دانشکده مهندسی کامپیوتر

درس معماری کامپیوتر

الهام چشمی خانی

Introduction

معرفی

- **الهام چشمی‌خانی**

✓ فارغ التحصیل دکترای مهندسی کامپیوتر از دانشگاه شریف
✓ زمینه‌های کاری: معماری کامپیوتر، حافظه‌های نوظهور، حافظه‌های نهان، پردازش در حافظه‌ها، سیستم‌های ذخیره‌سازی، قابلیت اطمینان، طراحی سیستم‌های مطمئن، طراحی شتاب‌دهنده‌ها، طراحی توامان سخت‌افزای و نرم‌افزاری
✓ مرکز HPC دانشگاه شریف، شرکت Huawei
✓ Google Scholar -> Elham Cheshmikhani
✓ cheshmikhani_e@aut.ac.ir

- **تیم دستیاران:**
  - سردستیار: آقای فرشاد احدی‌نیا

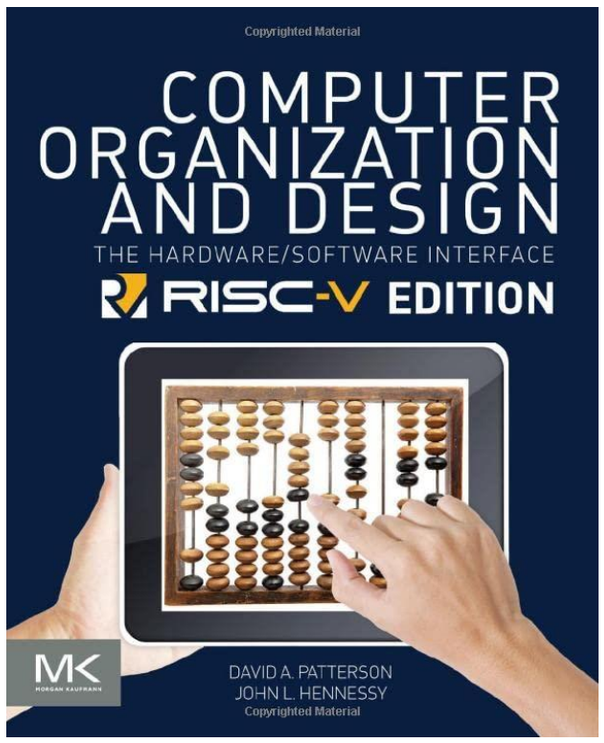
✓ farshad.ahn@aut.ac.ir

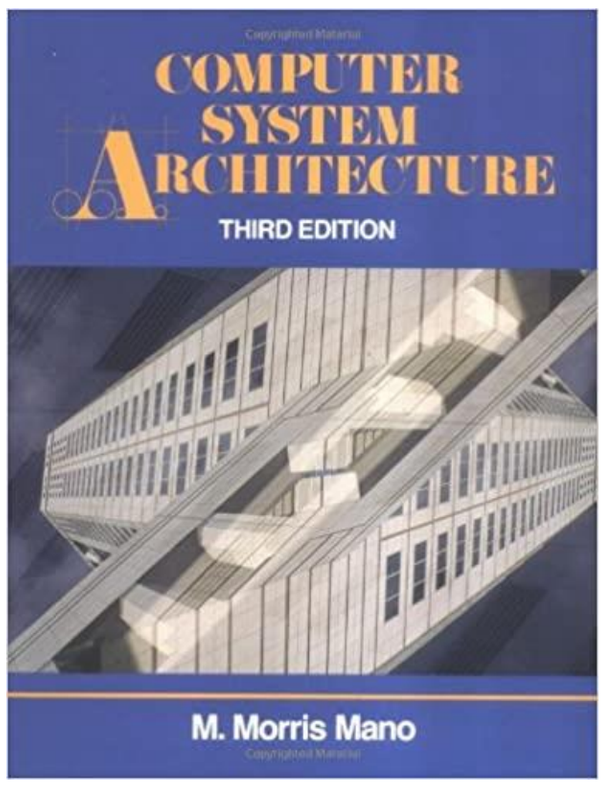
  - دستیاران مسئول: آقایان بردیا اردکانیان و محمدرضا صادقیان

✓ b.ardakanian@aut.ac.ir, sadeghian.m79@gmail.com

  - سایر دستیاران: خانم‌ها ریحانه آهنی و صبا رمضانی و آقای آراد فیروزکوهی

منابع





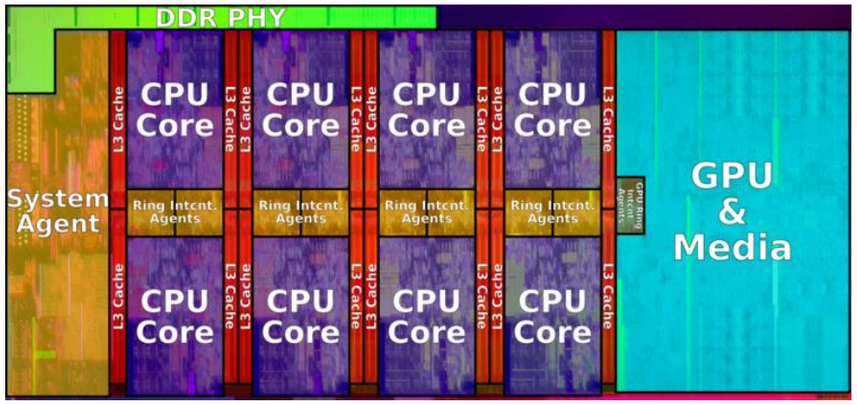
- **مراجع اصلی:**
  - David A Patterson and John L. Hennessy, “Computer Organization and Design RISC-V Edition: The Hardware Software Interface,” Morgan Kaufmann Series
  - M. Morris Mano, “Computer System Architecture,” Pearson Education (US)
- **مراجع دیگر:**
  - Linda Null, “The essentials of computer organization and architecture,” Jones & Bartlett Publishers
  - David Harris, “Digital Design and Computer Architecture,” Elsevier
  - M. Mazidi, S. Naimi, “ARM Assembly Language Programming & Architecture,” MicroDigitalEd

نمره‌دهی

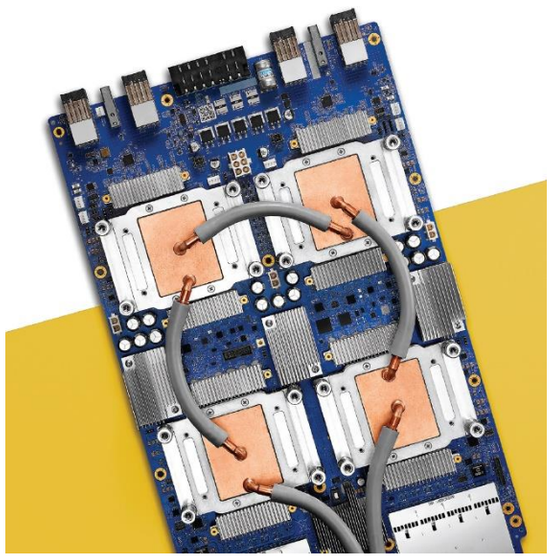
- **تمرین:** (شرایط: ۲۴ ساعت-کپی)
  - ٪۳۰
- **میان‌ترم:**
  - ٪۲۰
- **پایان‌ترم:**
  - ٪۲۵
- **پروژه:**
  - ٪۱۵
- **کوییز:**
  - ٪۱۰

