# Proprietary versus Open Instruction Sets

Mark D. Hill
University of Wisconsin–Madison

Dave Christie
Advanced Micro Devices

David Patterson
University of California, Berkeley

Joshua J. Yi
Dechert

Derek Chiou
University of Texas at Austin

Resit Sendag
University of Rhode Island

Most widely used instruction set architectures (ISAs) are proprietary, while much software innovation is facilitated by being open. Might hardware innovation also be accelerated by ISAs being open? This article examines this question with an edited transcript of a debate from the 4th Workshop on Computer Architecture Research Directions.

An instruction set architecture (ISA) is one of the most important interfaces in a computer system because it divides software from hardware. Most widely used ISAs were developed decades ago and are proprietary. This may make sense because hardware implementations were and are mostly proprietary and most software was proprietary when these ISAs were developed. Today, however, we appreciate how open source software (such as LAMP stack) and open standards (such as TCP/IP) can unleash competition and creativity.

Might open ISAs accelerate the innovation of computer hardware, or are the lessons of open software not pertinent? That was the subject of the debate Mark D. Hill moderated at the 4th Workshop on Computer Architecture Research Directions in June 2015. Normally, a moderator is conflict free. In this case, Hill is conflict full, because David Patterson coadvised Hill's PhD with Alan Smith, and Hill currently consults for Advanced Micro Devices, where Dave Christie has worked for decades.

The first panelist was Dave Christie, a senior fellow at AMD. He first worked for the Control Data Corp. before moving to AMD in the late 1980s. He spent two decades contributing to virtually all of AMD's x86 processors, including the K5, Athlon, and Opteron. He developed the microcode for the K5, the first superscalar x86 processor designed independently from Intel. He is also known for having co-designed the x86-64 instruction set (the 64-bit extensions to the x86 instruction set). Currently, he is serving as AMD's ARM architecture liaison. He argued in favor of proprietary ISAs.

The second panelist was David Patterson, who holds the E.H. and M.E. Pardee Chair of Computer Science at the University of California, Berkeley. Of all his accomplishments and honors, most pertinent to this debate was that he led the design and implementation of RISC I, an early VLSI reduced-instruction-set computer. This research served as the foundation of the Sparc architecture. He has consulted for 25 years at various microprocessor companies, including spending a sabbatical at Digital Equipment Corp. in 1979 working on the VAX minicomputer and consulting for Intel for a few years on microprocessors, and he was the first person hired by Sun Microsystems to develop the Sparc architecture. He is a member of both the National Academy of Engineering and the National Academy of Sciences, and a Fellow of both IEEE and

ACM. He argued in favor of open source ISAs.

The format of the panel was as follows: each panelist had 10 minutes to present his position statement, after which the moderator asked the panelists a few questions. Finally, the floor opened up to the audience to ask questions. Video of the panel is available at www.ele.uri.edu/CARD.

## Dave Christie: It's Not the ISA, It's the Ecosystem!

Mainstream commercial ISAs—particularly x86 and ARM but also Power, MIPS, and Sparc—have served the industry well. Commercial ISAs operate as very effective de facto standards; standards provide stability, stability supports strong ecosystems, and strong ecosystems enable a tremendous breadth of applications. These commercial ISAs achieved a critical mass of support and took on lives of their own not solely due to an inherent "architectural goodness" in the ISA; they did so because of the ecosystems that have developed around them and the underlying economics.

Broadly speaking, the necessary conditions for an ISA to be commercially successful are that it should provide the basics needed to support the software development needs at the present time (for example, an instruction set that compilers can readily target, and protection mechanisms and virtual memory for multitasking OS support) and be commercially feasible to implement. Over the years, many ISAs have met these two conditions. But simply meeting these conditions is not sufficient to achieve the critical mass required for commercial success. Rather, commercial success depends more on marketing, support capabilities, economics, timing, and luck than on specific features within the ISA. Because the world is full of tradeoffs, architecting the "perfect" ISA does not ensure its commercial success.

But once an ISA achieves critical mass, the huge ecosystem supporting it makes it quite formidable. For example, we all know what happened when Intel tried to kill off x86.

ISA owners also have to be responsive to the needs of their customer bases. As such, they add extensions that make sense for their customer base and that will be accepted. The evolution of mainstream commercial ISAs—at least in the past decade or two—has demonstrated responsible stewardship by the ISA owners through the addition of extensions that have been carefully thought out from a benefit and need perspective, often in consultation with others in the industry. For example, one cannot reasonably view SIMD instructions, 64-bit computing, and machine virtualization as architectural missteps. Yes, there are occasional missteps—some things that seemed like a good idea at the time. But the same will eventually be true of RISC-V. In any case, this evolution, coupled with backwards compatibility, keeps proprietary ISAs very much alive.

