



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



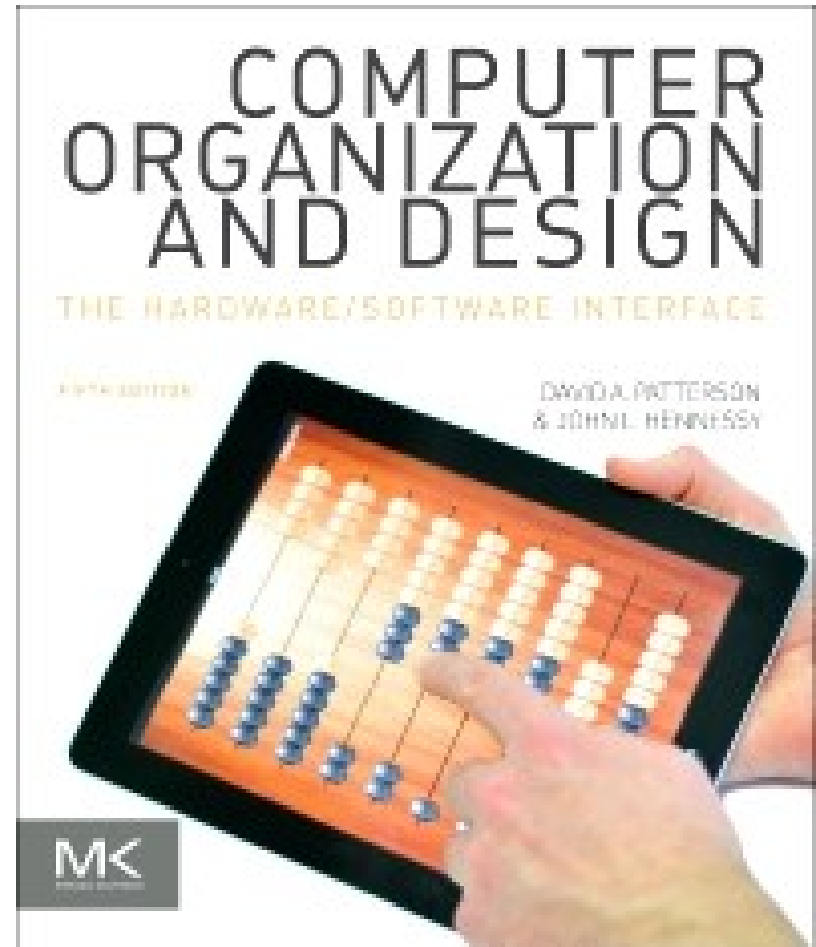
Structure of the Course

- (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



12

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

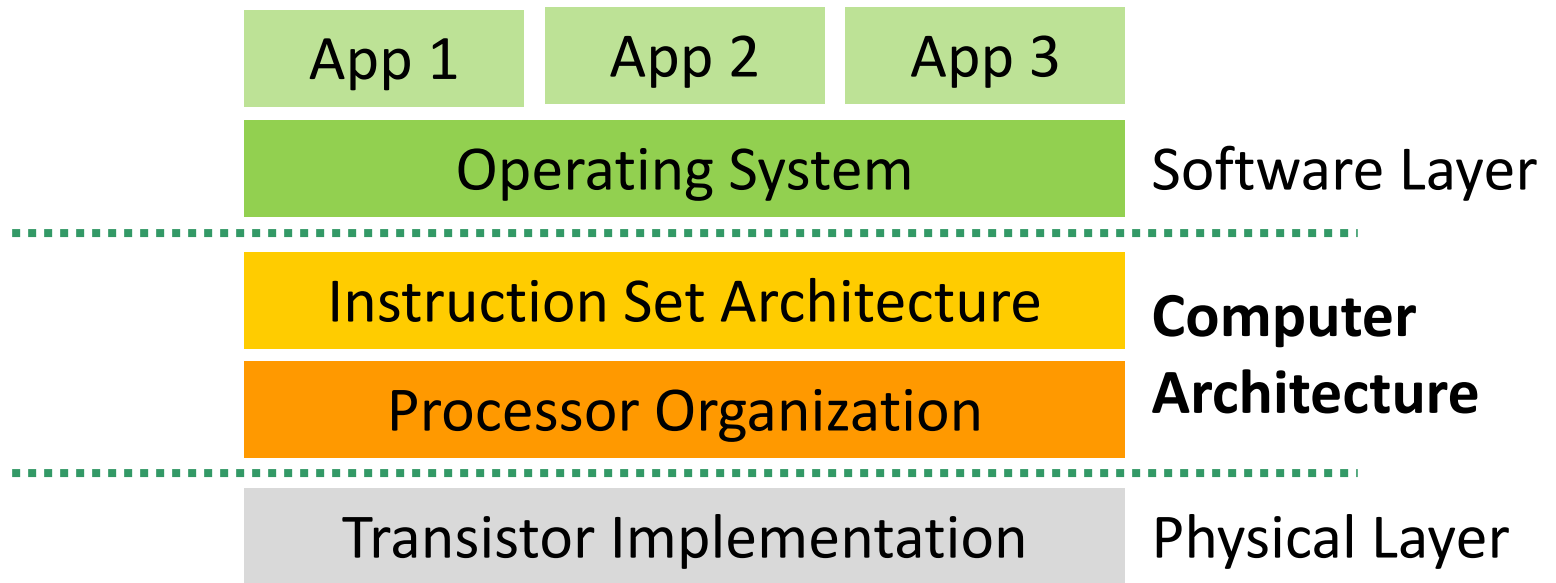


What is Computer Architecture?

- 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

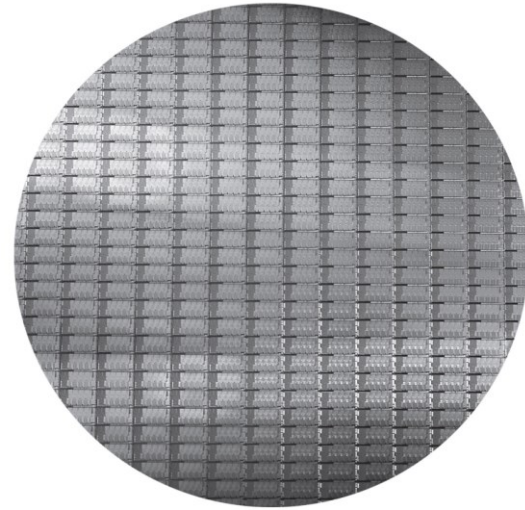


