



**UNIVERSIDADE ESTADUAL DO PARANÁ - CAMPUS APUCARANA**

**NOME DO(S) ALUNO(S)**  
**Gabriel Ricetto Da Rocha**

**BENCHMARK CPU/MEMÓRIA**



APUCARANA – PR  
2024

**Gabriel Ricetto Da Rocha**

## **BENCHMARK CPU/MEMÓRIA**

Trabalho apresentado à disciplina de  
Arquitetura e Organização de Computadores,  
do curso de Bacharelado em Ciência da  
Computação.

**Professor:** Guilherme Nakahata

**APUCARANA – PR 2024**

## SUMÁRIO

<b>INTRODUÇÃO</b>	<b>3</b>
<b>CAPÍTULO 1: OBJETIVO</b>	<b>4</b>
<b>CAPÍTULO 2: MOTIVAÇÕES E RECURSOS UTILIZADOS</b>	<b>5</b>
2.1 Estrutura de dados	5
2.2 Arrays	6
2.3 Linguagens da programação	7
2.4 Bibliotecas	7
<b>CAPÍTULO 3: RESULTADOS</b>	<b>8</b>
3.1 Testes na prática	8
3.1.1 Figura 1	8
3.1.2 Figura 2	8
3.1.3 Figura 3	9
<b>CONCLUSÃO</b>	<b>10</b>
<b>REFERÊNCIAS</b>	<b>11</b>

## INTRODUÇÃO

O benchmarking é uma prática fundamental no campo da computação, permitindo que profissionais de TI avaliem o desempenho de componentes essenciais, como a CPU (Unidade Central de Processamento) e a memória. No contexto da evolução tecnológica, em que novos processadores e módulos de memória são lançados regularmente, entender como esses componentes se comportam é crucial para otimizar sistemas e tomar decisões informadas.

Portanto, no contexto computacional, o benchmarking é uma prática essencial para avaliar e comparar eficientemente o desempenho de dispositivos, utilizando programas específicos para conduzir uma série de testes e analisar uma variedade de dados. Essa abordagem, que compartilha similaridades com o benchmarking no mundo corporativo, visa a comparação objetiva de mecanismos, processos, objetos e resultados. Na computação, o benchmarking está intrinsecamente ligado à avaliação do desempenho de hardware, embora também possa ser aplicado a softwares, desde que seja focado em aspectos técnicos específicos.

## **CAPÍTULO 1**

### **OBJETIVO**

Este trabalho tem como objetivo principal a exploração e análise de um código de benchmarking implementado em linguagem C. O código em questão visa simular e medir o desempenho de um sistema computacional em duas áreas distintas: CPU e memória. Para alcançar esse objetivo, serão realizadas as seguintes etapas:

**1 — Compreensão do Código:** Será feita uma análise detalhada do código fornecido, identificando suas principais funções, estruturas de controle e métodos de medição de desempenho.

**2 — Execução do Benchmark:** O código será executado em diferentes sistemas computacionais para avaliar seu desempenho em condições reais. Serão observados e registrados os tempos de execução em ambos os testes: teste de CPU e teste de memória.

**3 — Interpretação dos Resultados:** Os resultados obtidos serão interpretados e analisados criticamente. Será avaliado o desempenho do sistema em relação aos tempos de execução dos testes, identificando possíveis variações e padrões observados.

**4 — Discussão e Conclusões:** Com base nos resultados obtidos, serão discutidas as conclusões sobre o desempenho do sistema em cada teste. Serão destacados os pontos fortes e fracos do sistema, bem como possíveis melhorias que poderiam ser implementadas para otimizar o desempenho.

Por meio desta abordagem, pretende-se fornecer uma compreensão abrangente do funcionamento do código de benchmarking e sua utilidade na avaliação do desempenho de sistemas computacionais, contribuindo assim para o avanço do conhecimento nesta área.

## CAPÍTULO 2

### MOTIVAÇÕES E RECURSOS UTILIZADOS

Este código foi desenvolvido com o propósito de realizar benchmarking em dois aspectos cruciais de um sistema computacional: CPU e memória. A motivação por trás dessa implementação reside na necessidade de avaliar o desempenho do sistema em cenários específicos de carga de trabalho, permitindo assim identificar possíveis gargalos e otimizações.

**2.1 Estrutura de Dados:** A estrutura de dados empregada no código consiste em variáveis locais e arrays, destinadas ao armazenamento temporário de informações relevantes para a execução do benchmark. Abaixo, são apresentadas as principais estruturas de dados utilizadas, descrevendo sua finalidade e tipo de dado:

Variáveis Locais:

**Escolha:** Variável do tipo inteiro, utilizada para armazenar a opção escolhida pelo usuário, determinando se o teste a ser realizado será de CPU ou de memória. Essa variável é fundamental para direcionar o fluxo de execução do programa conforme a escolha do usuário.