In conclusion, the strong ecosystems that developed around mainstream commercial ISAs and the underlying economics of those ecosystems are a formidable hurdle for any open source ISA effort that would aspire to achieve parity, let alone replace them. In this vein, I have several important questions: What benefits and advantages would these open source ISAs have? How many open source ISAs do we need? Who will drive and participate in the development of these open source ISAs? And what is the business case for these ISAs?

I reviewed the RISC-V specification. Overall, it is interesting, a decent effort, and a clean design. It is clear that RISC-V's architects know what they are doing. They appear to recognize that widespread success is not going to happen overnight. RISC-V incorporates good lessons from the past, including base plus extensions. (While I have seen suggestions that this is a new invention from RISC-V, I disagree. x86 has had extensions for a long time; there are software visible ID bits that indicate whether an extension is present. ARM is similarly extensible.) Finally, it looks like a lot of fun. Designing an ISA is a lot of fun and something we do not do very often, but it is even more fun if we actually put it into wide use.

So, what are the holes in RISC-V? First, I think the need for it is questionable. Open RISC is a similar kind of effort. Lattice Mico32 is a 32-bit open source RISC processor that some people in industry are already using. Why do we need RISC-V when these other ISAs are commercially accepted alternatives, at least in certain embedded areas?

Table 1. Standards and implementations thereof in the computing industry.

| Field | Standard | Free, open implementation | Proprietary implementation |
|---|---|---|---|
| Networking | Ethernet, TCP/IP | Many | Many |
| Operating system | Posix | Linux, FreeBSD | Microsoft Windows |
| Compilers | C | GCC, LLVM | Intel icc, ARMcc |
| Databases | SQL | MySQL, PostgreSQL | Oracle 12C, Microsoft DB2 |
| Graphics | OpenGL | Mesa3D | Microsoft DirectX |
| Architecture | None | None | x86, ARM |

Second, a very long instruction word (VLIW) format appears to be an option in RISC-V. I think that this option goes a little too far and is too academic in that it is trying to be too many things for too many people. VLIW is best suited to highly specialized uses where there's little point in standardization. That said, I understand that this option may not be under serious consideration.

Third, I disagree with the premise that shared open-core designs, like RISC-V, shorten the time to market. It is not obvious to me that having a library of open source designs that you can pick from is faster than licensing a proprietary core, for example, an M3 core from ARM.

Fourth, I disagree with the premise that the industry needs a standardized ISA to save the world from proprietary ISA lock-in. Much has been done to improve software portability, and most software is ISA-agnostic (as illustrated by Apple's ISA shifts, Windows on ARM, XBOX ISA shifts, and smart TVs that are MIPS or ARM based from the same TV maker, running the same software).

Rather than standardizing at the ISA, I believe that the proper place to standardize the software/machine interface is with a virtual ISA or intermediate language. Heterogeneous System Architecture (HSA) is a great example of that approach.[1] It is designed to support the CPU and GPU, and it targets HSAIL (HSA Intermediate Language). ISA owners provide a finalizer that does a translation (at build time or runtime), and compilers can pass optimization information to it. HSAIL is currently in place for x86, ARM, and AMD GPUs, and is expected to be applied to other kinds of accelerators.

## David Patterson: The Case for Open Source ISAs

We live in a remarkable world, one where open source standards and open implementations of those standards really work. They are everywhere. Given that the computing industry has been revolutionized by open standards and open source software, why is one of the most important interfaces—the ISA—proprietary? Table 1 illustrates this fact. While ISAs may be proprietary for historical or business reasons, there is no good technical reason for the lack of free, open ISAs.

Dave Christie argues that ISAs do not matter that much because performance is due to the algorithms and software above that interface, or the hardware below it. I disagree; ISAs do matter. If they don't matter, why is the x86 ISA unsuccessful in mobile devices and why is the ARM ISA unsuccessful in datacenters? They matter because they are the most important interface in a computer system and where the hardware meets the software. This fact is particularly meaningful given that most of the cost of a new chip is the cost to port the software to it. On the other hand, I agree that most of the performance and energy running software on a computer is due to the algorithm, application code, compiler, OS/runtime libraries, microarchitecture, circuit design, physical design, and the fabrication process, but not the ISA. So if the ISA does not matter that much to the energy and performance, but it costs a lot to use different ones, why do we not have a free, open ISA that anyone can use for everything?