هدف درس

- طراحی سیستم اساساً بهتر/بهینه

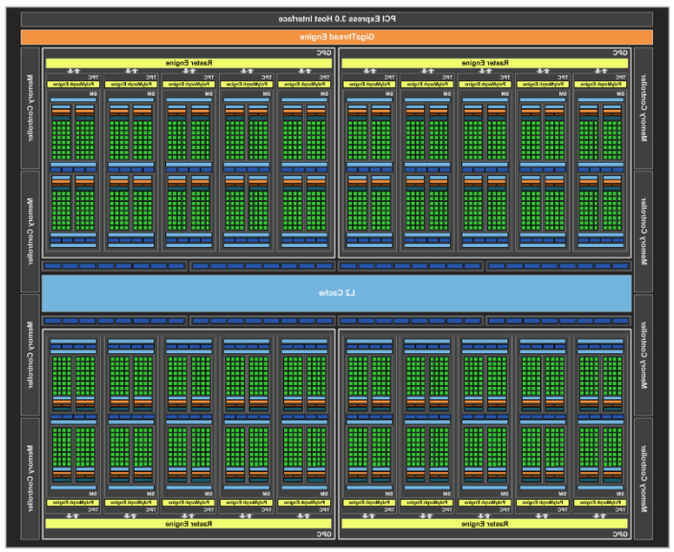


Heterogenous processors and accelerators

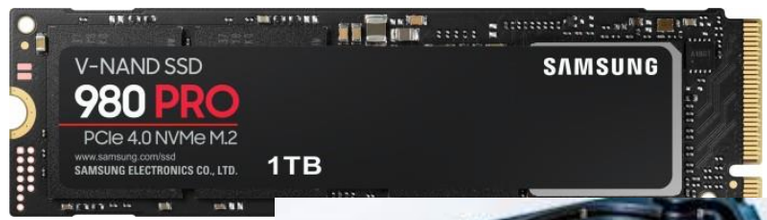


Hierarchical Memory

Graphics Processing

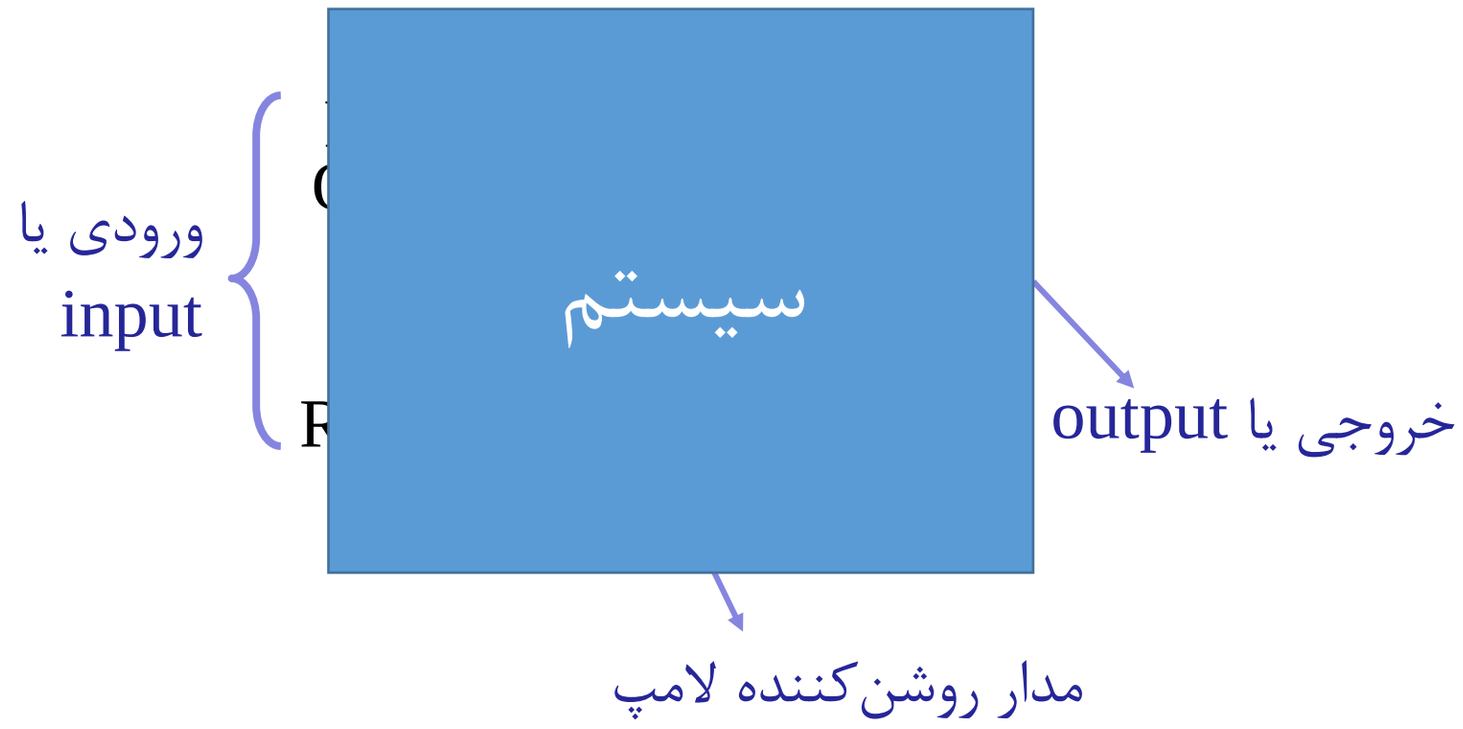


Persistent Storage



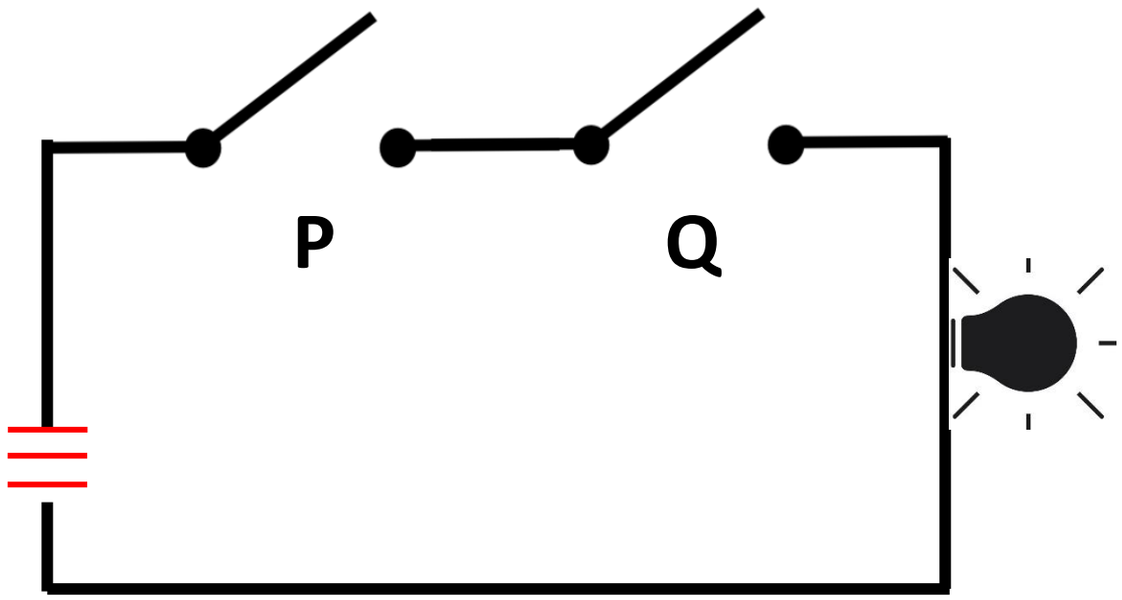
سیستم

- System is a set of components work as a whole to achieve a goal
  - Inputs, behavior, and outputs



  - Behavior is a function that translates input to outputs

سیستم



لامپ روشن می‌شود،
اگر هر دو کلید P و Q بسته باشد

Only if
اگر و تنها اگر

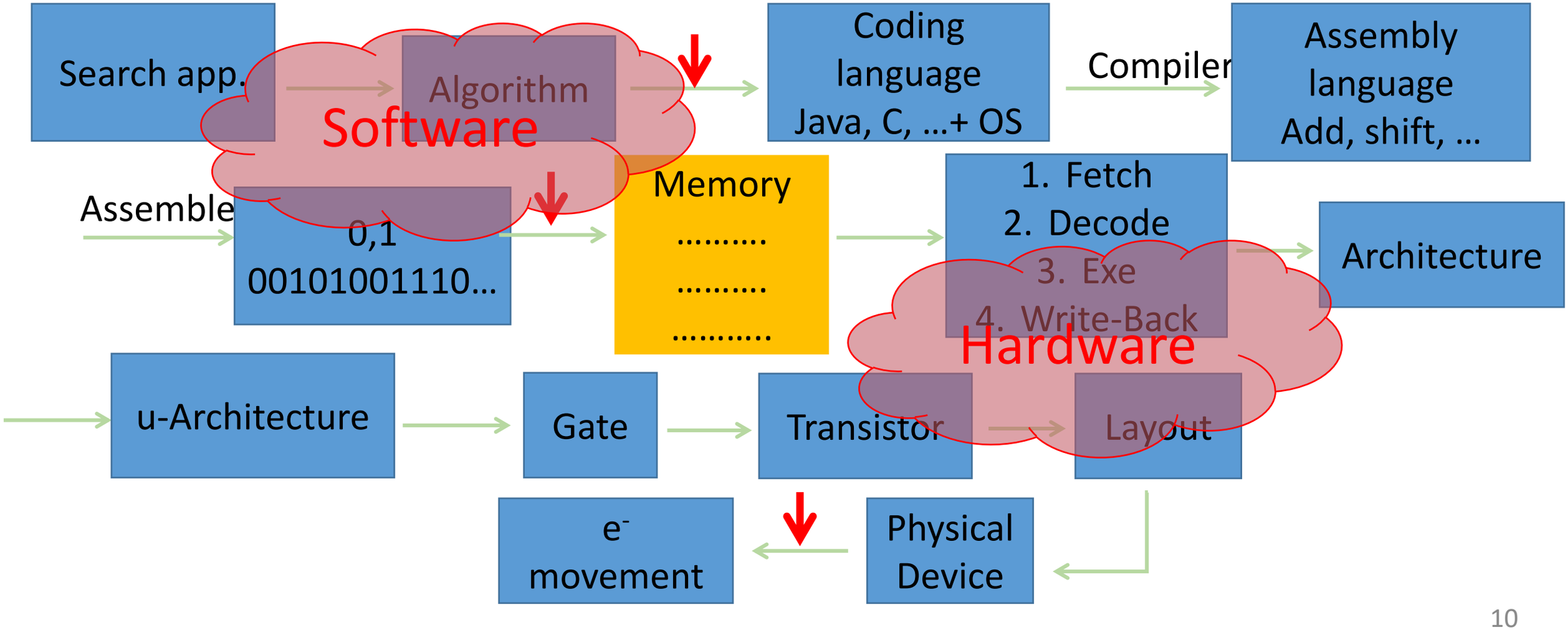
