Welcome to *ECE 116C/CS 151B*

Fall 2023

Lecture 1 – Introduction

(ECE M116C- CS M151B) Computer Architecture Systems

Nader Sehatbakhsh
Department of Electrical and Computer Engineering
University of California, Los Angeles

Plan for today

- Brief Introduction
- Administrivia
- Q/A
- Basic Concepts

About Me

- Got my PhD in *computer science* from Georgia Tech in 2020.
- *Fourth* time teaching this course at **UCLA;**

- ***Research interests:***
  - Computer Architecture with emphasis on security and privacy; IoT security and privacy.

About Me

- Got my PhD in *computer science* from Georgia Tech in 2020.
- *Fourth* time teaching this course at **UCLA;**

- ***Research interests:***
  - Computer Architecture with emphasis on security and privacy; IoT security and privacy.

★ *Contact me if you are interested in doing research!*
★ *Special Topic Course on Computer Engineering Security in Winter.*

About Me

Nader Sehatbakhsh <nsehat@ee.ucla.edu>

How to pronounce my name

- Naaaaaaa- der
- Se – hat – bakhsh

How to pronounce my name

- Naaaaaaa- der
- Se – hat – bakhsh

- You can call me:
  - Dr. Sehat / Professor / Professor Sehat / Nader

  (Don't call me Dr. Nader or Professor Nader.) – Please DON'T

Contact Information

- DM me on Campuswire.
  - You can contact me to ask questions or setting an appointment for office hours.
- Email me for more serious questions/issues:
  - nsehat@ee.ucla.edu
- Office Hours:
  - Thursdays 4-5 PM, In-person (Office: BH 6731-G).
  - Appointment (Zoom or in-person).

How was YOUR summer?

PJ in Summer

PJ in Fall

Anything interesting to share?
(bonus)

TAs

Justin Feng

Pooya Aghanoury

Fatemeh Arkannezhad

- DM on Campuswire.
- No discussion tomorrow.

Please Introduce Yourself on Padlet (link)

Lecture Format

- In-person lectures (unless we have to switch to online).
- Lectures will be recorded and posted online (YouTube).
  - Hopefully with good quality and within 24 hours.
  - Online Zoom attendance is not allowed.
- Attendance is encouraged but not mandatory (except for quizzes).
- *Please follow UCLA's COVID-19 guidelines.*

Slides

- Slides (in PDF) will be shared on the website by midnight the day before each lecture.
  - Link to website: https://ssysarch.github.io/ECE_M116C-CS_M151B/F23/schedule.html
  - Last year materials can be found in …/F22/…

- In-class notes will NOT be shared!

Campuswire

- Please enroll ASAP (details on Bruinlearn).
- Strongly encourage you to ASK and ANSWER questions on Campuswire.
  - Questions can be about the lectures, assignments, and relevant topics to the course. Everything is *anonymous!* *(except your participation scores)*
  - TAs and I will regularly check the website.

- Participate in the discussion. There will be bonus points at the end of the quarter.

Exams/Quizzes

- No Final (or midterm).

- 3 quizzes
  - In-person. 1-1.5-hour. Previous samples are on the website.
  - Dates:
    - Oct. 31 (Tue. W5), Nov. 21 (Tue. W8), Dec. 7 (Th. W10).
  - If you cannot attend, contact me ASAP.

Homework

- Several Homeworks (roughly 4-6)
  - Usually short.
  - Sometimes one or two mini programming questions.
  - Will be posted on Gradescope typically on Fridays. Submit on Gradescope.
  - No late submissions.
  - Deadline: usually one week from the posting date.

No SHARING!

Projects

- Two (larger) Projects / Computer Assignment
  - Designing a processor using C/C++
  - On the first you design the datapath and controller of a single-cycle processor.
  - The second project is the continuation of the first one (mostly), and will be about adding more advanced features.
  - Submission through Gradescope. Auto-graded.
  - *Late submission allowed under special circumstances.*
  - Honors Students…

Grading

- **Projects** 30%
- **Homework** 10%
- **Quizzes** 55%
- **Participation** 5%
  - Participation includes being active on Campuswire (posting questions and answers, discussions), OR class Participation.
- Some **bonuses** along the way …

Grading Policies

- One homework assignment can be dropped – no question asked.
- We might do weighted averaging for your quizzes.
- Grading: absolute grading; *giving as much A and B as possible*

97%: A+, 92%: A, 88%: A-,
80%: B+, 75%: B, 70%: B-,
65%: C+, 60%: C, 40%: D,
<40%: F

(These ranges might be scaled up.)

Participation

- You will earn points in two ways:
  - **Class participation:** each lecture, you need to check in using the QR code.
    - Use the exact same format each time you print your name. We will count those and if you use a different format our script won't catch it!
    - Make sure to use the correct password each time!
    - You need to attend at least 70% of the classes to get the full credit.
  - **Campuswire Reputation Points:** you can track your points on Campuswire. Each question, answer, like, etc. gives you points.
    - You need 50 points (might change) to receive full credit.

Participation

- Scan the QR code.
- Use this password: [redacted]

Navigating the Course Information

- Course Website:
  https://ssysarch.ee.ucla.edu/courses/ECE_M116C-CS_M151B/F23/index.html
  - Lectures, videos, assignments, projects

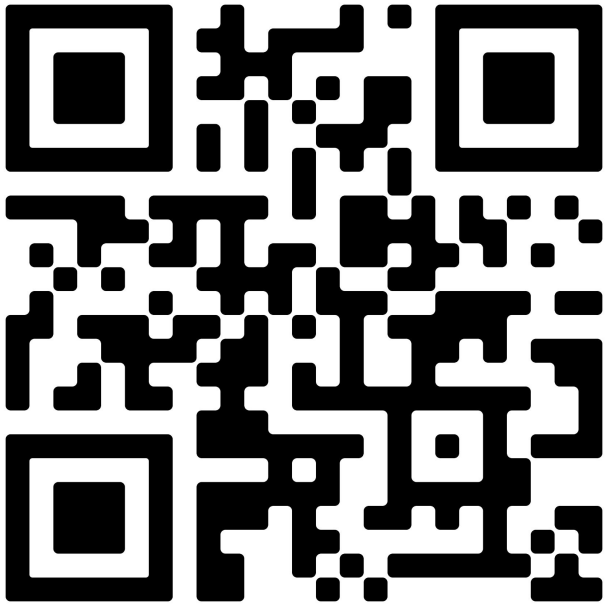
- Campuswire:
  - Announcements, Q/A, discussions, DM.

- Gradescope:
  - Submitting assignments and projects.

Waitlist and PTE

- The people already on the waitlist (6) will get in automatically.
- The class is (almost) full. We officially have ten more spots left and there seems to be an issue with the waitlist. To request for PTE, fill in the form in the next page.
- Requests will be considered in case-by-case basis, the priority will be given to those who are graduating and graduate students.

Scan this to request PTEs →



Discussion Sessions and TA Office Hours

- Two recorded videos (by two different TAs) will be posted each week (Thursday and/or Friday).

- The two TAs will hold office hours on Monday and Friday. (Mine is on Thursdays).
  - Details about the rooms and exact times will be posted on Campuswire.

- Ignore the discussion sessions.

Honors Section

- Classes on Thursdays at 2. No class this week and next week. Lectures will be mostly about the project.

- The plan is using a different project.
  - If you are interested in Verilog and hardware implementation, consider taking it.

Textbook

David A. Patterson and John L. Hennessy, Computer Organization and Design: the Hardware/Software Interface: RISC-V Edition

- Optional, we will assign readings from some chapters.
- We won't cover all the chapters.

Questions?

Why this course?

- Interaction between software and the underlying hardware.
  - Every computer scientist and engineer needs to know this.

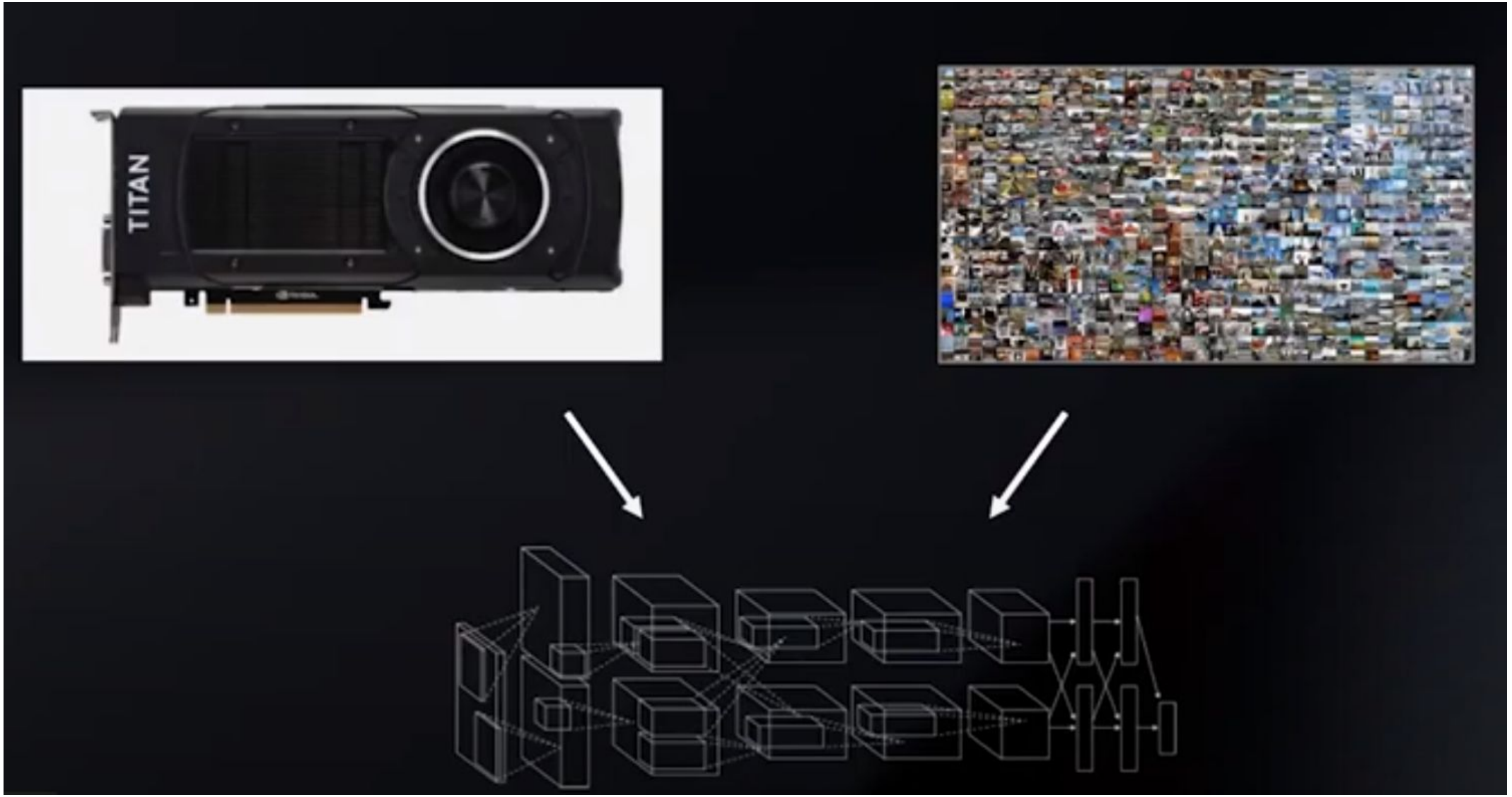
- What are the important metrics, what have been the trends, and where are we heading?

- What are the limitations and why?

Why (deep) machine learning is increasingly more capable?

