This is accepted version, as of 08/23/21, very close to final https://ieeexplore.ieee.org/document/9623447
Published in: IEEE Micro (Volume: 41, Issue: 6, Nov.-Dec. 1 2021) Page(s): 131 – 139.
In this version, any phrase in *Italics* names a Wikipedia page: go to https://en.wikipedia.org and search.

Interactions, Impacts and Coincidences of the First Golden Age of Computer Architecture

John R. Mashey *Techviser, Portola Valley, CA 94028*

In their 2018 Turing Award lecture and 2019 paper John Hennessy and David Patterson reviewed computer architecture progress since the 1960s. They projected a second golden age akin to the first, approximately 1986-1996, when new Instruction Set Architectures (ISAs), almost all Reduced Instruction Set Computers (RISCs) revolutionized the industry, eliminated most minicomputer vendors, rivaled mainframes and began takeover of supercomputing. The C language and derivatives came to pervade systems programming, while Unix derivatives came to run many servers, desktops and smartphones. Such outcomes were not inevitable, but depended on evolutionary interactions of computer architecture and languages, industry dynamics and sometimes random coincidences.

Hennessy and Patterson reviewed progress of computer architecture since the 1960s [1]. There is no need to repeat two leaders' well-known history or details from Wikipedia pages reviewed recently, shown in Italics on first use, for example, *Reduced instruction set computer*.

Instead, this paper highlights sometimes-subtle technical interactions among languages, compilers, benchmarks, quantitative design, ISAs, caches, operating systems and nontechnical issues such as industry structure, company politics, university connections, alliances, conferences, and even odd coincidences with large impacts.

Unix and C

Unix and *C (programming language)* success partly resulted from their creation at *Bell Labs* (BTL), whose university-style research had a huge internal *Bell System* market on other companies computers while AT&T was not allowed to sell computers.

Unix and C only existed due to an unusually-persistent BTL recruiter's pursuit of a reluctant *Ken Thompson*. Unix and C might have evolved differently if implemented on preferred *Digital Equipment Corporation* (DEC) *PDP-10* rather than *PDP-11*. C's success was hardly guaranteed. *IBM* might have released *PL/8*, DEC might have promoted *BLISS* more widely or some *Pascal (programming language)* variant might have taken C's niche, as per *Comparison of Pascal and C*.

Languages and benchmarks

Languages drive ISA design. C worked best on CPUs with *Byte addressing*, 32-bit *Word (computer architecture)* and later, 64-bit CPUs with full sets of size-specific memory accesses. At least 3 RISCs (Stanford MIPS, *AMD Am29000*, and *DEC Alpha*) initially omitted 8- and 16-bit loads/stores. That choice simplified hardware and had long history in early *Word addressing* CPUs. C usage patterns drove designers to add 8- and 16-bit loads/stores.

Quantitative methods increasingly replaced intuition, but made good benchmarks even more crucial. A coincidental 1988 email and resultant meeting of 4 competitors in a bar led to the impactful creation of the *Standard Performance Evaluation Corporation* (SPEC).

Installed bases and internal politics

Computing companies are constrained by past successes and installed bases, as happened here with the leading mainframe and minicomputer companies, IBM and DEC. Technical excellence often ran afoul of internal battles.

IBM pioneered RISC in the mid-1970s. *John Cocke*, Marty Hopkins and others kept pushing for it. Their optimizing *PL/8* compiler even targeted several ISAs, but IBM's priority was the *IBM System/360* family. Effective IBM RISC systems were delayed until 1990's *IBM RS/6000*.

DEC repeatedly started and stopped RISC projects from the early 1980s onward, but its focus was the highly-profitable *VAX*. Only in 1992 did its own Alpha ship [8].

Hewlett-Packard (HP) seemed to manage transitions from *HP 2100*, *HP 3000*, and *Motorola 68000* to *PA-RISC* more smoothly. Most other minicomputer vendors struggled and then disappeared.

Technical attributes of oft-mentioned CPUs

Table 1 summarizes key integer/addressing issues of some relevant CPUs. Bits/register are underscored for 32-bit CPUs later extended straightforwardly to 64/32-bit.

The # of registers accessible at one time are underscored for those with the *register window feature*.

Load/store sizes show number of bytes transferred, underscored for load/store multiple. Aligned (A) specifies integral alignment as in S/360, but not S/370 and others.

Most system RISCs use 4-byte instructions. Some added 2-byte operations for embedded market, where ARM got first real success.

Some CISC CPUs do Memory-memory operations, maximum size as shown.

As per *Addressing mode*, the PDP-10 had multilevel indirect addressing (M). The PDP-11 and VAX supported one level (1), added by Motorola in MC68020.

