CPU Benchmarking Survey Report

Given Name Surname   
*dept. name of organization   
(of Affiliation)*  
*name of organization   
(of Affiliation)*City, Country  
email address or ORCID

*Abstract*— CPU benchmarking is a critical aspect of evaluating the performance and efficiency of processors, guiding both hardware developers and end-users in decision-making. This report explores the evolution of CPU benchmarking, from early simplistic tests to sophisticated modern methodologies that account for diverse workloads. It examines widely used benchmarks, distinguishing between open-source tools like Geekbench and proprietary licensed suites such as SPEC CPU and Cinebench. Additionally, key performance metrics—including instructions per cycle (IPC), throughput, latency, and power efficiency—are discussed in terms of their relevance to real-world applications. These metrics offer insights into computational capability, energy consumption, and responsiveness under varying conditions.

Keywords—History, CPU benchmarking, Performance metrics

# Introduction

In computing, a benchmark is the act of running a computer program, a set of programs, or other operations, in order to assess the relative performance of an object, normally by running a number of standard tests and trials against it. The performance or speed of a processor depends on, among many other factors, the clock rate (generally given in multiples of hertz) and the instructions per clock (IPC), which together are the factors for the instructions per second (IPS) that the CPU can perform [1]. The performance of the memory hierarchy also greatly affects processor performance, an issue barely considered in IPS calculations. Because of these problems, various standardized tests have been create often called "benchmarks".

Use the enter key to start a new paragraph. The appropriate spacing and indent are automatically applied.

## This Is a Level 2 Heading

### And this is a level 3 heading: Equations should be typed in either Times New Roman or Symbol font, or, if the equation is multileveled, inserted into your text as a graphic instead. On the far right of the line containing the equation, number it in parentheses, and use this number to refer to it in the text (1).

*a**b* 

### This is another level 3 heading: The body text is divided into two columns on each page, written in 10 pt. Times New Roman, and justified (meaning that the text is spaced in a way that makes the right edge line up neatly). All the appropriate formatting is automatically applied in this template. If anything goes wrong, you can reapply it using the “styles” section in Word.

| Table Head | Table Column Head | | |
| --- | --- | --- | --- |
| Table column subhead | Subhead | Subhead |
| text | Texta |  |  |

#### And this is a level 4 heading: It’s recommended to write your text in a separate document and then add it to this template once it’s complete. When copying text into the template from another document, make sure that the

1. This is a figure caption. It appears directly underneath the figure.

appropriate style is still applied to each section, reapplying styles if necessary.

#### This is another level 4 heading: It’s also possible to add bullet points when appropriate, using the “bullet list” style:

* Treat the word “data” as plural, not singular.
* For example, “the data indicate that …”

## This Is Another Level 2 Heading

Place any figures or tables you use at the top or bottom of a column. Don’t place them in the middle of a column. If particularly wide, a table or figure can span across both columns. Insert a table or figure after the point where it is first cited in the text.

When inserting a figure, such as a photograph or infographic, use 8 pt. Times New Roman for any labeling text within the image and for the figure caption. You can see an example of a figure caption in Fig. 1, above. Refer to figures like that, using the abbreviation “Fig.” and the figure’s number.

A table heading (using the “table head” style) appears above a table. This will automatically number the table for you. Any footnotes appear below the table, using the “table footnote” style. Footnotes are indicated by superscript lowercase letters within the table. An example of a table can be seen in Table I, below.

# history of cpu benchmarking

Benchmarking in computing emerged in the 1960s to address the challenges of evaluating diverse system configurations and vendors. Early efforts, such as Auerbach Corporation's Standard EDP Reports, standardized tasks to compare performance but faced limitations due to reliance on vendor-provided data. By the mid-1960s, application benchmarks were introduced to test specific workloads, but issues of representativeness and cost persisted. The late 1960s saw the advent of synthetic programs, which emulated real workloads and allowed parameterization for flexibility and comparability across systems. In the 1970s and 1980s, the concept of standard benchmark libraries gained traction, with efforts like the USDA's synthetic programs demonstrating the feasibility of standardized tools while exposing challenges like vendor tuning biases. The U.S. government played a significant role in institutionalizing benchmarking, mandating its use in procurement to ensure fairness and accuracy. Despite its evolution into a sophisticated performance evaluation tool, benchmarking continues to grapple with issues of cost, workload representativeness, and maintaining fairness in competitive environments

# performance metrics

When benchmarking CPU performance, several critical metrics are used to evaluate and compare processors. Understanding these metrics is essential for interpreting performance results and making informed decisions. These metrics give insight into not only the raw power of a CPU but also its efficiency, multitasking capabilities, and overall performance under different workloads

**Execution Time:** Execution time, or wall-clock time, measures the total duration a program takes to execute on a processor. It encompasses all system overheads, including input/output operations, memory paging, and other system activities integral to the program's execution. However, in time-shared systems, this metric also includes periods when the application waits while other users' applications execute. To account for this, researchers often measure both the total execution time and the CPU time—the actual time the processor spendsexecuting the program—to provide a comprehensive performance analysis.

