University of Pittsburgh

CS 1541 Introduction

Instructor Introduction

Course Introduction

Wonsun Ahn
Department of Computer Science
School of Computing and Information

Instructor Introduction

My Technical Background

- Wonsun Ahn
  - First name is pronounced *one-sun* (if you can manage)
  - Or you can just call me Dr. Ahn (rhymes with *naan*)
- PhD in CPU Design and Compilers
  - University of Illinois at Urbana Champaign
- Industry Experience
  - Software engineer, field engineer, technical lead, manager
  - Bluebird Corporation (70-person startup company)
    - Manufactures industrial hand-held devices from top to bottom
    - Me: Built software stack based on Windows Embedded
  - IBM Research (thousands of people)
    - Does next-gen stuff like carbon nanotubes, quantum computers
    - Me: Designed supercomputers for ease of parallel programming

My World View

- Everything is connected
  - Pandemic: If my neighbors catch the virus, so will I
  - Environment: If my neighbors pollute, I will feel the effects
  - Economy: Think of how the subprime mortgage crisis spread
- Zero-sum thinking (old way of thinking)
  - “If you get a larger slice of the pie, I get a smaller slice.”
  - Therefore, if you lose, I win (and vice versa)
- Zero-sum thinking no longer works
  - If you catch the virus, do I become safer from the virus?
- Collaboration is replacing competition

Collaboration is Replacing Competition

- Is happening in all spheres of life
- Collaboration is also happening in the IT industry
  - The *open source* movement
  - Increasing importance of the software/hardware *ecosystem*
  - Increasing importance of the developer *community*
- Collaboration is also important for learning
  - During my undergrad years, what do I remember best?
  - Stuff that I explained to my classmates
  - Stuff that my classmates taught me

Supporting Collaborative Learning

- I *never* grade on a curve
  - You will not be competing against your classmates
  - You are graded on your own work on an absolute scale

- You will be a member of a Team
  - You are already a member of the class on Microsoft Teams
  - I encourage you to be on Teams at most times (I will too)
    - You can install app on both laptop and cell phone
  - If you have a question, you can ask in the Team “Posts” tab
    - Either your classmate or your instructor will answer
  - You can chat with any individual on the Team
    - “Manage Team” item in the “…” Team context menu

Supporting Collaborative Learning

- You will be a member of a Group
  - On Teams, you are part of a chat group of 7~8 members
  - Members are a good mix of in-person and online students
  - Your instructor is also a member of each Group
  - It is a smaller support group where you can talk more freely

- You are allowed to discuss TopHat lecture questions
  - The goal is no-student-left-behind (pun intended)
  - Discuss answers on Teams even before submitting them
  - Form a basis of knowledge for doing homeworks and exams

