CS 110
Computer Architecture
(a.k.a. Machine Structures)
Lecture 1: *Course Introduction*

Instructor:
Sören Schwertfeger

http://shtech.org/courses/ca/

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

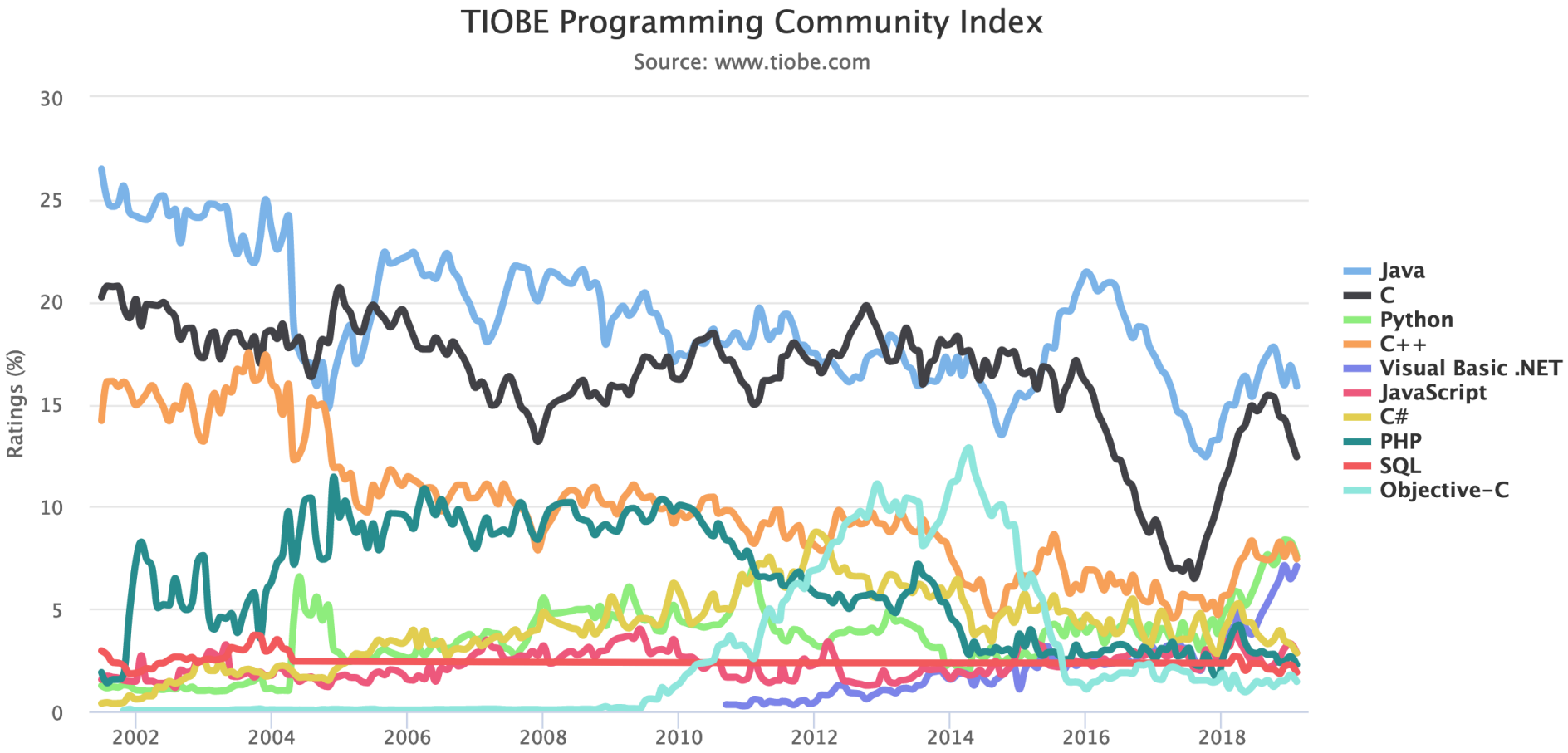
Agenda

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

Language Types (click to hide)

Web | Mobile | Enterprise | Embedded

Language Rank	Types	Spectrum Ranking
1. Python	Web, Enterprise, Embedded	100.0
A scripting language that is often used by software developers to add programmability to their applications, such as engineering-analysis tools or animation software.		
2. C++	Mobile, Enterprise, Embedded	99.7
Essentially a version of C with built-in support for "objects"--self-contained modules of code and data--C++ proved to be a natural fit for software driven by graphical user interfaces.		
3. Java	Web, Mobile, Enterprise	97.5
Designed to allow the creation of programs that can run on different platforms with little or no modification, Java is a popular choice for Web applications.		
4. C	Mobile, Enterprise, Embedded	96.7
C is used to write software where speed and flexibility is important, such as in embedded systems or high-performance computing.		
5. C#	Web, Mobile, Enterprise	89.4
6. PHP	Web	84.9
7. R	Enterprise	82.9
8. JavaScript	Web, Mobile	82.6
9. Go	Web, Enterprise	76.4
10. Assembly	Embedded	74.1
Writing assembly code requires considerable expertise, but it allows the creation of high-speed software that can run directly on a computer processor.		

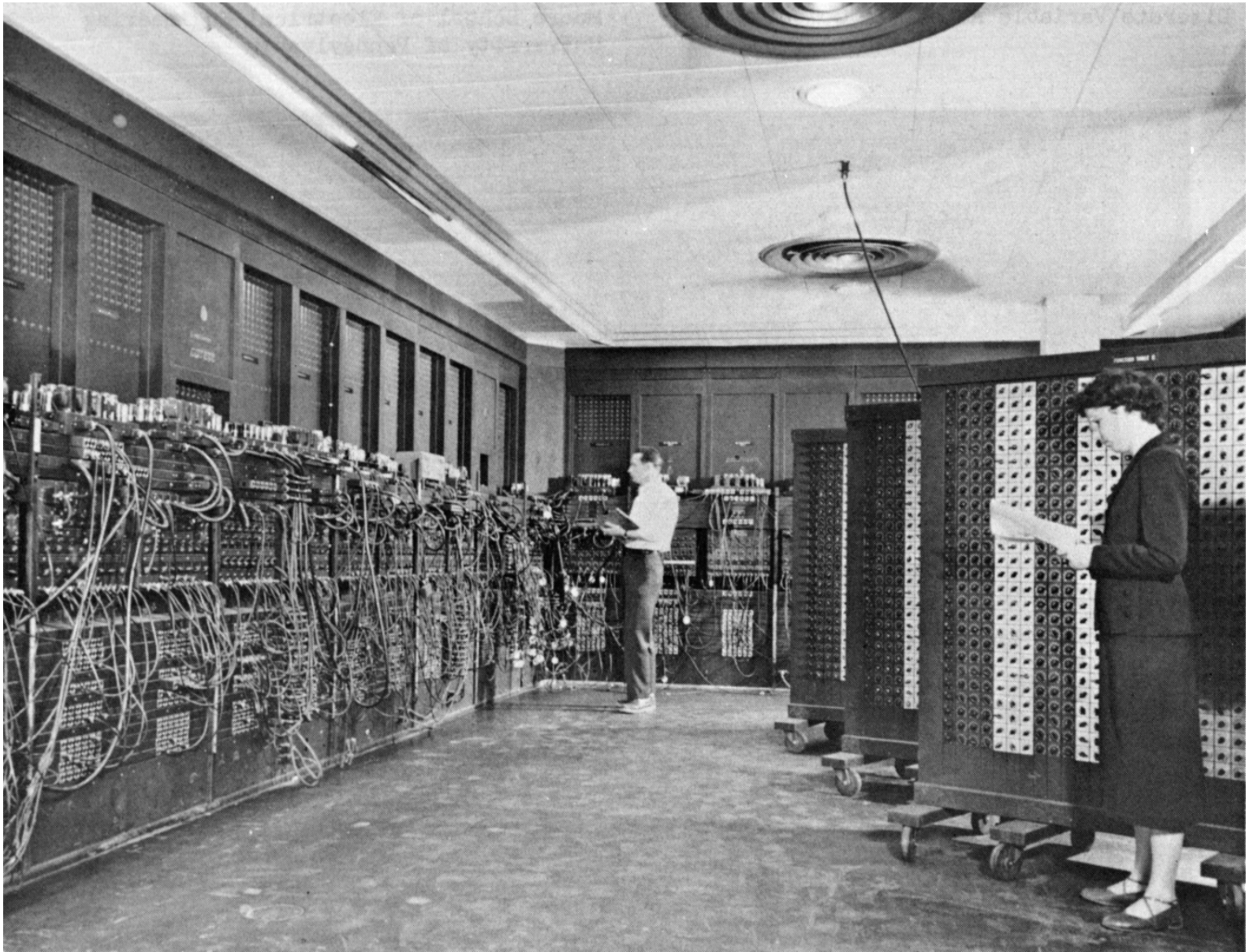

TIOBE Programming Community Index
Source: www.tiobe.com
Ratings (%)
30
25
20
15
10
5
0
2002
2004
2006
2008
2010
2012
2014
2016
2018
Java
C
Python
C++
Visual Basic .NET
JavaScript
C#
PHP
SQL
Objective-C

Why You Need to Learn C!

