SIMATS ENGINEERING

Saveetha Institute of Medical and Technical Sciences

Chennai-602105

Evaluating the performance of diverse computer architecture

A CAPSTONE PROJECT REPORT

Submitted in the partial fulfilment for the Course of

CSA1209 – Computer Architecture for pipeline process *to the award of the degree of*

BACHELOR OF ENGINEERING IN CSE

BACHELOR OF ENGINEERING IN ECE

BACHELOR OF TECHNOLOGY IN AI&ML

Submitted by

Chethrika Sri (192511112)

Janani J (192512155)

Shaik Arshiya (192525332)

Under the Supervision of

Dr. Kumaragurubaran T

Dr. Senthil Vadivu S

SIMATS ENGINEERING

Saveetha Institute of Medical and Technical Sciences

Chennai-602105

September 2025

SIMATS ENGINEERING

Saveetha Institute of Medical and Technical Sciences

Chennai-602105

Engineer to Excel

DECLARATION

We, Chethrika Sri, Janani J, Shaik Arshiya of Saveetha Institute of Medical and Technical Sciences, Saveetha University, Chennai, hereby declare that the Capstone Project Work entitled "**EVALUATING THE PERFORMANCE OF DIVERSE COMPUTER ARCHITECTURE**' is the result of our own Bonafide efforts. To the best of our knowledge, the work presented herein is original, accurate, and has been carried out in accordance with principles of engineering ethics.

Place: Thandalam , Chennai

Date:

Signature of the Students with Names

Chethrika Sri (192511112)

Janani J (192512155)

Shaik Arshiya (192525332)

SIMATS ENGINEERING

Saveetha Institute of Medical and Technical Sciences

Chennai-602105

Engineer to Excel

BONAFIDE CERTIFICATE

This is to certify that the Capstone Project entitled "**EVALUATING THE PERFORMANCE OF DIVERSE COMPUTER ARCHITECTURES**" has been carried out by Chethrika Sri, Janani, Arshiya under the supervision of Dr**. Kumaragurubaran** and is submitted in partial fulfilment of the requirements for the current semester of the BE -CSE, BE-ECE, BTECH AI&ML program at Saveetha Institute of Medical and Technical Sciences, Chennai.

SIGNATURE
Dr. S. Anushya
Professor
Program Director
Computer Science and Engineering
Saveetha School of Engineering
SIMATS

SIGNATURE
Dr. Kumaragurubaran T
Dr. Senthil Vadivu S
Professor
Computer Science and Engineering
Saveetha School of Engineering
SIMATS

Submitted for the Project work Viva-Voce held on _____

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

We would like to express our heartfelt gratitude to all those who supported and guided us throughout the successful completion of our Capstone Project. We are deeply thankful to our respected Founder and Chancellor, Dr. N.M. Veeraiyan, Saveetha Institute of Medical and Technical Sciences, for his constant encouragement and blessings. We also express our sincere thanks to our Pro-Chancellor, Dr. Deepak Nallaswamy Veeraiyan, and our Vice-Chancellor, Dr. S. Suresh Kumar, for their visionary leadership and moral support during the course of this project.

We are truly grateful to our Director, Dr. Ramya Deepak, SIMATS Engineering, for providing us with the necessary resources and a motivating academic environment. Our special thanks to our Principal, Dr. B. Ramesh for granting us access to the institute's facilities and encouraging us throughout the process. We sincerely thank our Head of the Department, **Dr. S. Anushya** for his continuous support, valuable guidance, and constant motivation.

We are especially indebted to our guide, Dr. Kumaragurubaran T and Dr S Senthil Vadivu for their creative suggestions, consistent feedback, and unwavering support during each stage of the project. We also express our gratitude to the Project Coordinators, Review Panel Members (Internal and External), and the entire faculty team for their constructive feedback and valuable inputs that helped improve the quality of our work. Finally, we thank all faculty members, lab technicians, our parents, and friends for their continuous encouragement and support.