CPUs here illustrate 3 key lessons. First designs are driven above all by data types. A CPU must provide fast parallel hardware for its market's frequently-used data types. Hardware floating point is much faster than software emulation via integer instructions. Many ISAs have been extended to handle multimedia data. However, sequences of simple instructions have often proved as fast as complex function call instructions.

Second, complex instruction decoding, as in Intel i386, seems to be less troublesome for fast pipelining than multiple and/or indirect address mechanisms, as found in MC68020 and especially the VAX.

Third, some wrongly thought RISC meant reduced number of instructions, but it really meant reduced complexity of instruction decode, easier pipelining and elimination of features rarely used by selected compilers.

Table 1 – Key integer attributes, CISCs and RISCs

System	Bits/register	# of Registers.	Word/byte address	Load/store bytes, multiple registers	Alignment integral	Instruction bytes	Memory-memory, maximum # bytes	Indirect addressing
PDP-10	36	16	W	36,18b	w	36b		M
S/360	32	16	b	1,2,4	A	2,4,6	256	
S/360 44	32	16	b	1,2,4	A	2,4		
S/370	**32**	16	b	1,2,4		2,4,6	16MB	
XDS Sigma 5	32	16	W	1,2,4	A	4		1
PDP-11	16	8	b	1,2		2,4	2	1
VAX	32	16	b	1,2,4		1-56	2GB	1
Intel 8086	16	8	b	1,2		1-6	2	
Intel i386-	**32**	8	b	1,2,4		1-15	4	
MC68000	32	8,8	b	1,2,4	2	2-10	4	
MC68020…	32	8,8	b	1,2,4		2-22	4	1
UCB RISC II	32	32	b	1,2,4	A	4		
SPARC	**32**	32	b	1,2,4	A	4		
Stanford MIPS	32	16	W		w	4		
MIPS R2000-	**32**	32	b	1,2,4	A	4		
microMIPS	32	32	b	1,2,4	A	2,4		
HP PA RISC	**32**	32	b	1,2,4	A	4		
Am29000A	32	32	b	4	A	4		
Am29000C	32	32	b	1,2,4	A	4		
Alpha 1992	64	32	b	4,8	A	4		
Alpha 1996	64	32	b	1,2,4,8	A	4		
PowerPC	**32**	32	b	1,2,4		4		
ARM T32	32	16	b	1,2,4,8		2,4		
ARM A32	32	16	b	1,2,4,8		4		
ARM A64	64	32	b	1,2,4,8		4		

PROLOG - BEFORE THE GOLDEN AGE

Early 1960s mainframes

In early 1960s, *COBOL*-oriented commercial systems, such as *IBM 7080*, *IBM 1401* and *Honeywell 200* used variable character strings and addressing.

Technical mainframes were programmed in *Fortran*, using word addressing of various sizes, such as 48-bits in the upper *CDC 3000 series*, or *CDC 6600's* 60-bits. More popular were 36-bit CPUs, which often included instructions for extracting and inserting packed (usually 6-bit) characters. Such included the *IBM 7090, IBM 7040, Univac 1100/2200 series, GE 600 series,* and *Digital Equipment Corporation* (DEC) *PDP-10.*

Although many wanted a higher-level *System programming language*, operating systems were usually written in assembly language for efficiency. Any such language was strongly influenced by the computer architecture(s) expected to use it. Unusually, *Burroughs large systems* were programmed in *ALGOL 60* variants, Fortran and COBOL.

IBM System/360 and consequences

The *IBM System/360* replaced most of IBM's existing computers with a single unified computer family using 8-bit *Byte addressing*. Operations on 4-byte integer registers resembled those of earlier *word-addressing* computers, but added loads/stores of bytes and 2-byte halfwords, now first-class addressed data, all aligned on integral multiples of their size. The S/360 also executed varying-length memory-to-memory operations for commercial applications. The design was driven by need to support both Fortran and COBOL, plus (unfulfilled) hope that the new *PL/I* would eventually supplant both.

Hennessy and Patterson [1] noted that most models used *Microcode*, whose elimination was the focus of RISC. Three models were hardwired, including the *IBM System/360 Model 44,* which removed memory-to-memory and decimal instructions, keeping the simpler instructions akin to RISCs'. *Gordon Bell* and *Allen Newell*'s classic 1971 book [2] showed it had unusually good cost/performance compared to the microcoded models. IBM's *PL/8* compiler used a similar subset, in co-evolution with the seminal 1970s RISC *IBM 801*..

The *IBM System/370* (1970*)* added more virtual-memory models and relaxed the integral data-alignment rules, as did the later DEC VAX and many CISC microprocessors. The C language specified strict alignment and early RISC designers were happy to oblige, as it avoided crossing cache line or page boundaries. In effect, they stuck with 360/44.

