ARM vs. RISC-V: A Comparative Analysis

Student ID: **1906055**
Student Name: **Tasmin Khan**

***Abstract*— In this term project, I will compare ARM and RISC-V microprocessors, drawing on skills gained from studying ARM processors in EEE 415. ARM, founded in 1985, transformed technology by developing code-dense, cost-effective, and energy-efficient microprocessors that now dominate gadgets such as smartphones and tablets. The current ARMv8, which is a 64-bit CPU with backward compatibility, continues to shape the industry. In contrast, RISC-V, which emerged in the 2010s, is an open-source alternative used largely in embedded devices and specialized applications. This assignment investigates the historical evolution, instruction sets, operand-addressing modes, pipelining structures, and future possibilities of both architectures.**

Keywords—ARM, RISC-V, microarchitecture, processor, chip, single cycle processor, pipeline processors, assembly instruction, machine language

I. OVERVIEW OF ARM AND RISC-V ARCHITECTURE

ARM is a proprietary instruction set (ISA) and has historically dominated the market offering extreme customizability and diverse applications with different hardware ecosystems. This is offered by multiple processing families such as Cortes-A (for high performance), R (for Real-time application) and M (for energy efficiency). Its proprietary model however limits the innovative researches. On the other hand, RISC-V architecture is an open-source alternative already being used by big tech companies. It also offers modular extensions for additional features and functionality such as B (Bit Manipulation), C (compressed) and V (vector) etc.

A. History of development of ARM

ARM (Advanced RISC Machine) architecture was initially developed by Acorn Computers back in 1980s for use in their Archimedes computers which demanded a high performance but energy efficient processor. This was a game-changer for computing revolution. The success of this first launch led to ARMv2 which was seven times faster. Each new version of ARM introduced new features making it even more robust and code dense. In 1990, ARM holdings were established as a joint venture between Acord computers, Apple and VLSI technology. This organization now controls the licenses for use. Even though it started with 180nm technology, the current chips are being made with 5nm technology. The latest ARMv9 architecture supports both 32-bit and 64-bit processing. Arm processors power the majority of the devices. The energy efficiency has significantly impacted the mobile and wearable tech industry.

B. History of development of RISC-V

RISC-V architecture was developed at the University of California, Berkeley in 2010s. This being an open-source instruction set, was subjected to rapid development and innovation. In 2015, the RISC-V foundation was established to standardize the architecture with contributions from academia and industry partners. RISC-V supports different extensions for different purpose and thus is increasingly getting popular. The first commercially available processor is SiFive's Freedom E310 processor. The latest in this series is the SiFIve performance P870 supporting up to 16 cores. The open-source nature is a game-changer in the chip industry where it is showing a promise to be used as an alternative of ARM in all use cases. This not only enables companies and even individuals across the world to use custom processors but also encourages a rapid growth in terms of innovation.

C. Basic Principles of RISC computing

Reduced Instruction Set Computing (RISC) is a design philosophy for microprocessors that emphasizes simplicity and efficiency. The basic principles mentioned by Sarrah L. Harris are: 1) Regularity supports simplicity, 2) Make the common case faster, 3) Smaller Addressing modes, and 4) Good design demands good compromise.

In both the architectures, the instruction formation and operations are simple and similar with minor difference indicating a regular and consistent design trend. Some common instructions are already defined in both the architectures to make programming and decoding faster for the users. They both have small SRAM arrays with limited registers for swift access and use instead of using the large memory every time accessing which is significantly slower. To keep it compact and efficient, ARM and RISC-V has limited and defined instruction sets of different categories. Ultimately, good design is about making thoughtful compromises for effective and efficient solutions. And from the discussion it can easily be said that both the architectures are revolutionary judging not only by their designs but also by their impacts.

