frontiers | Frontiers for Young Minds

ENGINEERING AND TECHNOLOGY
Published: 18 August 2025
doi: 10.3389/frym.2025.1474522

WHY YOUR PHONE IS SO FAST: THE SPORTS CAR VS. THE TRUCK

David Patterson *

Computer Science Division, University of California, Berkeley, Berkeley, CA, United States

YOUNG REVIEWERS:

ANSAR
AGE: 14

ZICHEN
AGE: 12

In 1965, Gordon Moore graphed the number of transistors—the tiny switches that you can think of as the "atoms" of computer hardware—that could fit on each microchip over the prior few years, and he noticed that the number doubled each year. He predicted that annual doubling would continue for decades—a bold statement known as Moore's law. When building computers, one important decision is the vocabulary that the components use to communicate, called the instruction set. A big question in 1980 was whether instruction sets needed to change form as computers rapidly improved. Back then, most computer builders believed we should keep using the added transistors, following Moore's Law, to give new computers more complex instructions. Instead, I believed that we should keep the instructions simple...and that turned out to be the most successful method. Many awards honored these contributions, the most distinguished being the 2017 ACM A.M. Turing Award and the 2022 NAE Charles Draper Prize.

HOW DO WE KEEP MAKING COMPUTERS FASTER?

When I was your age, computers could be as big as cargo containers: 2.4 m wide by 2.5 m high by 12 m long! By the time I received my doctorate in computer science in 1976 from the University of California in Los Angeles, they shrank to the size of refrigerators by using **microchips**. The reason was one of the most amazing visionary and accurate predictions in the history of technology.

In 1965, Gordon Moore graphed the number of **transistors**—the tiny switches that you can think of as the "atoms" of computer **hardware**—that could fit on each microchip over the prior few years, and noticed that the number doubled each year. Microchips enabled computers to shrink from cargo containers to refrigerators. Moore then boldly and publicly predicted that annual doubling would continue for decades. His prediction became known as **Moore's Law**. A cartoon in his original paper (Figure 1) suggested that computers would eventually be sold alongside personal items like buttons and lipstick. In 1976, Apple Computer fulfilled Moore's prophecy by manufacturing computers for the home.

Figure 1

For those of us who believed in Moore's Law in the 1970s, it meant that ultimately the fastest computers in the world would shrink down to a single microchip since the distance between computer components shortens, so it would take less time for the electrical signals to get there. I remember telling that to a class at the University of California in Berkeley—where I became a professor after graduation—when computers were larger and heavier than I was, and the students laughed!

Moore's Law continued for decades, so we were able to design faster computers with the increasing number of better transistors that Gordon Moore predicted. To fit more transistors on a chip, transistors had to be even smaller and thereby faster. Faster computers enabled many of the advances in technology that we enjoy today. For example, they enabled the smartphones and smartwatches that many of us now rely upon. Computers that are even more powerful will be needed

MICROCHIP

Properly called an integrated circuit, it is a tiny, powerful electronic device made of semiconductor material that contains many transistors.

TRANSISTOR

The tiny switches that you can think of as the "atoms" of computer hardware. A microchip can contain billions of transistors.

HARDWARE

The physical computer that runs programs, so named because it is hard to change.

MOORE'S LAW

A prediction by Gordon Moore that the number of transistors per microchip would double every year or two.

Figure 1

A cartoon in the original Moore's Law paper in 1965, predicting that it would lead to computers for the home.

to keep us on this exciting path—particularly if we want computers that perform tasks as well as or better than people do, which is called **artificial intelligence** (AI).

ARTIFICIAL INTELLIGENCE (AI)

Software that performs tasks as well as or better than people do.

TAKING A RISC TO HELP MAKE COMPUTERS FASTER

An architect designs buildings, so we call the person who designs a computer a computer architect. We distinguish between physical computers and the programs that run on them by calling the physical computers **hardware** (because they are hard to change) and the programs **software** (because they are easy to change). An important computer architecture decision is the vocabulary that software and hardware use to communicate. This vocabulary is called the instruction set, and the "words" are called instructions. Examples of instructions are the functions you see on the keys of a calculator: add, subtract, multiply, divide. Computer programs consist of millions of these instructions, executed billions of times per second.

SOFTWARE

The programs that run on the hardware computer, so named because it is easy to change.