Time-sharing, IBM and DEC

IBM mainframes were originally intended for *batch processing*, but by late 1960s, leading-edge groups wanted *time-sharing* operating systems, as in *Multics* on *GE 645* or *TENEX (operating system)* on PDP-10. IBM was working on its own *TSS (operating system)*, but it was late and slow, leading to creation of the *Michigan Terminal System* for *IBM System/360 Model 67*, the first model with *Memory paging*. This was followed by IBM's *CP-67*, *CP/CMS*, and finally TSO (*Time Sharing Option*).

DEC *PDP-10*s and later *DECSYSTEM-10* and -20 were designed for *time-sharing* and were strongly preferred by many leading electrical engineering / computer science departments. For example, the April 1971 *ARPANET* map showed 15 sites. Just 5 included IBM systems, but 8 had *PDP-10*s: SRI, Utah, MIT, Case, Carnegie, Harvard, BBN, Stanford. At Stanford, these remained in wide use into the mid-1980s. Coincidentally in 1971, the *Microprocessor chronology* started with the Intel 4004.

Coincidence, minicomputers, C, Unix

The BTL Computing Research department worked with MIT and General Electric on *Multics* and in 1966 hired *Ken Thompson* from UC Berkeley, after some real hesitation. Unix and C almost never happened. As per his 2005 *Computer History Museum* oral history, his teachers recommended him to a BTL recruiter, who he ignored for a week, but who called and visited him at home. BTL invited him to interview in New Jersey. He replied that he was not interested in a job but would take interviews to visit friends on East Coast, which BTL said was fine(!). He got an offer and after a few weeks' thought, finally accepted. He kept close ties with Berkeley, spent sabbaticals there, helped it become the leading university for Unix work, but on PDP-11 and VAX, not PDP-10.

After BTL halted Multics work, Thompson and *Dennis Ritchie* wrote the first Unix on an 18-bit *PDP-7*. The department wanted a *PDP-10*, but only had budget for a 16-bit *PDP-11*/20, limited to $1/20^{th}$ the memory, only 56KB shared by kernel and user program. It was first DEC CPU to switch from (12, 18 or 36)-bit words to 8-bit byte-addressing, driven by IBM S/360 dominance. C evolved from *BCPL* via *B (programming language)*, typeless languages well-suited to word-addressed CPUs, not so well matched to PDP-11s.

Software was then often cross-compiled on a large system for downloading to a smaller minicomputer, but Unix was self-supporting on the PDP-11. The C compiler had to be kept small, so BTL did not write global optimizers like those of mainframe-based PL/8 or BLISS. C first appeared in *Research Unix* 2nd Edition (June 1972.

By 1973, BTL had about 20 such systems, but Computing Research had acquired a much more capable *PDP-11*/45, with segmented memory management, 64KB Instruction and 64KB Data memories per user process. They had rewritten the kernel from assembly into C.

In October 1973 the Programmer's Workbench (*PWB/UNIX)* department got a *PDP-11*/45, the second at BTL, with 248KB memory, to support 16 interactive users. We ran the first, and for years, the largest, real Unix computer center, eventually supporting about 1000-programmers building software *for IBM System/370*, (XDS) SDS *series* 5 and 36-bit *Univac 1100/2200 series computers*. By 1976, *PDP-11* Model 70s with 1MB memory supported as many as 48 users, demanding great attention to memory usage and performance tuning.

Ritchie's *PDP-11* C compiler was ported to the S/370 and other architectures, in some cases easily, in other cases painfully. It could be made to work on word-addressing CPUs, but fit byte-addressed CPUs better. People noticed that C had little use for many ISA's complex instructions.

BLISS and Pascal were considered for BTL projects, the latter especially for tighter type-checking. P. J. Plauger wrote the *PL360*-like Little Implementation Language LIL, but C kept improving and LIL never caught on. Many suggested or even implemented changes to their copies of the C compiler. Ritchie certainly listened to suggestions, but he retained control over mainstream C, unlike Pascal, which spawned many variations to make a teaching language more useful in production. *Apollo Computer* wrote its fine Aegis operating system in Pascal, proving it possible, but it was a proprietary version not spread around schools.

Given an earlier consent decree, AT&T could not sell computers externally. *Research Unix* Editions V5 and V6 had been licensed very cheaply on a don't-call-us basis to schools and later, to companies, not so cheaply. This made AT&T lawyers nervous, but the school-licensing had huge impacts, especially via UC Berkeley's enthusiastic support. The lawyers disliked *Lions' Commentary on UNIX 6th Edition, with Source Code* (1976), but it was widely copied.