II. COMPARISON OF ARM AND RISC-V INSTRUCTIONS

ARM and RISC-V both have distinct features from the number if available resisters to how the instructions are encoded. ARM has only 16 registers for use whereas RISC-V has 32 registers. This defines how many registers can be allocated for what kind of operation. For example – number of preserved and non-preserved register. RISC-V exclusively has a defined zero register whose value is always set at zero. Along with these there are more differences as well as similarities in instructions as we will explore now [1] [2].

A. Common Instructions

Basic operations like AND, OR, XOR, ADD, SUB and MUL are available in both the architectures. They both have some of the same shift operations, unconditional branching and memory access instructions with different names. Additionally, in RISC, jr instruction si similar to j but it jumps to the address value stored in a source register instead of using labels

ARM	RISC-V	Description
AND	and	
ORR	or	
EOR	xor	
ADD	add	
SUB	sub	

MUL	mul	Store only the lower 32 bits of the multiplication result
LSL	sll	Shift left logical
LSR	srl	Shift right logical
ASR	sra	Shift right arithmetic
B	j	Unconditional branch
STR	sw	Store register
LDR	lw	Load register
STRB	sb	Store byte
LDRB	lbu	Load byte unsigned
LDRSB	lb	Load byte signed

B. Instructions only available in ARMv7

Operations such as BIC or MVN are only available in ARM. However, NOT operation can be done in RISC-V by XOR-ing the source bits with negative 1. RISC-V also does not have rotate operation. Whereas ARM offers rotations in the right direction (ROR), since the left rotations can be achieved complimentarily from ROR. MOV instruction is also available in ARM, the similar functionality is achieved by adding zero with an immediate and storing the value in the destination register in RISC-V. ARM also offers extended memory access operations for loading halfword.

There is no conditional flags accessible in RISC-V. But since this is a feature of ARM, instructions may or may not affect the flags. Some instructions can be carried out in both ways using an additional 'S' indicating this is going to affect the flags. For example, ADD does not change the flags but ADDS does.

C. Instructions only available in RISC-V

There is a concept of pseudo instructions in RISC where some instructions are actually converted to one or more instructions in machine language, achieving the same result. Such as j is actually jal with no return address. I have already discussed about NOT, MOV and BGT operations. Similarly, nop can be done via adding zero to zero and storing that in the zero register etc.

D. Instructions with differences

ARM offers some advanced multiplication operations like UMULL (unsigned) and SMULL (signed) where it takes two destination registers to store the multiplication result values. This is necessary when the result is higher than what can be stored in 32 bits. In contrast, RISC-V has mulh(signed signed), mulhsu (signed unsigned) and mulhu (unsigned unsigned) instructions putting only the higher 32 bits of the result in the one destination register it takes. The difference in these three instructions lies in whether they treat one or both or none of the operands as signed.

Both the architectures have almost the same conditional branch instructions. However, the condition checks are done via conditional flags in ARM. Whereas the condition checking is done directly at the branch instruction command in RISC-V. This is almost equivalent to the CMP operation in ARM.

C code	ARM	RISC-V
if (operand1==operand2) { go to: LABEL; }	CMP R0, R1 BEQ LABEL	beq s0, s1, label

There are several common branch instructions as BEQ/ beq or BNE/ bne etc. However, there are far more mnemonics available in ARM like MI (minus), PL (positive or zero), VS (overflow) etc enabling a lot more conditional instruction. Even though some of them are absent in RISC-V, the same purpose can be achieved by just swapping the plas of registers, for example: BGT (greater than) or BLE (less or equal). Function calling is done by BL in ARM and by either jump and link (jal) or jump and link register (jalr) in RISC-V. jal follows a program counter relative addressing but jalr follows register value-based addressing.

For indexing we use [R0, offset] formatting in ARM. Where the address we access is the sum of these two inside the bracket. There are also options like post indexing, pre indexing indicating whether to alter the value in R0. There is no such direct function available in RISC. However, the similar result can be achieved with just a tweak in the code. Here the indexing format is like 0(t0) which accesses the value in the t0 address in memory.