سیستم اساساً بهینه

- سیستم اساساً قابل اطمینان، دارای امنیت، و امن
  - Reliable, secure, and safe
- سیستم اساساً بهینه از نظر انرژی
  - Memory-centric
- سیستم اساساً کم‌تاخیر
  - Low latency
- سیستم اساساً کارایی بالا
  - High performance
- معماری در حوزه‌های هوش مصنوعی، یادگیری ماشین، درمان و سلامتی
  - AI, ML, Medicine and health

حوزه تحقیقاتی و تدریس در هم تنیده شده

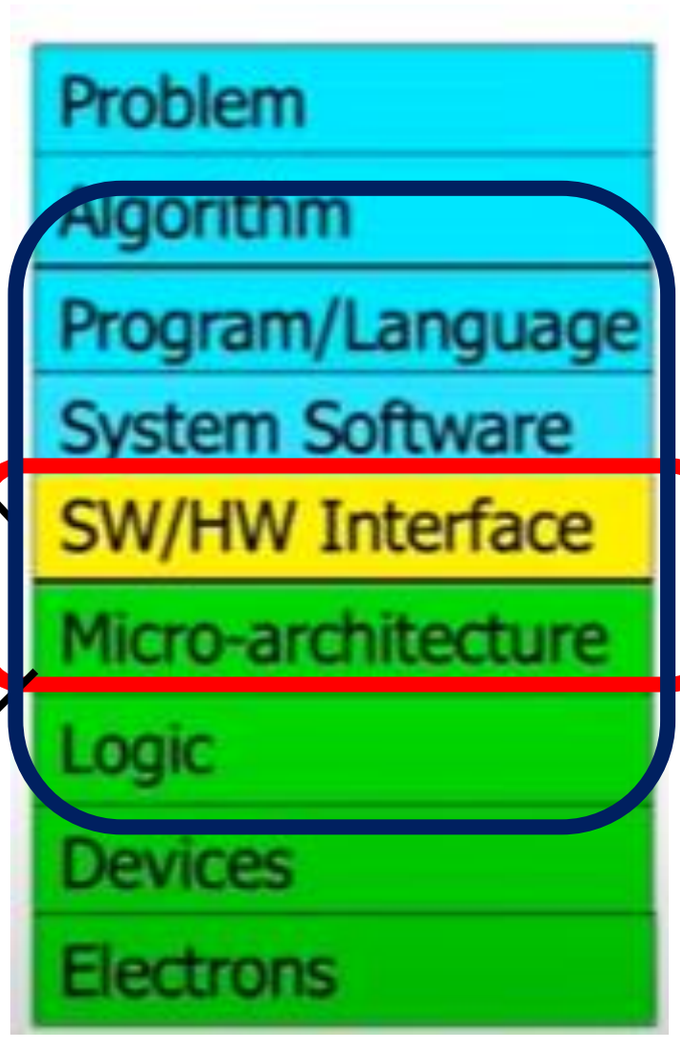
سیستم ← کامپیوتر

- Computer
  - A general purpose device that can be programmed to carry out a set of arithmetic or logical operations (wiki)
  - A system that can execute bunch of instructions