As *VAX*-11/780 and other 32-bit superminicomputers appeared, Unix evolved to the far more portable Unix V7 (1979). It included *Stephen C. Johnson*'s *Portable C Compiler* (PCC), designed for easy retargeting to different ISAs. The software was just in time for *Onyx Systems*' *Zilog Z8000*-based systems and others based on the *NS320*00 or *Motorola 68000*. Unix and C were portable, acceptable fits for the ISAs, PCC was not too big or slow to be unworkable on small systems and many systems programmers had learned Unix and C in school.

DEC had invested heavily in a good operating system, BLISS-based *VAX/VMS*, only to find many schools preferred Unix., so DEC eventually released its own *Ultrix* in 1984. Key DEC BLISS developer Ronald Brender years later wrote [3], "Over time management became painfully aware that most new employees already knew C but few knew anything about BLISS, which meant more lengthy training and integration periods."

THE FIRST GOLDEN AGE

Coincidence and Stanford MIPS

At UC Berkeley and Stanford, research groups led by Patterson and Hennessy were laying the research groundwork for expansion of RISC into the broader market. *Sun Microsystem*'s *SPARC* was a closely-related commercial adaptation of Berkeley RISC [4].

Stanford MIPS evolved more radically, as per **Table 1**, partially by coincidence. Entrepreneur/marketeer *Steve Blank* had worked at *Convergent Technologies*, but by Fall 1984 was consulting for the founders of *MIPS Computer Systems*. As sanity check, he brought them to talk to Convergent people he knew, an impactful coincidence, at least personally and perhaps for MIPS.

The MIPS folks presented Stanford MIPS CPU [5] [6], and optimizing compilers [7] and asked our opinions. I said I had followed RISC at BTL, had reviewed proposals favorably there, thought it was the right direction, but that their specific design had some problems for C and Unix. A bit surprised, they said they were really doing a commercial-grade CPU inspired by Stanford MIPS, but not identical. They invited me to visit and review draft specifications on stimulating visits that consumed most Saturdays for a few months. Given the rare opportunity for a software person to influence an ISA design I joined MIPS in January 1985.

Quantitative design needs good benchmarks

RISC designers at IBM, HP, DEC, Stanford and UC Berkeley focused on quantitative methods. All analyzed performance of compiled high-level language benchmarks, but designs could differ according to the choice of benchmarks, relative importance of languages and compiler technology. Unix-focused UC Berkeley used PCC and included *Register window*s to speed C function calls. IBM, HP and Stanford had aggressive global optimizing compiler technology that made windows seem less useful. Given its installed base of *HP 3000* customers, HP included a few instructions to help decimal arithmetic for COBOL. Stanford's compilers (C, Fortran, Pascal) were written in Pascal, running on DECSYSTEM-20 and Pascal benchmarks were used more than C for driving ISA design.

University research efforts in microprocessor design inherently differ from creation of commercial-grade chips. The former should explore interesting ideas and hopefully produce test chips, but need not implement all features of already well-understood features of commercial chips, so do not advance research. However, sometimes extensive relevant industrial experience never gets published, but matters greatly, as illustrated next.

Hardware/software interactions at MIPS

Word addressing. Stanford MIPS had 16 registers and 32-bit word-addressing, 32-bit loads/stores, plus some support for 8-bit byte data in packed arrays [6] p.15, somewhat akin to the PDP-10 or 32-bit SDS/XDS Sigma 5 (*SDS Sigma Series*). Word address bits were shifted left 2 bits, freeing them to specify the byte. The same bit pattern might access different words depending on how it was interpreted. For instance, the binary bit pattern 101 could mean Word 5 or Byte 1 of Word 1. Pascal's tight type checking might catch that, but C programmers sometimes passed int pointers to functions calling them char pointers, not usually caught by compilers. On byte-addressed CPUs, there is no ambiguity.

The Stanford designers were fine architects (I've worked with many) who argued very strongly for this approach, based primarily on statistics of integer Pascal programs, which showed low percentages of 8-bit accesses [6] pp.15-16. They also expressed little use for 16-bit integers, unsurprising given nature of Pascal, in which halfwords were not first-class data as in C.

I strongly urged Hennessy that we implement the more usual 8-, 16, and 32-bit load/store operations, ideally with both zero- and sign-extension loads, given personal compiler experience with *Motorola 68010*, which often required use of sign-extension instructions.

Four problems were worrisome. The first two were software challenges from firsthand experience 1973-1984 moving Unix code around PDP-11, VAX, *ATT 3B series computer*s, MC68000, 68010, 68020 and others secondhand. The third degraded performance and code size, but the fourth was a serious structural issue for restricted 32-bit load/stores.