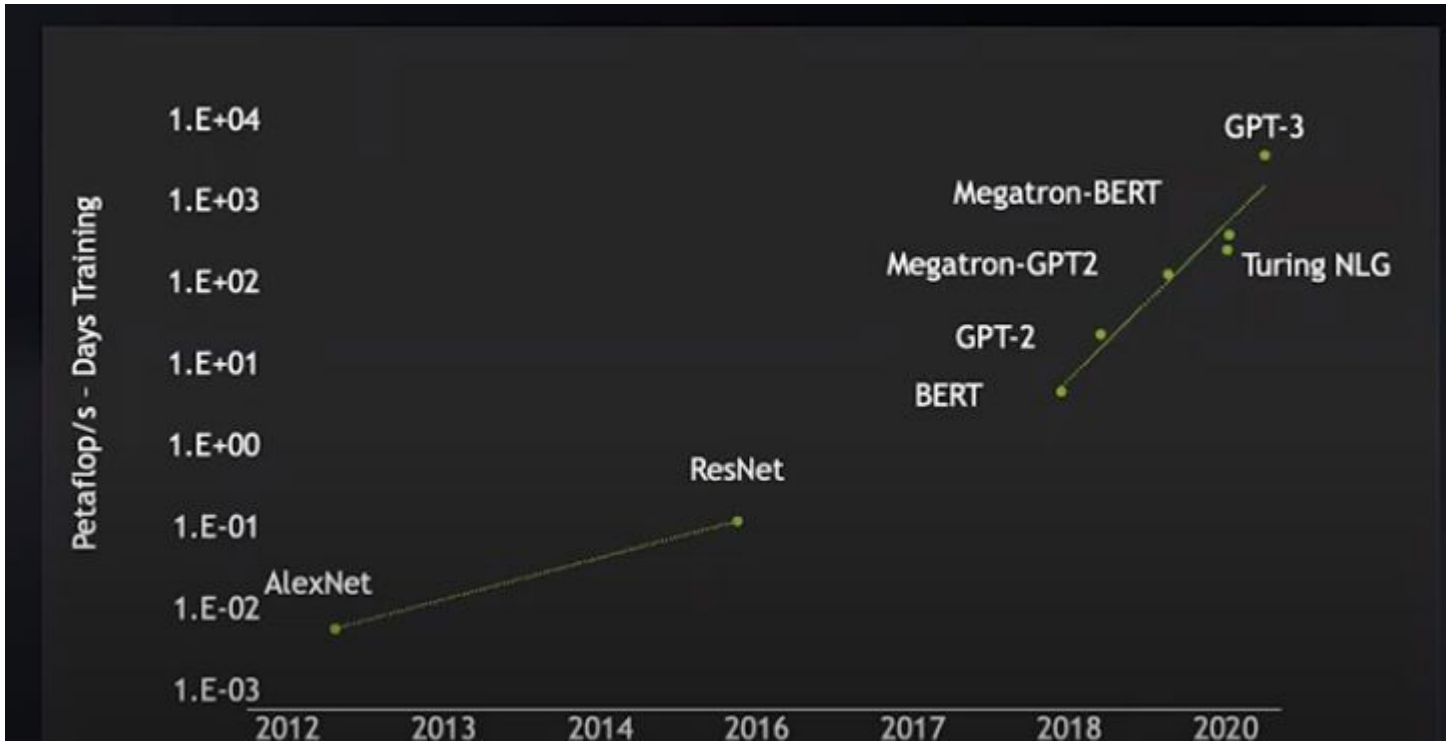
Why we didn't/couldn't invent ChatGPT 20 years ago?

Nader Sehatbakhsh <nsehat@ee.ucla.edu>

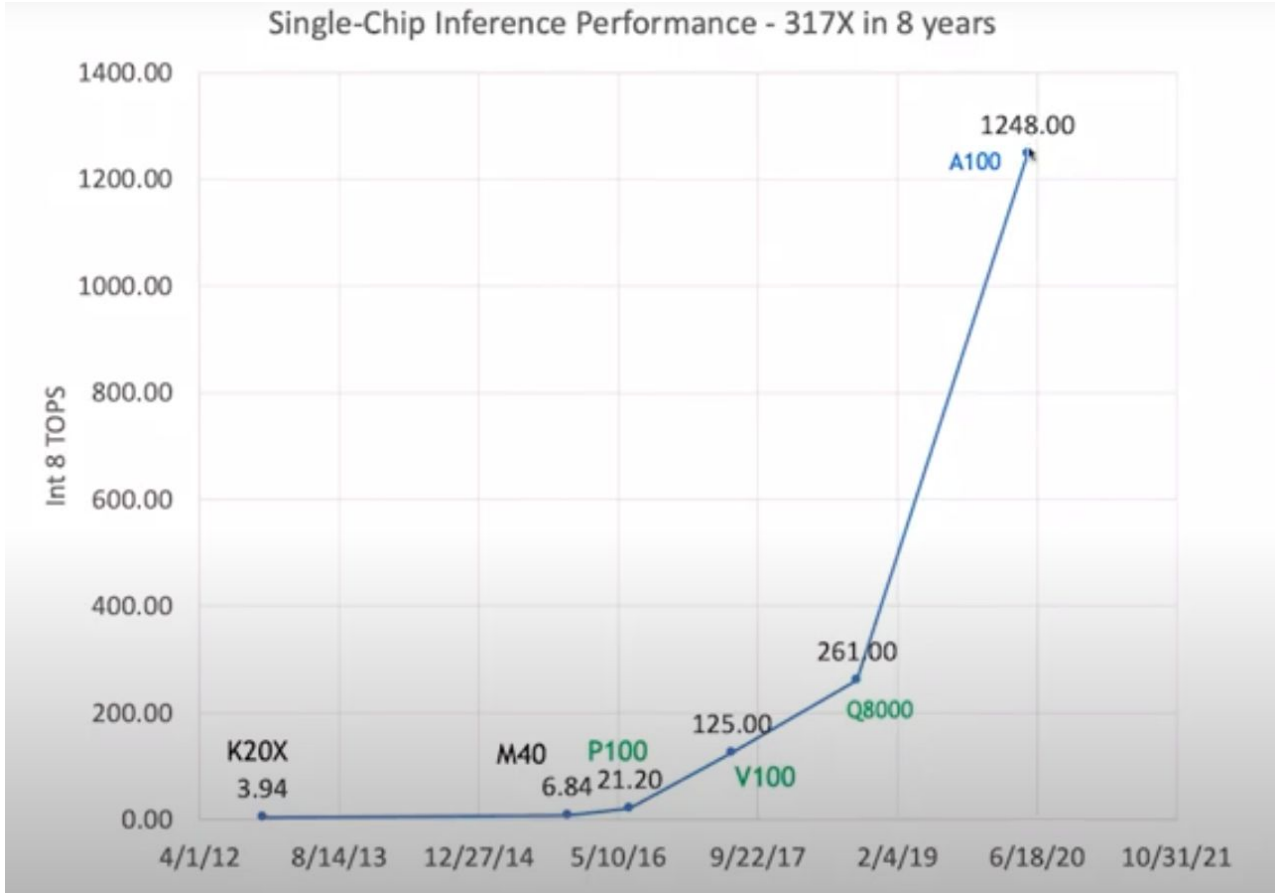


Slides were borrowed from Bill Dally's keynote at ERI Summit.

Hardware is the bottleneck!



Improvements

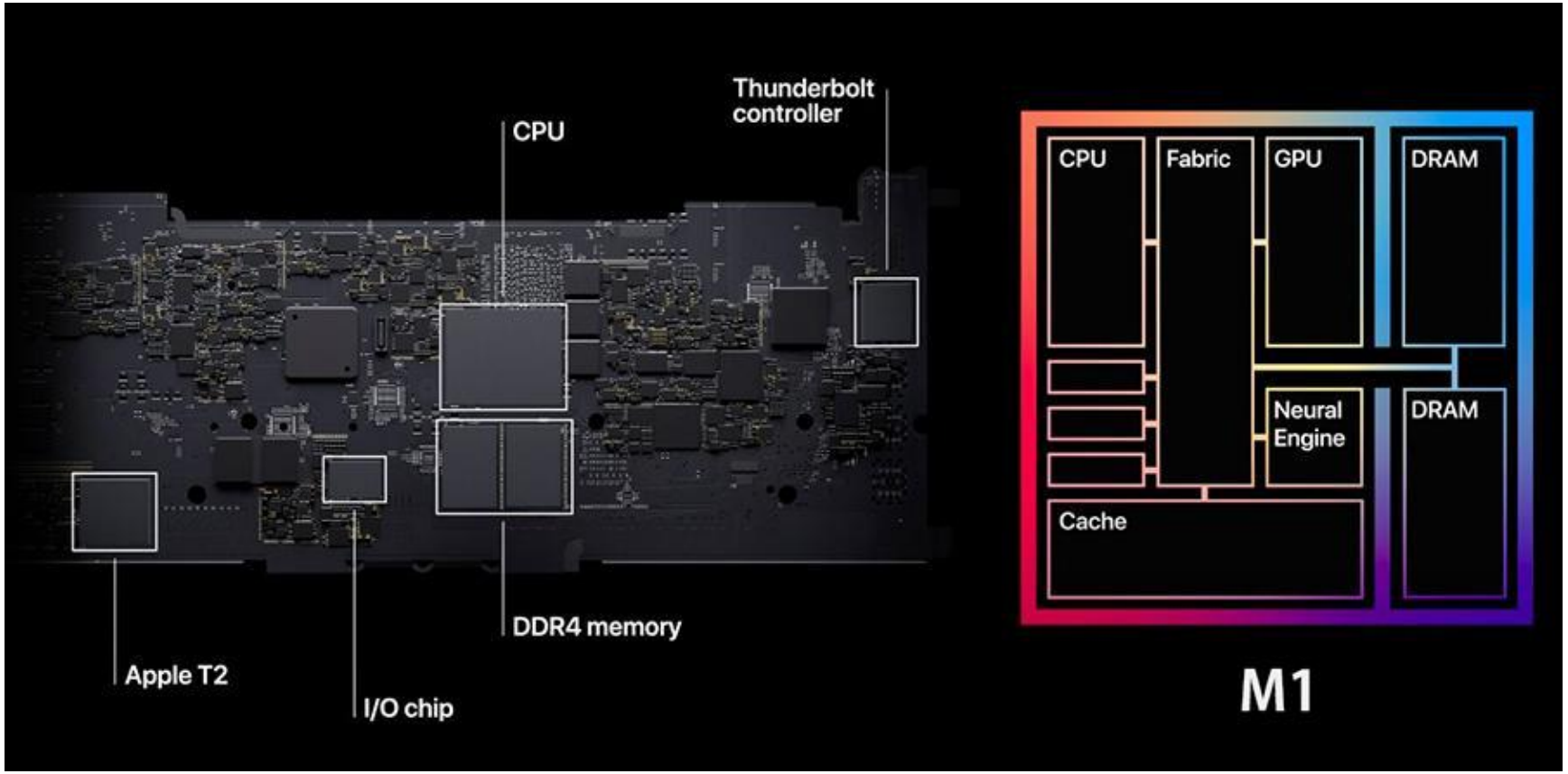


Learning about what causes this significant improvement is *important*, *impactful*, and *useful* for your career.

What we'll learn about hardware

- Hardware-Software interface: ***Instruction-Set-Architecture (ISA)***
- How ISA is implemented, and how different hardware components are organized: ***Microarchitecture***
- Main principles and techniques in designing a Processor: ***pipelining***, ***speculation***, ***parallelism***
- Memory hierarchy design: ***caches and main memory***
- Multicore systems**: *coherency, consistency***
- *Advanced Topics: GPUs*

What we'll be focused on:
Hardware/ Software Interaction



Let's begin …

How do computers work?

How do computers work?

- We can see a computer as a *box* that runs our programs and shows us the results (display, print, etc.)

How do ~~computers work~~ we see computers?