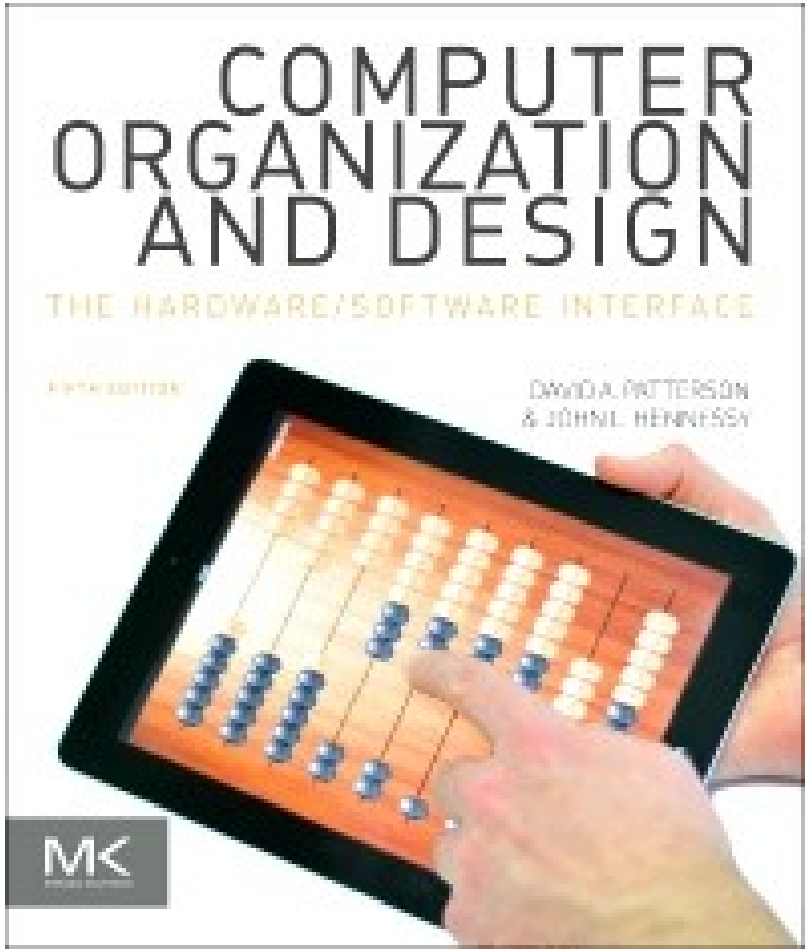
Course Introduction

Structure of the Course

- (45% of grade) Three midterms
- (20% of grade) Two projects
  - Implementing a CPU simulator using C programming language
- (20% of grade) Four homeworks
- (15% of grade) Participation
  - TopHat lecture questions, Teams participation
- Class resources:
  - Canvas: announcements, Zoom meetings, recorded lectures
  - GitHub: syllabus, lectures, homeworks, projects
  - Tophat: online lecture questions
  - GradeScope: homework / projects submission, grading and feedback
  - Microsoft Teams: Online / off-line communication

Textbook (You Probably Have it)

- "Computer Organization and Design - The Hardware/Software Interface" by David Patterson and John Hennessy Fifth Edition - Morgan & Kaufmann.



For More Details

- Please refer to the course info page: https://github.com/wonsunahn/CS1541_Fall2020/blob/master/course-info.md

- Please follow the course schedule syllabus: https://github.com/wonsunahn/CS1541_Fall2020/blob/master/syllabus.md

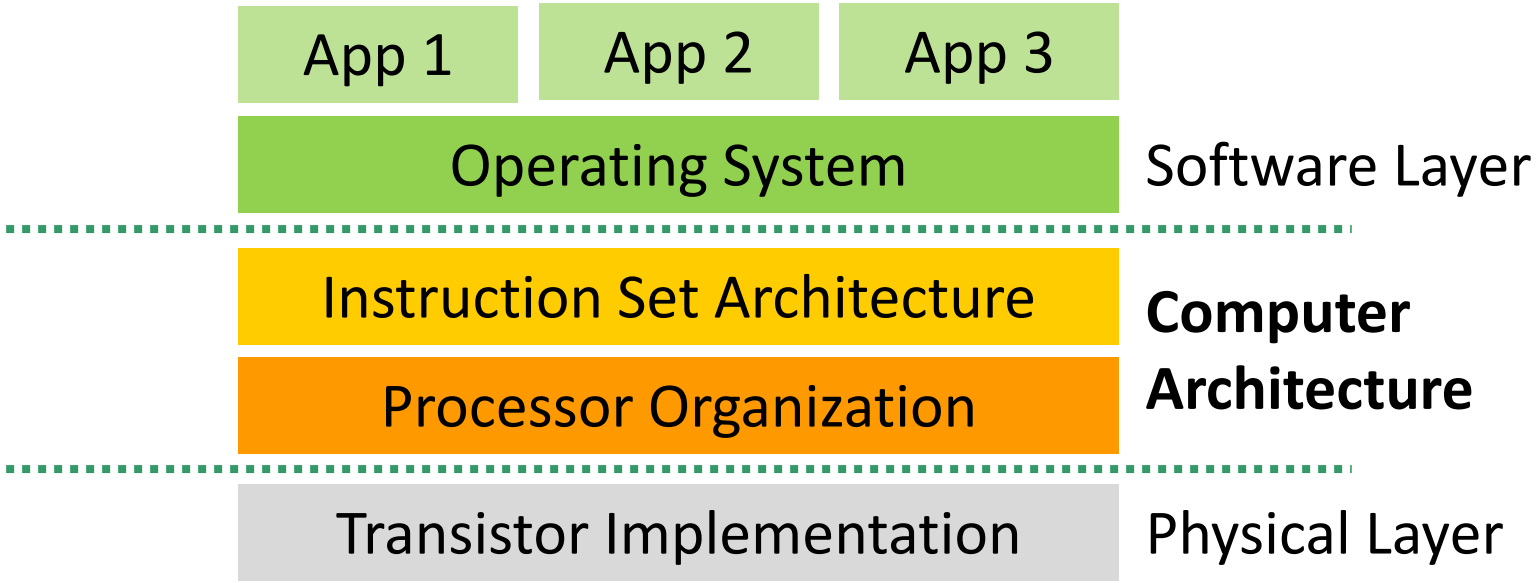
TODO

- Submit the TopHat survey questions (due 8/21)

What is Computer Architecture?