An important computer architecture question in 1980 was whether the instruction set for the single-chip computer should follow the style of the large, multi-chip computers or change its nature to better match the rapidly improving **microprocessors** (the **processor** is the piece of the computer hardware that does the arithmetic, so a processor inside a single microchip was sensibly named a microprocessor).

MICROPROCESSOR

A processor inside one microchip.

PROCESSOR

The piece of the computer hardware that performs arithmetic.

In 1980, most computer architects believed we should use the principle of Moore's Law to keep increasing the number of transistors to provide computers with more complex instructions. In response, I wrote an article called *The Case for the Reduced Instruction Set Computer (RISC),* arguing that we should instead keep the instructions simple (reduced) (in some engineering fields the appropriate catchphrase is KISS, for "keep it simple, stupid"). I sent a draft to friends in the computer industry to get their comments. They were building refrigerator-sized computers that used complex instruction sets, which I called complex instruction set computers (CISC). Instead of sending me comments, they wrote a paper that contradicted our arguments, to appear next to my writing.

This resulting RISC vs. CISC controversy led to heated arguments within the computer architecture community. John Hennessy—another young professor at nearby Stanford University—and I defended the RISC side in debates at several conferences. To tease my debate opponents, I used a large, heavy, slow vehicle to represent CISC and a small, light, fast, modern sports car for RISC (Figure 2).

Besides publications and philosophical debates, Berkeley and Stanford built RISC microprocessors. For fun, we put an image of a sports car and the project name on the Berkeley microchip (Figure 3). In 1983, students presented their designs at the leading

Figure 2

Vehicles to suggest the differences between CISC and RISC. The top two pictures are a modern version of CISC and RISC that may be easier to understand. The bottom two drawings show the original version of this comparison. As these images were created before PowerPoint was developed, I asked an artist to draw them—my wife!

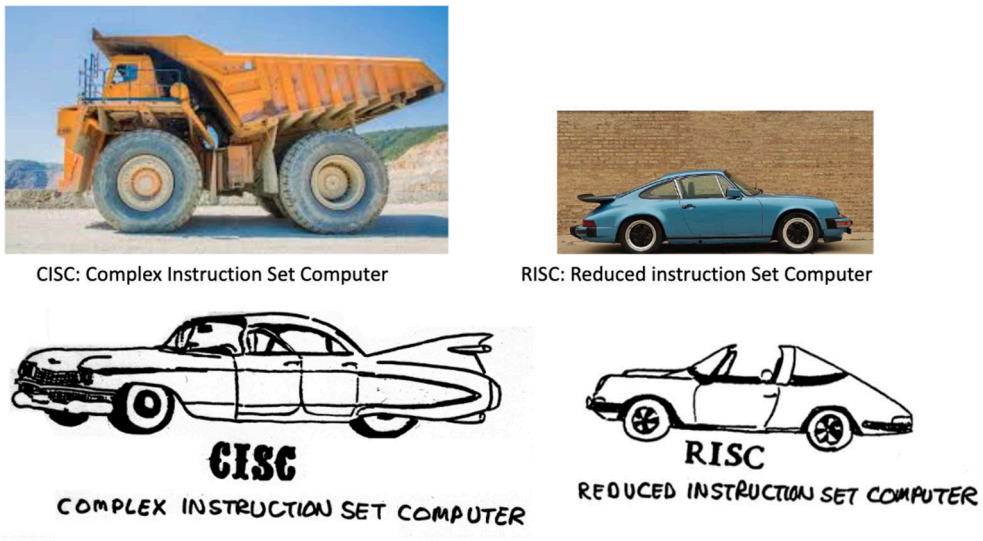

Figure 2

conference on microchips. The talks stunned the giant audience. Remarkably, students designed a better microprocessor than the computer industry did! In 2015, a computer history organization placed a brass plaque on the Berkeley campus to honor the first RISC microprocessor.

Figure 3

(A) The UC Berkeley RISC microchip, and **(B)** a closeup showing the sports car logo in the upper left corner. The car is just 0.4 mm long in the 4 mm by 8 mm microchip, so you need a magnifying glass to see it.

