ISSN: 2993-8643

Review Article

Engineering: Open Access

Exploring Instruction Set Architectural Variations: x86, ARM, and RISC-V in Compute-Intensive Applications

Wajid Ali*

Electrical Engineering dept University of Engineering and Technology Lahore, Pakistan

***Corresponding Author**
Wajid Ali, Electrical Engineering dept University of Engineering and Technology Lahore, Pakistan

Submitted: 2023, Aug 06; **Accepted**: 2023, Sep 26; **Published**: 2023, Oct 02

Citation: Ali, W. (2023). Exploring Instruction Set Architectural Variations: x86, ARM, and RISC-V in Compute-Intensive Applications. *Eng OA, 1*(3), 157-162.

Abstract

As computational demands continue to evolve in the modern era, the choice of hardware architecture plays a pivotal role in optimizing the performance of compute-intensive applications. This research paper delves into the exploration and comparison of three prominent hardware architectures: x86, ARM, and RISC-V, within the context of compute-intensive applications. The study begins with a comprehensive overview of these architectures, highlighting their distinctive features, and strengths. Subsequently, we investigate their suitability and adaptability in diverse compute-intensive workloads. Our analysis encompasses a wide spectrum of parameters, including computational throughput, power efficiency, scalability, and architectural flexibility. We scrutinize the architectural intricacies that impact the execution of compute-intensive tasks, shedding light on both the advantages and limitations of each architecture. We used the gem5 simulator to compare these Instruction Set Architectures (ISA). We run different benchmarks on gem5 with different ISA and different configurations and compare the result. Based on these results we predict which architecture is better in which scenario. Gem5 is not a cycle accurate simulator but it's a model accurate. In conclusion, "Exploring Architectural Variations: x86, ARM, and RISC-V in Compute-Intensive Applications" offers a comprehensive insight into the nuances of hardware selection for compute-intensive workloads. Our findings aid system architects, researchers, and technology enthusiasts in making informed decisions about the most suitable architectural choice for their specific compute-intensive applications, ultimately contributing to advancements in computational performance and efficiency.

Keywords: Computer Architecture, Gem5, Isa Comparison, Intensive Workloads

I. Introduction

In the realm of computer architecture, Intel's x86 design has long championed peak performance, catering primarily to the demanding needs of personal computers and servers. Conversely, ARM diligently pursued energy efficiency enhancements, targeting the mobile devices and wearable technology sectors. However, the relentless march of technological progress has ushered in a convergence of objectives [2]. Intel, a stalwart in the realm of traditional computing, has expanded its horizons by producing processors for handheld devices. Simultaneously, ARM, synonymous with energy-efficient mobility, has ventured into the realm of servers. This intriguing shift marks a departure from the well-defined roles these architectures once occupied.

In today's landscape, where x86 and ARM-based processors engage in head-to-head competition, it is imperative to scrutinize their performance capabilities across a diverse spectrum of applications. Furthermore, the hardware industry has witnessed a seismic transformation with the advent of RISC-V, an open-source architecture that challenges established conventions. As these architectural forces converge and compete, our research embarks on a comprehensive journey to assess their respective strengths and weaknesses across various application domains. In this rapidly evolving technological arena, where x86 and ARM architectures realign their objectives, and RISC-V disrupts the traditional hardware paradigm, this study serves as an indispensable compass. It illuminates the nuanced capabilities and far-reaching implications of these architectures, providing invaluable insights for decision-makers navigating the complex landscape of contemporary computing environments. Two primary types of instruction set govern modern computer architectures: CISC (Complex Instruction Set Computer), exemplified by x86, and RISC (Reduced Instruction Set Computer), represented by ARM. The RISC architecture stands out for its streamlined instruction set, which contrasts with the complexity of CISC. A key distinction lies in how these architectures access memory. In the RISC paradigm, memory access is achieved through dedicated instructions, namely, 'LOAD' and 'STORE.' In contrast, CISC architectures embed memory access methods within other instructions, resulting in variable instruction lengths. While this approach reduces the number of instructions required to execute a program, thereby easing the compiler's workload in translating high-level instructions to assembly-level equivalents, it places a higher demand on hardware to support these intricate instructions. RISC architectures, on the other hand, employ fixed-length instructions, simplifying the decoding process. This shift, however, places a greater burden on the compiler to efficiently convert high-level instructions to their assembly-level counterparts. Nevertheless, the reduced hardware complexity in executing these straightforward instructions facilitates pipelining. With RISC architectures steadily improving their hardware efficiency, Intel embarked on a development path that incorporates RISC-like micro-architecture to harness these advantages. This entails the translation of x86 instructions into RISC-like instructions within the hardware before execution. While this approach maintains the appearance of traditional x86 operation to external observers, it internally executes RISC-like instructions [2]. In order to check which ISA is better many experiments are already done but these experiments and research is on the real hardware our experiment is on the gem5 simulator which is the model accurate simulator [3-5]. Gem5 is the simulator that made by the merging of two simulator M5 by the University of Michigan and gems by the University of Wisconsin.Gem5 has various ISA like MIPS, RISCV, X86, ARM, ALPHA,