C on word-addressed machines considered harmful. BTL had ported C to many ISAs, including word-addressed Univac 1110, GE/Honeywell 645, 32-bit SDS/XDS Sigma 5, *HP2100*, *Cray-1*. Compiler and library writers recorded serious challenges via internal BTL memos rarely published outside. I heard similar stories from others at *USENIX* conferences. It was possible, just painful.

Applications software not quite portable. From long experience, applications writers sometimes assumed all pointers, regardless of data type, to be simple memory addresses, given long history on PDP-11, VAX, IBM S/370, MC68K. While such code might be unclean, in practice it would likely work on byte-addressed RISCs, but break oddly on Stanford MIPS, catastrophic for a small company begging ports from third-party vendors.

Shorts in Unix kernel. Pascal usage differed from C, which had supported 16-bit **short**s as first-class data since 1974, when **short** and **long** had been added for the Sigma 5 C project. Unix kernel memory was precious, so its code often used 8- and 16-bit data in packed structures, not strings. I searched our kernel source code for **short** and got 1000s of hits. A single load and especially store on most computers needed multi-instruction MIPS sequences.

Memory-mapped I/O device registers. These often packed 8- and 16-bit data together, easily described by simple C structures. The CPU had to issue byte address plus length field (1,2,4 bytes). Reading/writing 32-bit works either did not work or could cause unwanted side-effects. Off-the-shelf device drivers would need major and somewhat unnatural rewrites, as others found later.

I said if we wanted to get software, we needed byte-addressing and a full set of loads/stores. Thankfully, Hennessy listened.

Later confirmation. The *Advanced Micro Devices* (AMD) *AMD Am29000* (1988) and *DEC Alpha* (1992) had byte addressing, but provided no 8- or 16-bit loads/stores, instead using load word+extract byte/half, and load word+insert byte/half, store word. Byte addressing solved the first two problems above. The third remained as a performance and code size issue. The fourth was painful. AMD quickly realized the problem and the Am29000 Rev C (by 1990) added byte/halfword loads/stores, as did the Alpha 21164A (1996), as per **Table 1**.

Optimizers and caches versus kernel

Unlike PCC, Stanford-derived MIPS compilers supported aggressive global register allocation, code motion and elimination of redundant operations. This could cause trouble for drivers when they accessed memory-mapped I/O devices, where status registers and counters changed externally and any load or store might cause side effects. For example, code might load the value of device register inside a loop and test it. The optimizer would hoist the test outside the loop and execute the test once. Luckily, in 1985, the C **volatile** keyword was coming into use. It took experience to get the right effect, that **volatile** accesses had to act exactly as if generated by a simpler non-optimizing compiler.

I/O device register structures not only had to be declared **volatile**, to avoid irregular bugs they had to use uncached memory locations, for MIPS determined by memory mapping. In mid-1986, a tape device usually worked in a busy system, but failed in unloaded one, opposite the usual pattern. For a month we blamed hardware, but the device structure accidentally had been mapped as cached. When a status register was read, in an empty system it sometimes got a cached value, but in a busy system, the resulting cache miss would deliver (desired) current status.

Global optimizers sometimes had bugs and in 1986, likely only HP and IBM were compiling Unix with similar optimization levels.. At one point, our kernel crashed only if compiled with high optimization. We had to binary search of kernel functions, compiling half with optimization, half without, eventually finding the optimizer had wrongly eliminated a single store. It is exciting to debug the combination of new ISA's first implementation, new system hardware, young Unix port and aggressive optimizing compilers. Bugs might be anywhere.

Industry structure and coincidence in 1988.

By mid-1980s, microprocessors were designed, manufactured and sold by semiconductor companies such as Intel or Motorola, or else designed and manufactured by large companies such as IBM, HP, DEC or AT&T for use in their own computers. The latter usually provided key compilers and operating systems, the former often contracted with outside software companies.

That changed somewhat around 1986, as both MIPS and Sun Microsystems designed RISCs, and licensed the designs to chip companies to sell to others as well. Sun encouraged many competing designs and was usually the biggest customer, while MIPS chips were sold to *Silicon Graphics* (SGI), DEC, *Tandem Computers*, Cisco and many others. MIPS and Sun were somewhat akin to modern fabless companies, but independent foundries were only just starting. That now well-established model might have made life simpler. Alliance and ecosystems issues often outweighed technical design and many tricky negotiations occurred. One might have to present future plans to a company that might become a customer, a partner or a competitor, as in the MIPS-DEC relationship.