Signature With Students Name

Chethrika Sri (192511112)

Janani J (192512155)

Shaik Arshiya (192525332)

ABSTRACT

The rapid advancement of computing technology has led to the development of diverse computer architectures, each optimized for specific workloads, performance goals, and energy efficiency. This capstone project focuses on the performance analysis of different computer architectures, with the objective of evaluating their strengths, weaknesses, and suitability for various application domains. The primary problem addressed in this study is the lack of clarity on how different architectural designs—such as RISC (Reduced Instruction Set Computing), CISC (Complex Instruction Set Computing), parallel architectures, and modern multi-core processors—affect system performance when subjected to real-world computational tasks. The purpose of this project is to conduct a systematic performance evaluation across multiple architectures by examining critical parameters such as instruction execution speed, throughput, memory access latency, scalability, and energy consumption. Simulation tools, benchmarking software, and experimental setups are employed to compare how different architectures handle workloads ranging from simple arithmetic operations to complex parallel processing tasks Key outcomes of the project highlight that no single architecture universally outperforms the others; instead, performance is highly workload-dependent. RISC architectures excel in efficiency and speed for simpler tasks, while CISC offers advantages in handling complex instructions. Parallel and multi-core designs significantly improve throughput for concurrent workloads but may suffer from synchronization overhead. The findings emphasize the importance of workload-architecture alignment in system design and resource allocation. This project contributes to a deeper understanding of architectural trade-offs, aiding researchers, engineers, and organizations in selecting optimal computing platforms for specific performance requirements.

TABLE OF CONTENTS:

LIST OF TABLES :

LIST OF FIGURES :

CHAPTER 1

INTRODUCTION

1.1 Background Information

In today's digital era, computer systems form the backbone of every sector—from scientific research and engineering design to business applications, artificial intelligence, healthcare, and entertainment. The efficiency, speed, and overall performance of these systems largely depend on their underlying computer architecture, which dictates how instructions are executed, how memory is accessed, and how efficiently tasks are processed.

Over the past several decades, multiple architectural paradigms have emerged. Two of the most prominent instruction set architectures are:

RISC (Reduced Instruction Set Computing): Focuses on simplified instructions that can be executed rapidly, often in a single cycle. This architecture is widely used in mobile devices and embedded systems due to its power efficiency and speed in handling repetitive workloads.

CISC (Complex Instruction Set Computing**):** Incorporates a larger set of complex instructions, allowing each instruction to perform multiple tasks. This design reduces the number of instructions needed for complex operations, making it suitable for general-purpose desktops, servers, and enterprise applications.

Beyond instruction set design, advances such as multi-core processors and parallel architectures (e.g., GPUs) have transformed computing. Multi-core CPUs enable simultaneous execution of tasks, thereby improving throughput and responsiveness. Parallel architectures, particularly GPUs, excel at executing thousands of threads concurrently, making them the foundation of AI model training, graphics rendering, and scientific simulations.

The challenge is that no single architecture consistently outperforms others across all workloads. For example, a GPU may provide enormous speedups for matrix-based AI workloads but performs poorly on sequential applications. Similarly, RISC excels in energy efficiency but may require multiple instructions for tasks that CISC can handle in one. This diversity in performance creates uncertainty when selecting the most appropriate architecture for a given application. Thus, there is a need for a systematic analysis of performance across multiple architectures to guide researchers, engineers, and organizations in making informed. Thus, there is need for computer architecture.

1.2 Project Objectives

The main aim of this project is to evaluate and compare the performance of different computer architectures using simulation tools, benchmark tests, and workload analysis. The specific objectives are:

1. To study and understand the principles of major computer architectures, including RISC, CISC, multi-core, and parallel (GPU-based) systems.
2. To measure and compare performance metrics such as:
   Energy efficiency and power consumption
3. To identify workload–architecture alignments, i.e., determine which types of tasks are best suited for each architecture.
4. To provide a framework that supports decision-making in system design, workload deployment, and resource optimization.
5. To document key findings and propose recommendations for future computing systems.

1.3 Significance of the Project

- For Academia: This study bridges theoretical knowledge of computer architecture with practical performance analysis, offering insights into the trade-offs of different designs. It can serve as a reference for students and researchers exploring system design and high-performance computing.
- For Industry: Organizations rely on computing systems for critical tasks ranging from data analytics to machine learning. By aligning workloads with the most suitable architecture, companies can achieve higher efficiency, reduced costs, and better energy utilization.
- For Society: The efficient use of computer architectures has direct societal benefits, such as reduced carbon footprint through energy-efficient systems, improved healthcare applications (faster data processing for diagnostics), and enhanced user experiences in consumer electronics.

In short, this project emphasizes the importance of architecture-aware workload deployment in an age where computational demands are constantly evolving. This diversity in performance creates uncertainty when selecting the most appropriate architecture for a given

application. The challenge is that no single architecture consistently outperforms others across all workload

In short, this project emphasizes the importance of architecture aware workload deployment.