CS 110 is NOT really about C Programming

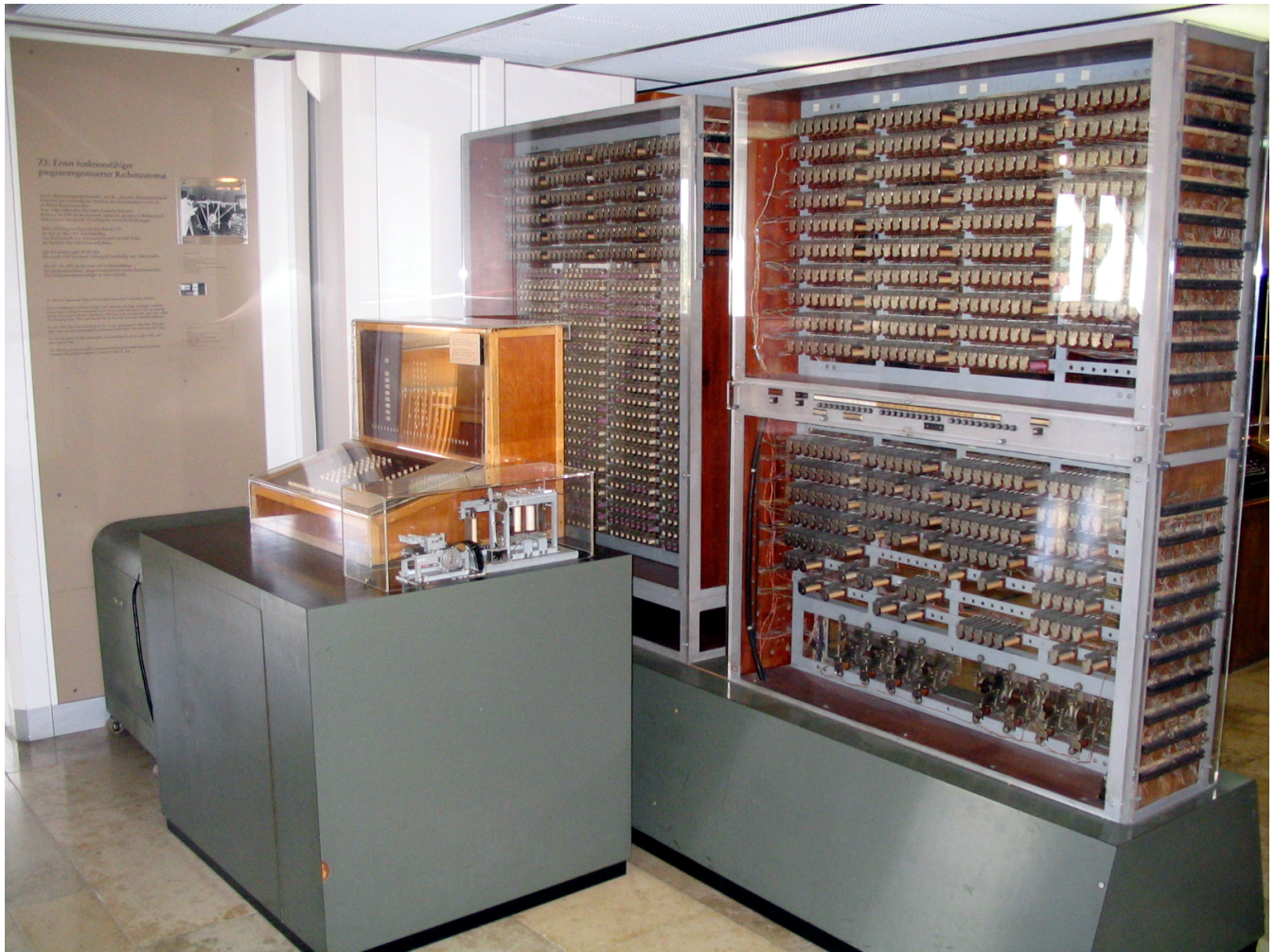
- It is about the *hardware-software interface*
  - What does the programmer need to know to achieve the highest possible performance
- C is close to the underlying hardware, unlike languages like Rust, Python, Java!
  - Allows us to talk about key hardware features in higher level terms
  - Allows programmer 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)

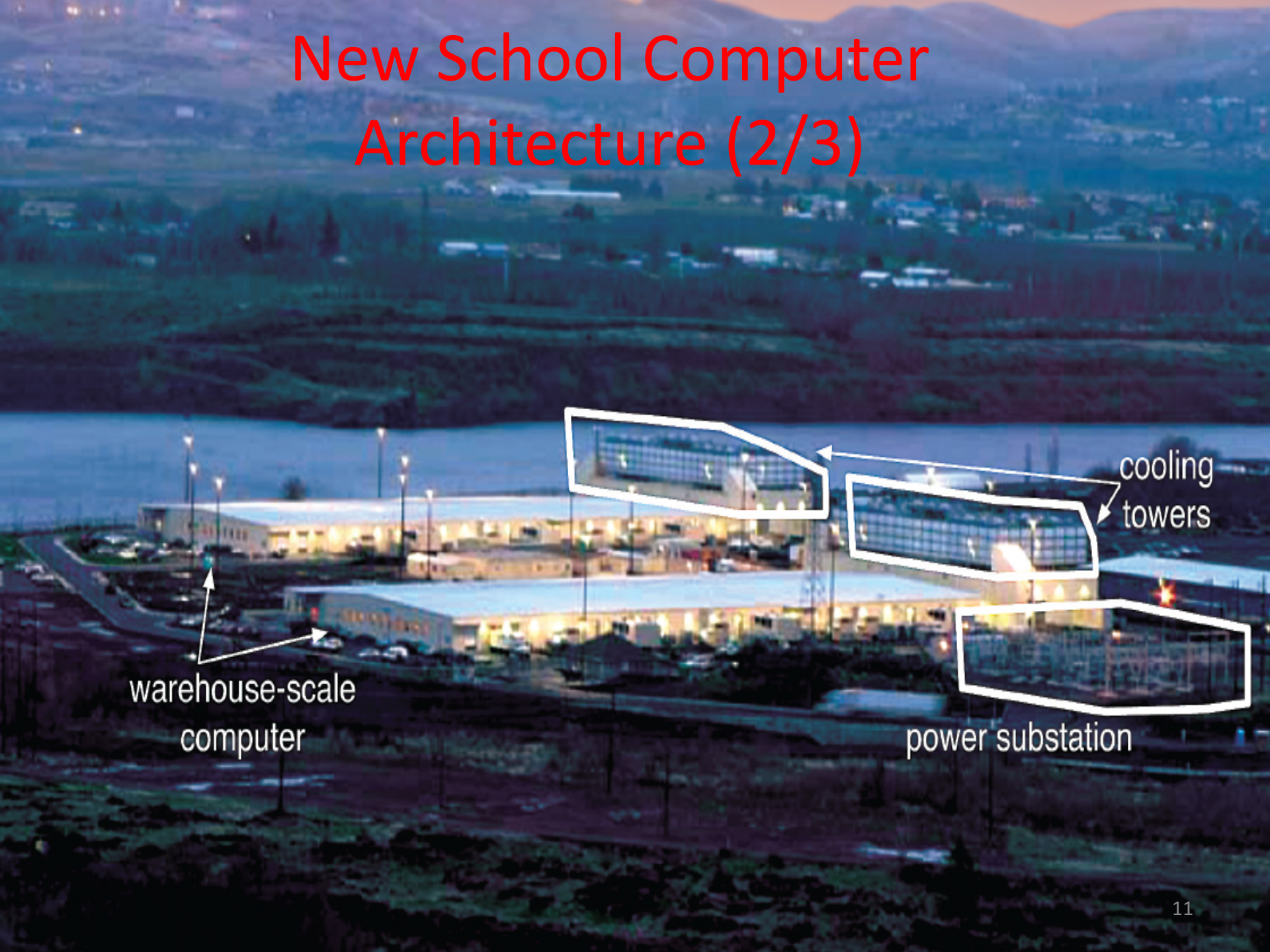


New School Computer Architecture (1/3)

