Segundo Projeto Extra – Simulador de Acesso à Memória

Prazo de Entrega: 12/11

Entrega: via Classroom

Objetivo

Desenvolver um **simulador de acesso à memória** em Python, com foco no funcionamento de uma **TLB (Translation Lookaside Buffer)** e na **memória virtual com substituição de páginas**.

O simulador deve permitir avaliar estatísticas como **acertos na TLB (TLB hits), faltas na TLB (TLB misses)** e **page faults**, simulando o comportamento de um sistema de memória virtual simplificado.

Descrição do Projeto

A implementação principal será feita na classe `MemorySimulator` (arquivo `mem_sim.py`).

As estruturas de dados da classe são:

- `tlb` : pode ser uma lista de tuplas (page_number, frame_number) ou OrderedDict para facilitar a implementação de LRU
- `page_table` : dicionário que mapeia `page_number → frame_number`
- `frames` : memória física simulada representada por um array que armazena qual página está em cada quadro

> **Observação:** É permitido criar estruturas auxiliares para implementar as políticas de substituição, desde que elas sejam documentadas no relatório.

As variáveis de saídas são:

- `tlb_hits` : número de acertos na TLB
- `tlb_misses` : número de faltas na TLB
- `page_faults` : número de page faults

Parâmetros da Simulação

A classe `MemorySimulator` deve aceitar os seguintes parâmetros:

Parâmetro	Descrição
`page_size`	Tamanho da página em bytes
`num_tlb_entries`	Número de entradas na TLB (recomendado ≥ 16)
`num_frames`	Número de quadros na memória física
`rep_policy`	Política de substituição de páginas (`LRU` ou `SecondChance`)

Funcionalidades a Implementar

Método principal

```
access_memory(virtual_address)
```

- Simula o acesso a um endereço virtual.
- Atualiza os contadores de **hits, misses e page faults**.
- Aplica a **política de substituição** se necessário.

Métodos auxiliares recomendados

Você pode implementar metódos auxiliares para o método princial, alguns recomendados são:

- Método para inserção/atualização da TLB
- Método para alocação de frames na memória física
- Método para implementação das políticas `LRU` ou `SecondChance`

Não é obrigatório essas implementações mas deixa o código mais organizado.

> **Observação:** Documente claramente cada função auxiliar, explicando seu papel no simulador.

Testes

- Um arquivo `trace.in` é fornecido como base e a saída esperada da simulação.
- É obrigatório criar **pelo menos 2 conjuntos de testes**, cada um com **≥ 500 acessos**, com entradas e saídas esperadas.

- Recomenda-se gerar **traces aleatórios** ou baseados em algoritmos simples (ex.: loops GEMM, acessos sequenciais ou randômicos).

Entrega

O material entregue deve conter **todos os arquivos do projeto**, organizados da seguinte forma:

1. **Código-fonte completo do simulador**.
2. **Conjunto de testes** com entradas e saídas esperadas (pelo menos 2 conjuntos com 500 acessos cada).
3. **Relatório breve**, em **PDF**, contendo:
    - Visão geral do código e fluxo de execução do simulador
    - Descrição da abordagem de implementação adotada, assim como detalhes das funções implementadas
    - Desafios encontrados e como foram solucionados

Regras de nomenclatura e organização

- Todos os arquivos devem ser reunidos em um único arquivo `.zip`.
- A estrutura do nome do arquivo `.zip` deve ser:

```
mem-sim-[nomedoaluno].zip
```

> **Atenção:** Substitua `[nomedoaluno]` pelo seu nome completo. **É obrigatório seguir esta estrutura de nome.**

- O relatório deve ter o nome:

```
mem-sim-relatorio.pdf
```

- Não modifique este arquivo `Readme.md`.
- O envio deve ser realizado via **Classroom** até a data limite do projeto.