Advantages and disadvantages of the RISC-V ISA (Instruction Set Architecture) in comparison to the ARMv8 ISA

1st Michael Schneider
Faculity of Computer Science and Mathematics
OTH Regensburg
Regensburg, Germany
michael4.schneider@st.oth-regensburg.de

2nd Florian Henneke
Faculity of Computer Science and Mathematics
OTH Regensburg
Regensburg, Germany
florian.henneke@st.oth-regensburg.de

3rd Alexander Schmid
Faculity of Computer Science and Mathematics
OTH Regensburg
Regensburg, Germany
alexander2.schmid@st.oth-regensburg.de

Abstract—text
Index Terms—keyword1, keyowrd2

I. Introduction

A. Topic

B. Motivation

C. Goal

D. Overview of paper

II. Background

A. Instruction Set Architectures

B. RISC

C. ARM

History, ... closed source (pay for licenses), exists longer -¿ more implementations, weiter entwickelt, ...

D. RISC-V

geschichte, ... open source, leichter und ggf kostengünstiger zu erstellen, ...

III. Concept and Methods (Initial section written by Alexander Schmid)

Given that the goal of this paper is to compare the two Instruction Set Architectures (ISAs) across a set of criteria that are relevant to semiconductor companies when evaluating which ISA to use with a new CPU design, these criteria shall be defined in the following section.

The first of these criteria is the ISA's business model. ISAs are often protected by patents that prohibit anyone not licensed by the patent owner from distributing Central Processing Units (CPUs) that implement that ISA. [1]

Whether these patents exist and the licensing terms are an important factor when deciding which ISA to use.

The ISA's complexity refers to the amount of effort required to implement the ISA. The more complex an ISA is, the more developer time is spent on implementing and verifying the implementation of the ISA, instead of optimizing the CPU for performance and efficiency, increasing a CPU's development cost. [2]

A CPU's performance usually refers to the speed at which the CPU executes a given program. Efficiency considerations, such as the code size of a given program or the amount of power the CPU consumes when executing a given program are closely related to performance and shall, for the purposes of this paper, be grouped under performance.

ISAs often allow for a number of instruction set extensions that may or may not be implemented by a given CPU. The choice of extensions greatly influence the flexibility of an ISA. Thus they are an important factor to consider which ISA to implement for a new CPU, because these have a large impact on performance and development cost of a CPU.

An ISA's ecosystem refers to the software that supports that ISA, especially compilers that compile to that ISA, operating systems and libraries. When developing a new CPU it is preferrable to use an ISA with a large ecosystem, in order to maximize the amount of software that can run of that CPU. This is especially important in consumer desktop and mobile devices where a large variety of software is to be executed.

A. Business Models (written by Florian Henneke)

[Introduction shortend because of word limit]

Beginning with the classic model of ARM, the following sections will cover the two license models: ARM sells its ISAs in various licensing models. [3] These are staggered in multiple levels of access. The 'Design Start' level includes free access to the IP-Core of the simplest ARM Chips Cortex-M0 and Cortex-M3 aswell as the corresponding toolchain and

processor models. For a fee between \$0 and \$75K the IP-Core of the Cortex-A5 aswell as the permission for 'single use' chip production can be aquired. This allows the customer to produce and sell one type of chip for a single purpose e.g. a network controller. For every chip produced, a royality must be given to ARM. The 'Design Start' access level also includes a license for accessing a 'artisan physical IP library', a license for universities which includes teaching and prototyping and allows non commercial production of own chips without royalties in small volumes. At last there is an 'FPGA' license which is free and includes a Field-programmable Gate Array (FPGA) optimized version of the Cortex-M3 and M1. Production is not allowed with this license.

[Removed short section due to word limit]

Above those access levels, there only officially exists the 'Standard' licensing model. This means one makes a individual contract with ARM. Several articles from 2013 [4] [5] talk about an older licensing model which contains special categories for higher access licenses. The highest of these, often referred as the 'Architectural' license is the only one that allows editing of the ISA and developing completely freely. The most prominent companies with such a license are Qualcomm which develops and sells mobile phone chips and Apple who just announced a 'Apple Silicon' developed laptop chip based on ARM. [6] The article also mentions that preparing a license of this form often takes about 6-24 months and states per-chip royalties of about 1-2.5%. It also notices so called 'foundry contracts' where customers can buy silicon ready ARM designs in cooperation with a silicon foundry. The most prominent example here are the Mali GPUs. This offers customers a fast and easy way to expand their chip with, for example, graphic accelerators.