POWER and SPARC. There are different prebuilt boards in gem5 like X86 board, ARM board and RISC-V board. There
II. *Workload*
III. EXPERIMENTAL WORK

are two modes of simulation sys-call emulation and full system. In the sys-call emulation we don't use any Disc image however while doing a full system simulation we need a proper Disc image. We can also use the kernel or terminal during the full system simulation of that Disc image by using m5 terms. There are different cache models like Ruby cache by gems and Classic cache by m5. There are different memory access options also present in it like Timing memory access and atomic memory access. We can use different CPU models such as KVM, in-order, out-order, Timing and Atomic. According [1] the gem5 is the cycle accurate model but according to the latest study it is known that gem5 is not the cycle accurate it's a model accurate simulator we can validate any model on gem5 because it is not cycle accurate but we can build a different model and test it on the gem5.

RISC is far better than CISC when it come into the highly intense computing [4]. The concept of RISC is normally integrated in the CISC. In some early research the author compares the in-order and out-order CPU available in gem5.

Early investigations have suggested that in the realm of highly parallel computing environments, RISC architectures may offer advantages over CISC counterparts [4]. Furthermore, the incorporation of RISC principles into CISC designs seems to be an ongoing trend in the field [6].

A comprehensive study conducted by [7] extensively compared ARM and x86 microprocessors within the gem5 framework. The assessment encompassed both in-order and out-of-order CPU models and featured an evaluation based on four critical performance metrics: average cycles-per- instruction (CPI), L2 cache miss rate, throughput, and total energy consumption. Their analysis, executed using the MiBench benchmark suite, notably showcased the ARM microprocessor's superior performance across most scenarios when compared to its x86 counterpart.

Another comprehensive exploration, undertaken by [5], involved an in-depth analysis of x86, ARM, and RISC-V Instruction Set Architectures (ISAs) within the gem5 framework. This investigation employed three distinct configurations: in-order, out-of-order1, and out-of-order2. McPAT was utilized to estimate power consumption, and simulations relied on benchmarks drawn from SPEC2006 and BEEBS. The findings from this research point towards the remarkable performance and energy efficiency of the ARM ISA, surpassing both RISC-V and x86 architectures. Interestingly, the performance distinction between ARM and RISC-V proved to be marginal.

In subsequent research efforts, [3] and [8] leveraged the gem5 framework to delve into the ramifications of altering cache parameters on overall system performance. These studies provided valuable insights into the intricate relationship between cache configurations and the optimization of hardware architectures for improved system performance.

In this research paper we used different benchmarks and run the Radix Sort Algorithm with the help of Timing CPU. This algorithm sort the given reverse sorted array. For the comparison of different ISA, we use the C language code of the Radix Sort of different number of arrays then compile it with the compiler. As my host machine is X86 so for X86, we simply compiled my C code by the following command:

gcc RadixSort.c -o Radixsort
For the ARM and RISC-V we used the cross compiler we cross compiled our C code to form the corresponding binaries. For ARM we used the following command: ***aarch64-linux-gnu-gcc --static -o RadixSort RadixSort.c*** For the RISC-V we used the cross-compiler for RISC-V to compile our C code ***riscv64-linux-gnu-gcc --static -o RadixSort RadixSort.c***

The binaries that we made are then simulate by the gem5.

A. *Configuration of gem5*
Initially, we prepared the gem5 program executables. Specifically, we utilized the gem5.fast executable for all three Instruction Set Architectures (ISAs). To create these executables, we executed the following commands:

For RISCV: ***scons build/RISCV/gem5.fast -j{nproc}*** For ARM: ***scons build/ARM/gem5.fast -j{nproc}*** For X86: ***scons build/X86/gem5.fast -j{nproc}***

To determine the appropriate number of threads to use, we employed the "***nproc***" command in the terminal. This command provided us with the number of available threads on our machine, helping us optimize the build process.

IV. Results

A. Timing simple CPU with no cache hierarchy :

We created a Python script where we utilized a Simple CPU configuration that doesn't include any cache. The CPU operated at a clock frequency of 1 gigahertz (1GHz).

