

# CS 110

## Computer Architecture (a.k.a. Machine Structures)

### Lecture 1: *Course Introduction*

Instructors:

**Sören Schwertfeger & Chundong Wang**

<https://robotics.shanghaitech.edu.cn/courses/ca/20s/>

**School of Information Science and Technology SIST**

**ShanghaiTech University**




**Slides based on UC Berkley's CS61C**




# Agenda




- Thinking about Machine Structures
- Great Ideas in Computer Architecture
- What you need to know about this class
- Everything is a Number




# Agenda

- Thinking about Machine Structures
- Great Ideas in Computer Architecture
- What you need to know about this class
- Everything is a Number

Rank	Language	Type	Score
1	Python	  	100.0
<p>An object-oriented, interpreted language that gains much of its power from a large constellation of libraries, including popular modules for machine learning and scientific computing.</p>			

Rank	Language	Type	Score
2	Java	  	96.3
<p>An object-oriented language that creates code intended to be run on a virtual machine, allowing it to run on different platforms with little or no modification. Java is a popular choice for Web applications.</p>			

Rank	Language	Type	Score
3	C	  	94.4
<p>C is used to write software where speed and flexibility is important, such as in embedded systems or high-performance computing.</p>			

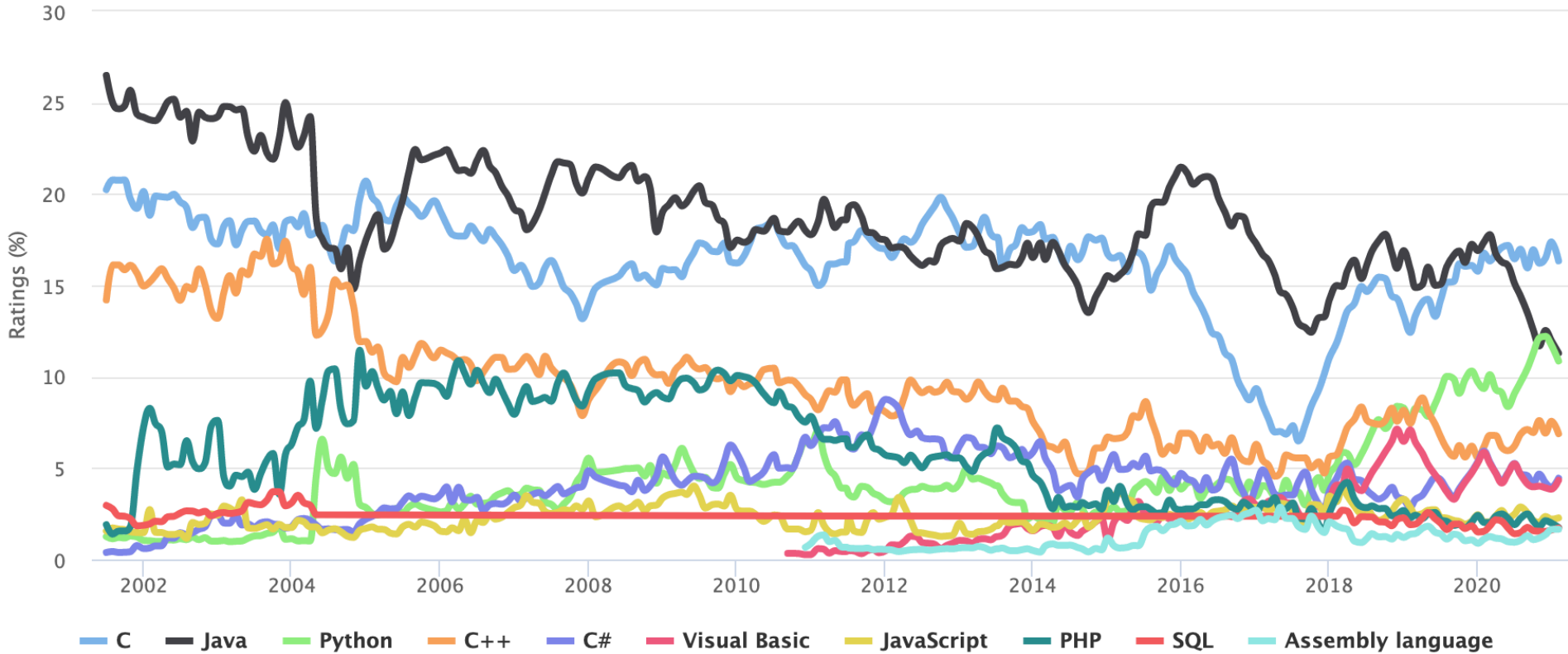
Rank	Language	Type	Score
4	C++	  	87.5
<p>Essentially an object-oriented version of C that proved to be a natural fit for software driven by graphical user interfaces.</p>			

<b>Web</b> 	<b>Enterprise</b> 
<b>Mobile</b> 	<b>Embedded</b> 

Rank	Language	Type	Score
1	Python	  	100.0
2	Java	  	96.3
3	C	  	94.4
4	C++	  	87.5
5	R		81.5
6	JavaScript		79.4
7	C#	   	74.5
8	Matlab		70.6
9	Swift	 	69.1
10	Go	 	68.0
11	Arduino		67.2
12	HTML,CSS		66.8
13	PHP		65.1
14	Assembly		63.7
<p>A catchall term for a vast family of processor instruction sets, writing assembly code requires considerable expertise, but it allows the creation of high-speed software that can run directly "on the metal."</p>			
15	SQL		63.4

## TIOBE Programming Community Index

Source: [www.tiobe.com](http://www.tiobe.com)

