Relatório - Exercício Programa #01

MAC0216 - Técnicas de Programação I

Professor Daniel Macêdo Batista

Instituto de Matemática e Estatística - IME USP

Aluno: Francisco Nassif Membrive

Data de Entrega: 11.09.2023

1. **Programas**

   No presente exercício, foram elaborados dois programas, utilizando as linguagens Python 3.10 e Linux Assembly x86_64. O objetivo dos programas foi implementar uma função hash capaz de criptografar textos de até 100000 caracteres ASCII em um hash de tamanho fixo (16 bytes) e formato hexadecimal. O intuito de fazer dois programas equivalentes, um em uma linguagem de baixo nível e um em uma linguagem de alto nível foi comparar a velocidade de execução e demonstrar como a linguagem de máquina pode ser mais rápida mesmo em um programa em que a eficiência não foi um foco.

Entrada	Média (Python)	Média (Assembly)	Eficiência (vezes)
texto100000.txt	0,3933494973	0,0334355001	11,76442692
texto10000.txt	0,0502314623	0,0042174751	11,91031627
texto1000.txt	0,0153854370	0,0013137202	11,71134995
texto100.txt	0,0117777545	0,0011081430	10,62837062
texto10.txt	0,0113011773	0,0010906302	10,36206159

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

Python	**Assembly**
5726 bytes	8816 bytes

Tabela 2: Tamanhos dos programas em bytes

2. **Execução**

   A execução e cálculo dos tempos foi feita através de um bash gerado via ChatGPT 3.5 e rodado no Ubuntu 22.04 (Máquina Virtual) de 64 bits, com processador Ryzen 5 3600 utilizando 3 núcleos e 3 threads através da Oracle VM VirtualBox e 16GB de memória RAM. Foi utilizado o montador nasm versão 2.16 no caso do código assembly a versão 3.10 do Python no caso do python. Os programas e o bash foram

escritos utilizando o Visual Studio Code e rodados no terminal. O cálculo da eficiência foi feito no Google Sheets.

3. Resultado e conclusões

Mesmo com o tamanho bem maior, o código assembly é muito mais rápido. É preciso observar que o código poderia ser mais leve, caso fosse feito de maneira eficiente. Como não é o objetivo central da disciplina neste momento, há várias redundâncias no código assembly que poderiam ser convertidas em mais economia de memória e tempo. Já o código python é simples e não há grandes melhorias a mais para implementar. O desempenho do assembly se manteve entre 10 e 12 vezes mais rápido que o python. Em uma versão anterior do código, que não estava com saídas correspondentes, o código assembly rodou 342 vezes mais rápido. A suspeita inicial foi que a velocidade do assembly poderia ser comparada com a velocidade do python de maneira não linear. No entanto, posteriormente foi verificado que a divergência entre os códigos era um loop da ordem do tamanho da string. Como o código em assembly estava sem este loop, ele pôde atingir esta velocidade muito maior. Os resultados foram conforme o esperado, embora não tenham chegado próximo a estimativa de pelo menos 47 vezes mais rápido na conversão de python para C exposta no começo da disciplina. Ainda assim, o tempo foi menor e o tamanho do arquivo maior para o assembly, já que teve mais de 3 vezes a quantidade de linhas do programa em python e contou com muito mais comentários.