- At a high-level: how a computer is built
  - Computer here meaning the processor (CPU)
- You probably heard of a similar term before: ISA
  - ISA (Instruction Set Architecture)
- Review: what is defined by an ISA?
  - Set of instructions usable by the computer
  - Set of registers available in the computer
  - Functional attributes are clearly defined (it's the interface)
- What is *not* defined by an ISA?
  - *Speed* of computer
  - *Energy efficiency* of computer
  - *Reliability* of computer
  - Performance attributes are undefined (intentionally so)

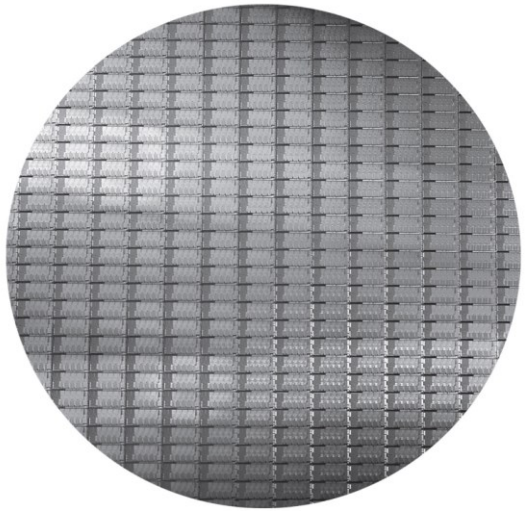
Computer Architecture Defined



- Computer Architecture = ISA + Processor Organization
  - Processor organization is also called *Microarchitecture*
- Performance is decided by:
  - Processor organization (internal design of the processor)
  - Transistor implementation (semiconductor technology)

Scope of Class

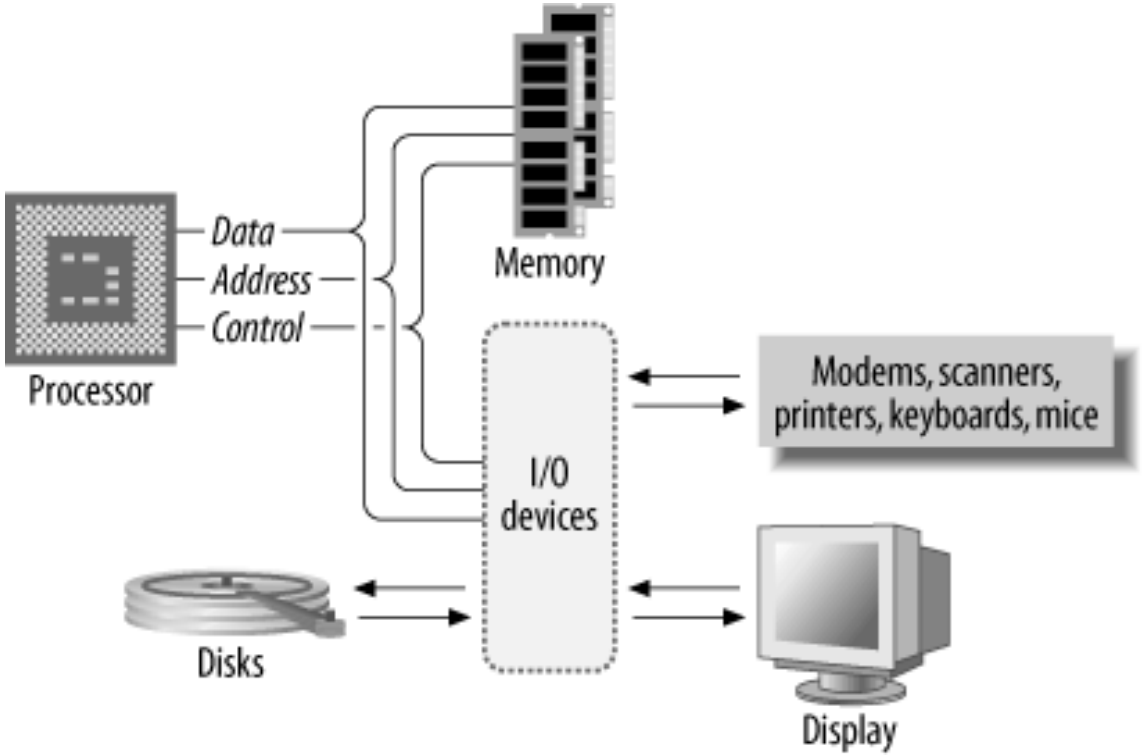
- Physical layer is beyond the scope of the class



- We will focus mostly on processor organization
  - And how performance goals are achieved