In contrast to the ARM license model, RISC-V is published using the 'Creative Commons Attribution 4.0' license. [7] [8] This license allows the user to 'share' and 'adapt'. This means you are free to copy and redistribute as well as modify, change, build upon and sell it commercially. It is not necessary to share changes in an open source manner and you are only restricted by the obligation to give credit to the original licensor. [9] An important addition is also, that the license cannot be revoked by the licensor. This means everything about RISC-V that is already published will always be free to use. Originally founded by Berkley University, the RISC-V ISA standard is now managed by the nonprofit organization 'RISC-V International', founded in 2015. [10] As the statutes of the organization include, the association has 'no pecuniary, self-help or commercial purpose' [11]. Running expenses and further development of the standard do however require a certain liquidity. This is ensured by a membership program surrounding the specification. [12] Resembling the ARM licensing model, it contains three levels: 'Premier', 'Strategic' and 'Community'. Costing between \$2K and \$250K annually, these levels do not restrict access to the ISA, but grant several levels of taking influence on the future development of the standard through seats in the 'Technical Steering Commitee', speaker slots on conferences and representation on the official

RISC-V International website and blog. There are also three 'Strategic Directors', which are elected out of the 'Premier' and 'Strategic' Members and one Academic as well as one Communtiy Director, which is elected by the 'Communtiy' level of members. [13]

Besides taking influence in the development process, the membership also includes help in designing CPU Cores, teaching for employees and more. It also allows the usage of the trademark 'RISC-V'.

B. Complexity (Michael Schneider)

RISC-V and ARMv8 are both Reduced Instruction Set Computer/Computing (RISC) based architectures. Various RISC ISAs are different in complexity. To compare those differences, the basic instruction sets with corresponding extensions, the different realisations and two basic assembly instructions will be covered in the next chapters.

1) Instruction sets: The extension concept of the RISC-V architecture makes it only as complex as necessary, because the ISA can be tailored to the specific needs. In RISC-V the only mandatory instruction set is the Integer instruction set. Those base integer instructions cannot be redefined, only extended by optional instruction sets. [7] Further details about the extensions are in chapter III-D.
ARM instead defines the ARMv8 architecture in a completely different way. The ARMv8 already supports many more extensions in the basic version, also called v8.0. Further extensions are available in later versions, as explained in the chapter III-D. [14]
Because almost all the optional extensions of RISC-V are covered by the basic v8.0, the ARMv8, regarding only the instructions, is at least as complex as a fully extended RISC-V architecture.

2) Instruction set implementations: The different ISAs are able to implement the explained instruction sets in various ways. The RISC-V architecture is able to implement the instruction sets in 3 different word length, a 32-bit (RV32I), a 64-bit (RV64I) and a 128-bit version (RV128I). For the 32-bit and the 64-bit implementations are also multiple subversions available, RV32E and RV32G/RV64G. For 128-bit, RV128I is the only 128-bit implementation so far. RV32E is a version with only 15 instead of 31 registers. The subversion RV32G/RV64G is less a own version than a stable release. The RV32G/RV64G is combining a basic ISA (RV32I or RV64I) plus different selected standard extensions (IMAFD). [15]
Also ARMv8 has, different implementations, A64, A32 and T32. A64 is the 64-bit version and A32, T32 are both 32-bit versions. AArch64 and AArch32 are two different execution states in ARM (AArch64 for 64-bit and AArch32 for 32-bit). These execution states support the A64 instruction set in AArch64 and A32 and T32 in AArch32. While A32 uses fixed 32-bit instruction encodings, T32 is variable, which means it uses both 16-bit and 32-bit instructions. This is good to optimize your code, because a common instruction selection rule says, if a 16-bit encoding and a 32-bit encoding

are available, the 16-bit is the one you should use, to optimize the code density. [14]