- **Computer Architecture = ISA + Processor Organization**
 - Processor organization is also called *Microarchitecture*
- Performance is decided by:
 - Processor organization (internal design of the processor)
 - Transistor implementation (semiconductor technology)



Scope of Class

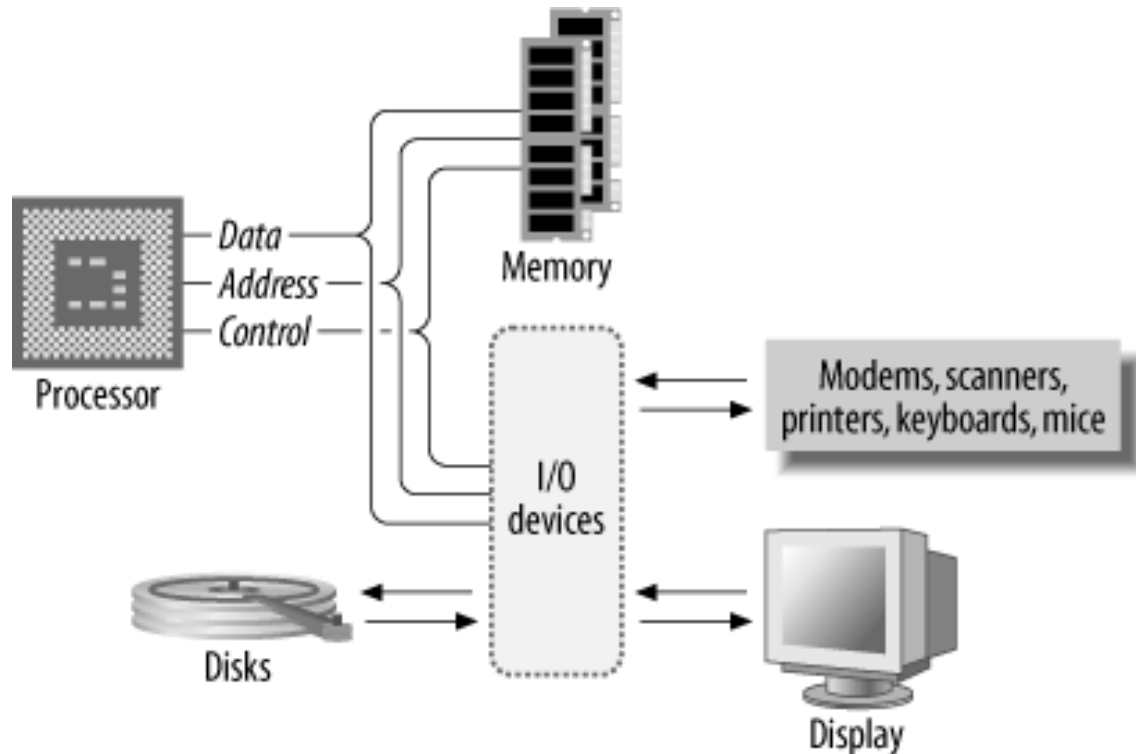
- Physical layer is beyond the scope of the class
- We will focus mostly on processor organization
 - And how performance goals are achieved