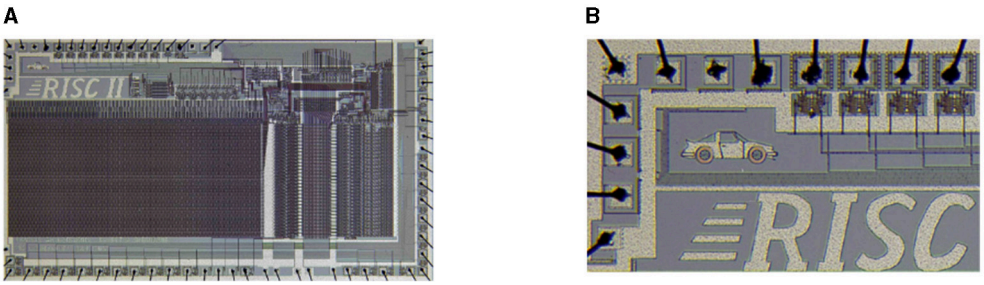

Figure 3

To understand the RISC vs. CISC argument, let us use transportation as an example. Which would deliver 10 passengers between two cities in less time: a small, fast sports car or a large, slow vehicle that holds more people? The answer depends on how many trips are needed and how long trips take. When we measured hardware, programs using the RISC vocabulary read about 1/4 to 1/3 more instructions than CISC (more "trips"), but they read them 4 to 5 times quicker (faster "trips"), so RISC ended up being three to four times faster than CISC!

Today, all smartphones, tablets, and smartwatches use RISCs—the company that supplies the computer architecture is Advanced RISC Machine (ARM), and 99% of all microprocessors shipped today are RISC. We believed our ideas would win eventually, but we are still amazed at its popularity [1].

My friend and collaborator John Hennessy and I later wrote a textbook on computer architecture [2]. We received many awards for our contributions to RISC and the textbook; the most distinguished being the 2017 ACM A.M. Turing Award, considered the Nobel Prize of computing, and the 2022 NAE Charles Draper Prize, considered a Nobel Prize of Engineering.

HOW WILL WE KEEP MAKING COMPUTERS FASTER AFTER MOORE'S LAW SLOWS?

After doubling the number of transistors per chip like clockwork for decades, Moore's Law is finally stalling. The number of transistors per chip is still increasing, but much more slowly than in the past. Programmers still count on faster computers to keep developing new, more demanding software that users enjoy, like fancier video games. How can a computer architect design faster computers without the big help from Moore's Law?

The only option left is to make special-purpose computers that are dedicated to a narrow class of software. The idea is to dedicate the hardware resources on the microchip to speed up operations important for one application but not for most others. For example, suppose an application needed many more "multiply" operations than typical, but fewer "divides". A computer focused on multiplying could reduce the hardware needed for divide operations and use the savings to make multiplying faster. In our acceptance speech for the Turing Award, John Hennessy and I predicted that the next decade would be a Golden Age for computer architecture, in part because architects would need to design many narrow-application microprocessors.

The question then is "which applications should we help"? About the same time that Moore's Law started to slow, an important domain emerged. As Turing laureate Yann LeCun wrote in another Frontiers for Young Minds article, starting a little over a decade ago one approach to AI made a great leap forward. One reason for this advance is that computers finally got fast enough to make LeCun's version of AI beat alternatives in head-to-head competitions. Examples are an AI that beat a human world champion at a game called Go, and an AI that solved a long-standing problem in biology called protein folding, leading to a Nobel Prize.

By designing microprocessors only for the AI domain, we can continue to build much faster computers despite Moore's Law winding down. In 2016, the first commercial AI microchip was 30 times faster than conventional microprocessors! AI microchips have continued to make big gains every year. The potential societal impact of such AI microchips is one reason why NVIDIA, a company that makes them, saw its value skyrocket in the past year.

HOW DOES ONE BECOME A COMPUTER ARCHITECT?

The first step to becoming a computer architect is to learn to program, which is how I first fell in love with computing. An architect needs to understand software in order to invent new hardware to speed it up.

GRAPHICS PROCESSING UNIT (GPU)

A special-purpose computer that performs graphics processing and AI well, but not applications outside of this narrow range (GPU).

If you have access to a personal computer on which to play games, it probably includes an NVIDIA **graphics processing unit**. As the name suggests, a GPU is narrow application processor used to generate the graphics that you see in video games and movies. GPUs contain hundreds of processors, and they are faster because they can read more instructions per second in parallel. Recent GPUs also have support for AI, so they try to be good for both graphics and AI. NVIDIA invented the programming language CUDA to make graphics programming easier, and researchers soon used CUDA to program AI. Thus, a second step toward becoming a computer architect would be to learn how to write fast programs in CUDA on GPUs, which will likely require you to learn about the unusual hardware features of GPUs that are not found in conventional computers. You can even try your hand at AI!