The Transformation Hierarchy

- **ترجمه مشکل به سطوح مختلف تجرید**
  - ✓ تبدیل حرکت الکترون‌ها به راه‌حل مشکل خود



دسته‌بندی کامپیوترها

- **کامپیوترهای رومیزی**
  - در حال انقراض
- **سرورها**
  - قابلیت بیشتر
  - خدمات دادن به کاربران بیشتر
- **کامپیوترهای نهفته**
  - موبایل‌ها
  - لوازم خانگی
  - در اتومبیل‌ها
  - ...

بهینه ← کاریی؟

- **کدام سیستم بهتر است؟**
  - گوشی ساده با پردازنده ARM11 با فرکانس 400MHz
  - پردازنده پنتیوم با 66MHz

- **چه آیتم‌هایی وجود دارد که تعیین‌کننده در کارایی باشد؟**

- **برنامه محک**
  - نحوه نوشتن برنامه، الگوریتم برنامه
  - زبان برنامه‌نویسی (مقایسه C و Java)
  - نوع پردازنده و سرعت حافظه‌ی سیستم
  - سیستم عامل
  - ارتباطات بین اجزا
  - ISA

بهینه ← کاریی؟

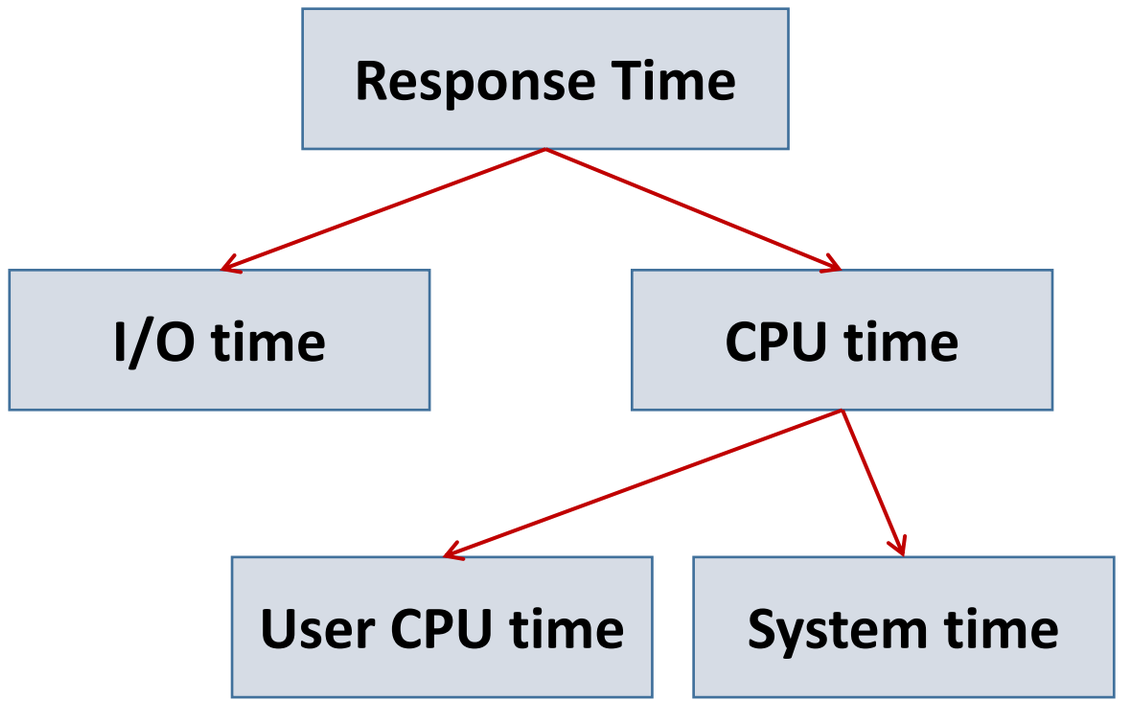
- Response time
  - Total time to complete a task
  - OR: how long it takes to do a task
    - Processing, I/O, OS overhead, idle time
- Response time consists of:
  - CPU time
    - CPU time spent on a program
    - The amount of time used for processing instructions
  - I/O time
    - Time elapsed to wait for I/O operations

بهینه ← کاریی؟

- CPU time: Time spent processing a given job
  - Discount I/O time, other jobs' shares
- CPU time comprises
  - User CPU time
    - CPU time spent on a program
  - System time
    - CPU time spent on OS doing tasks on behalf of program
- Performance = 1/response time

بهینه ← کاریی؟

- Performance = 1/response time
- System Performance
  - 1/elapsed time
- CPU performance
  - 1/user CPU time



CPU Time

CPU Time = # of Clock Cycles * Clock Cycle Time

= # of Clock Cycles * T

= # of Clock Cycles / F

F: CPU Frequency (Hz)

- Example: Which one is faster? A or B?

Computer	Frequency	# Clks
A	3.6 GHz	2000
B	2.6 GHz	1500

CPU Time

- CPI: # of clocks per instruction
- # of Clock Cycles
  - # of Clks = # of instructions * CPI

- Example: CPU time = ?

# of instr.	4,000
CPI	3
Freq.	2.6 GHz

CPU Time

- If different instructions have different CPI
  - CPI: Average CPI (Average cycles per instr.)

$$CPI = \frac{1}{n}\sum_{i=1}^{n} \#\ of\ Clk\ for\ Instr._i$$

N: classes of instr.

- CPU time = # of instr. * CPI * T
- # of instructions or Instruction Count (IC)
  - code size or lines of code????

Performance Evaluation

- Consider two CPUs
  - CPU1 runs faster then CPU2 on program A
  - CPU2 runs faster then CPU1 on program B
- Which CPU is faster? or
  - Which program should we choose to compare CPUs?
- Benchmarks: a **set** of programs used to compare performance of processors

Performance Evaluation

- Benchmarks
  - Why a **set** of programs?
  - Why not just running a simple program?
    - Agreeing on one simple program is very hard
    - Designers can optimize their CPUs towards fast running that simple program
- Standard benchmarks
  - SPEC: Standard Performance Evaluation Corporation
    - SPCE2000, SPEC2006, SPEC2017

Performance Metric

- MIPS: Million Instruction Per Second
- MIPS Drawbacks
  - Not into account capabilities of instructions
    - Comparing two different ISAs is not fair
  - Not realistic metric even on same CPU
    - MIPS(A) > MIPS(B) → Performance(A) > Performance(B)??