Scope of Class

- Computer architecture is part of system architecture

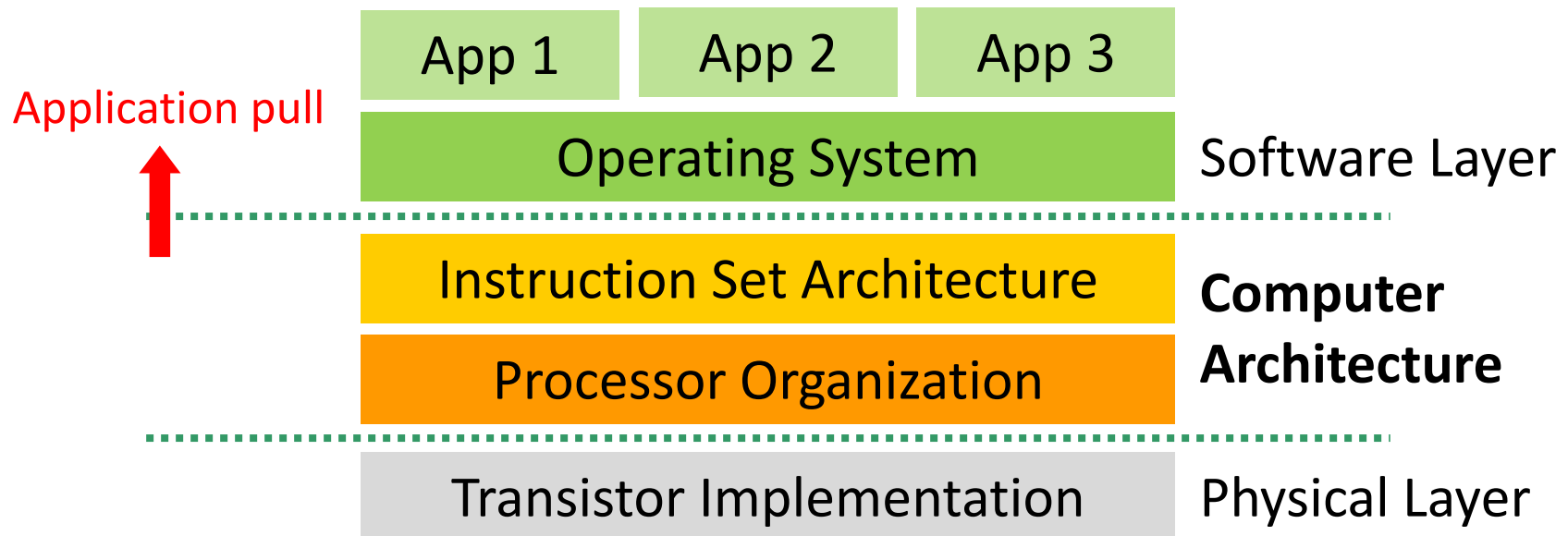


- Other components beside processor is beyond the scope



What are the Performance Goals?