It is not an error of omission that ISAs are proprietary. It is not that AMD and Intel simply forgot to make x86 open. Rather, companies with successful ISAs like ARM, IBM, Intel, and MIPS have patents on quirks of their ISAs, which prevent others from using them without license. And they have lawyers to sue you if you allegedly infringe their patents and/or breach your license with them. Even taking a license, however, may not protect you from lawsuits. For example, Nvidia thinks the Imagination and ARM GPUs violate their patents. But Nvidia is not suing ARM; it is suing Qualcomm for using their intellectual property and Samsung for using Qualcomm chips.

Even IBM's Open Power is an oxymoron because you must pay IBM to use its ISA.

An ARM license does not let you design an ARM core; it only allows you to use ARM's designs. (Only about 10 to 15 big companies have licenses that allow them to design custom versions of ARM cores.) The cost of an ARM license is prohibitively high such that academics and many small companies cannot afford to take a license. While this business model may be sound, licenses stifle competition and innovation by stopping many from designing and sharing their ISA-compatible cores.

Apart from the intellectual property issues, there is no technical reason why an ISA should be proprietary. First, despite the value of the software ecosystems that grow around popular proprietary ISAs, the owners of proprietary ISAs do not do most of that software development. Rather, outsiders build almost all of the software in the ecosystem. Second, these companies do not have a monopoly on the experience needed to design a competent ISA. While it is a lot of work, many today can design an ISA. Third, a company that designs an ISA is not the only one who can verify it. Rather, long ago, open organizations developed mechanisms to ensure compatibility with hardware standards, such as floating-point units (IEEE 754), networking chips and switches (Ethernet), and I/O buses (PCIe). If not for such organizations, open standards would not be so popular. Fourth, the most popular ISAs are not particularly clean or elegant. For example, ARM and x86 are not considered to be exemplary ISAs; they are just functional and successful.

Finally, proprietary ISAs are not guaranteed to last. Rather, the ISA is tied to the fortunes of a particular company such that if the company dies, it takes its ISAs with it. An excellent example of this point is Digital Equipment Corp.; its demise terminated the Alpha and VAX ISAs.

What are the benefits of an open source ISA? First, open source ISAs would produce greater innovation because more people would get to design them, not just the engineers at Intel, ARM, and so on. Rather, engineers working for both open and closed companies and researchers all over the world could design and improve them. Furthermore, as is true in other fields, open source means that engineers can share their designs, which in turn produces further innovation.

Second, sharing cores will reduce the time to market and the cost because engineers do not have to design the cores themselves. Cost is becoming a very important factor because chips for the Internet of Things must cost less than a dollar.

Third, sharing cores reduces the number of errors because more people are looking at and debugging the designs. If you are afraid that some government agency has inserted a backdoor or kill switch into your chip, you have a better chance to discover it from the RTL itself when everybody can look at.

Finally, for those of us in academia, like our colleagues in operating systems and compilers, we can do our research on "industrial-strength" platforms that will not run into proprietary limits. For example, the lowRISC effort in Cambridge is trying to produce fully open hardware systems using RISC-V.

Why is this happening now? I think it is because Moore's law is ending. More specifically, improvements in cost and/or performance are not coming from the semiconductor manufacturers, but they are coming from architectural innovation. A lot of people believe that we are going to see a renaissance in domain-specific coprocessors like GPUs, DSPs, image processors, and so forth. If so, we do not want each type of coprocessor to have its own ISA. By contrast, we want to use an ISA that is standards-based, minimal, open, unencumbered by patents, that can run all standard software, and to which you can add your own coprocessors.

Table 2. Difference between monolithic (20th-century) and minimal modular (21st-century) instruction set architectures.

| 20th-century architecture | 21st-century architecture |
|---|---|
| ISA hardware is the microprocessor: • Microprocessor has a complete ISA, including what will not be used in the application because we cannot determine the environment when hardware is built. • ISAs only grow over time to support the past. | ISA hardware is intellectual property (IP) intended for a system on a chip (SoC): • SoC will customize the ISA used for this application when hardware is built. |
| Monolithic ISA microprocessors | Minimal modular ISA, such as RISC-V |

The difference between proprietary ISAs, labeled as "20th-century architecture" in Table 2, and minimal modular ISAs like RISC-V, labeled "21st-century architecture," is where the ISA is intended to run. In a 20th-century architecture, the ISA is bundled into the hardware. As such, I call these "monolithic ISA microprocessors." In this architecture, a company like Intel designs the ISA and gives you the chip. This microprocessor has to run anything and everything. As such, it cannot shrink; it will only enlarge over time. For example, the x86 ISA has gained an average of two new instructions a month for each of the past 40 years.

