Reduced Instruction Set Computers Then and Now

David Patterson

A widely cited Computer *article published in 1982 described the reduced instruction set computer (RISC) as an alternative to the general trend at the time toward increasingly complex instruction sets. A RISC executes most instructions in a single short cycle.*

FROM THE EDITOR

As part of our 50th anniversary celebration, this special feature revisits influential *Computer* articles from the past. This final installment reflects on the history and impact of the reduced instruction set computer (RISC) and highlights recent RISC developments. *–Ron Vetter, Editor in Chief Emeritus*

Amazingly, 35 years after "A VLSI RISC" was published in *Computer*,[1] the consensus remains that the reduced instruction set computer (RISC) is the best instruction set architecture (ISA). Indeed, we haven't seen any general-purpose complex instruction set computers (CISCs) since.

The hardest part of computer design is control. In 1958, Maurice Wilkes proposed that control be built from memory. He dubbed this microprogramming, in that designing control resembled low-level programming.[2] IBM embraced microprogramming to deliver their System/360 family of computers. Moore's law meant that control store could grow, which enabled bigger ISAs with many sophisticated instructions. Examples of CISC ISAs were Digital Equipment Corporation's (DEC's) VAX and the Intel 80x86. DEC engineers later found that 20 percent of VAX instructions were responsible for 60 percent of microcode, but only accounted for 0.2 percent of execution time.[3] Such results called into question the value of big microcoded interpreters and the CISC ISAs they enabled.

An alternative was having an ISA so simple that it didn't require a microcoded interpreter. The next insight was to convert the fast memory of control store into an instruction cache of user-visible instructions. The contents of fast instruction memory could change to fit what the executing application needs rather than always hold an ISA

0018-9162/17/$33.00 © 2017 IEEE

interpreter. The simple ISA also made it easier to pipeline, which led to faster clock rates. The advances of Moore's law meant that in the early 1980s a 32-bit datapath with small caches could fit on a single chip. This ISA style became known as a RISC.[4]

One skepticism of the RISC was that it would execute more instructions than a CISC. So how could it be better? This formula answered the question:

$$\frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Clock cycle}}{\text{Instructions}} \times \frac{\text{Time}}{\text{Clock cycle}} = \frac{\text{Time}}{\text{Program}}.$$

CISCs executed fewer instructions per program, but RISCs executed many fewer clock cycles per instruction (CPI) on average. DEC engineers later found about a three-fold advantage for RISCs.[5] A CISC executed roughly half as many instructions as a RISC, but its CPI was about six times higher. RISC research projects at IBM, UC Berkeley, and Stanford paved the way for commercial success—including ARM (Advanced RISC Machine),[6] MIPS, POWER, SPARC, and many others—as RISCs offered the highest-performing processors in the mid-1980s and 1990s.

Intel 80x86 microprocessors eventually surpassed RISCs. Hardware translated 80x86 instructions into internal RISC-like instructions, which allowed Intel to then copy any performance enhancements developed for RISC processors. Examples include long pipelines, fetching multiple instructions per clock cycle, and branch prediction. Given superior semiconductor processing and circuit design, the 80x86 ISA eventually offered the fastest processors. It largely overtook the market of small servers from RISCs in the 2000s to complement its near monopoly of the PC market.

The extra overhead in energy and area of hardware translation was affordable given the large transistor budgets of microprocessors for PCs and servers, but it was too costly for the embedded market. For example, close to 100 percent of Android and Apple phones and tablets use RISC processors. Figure 1 shows the billions of RISC shipments over the past decade, increasing seven-fold since 2007 and growing at an annual rate of 24 percent.

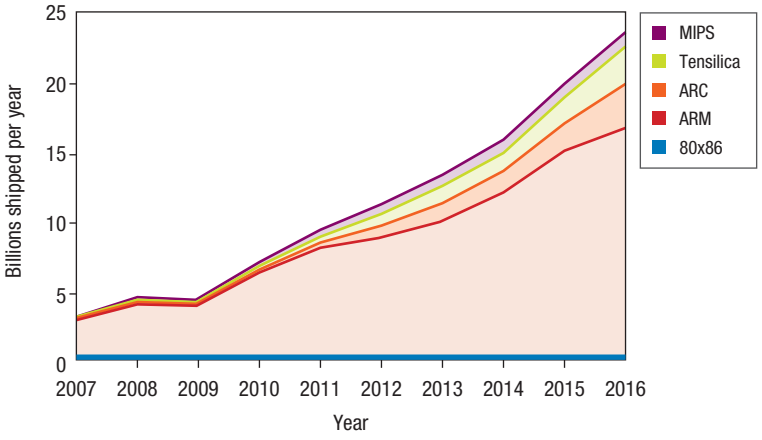

Figure 1. Billions of chips shipped from 2007 to 2016 using ARM (Advanced RISC Machine), ARC (Argonaut RISC Core), Tensilica, or MIPS instruction set architectures (ISAs) versus Intel's 80x86, which peaked at 0.365 billion in 2011.

The 80x86 ISA dominated sales in the PC era, but RISC architectures are kings of the post-PC era. Annual 80x86 shipments peaked in 2011 at 0.365 billion and have been declining about 8 percent annually since; Intel made fewer 80x86 chips in 2016 than in 2007. While the 80x86 dominates the cloud portion of the post-PC era, it's estimated that Amazon, Google, and Microsoft clouds collectively contain only 10 million servers.[7] Although these chips are expensive, their volume is negligible; in 2017, 10 million RISC chips ship every four hours. RISCs make up 99 percent of microprocessor volume today.