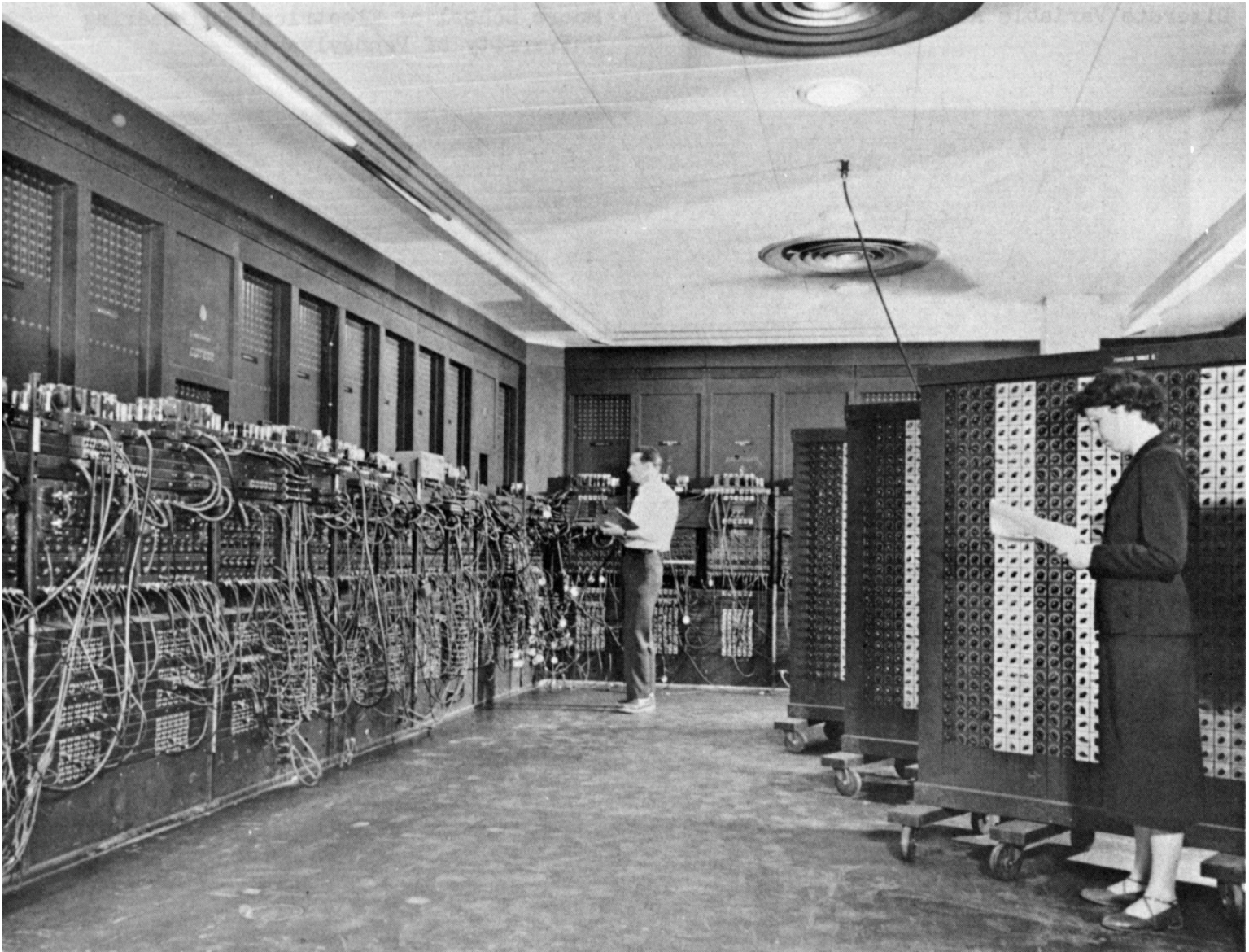


# Why You Need to Learn C!

# CS 110 is NOT really about C Programming

- It is about the *hardware-software interface*
  - What does the programmer need to know to achieve the highest possible performance
- C is close to the underlying hardware, unlike languages like Rust, 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*

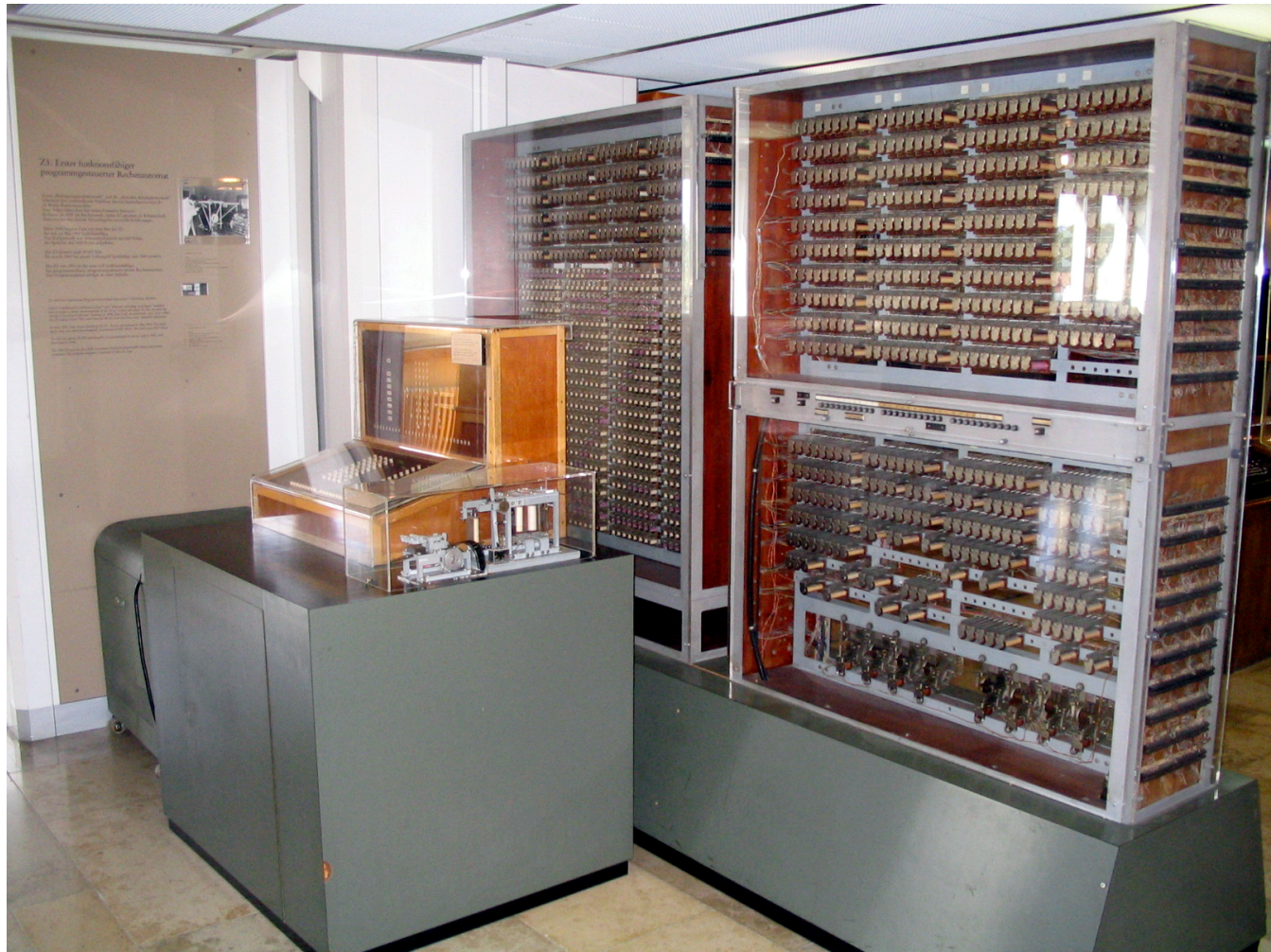
# Old School Computer Architecture





# Zuse Z3

first working programmable, fully automatic digital computer  
by Konrad Zuse in Berlin, 1941 (Inventor of Computer)



# New School Computer Architecture (1/3)

Personal  
Mobile  
Devices



Network  
Edge  
Devices

# New School Computer Architecture (2/3)



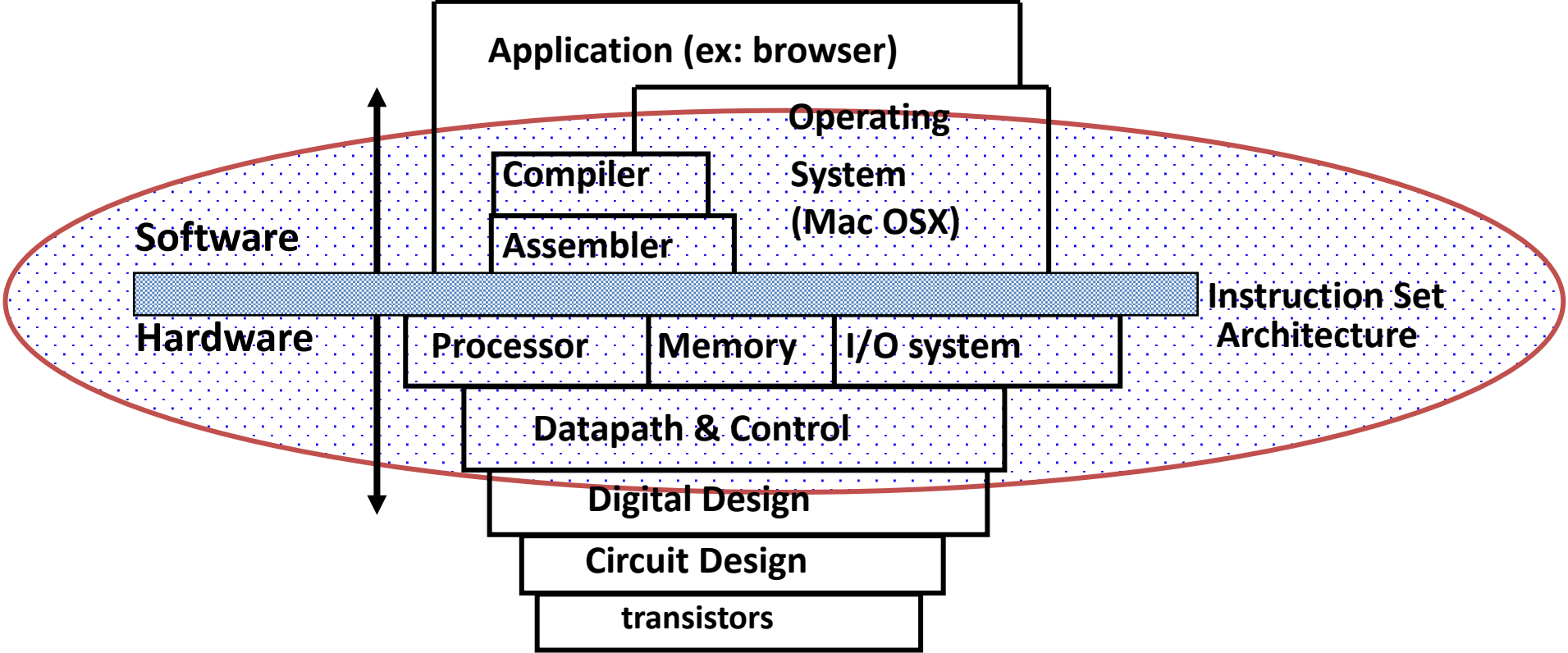
# New School Computer Architecture (3/3)