Personal Mobile Devices

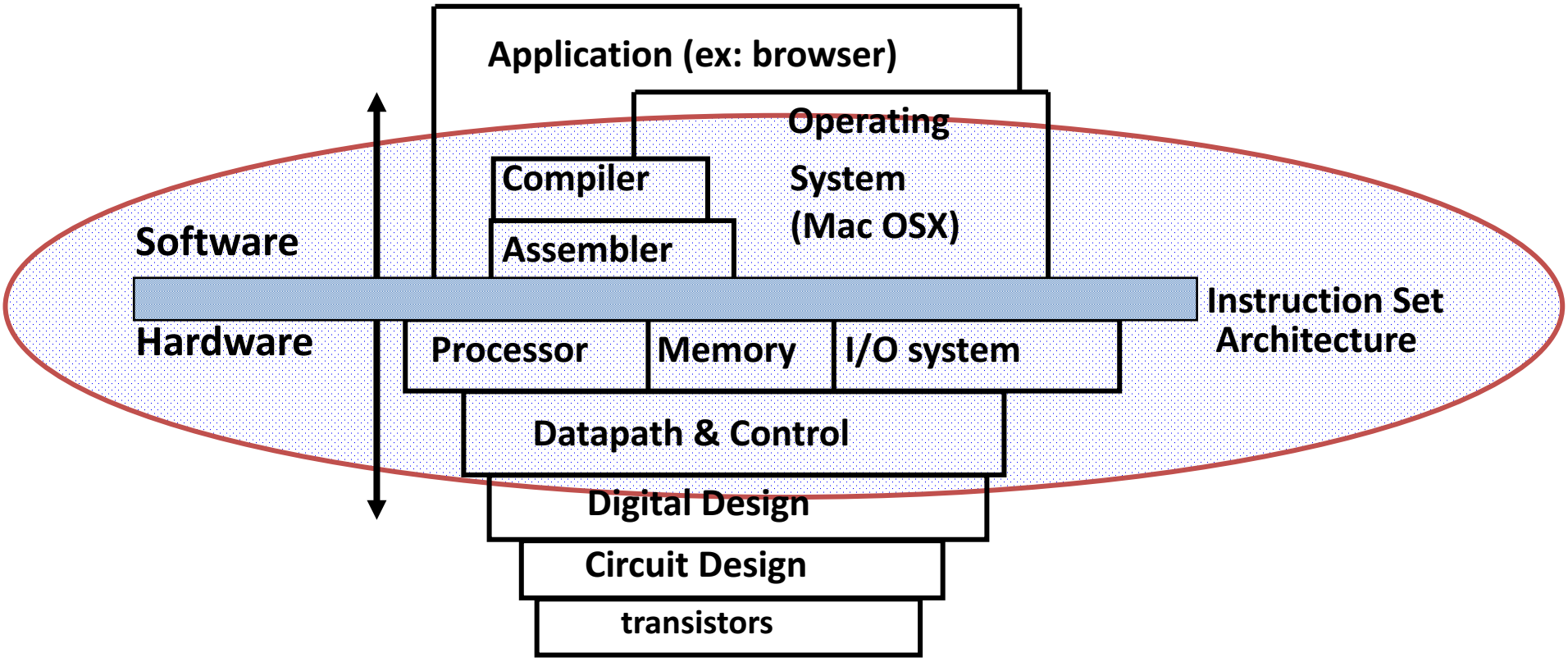
Network Edge Devices

New School Computer Architecture (2/3)



New School Computer Architecture (3/3)

Old School Machine Structures



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

Software | *Hardware*

- 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

Harness Parallelism & Achieve High Performance

Warehouse -Scale Computer

Smart Phone

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;

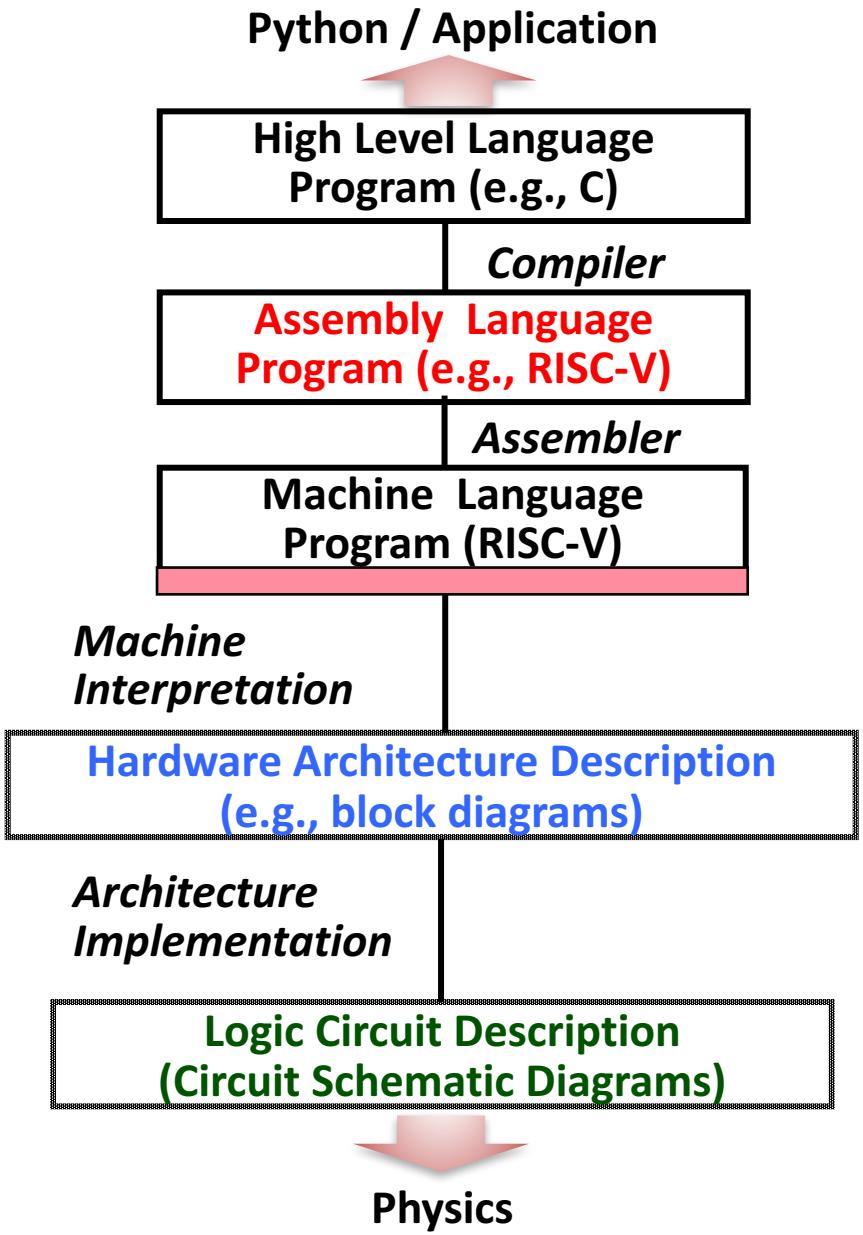
Agenda

- 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)