In a 21st-century architecture, the ISA should be intended for a SoC. The SoC will normally customize to the application, so it knows what software it is running. I called this approach "minimal modular ISA," an example of which is RISC-V.

In a nutshell, the base RISC-V ISA has a minimal number of instructions, less than 50, and it supports three different address sizes; 32, 64, and 128 bits. If you want the standard extensions, you add integer multiply and divide; atomic memory operations; and single, double, and quad floating-point instructions. RISC-V also supports smaller instruction sizes (16-bit and 32-bit). The key difference between monolithic and modular ISAs is that in the latter, software runs just on the base. You can add or subtract instructions and the rest of the software will still work. Finally, RISC-V has reserved space for domain-specific SoC instructions. See the "RISC-V History and Momentum" sidebar for more information.

Dave Christie suggested that we already tried the open ISA approach in the past and it failed. Perhaps surprisingly, I agree. Twenty years ago, Sun Microsystems had an open version of Sparc, and 15 years ago, there was another effort based on Deluxe (DLX). Why did these efforts not catch on? I think that they were simply too early. They were introduced in the monolithic microprocessor era, and there was no business need for it. By contrast, the era of minimal modular ISAs for SoCs may embrace open source ISAs.

Dave Christie opined that there's been good stewardship by proprietary ISAs. Again, I disagree. For example, 10 years ago, Intel tried to foist the Itanium architecture on everyone. The only reason it did not work is because there was a second source, AMD, which is a rarity for proprietary ISAs. Nevertheless, the industry wasted billions of dollars and companies went out of business all because Intel tried to force everyone to use a new ISA. ARM is doing the same thing right now. It's forcing the industry to use the ARMv8 ISA. But in this case, there is no second source.

I have spent a lot of time analyzing the ARMv8 ISA, and while it is gigantic—the ARMv8 manual is 5,400 pages long—it is still missing things. For example, while it has more than a thousand instructions, it does not have 16-bit instructions, so the code size is very large, even bigger than x86. As such, running the code will result in higher instruction cache miss rates. To compensate, their current cores have larger instruction caches.

In conclusion, given the consensus on ISA principles, there are no good technical reasons

### RISC-V History and Momentum

RISC-V ("RISC Five") is a modern RISC instruction set developed at the University of California, Berkeley, that was made free and openly available in response to requests from industry. In addition to a full software stack (compilers, operating systems, and simulators), there are several RISC-V implementations available for use in custom chips or in field-programmable gate arrays (FPGAs). Developed 30 years after the first RISC instruction sets, RISC-V inherits its ancestors' good ideas (a large set of registers, easy-to-pipeline instructions, and a lean set of operations) while avoiding their omissions or mistakes (branches are not delayed, it has support for virtualization, and it offers both 32- and 64-bit addresses).

RISC-V has gained considerable momentum since this debate last year.

- Four RISC-V workshops sold out, with the last workshop in Boston having 250 attendees from more than 100 organizations.
- The RISC-V foundation was established in August 2015 to support and evolve the RISC-V instruction set. Initial sponsors include Google, Hewlett Packard Labs, IBM, Lattice Semiconductor, Mellanox, Microsemi, Nvidia, Oracle, Rambus, Western Digital, and several smaller companies and start-ups.
- An optional compact extension was announced that makes RISC-V programs smaller than both the x86 programs and other RISC architectures.
- In addition to the initial free and open RISC-V core with a conventional five-stage pipeline and in-order execution core (Rocket), we've released a tiny, 0.01 mm$^2$ RISC-V core (Zscale) and a more sophisticated core that issues multiple instructions per clock cycle and executes instructions out-of-order (BOOM).
- Beyond the cores designed at UC Berkeley, designs are underway in Colombia, England, India, and Russia. In fact, the Indian government has invested US$45 million so far in RISC-V implementations.
- One RISC-V design optimized for FPGAs (Jan Gray's GRVI-Phalanx) runs faster than 300 MHz and fits 400 cores in a single affordable FPGA, thereby offering a peak performance of more than 100 billion instructions per second.

As a result of the rising popularity of RISC-V, the 2017 edition of the venerable textbook *Computer Architecture: A Quantitative Approach* will switch to RISC-V as its reference instruction set.

not to have a free and open ISA standard, as is the case in other fields. A standard ISA enables an open source ecosystem of cores and peripherals. An open source ecosystem allows many more people—not just a few companies—to use and build upon it, which will spur more innovation and reuse. If we are going to do that, which ISA would you pick? We should pick the one designed for the 21st century. It should be free and open. It should be minimal because a lot of innovation will come from the coprocessor, as compared to the cores. It should have a full stack of software running on it. Finally, it should be modular. It should have a minimal, standard subset of instructions, standard extensions, and space for application-specific unique instructions. I believe RISC-V should be that ISA.