3) registers and access: To complete the overview in a, for users more abstract point of view, two basic assembler commands (load and store) and the different instruction formats are compared. To load a value from a RISC-V register, LW, LH or LB is used. The "L" means load and the following characters stand for word (32 bit), halfword (16 bit) and byte. LH and LB are signed and can be extended by an "U" (LHU, LBU) to load unsigned values. [7] All instructions take 2 parameters, a register to store the value in and an address to load the value from. The address consists of the value stored in a register with an immediate offset. The store instructions SW, SH and SB work in the same way. SW stands for store word, SH store halfword and SB means store byte. The commands are structured in the same way as the loading commands are. The left side of the command is the register to take the value from and the second parameter is the register, containing the address, and the offset, where the value should be stored. [16] ARM instead has a few more instructions but the basics are almost the same. The (LDR) command can be extended by an B to load only a byte, SB to load a signed byte, H to load a half word, SH to load an signed halfword and SW to load a signed word. To store a value, there are 3 possible ways STR to store the complete register, STRB to store a byte and STRH to store a halfword. These basic load and store commands are followed by: load/store pairs, unscaled offsets and much more. Because there is no such possibility in RISC-V, there is no comparison about them. [14]

Additional to the explained load and store instructions both architectures have many other instructions, viewed are both times the clear 32-bit instructions RV32I for RISC-V and A32 for ARMv8. In RISC-V the instructions are summarised to Branch instructions, jump instructions, instructions with immediates, arithmetic/logical ops, instructions with upper immediates. The arithmetic/logical ops are called R-Format instructions and RISC-V has 10 of them. For the immediates (I-Format) RISC-V has 9 plus the 5 load instructions from above and EBREAK, ECALL as environment Call and Breakpoints calls. The Store instructions are limited to 3 and the branch instructions to 6 instructions. 2 Upper Immediate instructions and only 1 jump instruction. [7] In the ARMv8 ISA the instructions are summarised in a different way. The branch instructions (10 - 5 for conditional and 5 for unconditional branches) is the only overlapping format and already this is bigger in ARM. Further ARMv8 has data-processing instructions (18 standard instructions, 10 shift instructions, 48 multiply instructions, 8 saturating instructions, 14 packing and unpacking instructions, 2 divide instructions, 13 miscellaneous data-processing instructions). PSTATe and banked register access instructions (5), 22 miscellaneous instructions and 20 instructions to generate and handle exceptions. To complete the ARMv8 instructions the store/load instructions are also basic instructions. In ARMv8 there are 40 different instructions to do so. [14]

C. Performance (written by Alexander Schmid)

Code size is the performance parameter most influenced by the ISA and not by the CPU's implementation, given that the ISA and the compiler are the only two factors that influence code size. For the other performance aspects, the influence of the ISA is debatable [17] [18], so code size shall be the main focus of the performance comparison. Given that code size is most critical in embedded applications, the Embench benchmark suite is a good benchmark with which to compare code sizes. It consists of a number of programs frequently used in embedded applications, such as CRC, signal filtering, AES and QR code reading. [19] When compiling the Embench suite for both RV32IMC as well as 32-bit ARM with the Thumb-2 extension using GCC 7, the code for RISC-V is approximately 11% larger than the code for ARM. [20] Part of this gap in code size can be explained by the relative immaturity of the RISC-V implementation of GCC. RISC-V was introduced in 2017 and the code size of the Embench suite compiled for RISC-V is lower with newer versions of GCC, however it is still larger than the code generated for ARM as of 2019. [19]

In [20] an extension for RISC-V is introduced, called HCC, that is aimed at reducing the code size of RISC-V. This extension brings the code size gap down to 2.2% for the Embench suite and makes the RISC-V code smaller than ARM by 1.75% in a proprietary IoT benchmark developed by Huawei. [20]

TODO: In subsequent submissions, mention RV64 being significantly smaller than AArch64 as described in [15, page 62], and possibilities of comparing execution speed and energy efficiency as described in [17] and [18].

D. Extensibility

Extension system of RISC-V (only a base instruction set (integer instruction set)) and all others are extensions. ARM has a complete base version which contains almost all extensions of RISC-V. Furhter versions have much more extensions. Who can develop new extensions?

E. Ecosystem

Which compilers support ARM and RISC-V? Which operating systems and libraries?

