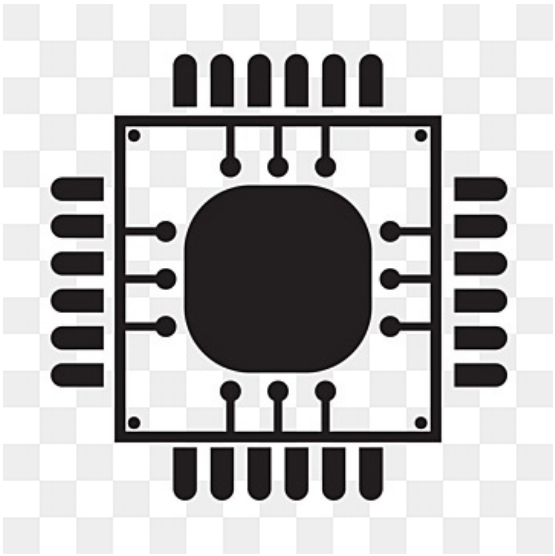
Multiarchitecture Virtualized Benchmark Analysis

ECE 4300



Team Members:
Isaac Blekhman, Hyosung Kim, Caden Nihart, Ricardo Godinez

Faculty Advisor:
Professor Mohammed El Hadedy Aly
Email: maly@cpp.edu

California State Polytechnic University, Pomona Department of Electrical and Computer Engineering, College of Engineering

November 29, 2024

CONTENTS

***Abstract*—This study analyzes the performance of modern CPU architectures using the SHA-3 cryptographic hashing algorithm as a benchmark. We assessed various processors, including x86-based Intel and AMD CPUs—specifically the Ryzen 7 7840H, Intel i7-8750H, i7-1355U, and i7-8665U—and the ARM Cortex-A72 in the Raspberry Pi 4. By standardizing testing conditions and using virtual machines, we highlight the strengths and weaknesses of each architecture in single-core and multi-core workloads. While x86 processors showed better computational performance, the Raspberry Pi was notably efficient given its resource constraints. The study also discusses challenges in benchmarking within virtualized environments and offers recommendations to improve methodologies, incorporate diverse workloads, and address scalability challenges through principles like Amdahl's Law. [1] [2]**

***Index Terms*—SHA-3, CPU Benchmarking, ARM vs. x86, Cryptographic Performance, Virtualization**

I. Introduction

The rapid growth of data and increasing sophistication of cyber threats have amplified the need for strong cryptographic algorithms. Secure data transmission, authentication protocols, and information integrity are essential to modern computing. The Secure Hash Algorithm 3 (SHA-3), standardized as FIPS 202 by the National Institute of Standards and Technology (NIST), represents a major advancement in cryptographic hash functions. Its unique sponge construction and strong resistance to collision and pre-image attacks provide enhanced security over earlier versions. [3]

SHA-3 is also useful for evaluating CPU performance due to its computational complexity and memory usage patterns. The algorithm's reliance on bitwise operations and substitutions requires efficient integer arithmetic, making it a good benchmark for analyzing architectural efficiencies and instruction set optimizations. [4]

This study uses SHA-3 to benchmark various CPUs, including high-performance x86 processors from Intel and AMD—the Ryzen 7 7840H, Intel Core i7-8750H, i7-1355U, and i7-8665U—as well as the ARM Cortex-A72 processor in the Raspberry Pi 4. The ARM Cortex-A72 highlights energy efficiency, contrasting with the high-performance focus of x86 processors. By standardizing testing conditions across all platforms, we aim for an equitable comparison of these architectures. [5]

The analysis encompasses both single-core and multi-core workloads, with objectives to:

- **Quantify Performance Across Architectures:** Evaluate the computational performance of various CPU architectures under standardized conditions using the SHA-3 algorithm as a benchmark.
- **Highlight Architectural Advantages and Limitations:** Analyze the intrinsic strengths and weaknesses of each CPU architecture, with a focus on computational efficiency, energy consumption, and scalability in both single-threaded and multi-threaded scenarios.
- **Investigate Virtualization Challenges:** Evaluate the effects of virtualization on performance metrics, including any anomalies or overhead that arise from executing benchmarks within virtual machines (VMs).
- **Enhance Benchmarking Methodologies:**Propose enhancements to benchmarking practices based on identified challenges, incorporating diverse workloads and theoretical models like Amdahl's Law to tackle scalability and parallelism limits.

