Trabalho de Organização e Arquitetura de Computadores

Um Simulador que não é bem um Simulador!

Entrega: 19/06/2024

Descrição

- O trabalho consiste na construção de uma aplicação para emular os o tipo de mapeamento de memória *cache* com **Mapeamento associativo por conjunto.**
- Contudo, se um conjunto memória *cache* estiver cheio, teremos de escolher um algoritmo de substituição. Em nosso caso, implementaremos o algoritmo de substituição **LFU - Least Frequently Used (Menos frequentemente utilizado)**.
- Linguagens de programação permitidas:
  - ***C/C++ e Python***.
- Número de pessoas:
  - **Grupo de, no máximo, 2 alunos.**

Descrição

- Para desenvolver o trabalho, você deve considerar um arquivo de entrada, que conterá todas as informações necessárias sobre:
  - a memória principal (esta poderá ser preenchida (por completo) com valores pseudo-aleatórios);
  - a memória cache.
- Seu programa deverá conter um menu que permita (enquanto o programa não for encerrado):
  - Ler o nome de um arquivo que contenha as informações necessárias;
    - Baseado no conteúdo do arquivo, ler informações e apresentar na tela conforme tipo do mapeamento.
    - Informar um endereço de MP válido para acesso à MP; ou
    - Ler um arquivo que contém uma sequência de endereços da MP.
  - Sair do programa.

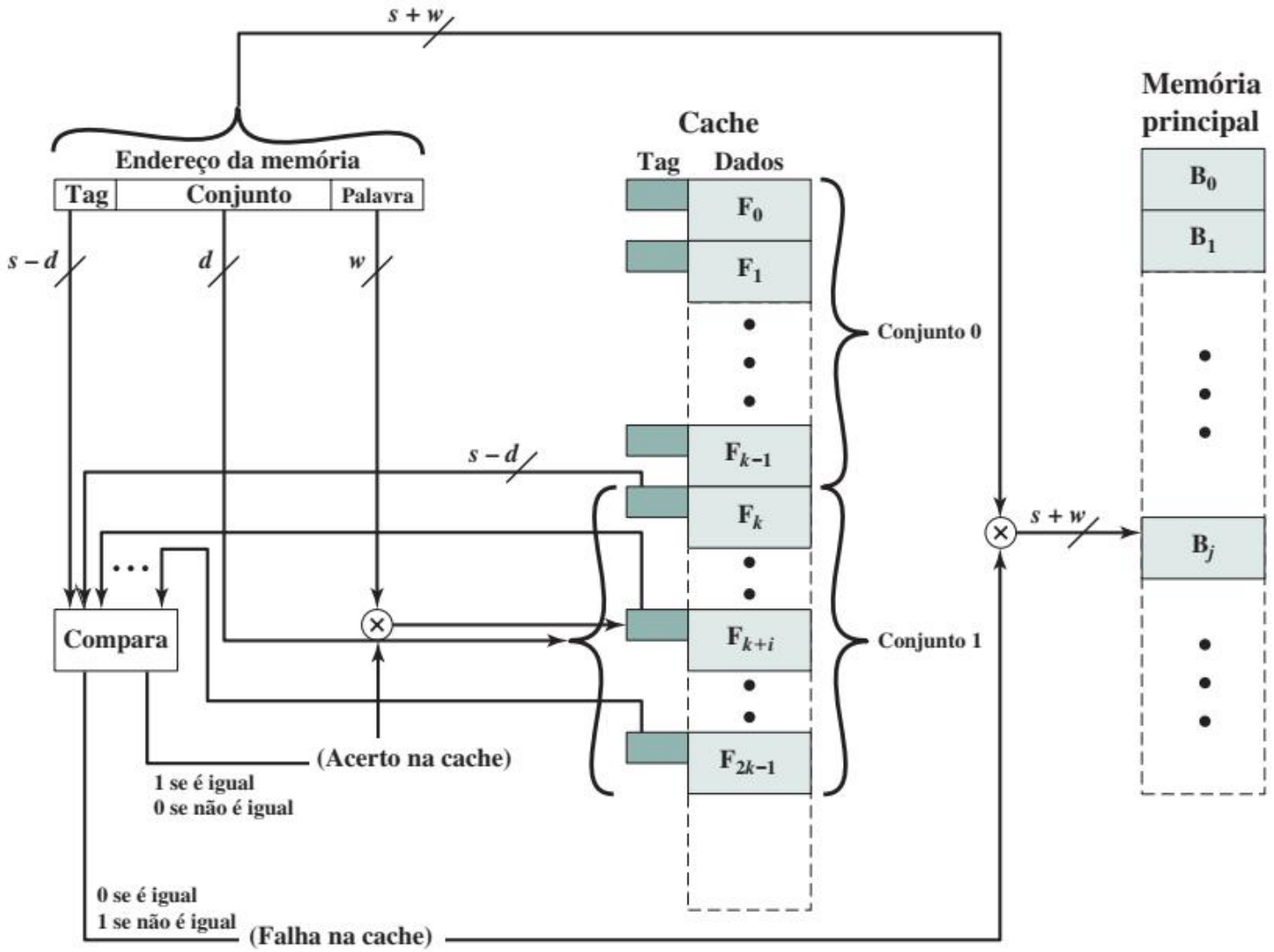
Descrição

- A apresentação na tela:
  - A todo momento, durante a execução do seu programa, ele deverá apresentar todas as informações necessárias sobre a MP e memória cache;
  - Além disso, considerando a memória cache, deverá apresentar a linha/conjunto onde acessou com 2 linhas/1 conjunto antes e depois;
  - Também, se um conjunto da cache encheu, deve apresentar qual linha foi substituída;
  - Ao final, seu programa deve apresentar a memória cache resultante, as taxas de falhas e acertos e quantas posições sofreram substituições devido ao LFU.

Mapeamento Associativo por Conjunto

- **Mapeamento**: implementado por meio do **endereço** da MP.
- Cada endereço da MP pode ser visto como consistindo em **três campos:**
  - Os *w bits* menos significativos identificam uma palavra ou um byte dentro de um bloco da MP;
  - Os *s bits* restantes especificam um dos 2^s **blocos** da MP;
  - A **lógica de cache** interpreta esses *s bits* como uma ***tag*** de *s - d bits* (parte mais significativa) e um campo de conjunto de d *bits*.
  - O segundo campo identifica um dos $v = 2^d$ conjuntos da cache.

Mapeamento Associativo por Conjunto



Arquivos de Entrada (modo texto)

- **Tamanho da MP (no máximo 256KB)**
- **Qtde de palavras por bloco na MP (2, 4 ou 8)**
- **Tamanho da cache (no máximo 32KB)**
- **Número de linhas por conjunto da cache (mínimo de 2 linhas/máximo é número de linhas/2).**
- **Obs:**
  - Tamanho da palavra sempre será de 32 bits (4 bytes) e cada linha da MP conterá 1 palavra.
  - Seu programa deverá calcular:
    - Tamanho do endereço da MP;
    - Os valores para w, d, s e tag.

Data e Critérios

- Data de entrega do trabalho:
  - 19/06/2024.
- Critérios considerados para determinação da nota do trabalho:
  - Código limpo, bem elaborado e modularizado;
  - Comentários no código bem redigidos.
    - Cada função criada deve possuir um comentário bem escrito sobre a sua finalidade.
  - Se, no momento de correção, o programa não funcionar, por erro de compilação/interpretação ou por erro em tempo de execução, a nota considerada é zero;
  - Cópias também receberão nota zero.