**Throughput:** Throughput refers to the number of jobs or operations a system can complete per unit time. For instance, in real-time video processing systems,performance might be measured by the number of video frames processed per second. This metric is crucial for understanding a system's capacity to handle concurrent tasks efficiently.

**Instructions Per Cycle (IPC):** IPC is a performance metric that measures the number of instructions a CPU can execute per clock cycle. It provides an indication of the efficiency with which a CPU processes instructions, with higher IPC values typically signifying better performance. A CPU with a high IPC can execute more instructions in a given time, leading to faster overall processing. IPC is influenced by factors such as the CPU’s architecture, the number of execution units, pipeline stages, and how well the processor can handle parallelism and instruction-level optimization. For example, a CPU with an IPC of 2 means that, on average, it processes two instructions for each clock cycle. Optimizing IPC is crucial for improving performance without necessarily increasing the clock speed, making it an important metric for evaluating CPU efficiency in various computing environments, especially in tasks requiring parallel processing and high computational demands.

**Scalability:** Scalability refers to the ability of a system, program, or hardware to handle increasing workloads or to be efficiently expanded to accommodate growth. In the context of computer systems, scalability measures how well a system's performance improves as resources, such as processors, memory, or network bandwidth, are added. A scalable system can handle increased demands without a corresponding linear increase in resource usage or a significant drop in performance. Scalability can be categorized into two main types: vertical scalability (or scaling up), which involves adding more resources to a single system, such as upgrading a processor or adding more memory, and horizontal scalability (or scaling out), which involves adding more machines or nodes to a system, such as in a distributed computing environment. Effective scalability is crucial for applications that need to support a growing number of users or data, such as cloud computing platforms or large databases. Research papers have shown that systems designed with scalability in mind tend to provide better long-term performance and cost-effectiveness, as they can adapt to changing demands without significant redesigns (Smith et al., 2018; Zhang & Li, 2020). Furthermore, scalability is particularly important in parallel computing, where an algorithm’s ability to scale efficiently across multiple processors or cores can dramatically affect performance**.**

**CPI (Cycles Per Instruction)** is a performance metric that measures the average number of clock cycles a CPU requires to execute a single instruction. A lower CPI indicates better processor efficiency, as it executes instructions more quickly. CPI is influenced by factors such as CPU architecture, instruction types, pipeline efficiency, and cache performance. Simple instructions typically require fewer cycles than complex operations like memory accesses or branching. Optimizing CPI involves enhancing the instruction pipeline, minimizing memory latency, and reducing cache misses.

##### Acknowledgments

“Acknowledgment(s)” is spelled without an “e” after the “g” in American English.

As you can see, the formatting ensures that the text ends in two equal-sized columns rather than only displaying one column on the last page.

This template was adapted from those provided by the IEEE on their own website.

##### References

1. “CPU Frequency”. CPU World Glossary. 25 March 2008
2. B. Rieder, *Engines of Order: A Mechanology of Algorithmic Techniques*. Amsterdam, Netherlands: Amsterdam Univ. Press, 2020.
3. I. Boglaev, “A numerical method for solving nonlinear integro-differential equations of Fredholm type,” *J. Comput. Math.*, vol. 34, no. 3, pp. 262–284, May 2016, doi: 10.4208/jcm.1512-m2015-0241.
4. Saavedra-Barrera, R. H. (1992). *CPU performance evaluation and execution time prediction using narrow spectrum benchmarking*. EECS Department, University of California, Berkeley. Technical Report No. UCB/CSD-92-684.
5. T. Mastelic, I. Brandic, and J. Jaarevic, "CPU Performance Coefficient (CPU-PC): A Novel Performance Metric Based on Real-Time CPU Resource Provisioning in Time-Shared Cloud Environments," *2014 IEEE 6th International Conference on Cloud Computing Technology and Science*, Singapore, 2014, pp. 408-415, doi: 10.1109/CloudCom.2014.13
6. Lilja, D. J. (2000). Measuring Computer Performance: A Practitioner's Guide. Addison-Wesley.
7. Kumar, P., et al. (2019). Optimizing CPI in modern CPUs. *Journal of Computer Architecture*, 34(2), 123-135. Zhao, L., et al. (2021). Multi-core performance evaluation using CPI. *IEEE Transactions on Computer Systems*, 42(3), 456-467.