# SIST Computing Center



# Old School Machine Structures



# New-School Machine Structures (It's a bit more complicated!)

*Software*

*Hardware*

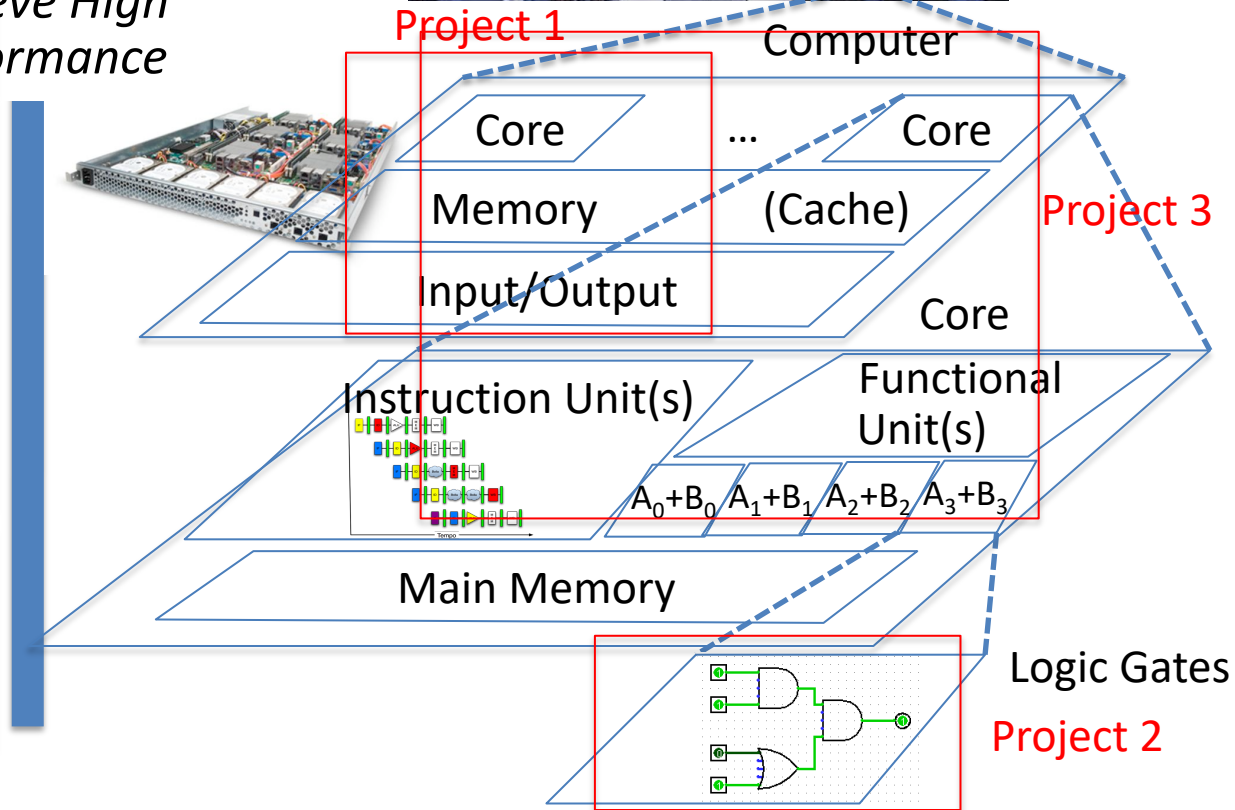
Warehouse  
-Scale  
Computer

Smart  
Phone



*Harness  
Parallelism &  
Achieve High  
Performance*

- 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 descriptions  
All gates functioning in  
parallel at same time



# Early 2018: Meltdown and Spectre

- Hardware vulnerability
- Affecting Intel x86 microprocessors, IBM POWER processors, and some ARM-based microprocessors
- All Operating Systems effected!
- They are considered "**catastrophic**" by security analysts!
- Allow to read all memory (e.g. from other process or other Virtual Machines (e.g. other users data on Amazon cloud service!))
- Towards the end of this CA course you can understand the basics of how Meltdown and Spectre work. Keywords:
  - Virtual Memory; Protection Levels; Instruction Pipelining; Speculative Execution; CPU Caching;





# Agenda

- Thinking about Machine Structures
- **Great Ideas in Computer Architecture**
- What you need to know about this class
- Everything is a Number

# 6 Great Ideas in Computer Architecture

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

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

Python / 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)

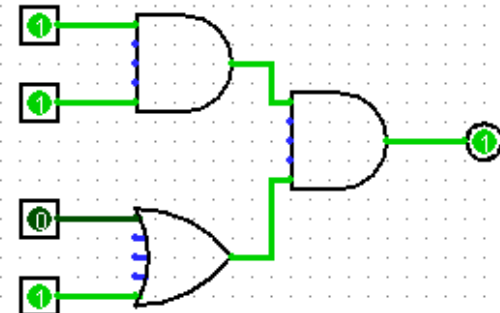
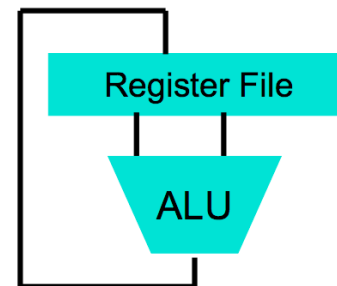
Physics