- We can see a computer as a *box* [with multiple layers] that runs our programs and shows us the results (display, print, etc.).

How do ~~computers work~~ we see computers?

- We can see a computer as a *box* [with multiple layers] that runs our programs and shows us the results (display, print, etc.).

  ➔ We define these layers as ***abstraction*** layers.

Using Abstraction

- We see/define a computer as a *box* with multiple layers of **abstraction.**

- Depending on which layer we want to work on, we *abstract away* the irrelevant layers.

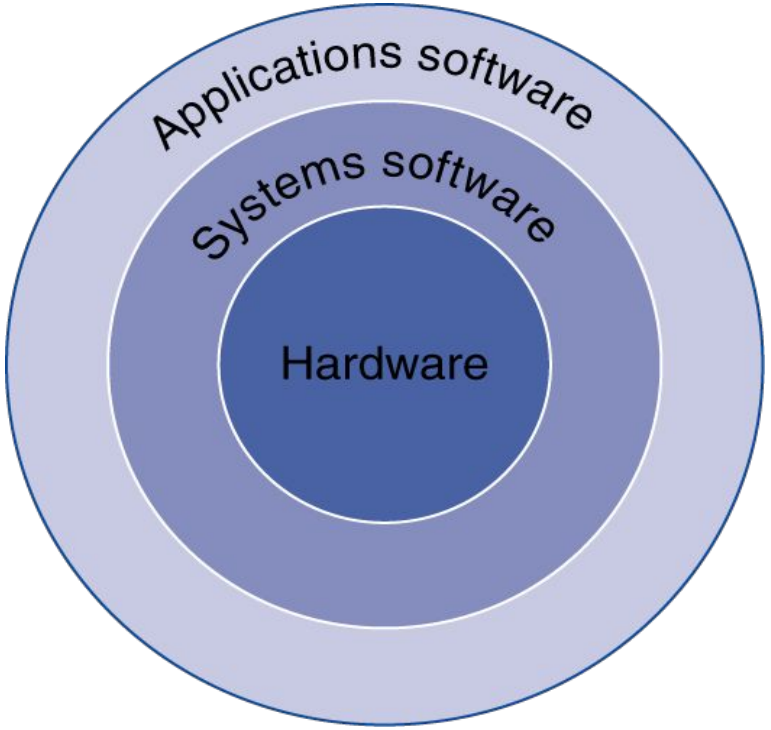
Using Abstraction

- We see/define a computer as a *box* with multiple layers of **abstraction.**

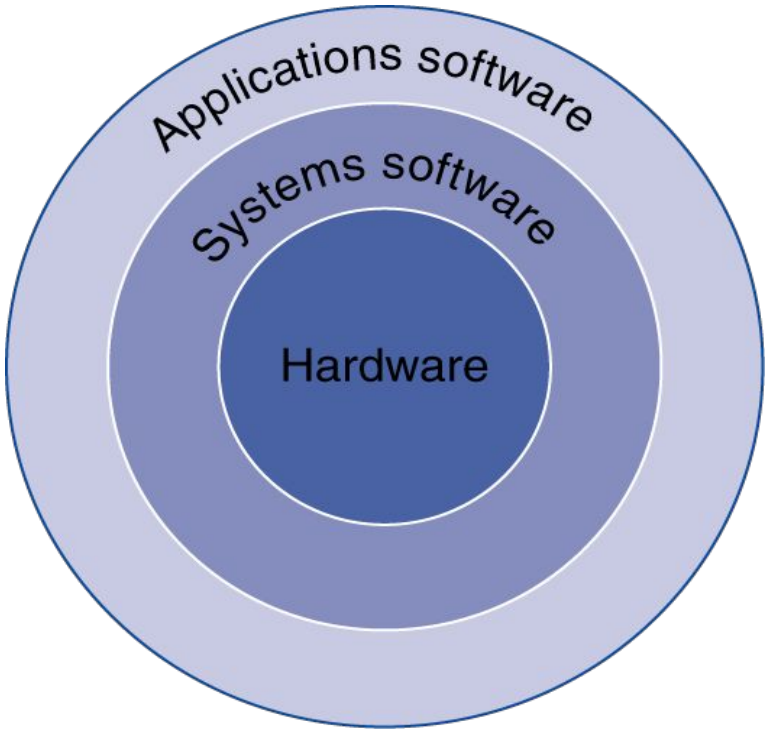
- Depending on which layer we want to work on, we *abstract away* the irrelevant layers.

➔ The ***main benefit*** is that we don't need to know the unnecessary details of the other layers in order to be able to work on our layer.

Computer Abstractions (simplified)



Computer Abstractions (simplified)



- Application software
  - Translation from *algorithm* to code
  - Written in high-level language (e.g., C, JAVA)
- System software
  - ***Compiler***: translates HLL code to machine code
  - ***Operating System***: service code
    - Handling input/output
    - Managing memory and storage
    - Scheduling tasks & sharing resources
- Hardware
  - Processor, memory, I/O controllers

Computer Abstractions

- Application software
  - Translation from *algorithm* to code
- Operating system
  - Managing memory and storage
  - Scheduling tasks & sharing resources
- Hardware
  - Processor, memory, I/O controllers

Watch This:
https://www.youtube.com/watch?v=_y-5nZAbgt4

How do computers work?

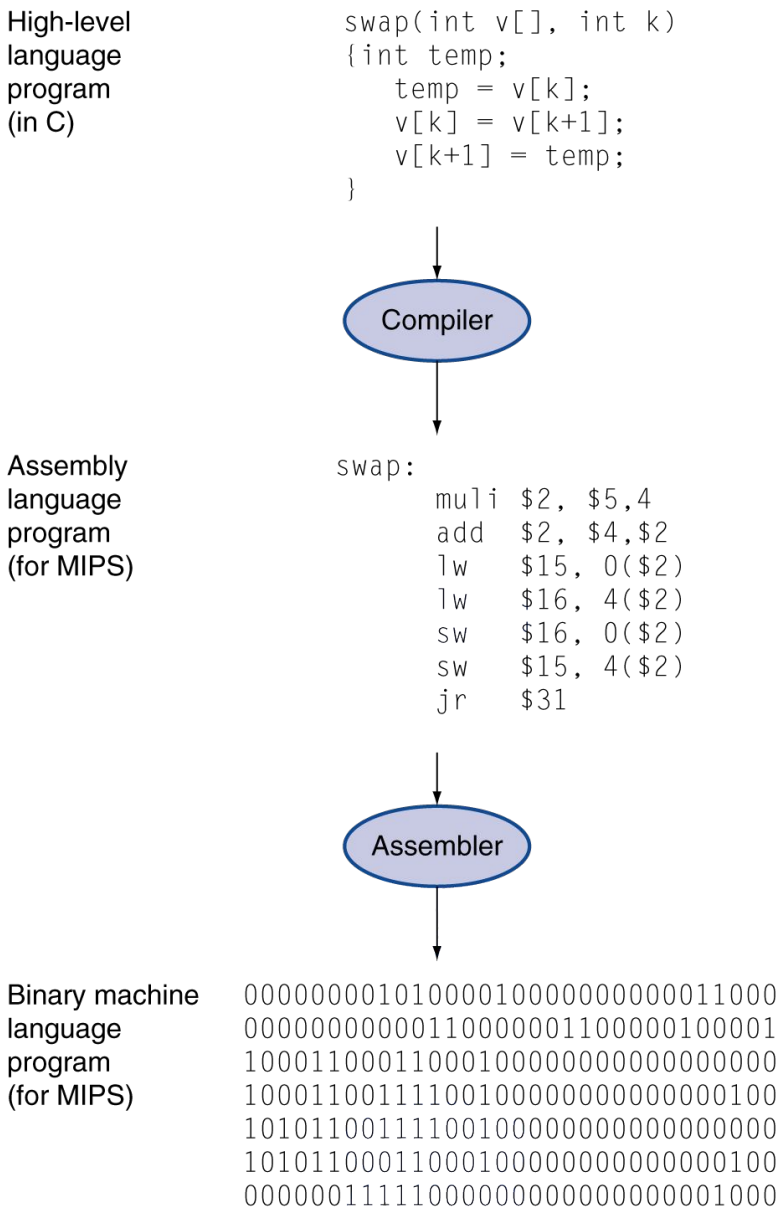
- Theoretical and Historical Points of View
  - Turing machines and history of electronics and computers.

Watch This:

- Theory of Computation: https://www.youtube.com/watch?v=PLVCscCY4xI
- History of Computers: https://www.youtube.com/watch?v=pBiVyEfZVUU

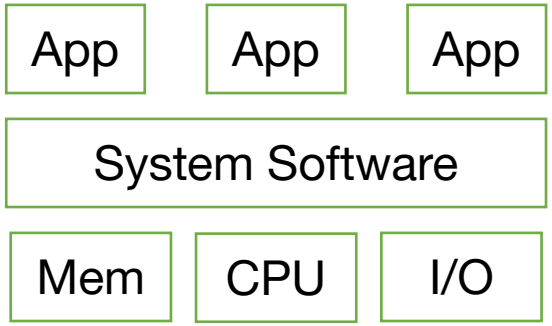
Level of Program Code

- High-level language
  - Level of abstraction closer to problem domain
  - Provides for productivity and portability
- Assembly language
  - Textual representation of instructions
  - Architecture-dependent.
- Hardware representation
  - Binary digits (bits)
  - Encoded instructions and data