The renewed need in the post-PC era for simpler ISAs led to the RISC-V.[8] (This is pronounced "RISC five" since it's the fifth RISC architecture from UC Berkeley.) Keeping with its heritage, the RISC-V is a minimalist ISA; in fact, the base ISA is remarkably similar to its great-great-grandparent RISC-I.[9] One indication of complexity is the size of the documentation. The ISA manual for x86-32 is 2,198 pages or 2,186,259 words.[10] The RISC-V equivalents are 236 pages or 76,702 words.[11] If someone were to read manuals as an (incredibly boring) full-time job—eight hours a day for five days a week—it would take a month to read the x86-32 manual but less than a day to read the RISC-V manual.

Because it's new, the RISC-V avoids the mistakes of past ISAs.[8] For example, it's modular: a small base ISA runs a full software stack (OS, libraries, debuggers, and compilers). The base is frozen and will never change, giving programmers a stable target. The modularity comes from optional standard extensions: multiply and divide, floating-point arithmetic, atomic operations, compact code, and vector instructions.

To achieve the software-desirable goal of a single ISA that works from the smallest to the largest computers, it needs to lead to efficient designs for

both edge devices and the cloud. To empower large-scale computers, the RISC-V offers 64-bit as well as 32-bit address versions. Minimalism and modularity enable small and low-energy implementations of the RISC-V, which helps embedded applications. While some argue that ISA complexity doesn't matter for high-end processors, it does matter for low-cost applications, which the lack of success of the 80x86 illustrates. A universal ISA must work well everywhere. To support domain-specific architectures (DSAs), such as Google's tensor processing unit (TPU),[12] the RISC-V reserves opcode space to allow tight coupling of custom accelerators.

However, the RISC-V's most unconventional feature is that it's open. Its future is free from the fate or decisions of a single corporation, which have doomed numerous ISAs in the past.[8] Instead, it belongs to a nonprofit foundation with more than 65 corporate members (riscv.org). Its goal is to maintain the stability of the RISC-V; evolve the ISA slowly and carefully, keeping technological changes in mind; and to try to make it as popular as software open source projects like Linux. This openness enables any organization to develop and share implementations of the RISC-V. Competition, a free market, and open implementations might lower costs and increase innovation, similar to the benefits of open source software. Open designs also reduce the odds of unwanted malicious secrets being hidden in a processor.

Ironically, the end of Moore's law might rejuvenate computer architecture research and development, as advances must come from ISA-level innovation such as DSAs. RISCs in general and the RISC-V in particular will likely thrive during this renaissance.

ARCHIVED ARTICLES

The original paper remains very popular, as indicated by the number of downloads it receives from the IEEE Computer Society Digital Library. All of the original articles mentioned in this special column are free to view at www.computer.org/computer-magazine/from-the-archives-computers-legacy.

REFERENCES

1. D.A. Patterson and C.H. Sequin, "A VLSI RISC," *Computer*, vol. 15, no. 9, 1982, pp. 8–21.
2. M. Wilkes, W. Renwick, and D. Wheeler, "The Design of the Control Unit of an Electronic Digital Computer," *Proc. IEE–Part B: Radio and Electronic Eng.*, vol. 105, no. 20, 1958, pp. 121–128.
3. J.S. Emer and D.W. Clark, "A Characterization of Processor Performance in the VAX-11/780," *Proc. 11th Ann. Int'l Symp. Computer Architecture* (ISCA 84), 1984, pp. 301–310.
4. D.A. Patterson and D.R. Ditzel, "The Case for the Reduced Instruction Set Computer," *ACM SIGARCH Computer Architecture News*, vol. 8, no. 6, 1980, pp. 25–33.
5. D. Bhandarkar and D.W. Clark, "Performance from Architecture: Comparing a RISC and a CISC with Similar Hardware Organization," *Proc. 4th Int'l Conf. Architectural Support for Programming Languages and Operating Systems* (ASPLOS IV), 1991, pp. 310–319.
6. S. Furber, "Microprocessors: The Engines of the Digital Age," *Proc. Royal Society A*, vol. 473, no. 2199, 2017; rspa.royalsocietypublishing.org/content/royprsa/473/2199/20160893.full.pdf.
7. V. Reddi, "A Decade of Mobile Computing," *Computer Architecture Today*, 21 Jul., 2017; www.sigarch.org/a-decade-of-mobile-computing.
8. D. Patterson and A. Waterman, *The RISC-V Reader: An Open Architecture Atlas*, Strawberry Canyon, 2017.
9. D. Patterson, "How Close Is RISC-V to RISC-I?," ASPIRE UC Berkeley, 19 Jun., 2017; aspire.eecs.berkeley.edu/2017/06/how-close-is-risc-v-to-risc-i.
10. A. Baumann, "Hardware Is the New Software," *Proc. 16th Workshop Hot Topics in Operating Systems* (HotOS 17), 2017, pp. 132–137.
11. A. Waterman et al., *The RISC-V Instruction Set Manual, Volume I: User-Level ISA, Version 2.0*, tech. report, Electrical Eng. and Computer Sciences Dept., Univ. of California, Berkeley, 2014.
12. N.P. Jouppi et al., "In-datacenter Performance Analysis of a Tensor Processing Unit," *Proc. 44th Ann. Int'l Symp. Computer Architecture* (ISCA 17), 2017, pp. 1–12.

DAVID PATTERSON is the co-author of *Computer Architecture: A Quantitative Approach*. He is a retired professor with 40 years of experience teaching computer science at the University of California, Berkeley. He now serves as a distinguished engineer for Google Brain and is Vice Chair of the Board of Directors of the RISC-V Foundation. Patterson is a member of the US National Academy of Engineering and the National Academy of Sciences. Contact him at pattrsn@cs.berkeley.edu.