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

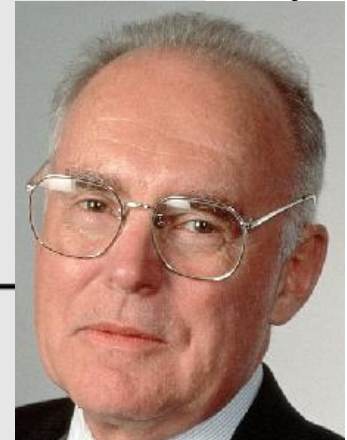
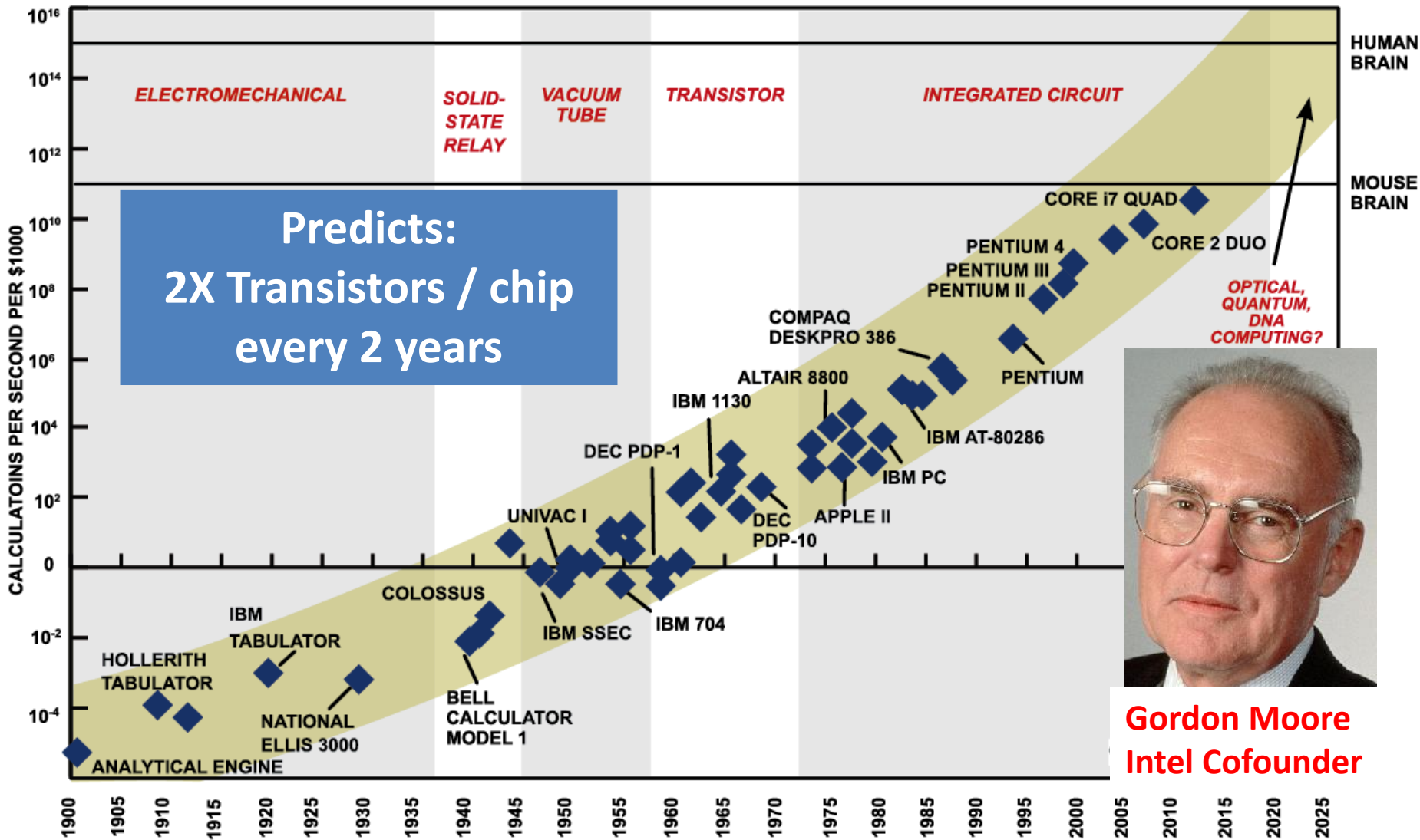
```
lw  t0, 0(s2)  
lw  t1, 4(s2)  
sw  t1, 0(s2)  
sw  t0, 4(s2)
```

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



**Gordon Moore**  
Intel Cofounder

SOURCE: RAY KURZWEIL, "THE SINGULARITY IS NEAR: WHEN HUMANS TRANSCEND BIOLOGY", P.67, THE VIKING PRESS, 2006. DATAPPOINTS BETWEEN 2000 AND 2012 REPRESENT BCA ESTIMATES.

# Interesting Times

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 <7nm, might have greater cost/transistor !!!!  
So, why shrink????



# Samsung's V1 fab opened Feb 2020

Now: 7 & 6 nm process

Future: up to 3 nm

Investment:

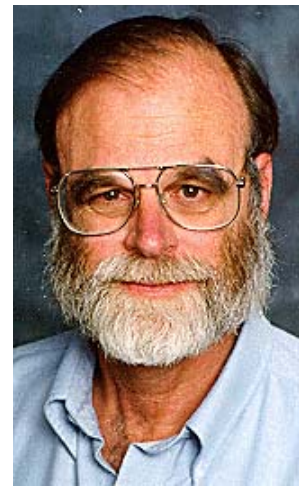
6,000,000,000 USD  
(4.2 million 万元)



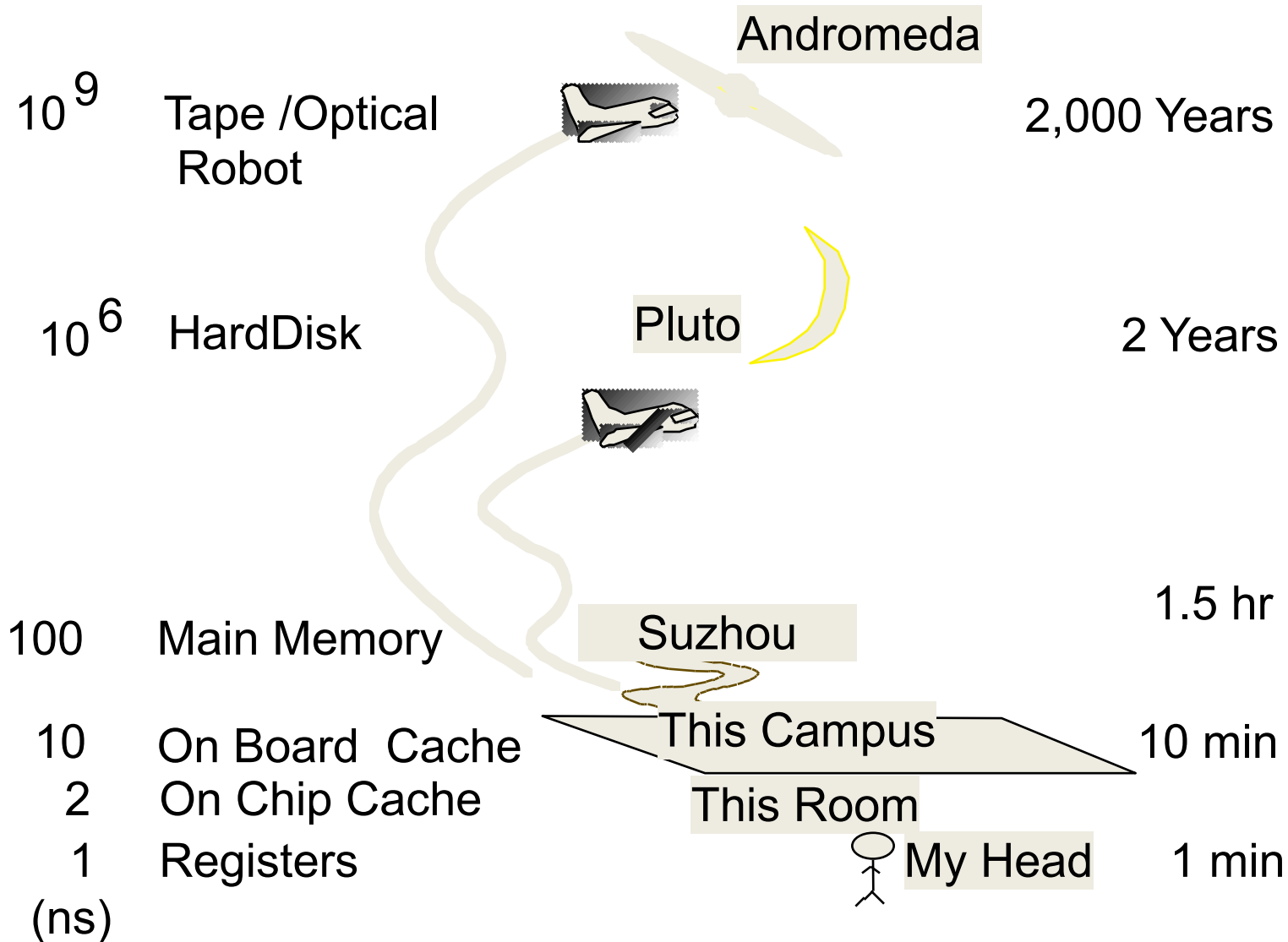
Other notable players: Intel, TSMC, GlobalFoundries  
(AMD), IBM

