Final Project Report: ARMv7 vs RISC-V Architectures

Humzah Imam - 100909864 | Suzanne Biju - 100926939| Hassaan Muhammad - 100928683

Introduction

This project investigates two prominent instruction set architectures—ARMv7 and RISC-V—through the implementation of basic programs in each while examining their design and efficiency. Our primary goals were to:

- Develop a Simple Arithmetic Calculator using ARMv7 (with QEMU or ARM-GCC)
- Create a Fibonacci Sequence Generator using RISC-V (utilizing RARS or Spike)
- Evaluate the instruction sets, usage of registers, and performance across both architectures
- Produce a demonstration video and a report detailing our implementations and insights

This project offered hands-on experience with low-level programming and enhanced our comprehension of contemporary processor architectures.

Background Study on ARMv7 & RISC-V

ARMv7

History

The ARM (Acorn RISC Machine) architecture was created in the 1980s and has since become one of the most prevalent processor architectures for mobile and embedded applications. ARMv7, launched in the 2000s, is a 32-bit architecture found in smartphones, tablets, and Internet of Things (IoT) devices.

General Description

The ARMv7 architecture employs a load/store design, enables conditional execution, and includes 16 general-purpose registers (R0–R15). It features advanced capabilities such as the Thumb-2 instruction set, SIMD, and floating-point operations.

Uses

- Mobile phones and tablets
- Embedded systems (such as smart TVs and cameras)
- Devices utilized in automotive and industrial applications

RISC-V

History

RISC-V is a contemporary open-source instruction set architecture (ISA) created at the University of California, Berkeley in 2010. It adheres to the principles of RISC (Reduced Instruction Set Computer) and is crafted to be straightforward, modular, and expandable.

General Description

RISC-V provides a fundamental 32-bit integer instruction set (RV32I), with optional extensions for floating-point operations, atomic functions, and additional features. It includes 32 general-purpose registers (x0–x31), with x0 being permanently set to 0.

Uses

- Academic research and education
- Customized hardware and system-on-chip (SoC) architectures
- Increasingly popular for low-power and embedded systems

Implementation Specifications

ARMv7 Math Calculator

- Developed with QEMU utilizing the ARM-GCC compiler
- Operations supported: ADD, SUB, MUL, UDIV
- Registers retrieved inputs from memory with the LDR instruction
- Results are displayed through a system call to output integers
- Incorporated an error check: division by zero activated a conditional branch

Example:

Input: 8 / 2

Registers: R0 = 8, R1 = 2

Operation: UDIV R2, R0, R1

Output: R2 = 4

RISC-V Fibonacci Generator

- Executed in RARS with both recursive and iterative approaches.
- Base cases managed through branching instructions (beq, bge).
- Registers x5 and x6 are employed for monitoring Fibonacci numbers in the iterative implementation.

- The output is displayed using the syscall print_int after the conclusion of the loop.

Example:

Input: N = 7

Output: 0 1 1 2 3 5 8

Comparison of Architectures

Similarities

- Both utilize load/store register-based architectures.
- They both support operations related to arithmetic, logic, and branching.
- Pipelining is employed in both to enhance instruction throughput.
- Both adhere to RISC principles to minimize complexity and enable quicker execution.

Differences

Feature	ARMv7	RISC-V
Register Count	16 general-purpose registers	32 general-purpose registers
Instruction Format	Mixed (Thumb-2 = 16/32-bit)	Fixed 32-bit (base)
Encoding Complexity	More complex	Simpler, consistent
Open-source?	No (requires licensing)	Yes (free and open)
Learning Curve	Higher (due to complexity)	Easier (simplified set)
Instruction Set	Richer set including conditionals	Minimal, modular extensions

Advantages & Disadvantages

ARMv7

Advantages

- Established and extensively supported
- High efficiency with optimized instruction set

Disadvantages

- Proprietary nature
- Higher learning curve

RISC-V

Advantages

- Open-source and adaptable
- Excellent for educational purposes and research

Disadvantages

- Limited acceptance in the industry (though increasing)
- May lack certain advanced optimizations in its basic version

Conclusions

This project enabled us to grasp the technical distinctions and practical uses of ARMv7 and RISC-V architectures. We acquired hands-on experience in writing assembly programs, diagnosing issues, and utilizing simulators like QEMU and RARS. Although ARMv7 continues to lead in commercial use, RISC-V presents a promising option as a lightweight, modular, and open alternative.

What we Learned:

- Crafting and debugging low-level assembly code necessitates accuracy and determination.
- CPU architectures have a significant effect on how programs are organized.
- Instruction set architectures that are simpler (such as RISC-V) are more comprehensible and easier to learn.
- Performance encompasses more than just speed; it also involves elements like pipeline effectiveness, memory access, and instruction design.

References

Patterson, D. A., & Hennessy, J. L. (2017). *Computer organization and design RISC-V edition: The hardware software interface* (1st ed.). Morgan Kaufmann.

ARM Ltd. (n.d.). *ARM architecture reference manual ARMv7-A and ARMv7-R edition*. Retrieved from https://developer.arm.com/documentation

Waterman, A., Lee, Y., Patterson, D., & Asanović, K. (2014). *The RISC-V instruction set manual Volume I: User-level ISA*. Retrieved from https://riscv.org/technical/specifications/