## Moderator and Audience Questions

Here, we present an edited transcription of the conversations that took place.

### Are There Reasons against a Non-x86/Non-ARM Ecosystem?

*Hill:* A question for Dave Christie: You can argue that the ARM ISA had no initial ecosystem and it started out as a response to x86 and its very closed ecosystem. Now, ARM has a flourishing ecosystem with different goals than x86, but it's still semiclosed. So why shouldn't an open source architecture start and we get going on it?

*Christie:* I have no reason at all. Have at it. But I don't think there's the commercial pull for it that David Patterson seems to think there is. Maybe in time; it is certainly worth taking a shot.

*Patterson:* I think that if there was no open source software and we were the first people to try that, then there's a lot of reason to be skeptical. But when Linux started, it didn't have an ecosystem. It remarkably caught on with companies, and they poured resources into it. Obviously, this is like the Innovator's Dilemma. It's incomplete yet very promising, but it doesn't have everything you need. In this open source world, we think volunteers will help supply the missing pieces and build that ecosystem. We bet that in five years, RISC-V+ is going to be a significant force. We'll see; time will tell.

### Is the ARM Ecosystem Sufficiently Open?

*Hill:* Another question for Dave Christie: Another reason why this might fail is if the

ARM ecosystem is sufficiently open to enable the SoCs. Would you say that is the case?

*Christie:* Yes. There are so many players in the ARM ecosystem in terms of tools; you can buy instruction set simulators, program development tools, and debug tools for silicon from a variety of companies. This shared development of the ecosystem and shared cost of tool development is very effective for ARM's customer base.

*Hill:* David Patterson, do you think ARM has opened enough?

*Patterson:* No. They have not defined a coprocessor interface in the ARMv8 architecture, and a lot of people in this room really believe that coprocessors are the future of architecture. I presume ARM wants to supply all the coprocessors you ever want from them, as opposed to making it open. That's a glaring omission. ARM is a great company if what you are building right now fits and you can afford it. But it has been documented where ARM says it takes six or more months to negotiate a contract,[2–4] so for a lot of small companies, this delay is a real problem. Furthermore, ARM doesn't even want to do business with a lot of companies unless you're going to have volume. So they're kind of a monopoly, and they get to pick who works with them; they are not as open to business as you might think. Why wouldn't they have a coprocessor interface? There are business reasons for that decision, but no technical reasons.

*Christie:* The ARMv7 has this coprocessor interface—at least in the ISA—which they did away with because it really wasn't put to any use except for mapping system registers into the various functions. Why isn't—if we're talking coprocessors though—memory mapped a solution?

*Patterson:* Memory map works for some things, but you wouldn't want to do floating-point or vectors that way. There are some things you'd rather have closer to the processor.

## What Are the Incentives for Open Source, and How Can One Develop Them as Quickly as Possible?

*Hill:* Assume that open ISAs are a good idea. To make that work, you need large ecosystems. How does one develop an ecosystem for a new open architecture, and what are the incentives for the various parties to participate, given that you can't just replicate hardware for free but you have to fabricate it?

*Patterson:* The incentives are similar to the incentives for open source software. You can participate in the design and evolution of those systems, or you buy the proprietary system. There are advantages in helping shape what the future is going to look like, both for the ISA and their implementations.

In terms of getting RISC-V started, we're launching a foundation that we hope will have its first members at the RISC-V workshop, so it's not just a Berkeley effort. And we think that people will be attracted to building the missing pieces. We have faith in open source systems in general, and we think it will apply to hardware as well.

Some people are frustrated with ARM, as it's kind of a monopoly right now. I think it is beneficial—just like it is beneficial to operating systems and compilers—that there are open ones and closed ones. I think it'd be good for our field if there was an open ISA, which might temper the behavior of companies if they realize they're not the only sources in town.

*Hill:* Dave Christie, if you wanted to do this, how would you develop an ecosystem as quickly as possible?

*Christie:* Building an ecosystem, it's not just the tools, it's critical mass—it's critical volume of implementations and software support, it's people knowing that they write applications (or tools) for processors that will be mainstream products, and that it's worthwhile investing money in developing your application because there will be a market for it. Reaching that critical mass is somewhat less tangible, I think; that's why you need marketing and luck, along with good tools. ARM has done this, but it's been a long, slow climb.

