Critiques and Discussions Following David Patterson's Talk CS701

Thomas Reps
[Based on notes taken by David Schlais on September 17, 2015]

Abstract

This lecture concerns discussions based on student critiques of David Patterson's talk, titled "Instruction Sets Want To Be Free: A Case for RISC-V," which was given on September 15, 2015 in CS 1240 at 4pm. Students were asked to write critiques, and various topics—some of them from the students' critiques—were brought up for classroom discussion.

1 Topics

1.1 Changes to ISAs

- "It sounds like companies spend millions of dollars to add, validate, and launch new ISA extensions that introduce more—and more complex—instructions. Because RISC instructions were meant to be kept simple, why would companies abandon simplicity to add these instructions? Doesn't this make one believe that simple isn't always best, and that more complex instructions are meant to be added over time?"
  - Several ideas were presented in the class on why this may happen. The first topic that was brought up was a financial reason. The idea is that when a company frequently changes the design of a product, customers are forced to continually buy new products. Additionally, because patents expire, it may be in the company's best interest to make modifications that can lead to new patents, securing their intellectual property.
  - The second idea is that there may be specific applications where an added instruction may be useful, even if it may not be of utmost importance for general-purpose processors. Patterson also claimed that the specific ISA is not a huge factor in overall performance, and rather the algorithms and efficiently generating code with a minimum assembly instructions will play a larger impact than changing of instruction sets themselves. (However, Patterson also showed the results of experiments in which his team obtained substantial performance improvements using RISC-V, which makes one wonder about the statement "the specific ISA is not a huge factor in overall performance.")

1.2 Simplicity of RISC

- Hardware and software simplicity in Patterson's RISC mentality
  - A few students contributed thoughts about Patterson's claims to simplicity of the RISC architecture. For hardware simplicity, having a smaller number of instructions makes it easier to validate and test the ISA. Complex systems with hundreds of instructions create an exponential growth of possible instruction combinations. There also may be complexities and additional hardware needed to support obscure corner cases. By sticking to a simple design, it is easier to get a sleek, working, and validated ISA to support high speed/performance computations.

- For software simplicity, one student discussed Patterson's RTL generator, "Chisel." Although none of the class knew specifically how many lines of VHDL code is typically needed to create a crossbar switch, it was mentioned that companies designate teams of engineers to design them, while Chisel requires only 10 lines of code. This simple-but-powerful end of the software spectrum may make programming much easier. From these lines of code, there is a translational step to create the RTL, and the class was unsure if this translation creates inefficiencies in the RTL due to having a compiler that carries out this translation step.

1.3 RISC versus CISC

- "Is RISC (reduced instruction set computing) actually dominant over CISC (complex instruction set computing)? I.e., did RISC actually defeat CISC? How do micro-ops fit into this discussion?"
  - Patterson claimed that RISC by far dominates the market, and that RISC defeated CISC. However, it was mentioned that Patterson is not an unbiased observer: being one of the pioneer contributors to the notion of RISC architecture, and being the first to coin the term, he has a vested interest in the "RISC defeated CISC" storyline.
  - The class discussed how many of the architectures have become more complex through evolution, so the boundary between RISC and CISC is not always 100% clear. However, it may be important to note that some of the architectures that appear to have complex instruction sets, such as x86, have some sort of translation that breaks down instructions into "micro-ops." Thus, the architecture only truly has to support the micro-ops, rather than the assembly-language instructions that the ISA exposes to the compiler (or assembly-language programmer). This idea supports the case that RISC may actually be the preferred ISA paradigm in industry.

1.4 Doubts about RISC-V Remaining Small

- Roughly 70% of the critiques mentioned some form of doubt regarding the ability of RISC-V to remain small (currently around 50 instructions in the core instruction set). What instructions could you potentially see being added?
  - Cryptography
    - Cryptography seems to be a recurring operation in many applications. There may be benefits to having instructions designated specifically to reorder, shift, and/or hash bits, or to performing encryption.
  - Application-specific instructions: "RISC-V may need to be modified for different computing devices like servers and embedded processors." What are the different requirements that have to percolate down to the ISA level?
    - In the talk, Patterson mentioned instructions to support transactional memory.
    - A student mentioned how different applications such as servers, databases, embedded systems, etc. all have specific requirements. Each category may have repetitive tasks that could be performed by having a specific instruction dedicated for that task. For example, an embedded system that requires image/video compression may need vector dot-product summations. Having a designated instruction to do so may be beneficial, while other applications may ask for a different instruction for application-specific types of memory loads.
    - Security privilege levels?
    - RISC-V already supports a 16-bit instruction set to permit code size to be reduced for, e.g., embedded processors. Is there a security issue where 16-bit code, read in