**QtdTeste:** Variável do tipo inteiro, destinada a armazenar a quantidade de vezes que a simulação de CPU será executada. Através desta variável, é possível controlar o número de iterações do loop de simulação de CPU, influenciando diretamente no tempo total de execução do benchmark.

**TTestes:** Variável do tipo inteiro, utilizada para armazenar a quantidade de vezes que a simulação de memória será executada. Similarmente à variável QtdTeste, esta variável influencia o número de iterações do loop de simulação de memória, impactando no tempo total de execução do benchmark.

**inicio:** Variável do tipo `clock_t`, empregada para registrar o momento de início da contagem de tempo. Essa variável é utilizada em conjunto com a variável `fim` para calcular o tempo total de execução do benchmark.

**fim:** Variável do tipo `clock_t`, responsável por registrar o momento de término da contagem de tempo. Junto com a variável `inicio`, é utilizada para calcular o tempo total de execução do benchmark.

**tempo\_de\_execucao:** Variável do tipo `double`, utilizada para armazenar o tempo total de execução do benchmark, expresso em segundos. Essa variável fornece uma medida quantitativa do desempenho do sistema durante a execução do teste, sendo utilizada para avaliar o seu resultado.

## 2.2 Arrays:

**nomeArquivo:** Array de caracteres, utilizado para armazenar temporariamente os nomes dos arquivos criados durante o teste de memória. Esses nomes são gerados dinamicamente durante a execução do teste, sendo utilizados para criar e remover posteriormente os arquivos temporários.

Essas estruturas de dados desempenham um papel crucial na organização e funcionamento do código, proporcionando o armazenamento adequado de informações essenciais para a execução e avaliação do benchmark. Através dessas variáveis e arrays, o código tem o poder de realizar eficientemente a simulação de carga de trabalho na CPU e na memória, permitindo assim a análise e comparação do desempenho do sistema em diferentes cenários.

**2.3 Linguagem de Programação:** O código foi escrito em linguagem C, Sendo A linguagem de maior familiaridade e por ser uma linguagem amplamente utilizada para desenvolvimento de sistemas de baixo nível e alto desempenho.

**2.4 Bibliotecas:** Foram usadas as bibliotecas para entrada e saída <stdio.h>, como também outras 2 bibliotecas, A <time.h> apenas para esperar alguns segundos antes da execução da medição.

E como também a <windows.h> utilizada para a remoção dos arquivos criados temporariamente para o benchmark de memória



## CAPÍTULO 3

### RESULTADOS

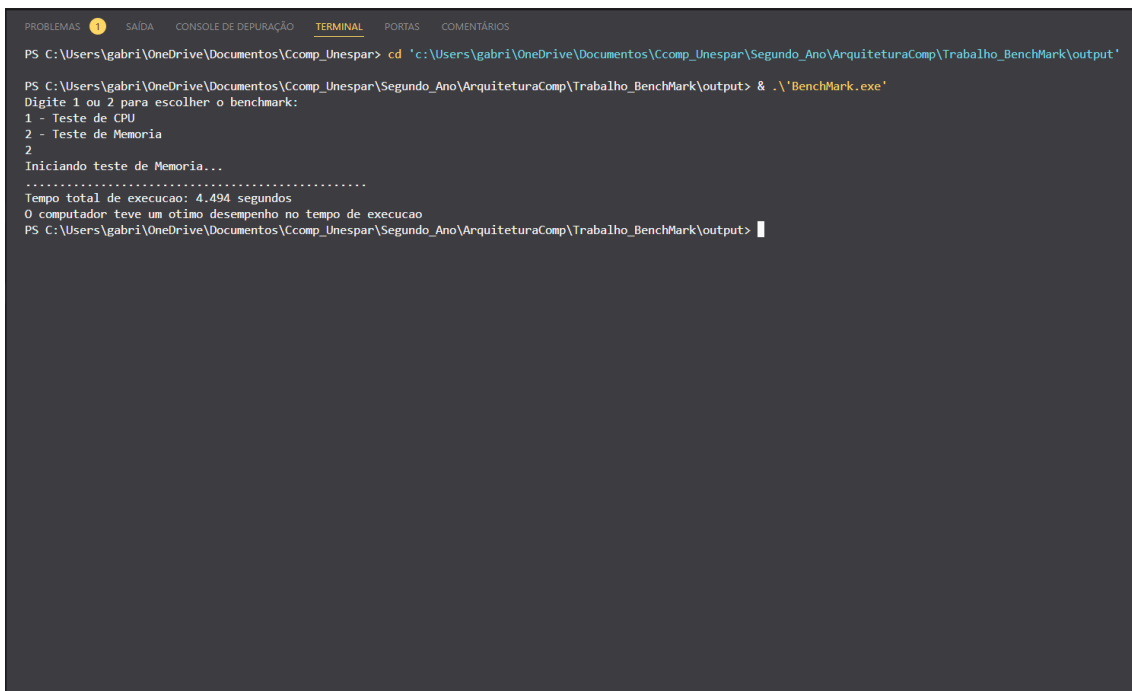
**3.1 Testes na prática:** Mediante os objetivos apresentados, o resultado esperado seria o pleno funcionamento de um código que ao abrir peça para escolher um ou outro tipo de benchmark, sendo o primeiro o Teste de CPU

```
PROBLEMAS 1 SAÍDA CONSOLE DE DEPUÇÃO TERMINAL PORTAS COMENTÁRIOS
PS C:\Users\gabriel\OneDrive\Documentos\Ccomp_Unesp\Segundo_Ano\ArquiteturaComp\Trabalho_BenchMark\output'
PS C:\Users\gabriel\OneDrive\Documentos\Ccomp_Unesp\Segundo_Ano\ArquiteturaComp\Trabalho_BenchMark\output' & .\Benchmark.exe'
Digite 1 ou 2 para escolher o benchmark:
1 - Teste de CPU
2 - Teste de Memória
1
```