E. Instruction Format

The type of instructions is vastly different in ARM and RISC. With the first one having Data processing, Memory and Branch instructions and the latter one having R-type, I-type, S/B-type and U/J type instructions. Bit allocation for RISC-V is as following:

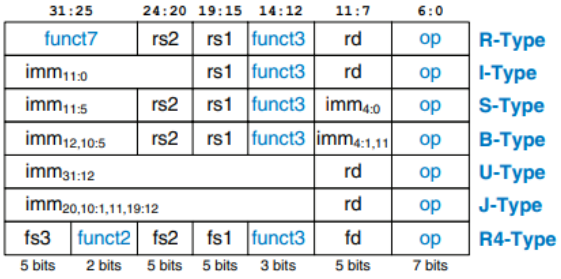

Fig 1: RISC-V instruction map [2]

Here, funct and op are used to differentiate instructions. Bit allocation for ARM is as following:

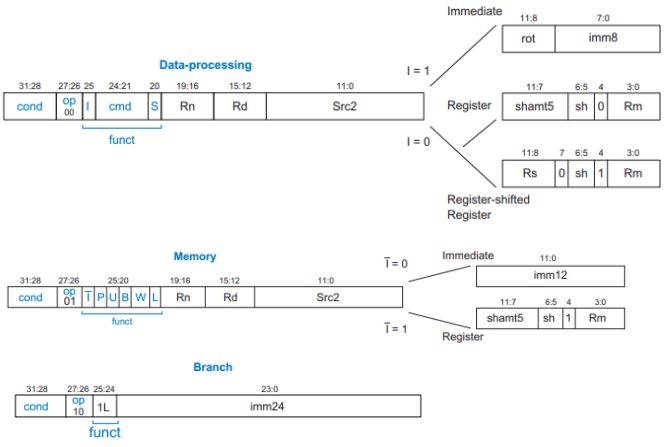

Fig 2: ARMv7 instruction map [1]

The branch instruction in ARM takes a 24-bit immediate value. It is however encoded a bit weirdly. The 24-bit immediate field gives the number of instructions between

the target and PC + 8 (two instructions past the branch). On the other hand, RISC-V branch instructions take both immediate values and registers. The immediate values are straight forward the direct address gap from the branching line. Since this will always be divisible by 4, the zero bit is ignored in the immediate. Immediates also are maintained in similar bit places for an easier hardware construction sacrificing easier instruction formatting.

III. ARM AND RISC-V MICROARCHITECTURES

The micro architectures are of three main variety for both ARM and RISC-V and those are single cycle processor, multicycle processor and pipelined processor. The following discussion will focus on two of these [1] [2].

A. *Single Cycle Micro Architecture*

Single cycle architecture refers to a system where the whole instruction is done in one single cycle of the clock. For these kinds of processors, the data memory and the register file are stored in different memories and can be accessed in the same cycle. Despite being very similar there are some differences such as only zero flag from ALU is fed into the control unit of RISC instead of four flags (NVCZ) like so in ARM. Inputs for the control signals are also less in RISC, even though number of output signals are same for both of them.

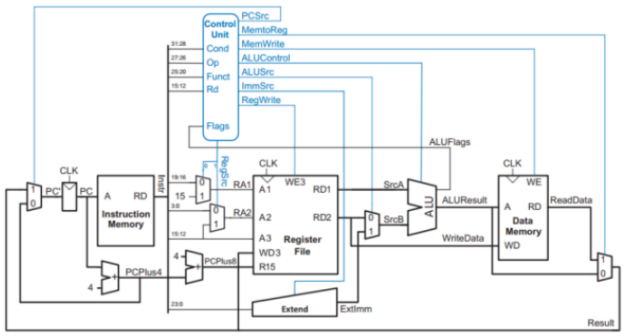

Fig 3: Single Cycle ARM processor [2]