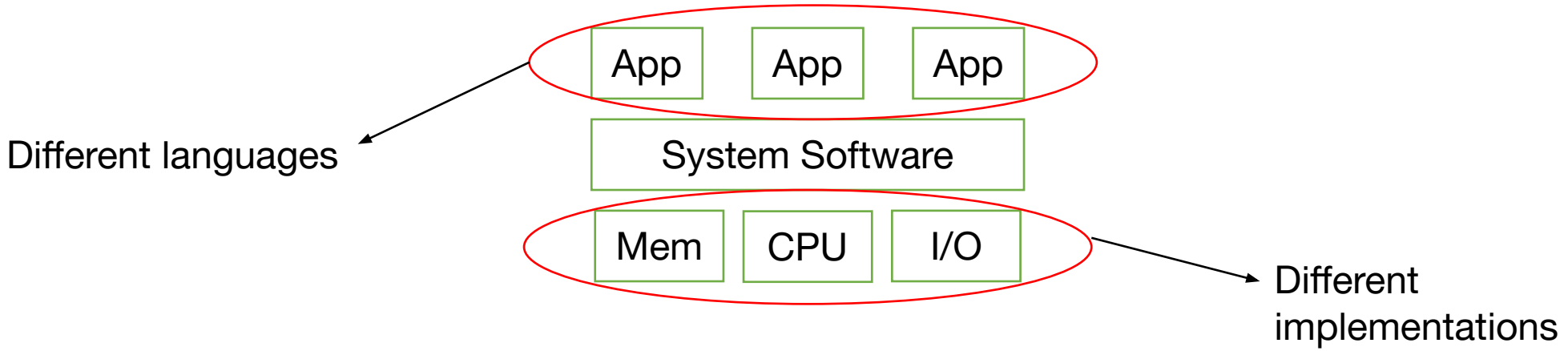


Nader Sehatbakhsh <nsehat@ee.ucla.edu>

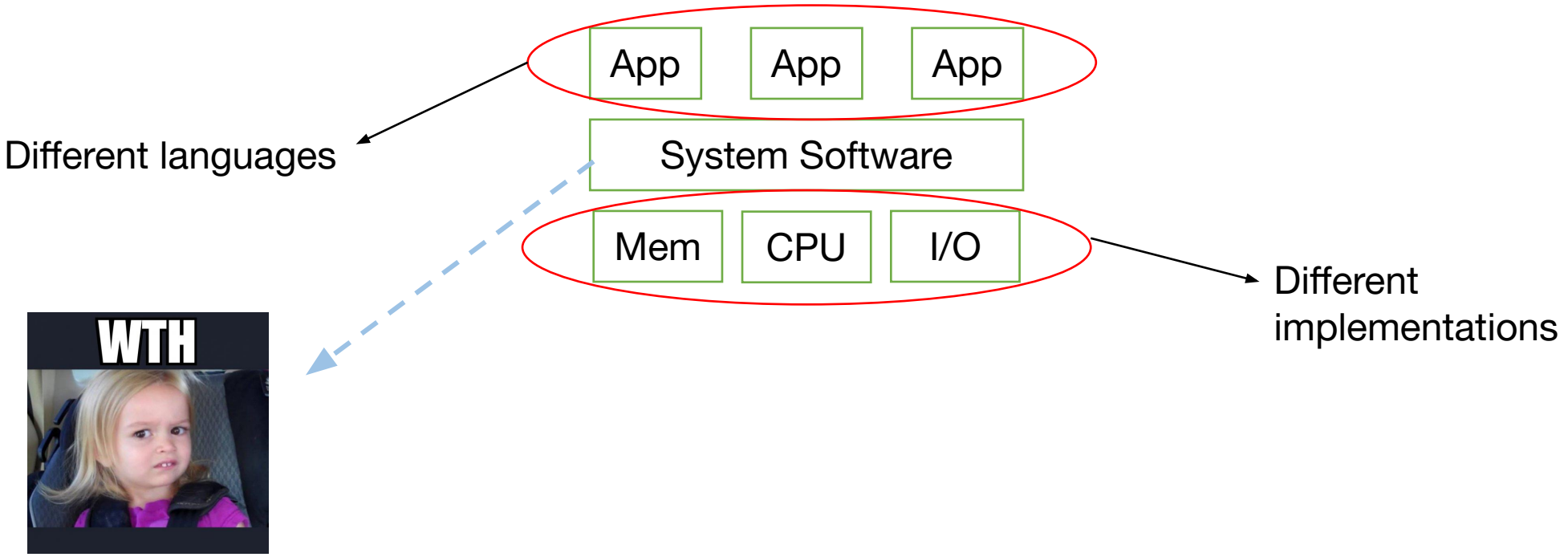
Running an Application



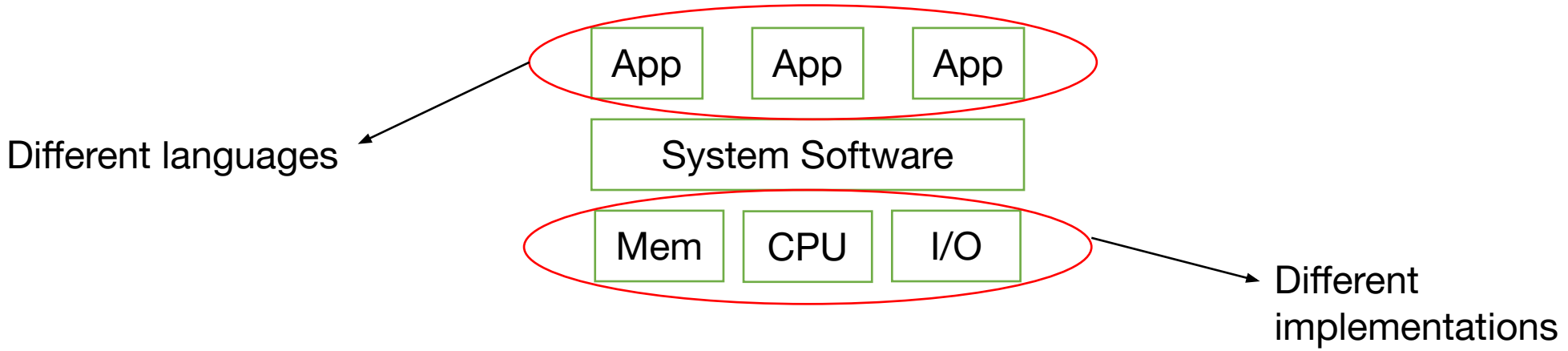
Running an Application



Running an Application



Running an Application



Challenge: How to maintain compatibility?

Challenge: How to maintain compatibility?

SW

*Image was taken from daria.fandom,com

Let's make this ***contract***:
We (SW and Sys. SW) promise to always give you a program with ***only a set of known instructions***, and you (HW) promise to be able to execute those!

Sys. SW

HW

*Image was taken from Shutterstock.com

Instruction Set Architecture

- ISA is the *interface* between hardware and software.

Instruction Set Architecture

- ISA is the *interface* between hardware and software.
- It allows HW and SW to change/evolve ***independently***.

Instruction Set Architecture

- ISA is the *interface* between hardware and software.
- It allows HW and SW to change/evolve ***independently***.
- SW sees:
  - Functional description of hardware:
    1) *Storage locations* (e.g., memory)
    2) *Operations* (e.g., add)

Instruction Set Architecture

- ISA is the *interface* between hardware and software.
- It allows HW and SW to change/evolve ***independently***.
- SW sees:
  - Functional description of hardware:
    1) *Storage locations* (e.g., memory)
    2) *Operations* (e.g., add)
- HW sees:
  - List of instructions and *their order.*

Instruction Set Architecture

- In this course, we will use RISC-V ISA (*why?*)
- Details in next few lectures…

What is the main objective in architecting a computer?

What is the main objective in architecting a computer?

- Make *efficient* computers

*Image was taken from https://www.theunionjournal.com/product-manager-vs-project-manager-the-main-difference/

Ok, what is *efficient* then?

Ok, what is *efficient* then?

- ***Fast*** (duh!)

aka ***performance.***

Ok, what is *efficient* then?

- Performance
- Power Consumption

Ok, what is *efficient* then?

- Performance
- Power Consumption
- Cost

Ok, what is *efficient* then?

- Performance
- Power Consumption
- Cost
- Reliable and Secure

Ok, what is *efficient* then?

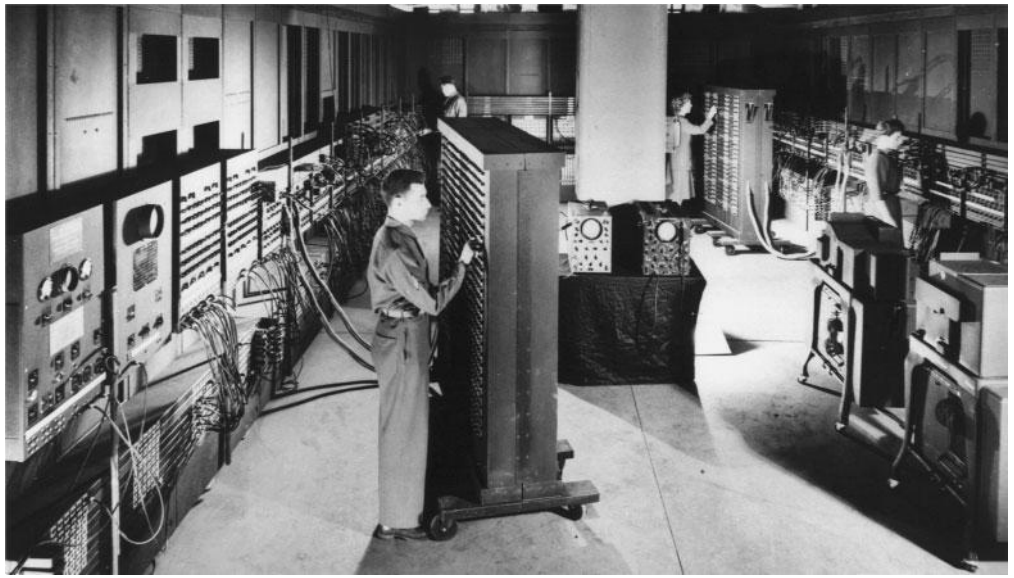
- Performance
- Power Consumption
- Cost
- Reliable and Secure

➔ *Performance* is typically the most important metric, but depending on the application, that might change.

Past, present, (and future)

Past, present, (and future)

We've come a looong way!

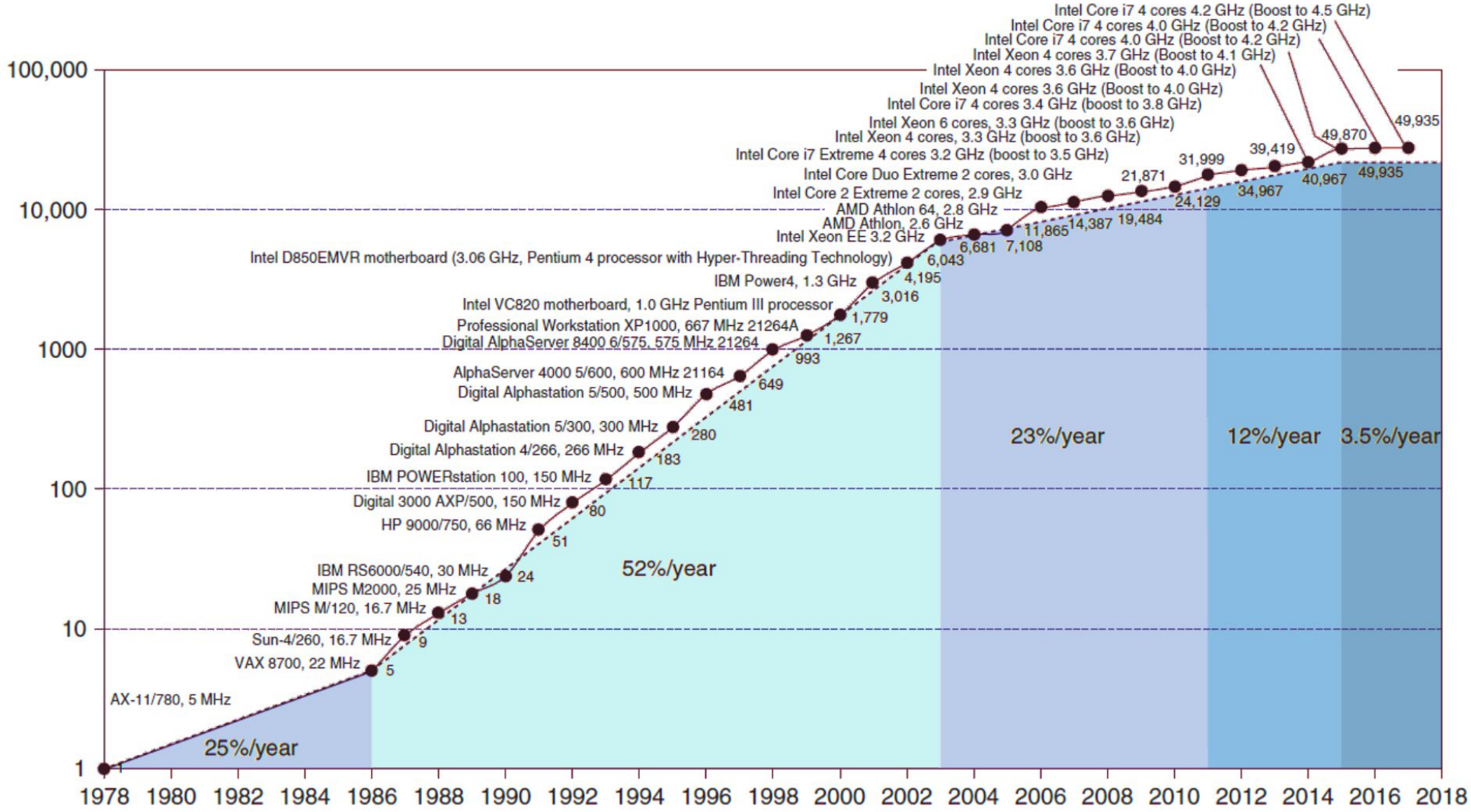


ENIAC, the world's first general-purpose electronic computer ~ ***1945.***

VAX-11, one of the first workstation computers, developed in 1970s.