1.4 Scope of the Project

Included Architectures:

1.RISC (ARM-based processors)

2.CISC (x86 processors)

3.Multi-core processors (Intel/AMD multi-threaded CPUs) ○Parallel architectures (NVIDIA GPUs as representative platforms)

4.Parameters Analyzed:

- Execution speed (runtime performance)
- Instruction throughput (MIPS, FLOPS)
- Memory hierarchy performance (cache hits/misses, latency)
- Scalability (with increasing threads or workload size)
- Energy efficiency (where tools and resources allow)

Scope:

- Emerging or experimental architectures such as quantum computing, neuromorphic computing, or FPGAs (not covered due to resource and time constraints).
- Hardware-level physical design and fabrication (focus is on simulation, benchmarking, and software-level analysis).
- Security and reliability aspects (the study is limited to performance evaluation).

By defining these boundaries, the project ensures a focused and achievable analysis, while leaving space for future extensions.

1.5 Methodology Overview

The methodology adopted for this project follows a systematic sequence:

1. **Literature Review:** Conducted an in-depth study of academic research, industry reports, and benchmark studies related to performance evaluation of architectures.
2. **Architecture Selection:** Chose representative architectures (RISC, CISC, multi-core, GPU).
3. **Benchmark Design & Selection:**

Standard benchmarks (SPEC CPU, LINPACK, Geekbench) were selected.

Custom workloads (matrix multiplication,image processing) were developed.

4. **Simulation & Testing:**

   Employed simulators such as GEM5 and real platforms (ARM boards, x86 PCs, NVIDIA GPUs) where feasible.

5. **Data Collection:**

   Gathered results for metrics like execution time, throughput, scalability, and power consumption.

6. **Analysis:**

   Used statistical tools and visualization (Python's Pandas, NumPy, Matplotlib) to compare performance across architectures.

7. **Validation:**

   Cross-verified results with published studies and industry benchmarks to ensure accuracy.

8. **Documentation & Recommendations:**

   Compiled findings into a comparative framework, highlighting workload-architecture suitability.

CHAPTER 2
PROBLEM IDENTIFICATION AND ANALYSIS

2.1 Description of the Problem

Computing systems have become the backbone of modern society, powering everything from mobile devices and personal computers to large-scale data centers and high-performance supercomputers. At the core of these systems lies the computer architecture, which defines how processors execute instructions, interact with memory, and manage workloads.

However, with the diversity of computing requirements—ranging from lightweight mobile applications to highly parallel scientific simulations—a single architecture cannot optimally serve every domain. As a result, architectures such as x86, ARM, RISC-V, and GPU-based accelerators have evolved, each excelling in different contexts.

The main problem is the lack of systematic, workload-specific, and comparative performance analysis across these architectures. While technical specifications and theoretical models provide insights into capabilities (e.g., instruction set design, pipeline depth, and cache hierarchy), they do not directly translate into real-world performance under varied workloads.

Without structured analysis, decision-makers, researchers, and developers are forced to rely on assumptions or generic benchmarks that may not align with their actual use cases. This results in inefficient resource utilization, suboptimal system design, and missed opportunities for cost and energy savings.

Thus, the central problem this project addresses is:
How can we systematically evaluate and compare the performance of different computer architectures across a variety of workloads to provide reliable, evidence-based insights for academic, industrial, and societal applications?

2.2 Evidence of the Problem

Evidence for the existence and significance of this problem can be observed in several dimensions:

1.**Workload Diversity** Scientific Computing: Requires processors optimized for floating-point operations

- Scientific Computing: Requires processors optimized for floating-point operations, vectorization, and memory bandwidth.
- Business Applications: Rely on integer-heavy operations and high input/output efficiency.
- Artificial Intelligence and Machine Learning: Favor architectures with massive parallelism, such as GPUs or tensor processing units (TPUs).
- Mobile and Embedded Devices: Require architectures that balance performance with energy efficiency and thermal constraints.

Since no single architecture meets all these needs, the performance advantage is context dependent.

2.Industry Inefficiencies

- Cloud providers often deploy general-purpose CPUs (e.g., x86 servers) without fully leveraging workload-specific architectures like ARM-based processors or accelerators, leading to higher operational costs.
- Mobile chip manufacturers emphasize battery life but sometimes sacrifice computational power, making devices unsuitable for heavy workloads.

3.Research Gaps

- Benchmarking studies often use narrow datasets or isolated metrics, resulting in inconsistent findings.
- Contradictions arise: for instance, ARM processors may outperform x86 in energy efficiency but lag behind in raw throughput for server workloads.

