Ian Scott

11/10/2019

Wri 100

# Analysis on the Possibility of RISC-V Adoption

**Abstract:**

As the interface between hardware and software, Instruction Set Architectures (ISAs) play a key role in the operation of computers. While both hardware and software have continued to evolve rapidly over time, ISAs have undergone minimal change. Since its release in 2010, RISC-V has begun to erode the industry aversion to ISA innovation. Established on the principals of the Reduced Instruction Set Computer (RISC), and as an open source ISA, RISC-V offers many benefits over popular ISAs like Intel's x86 and Arm Holding's Advanced RISC Machine (ARM).

In this literature review I evaluate the literature discussing:

- What makes changing Instruction Set Architectures difficult
- Why might the industry choose to implement RISC-V

When researching this topic, I visited the IEEE (Institute of Electrical and Electronics Engineers), INSPEC (Engineering Village), and ACM (Association for Computing Machinery) digital library databases. I used the search terms, "RISC-V", "Instruction Set Architecture", "RISC-V" AND "x86", and "RISC-V" AND "Instruction Set Architecture". This literature review evaluates 10 papers on implementation of RISC-V. As this paper was

intended to cover recent developments in the field, publication dates were limited to from 2015 to present.

**Foreword:**

Instruction Set Architectures (ISAs) translate software instructions into simple, strictly defined operations. These simple instructions can then be translated directly to binary (on/off) instructions interpretable by processors. These instructions fall into one of four categories: load/store instructions, register based instructions, immediate based instructions, and jump instructions. All processes handled by a computer are eventually broken down to instructions on this level. ISAs all handle instructions in different ways however, fundamentally the purpose of an ISA is to allow communication between computer software and computer hardware.

**Historical Approaches to Instruction Set Architecture:**

Instruction Set Architectures (ISAs) are implemented directly onto the processor during manufacturing. As a result, the choice of ISA defines the types of fundamental instructions that the processor can interpret [1]. Part of the reason that ISAs undergo few changes is that they provide somewhat of a stable ground between rapidly changing hardware and software [1].

Every implementation of an ISA belongs to an established ‘architecture family’ such as x86, ARM, MIPS, RISC-V, etc. [1]. However, on a processor

level, there are multiple implementations of each ISA [2]. Each of these individual implementations is known as Microarchitecture [2]. While microarchitectures can deviate from their parent ISA to some degree, in general the parent ISA determines what instructions the processor can execute [2].

Modern processors fall into two general categories: Complex Instruction Set Computer (CISC), and Reduced Instruction Set Computer (RISC) [2]. RISC processors emphasize instructions which can be run by the processor efficiently [2]. CISC processor instructions tend to favor more complex instructions which means a shorter program, but instructions which take longer for the processor to run [2].

In recent years, a new ISA named RISC-V has seen a rise in popularity [3]. RISC-V is an ISA based on the principals of Reduced Instruction Set Computing (RISC) which emphasizes processing efficiency over program length [3]. Developed out of University of California, Berkeley in 2010, RISC-V is an atypical ISA [3]. While RISC-V is not the first ISA built on the principles of RISC, its open source nature allows customizations not possible under most ISAs [1][3]. Historically, popular ISAs have been closed source because they were developed by companies which wanted to charge money for their ISAs and protect their intellectual property by hiding the source code [1].

When new architectures develop, it becomes necessary to develop compilers and language libraries alongside them [4]. Compilers are programs which assemble code and translate it into assembly language (binary

instructions) for the processor. Language libraries are tools which provide documentation about different program instructions. When working with ISAs, these programs help engineers detect flaws in code may not be immediately apparent [4]. This is a complex task because in some cases problems manifest only after long periods of time or in different hardware [4]. This issue is complicated by the relaxed memory model used by RISC modeled ISAs in which instructions can be executed out of order based on type [1]. For example, when a RISC processor handles a store instruction, which moves information from a processor register back to storage in RAM, it will only execute once all prior memory accesses have resolved [1]. Some of the literature discusses the practicalities of creating and maintaining appropriate tools for the validation of ISA level codes [1][4].

**Limitations of Current ISAs:**

Throughout the literature it became clear that many authors felt that current ISAs had significant issues. This section will discuss the flaws that authors pointed out in their papers.

One problem present in many modern Instruction Set Architectures (ISAs) is that most ISAs used commercially are proprietary [5]. This means that their source code is privately owned, therefore hidden and uneditable [5]. In his thesis paper, “Design of the RISC-V Instruction Set Architecture,” Andrew Waterman makes the claim that opening ISAs to modification would

be beneficial to the field of computing because it would allow for greater optimization [5].