Scope of Class

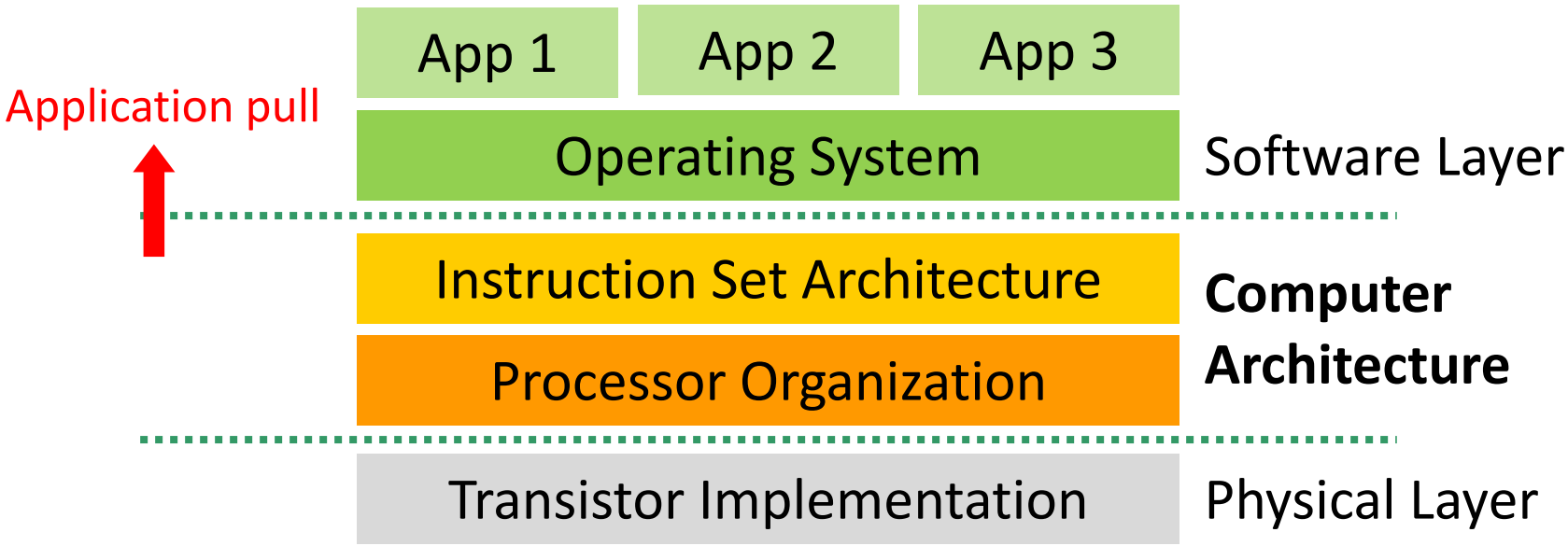
- Computer architecture is part of system architecture



- Other components beside processor is beyond the scope

Application Pull

- Different applications pull in different directions



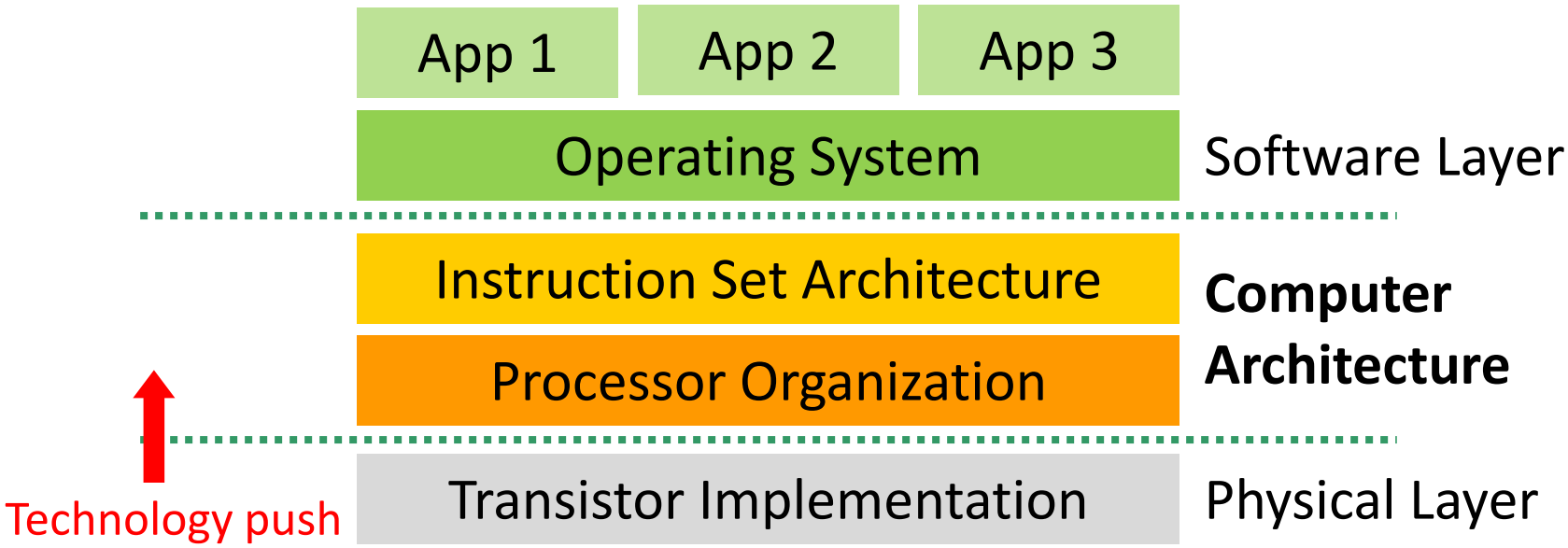
  - Real-time app (e.g. Game): *Short latency*
  - Server app: *High throughput*
  - Mobile app: *High energy-efficiency* (battery life)
  - Mission critical app: *High reliability*
- An app typically has multiple goals that are important