1) Sim Ticks

The data in Table 1 SimTicks illustrates fluctuations in Sim- Ticks, with values presented in billions.

No elements in Array	ARM	RISCV	X86
100	11.029	13.811	23.117
512	54.340	69.035	81.831
1024	116.730	148.843	161.609
2048	230.617	297.148	13.506
4096	464.469	599.624	618.778
8192	1040.880	1268.804	1315.286
16384	2055.154	2639.056	2717.258
32768	3989.682	5093.749	5238.690

Table 1 SimTicks

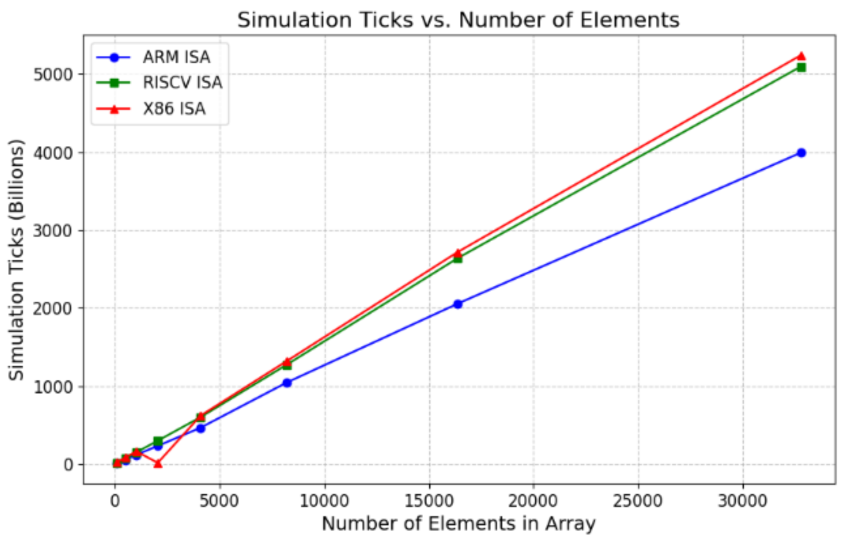

Figure 1: visually represents the fluctuation of Sim-Ticks across different ISA configurations.

The data in the Table 2 illustrates the fluctuations in SimOps, with values presented in Millions.

No of elements in array	ARM	RISCV	X86
100	0.1883	0.1733	0.5057
512	0.9288	0.8753	1.8168
1024	1.9839	1.8829	3.7387
2048	3.9561	3.7591	7.2972
4096	7.9682	7.5095	14.4239
8192	17.0019	16.0371	30.8756
16384	34.0607	32.1598	61.8142
32768	61.1621	64.31044	123.6229

Table 2 SimOps 1

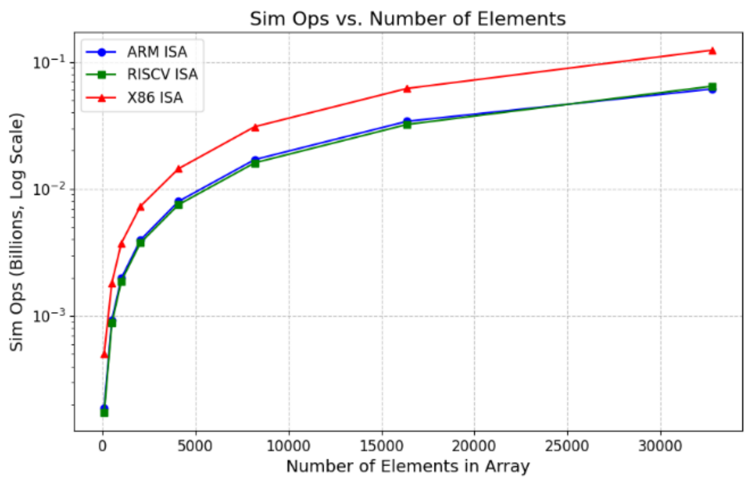

Figure 2: visually represents the fluctuation of Sim-Ticks across different ISA configurations.

a) Histograms to compare results
The Figure 3 shows the histogram for efficient comparison of the ARM, X86 and RISC-V architectures