China: SMIC: 14nm production, developing 10 and 7nm

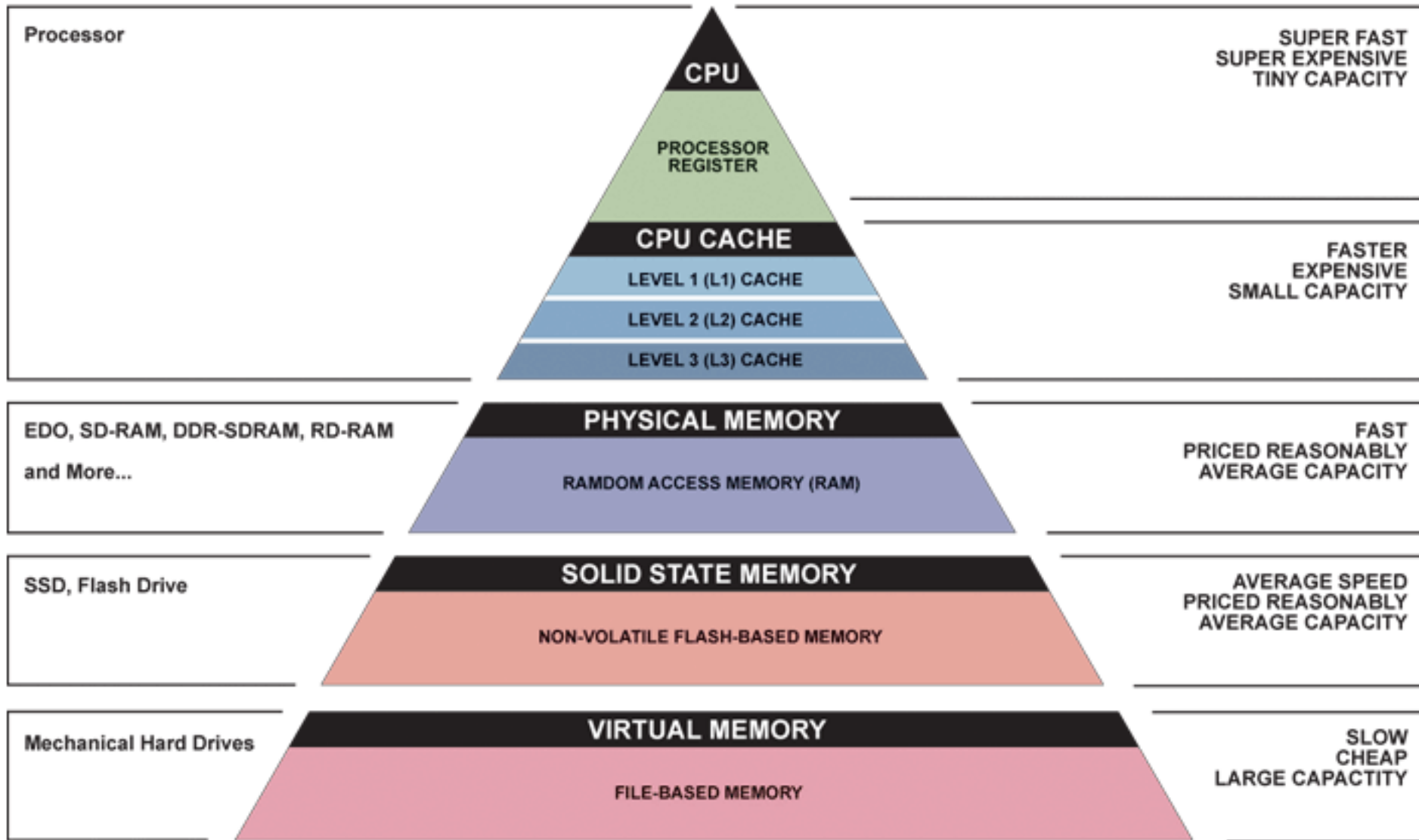
# Jim Gray's Storage Latency Analogy: How Far Away is the Data?