4.Supporting Statistics

- Moore's Law slowdown: The traditional reliance on transistor scaling for performance gains has plateaued, making architectural innovations the key driver of performance (Hennessy & Patterson, 2021). Contradictions arise: for instance, ARM processors may

outperform x86 in energy efficiency but lag behind in raw throughput for server (Hennessy & Patterson, 2021). Contradictions arise: for instance, ARM processors may workloads Benchmarking studies often use narrow datasets or isolated metrics, resulting in inconsistent findings.

-
- SPEC Benchmark Results: Demonstrate that no single architecture dominates across all categories—some excel in computation-heavy tasks, while others shine in memory intensive workloads.
- Green Computing Trends: Reports indicate that data centres consume ~1% of global electricity, making architectural efficiency crucial for sustainability.

This evidence highlights the urgency of conducting comprehensive, reproducible performance analyses.

2.3 Stakeholders

The problem directly affects multiple stakeholders, each with distinct needs and perspectives:

1. Academic Stakeholders (Students, Researchers, Educators):
    - Students require practical understanding of architecture to apply theory to real world problems.
    - Researchers depend on accurate benchmarks to test hypotheses about system design.
    - Educators use case studies to teach the trade-offs in architecture design.

2. Industry Stakeholders (Hardware and Software Developers, Cloud Providers, System Integrators):
    - Hardware manufacturers (Intel, AMD, ARM, NVIDIA, RISC-V communities) must validate architectures against real workloads to maintain competitiveness. ○ Software developers need to optimize code for specific hardware (e.g., SIMD vectorization for ARM NEON vs. Intel AVX).
    - Cloud providers (AWS, Azure, Google Cloud) must select architectures that balance cost, scalability, and performance for diverse clients.

3. End-Users and Society:

Society benefits from green computing initiatives , where energy-efficient architectures

- Consumers indirectly benefit from efficient architectures through faster applications, longer device battery life, and reduced service costs. Hardware manufacturers (Intel, AMD, ARM, NVIDIA, RISC-V communities) must
- Society benefits from green computing initiatives, where energy-efficient architectures help mitigate climate impact.

2.4 Supporting Data and Research

Hennessy & Patterson, 2021). Contradictions arise: for instance, ARM processors may

Theoretical Support

- Amdahl's Law: Demonstrates that parallelism has diminishing returns if parts of the workload remain sequential. This highlights the importance of analysing architecture specific execution behaviours.
- Gustafson's Law: Suggests that larger workloads benefit more from parallel architectures, making workload size a key factor in performance evaluation.
- SPEC Benchmarks: Reveal differences in performance between x86 and ARM-based systems across standard workloads.
- Green500 Supercomputer Rankings: Show that architectural choices directly impact energy efficiency, with some ARM-based systems outperforming traditional x86 designs in FLOPS per watt.
- Case Study – Apple M1 (ARM-based): Demonstrated significantly better performance-per-watt compared to Intel x86 chips, reshaping the laptop and mobile processor market.

Industry Practices

- Companies like Google and Amazon invest heavily in custom chips (e.g., Google's TPU, AWS's Graviton ARM processors), emphasizing the importance of architecture specific optimization.
- NVIDIA GPUs dominate AI workloads, but CPUs remain central for general-purpose computing, reinforcing the need for context-aware comparisons.

2.5 Summary of Problem Analysis

- Workload diversity makes one-size-fits-all architectural decisions ineffective.
- Industry practices often lead to inefficiencies and higher costs due to inadequate benchmarking.

CHAPTER 3
SOLUTION DESIGN AND IMPLEMENTATION

3.1 Development and Design Process

The development of this project followed a systematic and structured process to ensure reliability, reproducibility, and alignment with academic and industry standards. The process was divided into multiple phases:

1. Problem Definition and Requirement Analysis:
    - Identified the need for comparing different computer architectures based on performance. o Determined the performance metrics to evaluate, such as latency, throughput, CPI (Cycles per Instruction), FLOPS (Floating Point Operations per Second), scalability, and energy efficiency.
    - Defined the workloads and benchmarks that would best represent real-world applications (scientific computing, machine learning, business transactions, embedded systems).
2. System Design and Planning:
    - Designed a framework that included benchmark selection, simulation setup, test environment configuration, and data collection methodology.
    - Established criteria for fair comparisons (e.g., normalizing results across clock speeds and hardware configurations).
3. Implementation Phase:
    - o Installed and configured benchmarking tools on different architectures (e.g., x86-based processors, ARM-based processors, and simulated RISC-V environments).
    - o Executed a series of controlled tests to measure architecture-specific performance under varying workloads.
    - o Automated data logging and collection to minimize human error.
4. Analysis and Validation:
    - Processed results using statistical techniques and visualization tools.
    - Cross-validated benchmark outcomes with theoretical expectations (e.g., pipelining behaviour, memory hierarchy performance).