Performance Metric

- MFLOPS: Million Floating Point Operation Per Second
  - Popular in scientific computing
  - Drawbacks
    - Ignore other instructions (Load/Store)
    - Not all FP operations have common format
    - Depends on how FP-intensive program is
- So, how determine the best processor?
  - What does the best mean?
    - Performance, cost, power, energy, ISA, …. ????

معماری کامپیوتر؟

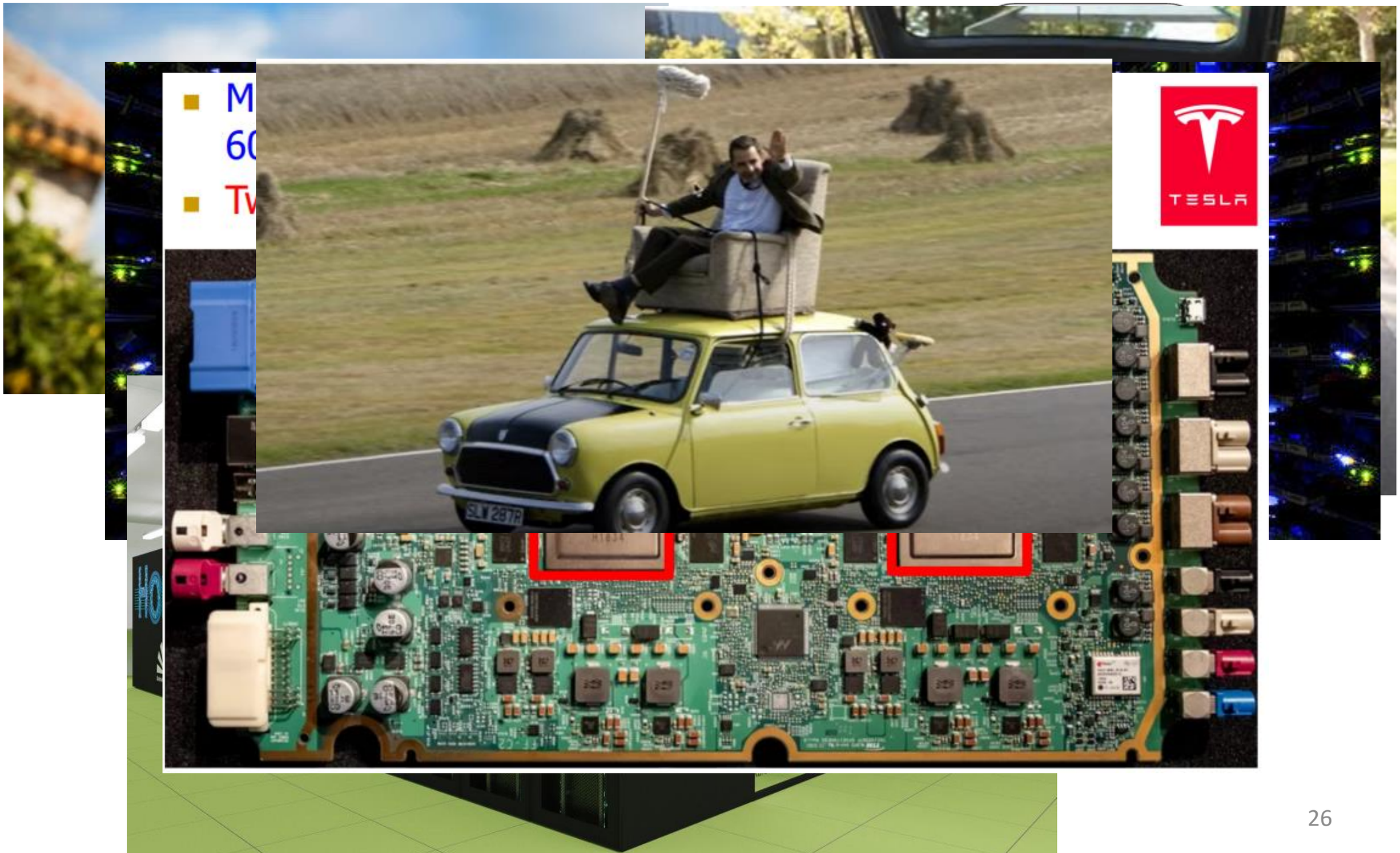
- **علم و هنر طراحی سیستم‌هایی برای محاسبات**
  - سخت‌افزار
  - رابط
  - SystemSW
  - مدل برنامه‌نویسی
- **برای دستیابی به مجموعه‌ای از اهداف طراحی**
  - مثال: دستیابی به بالاترین کارایی برای برنامه‌های X, Y, Z
  - مثال: طراحی سیستمی با بالاترین عمر باتری
  - ...
- **نوع ماشین و سیستم مطرح نیست بلکه هدف طراحی مطرح است**
  - طراحی یک سوپرکامپیوتر متفاوت از طراحی یک گوشی هوشمند ← بسیاری از نکات طراحی اصلی یکسان است
  - استفاده از مفاهیم برای دسترسی به طراحی موردنیاز

هدف معماری کامپیوتر؟

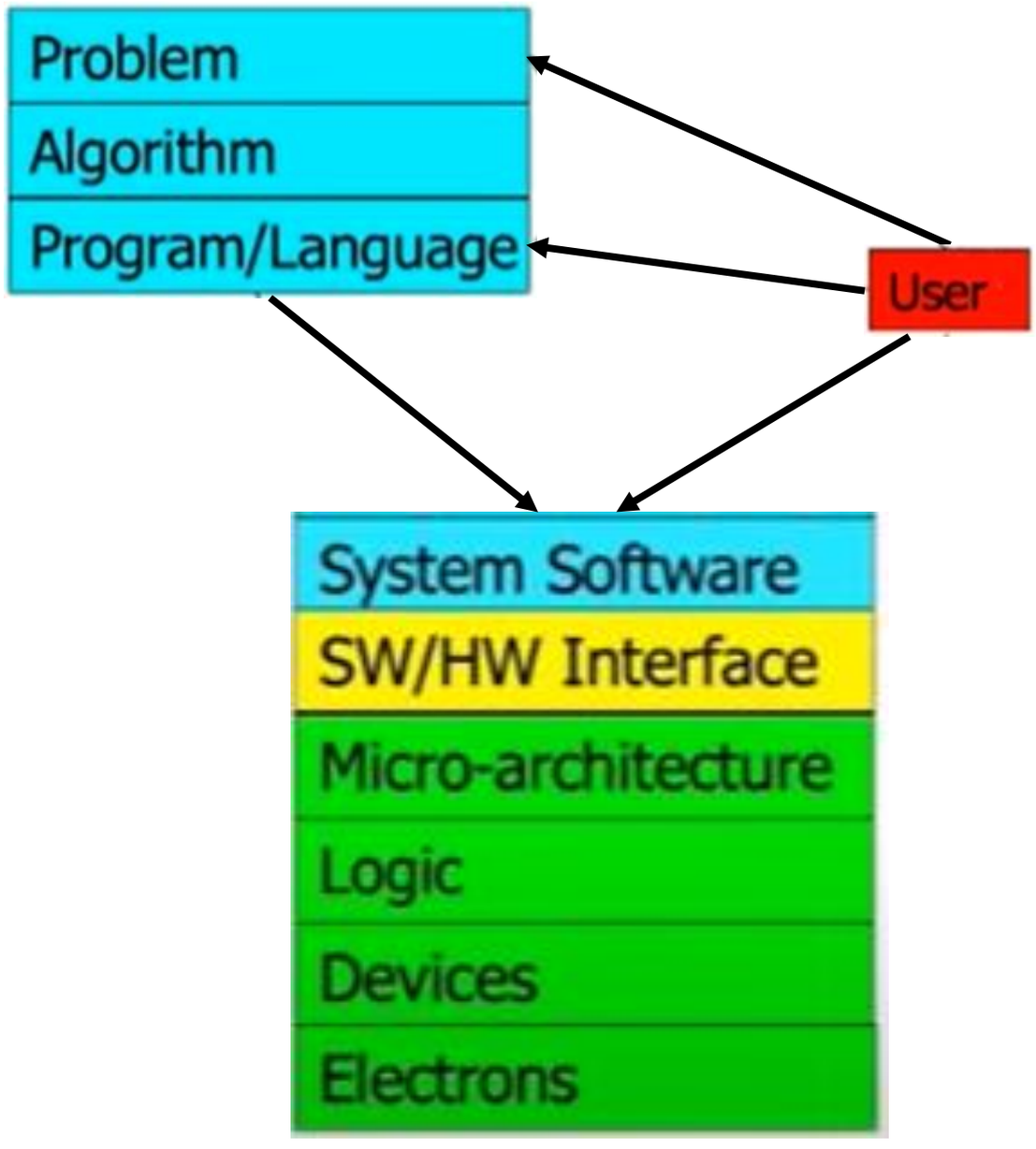
- **علم و هنر طراحی، انتخاب، و اتصال اجزای سخت‌افزاری، و طراحی رابط سخت‌افزاری/نرم‌افزاری برای ساخت سیستم محاسباتی برای رسیدن به اهداف خاص مانند انرژی، هزینه، کارایی، مساحت و ...**