Application Pull

- Some goals can be incompatible
  - E.g. Speed and energy-efficiency are incompatible
    - Running is faster than walking but uses more energy
    - A Ferrari is faster than a Prius but has worse fuel efficiency
  - E.g. Reliability is incompatible with many other goals
    - If you use redundancy, you use twice the amount of energy
- Even when sharing a goal, apps have unique needs
  - Scientific apps need lots of *floating point units* to go fast
  - Database apps need lots of *memory cache* to go fast
- An architecture is a compromise among all the apps
  - When app achieves market critical mass, designs diverge (Mobile chips / Server chips / GPUs / TPUs diverged)
  - Sometimes even ISAs diverge (GPUs and TPUs)

Technology Push

- Trends in technology pushes architecture too



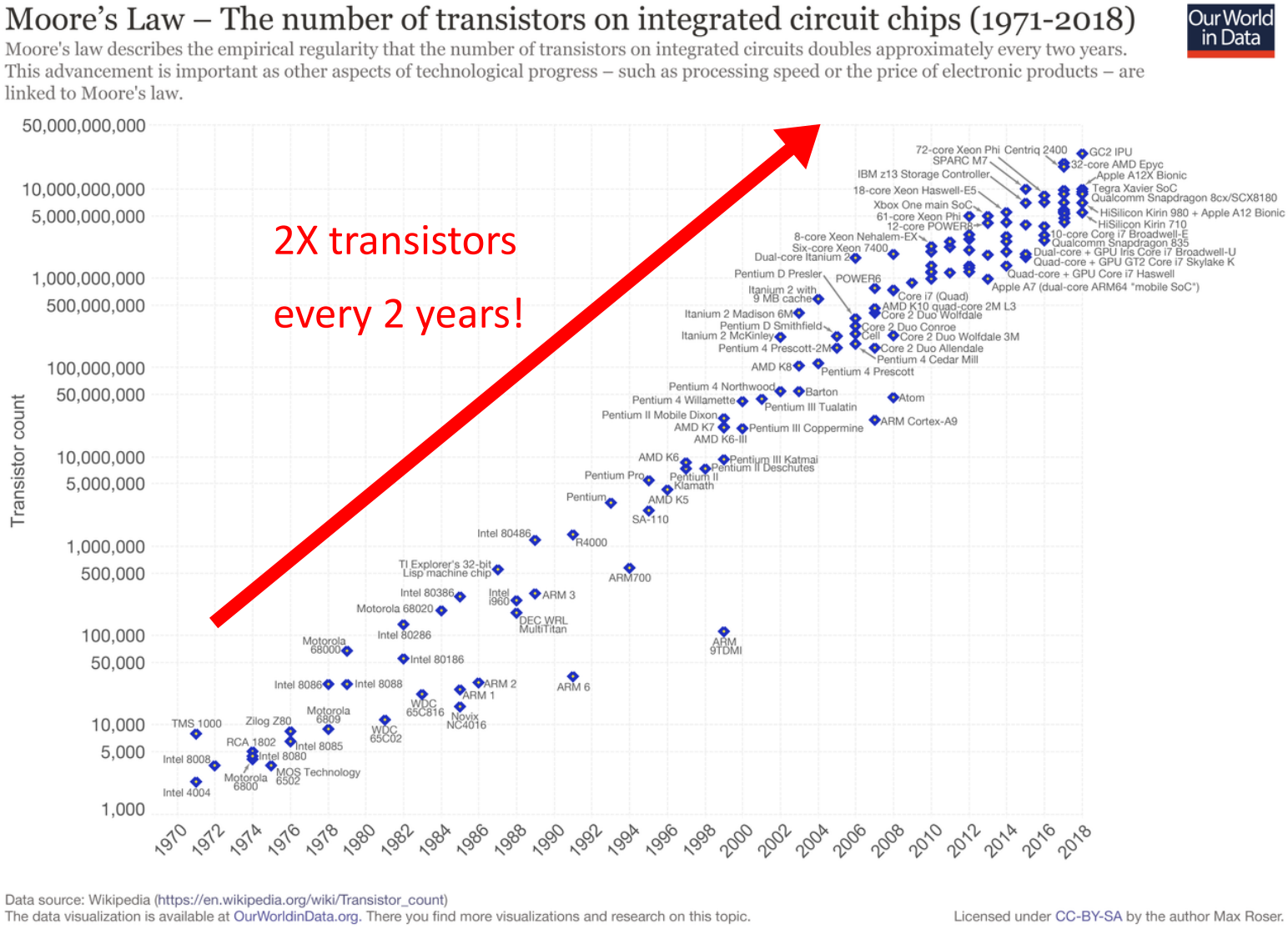
  - Trends can be *advances* in technology
  - Trends can be *constraints* technology couldn't overcome
- "Technology" in CPU design refers to the physical layer
  - Manufacturing technology used for transistor implementation

