

"Any clod can have the facts,  
but having opinions is an art."

Charles McCabe,  
San Francisco Chronicle

THE

OPEN CHANNEL

## Response to "Computers, Complexity, and Controversy"

The causal reader of "Computers, Complexity, and Controversy" (September 1985, pp. 8-19) might erroneously conclude that the performance benefits of RISCs stem solely from multiple register sets, or MRSs, rather than from a wide range of benefits from the RISC approach. Furthermore, the authors suggest that architectural metrics—metrics we deny are of proven validity—show that RISCs are not good computers. They also declare that the 432 and MicroVAX 32 do not constitute evidence for the RISC approach, while the data suggest the opposite conclusion. While we welcome a scientific evaluation of the RISC approach, we feel most of the opinions in the article are not supported by evidence or by scientific reasoning.

The quantitative part of the article is the study on MRSs. A careful analysis of MRSs is worthwhile, but the relevance of the benchmarks used in such a study determine the relevance of its results. The authors mistakenly say that the "benchmarks used were the same ones originally used to evaluate RISC I." Table 1 shows the programs compiled, simulated, and published in papers about Berkeley RISCs. (We at Berkeley, by the way, supplied the authors with our compiler and simulator to run these programs.)

**Table 1.**  
RISC programs ordered by size in lines of code.

SIZE (lines)	BENCHMARKS	SUBJECT	PUBLICATION (year)
29095	PCC	C compiler	1984
1608	SED	Text editor	1982
219	MCF #I	Iterative sort	1982
201	Whetstone	Floating point benchmark	1984
177	Puzzle	Bin-packing (pointer)	1982
136	Quicksort	Recursive sort	1981
133	Puzzle	Bin-packing (array)	1981
89	MCF #K	Bit matrix	1982
77	MCF #H	Linked list	1982
61	MCF #E	String search	1982
60	MCF #F	Bit test	1982
16	Towers	Recursive benchmark	1982
13	Ackerman	Recursive benchmark	1982

From this list, the authors chose the two smallest and most recursive programs, Ackerman and Towers of Hanoi, and added Fibonacci, a highly recursive two-line Lisp program. Since MRSs have major impact on procedure calls, their importance can be artificially inflated by programs with unusual procedure call patterns. Why did the authors choose the smallest programs with the largest number of recursive procedure calls to test their hypothesis that the effects of

MRSs are orthogonal to instruction set complexity? We asked this question at the June '84 International Conference on Computer Architecture in Boston, where they reported their results. Their only explanation was that some of the other programs were not used because they did not have enough procedure calls to support their hypothesis.

In concluding their discussion here of MRSs, the authors acknowledge the pathological nature of the programs ("It has not been established, however, that these benchmarks are representative of any computing environment."), yet they claim that the performance of RISC I "has been shown to be attributable to (MRSs)" and that "performance claims for RISCs that do not remove effects due to MRSs are inconclusive at best." Furthermore, these remarks ignore the benefits of RISCs on average instruction execution time, obviously a key to performance. And even if the authors could

**The Open Channel** is exactly what the name implies: a forum for the free exchange of technical ideas. Try to hold your contribution to one page maximum in the final magazine format (about 1000 words).

We'll accept anything (short of libel or obscenity) so long as it's submitted by a member of the Computer Society. If it's really bizarre we may require you to get another member to cosponsor your item.

Send everything to **Jim Haynes, Applied Sciences, UC Santa Cruz, CA 95065.**

apply their study to nonrecursive programs larger than 16 lines, they overlook that both the 801 and MIPS have the same significant benefits in cost and performance—and neither of these designs use MRSs. If, as the authors suggest, MRS is the “real secret” behind RISCs, we wonder why MRS wasn’t included in their list of RISCs characteristics.

The authors report that architecture metrics show that the VAX outperforms RISCs. We believe the emphasis on architecture metrics over implementation issues is in part responsible for a generation of machines with good metrics but poor cost/performance and long development cycles. The measurement of those metrics by assembly language programs exacerbates the gap between abstract metrics and the real implementation, which will be measured by applications that use high-level languages and compilers. The RISC approach advocates a methodology that balances the needs of compilers with the implementation efficiency of the architecture.

The illusion created by measuring architectural metrics in isolation from the implementation is illustrated by the history of the VAX-like Nebula architecture, which has the best MCF metrics. Although the architecture was completed in 1980 and was reported in the February 1981 *Computer* (pp. 35-41), there are no commercial versions of this machine, and even though the Air Force funded this effort as a new standard computer, the government has yet to put out requests for bids to build Nebula machines. In sharp contrast, RISC architectures have been vigorously pursued by both startups and established computer manufacturers. Even though RISCs were not funded to be a standard, the government recently put out requests for bids to build both gallium arsenide and CMOS versions of the MIPS RISC architecture. If the validity of the MCF architecture metrics was widely endorsed, we would expect industry and government to have rushed to build Nebula and ignored RISCs, not vice versa.