- **طراحی سیستم بهتر**
  - کامپیوتر سریع، ارزان، کوچک، کم مصرف، قابل اطمینان و ...
- **فعال کردن اپلیکیشن‌ها/نیازهای جدید**
  - خودروهای خودران، واقعیت مجازی، سلامتی، و ...
- **درک اینکه چرا کامپیوترها به این شکل امروزی کار می‌کند**

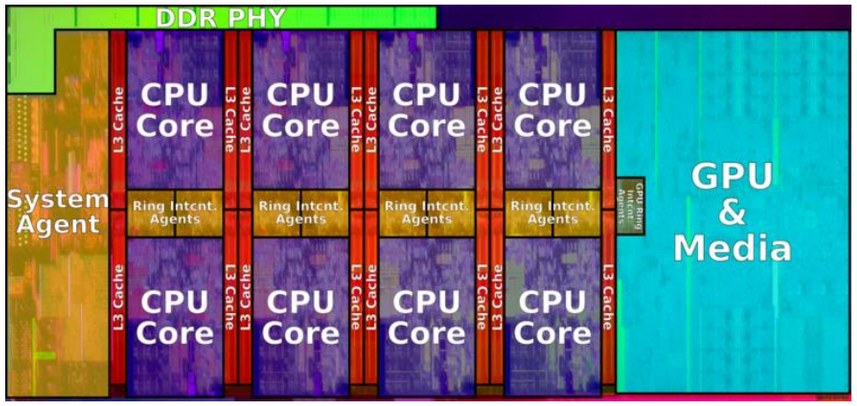
Different platforms and goals



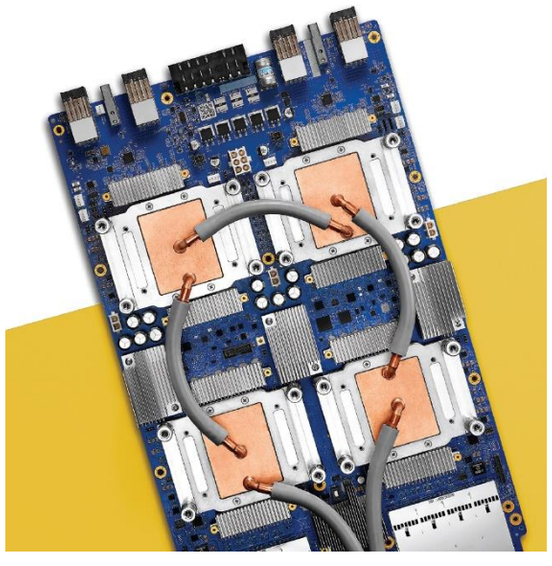
هدف معماری کامپیوتر؟



Different platforms and goals



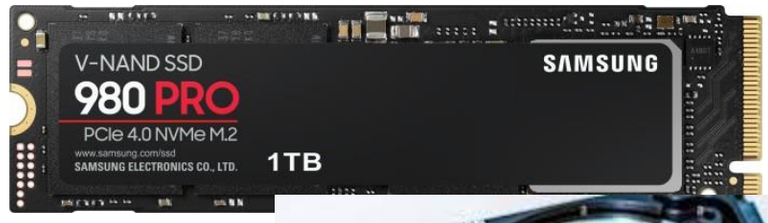
Heterogenous processors and accelerators