- Ensured results were consistent by repeating tests under different conditions.

5. Documentation and Reporting:
    - Structured all findings into comparative performance reports.
    - Created visual representations (graphs, tables, charts) to make results easier to interpret.
    - Documented methodologies for reproducibility, aligning with engineering best practices.

This process ensured that the project-maintained clarity, accuracy, and repeatability, making it a valuable contribution to both academic and industrial stakeholders.

3.2 Tools and Technologies Used

The project made use of several tools and technologies, categorized into benchmarking tools, simulation platforms, programming languages, and supporting software:

1.Benchmarking Suites:

- SPEC CPU – for evaluating integer and floating-point performance.
- LINPACK – for measuring floating-point computation capacity. ○ STREAM – for analysing memory bandwidth.
- Custom Microbenchmarks – written to test cache performance, instruction throughput, and memory latency.

2.Simulation Platforms:

- GEM5 Simulator – used for detailed architectural simulations and analyzing hypothetical processor designs.
- QEMU – for virtualizing different architectures and running workload simulations.

3.Programming Languages and Libraries:

- Python – for automation, data collection, and visualization.
- C and C++ – for writing low-level performance tests and benchmarks.
- NumPy, Pandas, and Matplotlib – for data analysis and visualization.

4.Version Control and Collaboration Tools:

- Git and GitHub – for managing code, ensuring version control, and enabling collaborative development.
- LaTeX/MS Word – for preparing technical documentation and reports. By using a combination of industry-grade and open-source tools, the project ensured both practical applicability and academic rigor.

3.3 Solution Overview 7

The solution was designed as a comparative benchmarking framework capable of systematically analysing different computer architectures under controlled conditions.

Architecture Selection

- x86 (Intel/AMD): General-purpose processors widely used in desktops and servers.
- ARM: Known for energy efficiency and dominance in mobile/embedded systems.
- RISC-V: An emerging open-source ISA, tested through simulation for its potential.
- GPU (NVIDIA): Evaluated for parallel computing tasks relevant to AI/ML workloads.

Performance Metrics

The following metrics were used to assess performance:

- Instruction-level performance: CPI, IPC (Instructions per Cycle).
- Memory hierarchy performance: Cache hit/miss rates, memory latency.
- Execution speed: Runtime for standard benchmarks.
- Scalability: Performance scaling with increased workloads.
- Energy efficiency: Power consumption vs. performance ratio.

System Workflow

1. Workload Selection: Representative workloads were selected from standard benchmark
2. Benchmark Execution: Benchmarks were executed on multiple architectures under identical software environments.
3. Data Logging: Results were collected in structured formats.
4. Analysis: Data was processed to generate comparative graphs.
5. Interpretation: Results were analysed against theoretical expectations
6. Data Logging: Results were collected in structured formats.

This framework allowed for comprehensive and reproducible performance evaluation, bridging the gap between theoretical models and practical insights.

3.4 Engineering Standards Applied

The project adhered to several engineering standards and best practices to ensure reliability, reproducibility, and professionalism:

1.IEEE Standards:

- Followed IEEE 754 standard for floating-point arithmetic consistency across platforms. ○ Applied IEEE Software Engineering Guidelines for structured development, modular code, and documentation.

2.ISO Standards:

- Considered ISO/IEC 25010:2011 (Software Product Quality Model) for evaluating software tools used in benchmarking. ○ Applied ISO/IEC 2382 terminology for consistent use of computing terms.

3.Benchmarking Standards:

- Followed SPEC guidelines for running benchmarks to ensure fairness and comparability.
- Used Green500 energy efficiency benchmarks as a reference for evaluating FLOPS per watt.

4.Data Integrity Standards:

- Ensured data reproducibility by maintaining logs and scripts following FAIR (Findable, Accessible, Interoperable, Reusable) data principles.

By aligning with recognized standards, the project ensured scientific validity, industry relevance, and ethical responsibility.

