**Universidade de São Paulo**
**Instituto de Matemática e Estatística**
**MAC0216 - Técnicas de Programação I**

# EP1:
# Comparação de função hash em linguagem de montagem e Python

Ana Lívia Rüegger Saldanha
Nº USP: 8586691

11 de setembro de 2023

**Resumo**

Este relatório apresenta uma breve análise de desempenho e comparação entre os dois programas desenvolvidos para o Exercício Programa 1 da disciplina MAC0216, um em linguagem de montagem x86_64 e outro em Python3, ambos implementando um mesmo algoritmo de hash.

---

## 1 Introdução

O projeto a que este relatório se refere, realizado como Exercício Programa para a disciplina MAC0216 do IME-USP, apresenta implementações de um mesmo algoritmo de hash em Python3 e em linguagem de montagem para arquitetura x86_64, com intuito de comparar o desempenho desses dois programas. Essa comparação será desenvolvida adiante.

O algoritmo de hash implementado, como especificado, foi estruturado em quatro passos, que foram implementados segundo a mesma lógica em ambos os programas. Estes devem receber, da entrada padrão, uma string contendo de 1 a 100000 caracteres ASCII e imprimir, na saída padrão, um hash produzido pelo algoritmo. Esse hash é uma string de 32 caracteres representando, em hexadecimal, valores referentes a 16 bytes produzidos pela função.

A seguir, serão apresentados os resultados obtidos através de experimentos realizados com os dois programas, utilizando entradas de diferentes tamanhos (10, 100, 1000, 10000 e 100000 caracteres).

## 2 Desempenho

Nesta seção, apresentamos dados obtidos experimentalmente a fim de demonstrar o desempenho, em termos de tamanho e de tempo de execução, de cada um dos programas.

### 2.1 Configurações

Antes de apresentarmos os resultados das medições, cabe especificar quais são as configurações do computador no qual os experimentos foram executados, ressaltando que as medições, para ambos os programas, foram realizadas sob as mesmas condições. O computador em questão possui processador AMD Ryzen 5 5500U, 8GB de RAM e sistema operacional GNU/Linux x86_64 (distribuição Linux Mint 20.3; kernel Linux 5.8.0-63-generic).

### 2.2 Tempo de execução

As medições de tempo de execução foram feitas com uso do utilitário `/usr/bin/time`, como sugerido na especificação do exercício. Cada programa foi executado 10 vezes para cada uma das entradas disponibilizadas e a tabela abaixo apresenta as médias de tempo de execução considerando essas 10 medições para cada caso.

| **Entrada** | **Média (Python)** | **Média (Assembly)** |
|---|---|---|
| texto10.txt | 0.079 | 0.000 |
| texto100.txt | 0.089 | 0.000 |
| texto1000.txt | 0.110 | 0.000 |
| texto10000.txt | 0.201 | 0.000 |
| texto100000.txt | 1.074 | 0.005 |

Tabela 1: Tempos de execução em segundos (média de dez execuções para cada entrada)

### 2.3 Tamanho

A tabela abaixo mostra os tamanhos em bytes, obtidos pelo comando `ls -l`, do código fonte em Python e do executável Assembly.

| **Python (código fonte)** | **Assembly (executável)** |
|---|---|
| 2869 | 8816 |

Tabela 2: Tamanhos dos programas em bytes

## 3 Conclusão

Como esperado, a implementação da função hash em Assembly alcançou um significativo *speedup* em relação ao mesmo algoritmo implementado em Python. Isso se deve principalmente ao fato de que, na linguagem de montagem, o programador pode se valer das instruções estritamente necessárias para realizar as operações desejadas, sem precisar se ater aos padrões definidos pelos comandos de uma linguagem de mais alto nível. Ainda, por se tratar de uma linguagem interpretada, o programa em Python tende a ser mais lento do que os programas escritos em linguagens compiladas, como C ou o próprio Assembly.

Da mesma forma, é esperado que o executável tenha um tamanho maior em bytes do que o código fonte em Python, visto que o processo de montagem e ligação adiciona a ele todas as informações necessárias para sua execução. No caso do código em Python, uma linguagem interpretada, informações externas ao código em si são de responsabilidade do interpretador e não precisam ser ligadas ao arquivo fonte, como ocorre no caso das linguagens compiladas.