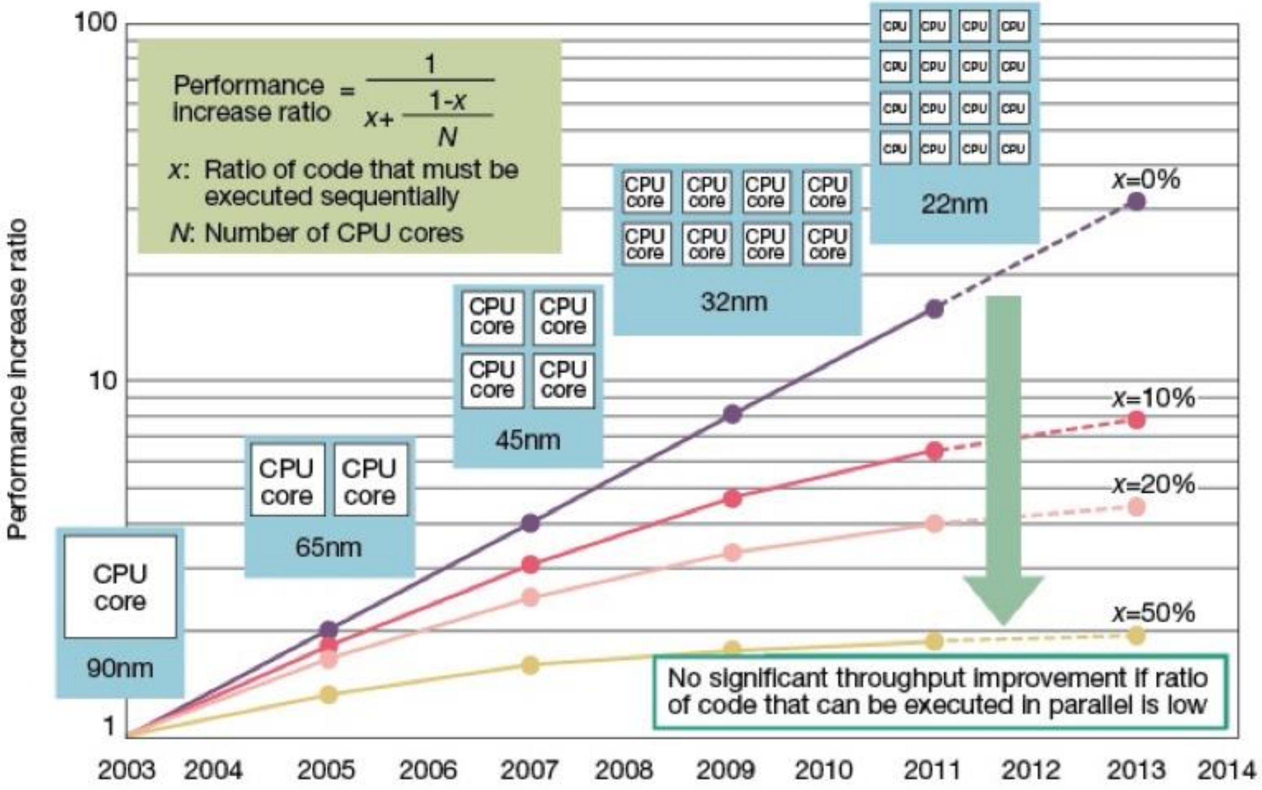


Fig 3 Amdahl's Law an Obstacle to Improved Performance Performance will not rise in the same proportion as the increase in CPU cores. Performance gains are limited by the ratio of software processing that must be executed sequentially. Amdahl's Law is a major obstacle in boosting multicore microprocessor 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.