- 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



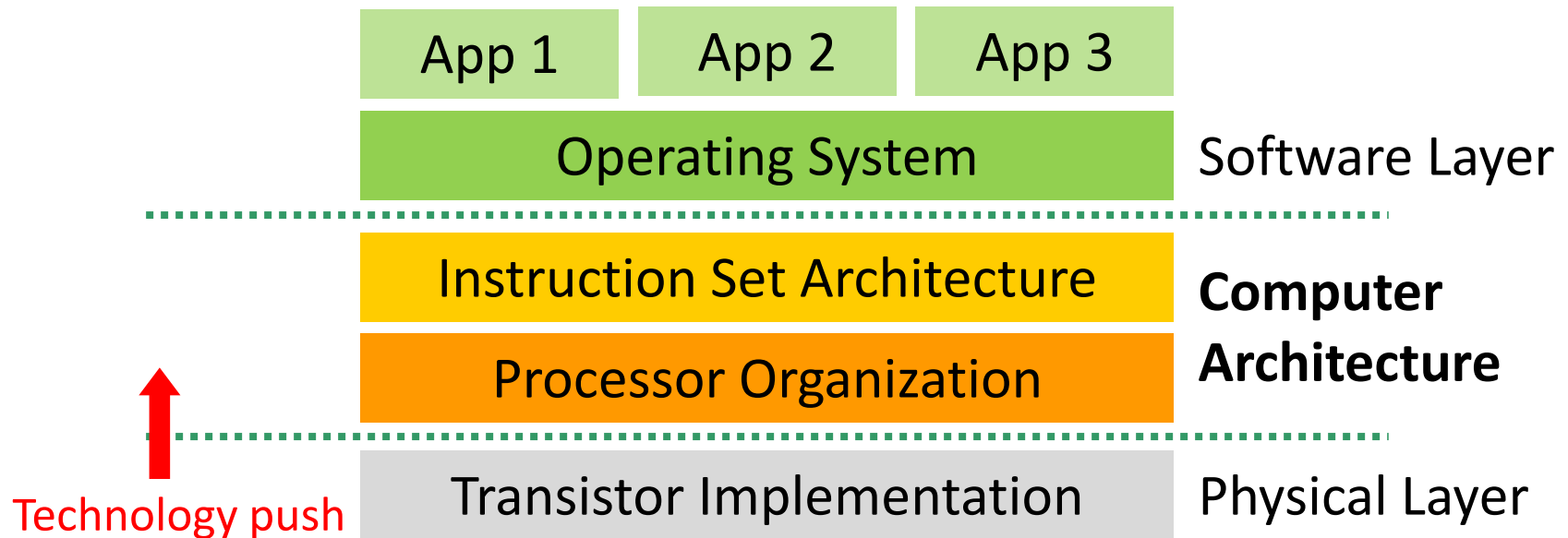
What are the Performance Goals?

- 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* to go fast
- An architecture is a **compromise** among all the apps
 - When app achieves market critical mass, architectures diverge (Mobile chips / Server chips / GPUs / TPUs diverged)
 - Sometimes even ISAs diverge (GPUs and TPUs)



There are also Technology Constraints

- Constraints in technology pushes architecture too



- *Power Wall*: Thermal Design Power (TDP) constraint
- *Bandwidth Wall*: Constraint in number of pins in CPU
- *Memory Wall*: Constraint in system bus speed to memory