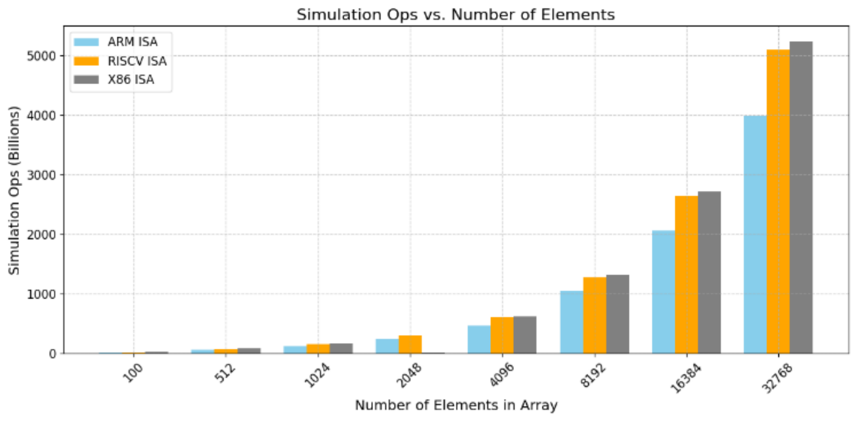

Figure 3

The Figure 4 shows the histogram to compare SimOps

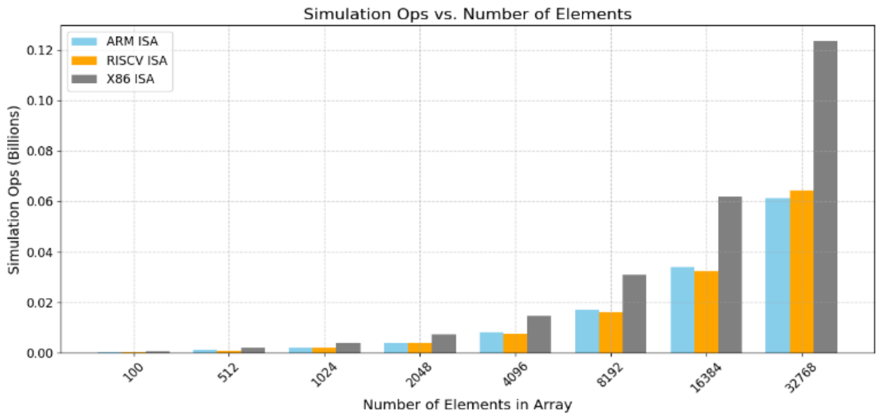

Figure 4

3) CPI (Cycle per Instruction)

We run the Radix sort on different sizes of array then calculate the average CPI (cycle per instruction) of each ISA. The average CPI value for ARM is 65.23, RISC-V 77.01 and for X86 is 83.

The figure 5 shows the histogram to compare the average CPI (Cycle per Instruction) of different ISA's

Cycle per Instruction	ARM	RISCV	X86
1-	65.23	77.01	83

Table 3 CPI

The cycle per instruction (CPI) statistics in gem5 provide critical insights into the efficiency and performance of computer architectures and microarchitectures. CPI is a fundamental metric that quantifies the average number of clock cycles required to execute a single instruction. It serves as a key indicator of how effectively a processor utilizes its computational resources. By analyzing CPI statistics in gem5, researchers and engineers can gain a deeper understanding of the instruction execution efficiency within a given simulation or real-world scenario. These statistics are instrumental in pinpointing performance bottlenecks, optimizing processor designs, and benchmarking various architectural configurations, making CPI a cornerstone metric for performance evaluation and improvement in computer systems.

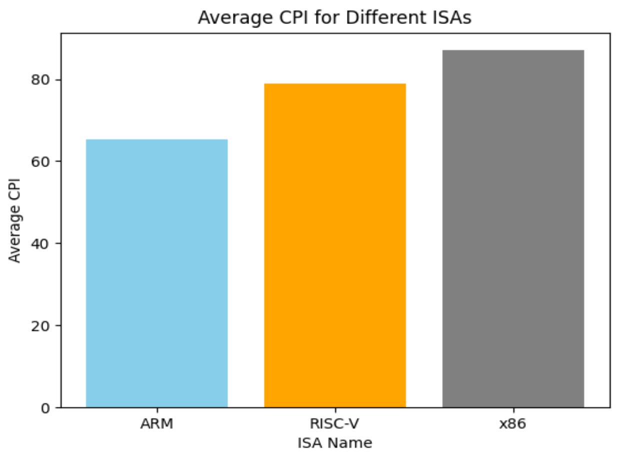

Figure 5

4) Overall performance

Upon a comprehensive examination of various performance metrics and statistics produced by gem5, a clear pattern emerges regarding the performance of the three different Instruction Set Architectures (ISAs).