Hierarchical Memory

Graphics Processing

Persistent Storage



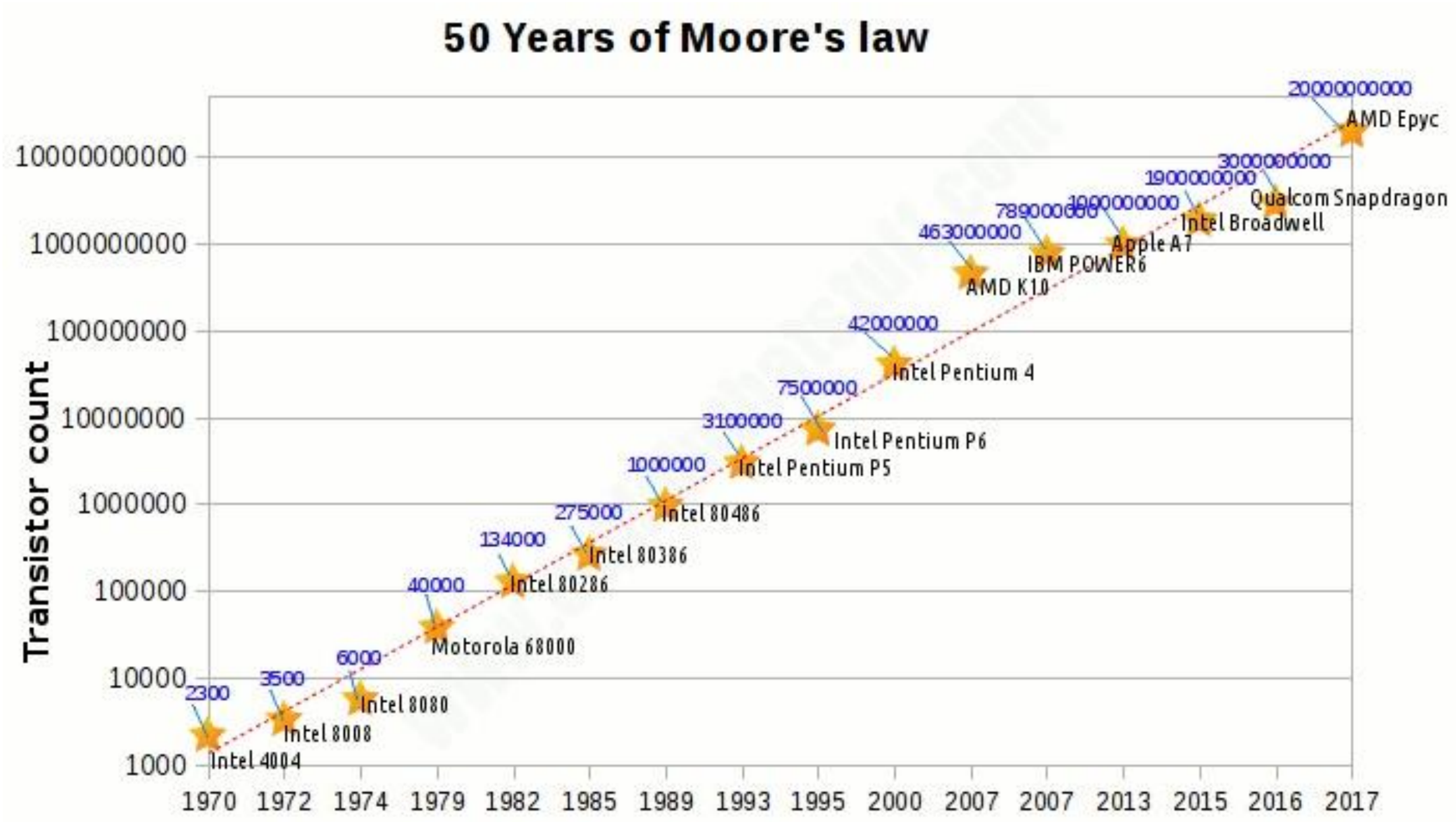
کدام سطح؟

There's Plenty of Room at the Bottom

From Wikipedia, the free encyclopedia

"There's Plenty of Room at the Bottom: An Invitation to Enter a New Field of Physics" was a lecture given by physicist Richard Feynman at the annual American Physical Society meeting at Caltech on December 29, 1959.[1] Feynman considered the possibility of direct manipulation of individual atoms as a more powerful form of synthetic chemistry than those used at the time. Although versions of the talk were reprinted in a few popular magazines, it went largely unnoticed and did not inspire the conceptual beginnings of the field. Beginning in the 1980s, nanotechnology advocates cited it to establish the scientific credibility of their work.

قانون مور


50 Years of Moore's law
Transistor count
10000000000
1000000000
100000000
10000000
1000000
100000
10000
1000
1970
1972
1974
1979
1982
1985
1989
1993
1995
2000
2007
2007
2013
2015
2016
2017
20000000000
AMD Epyc
3000000000
1900000000
Qualcom Snapdragon
1090000000
789000000
Intel Broadwell
463000000
Apple A7
IBM POWER6
AMD K10
42000000
Intel Pentium 4
7500000
3100000
Intel Pentium P6
Intel Pentium P5
1000000
Intel 80486
275000
Intel 80386
134000
Intel 80286
40000
Motorola 68000
6000
Intel 8080
3500
Intel 8008
2300
Intel 4004

کدام سطح؟

Feynman considered some ramifications of a general ability to manipulate matter on an atomic scale. He was particularly interested in the possibilities of denser computer circuitry and microscopes that could see things much smaller than is possible with scanning electron microscopes. These ideas were later realized by the use of the scanning tunneling microscope, the atomic force microscope and other examples of scanning probe microscopy and storage systems such as Millipede, created by researchers at IBM.

Feynman also suggested that it should be possible, in principle, to make nanoscale machines that "arrange the atoms the way we want" and do chemical synthesis by mechanical manipulation.

He also presented the possibility of "swallowing the doctor", an idea that he credited in the essay to his friend and graduate student Albert Hibbs. This concept involved building a tiny, swallowable surgical robot.

کدام سطح؟

RESEARCH

REVIEW SUMMARY

COMPUTER SCIENCE

There's plenty of room at the Top: What will drive computer performance after Moore's law?

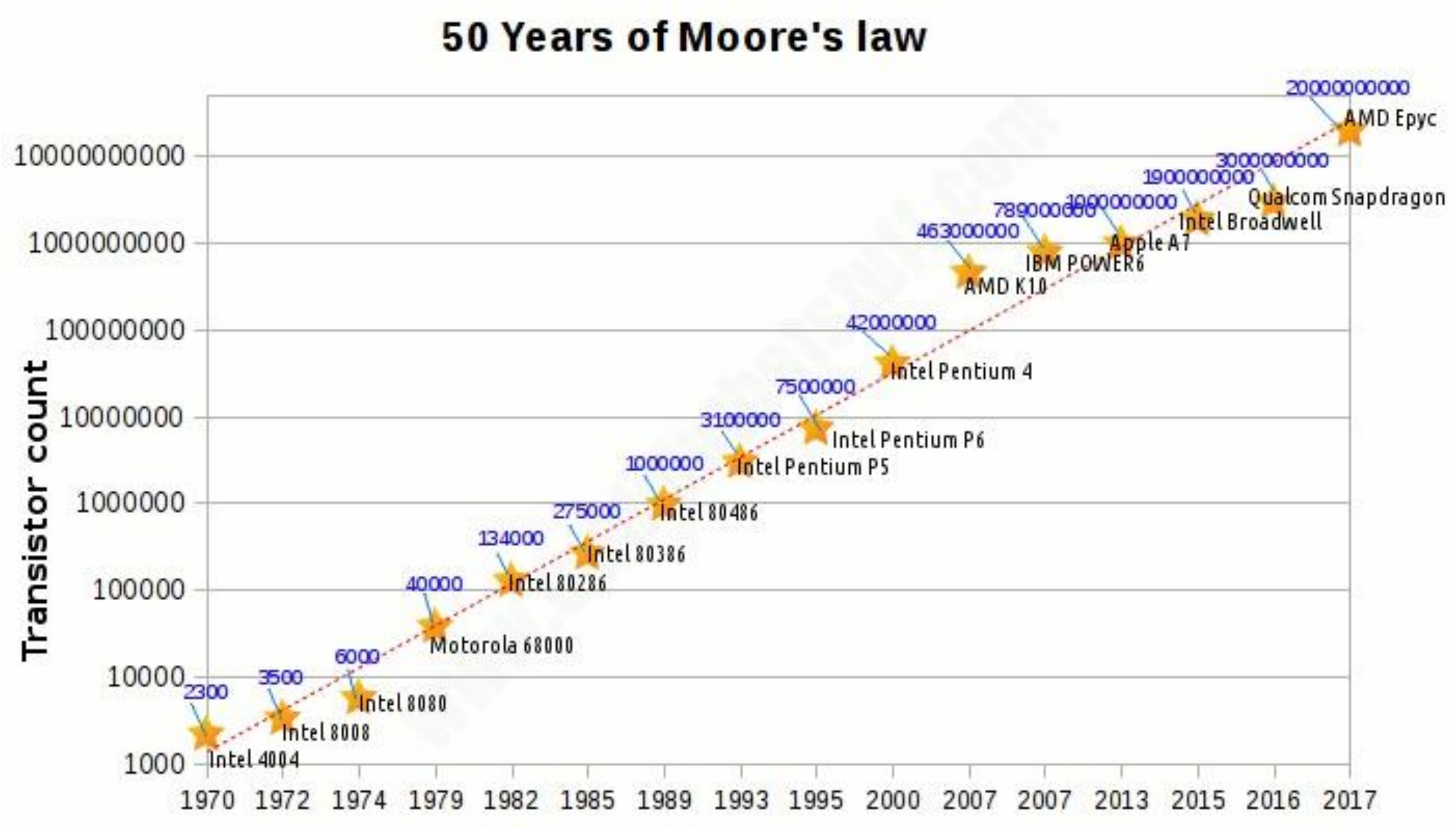
Charles E. Leiserson, Neil C. Thompson*, Joel S. Emer, Bradley C. Kuszmaul, Butler W. Lampson, Daniel Sanchez, Tao B. Schardl