3.5 Solution Justification

The inclusion of engineering standards and structured methodologies significantly impacted the design quality, reliability, and success of the project:

- Reliability and Reproducibility: Following IEEE and SPEC benchmarking protocols guaranteed that results were consistent and reproducible, reducing experimental bias.
- Credibility: Adherence to ISO and IEEE standards increased the academic and professional credibility of the study
- Reliability and Reproducibility: Following IEEE and SPEC benchmarking protocols guaranteed that results were consistent and reproducible, reducing experimental bias.

- Fairness in Comparison: By normalizing workloads and using standardized benchmarks, the project avoided unfair biases between architectures with different configurations.
- Industry Relevance: Using widely accepted practices ensured that the project outcomes could be directly applicable to data centres, software optimization, and chip design decisions.
- Long-Term Value: Since RISC-V is emerging, adhering to open standards ensured that the project's methodology could be reused in future evaluations.

Ultimately, these choices ensured that the project was not just a technical experiment, but a robust, industry-aligned study that could inform academic learning, industrial decisions, and future research directions.

CHAPTER 4
RESULTS AND RECOMMENDATIONS

4.1 Evaluation of Results

The performance evaluation framework provided a clear comparison of RISC, CISC, multicore, and parallel architectures across multiple workload categories. The analysis yielded the following insights:

- RISC (ARM-based processors): Demonstrated superior performance in terms of energy efficiency and execution speed for simple, repetitive tasks. They consumed less power, making them ideal for mobile and embedded systems.
- CISC (x86 processors): Outperformed RISC in handling complex instructions and memory-intensive workloads due to its ability to execute more functionality per instruction. However, it required higher energy consumption.
- Multi-core CPUs: Showed significant improvements in throughput for concurrent workloads, particularly when applications were optimized for parallel processing. The performance scaled well with increasing thread count, though overheads limited scalability in certain tasks.
- GPUs (parallel architectures): Delivered the highest speedups for massively parallel workloads such as image processing and deep learning but were inefficient for sequential operations. Overall, the framework successfully demonstrated that no single architecture dominates across all workload types, confirming the importance of workload-architecture alignment.

4.2 Challenges Encountered

Several challenges were faced during the project:

- **Simulation Complexity:** Configuring tools such as GEM5 and SPEC required extensive setup and resource allocation. This was mitigated by starting with smaller micro-benchmarks before scaling to full benchmarks.
- The performance scaled well with increasing thread count, though overheads limited scalability in certain tasks.

- **Hardware Limitations:** Testing across diverse hardware (ARM, Intel, GPUs) required access to multiple platforms. This was addressed by using publicly available benchmark data where direct testing was not feasible.
- **Performance Variability:** Benchmark results fluctuated due to background system processes and environmental conditions. To overcome this, each experiment was repeated multiple times, and averages were used for analysis.
- **Energy Measurement:** Accurate energy consumption data collection was difficult. Tools such as Intel Power Gadget and NVIDIA SMI were used to approximate power usage.

4.3 Possible Improvements

Inclusion of More Architectures: Expanding the study to cover newer architectures such as RISC-V or specialized accelerators (TPUs, FPGAs) would provide broader insights.

- Hardware Counters: Using low-level hardware counters (e.g., cache miss rates, branch prediction statistics) could improve the depth of analysis.
- Automation: Automating the benchmarking framework with scripts would reduce manual intervention and improve consistency.
- Real-World Applications: Incorporating domain-specific workloads (e.g., AI training, database queries, gaming workloads) could make the findings more industry-relevant.

4.4 Recommendations

Based on the findings, the following recommendations are proposed:

- **For Researchers:** Explore emerging architectures such as RISC-V and domain-specific accelerators to assess their competitiveness.
- **For Industry Practitioners:** Select architectures based on workload characteristics—RISC for low-power embedded applications, CISC for desktop/server environments, multi-core for general-purpose high-performance tasks, and GPUs for AI/parallel workloads.
- **For Developers:** Optimize software applications to exploit the strengths of target architectures (e.g., multi-threading for multi-core CPUs, CUDA/OpenCL for GPUs).

CHAPTER 5
REFLECTION ON LEARNING AND PERSONAL DEVELOPMENT

5.1 Key Learning Outcomes

Academic Knowledge

This capstone project provided me with the opportunity to integrate and apply theoretical knowledge from computer architecture, performance evaluation, and system design. I deepened my understanding of architectural paradigms such as RISC, CISC, multi-core, and parallel processing and how they influence performance in real-world applications. The study of performance metrics like MIPS, FLOPS, cache efficiency, and energy consumption enabled me to connect classroom concepts with practical benchmarking. This project helped bridge the gap between theory and practice, solidifying my grasp of computer architecture principles.