Another problem that authors encountered is that in many cases popular commercial ISAs perform worse than their newer counterparts [2]. In his thesis paper, “A Study on the Impact of Instruction Set Architectures on Processor’s Performance,” Ayaz Akram investigated how processors handle the execution of code in different ISAs [2]. His study measured the efficiency of execution from different ISAs by tracking the number of cycles it took for the processor to execute instructions [2]. According to the results of his study, on average x86 had the highest number of instructions and used more registers while running [2].

A large part of Akram’s thesis discussed Macro-Operation Fusion (M-Op Fusion). M-Op Fusion refers to the process of combining multiple small instructions into one larger instruction either in the compiler or in runtime [2]. The purpose of M-Op fusion is to consolidate instructions which run through the processor and reduce the number of cycles which a given program takes to run [2]. Akram’s data indicates that x86 instructions generated the greatest number of runtime Macro-Operation (M-Op) fusions on average among all ISAs tested [2]. RISC based ISAs tend to require fewer M-Op fusions because they carry a more efficiently compilable instruction base [2]. In Akram’s testing, ARMv8, which is a RISC based ISA never exceeded the number of M-Op fusions used by x86, which is a CISC based ISA [2].

Another trouble spot for modern ISAs is that tools provided by compilers, architecture specifications, and languages libraries for validating program stability are at best inefficient, and at worst non-existent [1][4]. In their paper "ISA Semantics for ARMv8-A, RISC-V, and CHERI-MIPS" presented at the Principles of Programing Languages (POPL) conference in 2019, Alasdair Armstrong, Thomas Bauereiß, Brian Campbell, Alastair David Reid et al. claimed that architecture specifications (materials intended to guide engineers on the use of a particular ISA) rarely amount to more than pseudocode documents [1]. They explain that, due to this, it is difficult to verify software [1]. The authors of “Promising ARM/RISC-V: A Simpler and Faster Operational Concurrency Model,” Christopher Pult, Jean Pichon-Pharabod, Jeehoon Kang, Sung-Hwan Lee et al. agree that most available programs fail to provide any kind of ISA model or debugging tools, choosing instead to focus on surface level instruction documentation [4]. Both teams went on to present their own verification tools: Sail and Promising respectively [1][4].

Sail is a custom language that was developed to verify ISA code [1]. In order to do this Sail emulates the selected ISA, while providing debugging tools [1]. During the creation of Sail, the developers struggled to balance including enough features to help engineers debug against the increased processing expenses that this incurred [1]. Promising, like Sail, is an emulator intended to aid in debugging [1][4]. Because Promising was developed after Sail, as such it makes use of some parts of Sail [4]. However

Promising focuses more on verification of edge cases by performing an exhaustive number of test cases [4].

Despite their contributions in the area, both sources agreed that the field could benefit from greater research into performance and debugging modeling for ISAs [1][4].

**Problems With Changing to RISC-V:**

In the literature, authors agreed that changing Instruction Set Architectures (ISAs) is problematic for various reasons [4][6][10]. According to the developers of RISC-V, the main problems were porting software to a new ISA and acquiring hardware which understood the ISA [6]. Other authors found difficulty in ISA emulation and verification [4][10]. This section will discuss the literatures' take on the difficulties of replacing existing ISAs.

At the ACM Design Automation conference, some of the engineers who designed RISC-V commented on the difficulty of making a new ISA [6]. Elad Alon, Krste Asanović, Jonathan Bachrach, Borivoje Nikolić et al. claimed that, while they had initially expected to finish RISC-V in a single summer, the process had actually taken the better part of four years of work [6]. They claimed that while building software capable of interfacing with given hardware was relatively easy, porting and maintaining software functionality on a new ISA was extremely difficult and time consuming [6]. This leads them to conclude that the primary difficulty with changing ISAs was rebuilding the software one wants to use with it [6].

Exacerbating the previous issue, authors in the literature noted that due to a lack of established tools, programming software for RISC-V is extremely difficult [10][4]. The authors of “Towards a High Performance RISC-V Emulator,” Leandro Lupori, Vanderson Rosario, and Edson Borin claim that programs allowing efficient RISC-V emulation are unavailable [10]. Currently the most efficient emulation of RISC-V carries a 12% overhead in x86 and a 35% overhead in ARM [10]. Overhead in this instance refers to the amount of additional work done by the processor compared to direct compilation to its native ISA. The authors go on to assert that the availability of efficient RISC-V emulation would both validate it as a useful architecture and ease deployment of RISC-V based systems [10]. In “Promising ARM/RISC-V: A Simpler and Faster Operational Concurrency Model,” Christopher Pult et al. assert that the reason it is difficult to write software without these tools is that in memory-relaxed languages such as RISC-V, non-deterministic behaviors can occur with extraordinary infrequency (on the order of once in a million executions), or only in hardware different from what it is being tested on [4]. Code exhibiting non-deterministic behaviors will offer variable output for a single input. This type of behavior is problematic because it makes code execution unpredictable [4].