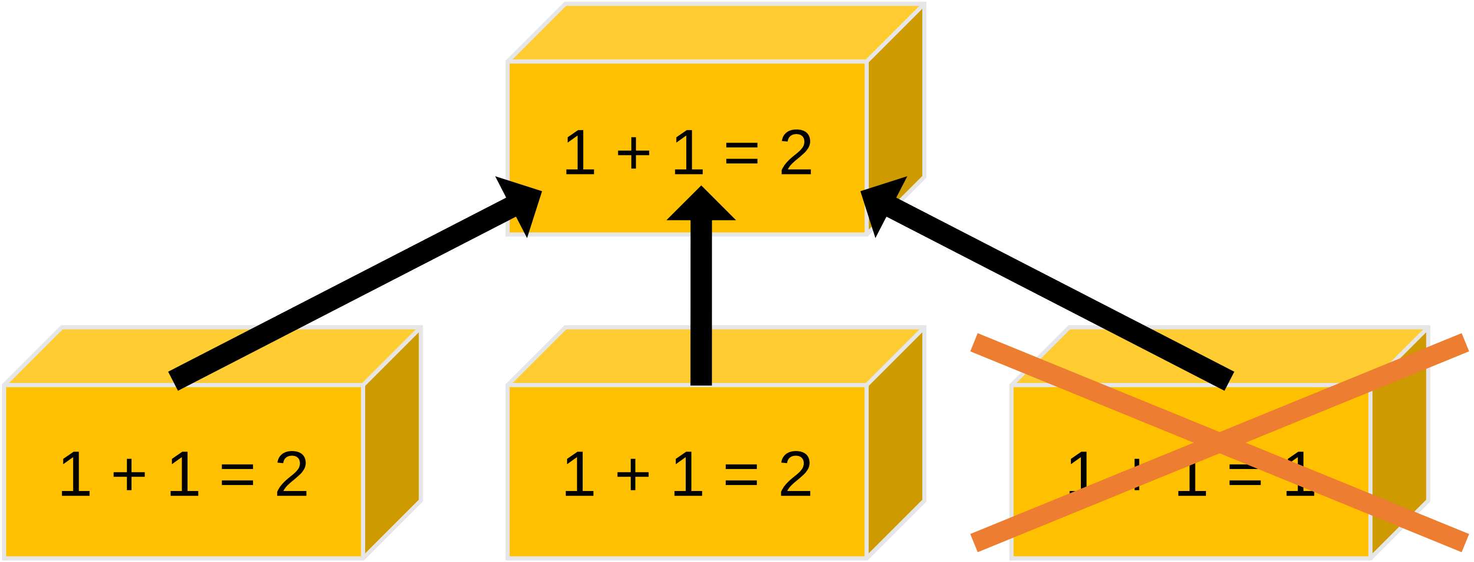
Great Idea #5: Performance Measurement & Improvement

- Match application to underlying hardware to exploit:
  - Locality;
  - Parallelism;
  - Special hardware features, like specialized instructions (e.g., matrix manipulation).

- Latency/Throughput:
  - How long to set the problem up and complete it (or how many tasks can be completed in a given time)
  - How much faster does it execute once it gets going

Great Idea #6: Dependability via Redundancy

- Unintended transistor behavior can be caused by unintended electron flow from cosmic rays (among other reasons)!
- Design with redundancy so that a failing piece doesn't make the whole system fail.



Great Idea #6: Dependability via Redundancy

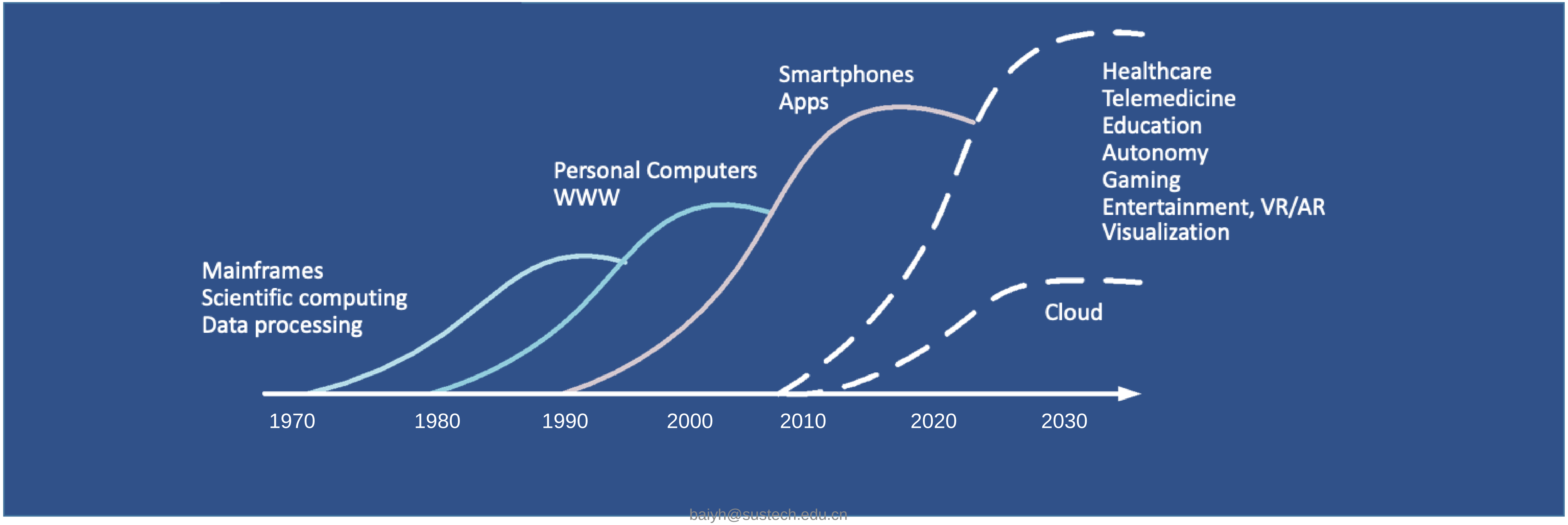
- Applies to everything from datacenters to storage to memory…to instructors!
  - Redundant datacenters so that can lose 1 datacenter but Internet service stays online
  - Redundant disks so that can lose 1 disk but not lose data (Redundant Arrays of Independent Disks/RAID)
  - Redundant memory bits so that can lose 1 bit but no data (Error Correcting Code/ECC Memory)
  - Increasing transistor density reduces the cost of redundancy