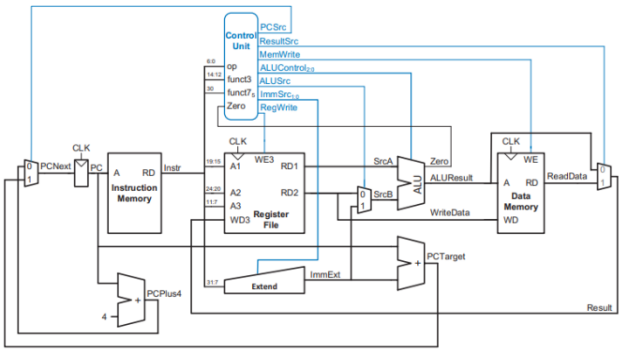

Fig 3: Single Cycle RISC-V processor [1]

There is one very striking difference in terms of Program counter (PC). The value of PC is stored after an increment of 8 in the register R15 in ARM. Reading PC value is thus structured to return PC+8. No such thing is done in RISC-V machines. The overall complexity is less and numbers of multiplexer and ALU components are thus less in RISC-V.

B. *Pipelined Micro Architecture*

The basic principles of in the pipelining process is very similar with the same five stages: Fetch, Decode, Execute. Memory Read/Write and Write Register. In order to achieve this structure, both of them uses some non-architectural registers. There are some minor differences in terms of how the different signals are handled. For example, in ARM the PCSrc signal comes from Write Register Stage but in RISC-V the same signal comes from the execute stage.

IV. FUTURE TRENDS AND DEVELOPMENTS

The future of computing is vastly dependent on the changes that both of these architectures will bring and the trends they will set. ARM has been the pioneer in revolutionizing the smartphone industry. It has also expanded into the industry of computers and laptops competing with x86_64 architecture. This in future is going to challenge the dominance of AMD and Intel. Recent developments, such as Qualcomm's Snapdragon Gen2 and Apple's transition to Apple Silicon M1 and M2 chips, highlights ARM's advancement towards high performance and robust computing, AI services and cloud applications. NVIDIA and AMD have already collaborated with ARM to use the chips for their GPUs. Even though ARM chips are not yet ready for heavy processing purposes, the energy efficiency and growing trend of high performance leaves an obvious promise of dominance in the chip industry for all kinds of devices. [3]

On the other hand, the RISC-V architecture being open source, is improving rapidly trying to replace ARM in the market. They are so far mostly popular for embedded systems. Intel and Google already plan to use RISC-V commercially for internal devices. One amazing perk of RISC-V is its great compatibility with another open-source OS – LINUX and so they are widely used in many LINUX CPU chips. Although still in its nascent stages, RISC-V's future prospects are promising, particularly in areas such as quantum computing, IoT, and gaming [4].

In conclusion, the strengths of the ARM and RISC-V architectures promise to change computing. Although RISC-V's open-source nature and customization capability promise disruptive breakthroughs, ARM's supremacy in high-performance computing and recent expansions predict sustained innovation. Together, they usher in a revolutionary era of computing, enabling both individuals and industries to explore new technological frontiers.

ACKNOWLEDGMENT

I would like to acknowledge the support of my teacher, Dr. Zunaid Baten, who provided invaluable guidance and insights for this assignment.

REFERENCES

[1] Digital Design and Computer Architecture RISC-V Edition Sarah L. Harris and David Harris

[2] Digital Design and Computer Architecture ARM Edition Sarah L. Harris and David Harris.

[3] "Trends Review: The Rise of the ARM Architecture," Plesk Blog. [Online]. Available: https://www.plesk.com/blog/plesk-news-announcements/trends-review-the-rise-of-the-arm-architecture/

[4] "Is RISC-V the Future?" SemiEngineering. [Online]. Available: https://semiengineering.com/is-risc-v-the-future/