Advances in Technology

- Technology has been advancing at lightning speed
- Architecture and IT as a whole were beneficiaries

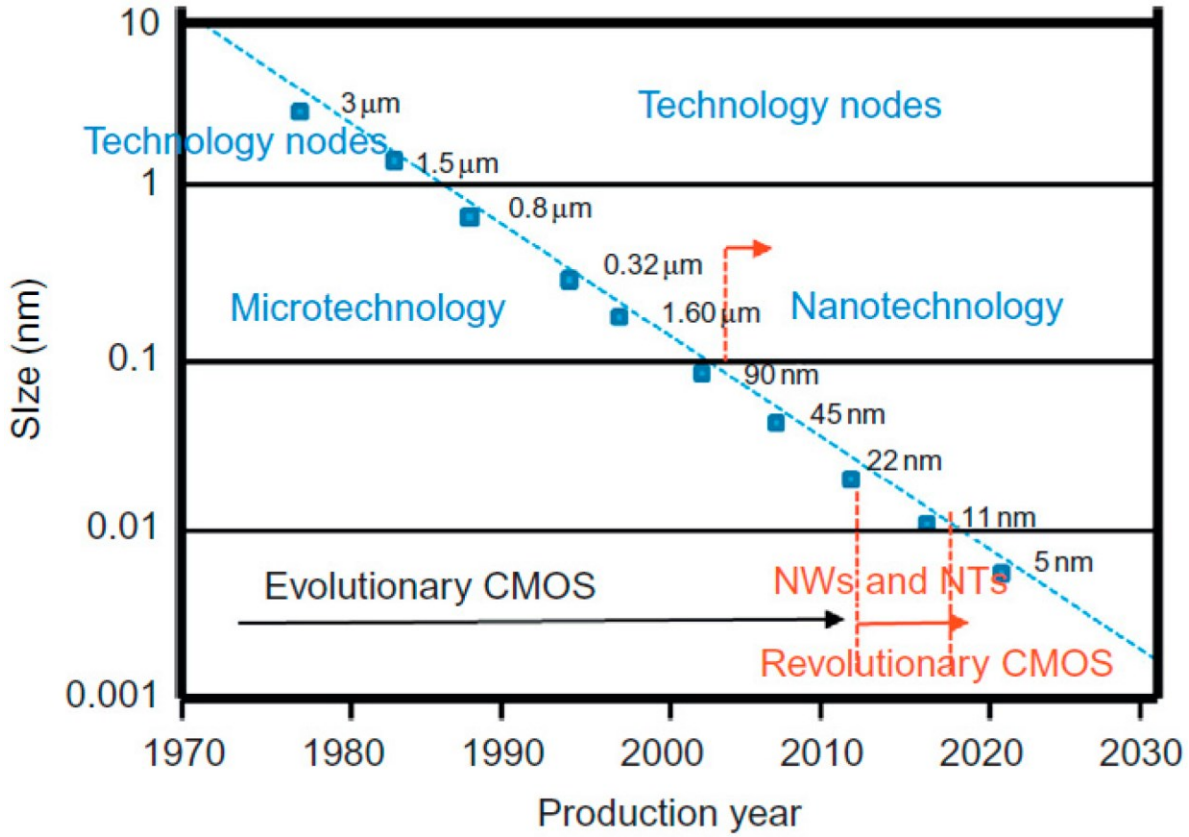
- Technology advance is summarized by *Moore's Law*
  - You probably heard of it at some point. Something about …
  - "X doubles every 18-24 months at constant cost"
- Is X:
  - CPU performance?
  - CPU clock frequency?
  - Transistors per CPU chip?
  - Area of CPU chip?