Outline

- Why to Learn Computer Organization
- Evolution of Computer Architecture
- Concept of Computer and Manufacturing Process
- Great Ideas in Computer Architecture
- **Why is Computer Architecture Exciting Today?**

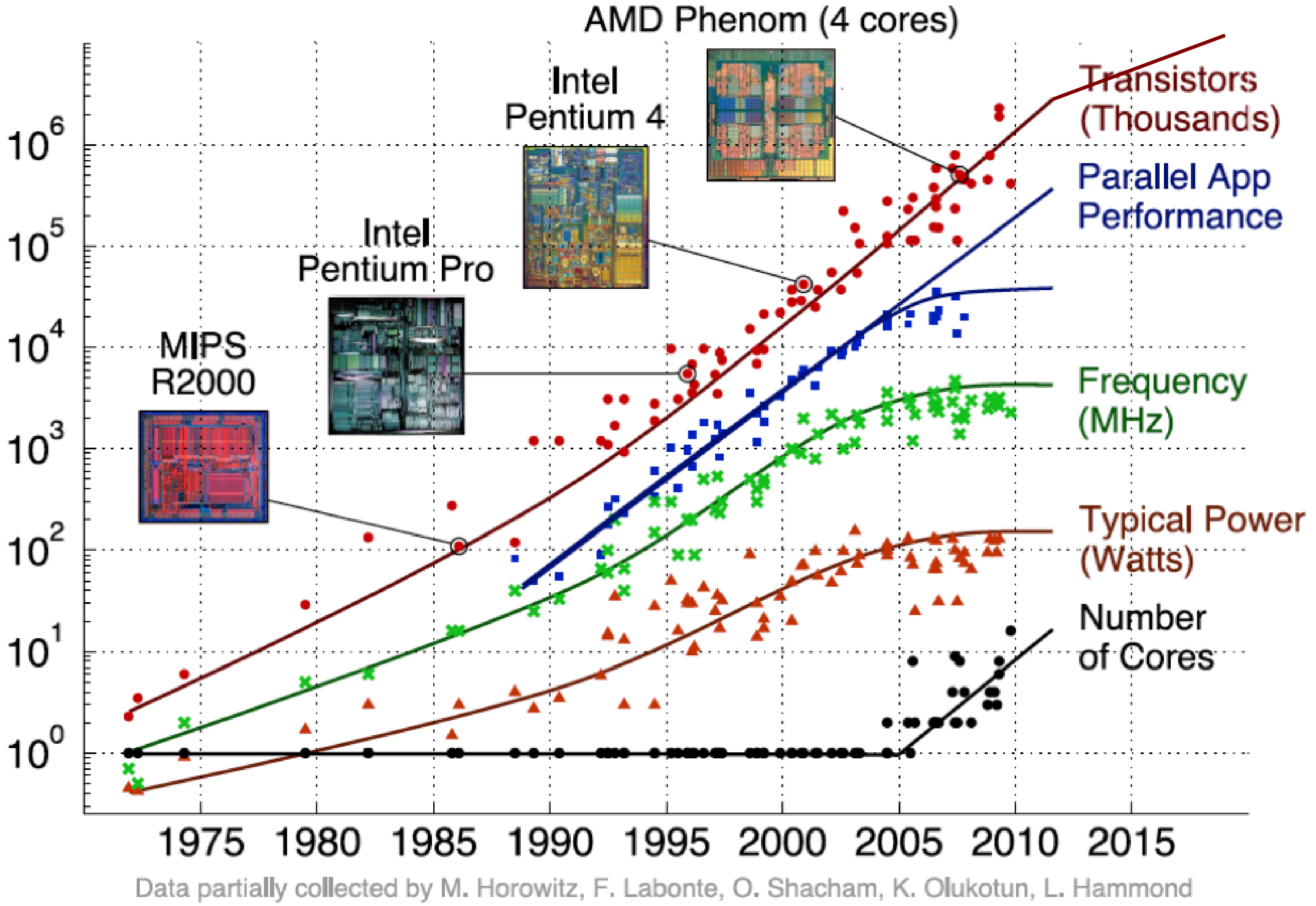
Reason: Era of Domain-Specific Computing