## How Many Open Source Cores Do We Really Need?

*Christie:* I have a question for David Patterson. I'm not sure how big the market is for people who really want to design their own cores. A lot of people who want to design systems for a particular function like a set-top box or a TV—they aren't necessarily keen on designing the host processor or even any of the microcontrollers that handle various functions, power management, et cetera. They've got all they can deal with just to assemble the SoC functionality that they need for the product they envisioned. They're quite happy to use whatever ISA and processor implementation they can get their hands on. So there's limited appeal for people wanting to design those cores. There will be libraries, in some form, and there may be some financial backing for some of that from for-profit companies along the lines of Red Hat. So, I can see the ecosystem developing that way. But I wonder what the demand is for a wide variety of different cores: how many cores does an SoC or TV designer want to be able to choose from? Five or maybe 15—how many?

*Patterson:* We shall see. I think that in the open source community, a lot of the software gets done in academia; some of the implementation is done by the students and faculty. In terms of who would want to design that, I could imagine that that would be one source. I believe ARM has somewhere between 10 and 15 licensed companies who have the right to design their own cores and who have spent a lot of money for that right. So there's at least 10 to 15 companies that want to design ARM cores even though they can get one directly from ARM. For the high-end stuff—which AMD does—that's really rocket science. I think opportunities for RISC-V are going to be in the Internet of Things; we're talking about amazingly energy-efficient, amazingly involved small things, things that are 0.01 $mm^2$. There are no legacy barriers for the Internet of Things, there is a lot of innovation there that a lot of people can do, so I can imagine many people might be building cores for the Internet of Things.

## Will Open Source Kill Innovation Rather than Grow it?

*Audience member:* I am a little worried that open source ISA and open source microprocessor cores will change the business model, possibly through killing innovation in much the same way that open office software has killed innovation.

*Patterson:* Right now, there's only a couple of microprocessor companies, so to innovate in cores you have to work for one of those companies. Let's try an experiment where we let a lot more people design cores. What's interesting, especially over my career, is the widespread agreement on ISA designs. A lot of stuff we tried over the decades didn't work out so well, but there is also stuff that still seems like a pretty good idea. I don't know that we need a lot of innovation in ISA design now, but I think for cores, it's wide open given that Moore's law is slowing down and energy is becoming more important. These $1 chips for the Internet of Things offer a vast design space to explore. I think this will enable innovation because most architects will try something on their own. If they did a custom ISA, they'd try to port software over to it and nothing would run, and it would take several years. With RISC-V, you have the OS base, Linux, and all these compilers ready, so you can innovate on the things you want to do and innovate underneath the ISA, but the big pieces of software will run. I'm willing to take bets that, at worst, we're going to see a lot more innovation than we get from the ARM. Those cores are designed at one company. By contrast, if you look at the cores we designed at Berkeley—just one group at Berkeley—I think we designed some cores better than ARM. We'll show you the data.

*Christie:* I think that's an interesting observation. RISC-V has got a huge fan club and a lot of very smart people participating in the effort and doing really cool things. But over time, the initial people go off and do other things, and the business that was built around it gets certain constraints, and things mire down and …

*Patterson:* So, don't do anything, is that it?

[*Mark Hill throws penalty flag.*]

*Christie:* No, I just mean innovation that's happening now might taper off.

## How Does One Experiment with New Instructions in a Proprietary ISA?

*Audience member:* I know with RISC-V that if I want to experiment with transactional memory, new security models, or new memory systems, I can see how I can build those in that infrastructure. In a proprietary ISA ecosystem, how does a researcher experiment with these new innovations and get them updated?

*Christie:* For transactional memory work—I noticed that's a blank chapter in the RISC-V specification, which I don't fault them for. But there's been no shortage in the ability to experiment and develop, for example, transactional capabilities that could be adopted by any ISA, so I think that's not been a problem.

*Patterson:* We're doing extensions where we think there's a consensus in the architecture community. There's no consensus in transactional memory, and that's why we put a placeholder in there. However, we think we need to make more progress to figure what to do on something like transactional memory, and once we do, there is space to add it to RISC-V. I agree with the questioner that this is an opportunity to do significant stuff; we can build chips and run software and try things out. We'll give you an ecosystem, a register transfer level (RTL) design that you can start modifying, and then do your experiments.