Moore's Law



Data source: Wikipedia (https://en.wikipedia.org/wiki/Transistor_count)
The data visualization is available at OurWorldinData.org. There you find more visualizations and research on this topic.

Miniaturization of Transistors



Data source: Radamson, H.H.; He, X.; Zhang, Q.; Liu, J.; Cui, H.; Xiang, J.; Kong, Z.; Xiong, W.; Li, J.; Gao, J.; Yang, H.; Gu, S.; Zhao, X.; Du, Y.; Yu, J.; Wang, G. Miniaturization of CMOS. *Micromachines* **2019**, *10*, 293.

- Moore's Law has been driven by transistor miniaturization
  - CPU chip area hasn't changed much

Future of Moore's Law

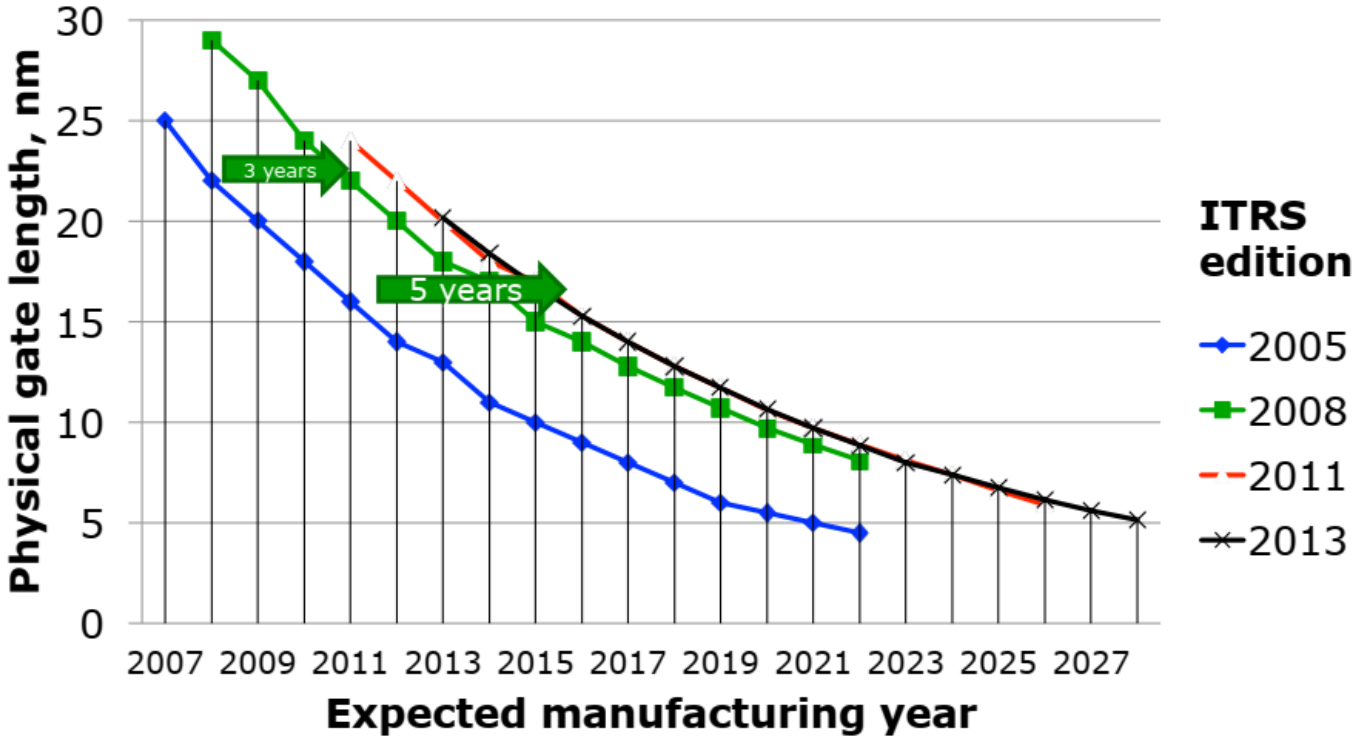
- The semiconductor industry has produced roadmaps
  - Semiconductor Industry Association (SIA): 1977~1997
  - International Technology Roadmap for Semiconductors (ITRS): 1998~2016
  - International Roadmap for Devices and Systems (IRDS): 2017~Present
- IRDS Lithography Projection (2020)

Year of Production	2018	2020	2022	2025	2028	2031	2034
Technology Node (nm)	7	5	3	2.1	1.5	1.0	0.7

  - Looks like Moore's Law will continue into foreseeable future
  - IRDS does not project significant increase in CPU chip size
  - Increases in transistors will come from *transistor density*

IRDS isn't Perfect

- ITRS (predecessor of IRDS) has made corrections before



  - After all, you are trying to predict the future
  - But architects rely on the roadmap to design future processors

Moore's Law and Performance

- Million-dollar question:
  Did Moore's Law result in higher performance CPUs?

