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

Risc-V and ARMv8 are both Reduced Instruction Set Computer (RISC) based architectures and copmared to Complex Instruction Set Computer (CISC) they are much less complex. RISC machines are having a lot of charactaristics, which are not necessary to be completely implemented, but they can leed to a few advantages like an simpler control unit and faster decode, both because of the less instructions and addressing modes. [1]
To compare the complexity of the both RISC Instruction Set Architecture (ISA) it is important to take a look into the RISC specific features and compare them in flexibility, amount and implementation.

1) Instruction sets: While the ARMv8 is clearly defined in relation to the instruction set, RISC-V is there much more variable. In RISC-V the only necessary Instruction set is the Integer instruction set. Those base integer instructions cannot be redened, only extended by more optional instruction-sets. Additional to operations for the base Integer Instruction Set, completely different extensions are available.

M	Extension for Integer Multiplication and Division
A	Extension for Atomic Instructions
F	Extension for Single-Precision Floating-Point
D	Extension for Double-Precision Floating-Point
Q	Extension for Quad-Precision Floating-Point
L	Extension for Decimal Floating-Point
C	Extension for Compressed Instructions
V	Extension for Vector Operations
B	Extension for Bit Manipulation
T	Extension for Transactional Memory
P	Extension for Packet-SIMD Instructions

The added alphabetical characters are defining the used instruction set with all his implemented extension. While an ”M” marks the architecture to be able to multiply the basic representation of on integer an extension like ”F” or ”V” makes the architecture undestanding a completely new number representation. This makes the RISC-V architecture only as complex as necessary because all the gratuitous instructions are not implemented. [2]
ARM insteed is defining the ARMv8 architecture in a completely different way. The ARMv8 supports also those extensions which are mentioned above but alread in the basic version, which is also called v8.0. To build a more complex CPU the ARMv8 also provides extensions like ARMV8.1, ... , ARMv8.6 plus a few optional extensions. Almost all the optional extensions of RISC-V are covered by the basic v8.0, which makes the ARMv8 only from the instructions set of view as complex as a fully extended RISC-V architecture. All those additional extensions like v8.1 etc makes the architecture much more complex than the RISC-V full extended one. [3]

2) Instruction set implemtations: Additonal to the basic instruction sets in the chapter above the different ISA are able to implement those basic or extension instructions in different ways. Again the start will be done by the RISC-V architecture. This architecture is able to implement the instruction sets in 3 different versions a 32-bit (RV32I), a 64-bit (RV64I) and a 128-bit version (RV128I). While for the 32-bit and the 64-bit implementations are again multiple subversions available, RV32E, RV32G/RV64G, the RV128I is the only 128-bit implementation so far. The lets say default version the RV32I implements 31 integer registers, the RV32E instead implements only 15 integer register. The subversion RV32G/RV64G is less a one version of implementation than a stable release. The RV32G/RV64G is not defining registers or instruction sets in its one way, it is combining a basic ISA plus different selected standard extensions (IMAFD). [2]
Also ARMv8 has, as aspected, different implementations, as well. While RISC-V is defining the implementations based on a number with the extensions, which are implemented, ARMv8 does it in a different way, because there are all the standard extensions already in the basic version included. ARMv8 provides 3 different implementations A64, A32 and

T32. As mentioned A64 is the 64-bit version and A32, T32 both 32-bit versions. AARCH64 and AARCH32 are two different execution states in ARM (AARCH64 for 64-bit and AARCH32 for 32-bit), which are supporting the A64 in AARCH64 or A32/T32 in AARCH32. [3]

Write some stuff comparing machine-level and supervisor level [4] with 3 basic architecture profiles (Application profil, real-time profil, microcontroller profil [3] S. 36

3) registers and access: To compare the both ISAs (RISC-V and ARMv8), which are the main components in this paper. I will base the argumentation on the amount of Instructions, the different modes of addressing and the complexitiy (how many different things are claryfied) in the manuel of each ISA. Only by counting the pages of the ARM manule (xxx) compared with the pages of the RISC-V manuel (xxx) we can assume that ARM is much more complex than RISC-V. By comparing - Instructions starts on [3] S. 222

How many instructions are there? How complex does that make the implementation of a core?

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:l [5], [3], [6], [7] [8], [2], [9] [10], [11].

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

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

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

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

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

[10] 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

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