**Advantages of RISC-V Architecture:**

It is the consensus of the literature that RISC-V carries some unique advantages over traditional ISAs. According to some of the creators of RISC-

V, the goal was to create an ISA equally suited to both simple and complex applications which could be easily updated and branched by the community [6]. Branching a piece of open source software refers to the process by which an engineer would add or remove parts of the source code in order to make the program better suit their needs.

Across the board, authors in the literature claimed the fact that RISC-V is open source software was a distinct advantage [1][6][7][8][9]. In fact, some of the literature reviewed document branches of RISC-V and how they optimize it for a specific, unique targeted task [7][8][9].

In their research paper, “Adding Tightly-Integrated Task Scheduling Acceleration to a RISC-V Multi-core Processor,” Lucas Morais, Vitor Silva, Alfredo Goldman, Carlos Alvarez et al. presented at the 52nd Annual IEEE/ACM International Symposium on Microarchitecture. The authors presented their research into optimizing RISC-V for multithreaded workloads [7]. One problem which occurs with multithreading is that if two cores attempt simultaneous access to the same location in memory, it will be inaccessible to one of them [7]. This error, known as ‘deadlock’, can cause execution of code to slow down or freeze up entirely [7]. To solve this problem, the authors branched RISC-V and modified its instructions to be non-blocking in order to ease the development of deadlock-free systems [7].

In their journal article, “Leveraging the Openness and Modularity of RISC-V in Space,” Stefano Di Mascio and Alessandra Menicucci discuss the possibility of developing special branches of RISC-V for use in space [8]. The

authors claim that one of the most appealing aspects of RISC-V is its modularity, which makes it an appropriate choice for a far broader number of use cases than a standard ISA [8]. In their article they cite the goals of space-oriented RISC-V branches to be increased ability to detect system faults, and increased stability in the case of faults [8]. According to their research, specially branched RISC-V implementations are capable of fault detection on par with processors specifically designed for use in space [8]. The advantage of using RISC-V in space over the traditional ground up specialized ISAs is that they are cheaper, easier to work on, and do not fall several years behind modern standards as specially engineered processors tend to [8].

In their conference paper presented at the 56th ACM/IEEE Design Automation Conference, Gai Liu, Joseph Primmer, and Zhiru Zhang presented their study on RISC-V ISA branches [9]. The team studied RISC-V branches which add custom instructions for use in cryptography and machine learning [9]. In testing of more than 60 processor implementations, branches of RISC-V demonstrated up to nine times the performance of a baseline RISC-V processor in cryptographic and machine learning applications [9]. In one particular case, the researchers observed a cryptographic RISC-V branch complete an encryption task 9.3 times faster than a baseline RISC-V processor [9]. It accomplished this with negligible increase to resources overhead and only increased cycle time by eight point four percent [9]. The

authors claim that the results of their paper demonstrate the benefits of RISC-V over traditional processors [9].

As shown by the examples, the literature seems to be in agreement that RISC-V carries some significant benefits over traditional ISAs [1][6][7][8][9]. Authors cite operational efficiency due to RISC construction as a reason to choose RISC-V, although RISC architecture is available on some other ISAs [2][6]. Authors seem to agree that the main thing that sets RISC-V ahead of the competition is the possibility of branching to better fit specific use cases [6][7][8][9].

**Modern Applications of RISC-V:**

The literature provided examples in which RISC-V is used in current computing processes [3][6][8][9].

In *Invited: Open-Source EDA Tools and IP, A View from the Trenches*, Elad Alon et al. discussed the fields in which RISC-V is beginning to see use [6]. They also commented on what they understand to be the reasons which RISC-V has seen adoption in modern fields [6]. The literature claimed that in order for RISC-V to be adopted, it was important for them to provide proven useful implementations of the software [6]. To this end, the team ported many useful open source programs to RISC-V [6]. The authors claim that, today, there are around six major companies manufacturing RISC-V cores [6].