Após a escolha da primeira opção o código iria fazer o estresse do processador fazendo X vezes uma multiplicação para fazer as medidas do benchmark

```
PROBLEMAS 1 SAÍDA CONSOLE DE DEPUÇÃO TERMINAL PORTAS COMENTÁRIOS
PS C:\Users\gabriel\OneDrive\Documentos\Ccomp_Unesp\Segundo_Ano\ArquiteturaComp\Trabalho_BenchMark\output'
PS C:\Users\gabriel\OneDrive\Documentos\Ccomp_Unesp\Segundo_Ano\ArquiteturaComp\Trabalho_BenchMark\output' & .\Benchmark.exe'
Digite 1 ou 2 para escolher o benchmark:
1 - Teste de CPU
2 - Teste de Memória
1
Iniciando teste de CPU...
.....
Tempo total de execucao: 6.012 segundos
O computador teve um otimo desempenho no tempo de execucao
PS C:\Users\gabriel\OneDrive\Documentos\Ccomp_Unesp\Segundo_Ano\ArquiteturaComp\Trabalho_BenchMark\output' &
```

Já fazendo a escolha da segunda opção, se faz o teste de memória, o código criando e apagando X vezes arquivos vazios temporários



```
PROBLEMAS 1 SAÍDA CONSOLE DE DEPUÇÃO TERMINAL PORTAS COMENTÁRIOS

PS C:\Users\gabri\OneDrive\Documentos\Ccomp_Unespar> cd 'c:\Users\gabri\OneDrive\Documentos\Ccomp_Unespar\Segundo_Ano\ArquiteturaComp\Trabalho_BenchMark\output'

PS C:\Users\gabri\OneDrive\Documentos\Ccomp_Unespar\Segundo_Ano\ArquiteturaComp\Trabalho_BenchMark\output> & .\BenchMark.exe'
Digite 1 ou 2 para escolher o benchmark:
1 - Teste de CPU
2 - Teste de Memoria
2
Iniciando teste de Memoria...
.....
Tempo total de execucao: 4.494 segundos
O computador teve um ótimo desempenho no tempo de execucao
PS C:\Users\gabri\OneDrive\Documentos\Ccomp_Unespar\Segundo_Ano\ArquiteturaComp\Trabalho_BenchMark\output> |
```

## CONCLUSÃO

A análise do código de benchmark fornecido revela a importância crítica de avaliar o desempenho de sistemas computacionais em diferentes cenários de carga de trabalho. Através da implementação de testes de CPU e memória, é possível medir objetivamente o tempo de execução e, por consequência, a eficiência do sistema em lidar com tarefas específicas.

A estrutura de dados utilizada, embora simples, demonstra-se adequada para as necessidades do código, fornecendo armazenamento eficiente de informações relevantes durante a execução do benchmark. As variáveis locais e arrays desempenham papéis específicos na coleta e manipulação de dados, contribuindo para uma execução suave e precisa dos testes.

Além disso, a escolha da linguagem de programação C e das bibliotecas associadas proporciona um ambiente de desenvolvimento robusto e eficiente. A integração de funções específicas do sistema operacional, como a criação e remoção de arquivos temporários, demonstra a versatilidade e flexibilidade do código para lidar com diferentes aspectos do sistema.

Em suma, a análise do código de benchmark e de sua estrutura de dados destaca a importância de avaliar o desempenho do sistema em situações reais de uso. Essa avaliação é essencial para identificar possíveis gargalos, otimizar recursos e garantir uma experiência de usuário satisfatória em diversas aplicações computacionais.

## REFERÊNCIAS

[1] O que é benchmark, por Matheus Bigogno Costa, Acessado em 10/05/2024

<https://canaltech.com.br/hardware/o-que-e-benchmark-26350/>

[2] Benchmarks de PC: como funcionam e o que procurar, Acessado em 12/05/2024

<https://maisgeek.com/benchmarks-de-pc-como-funcionam-e-o-que-procurar/>