32-bit mode, (or 32-bit code, read in 16-bit mode) can be forced to do something malicious (such as a return-oriented programming attack)?
  - External off-chip tasks
    * A point briefly discussed was that it seems that a lot of instructions that appear in CISC ISAs tend to try to remove external hardware by creating an instruction to do the same task. It is possible that some accelerator or other hardware device that takes up chip area would really push RISC-V to create an instruction to save power and area costs to have this extra hardware.

1.5 Potential Problems Regarding Open-Source

- What are some of the potential problems that may arise from transitioning into open-source? Is this as simple as Patterson made it sound?
  - Agile/SCRUM
    * One student brought up how Patterson described success in creating the RISC-V architecture using SCRUM and the agile programming method. The SCRUM model includes meetings with a small number of programmers contacting each other regularly to discuss what modifications are to be made and the carrying out of "sprints" (short bursts of modifications until a stable version is created). However, open-source coding appears to be the opposite: a large number of programmers that rarely interact with each other, and much less degree of coordinated interactions. Will this create problems? Is it maybe not as easy as Patterson makes it sound?
    * On the other hand, many successful open-source projects have a czar or inner circle that curate changes, so in practice the number of participants may actually be rather small.
  - Minimalism: Supporting Reduced Instruction Count Supported
    * Open-source projects usually do not have minimalism as one of the goals. With a RISC ISA, smaller and simpler is desirable. With multiple people looking to change the ISA, will keeping the instruction set small be a problem? The class discussed how "social friction" may help here: potentially conflicting ideas on what instructions should or should not be added may actually help to slow down the rate at which new instructions are added.
  - Security
    * A question was brought up about whether or not an open-source ISA would bring any security issues. More specifically, with all of the details of the ISA available to the public, could it present any issues that could later be exploited by hackers? Currently, Intel, AMD, and other companies have instructions that are not documented in the manual but are actually available for execution, which may not be possible in an open-source project.

      The class seemed to come to a consensus that open-source most likely does not present a security issue. There is already heavy documentation for most ISAs, which so far has not been a problem in terms of security.

1.6 When is the Best Time to Launch the Open-Source RISC-V? Are we Currently in a Post-PC Era? Are we in Transition Toward the Post-PC Era?

- Patterson claimed now is the best time, because we are in the post-PC era.
  - Although not heavily discussed in class, the point was brought up as to whether now was the best time to open RISC-V to become open-source. With emerging technologies, it is important to emerge into the market at the right time.

An analogy was made with Java: Java was originally developed to support television set-top boxes. That market failed (at least for Sun) at almost the exact time that web browsers started to emerge. The Java developers re-purposed their language to support programming web applets. Being the first to market as the opportunity developed was one of the biggest reasons for their success.

Being *in* the post-PC era is not necessarily the right time; better is to be in the *transition period* between the PC era and the post-PC era. What era are we in, and if we are already in the post-PC era is it already too late for RISC-V to succeed?

1.7 Other Topics Discussed

- Why couldn't we just subset an existing ISA?
- The importance of the 128-bit architecture
- An economic issue might work against RISC-V. For a variety of reasons, it may not be possible or desirable for large companies to switch (e.g., too much in the way of legacy investment in proprietary ISAs). Moreover, large companies can afford to pay royalties to ARM and Intel. So there is economic pressure for large companies to stick to the present proprietary ISAs.

  Could we really have a stratified world in which large companies use ARM and Intel, and small companies use RISC-V? However, there could be economic pressure against such stratification because one exit strategy for entrepreneurs and venture capitalists involved in small companies is for the small company to be acquired by a company, such as Apple, Samsung, Intel, Microsoft, Google, Facebook, or Amazon. A small company looking to be acquired may not want to be ISA-incompatible with potential acquirers that have deep pockets.