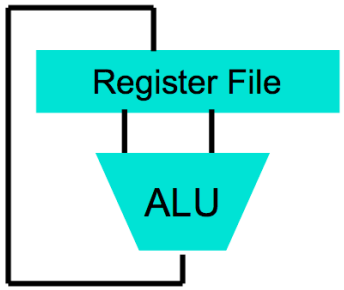


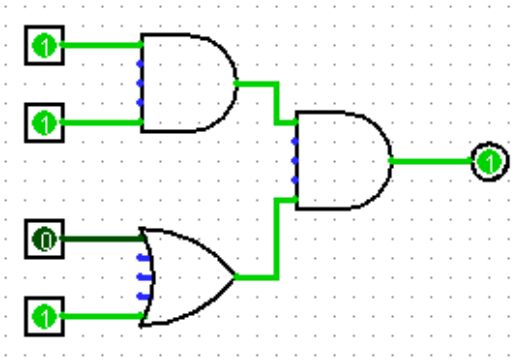

```
temp = v[k];
v[k] = v[k+1];
v[k+1] = temp;
```

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

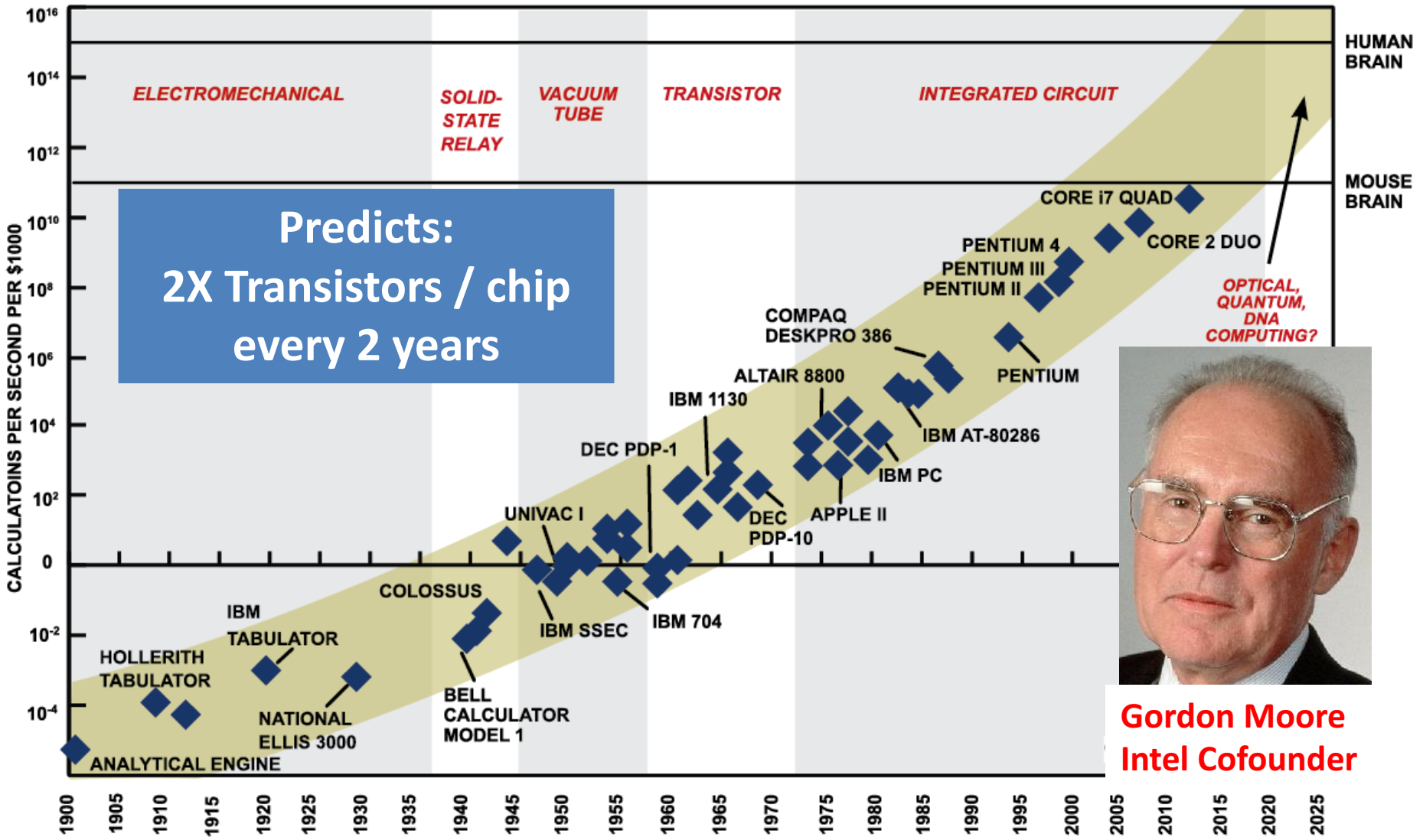
Anything can be represented as a *number*, i.e., data or instructions

```
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

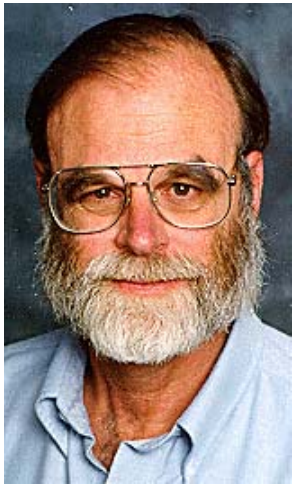
SOURCE: RAY KURZWEIL, "THE SINGULARITY IS NEAR: WHEN HUMANS TRANSCEND BIOLOGY", P.67, *THE VIKING PRESS*, 2006. DATAPOINTS BETWEEN 2000 AND 2012 REPRESENT BCA ESTIMATES.

Interesting Times

Moore's Law relied on the cost of transistors scaling down as technology scaled to smaller and smaller feature sizes.

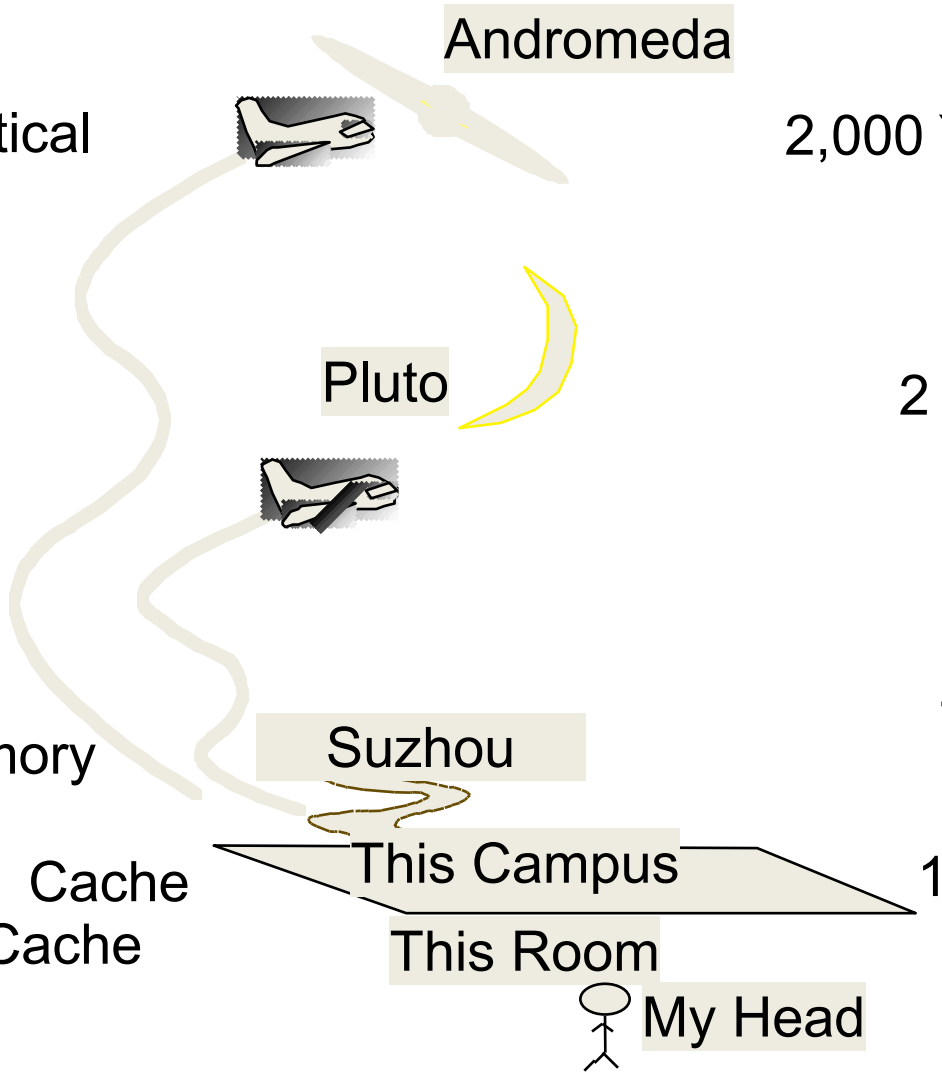
BUT newest, smallest fabrication processes <10nm, might have greater cost/transistor !!!!
So, why shrink????

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