DEC had excellent architects working on various RISC designs (*DEC_PRISM*), but were often diverted to work on improving VAX. By coincidence, an old BTL colleague then at DEC bumped into me at a January 1988 computer show. He asked about MIPS, so I gave him a MIPS Performance Brief that showed MIPS R2000s faster than any VAX. At the evening beer bust, after a few beers he asked if I thought it practical to port DEC's *Ultrix* to MIPS, saying he wanted to show DEC CEO *Ken Olsen* it would not take years and hundreds of people. I explained why it was plausible. Although it seemed unlikely DEC would use an outside RISC CPU, after months with lawyers, we lent him 2 systems. In less than 3 weeks, he and several others did a solid Ultrix port, a crucial step in DEC's decision to build MIPS-based workstations and servers.

DEC also partnered with MIPS, Microsoft and others in the *Advanced Computing Environment* (ACE) consortium. *Dave Cutler*, who had designed DEC's *Open VMS*, had worked on DEC RISC chips canceled when DEC adopted MIPS, but joined Microsoft to architect *Windows NT* as portable beyond Intel CPUs. Microsoft wished to support alternatives to Intel, just as Intel encouraged alternatives to Microsoft operating systems.

But finally, DEC announced its own DEC Alpha RISC in 1992, so became a competitor again [8].

1989 - Hennessy and Patterson books

Hennessy and Patterson [9] was published in 1989 and rapidly became the standard upper-level textbook, using MIPS ISA for examples until *RISC-V* in 2017. They somehow managed extensive revisions, 2 (1996), 3 (2003), 4 (2007), 5 (2011), and 6 (2017). It would be hard to overestimate the impact of these books on the practice and education of computer architecture.

1989 Influential chip conferences

Two conferences were quite influential, widely attended by key designers, potential partners, customers and employees, with much time for informal conversation and sometimes deals.

Hot Chips. *Hot Chips* started in 1989, continues to this day, organized with rare continuity over 30+ years. It always combined academe and industry, starting with one Program Co-Chair from each. People submit abstracts and if chosen, need only provide presentations, to allow for the most recent ("hot") work to be presented by lead engineers, who rarely had time to write full papers.

The Hot Chips archive [10] is a valuable resource for the history of microprocessors and related chips, starting with the first, which included SPARC, MIPS, *Motorola 88000* and 68040, Intergraph *Clipper architecture*, Intel i486, i860, i960 and more. At the end of a conference filled with glowing chip presentations by hardware designers, the last session collected independent compiler writers to talk about their experiences, not always positive. PCC author Steve Johnson said compilers almost always lagged hardware design. He mentioned exceptions of CPUs designed around compiler technology: Berkeley RISC:PCC, Stanford: Optimizing Pascal, and IBM 801: PL/8, also high-optimizing.

Microprocessor Forum. MicroDesign Resources (MDR) was founded in 1987 and published the widely-read commercial newsletter *Microprocessor Report*. MDR held its Microprocessors '89 conference in November 1988, renamed Microprocessor Forum the next year, focused on industrial products. A coincidental encounter at one with Microsoft people helped creation of the ACE consortium.

After some ownership changes, Microprocessor Report is now published by The Linley Group, which has run similar conferences since 2005.

1989 - SPEC foxes guard the henhouse

In the 1980s, companies were using quantitative design methods, but widely-cited small benchmarks, such as *Whetstone (benchmark)* and *Dhrystone* were less useful [9] pp.45-49. Optimizing compilers could legitimately eliminate some code and some people outright cheated with special-case optimizations, [9] pp.70-79. Vendor "MIPS-ratings" were quite inconsistent and customers disbelieved them.

The correct way to summarize performance of sets of benchmarks was endlessly argued [9] pp.49-53. Magazines ran their own benchmarks. Computer vendors published Performance Briefs, often providing data on more realistic, but different programs, hard to compare. Engineers who wrote these usually knew each other, traded documents, asked for updated numbers for fair comparisons. As in other areas of the industry, sometimes fierce competitors also cooperated and of course people moved around.

SPEC started somewhat by coincidence. Stan Baker wrote often on microprocessors in the widely-read *EE Times* magazine. In Fall 1988, he listed performance ratings in "Dhrystone-MIPS," performance relative to a VAX-11/780. I emailed him to complain and think someone else did, too. He challenged me: if we industry people thought it was so bad, why did we not create something better? He offered his bar in Campbell, CA as a neutral meeting place and would provide beer. Soon after, one representative from MIPS, HP, Sun and Apollo met there. We quickly agreed we would rather compete over realistic benchmarks, based on real programs, where improvements would deliver value to customers. We agreed that current practices wasted time and that if we could collect meaningful benchmarks and credible rules for running and reporting them, it would improve the industry. We identified a few programs we all used, but with different versions or inputs.

SPEC was founded in November 1988 [11], IBM, DEC and other computer companies quickly joined. Baker agreed to be (a neutral) President.