- Please go to your respective Teams chat groups
  - But stay in the Zoom room and use only chat on Teams
  - To have chat content accessible to asynchronous students

1. Get to know each other
2. And then try to answer the following questions:
   - What do you think? Are CPUs getting faster?
   - If not, why do you think so? If yes, again why do you think so?
3. After 10 minutes, we will share discussions with class

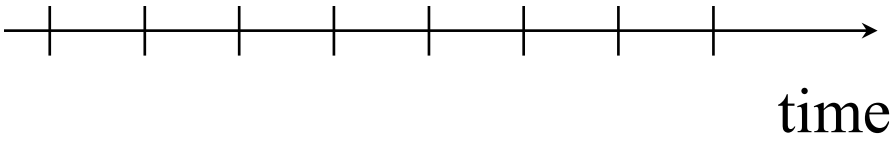
Are CPUs getting Faster?

- Yes!
- Clock speeds are increasing, power draw is decreasing – Andrew
- More cores – Jason
- Visual Studio compilation is actually faster - Nick

- No!
- Clock speeds are plateauing recently – Jason
- Seems stagnant from user's perspective e.g. Visual Studio - Josh

Components of Execution Time

- Processor activity happens on clock “ticks” or cycles



  - On each tick, bits flow through logic gates and are latched

- Execution time = $\frac{\text{seconds}}{\text{program}}$

$$\frac{\text{seconds}}{\text{program}} = \frac{\text{cycles}}{\text{program}} \times \frac{\text{seconds}}{\text{cycle}}$$

$$= \frac{\text{instructions}}{\text{program}} \times \frac{\text{cycles}}{\text{instructions}} \times \frac{\text{seconds}}{\text{cycle}}$$

Improving Execution Time

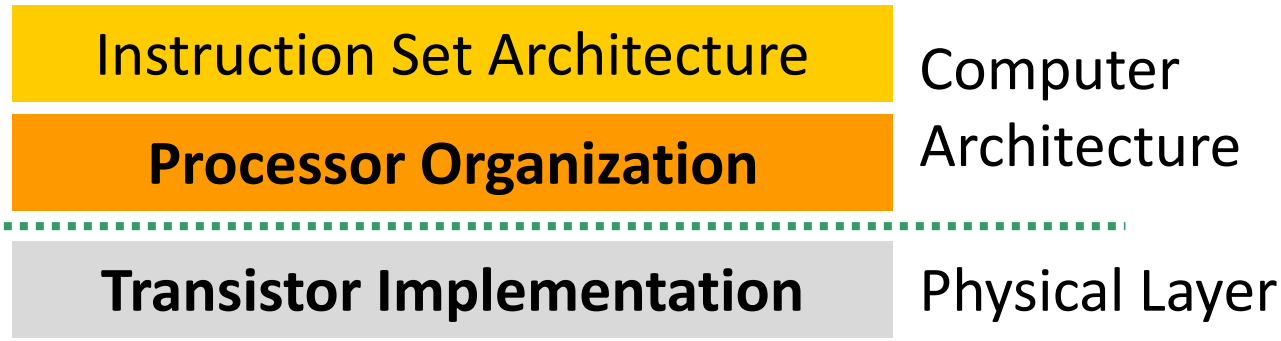
$$\frac{\text{instructions}}{\text{program}} \times \frac{\text{cycles}}{\text{instructions}} \times \frac{\text{seconds}}{\text{cycle}}$$

- Improving $\frac{\text{seconds}}{\text{cycle}}$:
  - Clock frequency = $\frac{\text{cycles}}{\text{second}}$ = reverse of $\frac{\text{seconds}}{\text{cycle}}$
  - Higher clock frequency (GHz) leads to shorter exec time
- Improving $\frac{\text{cycles}}{\text{instructions}}$:
  - Also known as CPI (Cycles Per Instruction)
  - IPC (Instructions Per Cycle) = $\frac{\text{instructions}}{\text{cycles}}$ = reverse of $\frac{\text{cycles}}{\text{instructions}}$
  - Higher IPC leads to shorter execution time
- Improving $\frac{\text{instructions}}{\text{program}}$:
  - Less instructions leads to shorter execution time
  - ISAs that do a lot of work with one instruction shortens time

Moore's Law and Performance

- Million-dollar question:
  Did Moore's Law result in higher performance CPUs?

- Law impacts both architecture and physical layers



  - Processor Organization: many more transistors to use in design
  - Transistor Implementation: smaller, more efficient transistors