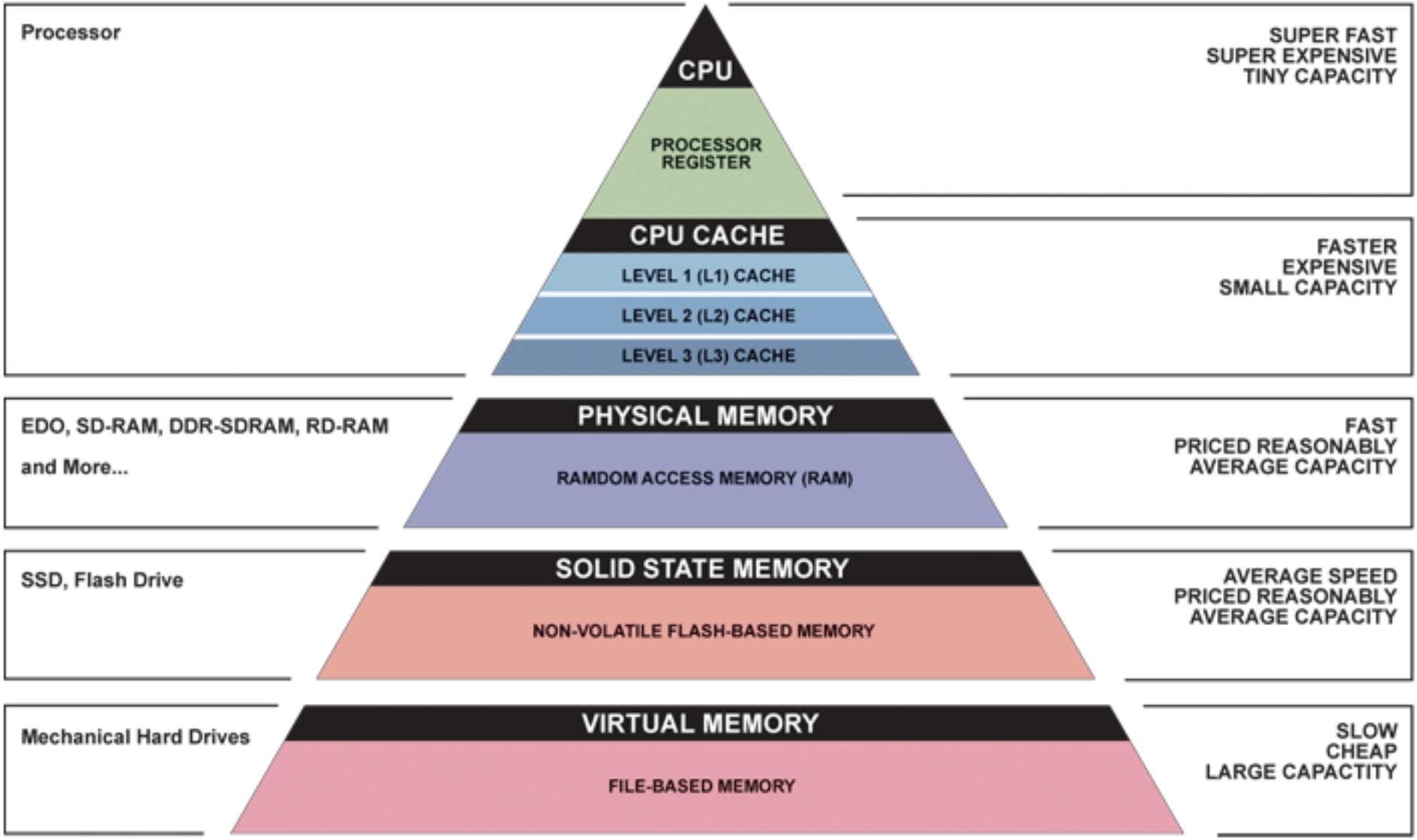


Jim Gray
Turing Award

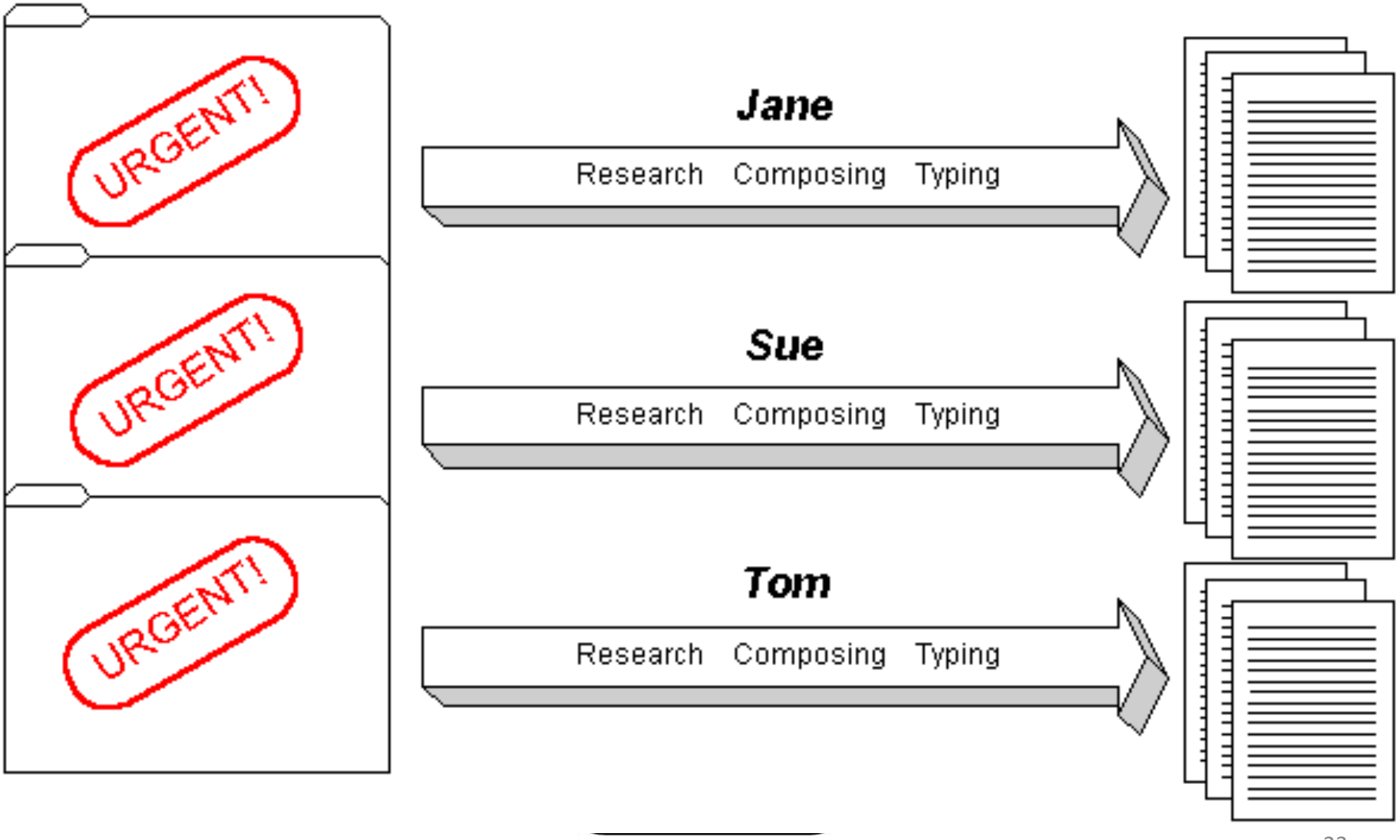
10^9	Tape /Optical Robot	2,000 Years
10^6	HardDisk	2 Years
100	Main Memory	1.5 hr
10	On Board Cache	10 min
2	On Chip Cache	
1	Registers	1 min
(ns)		



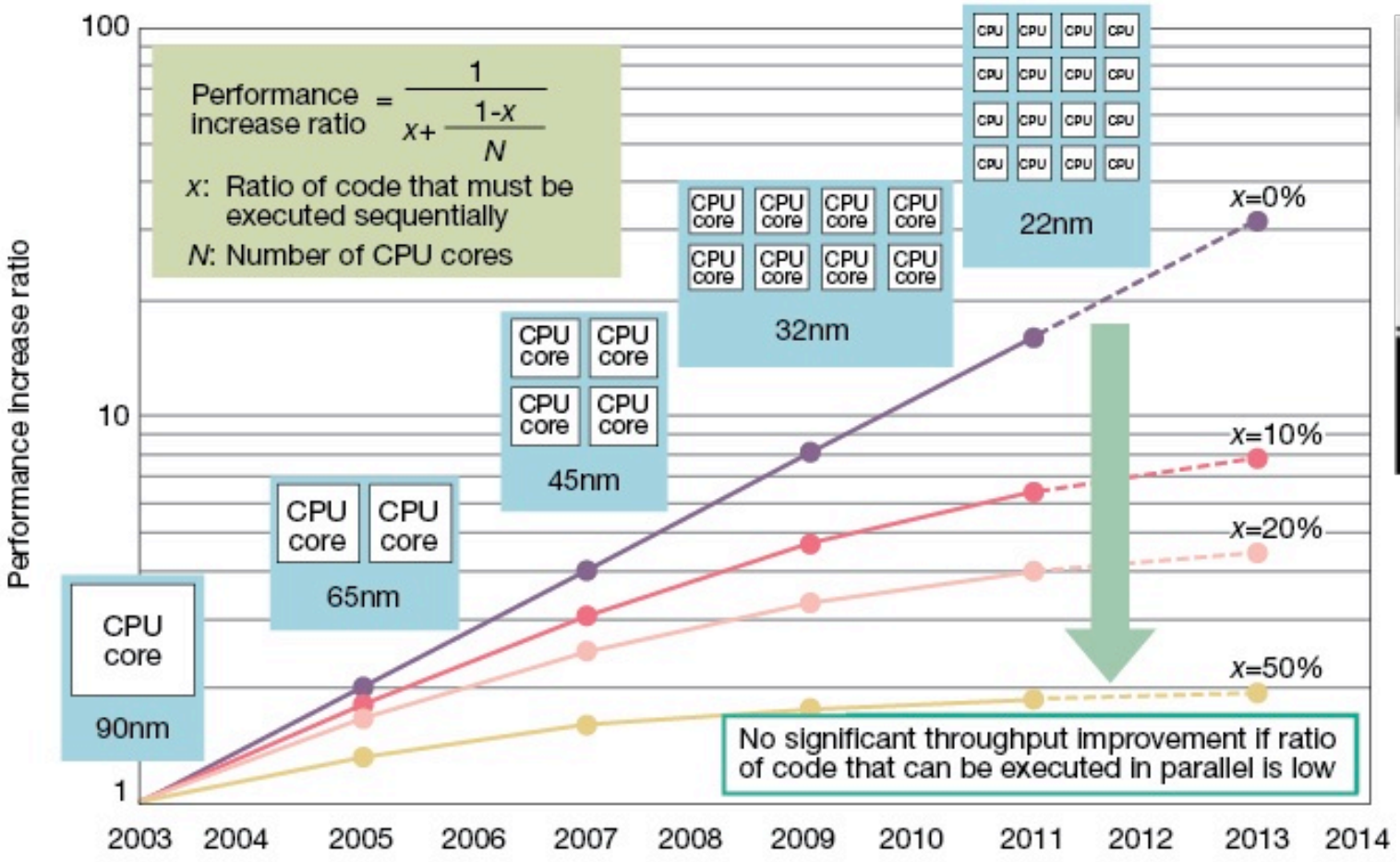
Great Idea #3: Principle of Locality/ Memory Hierarchy



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

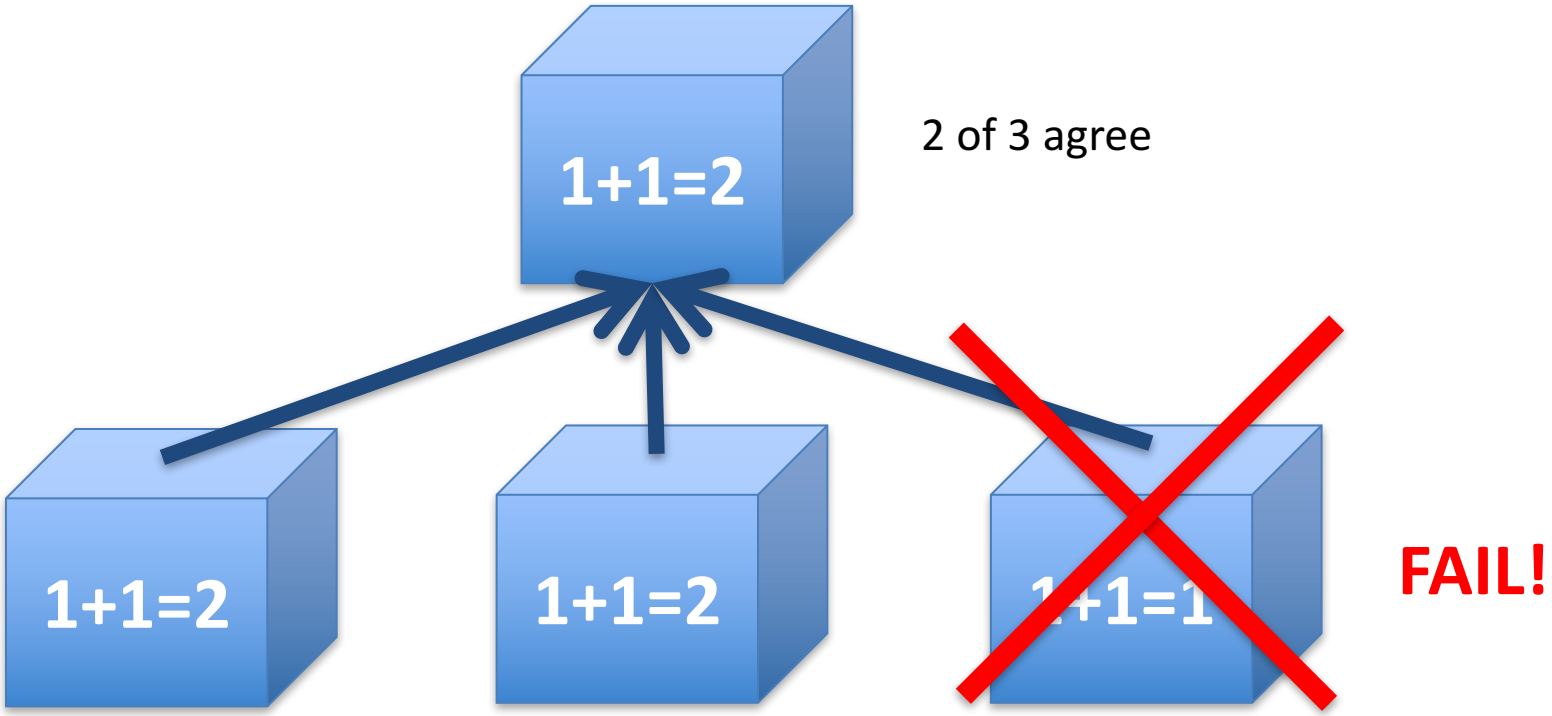
50,000 x 4 = 200,000 disks

200,000 x 4% = 8000 disks fail

365 days x 24 hours = 8760 hours

Great Idea #6: Dependability via Redundancy

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

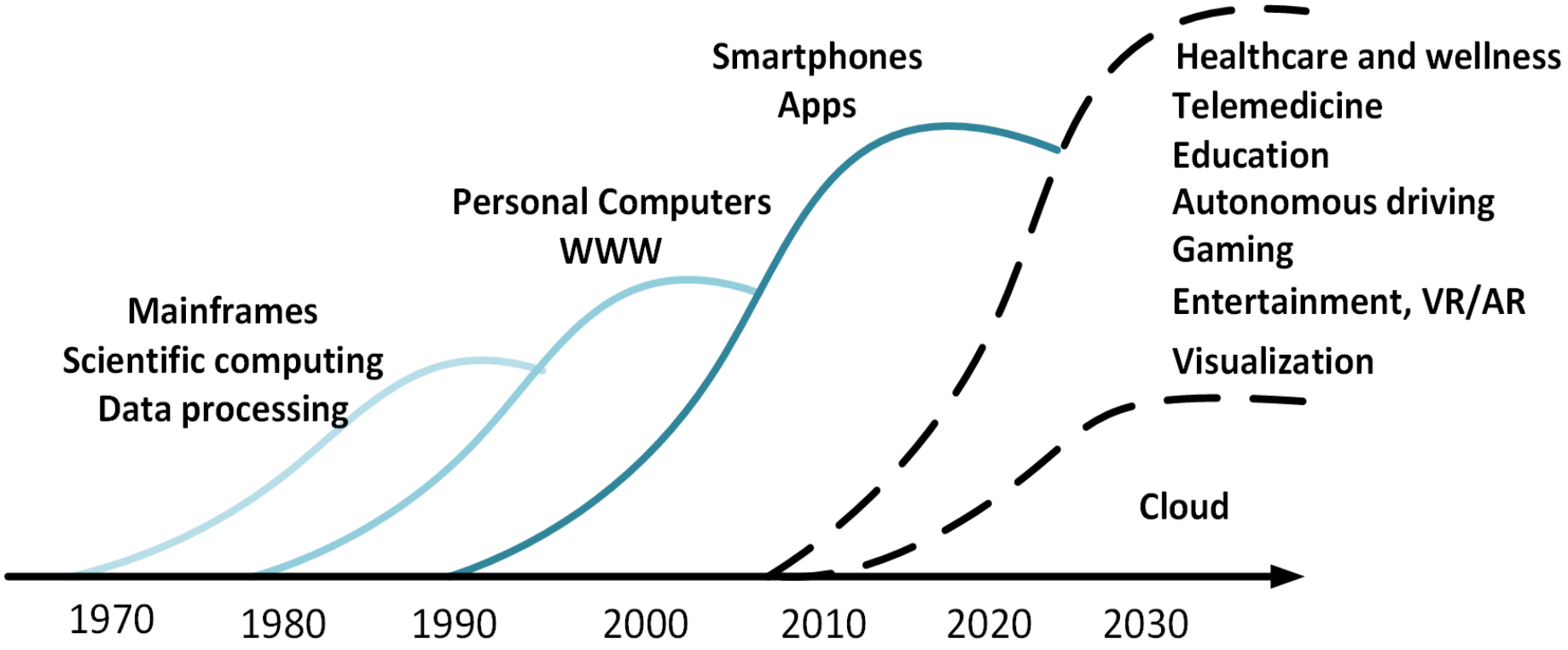


Increasing transistor density reduces the cost of redundancy

Great Idea #6: Dependability via Redundancy

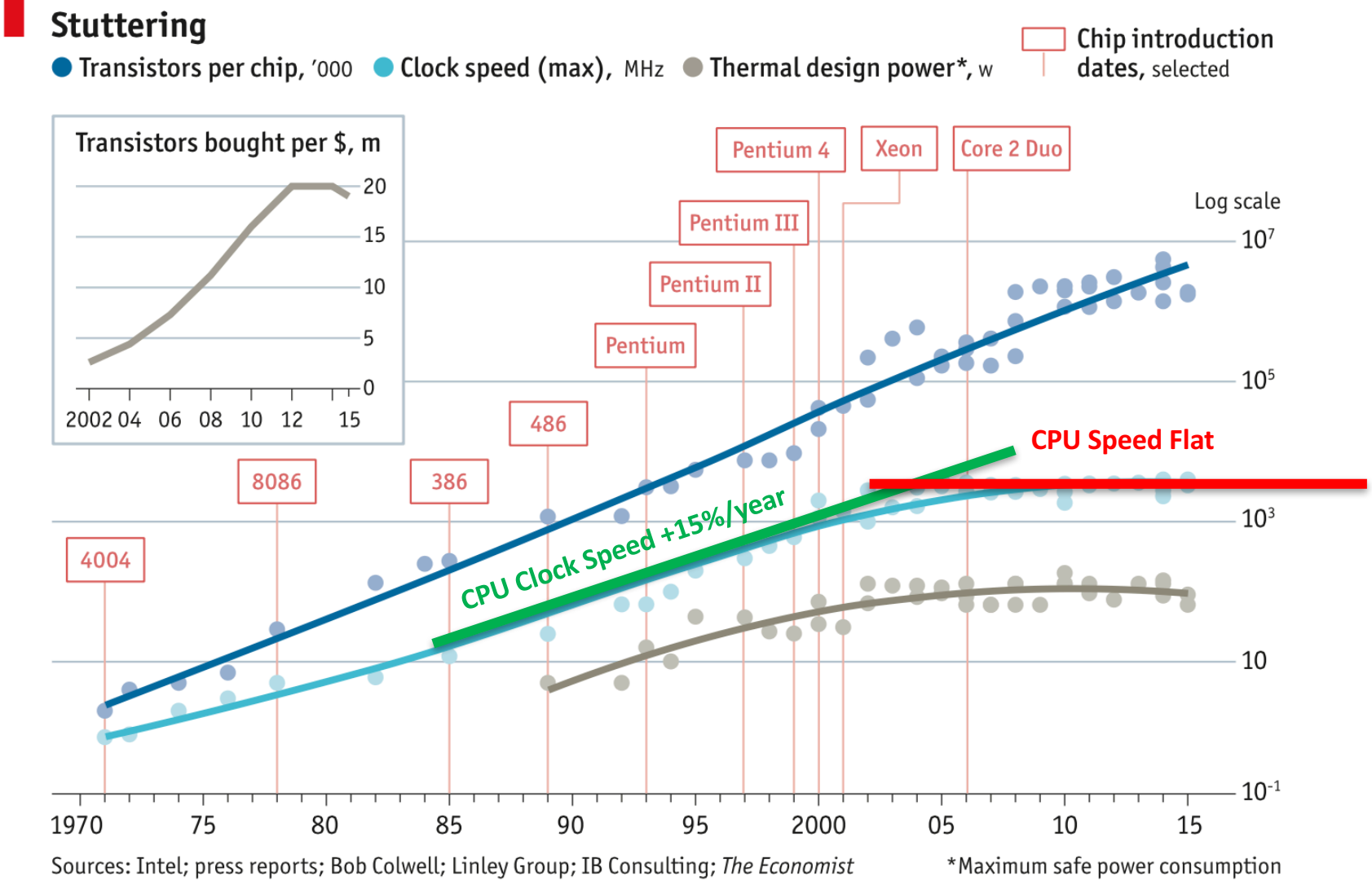
- Applies to everything from datacenters to storage to memory to instructors