- Number of deployed devices continues to grow, but there is no single killer application.
  - Diversification of needs, architectures
  - Machine learning is common for most domains



Reason: Changing Constraints

- Moore’s Law ending
- Power limitations
- Amdahl’s Law

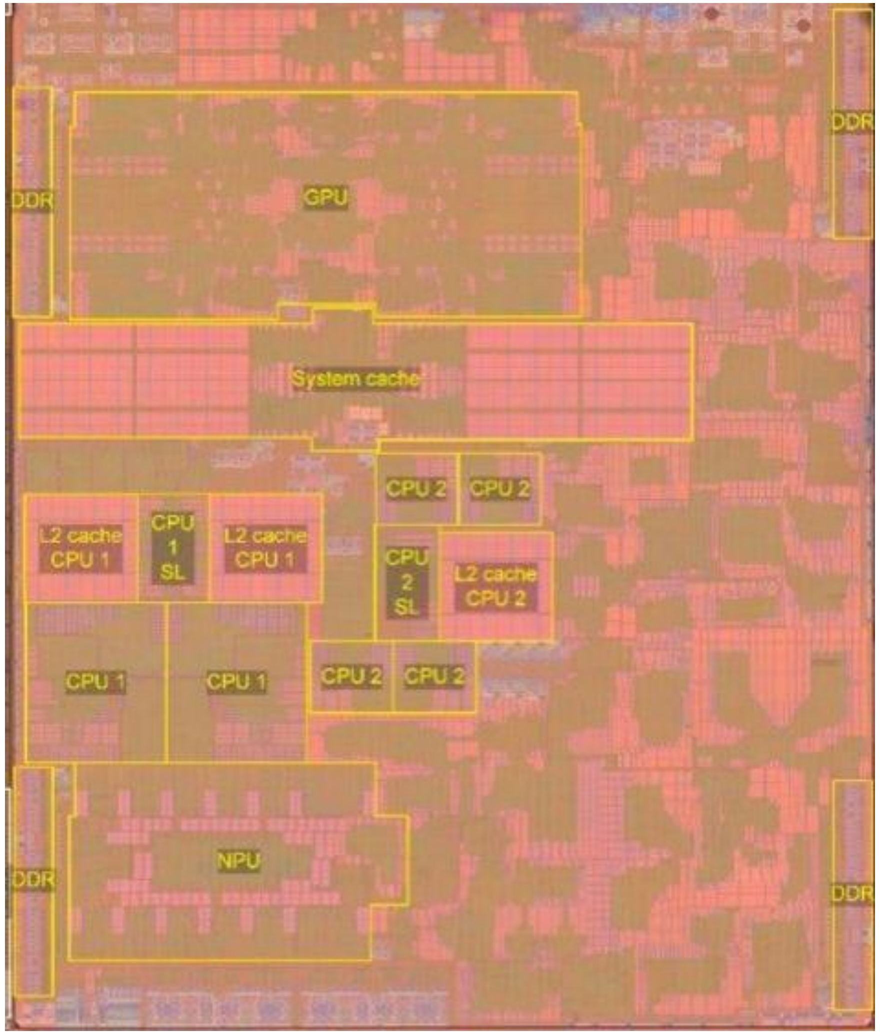


Old Conventional Wisdom

- Faster, cheaper, lower-power general-purpose computers each year
  - In glory days, 1%/week performance improvement!
- Dumb to compete by designing parallel or specialized computers
  - By time you’ve finished design, the next generation of general-purpose will beat you!