This performance analysis provides insights for system architects, developers, and researchers. Understanding the performance characteristics of various CPU architectures is essential for optimizing software, making informed hardware choices, and designing future computing systems that balance performance and efficiency.

II. Test Methodology

Ensuring fairness and consistency was vital in this benchmarking effort. Variations in hardware, software, or testing procedures can distort results. To combat this, we standardized all testing aspects to ensure that performance differences accurately reflect architectural variations.

A. Hardware Configuration

All CPUs were provided with identical resource allocations and operating conditions to establish a level playing field:

- **RAM Allocation:** Each system was provided with 4 GB of RAM, which was adequate to handle the SHA-3 computational workload without causing memory bottlenecks or performance limitations.
- **CPU Clock Frequencies:** CPUs operated at their maximum frequencies set by the system or hypervisor. In virtualized environments, we assigned virtual CPUs to dedicated physical cores to reduce the impact of hypervisor scheduling and context switching. [6]
- **Storage and Peripherals:** Benchmarking was conducted on solid-state drives (SSDs) to minimize input/output (I/O) latency and variability that could be induced by hard-disk drives(HDDs).
- **Virtualization Considerations:** For CPUs tested within virtual machines, the virtualization overhead was acknowledged. We attempted to standardize our deployment across the test devices to avoid discrepancies in resources.

B. Software Environment

A consistent software environment was essential to avoid discrepancies caused by variations in operating systems, libraries, or software configurations.

- **Operating System:**All tests were conducted on Raspberry Pi OS Desktop, a Debian-based Linux distribution supporting both ARM and x86 architectures, ensuring consistency in kernel versions, libraries, and software packages.
- **Software Libraries:**The OpenSSL library was used for SHA-3 hashing to ensure consistency across systems. It was chosen for its widespread use, optimization for various architectures, and reliable cryptographic implementation.

- **Benchmarking Script:** A custom Bash script was developed to automate the benchmarking process. This script controlled CPU utilization levels—100%, 75%, 50%, and 25%—by adjusting thread counts and workload distribution. It ensured that the computational tasks were executed uniformly across all systems.
- **System Services and Background Processes:** Non-essential services and background processes were turned off to reduce their impact on CPU and memory resources. The systems operated in a minimal state, concentrating resources on the benchmarking tasks. [7]

C. Test Repetition and Validation

To enhance the reliability and statistical significance of the results, we implemented rigorous test repetition and validation procedures:

- **Multiple Iterations:** Each test scenario was executed five times on each CPU. Repeating the tests allowed us to account for transient fluctuations in system performance and provided a dataset for statistical analysis.
- **Result Averaging:** The results from the five iterations were averaged to produce a mean performance metric for each test condition. This approach reduced the impact of any single anomalous result on the overall findings.
- **Anomaly Investigation:** Outliers or anomalies in the data were thoroughly examined. Factors such as thermal throttling, background process interference, and hardware inconsistencies were evaluated as potential causes.
- **Result Verification:** The correctness of SHA-3 outputs was verified against known hash values to ensure the integrity of the computations. This step confirmed that all systems were performing the computations accurately, not just quickly.

To ensure fair and meaningful performance comparisons, we used a strict testing methodology with standardized hardware and software. This provided a solid basis for analyzing the key performance characteristics of each CPU architecture.