**Jim Gray**  
**Turing Award**

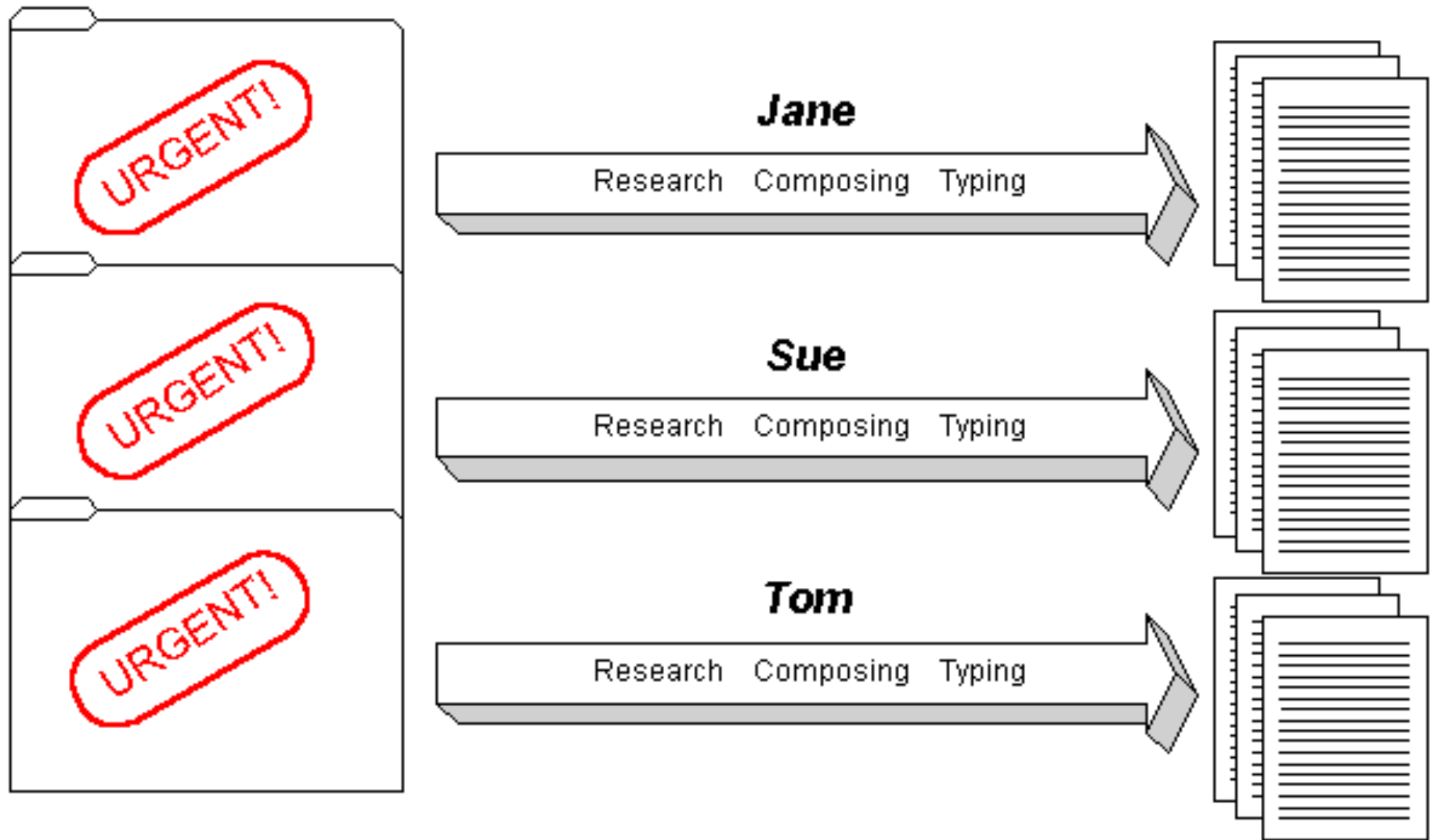


# Great Idea #3: Principle of Locality/ Memory Hierarchy

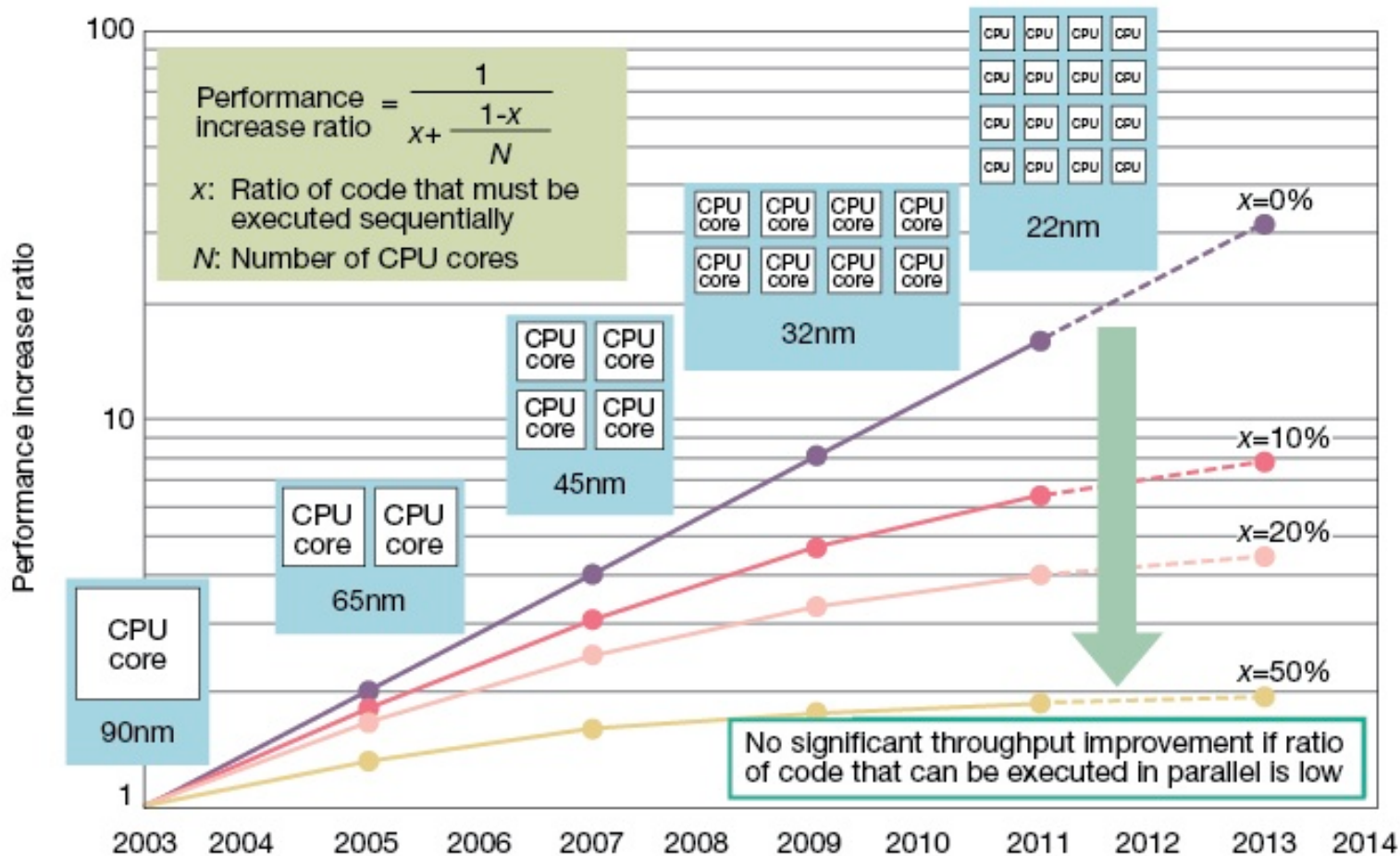




# Great Idea #4: Parallelism



# Caveat: Amdahl's Law



Gene Amdahl  
Computer Pioneer

**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 and Improvement

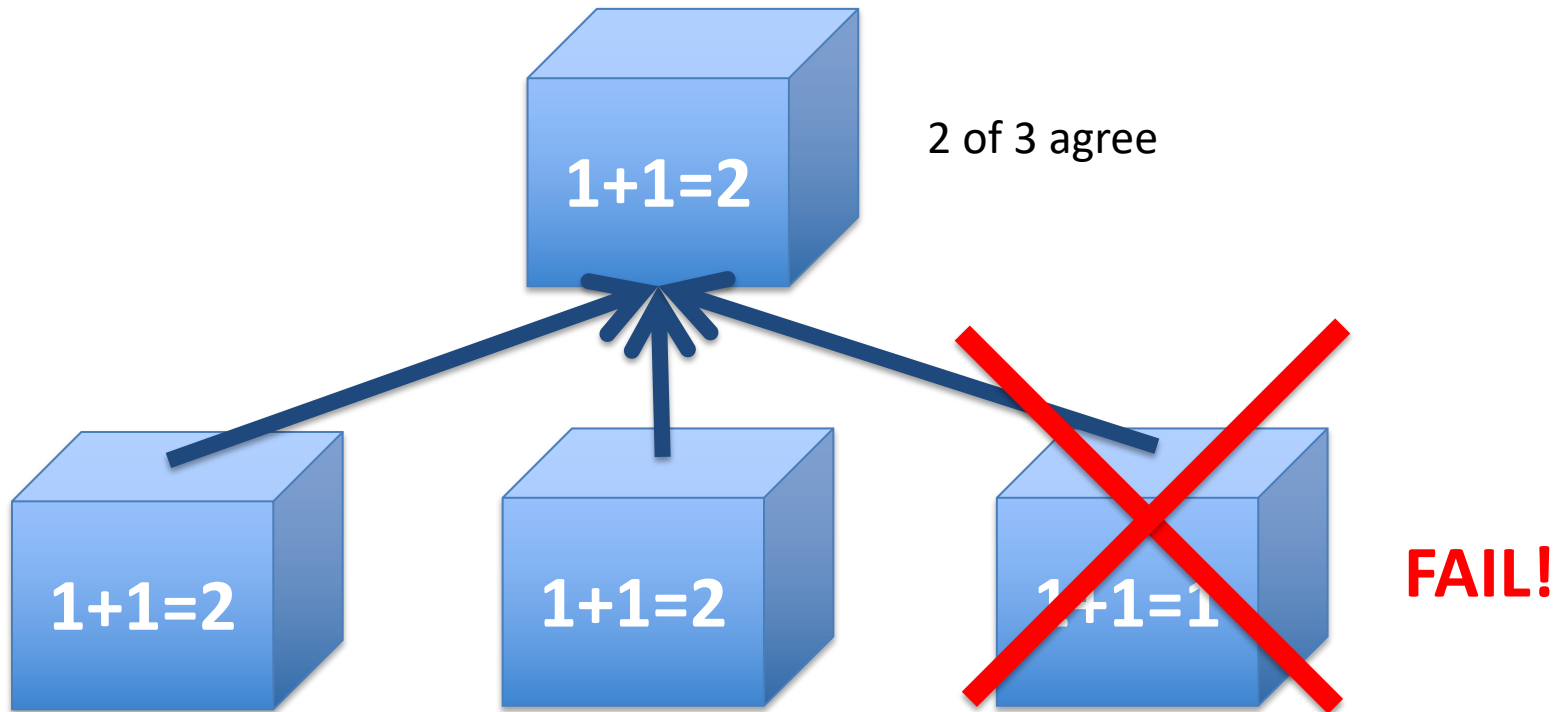
- Tuning application to underlying hardware to exploit:
  - Locality
  - Parallelism
  - Special hardware features, like specialized instructions (e.g., matrix manipulation)
- Latency
  - How long to set the problem up
  - How much faster does it execute once it gets going
  - It is all about *time to finish*