We spent the next year evaluating candidate programs, which had to be representative of real customer code, reasonably portable, distributable in source code, and have checkable output. This turned out to be harder than we thought. Engineers held "bench-a-thons" where we brought workstations together, moved programs around, helped each other debug incompatibilities, and created tests to make sure programs ran correctly. For example, floating point on VAX, MC68000 and the RISCs differed slightly, so we developed "fuzzy compares."

We settled on 4 integer and 6 floating point benchmarks officially announced in late 1989. As per [9] p.48, we erred on matrix300, which was too small and easy to optimize in legitimate, but unrepresentative ways. We adopted practice from some Performance Briefs of computing ratios of performance versus VAX-11/780, always showing all data, but using *Geometric mean* (GM) to compute a single SPECmark.. We preferred separate GMs for integer and floating point, but many in the industry told us we really needed one number if we wanted rapid acceptance. Vendors started reporting both as well and SPEC officially split them by 1992. Benchmark sets evolved over time to remove those found obsolete and add better ones.

In the absence of workload weights, the correct way to average ratios is the GM, not the *Arithmetic Mean* (AM). Suppose system A is 2X faster than system B on benchmark 1 and B 2X faster than A on benchmark 2. If A/B ratios are used, setting B to 1.0, then A's ratios are 2.0 and 0.5, whose AM = 1.25, so A seems faster than B. But switching to B/A, B seems 1.25X faster than A. The GM correctly yields 1.0.

Table 2 GM is the right mean for ratios

Sys	Benchmarks		Relative Perf			
	1 Perf	2 Perf	1 A/B	2 A/B	AM	GM
A	2	1	2.0	0.5	1.25	1.0
B	1	2	1.0	1.0	1.0	1.0

But in 1989, we missed that for a set of ratios x_i, the GM formula on left is equivalent to that on the right, a standard statistical transformation often used elsewhere in science.

$$GM = \overline{x_G} = \left(\prod_{i=1}^{n} x_i \right)^{\left(\frac{1}{n}\right)} = \exp\left(\frac{1}{n} \sum_{i=1}^{n} \ln(x_i) \right)$$

This suggested that our benchmark ratio sets could be treated as samples from a larger population, so we could apply standard statistical analyses, compute standard deviations and other metrics. If the set of logarithms were tested and found to be normally distributed (a *lognormal distribution)*, all its great properties became usable. This turned a sometimes-mysterious GM into standard statistics.

I did not notice this until 2004 [12], discussed it in more depth in a Stanford talk and elsewhere years[13], and after many email discussions it got into [9] 4th Edition, pp.33-37.

When we announced SPEC89, we were asked "But this sounds like foxes guarding the henhouse, how can we trust you?" We said no one is better than foxes at stopping other foxes from eating the hens.

SPEC cooperation emerged from the golden age's intense competition. It also engaged university researchers who studied and critiqued each benchmark iteration SPEC started with a meeting in a bar, continues to this day and has helped inspire other benchmarking efforts. It has had strong impacts on the design and tuning of systems for 30 years.

64-Bit and speculative execution

The MIPS *R4000* was an early 64/32-bit microprocessor, shipped in SGI systems in early 1992, followed later that year by the 64-bit DEC Alpha. Over the next few years other ISAs evolved to 64/32-bit models and C evolved as well, not without toil and trouble[14]. By 1996, speculative, out-of-order designs were appearing in the market.

EPILOG - RISC-V, THE ARM COINCIDENCE

The *History of free and open-source* software is long, including user groups such as IBM *SHARE (computing)* (1955) and DEC *DECUS* (1961-). *RISC-V* is an interesting open-source hardware effort, whose ISA has many commonalities with other RISCs [15] p.2. RISC ISAs were more alike than not, CISC ISAs varied greatly and many ended or dwindled in the golden age. As Tolstoy wrote in *Anna Karenina*, "All happy families are alike; each unhappy family is unhappy in its own way."

The *ARM architecture* had not prospered in the workstation market, but *Dave Jaggar* drove its 1992-2000 evolution towards lower cost and power for embedded systems. It was a winner for mobile phones and the big winner for smartphones. In a May 29, 2019 talk for Stanford's SystemX Alliance, Jaggar revealed the ironic coincidence behind this success. At the University of Canterbury in Christchurch, NZ, in 1989, he was inspired to pursue architecture more deeply by a MIPS R3000 lecture, as he noted in his 2012 Computer History Museum oral history. He got convinced that ARM could never compete with MIPS performance. When he joined ARM in 1991, he drove it towards low-power embedded applications. As I had been that speaker in NZ, at Stanford he gave me a bottle of Hennessy cognac, noting lack of a Patterson equivalent.

CONCLUSION

Computers were once designed for Fortran, COBOL and all too often for assembly language. Over time, quantitative analyses of compiled code prevailed. But such analyses depend heavily on studying truly representative sets of benchmarks. By 1989, intense competition had generated much technical progress, but also the first of a famous series of textbooks, microprocessor conferences and the SPEC benchmarking group, all continuing today.