III. SHA-3: Mechanism and Benchmarking Relevance

A. How SHA-3 Works

SHA-3 is an important advancement in cryptographic hash functions, based on the Keccak algorithm. It uses a sponge construction, which differs from the Merkle-Damgård structures of earlier SHA algorithms, offering enhanced flexibility and security against various cryptographic attacks. [2]

SHA-3's unique characteristics make it particularly well-suited for benchmarking CPU performance across different architectures:

- **Arithmetic and Logical Stress Testing:** The algorithm's reliance on bitwise operations evaluates the CPU's integer execution units and pipeline efficiency. CPUs with optimized instruction sets for bit manipulation, such as specialized XOR and rotate instructions, can demonstrate their advantages. [7]
- **Memory Hierarchy Evaluation:** SHA-3's frequent state updates and irregular memory access patterns place stress on the cache hierarchy and memory controllers. This can reveal differences in cache sizes, associativity, and latency, as well as the effectiveness of prefetching mechanisms.
- **Parallelism and Multi-Core Scalability:** Although SHA-3 is fundamentally sequential at a low level, it can be parallelized at higher levels by processing multiple messages simultaneously or by distributing the workload across multiple cores. This enables the evaluation of a CPU's capability to manage parallel tasks and the effectiveness of its threading mechanisms. [7]
- **Instruction Set Utilization:**Different architectures may vary in their support for advanced instructions, such as SIMD extensions like AVX on x86 and NEON on ARM. Benchmarking with SHA-3 can demonstrate how effectively these instructions are utilized by the compiler and runtime environment.
- **Isolation of CPU Performance:** SHA-3 is computationally intensive and has minimal dependence on other system components like disk I/O or network interfaces. This isolates the CPU as the primary factor affecting performance, allowing for clearer comparisons between different architectures.

Utilizing SHA-3 as a benchmarking tool enables a comprehensive evaluation of CPU architectures. It measures raw computational power, efficiency in processing complex algorithms, and the effectiveness of architectural optimizations. The data collected from this benchmarking process can offer valuable insights that guide decisions in system design, application development, and hardware procurement. [8]

IV. Performance Analysis

A. Analysis

1) ARM Cortex-A72 Efficiency: The Raspberry Pi 4's ARM Cortex-A72 processor showed impressive efficiency in our benchmarks. With a maximum frequency of 1.5 GHz and four cores, it achieved an average single-core score of 2,183.42 at full utilization and completed the SHA-3 hashing task in 32.47 seconds. While it is slower than x86 processors, its performance is noteworthy due to its low power consumption and thermal output.

2) Intel Dominance in Raw Performance: Among the tested processors, Intel's Core i7-1355U showed the best performance in both single-core and multi-core workloads. It reached a maximum frequency of 5 GHz and 10 cores, achieving an average single-core score of 4,575.55 and completing the SHA-3 hashing task in 14.63 seconds. This performance results from Intel's microarchitecture optimizations, higher Instructions Per Cycle (IPC), and effective branch prediction.

In multi-core tests, the i7-1355U maintained its excellence with an average score of 9,508.80, completing computations in 6.31 seconds using four cores. Technologies like Hyper-Threading and Advanced Encryption Standard New Instructions (AES-NI) help it efficiently manage both single-threaded

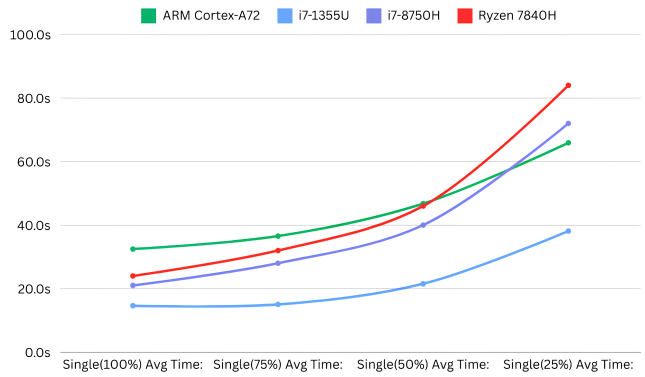


Fig. 1. Single-Core Performance Comparison

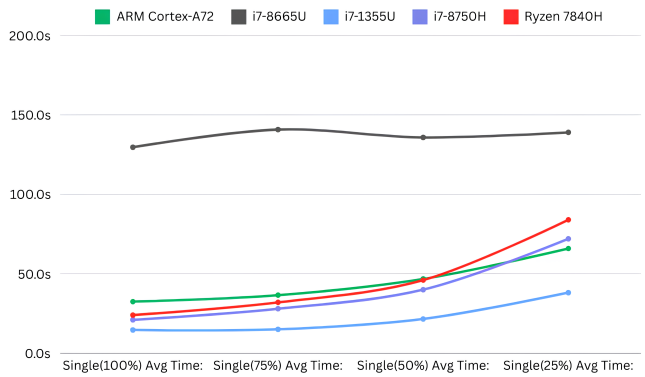


Fig. 2. Anomalous Performance of Intel Core i7-8665U

and parallel workloads, reflecting Intel's focus on maximizing computational performance.