- Several other personal computers, including Scelbi & Mark-8 Altair, IBM 5100, Radio Shack's TRS-80 (known as the "Trash 80"), the Commodore PET, and first Apple computer, were also introduced in 70s.

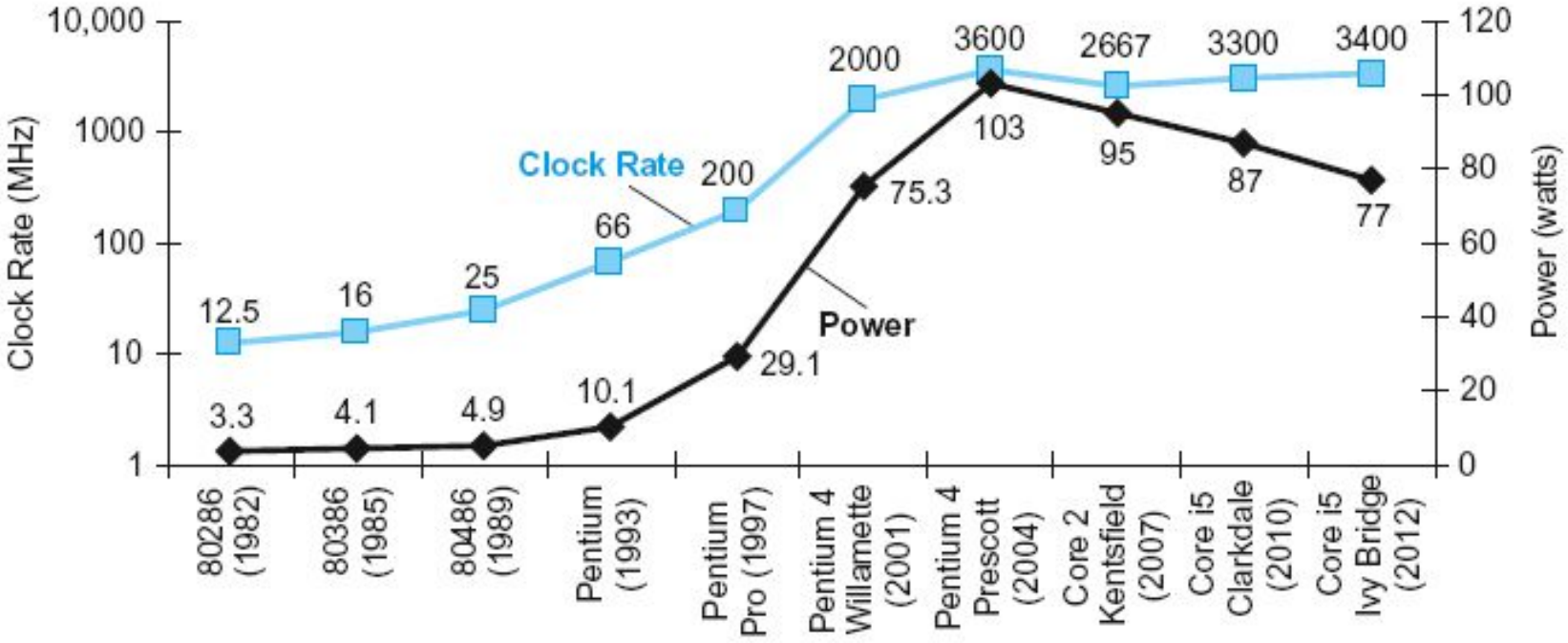
*Images were taken from Hennessy Patterson Book [1].



*Plot was taken from Hennessy Patterson Book [1].

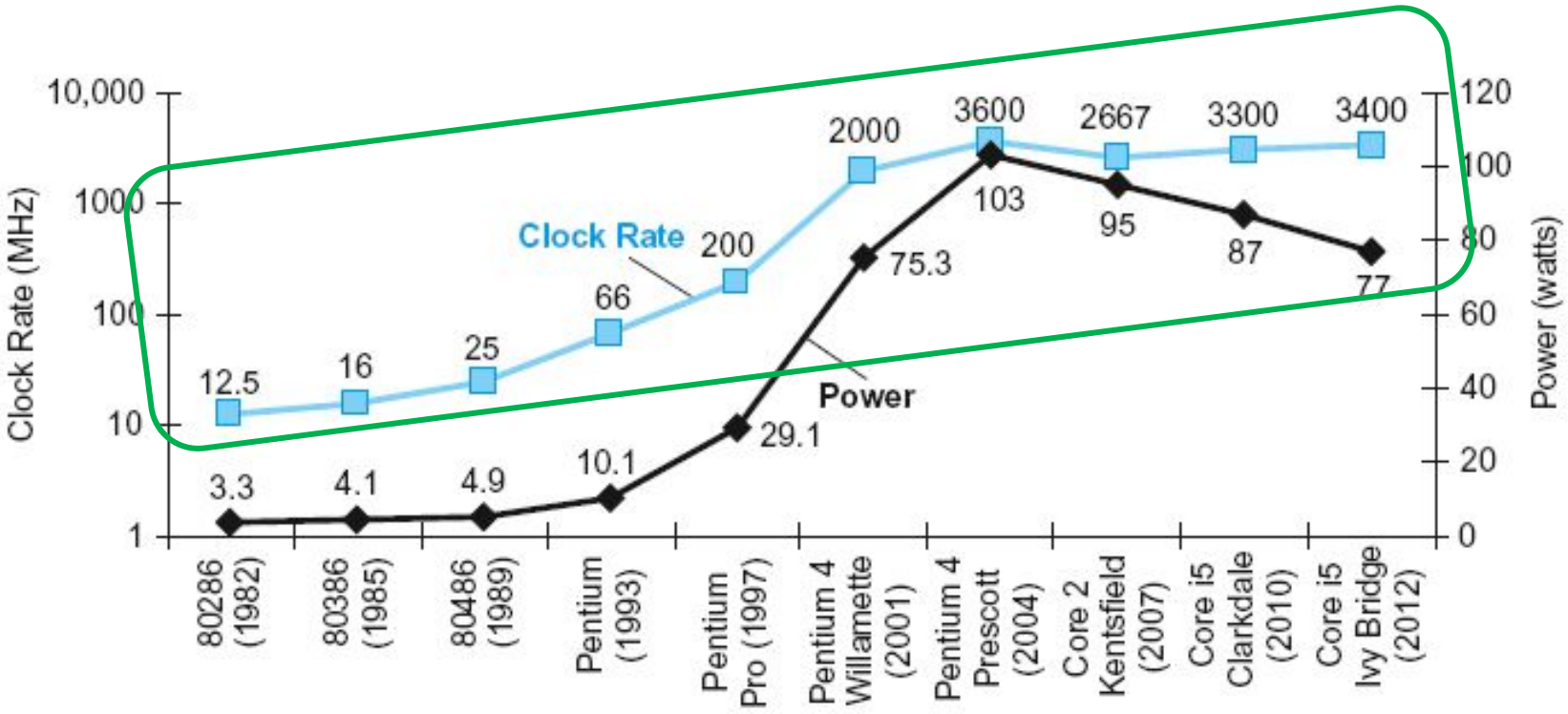
This chart plots performance relative to the VAX 11/780 on average.

A modern Intel Core i7 processor has about 50k higher performance than VAX.



*Plot was taken from Hennessy Patterson Book [1].

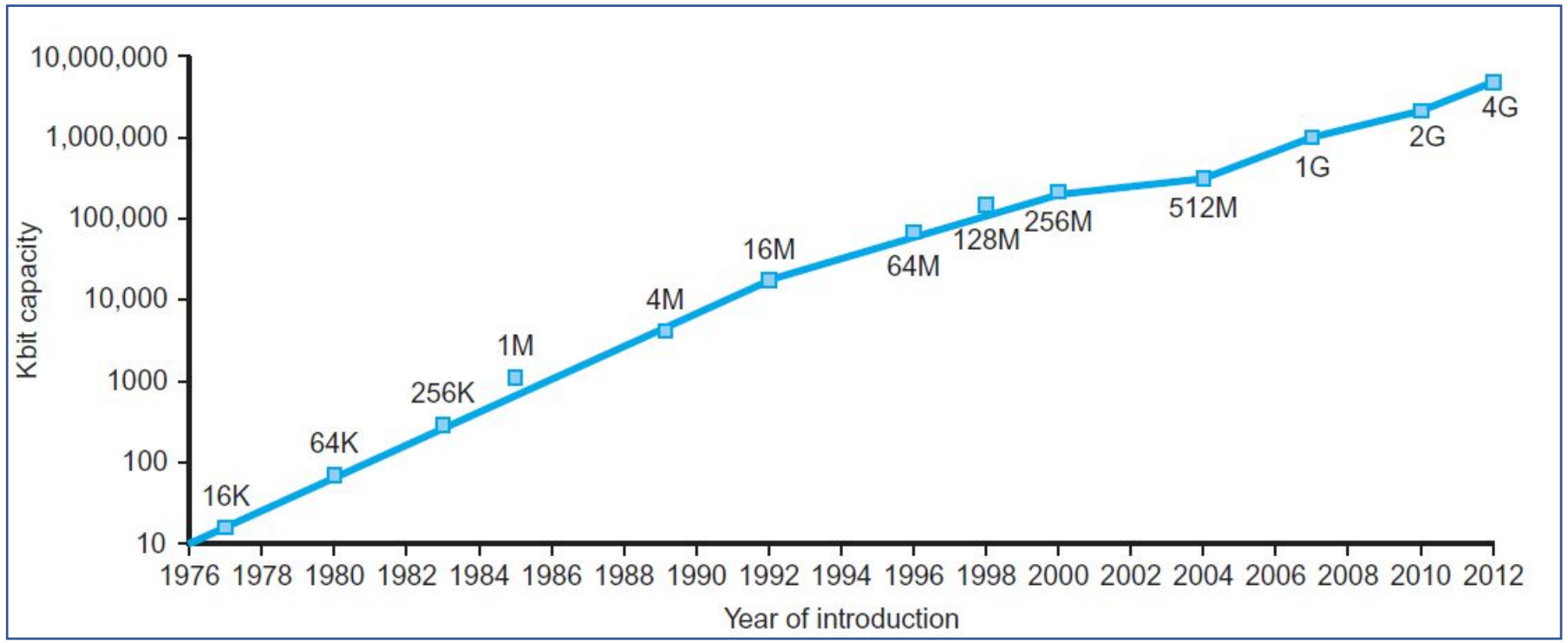
This chart plots the clock rate and power consumption of Intel microprocessors over time.



*Plot was taken from Hennessy Patterson Book [1].

This chart plots the clock rate and power consumption of Intel microprocessors over time.

A modern Intel Core i5 processor has about 300 times faster clock rate than the first version.



*Plot was taken from Hennessy Patterson Book [1].

Memory (DRAM) capacity over time.

A modern DRAM has about 2^{LZ} times more space than the first version of DRAM.

We've come a looong way!

Year	Name	Size (cu. ft.)	Power (watts)	Performance (adds/sec)	Memory (KB)	Price	Price/ performance vs. UNIVAC	Adjusted price (2007 $)	Adjusted price/ performance vs. UNIVAC
1951	UNIVAC I	1,000	125,000	2,000	48	$1,000,000	1	$7,670,724	1
1964	IBM S/360 model 50	60	10,000	500,000	64	$1,000,000	263	$6,018,798	319
1965	PDP-8	8	500	330,000	4	$16,000	10,855	$94,685	13,367
1976	Cray-1	58	60,000	166,000,000	32,000	$4,000,000	21,842	$13,509,798	47,127
1981	IBM PC	1	150	240,000	256	$3,000	42,105	$6,859	134,208
1991	HP 9000/ model 750	2	500	50,000,000	16,384	$7,400	3,556,188	$11,807	16,241,889
1996	Intel PPro PC (200 MHz)	2	500	400,000,000	16,384	$4,400	47,846,890	$6,211	247,021,234
2003	Intel Pentium 4 PC (3.0 GHz)	2	500	6,000,000,000	262,144	$1,600	1,875,000,000	$2,009	11,451,750,000
2007	AMD Barcelona PC (2.5 GHz)	2	250	20,000,000,000	2,097,152	$800	12,500,000,000	$800	95,884,051,042

*Table was taken from Hennessy Patterson Book [1].

