ARMv8 advantages and disadvantages to RISC-V

***Abstract*—text**
***Index Terms*—keyword1, keyowrd2**

I. Introduction

A. Overview

B. Motivation

C. Goal

II. Background

A. Instruction Set Architectures

B. RISC

C. ARM

text

D. RISC-V

text

III. Concept and Methods

A. Business Models

Who develops the CPU cores, how can you get access to them? Who supports chip manufacturers in designing a chip with that CPU core?

B. Complexity (written by Michael Schneider)

Risc-V and ARMv8 are both Reduced Instruction Set Computer (RISC) based architectures. Compared to Complex Instruction Set Computer (CISC) based architectures they are less powerful in their individual instructions and they are operating on register. RISC machines have a lot of characteristics, which are not necessary to be completely implemented, but they can lead to different advantages of RISC, if they are, like a simpler control unit and faster decoding. [1]
Various RISC Instruction Set Architectures (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: 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. Further details about the extensions are in chapter III-D. This concept makes the RISC-V architecture only as complex as necessary, because the ISA can be tailored to a specific application. [2]
ARM instead defines the ARMv8 architecture in a completely different way. The ARMv8 alread 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. [3]
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). [2]
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. [3]

3) registers and access: To complete the overview in a, for users more abstract point of view, two basic assembler commands (load and store) 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. 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. [2]
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. [3]

C. Performance

What are the differences in code size? Can we accurately compare the execution speed of both ISAs?

D. Extensibility

What instruction set extensions are there for both ISAs? Who can develop new extensions?

E. Ecosystem

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

IV. Discussion

A. ARM

What are the advantages of ARM compared to RISC-V?

B. RISC-V

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. Accuracy of results

C. Future directions

D. Literaturverzeichnis

Die Quellen befinden sich in der Datei *biblography.bib*. Meine Quellen sind: [1] [2] [3] [4] [5] [6] [7] [8] [9] [10] [11] [12] [13]

References

[1] A. D. George, "An overview of risc vs cisc," 1990.

[2] A. W. L. A. P. Asanović, "The risc-v instruction set manualvolume i: User-level isa," 2016.

[3] A. Limited, "Arm architecture reference manual," 2013.

[4] G. David Patterson and U. Berkeley, "50 years of computer architecture," 2018.

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

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

[7] 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.

[8] IEEE, "The rise of risc," *IEEE Spectrum*, 2018.

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

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

[11] K. Asanović, "Instruction sets should be free: The case for risc-v," 2014.

[12] BerkeleyEECS, "Intoduction to assemblylanguage and risc-v isa," 2019.

[13] K. A. Andrew Waterman, "The risc-v instruction set manualvolume ii," 2019.