# Coping with Failures

- 4 disks/server, 50,000 servers
  - Failure rate of disks: 2% to 10% / year
    - Assume 4% annual failure rate
  - On average, how often does a disk fail?
    - a) 1 / month
    - b) 1 / week
    - c) 1 / day
    - d) 1 / hour
- $50,000 \times 4 = 200,000$  disks
- $200,000 \times 4\% = 8000$  disks fail
- $365 \text{ days} \times 24 \text{ hours} = 8760$  hours

# Great Idea #6: Dependability via Redundancy

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



Increasing transistor density reduces the cost of redundancy

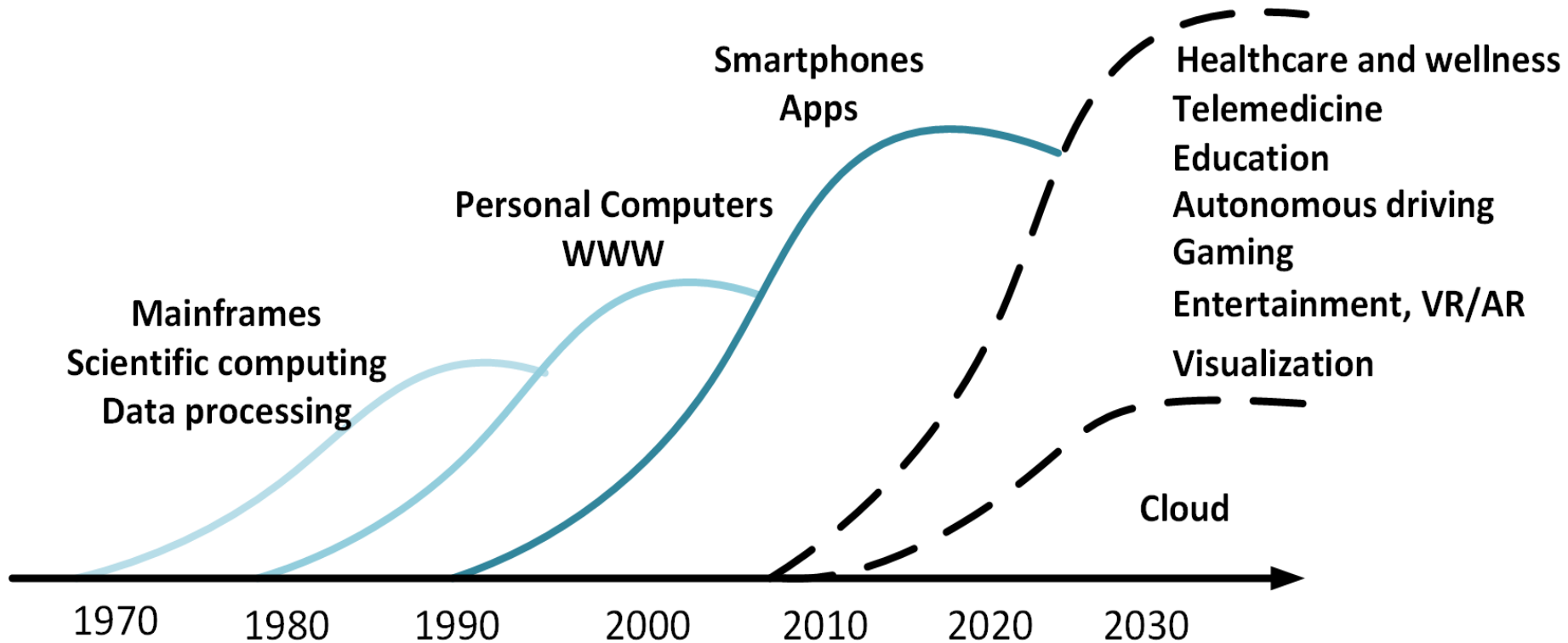
# Great Idea #6:

## 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 of so that can lose 1 bit but no data (Error Correcting Code/ECC Memory)



# Why is Architecture Exciting Today?

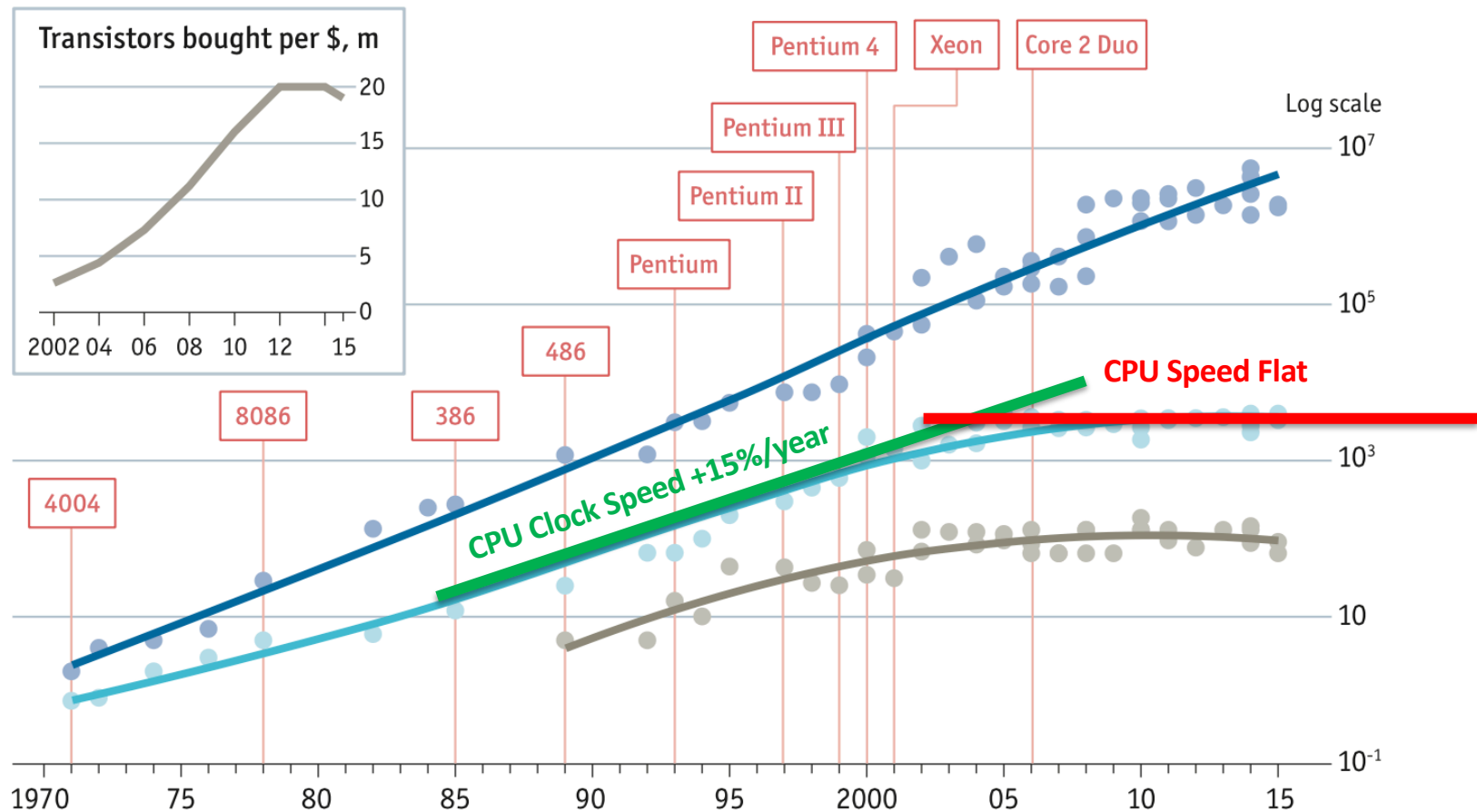


- Number of deployed devices continues growing, but no single killer app
  - Diversification of needs, architectures

# Why is Architecture Exciting Today?

## Stuttering

● Transistors per chip, '000 ● Clock speed (max), MHz ● Thermal design power\*, w □ Chip introduction dates, selected