- Processor must be designed with all constraints in mind

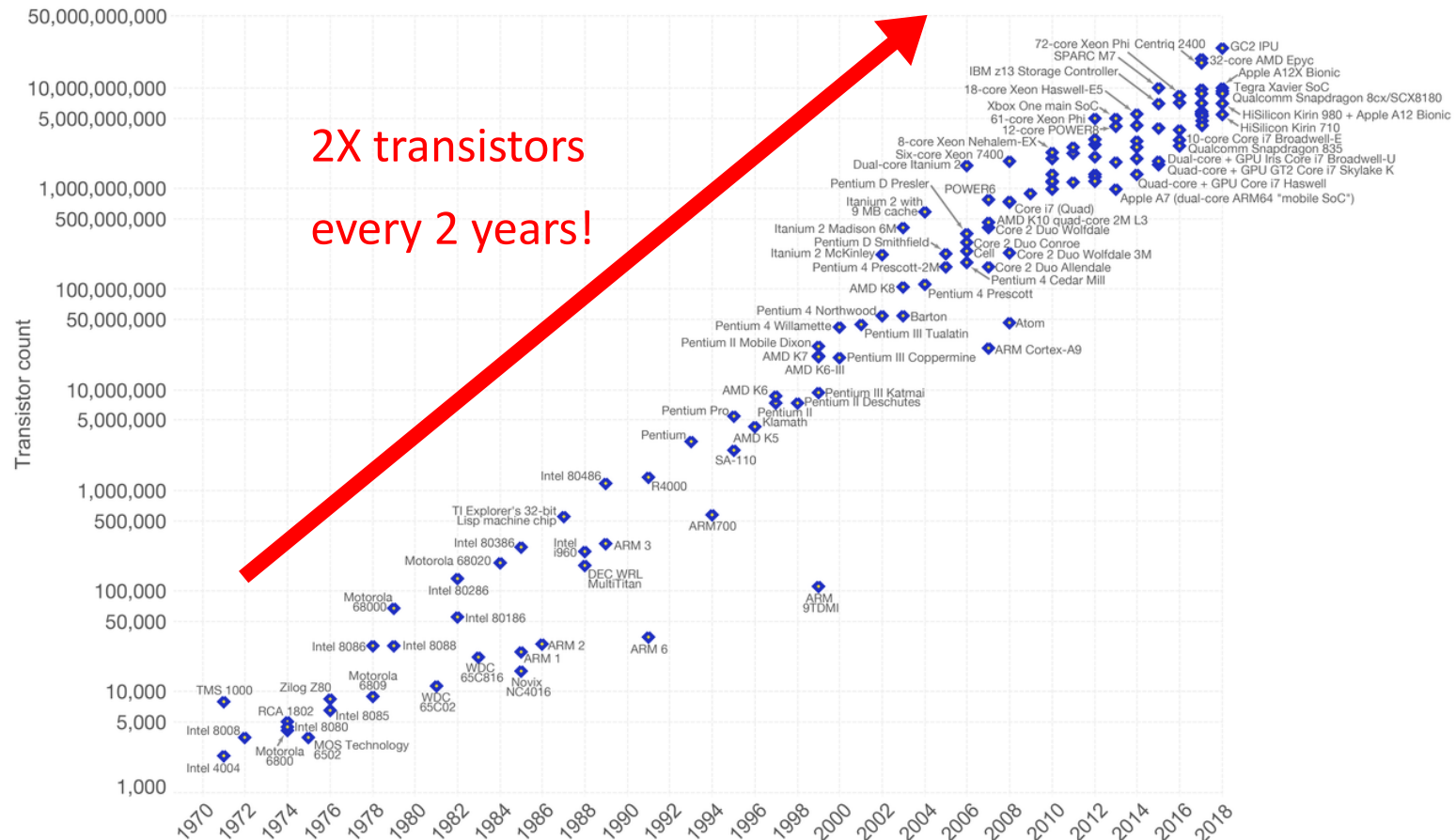


Moore's Law

Moore's Law – The number of transistors on integrated circuit chips (1971-2018)

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

OurWorld
in Data



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

The data visualization is available at [OurWorldinData.org](https://www.ourworldindata.org). There you find more visualizations and research on this topic.

Licensed under CC-BY-SA by the author Max Roser.



What happened to Performance?

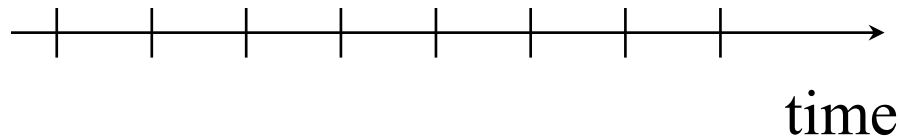
- Did it improve exponentially like Moore's law?
 - Did transistor count translate to performance?

- What do you think? Are computers getting faster?