  - Redundant datacenters so that can lose 1 datacenter but Internet service stays online
  - Redundant disks so that can lose 1 disk but not lose data (Redundant Arrays of Independent Disks/RAID)
  - Redundant memory bits of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

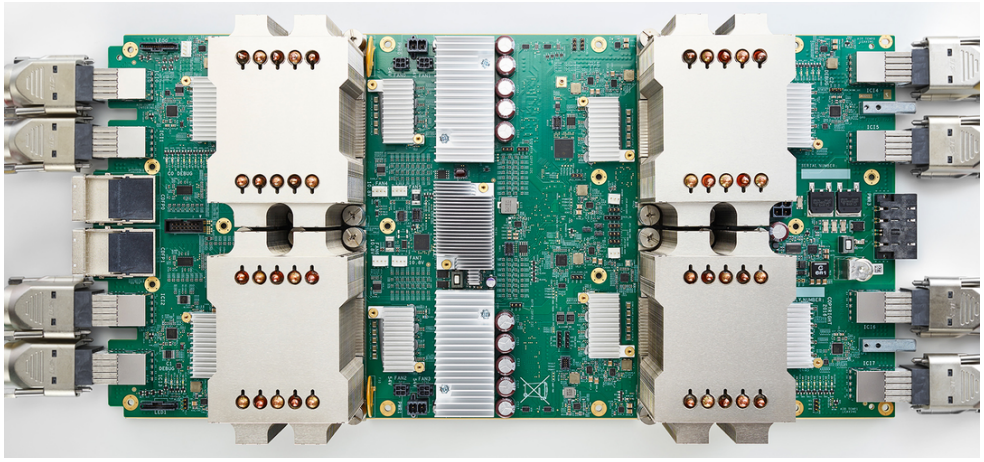


Old Conventional Wisdom

- 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

- Show of hands: CS & EE students
- 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	Wednesday, 08:15-09:55. 教学中心 (Teaching Center) 201
Lecture	Friday, 08:15-09:55. 教学中心 (Teaching Center) 201
Discussion	**???**
Lab 1	Monday, 15:55-17:35. SIST 1A-104; TA: Yanjie Song
Lab 2	Monday, 15:55-17:35. SIST 1A-106; TA: Zhijie Yang
Lab 3	Monday, 15:55-17:35. SIST 1A-108; TA: Zilin Si
Lab 4	Tuesday, 15:55-17:35. SIST 1A-104; TA: Wang Ruoyu
