

Instruções

1. Esta avaliação deve ser feita individualmente, em dupla ou trio.
2. Data de entrega: **31/10/2023 até 18:59**. Trabalhos não podem ser entregues em atraso.
3. Esta avaliação tem por objetivo consolidar o aprendizado sobre memória principal e memória virtual.
4. A implementação deverá ser desenvolvida utilizando a linguagem de programação de sua preferência (C, C++, Python, Java, C#, Javascript/Node.js, Rust, etc). Porém, a utilização e suporte as operações pela linguagem escolhida é de responsabilidade do(s) aluno(s). Bibliotecas que também implementem e que permitam usar conceitos de paralelismo também podem ser usadas, mas o aluno também é responsável pelo seu uso e apresentação.
5. O sistema deve ser entregue funcionando corretamente.
6. Deve ser apresentado um relatório eletrônico em formato PDF (em outro formato é descontado 1,5 ponto) que contenha:
 - Identificação do autor e do trabalho.
 - Enunciado dos projetos.
 - Explicação e contexto da aplicação para compreensão do problema tratado pela solução.
 - Resultados obtidos com as simulações.
 - Códigos importantes da implementação.
 - Resultados obtidos com a implementação (tabelas, gráficos e etc).
 - Análise e discussão sobre os resultados finais.
7. Deve ser disponibilizado os códigos da implementação juntamente com o relatório (salvo o caso da disponibilidade em repositório aberto do aluno, que deve ser fornecido o link). O repositório deve estar aberto do momento da entrega em diante, sendo que o professor não responsabiliza caso o projeto não esteja disponível para consulta no momento da correção, sendo do(s) aluno(s) essa responsabilidade de manter disponível.
8. O trabalho deverá ser apresentado em data definida pelo professor. É de responsabilidade do(s) aluno(s) explicar os conceitos, comandos, bibliotecas usadas. É de responsabilidade do(s) aluno(s) fazer a solução funcionar e ela deverá ser baixada do local de entrega no momento da apresentação. Trabalhos não apresentados terão como nota máxima 5,0.

Descrição do projeto a ser desenvolvido

Projeto

Suponha que um sistema tenha um endereço virtual de tamanho entre 16 bits à 32 bits com deslocamento na página de 256 b à 4 Kb. Escreva um programa que receba um endereço virtual (em decimal) na linha de comando ou leitura do arquivo addresses.txt faça com que ele produza o número da página e o deslocamento do endereço fornecido, sendo que essa posição indica qual a posição que será lido do arquivo data_memory.txt. Você irá encontrar esses arquivos no github da disciplina, mais especificamente na pasta Memory ([link repositório](#)).

Por exemplo, seu programa seria executado da seguinte forma:

```
./virtual_memory_translate.exe 19986
```

ou

```
./virtual_memory_translate.exe addresses.txt
```

Seu programa produzirá:

- O endereço 19986 contém:
 - número da página = 4
 - deslocamento = 3602
 - Valor lido: 50 (exemplo)

No caso, o número em binário é **0100 1110 0001 0010**, sendo que **0100** diz respeito à página e **1110 0001 0010** diz respeito ao deslocamento na página. Você consegue conferir isso com a calculadora do Windows/Linux no modo programador. Para manipular os números em nível de bit, é recomendado usar os operadores bitwise (bit-a-bit) da linguagem escolhida. No caso o exemplo apresentado é para 16 bits. No caso de 32 bits, haveria mais 16 bits a esquerda (mais significativo) referentes ao número de páginas, **0000 0000 0000 0000 0100 1110 0001 0010**, porém, ainda será traduzido para página 4 e deslocamento 3602.

Escrever este programa exigirá o uso do tipo de dados apropriado para armazenar 16 à 32 bits (short ou int). É recomendado que você também use tipos de dados sem sinal. Além disso, para endereços de 32 bits deve ser possível usar paginação hierárquica de 2 níveis mantendo 4 Kb, com cada nível tendo 10 bits de tamanho.

Para a implementação do código, você pode fazer um fork do [repositório da disciplina](#) no github e usar o [codespaces](#) do github para a implementação, onde o mesmo irá executar o Visual Code em uma distro Linux Ubuntu com 2 núcleos e 8 GB de memória principal. Mas onde será executado seus códigos fica a critério do(s) aluno(s)

Pontuação extra em prova (0,5):

A fim de estimular a ampliação dos seus conhecimentos sobre memória virtual, você deve fazer implementar paginação por hash para a versão de 32 bits usando a função $h(k) = k \bmod M$, onde k é valor do endereço e M é valor que será usado para obter o endereço. Use M igual à 11, 43 e 155.