3) Anomalies and Outliers: The Intel Core i7-8665U exhibited unusual performance during testing, primarily due to user error. It recorded an average computation time of 129.65 seconds and a low average score of 1,730.45. These results are unexpected for a CPU that has a maximum frequency of 4.8 GHz and six cores.

The most plausible explanation for the i7-8665U's underperformance—showing computation times five times slower than its competitor, the i7-8750H from the same generation—suggests that the tests or the virtual machine were likely set up incorrectly, and therefore, the results should be discarded.2

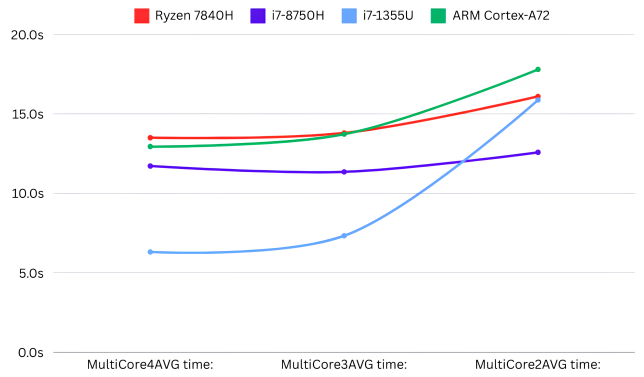


Fig. 3. Multi-Core Performance Comparison

V. Architectural Comparisons: Intel, AMD, and ARM

A. Intel vs. AMD

In comparing Intel and AMD processors, Intel consistently outperforms AMD in single-threaded tasks. For instance, the Intel Core i7-1355U and i7-8750H achieve higher single-core scores and faster computation times than the AMD Ryzen 7 7840H due to Intel's superior instructions per cycle (IPC) and branch prediction.

Conversely, the AMD Ryzen 7 7840H excels in multi-core performance, benefiting from its eight-core configuration and higher base frequencies. In tests using four cores, the Ryzen averaged a score of 4,450.08 with a computation time of 13.48 seconds, demonstrating its suitability for parallel workloads.

In summary, while Intel leads in single-threaded performance, AMD offers competitive multi-core capabilities, making both a viable option for different computing needs.

B. ARM's Role in Modern Computing

The ARM Cortex-A72's performance illustrates its efficiency-oriented design and its role in modern computing environments that prioritize energy efficiency over raw performance. While it lags behind the x86 processors in absolute performance metrics, the Cortex-A72 achieved respectable scores given its lower clock speed and core count.

The ARM processor's multi-core performance, with an average score of 4,535.93 and computation time of 12.94 seconds when utilizing four cores, highlights its ability to handle multi-threaded tasks efficiently within its architectural constraints. This efficiency makes ARM processors ideal for applications in mobile devices, embedded systems, and IoT devices, where power consumption and thermal management are critical considerations.

The results reinforce ARM's potential for lightweight, scalable applications and its growing influence in sectors where energy efficiency is paramount.

VI. Challenges in Virtualization

Benchmarking in virtualized environments introduces several complexities that can affect the accuracy of performance measurements:

- **Frequency Throttling:** Virtual machines may restrict CPUs to operate below their physical maximum frequencies, resulting in skewed results. This was observed in the lower virtualized frequencies of the tested processors.
- **Resource Contention:** Sharing resources among multiple virtual machines can result in inconsistent performance due to competition for CPU time, memory bandwidth, and I/O operations.
- **Hypervisor Overhead:** The hypervisor layer adds extra overhead, which can affect CPU performance, especially in tasks that require direct hardware access.

These factors emphasize the limitations of virtualization in performance benchmarking. For more accurate and reliable results, it is advisable to conduct benchmarks on native hardware.