Technical Skills

Throughout the project, I developed strong technical competencies. I gained hands-on experience with benchmarking tools (SPEC CPU, LINPACK, Geek bench), simulation environments (GEM5, Simics), and programming in Python, C, and C++. I also enhanced my skills in data analysis and visualization using libraries such as NumPy, Pandas, and Matplotlib. In addition, I learned how to configure and run benchmarks across diverse hardware platforms (ARM, Intel x86, and GPUs), which strengthened my ability to work with industry standard evaluation tools.

Problem-Solving and Critical Thinking

The project demanded critical thinking to address complex issues such as performance variability, energy measurement difficulties, and architectural trade-offs. I learned to design experiments that minimized bias, validated results with repeated trials, and interpreted conflicting data logically. These experiences sharpened my ability to analyse problems systematically and think critically when faced with uncertainty. I learned that decisions are not based solely on raw performance, but also on factors such as energy efficiency, cost, scalability, and workload suitability. For example, the industry trend of moving towards ARM--based processors in mobile and IoT devices and GPU adoption in AI workloads reflects the same .

5.2 Challenges Encountered and Overcome

Personal and Professional Growth

One of the biggest challenges I faced was the steep learning curve of simulation tools like GEM5, which required detailed configuration and understanding of low-level system behaviour. Initially, this was frustrating, but persistence and self-learning enabled me to overcome it. Another challenge was accessing diverse hardware platforms for testing; I resolved this by combining direct experiments with published benchmark results. These experiences taught me resilience, adaptability, and the importance of resourcefulness in professional work.

Collaboration and Communication

Although much of the project was independent, I interacted with peers, mentors, and faculty members for guidance. Communicating my methodology and results required me to refine my technical writing and presentation skills. Explaining complex architectural differences in clear and concise terms improved my ability to communicate technical concepts effectively. If this had been a team project, I believe my experience would have enhanced my ability to collaborate, delegate tasks, and share ideas constructively.

5.3 Application of Engineering Standards

Applying engineering standards was crucial in ensuring the reliability of my results. Using IEEE 754 for floating-point operations, ISO/IEC C standards for programming consistency, and SPEC benchmarks for performance testing provided a standardized framework that made my analysis credible and comparable to professional industry practices. Following these standards taught me the importance of consistency, reproducibility, and industry alignment in engineering projects.comparable to professional industry I learned the decisions are not based data logically. framework that made my analysis credible and comparable to professional industry practices

5.4 Insights into the Industry

This project provided valuable insights into how industry professionals evaluate and select computing systems. I learned that decisions are not based solely on raw performance, but also on factors such as energy efficiency, cost, scalability, and workload suitability.

For example, the industry trend of moving towards ARM-based processors in mobile and IoT devices and GPU adoption in AI workloads reflects the same trade-offs I analysed. This reinforced my awareness of real-world industry practices and the importance of aligning technical solutions with business and operational needs.

5.5 Conclusion of Personal Development

Overall, this capstone project has been a transformative learning experience. It not only expanded my academic knowledge but also enhanced my technical, analytical, and professional skills. I developed confidence in working with advanced tools, strengthened **my** ability to solve complex problems, and gained a deeper understanding of how theory translates into practice.

Most importantly, it helped me clarify my career direction, as I now feel more inclined toward roles in system design, performance engineering, and research in emerging architectures. This project has prepared me to contribute meaningfully to both academic research and industry applications, while continuously striving for innovation and efficiency.

At the start, my knowledge of computer architectures was limited to theoretical aspects such as instruction cycles, memory hierarchy, and basic CPU design. However, through practical analysis of different architectures—such as RISC vs. CISC, single-core vs. multi-core processors, and GPU vs. CPU processing—I developed a deeper appreciation for how architectural choices directly impact system performance.

One of the key learning outcomes was understanding performance metrics such as *CPI* (Cycles per Instruction), MIPS (Million Instructions per Second), throughput, and latency**.** Applying these metrics in real-world comparisons allowed me to critically evaluate why certain architectures perform better in specific contexts (e.g., GPUs for parallel tasks vs. CPUs for sequential processing).

From a personal development perspective, this process enhanced my analytical thinking and problem-solving skills. Instead of passively accepting performance results, I learned to interpret benchmarks, identify bottlenecks, and justify why one architecture outperforms

another. It also strengthened my ability to use tools and simulators for modelling performance, improving my practical competence in system evaluation.