Components of Performance

- Processor activity happens on clock “ticks” or cycles



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

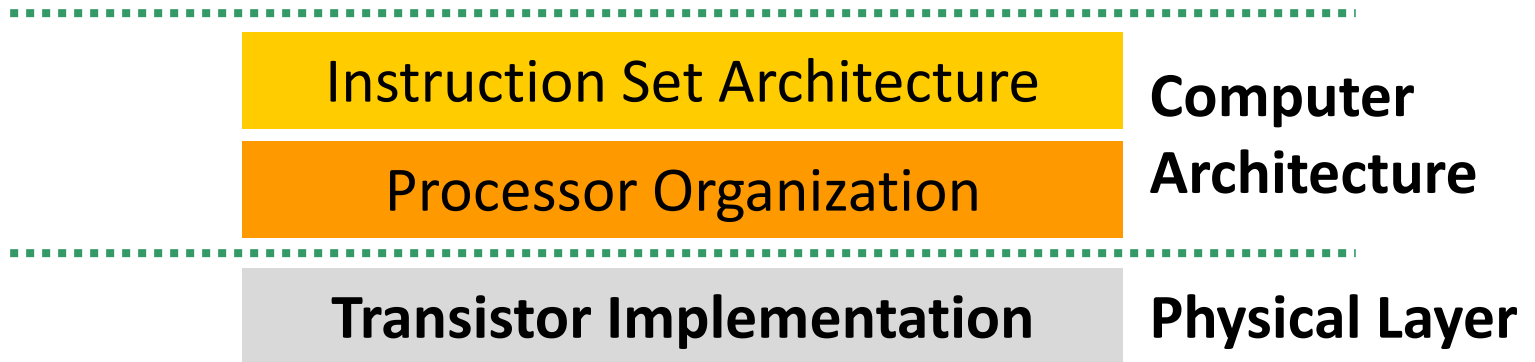
$$\begin{aligned}\frac{\text{seconds}}{\text{program}} &= \frac{\text{cycles}}{\text{program}} \times \frac{\text{seconds}}{\text{cycle}} \\ &= \frac{\text{instructions}}{\text{program}} \times \frac{\text{cycles}}{\text{instructions}} \times \frac{\text{seconds}}{\text{cycle}}\end{aligned}$$

- Clock frequency = $\frac{\text{cycles}}{\text{second}}$ = reverse of $\frac{\text{seconds}}{\text{cycle}}$
 - Higher clock frequency (GHz) leads to higher performance



What Determines Clock Frequency?

- Both the Computer Architecture and Physical Layer



- Physical layer determines how fast each transistor is
- Architecture determines number of transistors between clock ticks
 - Some designs have a lot of logic gates between latches
 - Some designs have less logic gates between latches