The section of the article on the 432 inadvertently provides further support for RISC ideas. The authors start by citing papers that report the 432 is 10 to 20 times slower than the VAX-11/780. They then hypothesize the 432 would be faster if it were redesigned: if it had more pins, a wider bus, and a faster memory system, it would be 25 to 40 percent faster; if it had a less complicated and faster procedure call, it would be 20 percent faster; and if it didn’t have variable bit length instructions, it would be three percent faster. According to their own numbers, the

new hypothetical 432 is still six to 12 times slower than the VAX, and likely three to six times slower than the 16-bit Motorola 68000 microprocessor. By concentrating on microcoding high-level functions, the 432 architects apparently sacrificed the performance of the more frequently occurring simple functions. Contrary to the authors’ statement that “it is wrong to conclude that the 432 supports the general RISC point of view,” we believe that the performance data and the authors’ suggested simplifications—to increase the 432’s performance—both provide evidence indeed for the value of the RISC approach.

---

Contrary to their conclusion, we think that the history of VLSI implementations of VAX supports the RISC approach.

---

Again contrary to their conclusion, we think that the history of VLSI implementations of the VAX, reported in the October 1984 *IEEE Journal of Solid-State Circuits* (pages 663-681), supports the RISC approach. The VLSI VAX is an eight-chip implementation of the full VAX architecture. During the course of the VLSI VAX project, DEC found that 20 percent of the instructions required 60 percent of the microcode, yet were only 0.2 percent of the executed instructions. Several years after the beginning of the project, DEC decided to modify the layout and microcode of the main CPU chip and then subset the architecture to fit the microcode memory and data path into a single chip, thus creating the MicroVAX-32. (The authors do not mention the VLSI VAX and mistakenly say that the MicroVAX-32 “was designed, realized, and tested in a period of several months.” The paper by the MicroVAX designers reports that it took 20 months from the start of the project to produce the first pass of the descriptions of the circuit masks. Chip fabrication, testing, and redesign lengthens the design cycle.) Although this subset project started later, it finished first and is now a product. Furthermore, the performance degradation of the MicroVAX-32 for the omitted instructions is just 4 percent while the fivefold reduction in control store size cuts the active chip area almost in half. DEC’s decision to subset its complex architecture to exploit VLSI—plus the reduced design effort of the simpler machine—is certainly some indirect support for the RISC approach.

The reader should note that this last year has been good for RISCs. John Cocke of IBM won the ACM-IEEE Eckert-Mauchly Award for Computer Architecture in part for his work on RISCs, and Manolis Katevenis’ dissertation on VLSI RISCs won the ACM Doctoral Dissertation Award. Gordon Bell, another winner of the Eckert-Mauchly award and one of the designers of the PDP-11 and the VAX, cited the Berkeley and Stanford RISC research on VLSI computers last October in this magazine (pp. 14-30) and said that given “the current speed of logic relative to memory, it is time again to return to direct (versus microprogrammed) execution of the instruction set.”

Without identifying a specific publication or quotation, the authors of “Computers, Complexity, and Controversy” claim that some RISC publications are “oversimplified, overstated, and misleading.” Fortunately, disagreements in computer design are not resolved on the pages of a magazine; hardware is the final judge of architecture. Many computer scientists and commercial computer designers have been convinced by the research presented in the RISC papers and dissertations, so we encourage readers to investigate the issue carefully.

David Patterson  
University of California, Berkeley  
John Hennessy  
Stanford University

## Suggested reading

M. Hopkins, “A Perspective on Microcode,” Proc. 21st Ann. Computer Conf., San Francisco, Calif., Feb. 1983, pp. 108-110.

M. Katevenis, “Reduced Instruction Set Computer Architecture for VLSI,” ACM Doctoral Dissertation Award Series, MIT Press, Cambridge, Mass., 1985.

D. Patterson, “Reduced Instruction Set Computers,” *Comm. ACM*, Vol. 28, No. 1, Jan. 1985, pp. 8-21.

S. Przybylski et al., “Organization and VLSI Implementation of MIPS,” *J. VLSI and Computer Systems*, Vol. 1, No. 2, Fall 1984, pp. 170-208.

R. Sherburne et al., “A 32-bit NMOS Microprocessor with a Large Register File,” *IEEE J. Solid-State Circuits*, Vol. 19, No. 5, Oct. 1984, pp. 682-689.