Lab 5	Tuesday, 15:55-17:35. SIST 1A-106; TA: Ziyuan Hu
Lab 6	Tuesday, 15:55-17:35. SIST 1A-108; TA: Zheqi Sheng
Lab 7	Tuesday, 19:35-21:15. SIST 1B-106; TA: Guanzhou Hu

Course Information

- Course Web: http://shtech.org/course/ca/
- Acknowledgement: Instructors of UC Berkeley's CS61C: http://www-inst.eecs.berkeley.edu/~cs61c/
- Instructor:
  - Sören Schwertfeger
- 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*, 2nd Edition
  - Barroso & Holzle, *The Datacenter as a Computer, 2nd Edition*
- Piazza:
  - Every announcement, discussion, clarification happens there

Course Grading

- Projects: 33%
- Homework: 17%
- Lab: 5%
- Exams: 40%
  - Midterm 1: 10%
  - Midterm 2: 10%
  - Final: 20%
- Participation: 5%
  - (in class, **in piazza**, non credit parts of HW/ projects, help other during labs)

- CA will use Autolab for grading
  - Setup maintained by TAs
    - => Please be patient and report any bugs/ problems in piazza or (if sensitive) via email.
  - Update your hosts file:
    - https://en.wikipedia.org/wiki/Hosts_(file)
    - Add:
    - 10.19.124.103 autolab.shanghaitech.edu.cn
  - Will only be available on campus!
  - Your logins will be created soon!
  - http://autolab.shanghaitech.edu.cn

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!

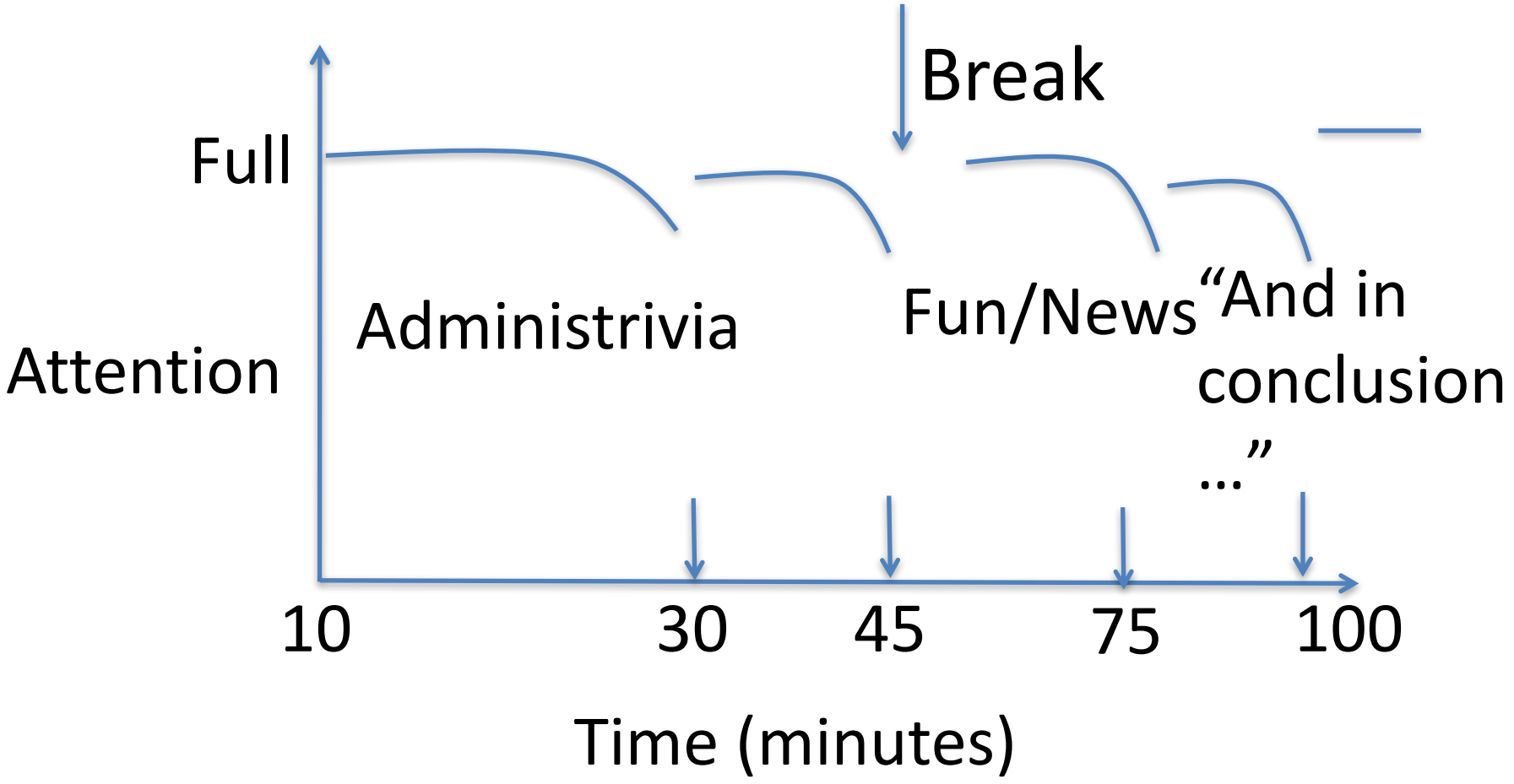
Policy on Assignments and Independent Work

- **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)
- 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**

Labs & HW1

- Labs: Find one partner for your lab-work from you lab class!
  - Labs start next week
  - Projects are done in 2-person teams again this year!
- HW1 will be posted this week on Autolab.
- Announcements will be posted on piazza only!

Architecture of a typical Lecture



Agenda

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