We've come a looong way!

Year	Name	Size (cu. ft.)	Power (watts)	Performance (adds/sec)	Memory (KB)	Price	Price/ performance vs. UNIVAC	Adjusted price (2007 $)	Adjusted price/ performance vs. UNIVAC
1951	UNIVAC I	1,000	125,000	2,000	48	$1,000,000	1	$7,670,724	1
1964	IBM S/360 model 50	60	10,000	500,000	64	$1,000,000	263	$6,018,798	319
1965	PDP-8	8	500	330,000	4	$16,000	10,855	$94,685	13,367
1976	Cray-1	58	60,000	166,000,000	32,000	$4,000,000	21,842	$13,509,798	47,127
1981	IBM PC	1	150	240,000	256	$3,000	42,105	$6,859	134,208
1991	HP 9000/ model 750	2	500	50,000,000	16,384	$7,400	3,556,188	$11,807	16,241,889
1996	Intel PPro PC (200 MHz)	2	500	400,000,000	16,384	$4,400	47,846,890	$6,211	247,021,234
2003	Intel Pentium 4 PC (3.0 GHz)	2	500	6,000,000,000	262,144	$1,600	1,875,000,000	$2,009	11,451,750,000
2007	AMD Barcelona PC (2.5 GHz)	2	250	20,000,000,000	2,097,152	$800	12,500,000,000	$800	95,884,051,042

We've made faster, smaller, AND cheaper computers!

*Table was taken from Hennessy Patterson Book [1].

So, where these improvements are coming from?

So, where these improvements are coming from?

- *The short answer:*

➔ A *combination* of new technologies and innovative techniques.

So, where these improvements are coming from?

1. Advancements in microelectronics and fabrication technologies.

Year	Technology	Relative performance/cost
1951	Vacuum tube	1
1965	Transistor	35
1975	Integrated circuit (IC)	900
1995	Very large scale IC (VLSI)	2,400,000
2013	Ultra large scale IC	250,000,000,000

➔ *Better performance per cost, better power efficiency and* ***MORE TRANSISTORS***

Moore's Law

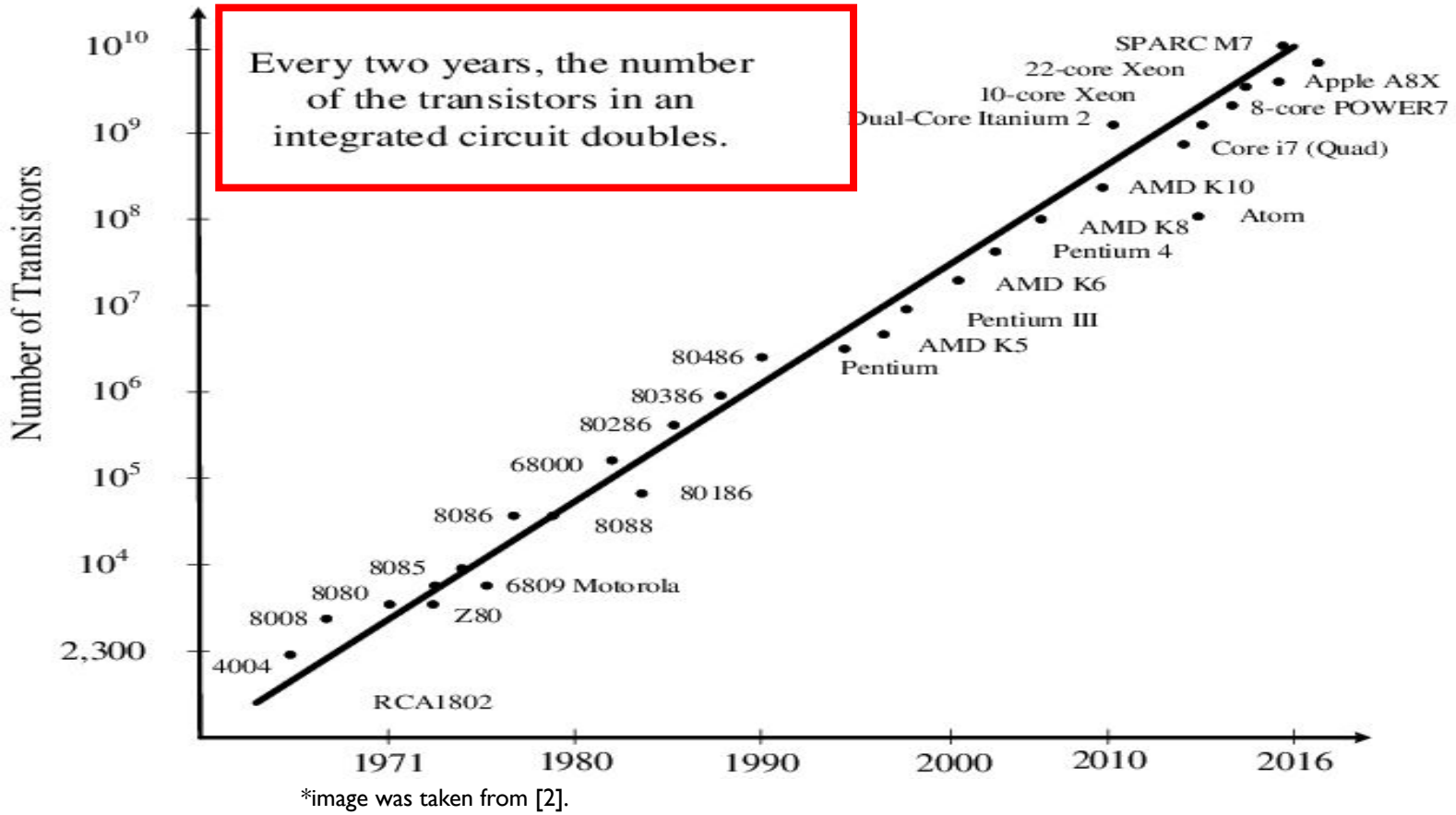


*image was taken from [2].

Nader Sehatbakhsh <nsehat@ee.ucla.edu>

Moore's Law

"The number of transistors in an IC doubles every two years."

- Moore's law, in essence, is less about the performance and is more about the economy.
- It suggests that the *cost per transistor* reduces every year (i.e., faster chips with same price).

*image was taken from [2].

Moore's Law Impacts

★ *What happens to other metrics when the number of transistors doubles?*

★ *How does the scaling work?*

Dennard's Scaling Law

- According to Moore's law, the number of transistors on a chip doubles every two years, thus each transistor's area is reduced by 50% (or every dimension by 0.7x).

Dennard's Scaling Law Impacts

- Voltage is reduced by -30% (0.7x) to keep the electric field constant. (V = EL)
- L is reduced, thus delays are reduced by -30% (x= Vt)
- Frequency is increased by +40% (f = 1/t)
- Capacitance is reduced by -30% (C = kA/L)

Scaling Power and Energy

$$P = CV^2 f$$

Scaling Power and Energy

$$P = CV^2 f$$

- Power consumption per transistor is decreased by -50%.

➔ What about the entire chip?

Scaling Power and Energy

$$P = CV^2 f$$

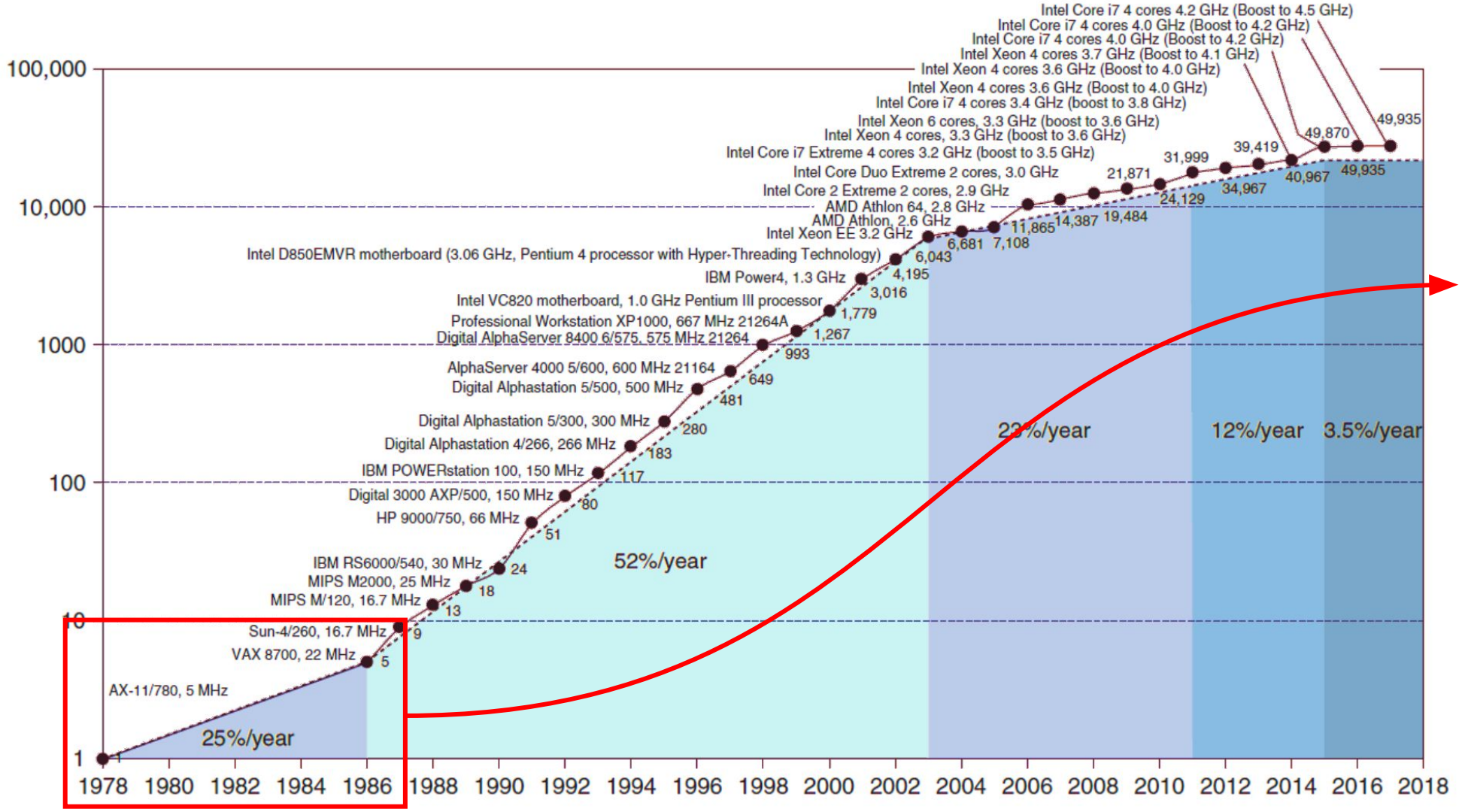
- Power consumption per transistor is decreased by -50%.

➔ What about the entire chip?
- it stays the same!