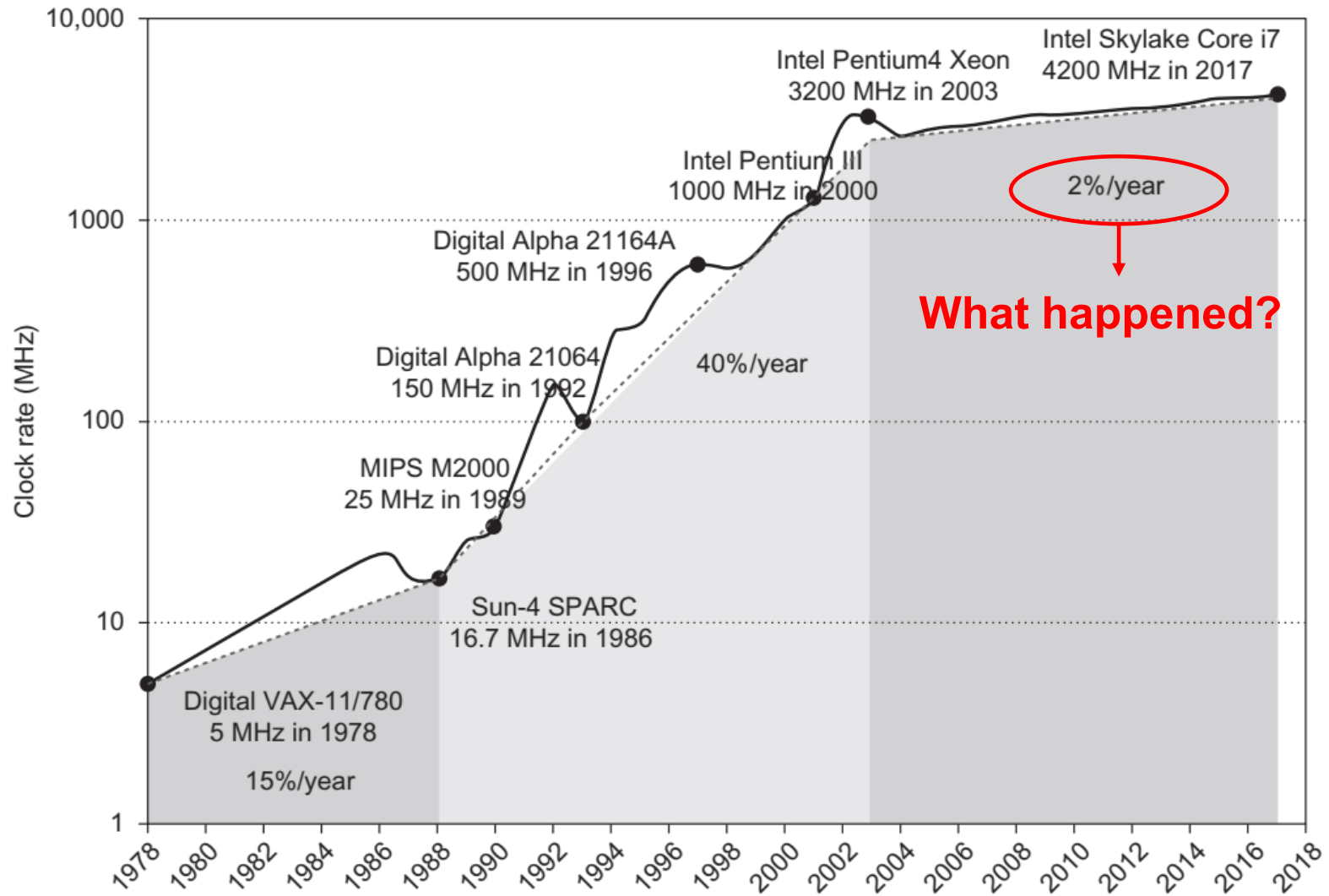


What Determines Clock Frequency?

- Historically frequency has been driven by physical layer
 - Improvements in transistor speed was primary driver
- Architecture contributed but in a limited way
 - We will learn about a technique called pipelining
 - Limit to how few transistors you can have between latches
- But ...



Processor Clock Frequency





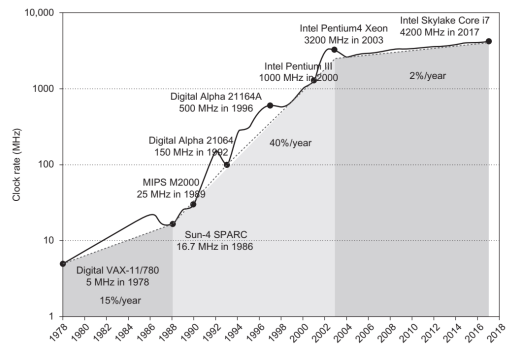
The “Power Wall” Happened

- Power (Watt): energy CPU uses per second
- Power \propto Heat Generation
 - Processor has a Thermal Design Power (TDP) cap
 - TDP is determined by cooling system (heat dissipation)
 - TDP has not improved much (usually just a CPU fan)
- Power \propto Transistor Count X Clock Frequency
 - Due to Moore’s law, transistor count increases exponentially
 - So power demand increases exponentially too!
- This is the “Power Wall”
 - The discrepancy between power demand and TDP
 - Clock frequency must be reduced to meet TDP



End of Dennard Scaling

- But why did the power wall only happen around 2003?



- Answer: $\text{Power} = V^2 / R \propto \text{Voltage}^2$
 - Each generation of transistors came with reductions in voltage
 - If voltage is reduced by 30%, power is reduced by 65%
 - This scaling trend is called *Dennard Scaling*
 - So clock frequency could keep on increasing until 2003
 - ... then Dennard Scaling came to an end