VII. Recommendations for Improvement

A. Expanding the Benchmark Suite

Including a broader range of workloads, such as matrix multiplications, machine learning algorithms, and database transactions, would enhance the analysis of CPU performance across different computational domains, including floating-point operations and memory-intensive tasks.

B. Applying Amdahl's Law

Future research should investigate workloads with varying levels of parallelizability to understand scalability limits defined by Amdahl's Law. This analysis can reveal how performance improves with more processor cores and inform optimization strategies for multi-core systems.

C. Conducting Native Testing

To eliminate variability from virtualization, benchmarks should be conducted in native environments for direct access to hardware resources and a more accurate representation of processor capabilities.

D. Standardizing Testing Conditions

Ensuring that all tests are performed under identical conditions is crucial for fair comparisons. This includes using the same operating system versions, disabling unnecessary background processes, and synchronizing CPU settings to prevent dynamic frequency scaling from affecting results.

VIII. Conclusion

This study analyzes the performance of modern CPU architectures using the SHA-3 cryptographic hashing algorithm. It highlights the strengths and weaknesses of x86-based Intel and AMD CPUs, alongside the ARM Cortex-A72 processor.

Intel processors excelled in single-threaded tasks due to higher instructions per cycle (IPC) and architectural optimizations. In contrast, AMD's Ryzen 7 7840H performed well in multi-core tasks, making it suitable for parallel workloads. The ARM Cortex-A72 demonstrated impressive efficiency within its power and thermal limits.

Anomalies with the Intel Core i7-8665U revealed the difficulties of benchmarking in virtualized environments, where resource contention, hypervisor overhead and configuration can skew results. Native testing is recommended for accurate performance assessments.

In summary, the study emphasizes choosing the right CPU architecture based on application needs and the necessity for robust benchmarking methodologies. Future research should aim to broaden the benchmark suite and include energy efficiency metrics while addressing scalability through theoretical models like Amdahl's Law.

REFERENCES

[1] K. Kobayashi, J. Ikegami, M. Knežević, E. Xu Guo, S. Matsuo, S. Huang, L. Nazhandali, Kocabaş, J. Fan, A. Satoh, I. Verbauwhede, K. Sakiyama, and K. Ohta, "Prototyping platform for performance evaluation of sha-3 candidates," in *2010 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, 2010, pp. 60–63.

[2] J. Ledin and D. Farley, 2022.

[3] K. E. Ahmed and M. M. Farag, "Hardware/software co-design of a dynamically configurable sha-3 system-on-chip (soc)," in *2015 IEEE International Conference on Electronics, Circuits, and Systems (ICECS)*, Dec 2015, pp. 617–620.

[4] Y. Akiya, K. T. Le, M. Luong, J. C. Wilson, A. S. Eddin, V. Formicola, and M. El-Hadedy, "Sha-3-lphp: Hardware acceleration of sha-3 for low-power high-performance systems," in *2021 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, 2021, pp. 393–398.

[5] E. Blem, J. Menon, and K. Sankaralingam, "Power struggles: Revisiting the risc vs. cisc debate on contemporary arm and x86 architectures," in *2013 IEEE 19th International Symposium on High Performance Computer Architecture (HPCA)*, 2013, pp. 1–12.

[6] Z. Ozkan, E. Bayhan, M. Namdar, and A. Basgumus, "Object detection and recognition of unmanned aerial vehicles using raspberry pi platform," in *2021 5th International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*, 2021, pp. 467–472.

[7] Z. Ali, T. Tanveer, S. Aziz, M. Usman, and A. Azam, "Reassessing the performance of arm vs x86 with recent technological shift of apple," in *2022 International Conference on IT and Industrial Technologies (ICIT)*, 2022, pp. 01–06.

[8] S. Büyükçolak and R. Yeniçeri, "Quadrotor model implementation on raspberry pi zero and pi 4 boards using freertos," in *2023 12th Mediterranean Conference on Embedded Computing (MECO)*, 2023, pp. 1–4.