BACKGROUND: Improvements in computing power can claim a large share of the credit for many of the things that we take for granted in our modern lives: cellphones that are more powerful than room-sized computers from 25 years ago, internet access for nearly half the world, and drug discoveries enabled by powerful supercomputers. Society has come to rely on computers whose performance increases exponentially over time.

in computing power stalls, practically all industries will face challenges to their productivity. Nevertheless, opportunities for growth in computing performance will still be available, especially at the "Top" of the computing-technology stack: software, algorithms, and hardware architecture.

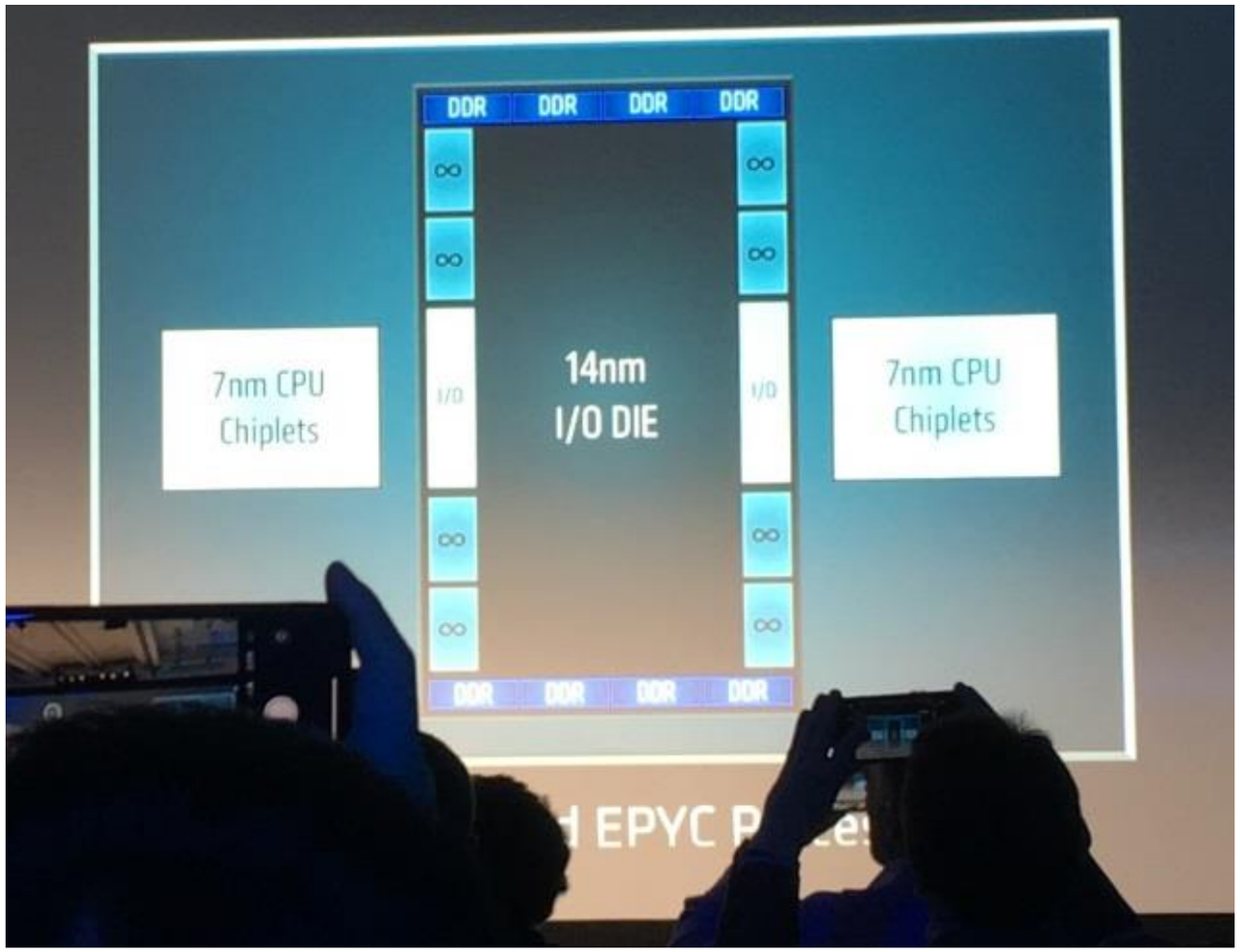
ADVANCES: Software can be made more efficient by performance engineering: restructur-

قانون مور


50 Years of Moore's law
Transistor count
10000000000
1000000000
100000000
10000000
1000000
100000
10000
1000
1970
1972
1974
1979
1982
1985
1989
1993
1995
2000
2007
2007
2013
2015
2016
2017
20000000000
AMD Epyc
30000000000
19000000000
Qualcom Snapdragon
7890000000
10000000000
Intel Broadwell
4630000000
Apple A7
IBM POWER6
AMD K10
42000000
Intel Pentium 4
7500000
3100000
Intel Pentium P6
1000000
Intel Pentium P5
275000
Intel 80486
134000
Intel 80386
40000
Intel 80286
6000
Motorola 68000
3500
2300
Intel 8080
Intel 8008
Intel 4004

قانون مور

- Lisa Su: “Forget about whether Moore’s Law is alive or dead. That’s an interesting conversation. But the more interesting thing is applications need for us to be above Moores’s Law. We need to do more than the industry has done in the past because the applications and data are such that there’s a lot more computation that’s necessary.”



https://www.extremetech.com/computing/309946-chiplets-do-not-reinstate-moores-law

کدام سطح؟

There is plenty of room both at the top and at the bottom

but much more so

when you communicate well between and optimize across

the top and the bottom.