First and foremost, ARM stands out as the top performer among the trio. It consistently exhibits significantly higher speed and efficiency compared to both RISC-V and X86. This superior performance can be attributed to ARM's architectural advantages and optimizations.

While ARM takes the lead, RISC-V follows closely behind. RISC-V demonstrates commendable performance and efficiency, making it a strong contender. Although it may not match ARM's speed, it still outpaces X86 by a notable margin.

Conversely, X86 emerges as the least performing ISA in this evaluation. The primary factor contributing to X86's slower performance is its reliance on CISC (Complex Instruction Set Computing) architecture. This inherently complex architecture results in slower execution times when compared to the more streamlined and efficient ARM and RISC-V architectures.

In conclusion, the assessment reveals a clear hierarchy in terms of performance, with ARM leading the pack, followed by RISC-V, and X86 trailing as the slowest ISA. These findings underscore the importance of selecting the appropriate ISA for specific computing tasks to achieve optimal performance.

V. Conclusion

In conclusion, this paper presented a comprehensive comparative analysis of three prominent CPU architectures: X86, ARM, and RISC-V. The study focused on evaluating the performance of these architectures through the execution of a specific application, Radix Sort, utilizing a simplified CPU timing model. The key performance parameters investigated included Sim-Ops (simulation operations), Sim- Ticks (simulation ticks), and CPI (cycles per instruction). After an in-depth examination and analysis of these parameters, it is evident that the ARM architecture emerges as the clear leader in terms of speed and efficiency. Following

ARM, the RISC-V architecture demonstrates commendable performance, albeit slightly behind ARM. In contrast, the X86 architecture lags behind both ARM and RISC-V in terms of performance for the Radix Sort application. This research underscores the significance of architecture selection when considering the execution of specific applications. The findings highlight that, for Radix Sort, opting for the ARM architecture would result in the fastest execution, while RISC-V offers a competitive alternative.

VI. Acknowledgements

We extend our heartfelt gratitude to our esteemed mentors, Umer Shahid from the University of Engineering and Technology Lahore and Ayaz Akram from the University of California, Davis, for their unwavering dedication, expert guidance. Their profound knowledge and insightful feedback were instrumental in shaping our research and guiding us towards meaningful outcomes. Furthermore, we wish to express our sincere thanks to the Digital Design Research Lab (DDRC Lab) within the Electrical Engineering Department at the University of Engineering and Technology, Lahore. Their invaluable contributions, including access to cutting-edge resources, technical expertise, and collaborative spirit, have played a pivotal role in our research endeavors. The synergy between our team and DDRC Lab has been integral to the seamless execution and completion of our collective work, underscoring the importance of strong academic partnerships and collaborative research environments in achieving meaningful advancements in our field.

References

1. Butko, A., Garibotti, R., Ost, L., & Sassatelli, G. (2012, July). Accuracy evaluation of gem5 simulator system. In 7th International workshop on reconfigurable and communication-centric systems-on-chip (ReCoSoC) (pp. 1-7). IEEE.
2. Bharadwaj, S. V., & Vudadha, C. K. (2022, December). Evaluation of x86 and ARM architectures using compute-intensive workloads. In 2022 IEEE International Symposium on Smart Electronic Systems (iSES) (pp. 586-589). IEEE.
3. Saha, R., Pundir, Y. P., Yadav, S., & Pal, P. K. (2020, August). Impact of Size, Latency of Cache-L1 and Workload Over System Performance. In 2020 International Conference on Advances in Computing, Communication & Materials (ICACCM) (pp. 390-393). IEEE.
4. George, A. D. (1990, January). An overview of RISC vs. CISC. In Proceedings The Twenty-Second Southeastern Symposium on System Theory (pp. 436-437). IEEE Computer Society.
5. Ling, M., Xu, X., Gu, Y., & Pan, Z. (2019, August). Does the isa really matter? a simulation based investigation. In 2019 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM) (pp. 1-6). IEEE.
6. Jamil, T. (1996). Fifth-generation microprocessors. IEEE Potentials, 15(5), 33-35.
7. Abudaqa, A. A., Al-Kharoubi, T. M., Mudawar, M. F., & Kobilica, A. (2018, May). Simulation of ARM and x86 microprocessors using in-order and out-of-order CPU models with Gem5 simulator. In 2018 5th International Conference on Electrical and Electronic Engineering (ICEEE) (pp. 317-322). IEEE.
8. Vikas, B., & Talawar, B. (2014, December). On the cache behavior of splash-2 benchmarks on arm and alpha processors in gem5 full system simulator. In 2014 3rd International Conference on Eco-friendly Computing and Communication Systems (pp. 5-8). IEEE.