In “Leveraging the Openness and Modularity of RISC-V in Space,” Stefano Di Mascio and Alessandra Menicucci wrote about modern applications of RISC-V in space [8]. They claimed that the idea of adapting commercial processors for use in space was becoming increasingly popular [8]. The main reason for this is that specialized processors tend to lag far behind commercially available ones in terms of performance [8]. The authors predict that the European space industry will likely flight test RISC-V cores in anticipation of future implementation in space [8].

In the literature “Design and Implementation of CNN Custom Processor Based on RISC-V Architecture,” Zhenhao Li, Wei Hu, and Shuang Chen make the claim that traditional ISAs are not optimized to take full advantage of parallel computing [3]. The authors predict that, as Moore’s Law slows and processor efficiency looks for other ways to improve, the industry may turn to RISC-V implementation in order to improve parallel processing performance [3].

In the conference proceeding "Rapid Generation of High-Quality RISC-V Processors from Functional Instruction Set Specifications," Gai Liu, Joseph Primmer, and Zhiru Zhang made a similar claim to that of the authors of “Design and Implementation of CNN Custom Processor Based on RISC-V Architecture” [3][9]. They claimed that the need for additional performance in the fields of encryption and machine learning would drive the adoption of RISC-V due to its ability to be branched for improved performance in these workloads [9].

**Conclusion:**

In reviewing this literature, I hoped to answer the following questions: what makes changing Instruction Set Architectures difficult? And why might the industry choose to implement RISC-V architecture?

According to the literature, the main problem with changing Instruction Set Architectures (ISAs) revolve around the amount of work required to translate existing programs and maintain supporting programs for new ISAs.

The literature shows that the industry may go towards implementing RISC-V architecture as a result of its open source nature. This, according to the literature, would allow companies the opportunity to better optimize their processors to their use case.

It seems that for RISC-V to continue to grow there will need to be improvement in emulation and translation. Further research in the field should focus on the optimization to translation and validation to improve the accessibility of the language for software engineers.

## References:

[1] A. Armstrong, T. Bauereiss, B. Campbell, A. Reid, K. E. Gray, R. M. Norton, P. Mundker, M. Wassell, J. French, C. Pulte, S. Flur, I. Stark, N. Krishnaswami, P. Sewell, "ISA Semantics for ARMv8-A, RISC-V, and CHERI-MIPS," Proceedings of the ACM on Programming Languages, Volume 3, Issue POPL, January 2019.

Article No. 71 Jan. 2019.

[2] A. Akram, “A Study on the Impact of Instruction Set Architectures on Processor’s Performance,” Master’s Thesis. Department of Electrical and Computer Engineering, Western Michigan University, August 2018.

[3] Z. Li, W. Hu, S. Chen, "Design and Implementation of CNN Custom Processor Based on RISC-V Architecture," IEEE 21st International Conference on High Performance Computing and Communications, 10-12 Aug. 2019.

[4] C. Pulte, J. Pichon-Pharabod, J. Kang, S. Lee, C. Hur, “Promising ARM/RISC-V: A Simpler and Faster Operational Concurrency Model,” Proceedings of the 40th ACMSIGPLAN Conference on Programming Language Design and Implementation (PLDI ’19), Phoenix, AZ, USA, 22-26, June, 2019.

[5] A. Waterman, “Design of the RISC-V Instruction Set Architecture,” Ph.D. Dissertation, Department of Computer Science, UC Berkeley, 2016.

[6] E. Alon, K. Asanović, J. Bachrach, B. Nikolić, *Invited: Open-Source EDA Tools and IP, A View from the Trenches.* In Proceedings of ACM Design Automation Conference (DAC ’19). ACM, New York, NY, USA, 4-6 June, 2019.

[7] L. Morais, V. Silva, A. Goldman, C. Alvarez, J. Bosch, M. Frank, G. Araujo, “Adding Tightly-Integrated Task Scheduling Acceleration to a RISC-V Multi-core Processor,” 2019 Association for Computing Machinery. MICRO ’52, October 12–16, 2019, Columbus, OH, USA

[8] S. D. Mascio, A. Menicucci, E. Gill, G. Furano and C. Monteleone, “Leveraging the Openness and Modularity of RISC-V in Space,” Journal of Aerospace Information Systems, 10, July 2019.

[9] G. Liu, J. Primmer, Z. Zhang, "Rapid Generation of High-Quality RISC-V Processors from Functional Instruction Set Specifications," 56th ACM/IEEE Design Automation Conference (DAC), June 2-6, 2019.

[10] L. Lupori, V. Rosario, E. Borin, “Towards a High Performance RISC-V Emulator,” Institute of Computing, UNICAMP, Campinas, SP,  Brasil, 16, October 2018.