As a concrete example, security research is done mostly by pointing out flaws rather than building things and seeing if they work. With RISC-V, you can try your idea and build a system, usually from field-programmable gate arrays, and try it out. You can put it up on the Internet and claim that it's going to work. You'd get sued if you try to do that with ARM. So you can't use the proprietary ISAs to do such experiments. ARM seems to believe in security by obscurity, while no security researcher that I know of agrees with that philosophy. Experts think their best path forward is where everybody knows what is going on, and then you try to break into it. You can do that with RISC-V. You could put your RTL out there and share it, so everybody doesn't have to do it themselves. I think one of the big releases of innovation in architecture is for the things we don't know how to do, because many could build systems that run software and try the ideas out. Otherwise, you have to wait for Intel or ARM to figure it out. And they'll give you their solution, and that's what you will be allowed to use.

## Should There Be More than One Open ISA?

*Audience member:* Should there be more than one of these open ISAs?

*Patterson:* No, I don't think so. [*Laughter*] First of all, it's a lot of work. When Krste Asanovic, Yunsup Lee, and Andrew Waterman started this, I said, "Don't do this, it's a bad idea." Krste said, "This will be three months." It took four years.

Remember that initially there were a bunch of competing BSD Unixes out there (FreeBSD, NetBSD, OpenBSD, PC-BSD), which divided the community. One of them had to be successful, but none of them were. Linux came later and took off when the field rallied around it. You could try to do another open ISA, and there are other efforts besides ours right now. For example, there is one that's based around the Hitachi SuperH. But my opinion is that the community needs to build the ecosystem to make one ISA successful before trying to build ecosystems for multiple ISAs.

I think if RISC-V became a monopoly or something, and you couldn't influence what was going on, that would be a good reason for another ISA. But we don't even know if RISC-V is going to work yet. The problem isn't that there's not enough open ISAs. The question is whether we can do this in hardware at all; can a free and open ISA be as successful as the LLVM compiler or Linux? That's the question.

*Christie:* I'm going to agree with the premise that one open source effort kind of makes sense. Although I disagree that they can cover anything you want to do. Do you have a GPU? Various accelerators? Is that the most efficient way to do a GPU?

*Patterson:* Yes, that's a good question.

*Christie:* Like I said, for certain accelerators, you want give the designers freedom to do their own customized ISA. And I think the intermediate layer is a good approach to deal with that.

*Patterson:* Yes, I think the question is could we build a GPU? We think you could build the GPU around RISC-V, but we haven't demonstrated it. [*Authors' note*: Since this talk, two other groups have announced open GPUs: the MIAOW GPU from the University of Wisconsin (http://miaowgpu.org) and Nyami from Binghamton University.]

## Why Can't We Have a Choice between Open and Proprietary ISAs?

*Audience member:* Will it be possible to have both an open ISA and a proprietary ISA together and have a choice, just like we have in the software world, such that we can go with the proprietary one if we wanted to and if we want innovation in the research community we can go with the open source?

*Patterson:* Okay, we promise not to drive Intel and ARM out of business.

*Hill:* You notice Dave Christie didn't promise. [*Laughter at implied reference to Intel.*]

*Patterson:* I think that in every case in Table 1, there was a viable, money-making proprietary one and a free and open one. Linux did not destroy all operating systems; Microsoft still makes a lot of money selling operating systems. The experiment is to see if we can have an open ISA. The proprietary ones are absolutely not going to go away, but can open source sustain a viable ISA, too?

## How Does One Deprecate Something from an Open Source ISA?

*Audience member:* So how do you deprecate something from an open ISA?

*Patterson:* What I expected from the question is how you prevent wretched excess like what's happening with x86 and ARMv7. Like other open source organizations, we are setting up a RISC-V foundation. I think of it like the US Constitution in that it gets seeded with principles, which are the instructions. And then we're going to have a voting mechanism by the members of the foundation to amend it. So we can add things to it and we can take things out, but it's going to be like the US Congress; it's a slow-changing system, and that's on purpose. For the areas we haven't defined, like transactional memory, there's a hole there; but once there's a proposal, that will be different. For stuff that gets established, it will probably be a slow-changing thing. That is why I think it's not going to add two instructions per month for 40 years like the x86. I think those parts will change slowly; the places that are brand new as they come along, but those will pop up.

*Christie:* Do you have a mechanism for the software discovery of features, namely, where the software is able to query whether or not the feature is present, and if not, it takes an alternate path?

*Patterson:* Yes, that's there.

## Will the Quality of an Open Source ISA Be Competitive with that of Proprietary Ones?

*Audience member:* My personal experience of the quality of most open source software is it is sort of like driving a Yugo. Why should we expect that the quality of this open source effort will be competitive with what we get commercially?