Recap

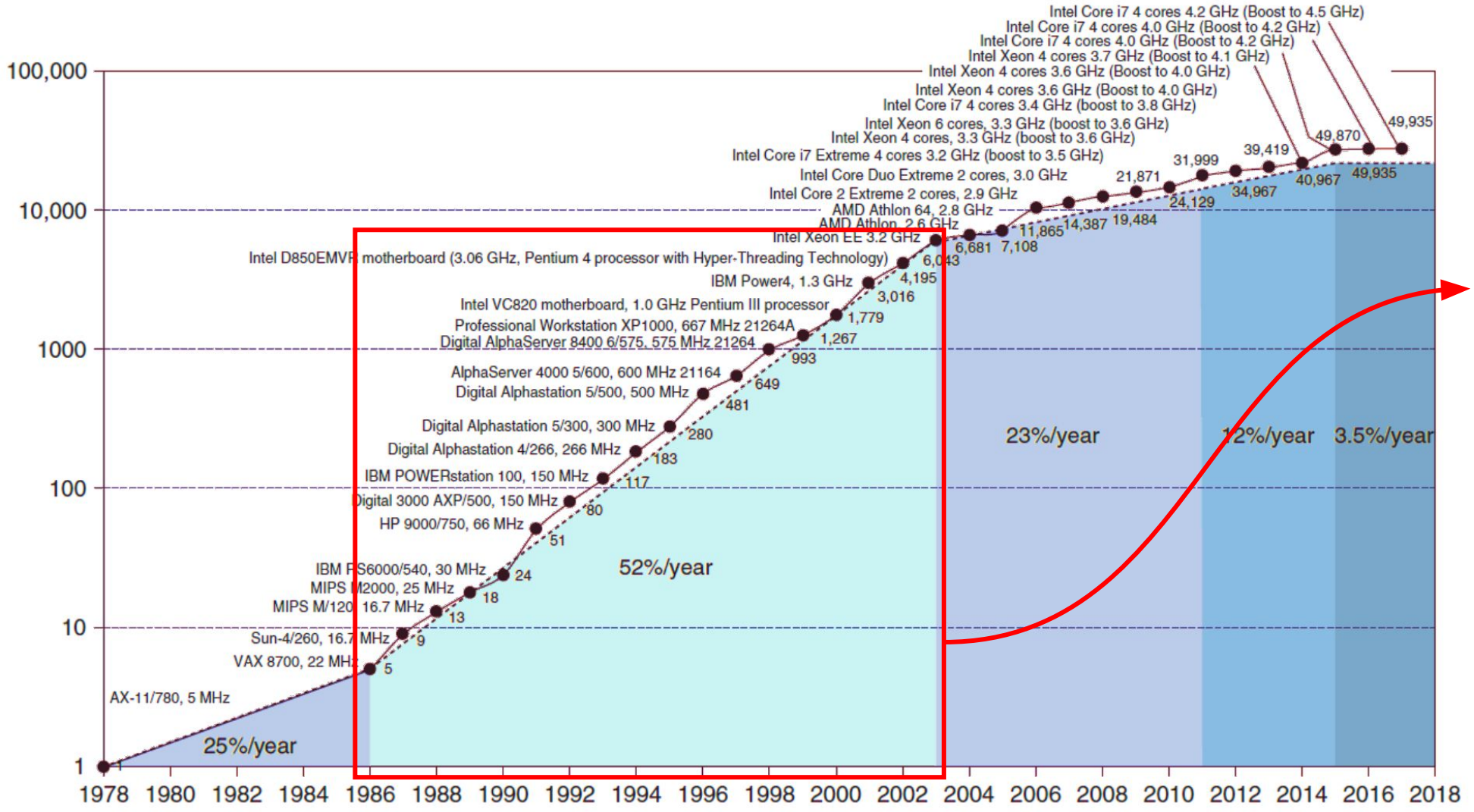
So, where these improvements are coming from?

1. Advancements in microelectronics and fabrication technologies.



Device technology advancements.

*Plot was taken from [1].



The increase in growth to about 52% since then is attributable to more advanced architectural and organizational ideas.

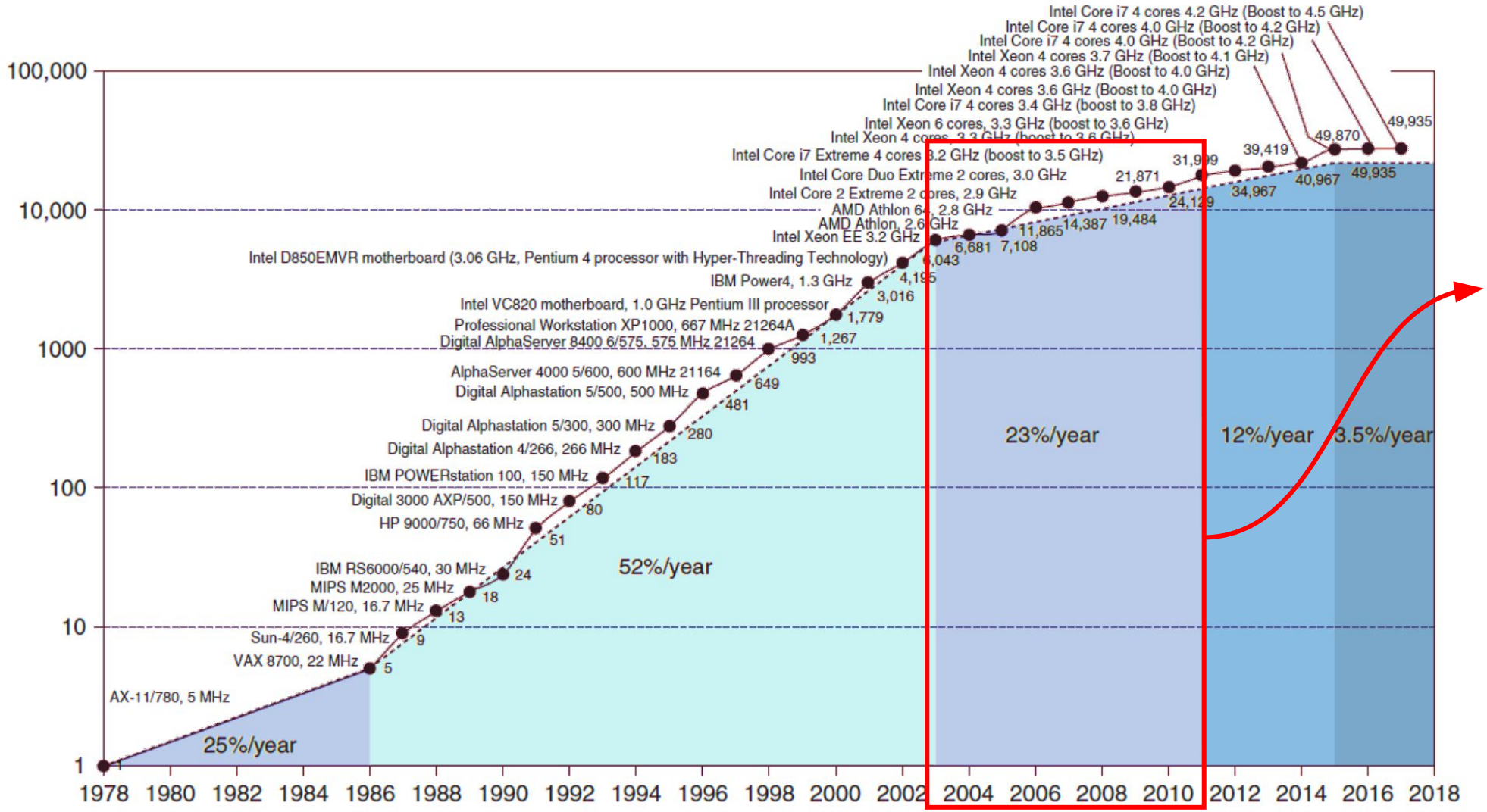
*Plot was taken from [1].

So, where these improvements are coming from?

1. Advancements in microelectronics and fabrication technologies.

2. Advancements in architectural techniques
   - *What this course is about! We will learn about these!*
   - This lead to an improvement by a factor of 25 vs. if we had only relied on 1.

But wait, there is more…



End of Dennard's scaling.
Hitting the ***power wall.***

*Plot was taken from [1].

Power Wall!

- Up to 2005, manufacturers could double the processor performance while keeping the power constant.

Power Wall!

- Up to 2005, manufacturers could double the processor performance while keeping the power constant.

- Since 2005, due to smaller transistor sizes, static power leakage becomes so dominant that the power consumption did not stay constant.

Power Wall!

- Up to 2005, manufacturers could double the processor performance while keeping the power constant.
- Since 2005, due to smaller transistor sizes, static power leakage becomes so dominant that the power consumption did not stay constant.
- This phenomenon generally is known as *hitting the power wall.*

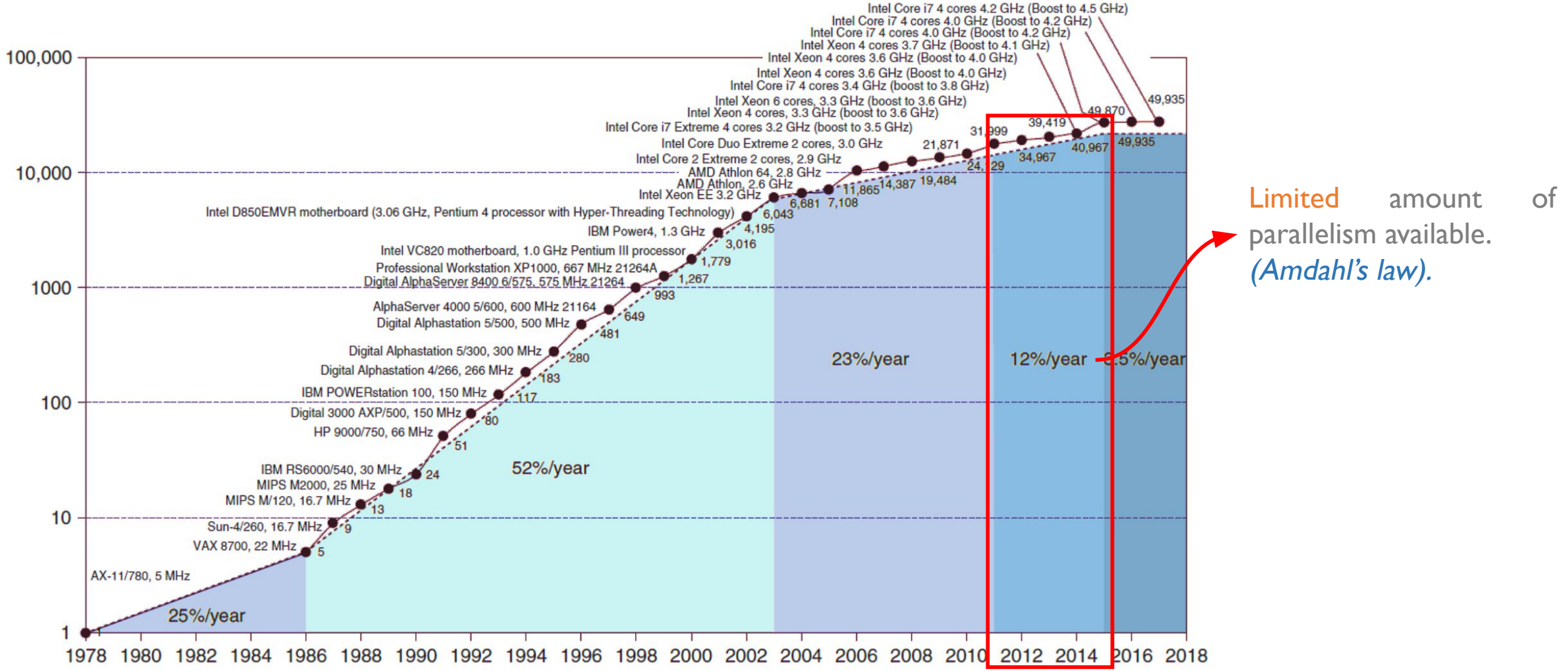
What happened after 2005?