IV. DISCUSSION

A. Advantages of ARM

What are the advantages of ARM compared to RISC-V? - more instruction formats and more basic instructions and extensions makes it more various applicable. - more different instructions makes the assembler code in a good written way smaller and more performant

B. Advantages of RISC-V

- lower amount of basic instruction sets and only optional extensions makes it less complex to implement. - less instruction formats makes it more easy to build - less different instructions makes the familiarization with programming easier What are the advantages of RISC-V compared to ARM?

C. Future directions and challenges

How can we more accurately measure performance differences between ARM and RISC-V and how do ISA extensions affect performance?

V. CONCLUSION AND OUTLOOK

A. Summary of results

B. Interpretation of results

C. Future directions

VI. OVERVIEW OF LITERATURE

Alexander Schmid [18] [14] [21] [22] [19] [20] [23] [15] [24]

Florian Henneke [15] [25] [21] [26] [27] [28] [29] [27]

Michael Schneider [7] [14] [30] [15] [31] [32] [33] [34] [35] [36] [37] [16]

REFERENCES

[1] G. Tang and I. W. Brown, "Intel and the x86 Architecture: A Legal Perspective," Harvard Journal of Law & Technology Digest, 2011, accessed on 2020-11-02. [Online]. Available: https://jolt.law.harvard.edu/digest/intel-and-the-x86-architecture-a-legal-perspective

[2] D. A. Patterson and D. R. Ditzel, "The Case for the Reduced Instruction Set Computer," *SIGARCH Comput. Archit. News*, vol. 8, no. 6, p. 25–33, Oct. 1980. [Online]. Available: https://doi.org/10.1145/641914.641917

[3] ARM, "How licensing works," accessed on 2020-11-09. [Online]. Available: https://www.arm.com/why-arm/how-licensing-works

[4] C. Demerjian, "A long look at how ARM licenses chips," Aug. 2013, accessed on 2020-11-09. [Online]. Available: https://semiaccurate.com/2013/08/07/a-long-look-at-how-arm-licenses-chips/

[5] ——, "How ARM licenses it's IP for production," Aug. 2013, accessed on 2020-11-09. [Online]. Available: https://semiaccurate.com/2013/08/08/how-arm-licenses-its-ip-for-production/

[6] Apple Inc., "Small chip. Giant leap." Nov. 2020, accessed on 2020-11-12. [Online]. Available: https://www.apple.com/mac/m1/

[7] Waterman et al., *The RISC-V Instruction Set Manual Volume I: User-Level ISA*, 2.2 ed., May 2017.

[8] ——, *The RISC-V Instruction Set Manual Volume II: Privileged Architecture*, May 2017.

[9] Creative Commons, "Attribution 4.0 International," accessed on 2020-11-15. [Online]. Available: https://creativecommons.org/licenses/by/4.0/

[10] RISC-V International, "About RISC-V," accessed on 2020-11-15. [Online]. Available: https://riscv.org/about/

[11] ——, "Articles of association of RISC-V International Association," Jan. 2020, accessed on 2020-11-15. [Online]. Available: https://riscv.org/wp-content/uploads/2020/04/Certified-copy-of-articles-of-RISC-V-International-Association.pdf

[12] ——, "Membership," accessed on 2020-11-15. [Online]. Available: https://riscv.org/membership/

[13] ——, "RISC-V International Association," 2020, accessed on 2020-11-15. [Online]. Available: https://riscv.org/wp-content/uploads/2020/03/RISC-V-International-Regulations-03-11-2020.pdf

[14] *Arm® Architecture Reference Manual. Armv8, for Armv8-A architecture profile*, Issue F.c ed., ARM, Jul. 2020.

[15] A. S. Waterman, "Design of the RISC-V Instruction Set Architecture," Ph.D. dissertation, University of California, Berkeley, 2016.

[16] N. Weaver, "Introduction to Assembly Language and RISC-V Instruction Set Architecture," 2019, accessed on 2020-11-20. [Online]. Available: https://inst.eecs.berkeley.edu/~cs61c/sp19/lectures/lec05.pdf

[17] E. Blem, J. Menon, and K. Sankaralingam, "Power struggles: Revisiting the RISC vs. CISC debate on contemporary ARM and x86 architectures," in *2013 IEEE 19th International Symposium on High Performance Computer Architecture (HPCA)*, 2013, pp. 1–12.