*Patterson:* Many think proprietary software also has dubious quality. ARM abandoned the ARM C compiler and endorsed the LLVM C compiler. That was an open source University of Illinois effort. I don't know if you used the Apache open source Web server, but I would be surprised if you don't. A huge part of this software you use every day is open source. I know there's lousy open source, but a lot of software the world depends on is open source. For example, Amazon's cloud computing software is all open source. As a second example, every time you do a Google search, you use open source software. I'd be thrilled if the RISC-V cores are the equivalent quality of the same "crappy" software that Google runs to do searches.

*Christie:* The idea of having all these eyes on the correctness for open source can be a good thing and may work really well, but at the same time, the Heartbleed bug is open source code, which is obviously a weakness. How would you like to have that—something of the magnitude of Heartbleed—cast in hardware?

*Patterson:* Yeah, fortunately, proprietary software has never had bugs. The argument is that more people looking at design will help. I guess if we had to build something with zero bugs, I don't think open source would do it; perhaps try formal verification. But I think it's common sense that reuse via sharing could well reduce the number of bugs.

H*ill:* Please make a one-minute concluding statement.

*Patterson:* Open architecture and implementations are a real possibility. I hope everybody in the room believes this could happen. It has already happened in other software fields. For it to happen in our field, it's going to need people to volunteer. What was great was that 150 people showed up at our first workshop. A lot of people came from these companies, but not representing their company. For example, there's a person who did validation for Qualcomm and he took a vacation day to come because he wanted to make this happen and was willing to contribute. People in this room could help make this happen. And it would be a very exciting future. I can see a lot of positives coming out of that, making our research even more relevant. We no longer would have to convince Intel or ARM to enable people to try our ideas. Rather, we could do it ourselves, put it out there, and have people start using it. So I see a potentially very exciting future, but it'll take volunteers to help make it happen. I hope some of you consider joining us.

*Christie:* I think if you want to write a finalizer for HSAIL [HSA's Intermediate Language] to target RISC-V, the HSA Foundation would be happy to include it.

*Hill:* Okay, so with that we will finish our debate. [*Blows whistle.*] MICRO

## References

1. HSA Foundation, 2013; www.hsafoundation.com.
2. A.L. Shimpi, "The ARM Diaries, Part 1: How ARM's Business Model Works," *AnandTech*, 28 June 2013; www.anandtech.com/show/7112/the-arm-diaries-part-1-how-arms-business-model-works.
3. C. Demerjian, "A Long Look at How ARM Licenses Chips: Part 1," *SemiAccurate*, 7 Aug. 2013; semiaccurate.com/2013/08/07/a-long-look-at-how-arm-licenses-chips.
4. C. Demerjian, "How ARM Licenses Its IP for Production: Part 2," *SemiAccurate*, 8 Aug. 2013; semiaccurate.com/2013/08/08/how-arm-licenses-its-ip-for-production.

**Mark D. Hill** is the John P. Morgridge Professor, Gene M. Amdahl Professor of Computer Sciences, and Computer Sciences Department Chair at the University of Wisconsin–Madison. His research interests include parallel-computer system design, memory system design, and computer simulation. Hill received a PhD in computer science from the University of California, Berkeley. Contact him at markhill@cs.wisc.edu.

**Dave Christie** is lead AMD64 architect emeritus, ARM architecture liaison, and a Senior Fellow at Advanced Micro Devices. His research interests include hardware/software interface, computer security, CPU microarchitecture, and performance instrumentation techniques. Christie began his never-ending computer architecture education at Ryerson Polytechnical [now University]. Contact him at david.christie@amd.com.

**David Patterson** is the E.H. and M.E. Pardee Chair of Computer Science at the University of California, Berkeley. His research interests include free and open instruction sets, domain-specific accelerators, and hardware support for improved security. Patterson received his PhD in computer science from the University of California, Los Angeles. Contact him at pattrsn@cs.berkeley.edu.

**Joshua J. Yi** is a patent litigation associate at Dechert. His research interests include microarchitecture, reliability, variation-tolerant processor design, and performance methodology. Yi received a PhD in electrical engineering from the University of Minnesota, Minneapolis, and a JD from the University of Texas at Austin. Contact him at joshua.yi@dechert.com.

**Derek Chiou** is a partner hardware architect at Microsoft and an associate professor at the University of Texas at Austin. His research interests include accelerating datacenter applications and infrastructure; rapid system design; and fast, accurate simulation. Chiou received a PhD in electrical engineering and computer science from the Massachusetts Institute of Technology. Contact him at derek@ece.utexas.edu.

**Resit Sendag** is a professor of electrical and computer engineering at the University of Rhode Island. His research interests include microarchitecture, memory systems, and simulation techniques. Sendag received a PhD in electrical engineering from the University of Minnesota, Minneapolis. Contact him at sendag@ele.uri.edu.