IX. Appendix

TABLE I
CPU Performance Metrics Across Single-Core and Multi-Core Benchmarks

CPU	Max Freq (GHz)	Single Avg Time (s)	Multi Avg Time (s)	Single Score	Multi Score	Cores
AMD Ryzen 7 7840H	3.8	24.17	13.48	2485.68	4450.08	8
Intel i7-8750H	2.2	21.07	11.72	2863.13	5118.96	6
Intel i7-1355U	3.7	14.63	6.31	4575.55	9508.80	10
Intel i7-8665U	2.1	129.65	96.85	1730.45	2697.78	6
ARM Cortex-A72 (Raspberry Pi 4)	1.5	32.47	12.94	2183.42	4535.93	4

A. Benchmark Sample Code

```
#!/bin/bash

# Function to display a menu and get user input
get_cpu_limit() {
  echo "Select CPU Usage:"
  echo " 1) Full Speed (100%)"
  echo " 2) Limited CPU (+-20% margin of error)"
  read -p "Enter your choice (1 or 2): " choice

  case $choice in
    1) cpu_limit=100 ;;
    2)
      read -p "Enter CPU limit (0-100): " cpu_limit
      if [[ ! $cpu_limit =~ ^[0-9]+$ ]] || [[ $cpu_limit -lt 0 ]] || [[ $cpu_limit -gt 100 ]];
          then
        echo "Invalid input. Please enter a number between 0 and 100."
        get_cpu_limit
      fi
      ;;
    *)
      echo "Invalid choice. Please enter 1 or 2."
      get_cpu_limit
      ;;
  esac
}

# Function to run a command with optional CPU limit and measure time
run_command() {
  local command="$1"
  local cpu_limit="$2"

  start_time=$(date +%s.%N)

  if [[ $cpu_limit -lt 100 ]]; then
    $command & # Run command in background
    pid=$!
    sudo cpulimit -p $pid -l $cpu_limit &
    wait $pid # Wait for process to finish
  else
    $command
  fi

  end_time=$(date +%s.%N)
  elapsed_time=$(echo "$end_time - $start_time" | bc -l)

  # Calculate score: higher score for faster completion (inverse time)
  score=$(echo "scale=2; 1000 / ($elapsed_time + 0.000001)" | bc)
  echo "Elapsed time: $elapsed_time seconds"
  echo "Score: $score"
}

# --- Main script starts here ---
```

```
get_cpu_limit

cd File_Compression || { echo "Failed to change directory"; exit 1; }

echo "Running single-core tests..."
total_elapsed=0
total_score=0
file_count=0 # Track number of files processed

# Run tests for each file
for file in file1.zip file2.7z file3.zip file4.xz file5.tar file6.mp4; do
  run_command "openssl dgst -sha3-256 $file" "$cpu_limit"
  total_elapsed=$(echo "$total_elapsed + $elapsed_time" | bc -l)
  total_score=$(echo "$total_score + $score" | bc -l)
  file_count=$((file_count + 1))
done

echo "Total elapsed time (single-core): $total_elapsed seconds"
echo "Total score (single-core): $total_score"

echo "Running multi-core tests in parallel at full speed..."
start_time=$(date +%s.%N)
parallel openssl dgst -sha3-256 ::: file1.zip file2.7z file3.zip file4.xz file5.tar file6.mp4
end_time=$(date +%s.%N)

elapsed_time=$(echo "$end_time - $start_time" | bc -l)

# Adjust parallel score by dividing by the number of files
parallel_score=$(echo "scale=2; (10000 / ($elapsed_time + 0.000001)) * $file_count" | bc)

echo "Elapsed time (parallel): $elapsed_time seconds"
echo "Score (parallel): $parallel_score"

# Final calculations
final_elapsed=$(echo "$total_elapsed + $elapsed_time" | bc -l)
final_score=$(echo "$total_score + $parallel_score" | bc -l)

echo "Final elapsed time (combined): $final_elapsed seconds"
echo "Final score (combined): $final_score"
```