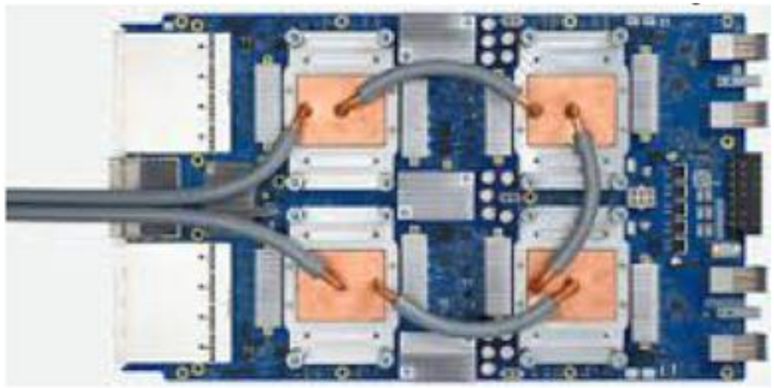
New Conventional Wisdom

- Each domain requires heterogeneous systems.
  - Multiple processor cores
  - GPUs,
  - NPUs,
  - accelerators,
  - interfaces,
  - memory, …

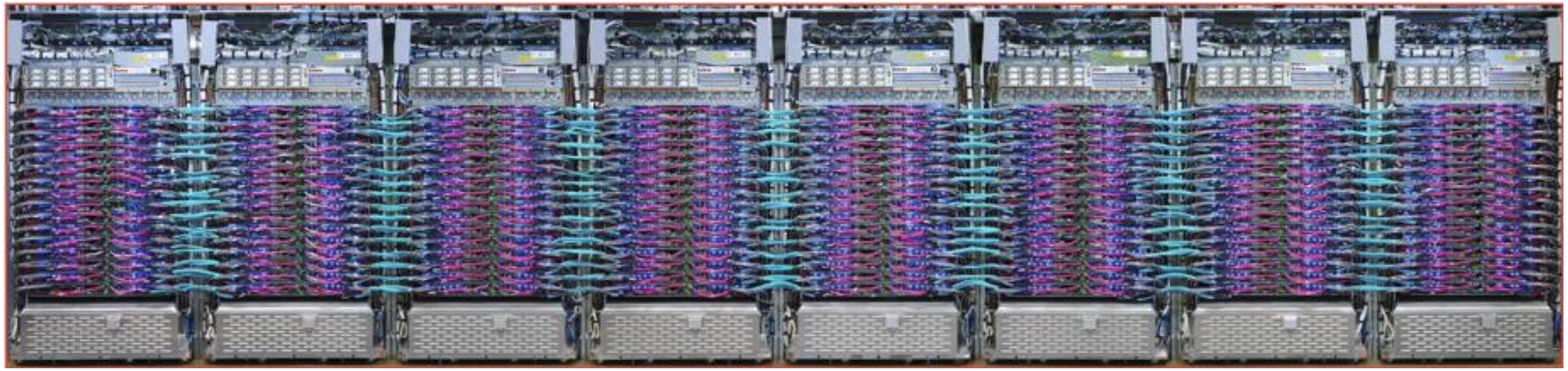


Apple M14

New Conventional Wisdom



Google TPU3
Specialized Engine for training Neural Networks
Deployed in cloud



1024 chips, > 100PetaFLOPs

KMGTP…

Patterson and Hennessy win Turing Award! (2017)

- Turing award 2017

  For pioneering a systematic, quantitative approach to the design and evaluation of computer architectures with enduring impact on the microprocessor industry.

David A. Patterson
Professor of UC Berkeley
Distinguished Engineer at Google

John L. Hennessy
President of Stanford University
Chairman of Alphabet

Innovations in computer architecture have a convention of defying conventional wisdom.

Summary: 6 Great Ideas in Computer Architecture

1. Abstraction (Layers of Representation/Interpretation)
2. Moore's Law
3. Principle of Locality/Memory Hierarchy
4. Parallelism
5. Performance Measurement & Improvement
6. Dependability via Redundancy