Moreover, this learning journey improved my research and adaptability skills. Since architectures evolve rapidly, I had to consult up-to-date resources, analysed case studies, and adapt to new terminologies. This helped me build confidence in self-directed learning and staying current in a fast-changing field.

Finally, working on performance analysis also boosted my communication skills, as I often had to present findings in structured reports and discussions. Explaining complex architectural concepts in simple terms to peers taught me the importance of clarity and technical storytelling.

CHAPTER 6
CONCLUSION

The purpose of this capstone project was to investigate and analyse the performance of different computer architectures—specifically RISC, CISC, multi-core, and parallel architectures— to identify their strengths, weaknesses, and suitability for various workloads. The problem addressed was the lack of clarity in selecting the most appropriate architecture for specific computational demands, which often results in inefficiency, higher costs, and underutilization of resources. The solution involved developing a comparative performance evaluation framework that utilized simulation tools, benchmarking software, and experimental workloads to measure performance across key parameters such as execution speed, throughput, memory access efficiency, and energy consumption. The findings clearly demonstrated that no single architecture outperforms all others in every context. Instead, performance is work load dependent:

- RISC architectures (e.g., ARM) are highly energy-efficient and excel in mobile and embedded systems.
- CISC architectures (e.g., x86) handle complex instructions effectively, making them suitable for desktops and servers.
- Multi-core processors improve throughput for concurrent workloads but require optimized software to maximize benefits.
- Parallel architectures (GPUs) provide massive speedups for AI, graphics, and scientific workloads but are inefficient for sequential tasks.

The significance of this project lies in its contribution to understanding architectural trade-offs. By aligning workloads with the most appropriate architectures, organizations and researchers can achieve better performance, energy efficiency, and cost-effectiveness. The application of engineering standards further ensured reliability, reproducibility, and industry relevance of the results. In conclusion, this project highlights the importance of informed architecture selection in the evolving landscape of computing. The insights gained not only address the immediate problem but also provide valuable guidance for future research, system design, and industry adoption

REFERENCES

- Standard Performance Evaluation Corporation (SPEC). (2023). *SPEC CPU benchmarks*. Retrieved from https://www.spec.org/cpu/
- SPEC CPU Benchmark Suite. (2022). Standard Performance Evaluation Corporation. Retrieved from https://www.spec.org/cpu/
- Intel Corporation. (2022). *Intel architecture instruction set extensions and performance benchmarks*. Retrieved from https://www.intel.com
- ARM Holdings. (2022). *ARM architecture reference manual*. Retrieved from https://developer.arm.com
- Hennessy, J. L., & Patterson, D. A. (2019). *Computer architecture: A quantitative approach* (6th ed.). Morgan Kaufmann.

APPENDICES

Appendix A: Abbreviations

CPU – Central Processing Unit
GPU – Graphics Processing Unit
CPI – Cycles Per Instruction
MIPS – Million Instructions Per Second
MFLOPS – Million Floating-Point Operations Per Second
RISC – Reduced Instruction Set Computer
CISC – Complex Instruction Set Computer
ILP – Instruction-Level Parallelism
DLP – Data-Level Parallelism
TLP – Thread-Level Parallelism

Appendix B: Performance Formulas

1. CPU Execution Time = (Instruction Count × CPI) ÷ Clock Rate
2. MIPS = (Instruction Count ÷ Execution Time) ÷ 10^6

3. MFLOPS = Floating Point Operations ÷ (Execution Time × 10^6)

4. Speedup = Performance of New System ÷ Performance of Old System

Appendix C: Benchmark Examples

SPEC CPU – Measures processor-intensive workloads.

LINPACK – Evaluates floating-point computing power.

TPC-C – Measures transaction processing capability.

PARSEC – Tests parallel architectures.

Appendix D: Case Study Architectures

8085 – Basic microprocessor, used for foundational study.

8086 – 16-bit architecture, improved addressing modes.

ARM – RISC-based mobile and embedded processor.

x86 – CISC-based widely used in desktops/servers.

GPU (NVIDIA/AMD) – Highly parallel processors optimized for graphics and AI workloads.

Appendix E: Example Tables

Table E.1 – Comparison of Performance Metrics

Architecture	**CPI**	**Clock Speed**	**Benchmark Score**	**Power Efficiency**
RISC (ARM)	1.2	2.5 GHz	High on SPEC	Very High
CISC (x86)	1.8	3.2 GHz	High on SPEC	Moderate
GPU	2.5	1.5 GHz	Very High (LINPACK)	High

EVALUATING THE PERFORMANCE OF DIVERSE COMPUTER ARCHITECTURE