Sources: Intel; press reports; Bob Colwell; Linley Group; IB Consulting; *The Economist*

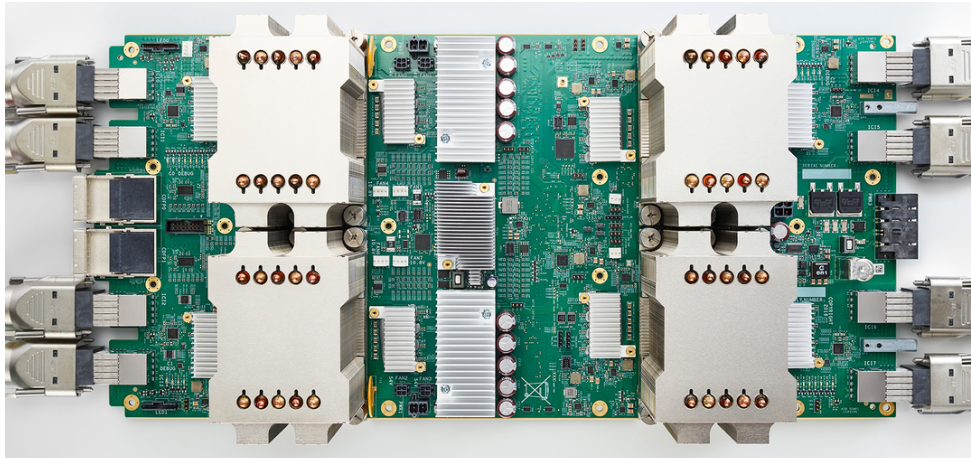
\*Maximum safe power consumption



# 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 parallel or specialized computers
- By time you've finished design, next generation of general-purpose will beat you

# New Conventional Wisdom



Google TPU2  
Specialized Engine for NN training  
Deployed in cloud  
45 TFLOPS/chip



→ Serious heatsinks!

# Agenda

- Thinking about Machine Structures
- Great Ideas in Computer Architecture
- **What you need to know about this class**
- Everything is a Number

# Computer Architecture

- CA is your most important course this semester!
  - **6 credit points**
    - 4 credit CA; 2 credit projects => projects in parallel to HW!
  - Your first CS only course
  - Spend a LOT of time on:
    - Textbook reading before class
    - HW and projects
    - Lab preparation
    - Learning for mid-terms and final
  - Understand how computers really work – complicated!
    - Too complicated? => Change major...

# Weekly Schedule

- Lecture** Tuesday, 10:15-11:55. 教学中心 (Teaching Center) 101
- Lecture** Thursday, 10:15-11:55. 教学中心 (Teaching Center) 101
- Lab 1** Monday, 15:55-17:35. SIST 1A-104; TA:
- Lab 2** Tuesday, 19:35-21:15. SIST 1A-104; TA:
- Lab 3** Tuesday, 19:35-21:15. SIST 1A-104; TA:
- Lab 4** Tuesday, 19:35-21:15. SIST 1A-104; TA:
- Lab 5** Tuesday, 19:35-21:15. SIST 1A-104; TA:
- Lab 6** Tuesday, 15:55-17:35. SIST 1A-106; TA:
- Lab 7** Tuesday, 15:55-17:35. SIST 1A-106; TA:
- Lab 8** Tuesday, 15:55-17:35. SIST 1A-106; TA:
- Lab 9** Monday, 15:55-17:35. SIST 1A-104; TA:



Sören Schwertfeger 师泽仁  
<soerensch>



Chungdong Wang  
<cd\_wang AT outlook.com>



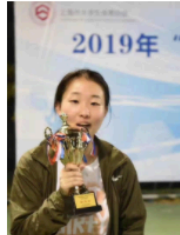
Kaiyuan Xu  
<xuky>  
Head TA



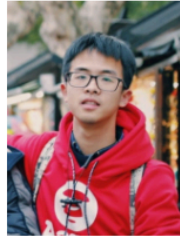
Tiansu Chen  
<chents>



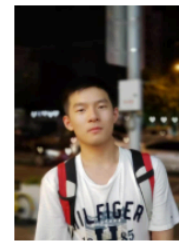
Guancheng Li  
<ligch>



Chenyu Wang  
<wangchy4>



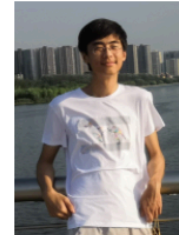
Zhe Ye  
<yezhe>



QiYuan Dai  
<daigy>



Peifan Li  
<lipf>



Jinrui Wang  
<wangjr>



Ziyi Yu  
<yuzy>



Yuting Zheng  
<zhengyt1>



Zihao Diao  
<diaozh>



Shaoting Peng  
<pengst>



Teng Xu  
<xuteng>



Chibin Zhang  
<zhangcb1>

# Course Information

- Course Web: <https://robotics.shanghaitech.edu.cn/courses/ca/21s>
- Acknowledgement: Instructors of UC Berkeley's CS61C: <https://cs61c.org/>
- Instructors:
  - Sören Schwertfeger & Chundong Wang
- Teaching Assistants: (see webpage)
- Textbooks: Average 15 pages of reading/week
  - Patterson & Hennessey, *Computer Organization and Design RISC-V* edition!
  - Kernighan & Ritchie, *The C Programming Language*, 2<sup>nd</sup> Edition
  - Barroso & Holzle, *The Datacenter as a Computer*, 2<sup>nd</sup> Edition
- Piazza:
  - Every announcement, discussion, clarification happens there



# Videos

- Lecture Videos from last year are available (with English subtitles):

<https://robotics.shanghaitech.edu.cn/courses/ca/20s/>

- Course content this year is very similar – still there might be differences!
- Participation in class is still highly recommended!



# Course Grading

- Projects: 33%
- Homework: 19%
- Lab: 5%
- Exams: 40%
  - Midterm 1: 10%
  - Midterm 2: 10%
  - Final: 20%
- Participation: 3%
  - (in class, in piazza, non credit parts of HW/ projects, help other during labs)



- 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!

# Labs

- Labs: Find one partner for your lab-work from you lab class!
  - Labs start next week
  - Projects are done in 2-person teams!
- Need Linux for the labs!
  - Recommendation: install natively (dual boot)!
  - Virtual Machine works fine
  - If in doubt: Ubuntu 20.04
  - Many labs/ homework/ project may work with Mac, but no guarantee, no support. Use Linux!

# Discussion

- Time:
  - Monday 20:30-21:30 Seems Best
  - Wednesday 20:30-21:30 Maybe
  - Thursday 20:30-21:30 No
- Content: The TA will give a presentation about current topics (in Chinese). Then you can ask questions.
- Participation is not mandatory

# Office Hours

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



- CA will use Autolab for grading HW & projects
  - Setup maintained by TAs
    - => Please be patient and report any bugs/ problems in piazza or (if sensitive) via email.
  - Your logins will be created soon!
  - <http://autolab.sist.shanghaitech.edu.cn>
- HW 1 is available on Autolab. Due March 2!
- Gradescope for text-based HW and exams.  
<https://www.gradescope.com>





- Gitlab for projects!
- Use for collaboration.
- Autolab will directly pull from your gitlab!
- <http://autolab.sist.shanghaitech.edu.cn/gitlab>
- Accounts will be created later

# Late Policy ... Slip Days!

- Assignments due at 11:59:59 PM
- You have 3 slip day tokens (NOT hour or min)
- Every day your project or homework is late (even by a minute) we deduct a token
- After you've used up all tokens, it's 25% deducted per day.
  - No credit if more than 3 days late
  - Save your tokens for projects, worth more!!
- No need for sob stories, just use a slip day!
- Autolab will take care of this!



# Policy on Assignments and Independent Work

- **ALL PROJECTS WILL BE DONE WITH A PARTNER**
- With the exception of laboratories and assignments that explicitly permit you to work in groups, all homework and projects 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 is NOT allowed!**
- Use the Office Hours of the TA and the Prof. 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 from other students.
- 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.
- **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!**
- 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**