[18] A. Akram, "A Study on the Impact of Instruction Set Architectures on Processor's Performance," Ph.D. dissertation, Western Michigan University, 08 2017.

[19] D. Patterson, J. Bennett, P. Dabbelt, C. Garlati, and O. Shinaar, "Initial Evaluation of Multiple RISC ISAs using the Embench™ Benchmark Suite," Dec. 2019, accessed on 2020-10-24. [Online]. Available: https://riscv.org//wp-content/uploads/2019/12/12.10-12.50a-Code-Size-of-RISC-V-versus-ARM-using-the-Embench%E2%84%A2-0.5-Benchmark-Suite-What-is-the-Cost-of-ISA-Simplicity.pdf

[20] M. Perotti, P. D. Schiavone, G. Tagliavini, D. Rossi, T. Kurd, M. Hill, L. Yingying, and L. Benini, "HW/SW Approaches for RISC-V Code Size Reduction," in *Workshop on Computer Architecture Research with RISC-V. CARRV 2020*, 2020.

[21] K. Asanović and D. A. Patterson, "Instruction Sets Should Be Free: The Case For RISC-V," University of California, Berkeley, Tech. Rep. UCB/EECS-2014-146, 2014.

[22] Heui Lee, P. Beckett, and B. Appelbe, "High-performance extendable instruction set computing," in *Proceedings 6th Australasian Computer Systems Architecture Conference. ACSAC 2001*, 2001, pp. 89–94.

[23] C. Shore, *ARMv8-A Architecture Overview*, ARM Limited, Sep. 2015.

[24] X. H. Xu, S. R. Jones, and C. T. Clarke, "ARM/THUMB code compression for embedded systems," in *Proceedings of the 12th IEEE International Conference on Fuzzy Systems (Cat. No.03CH37442)*, 2003, pp. 32–35.

[25] L. Ryzhyk, "The ARM Architecture," Tech. Rep., 2006.

[26] S. Furber, *ARM System-on-Chip Architecture*, 2000, no. a.

[27] Microsoft, "Windows 10 on ARM," 2020. [Online]. Available: https://docs.microsoft.com/en-us/windows/uwp/porting/apps-on-arm

[28] Greenwaves Technologies, "Arm® Mbed™ OS Porting Manual for GAP8 ," accessed on 2020-10-28. [Online]. Available: https://greenwaves-technologies.com/manuals/BUILD/MBED-OS/html/index.html

[29] Amazon Web Services, "Using FreeRTOS on RISC-V Microcontrollers," accessed on 2020-10-28. [Online]. Available: https://www.freertos.org/Using-FreeRTOS-on-RISC-V.html

[30] A. George, "An overview of RISC vs. CISC," in *Proceedings The Twenty-Second Southeastern Symposium on System Theory*. Los Alamitos, CA, USA: IEEE Computer Society, mar 1990, pp. 436,437,438. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/SSST.1990.138185

[31] D. Patterson, "50 Years of computer architecture: From the mainframe CPU to the domain-specific tpu and the open RISC-V instruction set," in *2018 IEEE International Solid - State Circuits Conference - (ISSCC)*, 2018, pp. 27–31.

[32] J. Hennessy, *Computer architecture : a quantitative approach*. Waltham, MA: Morgan Kaufmann, 2012.

[33] R. D. Vladimir Herdt, Daniel Große, *Enhanced virtual prototyping featuring risc-v case studies*. S.l: SPRINGER NATURE, 2020.

[34] M. D. H. of Wisconsin-Madison; Dave Christie; David Patterson; Joshua J. Yi; Derek Chiou; Resit Sendag, "Proprietary versus Open Instruction Sets," *IEEE Micro*, 2016.

[35] S. Higginbotham, "The Rise of RISC," *IEEE Spectrum*, 2018.

[36] R. Dirvin, "The arm ecosystem: More than just an ecosystem, it's oxygen for soc design teams," April 2019, accessed on 2020-10-25. [Online]. Available: https://www.arm.com/company/news/2019/04/the-arm-ecosystem-more-than-just-an-ecosystem

[37] Z. Bandic, IEEE Computing Society, March 2019, accessed on 2020-10-25. [Online]. Available: http://www.hsafoundation.com/the-inherent-freedom-of-heterogeneous-systems/