After learning about programming in general and for GPUs in particular, the next step would be to start reading articles that explain how computer hardware works. I have put in a few suggestions in the references listed below, in increasing levels of difficulty. The first is a recent magazine article about the popularity of RISC in general and about a recent variation called RISC-V ("RISC five") [1]. Next is a more detailed description of the history of RISC [3]. The following two papers appeared in the magazine *Scientific American*, intended for the general public. The 1983 article explains the insides of processors, particularly the difficult control hardware, and how that led to RISC [4]. The 1995 article, for the 150th anniversary of Scientific American, tried to predict what computers would be like 25 years later [5]. The next reference is the first chapter of our textbook, which is intended to introduce college students to computer design [6], and finally our most advanced textbook for senior college students [2].

Today, most architects have an advanced college degree, like a master's or Ph.D., in computer science. If you get the chance, I recommend taking a few more years after a four-year undergraduate college degree to get one. You get to work on really interesting problems, and since people of your generation are likely to live at least 100 years, you still have plenty of time for a long retirement. I continue to enjoy my work, nearly 50 years after my final college degree—Hennessy and I just completed the 7th edition of our textbook, 35 years after the first edition—yet I have plenty of time to retire if I ever get bored with computer architecture.

ACKNOWLEDGMENTS

I would like to thank Elaine Ma (age 11) and my grandnieces Penelope Garnetti (age 11), Clara Stafford (age 11), and Ella Stafford (age 14) for their comments that improved this paper.

REFERENCES

1. Kehe, J. 2025. Angelina Jolie was right about computers. *Wired* 32:27–32.
2. Hennessy, J., and Patterson, D. 2019. *Computer Architecture: A Quantitative Approach*, 6th Edn. Burlington, MA: Morgan Kaufmann.
3. Patterson, D. 2022. *2022 Draper Prize Acceptance Remarks*. Available online at: https://www.nae.edu/276664/2022-Draper-Prize-Acceptance-Remarks (Accessed August 1, 2024).
4. Patterson, D. 1983. Microprogramming. *Sci. Am.* 248:50–7.
5. Patterson, D. 1995. Microprocessors in 2020. *Sci. Am.* 273:62–7.
6. Patterson, D., and Hennessy, J. 2021. "Computer abstractions and technology", in *Computer Organization and Design RISC-V Edition: The Hardware Software Interface*, 2nd Edn. Burlington, MA: Morgan Kaufmann, 2–59.

SUBMITTED: 01 August 2024; **ACCEPTED:** 24 July 2025;
PUBLISHED ONLINE: 18 August 2025.

EDITOR: Idan Segev, Hebrew University of Jerusalem, Israel

SCIENCE MENTORS: Adiya Rakymzhan and Guangyu Jeff Zou

CITATION: Patterson D (2025) Why Your Phone is so Fast: The Sports Car vs. The Truck. Front. Young Minds 13:1474522. doi: 10.3389/frym.2025.1474522

CONFLICT OF INTEREST: The author declares that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

YOUNG REVIEWERS

ANSAR, AGE: 14
I am a 14-year-old boy and I live in Kazakhstan. I am a huge fan of swimming, and I compete on a professional level. I am also a part of local urban planning community focused on eco-friendly and sustainable solutions. In my free time from school, I love baking, dancing, reading fiction books, and spending time with my friends.

ZICHEN, AGE: 12

I am currently 13 years old. I enjoy reading and playing interscholastic sports. I am curious and like to learn. I am also volunteering in a non-profit organization that helps kids with autism.

AUTHORS

DAVID PATTERSON

I was always the youngest and smallest in school, which is likely why I wrestled in high school and college. Fortunately, I got bigger later. My third college year I studied computing and loved it. An undergraduate research job ultimately led to my doctorate. I got a chance to build computers, which was even more exciting than programming. My wife grew up near UC Berkeley, so near graduation she insisted I call about my application. The chairman said "you are in the top 10, but not the top five". But he said that to anyone who called! He then asked another professor to talk to me, and I eventually got the interview and job. After 40 fun years as a Berkeley professor, I started my second career as a computer architect for AI microchips at Google. *pattrsn@cs.berkeley.edu