RISC ISAs and RISCy implementations of the remaining CISCs co-evolved especially with Unix, C on byte-addressed/accessed 32 and 64-bit CPUs. Computing today owes much to that first golden age and some odd coincidences that shaped both ISAs and industry.

ACKNOWLEDGMENT

I have been lucky to work with and know many others from whom I could learn, too many to list here, but especially John Hennessy and Dave Patterson.

REFERENCES

1. J. L. Hennessy, D. A. Patterson, "A new golden age for computer architecture," *Comm. ACM,* vol.62, no. 2, pp.48-60, February 2019, doi:10.1145/3282307. https://cacm.acm.org/magazines/2019/2/234352-a-new-golden-age-for-computer-architecture/fulltext
2. C. G. Bell and A. Newell, *Computer Structures: Readings and Examples*, McGraw Hill. New York, 1971, p.569, pp.584-585.
3. R. F. Brender, ""The BLISS programming language: a history," *Software: Practice and Experience*. vol 32, no 10, pp. 955–981. doi:10.1002/spe.470. https://www.cs.tufts.edu/~nr/cs257/archive/ronald-brender/bliss.pdf
4. D. A. Patterson, C. H. Sequin, "A VLSI RISC," *IEEE Computer* vol. 15, pp. 8-21, Sept 1982. doi: 10.1109/MC.1982.1654133 https://www.computer.org/csdl/magazine/co/1982/09/01654133/13rRUwgyOfJ
5. S. A. Przybylski, T. R. Gross, J. L. Hennessy, N. Jouppi, C. Rowen, "Organization and VLSI implementation of MIPS," Stanford, Technical Report CSL-TR-84-259, April 1984. http://i.stanford.edu/pub/cstr/reports/csl/tr/84/259/CSL-TR-84-259.pdf
6 J. Hennessy, N. Jouppi, F. Baskett, T. Gross, J. Gill, S. Przybylski, "Hardware/software tradeoffs for increased performance,", Stanford, Technical Report 22.8, Feb 1983. http://i.stanford.edu/pub/cstr/reports/csl/tr/81/228/CSL-TR-81-228.pdf
7. F. C. Chow, A portable machine-independent global optimizer - design and measurements, Ph D dissertation, Stanford, January 1983, pp,1-187. https://www.researchgate.net/publication/213890050_A_portable_machine-independent_global_optimizer_-_design_and_measurements
8. G. Bell and W.D. Strecker, "What have we learned from the PDP-11 – what have we learned from VAX and Alpha", Proceedings ISCA '98: 2, pp.6-10, August 1998. https://doi.org/10.1145/285930.285934 https://gordonbell.azurewebsites.net/Digital/Strecker%20Bell%20PDP-11%20VAX%20Alpha%20Retrospective.pdf
9. J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*. Morgan Kauffman, San Francisco, CA, 1989. 1st Edition.
10. Hot Chips, archives. https://hotchips.org/archives/ Accessed 06/20/21, especially HC01 I(1989).
11. Standard Performance Evaluation Corporation (SPEC), "SPEC: For 30 years, a beacon of truth," https://www.spec.org/30th/ Accessed 06/20/21.
12. J. R. Mashey, "War of the benchmark means: time for a truce," ACM SIGARCH Computer Architecture News vol 32, issue 4, pp.1-14, September 2004. Doi: 10.1145/1040136.1040137
13. .J. R. Mashey, "Summarizing performance is no mean feat," October 30, 2008, talk given 2005-2008 at various places. https://techviser.com/docs/Mashey.nomeanfeat.2008.pdf
14. J. R. Mashey, "The long road to 64 bits," *ACM Queue* vol 4. Issue 8, pp.24-35, October 10, 2006. https://queue.acm.org/detail.cfm?id=1165766
15. T. Chen, D. A. Patterson, "RISC-V Geneology," University of California at Berkeley, Technical Report UCB/EECS-2016-6, January 24, 2016. https://www2.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-6.pdf

John R. Mashey received a 1974 PhD in Computer Science from Penn State, which gave him the first Outstanding Engineering Alumnus Award-Computer Science in 1997. He contributed to the Programmer's Workbench version of Unix at BTL, to Unix software, MIPS RISC architecture and systems designs at MIPS Computer Systems and Silicon Graphics. The USENIX Association gave him the 2012 "Flame" Lifetime Achievement Award. He cofounded SPEC, helped with the Hot Chips Conference for many years, and has been a Trustee of the Computer History Museum since 2001. He is a member of ACM and IEEE Computer Society. Contact him at mash@techviser.com