What happened after 2005?

Multi-Core Era

- Without faster processors, speed up can still be achieved using *thread-level*, *task-level*, and *program-level* parallelism.

➔ *We will briefly talk about this in the second half of the course.*



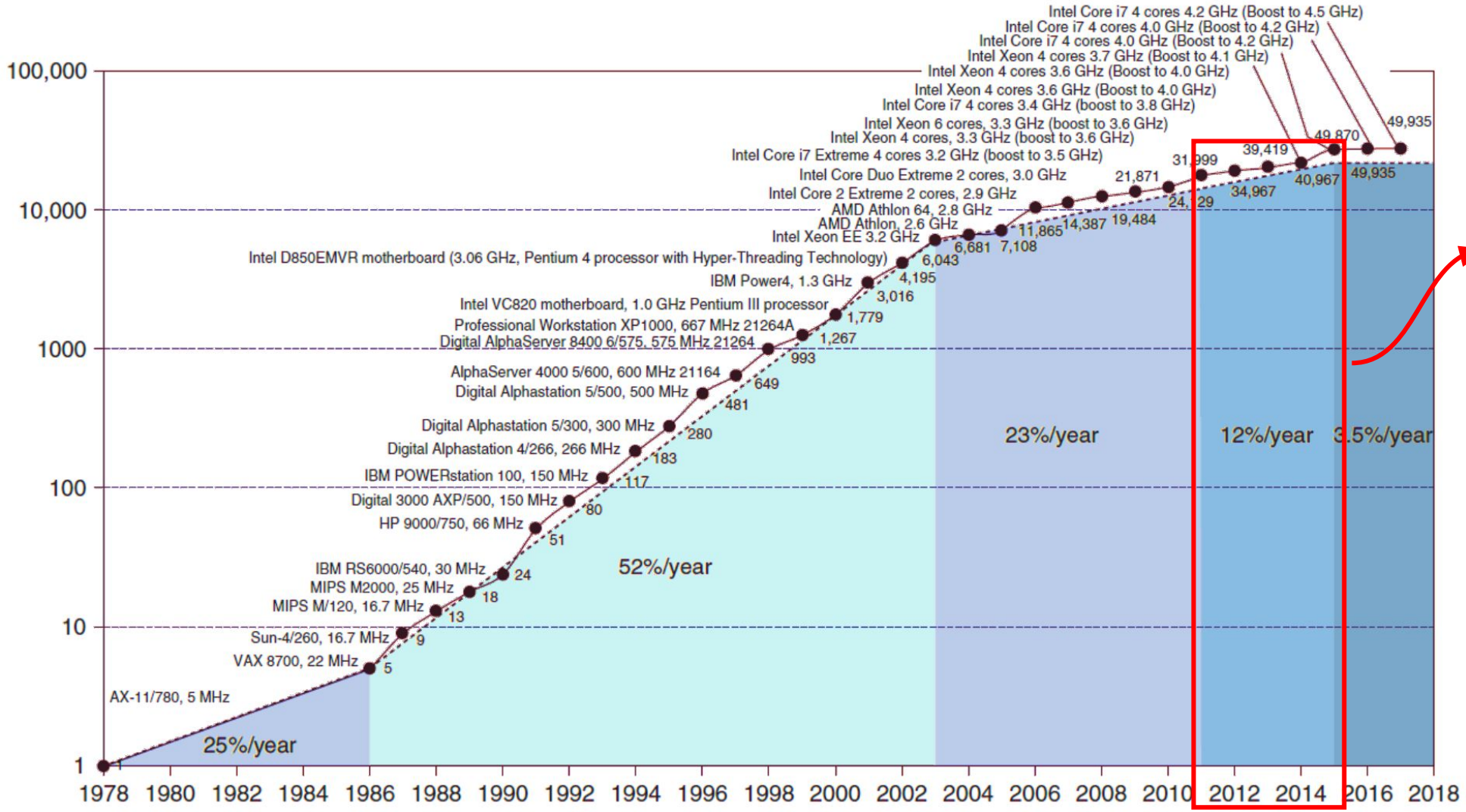
Limited amount of parallelism available. *(Amdahl's law).*

Amdahl's Law

– Performance improvement (speedup) is limited by the part you cannot improve (called sequential part).

$$speed\ up = \frac{1}{(1-p) + \frac{p}{s}}$$

- P is the part that can be improved (e.g., parallelized),
- S is the factor for improvement (e.g., more cores for parallelization).



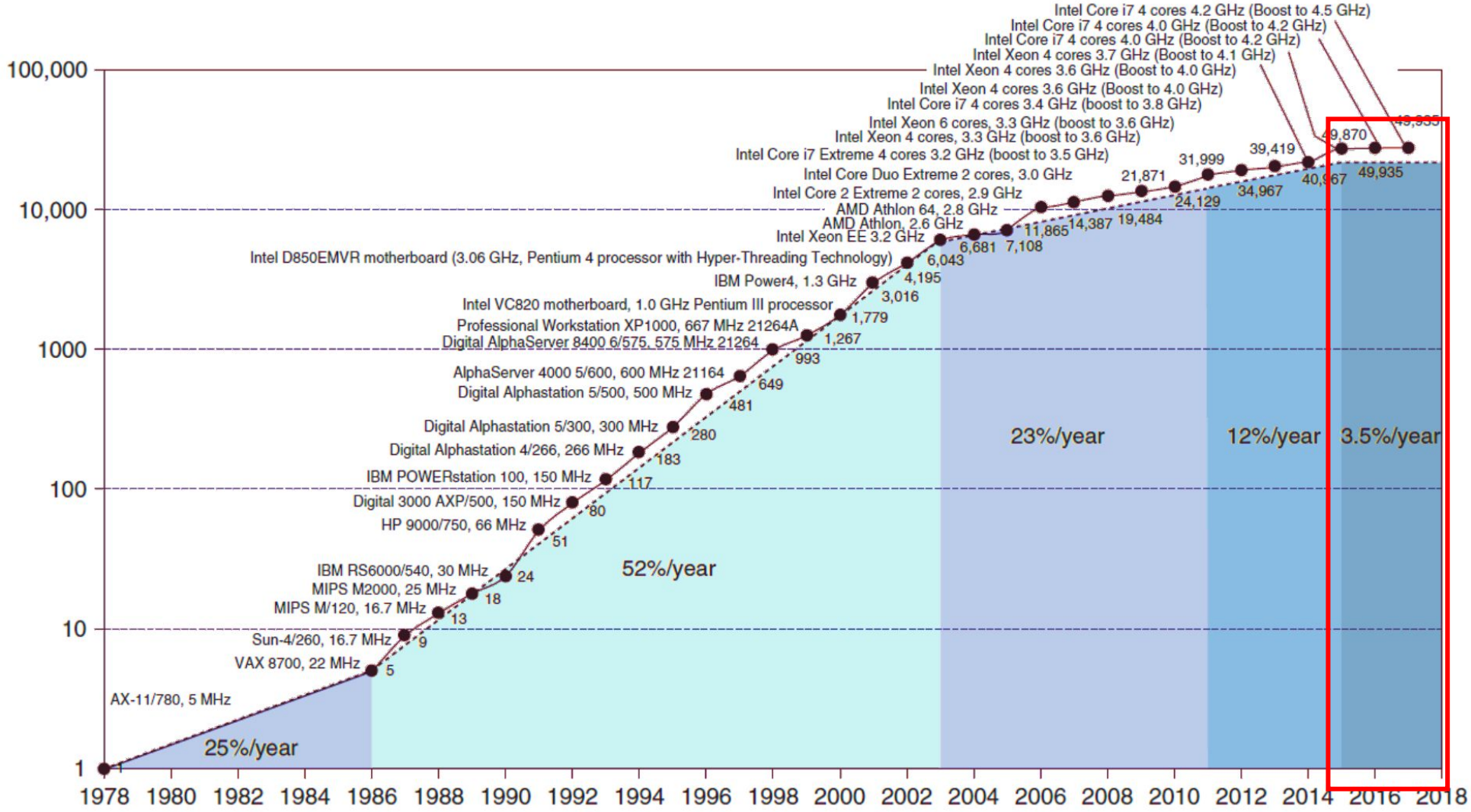
Limited amount of parallelism available. *(Amdahl's law).*

Multi-Core Era – The ***END***

Dark Silicon. Due to thermal management limitations, some parts of a chip cannot be turned on.

End of Moore's Law.

Multi-core


100,000
10,000
1000
100
10
1
1978 1980 1982 1984 1986 1988 1990 1992 1994 1996 1998 2000 2002 2004 2006 2008 2010 2012 2014 2016 2018
Intel Core i7 4 cores 4.2 GHz (Boost to 4.5 GHz)
Intel Core i7 4 cores 4.0 GHz (Boost to 4.2 GHz)
Intel Core i7 4 cores 4.0 GHz (Boost to 4.2 GHz)
Intel Xeon 4 cores 3.7 GHz (Boost to 4.1 GHz)
Intel Xeon 4 cores 3.6 GHz (Boost to 4.0 GHz)
Intel Xeon 4 cores 3.6 GHz (Boost to 4.0 GHz)
Intel Core i7 4 cores 3.4 GHz (boost to 3.8 GHz)
Intel Xeon 6 cores, 3.3 GHz (boost to 3.6 GHz)
Intel Xeon 4 cores, 3.3 GHz (boost to 3.6 GHz)
Intel Core i7 Extreme 4 cores 3.2 GHz (boost to 3.5 GHz)
Intel Core Duo Extreme 2 cores, 3.0 GHz
Intel Core 2 Extreme 2 cores, 2.9 GHz
AMD Athlon 64, 2.8 GHz
AMD Athlon, 2.6 GHz
Intel Xeon EE 3.2 GHz
Intel D850EMVR motherboard (3.06 GHz, Pentium 4 processor with Hyper-Threading Technology)
IBM Power4, 1.3 GHz
Intel VC820 motherboard, 1.0 GHz Pentium III processor
Professional Workstation XP1000, 667 MHz 21264A
Digital AlphaServer 8400 6/575, 575 MHz 21264
AlphaServer 4000 5/600, 600 MHz 21164
Digital Alphastation 5/500, 500 MHz
Digital Alphastation 5/300, 300 MHz
Digital Alphastation 4/266, 266 MHz
IBM POWERstation 100, 150 MHz
Digital 3000 AXP/500, 150 MHz
HP 9000/750, 66 MHz
IBM RS6000/540, 30 MHz
MIPS M2000, 25 MHz
MIPS M/120, 16.7 MHz
Sun-4/260, 16.7 MHz
VAX 8700, 22 MHz
AX-11/780, 5 MHz
49,870
39,419
31,999
21,871
19,484
14,387
11,865
7,108
6,681
6,043
4,195
3,016
1,779
1,267
993
649
481
280
183
117
80
51
24
18
13
9
5
1
49,935
40,967
34,967
24,129
25%/year
52%/year
23%/year
12%/year
3.5%/year

Is this *it*?

Takeaways

- Historically, performance scaled at a rate of 2x per 24 months.
- Microarchitecture tricks further improved performance (by a factor of 25).
- Scaling stopped due to the "Power Wall" in 2005.
- This caused the era of "Multicores". But it is ending.
- New technologies and techniques are being used to continue the scaling.

→ We need cross-stack approaches!

Summary

End of Presentation

Acknowledgement

- This course is partly inspired by the following courses created by my colleagues:
  - CS152, Krste Asanovic (UCB)
  - 18-447, James C. Hoe (CMU)
  - CSE141, Steven Swanson (UCSD)
  - CIS 501, Joe Devietti (Upenn)
  - CS4290, Tom Conte (Goergia Tech)
  - 252-0028-00L, Onur Mutlu (ETH)