Listing 1. Your Bash Script

```
username@raspberry:~/Documents $ bash Benchmark.sh
Select CPU Usage:
  1) Full Speed (100%)
  2) Limited CPU (+-20% margin of error)
Enter your choice (1 or 2): 1
Running single-core tests...
SHA3-256(file1.zip)= a8f94acb1b18fc76e0b8b303e9a64d8eacd195c87a155e350959b2a244de842d
Elapsed time: 1.553020746 seconds
Score: 643.90
SHA3-256(file2.7z)= dd6856d7d5adcf462943034ae709418797aefe75f70afa257339178b69c73a88
Elapsed time: 1.998991245 seconds
Score: 500.25
SHA3-256(file3.zip)= 22c1eef1f7b968e6baccb14795008bc7be1defab5d91dd0300f4d994a4faecda
Elapsed time: 2.248238895 seconds
Score: 444.79
SHA3-256(file4.xz)= 8c88145902658129a10a3ac43efab46346c08eacd6997d477a6ad1f92d0064c2
Elapsed time: 7.816083850 seconds
Score: 127.94
```

Fig. 4. Example Benchmark Output

CPU:	Ryzen 7840H	CPU:	i7-8750H	CPU:	i7-1355U	CPU:	i7-8665U	CPU:	ARM Cortex-A72
Max Frequency:	5.1Ghz	Max Frequency:	4.10Ghz	Max Frequency:	5Ghz	Max Frequency:	4.8Ghz	Max Frequency:	1.5Ghz
Cores:	8	Cores:	6	Cores:	10	Cores:	6	Cores:	4
Vm Max Freque	3.8Ghz	Vm Max Freque	2.2GHz	Vm Max Freque	3.7Ghz	Vm Max Freque	2.1Ghz	N/A	N/A
Architecture:	x-86	Architecture:	x86	Architecture:	x86	Architecture:	x86	Architecture:	ARM
Single(100%) A	24.17	Single(100%) A	21.07	Single(100%) A	14.63	Single(100%) A	129.65	Single(100%) A	32.47
Single(100%)Av	2485.68	Single(100%)Av	2863.13	Single(100%)Av	4575.55	Single(100%)Av	1730.45	Single(100%)Av	2183.42
Single(75%) Avg	32.43	Single(75%) Avg	27.73	Single(75%) Avg	15.04	Single(75%) Avg	140.76	Single(75%) Avg	36.54
Single(75%)Avg	2184.11	Single(75%)Avg	2307.98	Single(75%)Avg	4131.05	Single(75%)Avg	1745.37	Single(75%)Avg	1954.24
Single(50%) Avg	46.49	Single(50%) Avg	39.53	Single(50%) Avg	21.54	Single(50%) Avg	135.78	Single(50%) Avg	46.76
Single(50%)Avg	1368.64	Single(50%)Avg	1580	Single(50%)Avg	2970.52	Single(50%)Avg	1911.79	Single(50%)Avg	1398.04
Single(25%) Avg	83.65	Single(25%) Avg	71.93	Single(25%) Avg	38.13	Single(25%) Avg	138.98	Single(25%) Avg	65.89
Single(25%)Avg	732.05	Single(25%)Avg	871.55	Single(25%)Avg	1696.26	Single(25%)Avg	2064.6	Single(25%)Avg	1042.4
MultiCore4AVG	13.48	MultiCore4AVG	11.72	MultiCore4AVG	6.31	MultiCore4AVG	96.85	MultiCore4AVG	12.94
MultiCore4AVG	4450.08	MultiCore4AVG	5118.96	MultiCore4AVG	9508.8	MultiCore4AVG	2697.78	MultiCore4AVG	4535.93
MultiCore3AVG	13.84	MultiCore3AVG	11.35	MultiCore3AVG	7.33	MultiCore3AVG	96.01	MultiCore3AVG	13.72
MultiCore3AVG	4333.44	MultiCore3AVG	5285.52	MultiCore3AVG	8185.14	MultiCore3AVG	2737.86	MultiCore3AVG	4301.5
MultiCore2AVG	16.05	MultiCore2AVG	12.58	MultiCore2AVG	15.875	MultiCore2AVG	97.98	MultiCore2AVG	17.8
MultiCore2AVG	3737.58	MultiCore2AVG	4771.26	MultiCore2AVG	3779.46	MultiCore2AVG	2599.81	MultiCore2AVG	3560.2

Fig. 5. Full Project Data