Grupo Sigma

Simulador de Memória Cache

Caio Bernardo Brasil	202006840008
Heitor Mesquita Anglada	202006840018
Lucas Queiroz Costa	202006840012
Matheus Gabriel Pantoja	202006840039
Richard Douglas da Piedade	202006840011

01

INTRODUÇÃO

02

IMPLEMENTAÇÃO

03

CONCLUSÃO

INTRODUÇÃO

Durante o desenvolvimento da computação, a memória cache serviu para melhorar o desenvolvimento do computador, por ser uma memória mais rápida. Ela trabalha armazenando uma parte da memória principal de forma temporária para ser acessada de forma mais rápida.

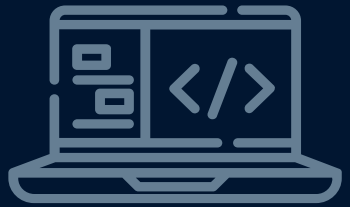
IMPLEMENTAÇÃO

- Desenvolvida na linguagem de programação Python.
- Foi implementado:

1. **Tipos de Mapeamento:**
   - Mapeamento Direto
   - Mapeamento Associativo
   - Mapeamento Associativo por Conjunto

2. **Política de substituição:**
   - Escolha aleatória
   - FIFO (“primeiro a entrar, primeiro a sair”)
   - LFU (saí o bloco menos utilizado)

3. **Política de escrita:**
   - Write Thought
   - Write Back (Não foi possível a implementação desta política de escrita)



MAPEAMENTO DIRETO

1. A quantidade de linhas de cache precisar ser um expoente de 2.
2. Identificar a Linha_de_cache na linha do traço.
3. Comparar se a tag da linha do traço está na cache na posição de Linha_de_Cache.
4. Analisar o caso de Instrução para identificar acréscimos de HIT, MISS e Acesso_Memória_Principal.

MAPEAMENTO ASSOCIATIVO
SUBSTITUIÇÃO ALEATÓRIA

1. Ocupar todos os espaços vazios da cache com tags diferentes conforme a ordem das linhas do traço.

2. Quando todos os espaços forem ocupados, um número aleatório entre 0 e o número de linhas da cache – 1 é gerado.

3. O número gerado representa a posição da cache onde uma tag diferente substituirá a tag preexistente nessa mesma posição.

4. Em paralelo, o caso de instrução da linha de traço vai determinar o acréscimo de HIT, MISS e Acesso_Memória_Principal.

MAPEAMENTO ASSOCIATIVO
SUBSTITUIÇÃO DOS PRIMEIROS A ENTRAR

1. Ocupar todos os espaços vazios da cache com tags diferentes conforme a ordem das linhas do traço.

2. Quando todos os espaços forem ocupados, uma variável Z inicialmente em zero irá determinar qual a posição da cache que a nova tag, diferente das que já estão na cache, irá ocupar, substituindo a tag preexistente.

3. Após a substituição, o valor de Z é acrescido de 1, o que levará a substituição para o próximo espaço da cache, e quando o valor de Z for igual ao do número de linhas, seu valor retornará a 0 para que a tag mais antiga possa ser substituída.

4. Em paralelo, o caso de instrução da linha de traço vai determinar o acréscimo de HIT, MISS e Acesso_Memória_Principal.

MAPEAMENTO ASSOCIATIVO
LFU

1. Ocupar todos os espaços vazios da cache com tags diferentes conforme a ordem das linhas do traço.

2. Enquanto os espaços da tag estão sendo ocupados, um vetor de frequência analisará a quantidade de vezes que uma tag é repetida numa posição da cache e também vai analisar qual a posição da cache em que houve a menor frequência de tag.

3. Após a ocupação de todos os espaços da cache, quando uma tag diferente de todas que estão na cache é analisada, ela substituirá a tag na posição em que houve a menor frequência. Caso haja mais de uma posição na cache em que a frequência é a menor, a substituição ocorrerá na primeira em que isso que foi constatado a frequência mínima.

4. Em paralelo, o caso de instrução da linha de traço vai determinar o acréscimo de HIT, MISS e Acesso_Memória_Principal.

MAPEAMENTO ASSOCIATIVO POR CONJUNTOS
SUBSTITUIÇÃO ALEATÓRIA

1. A divisão entre o número de linhas da cache e a número de linhas por conjunto precisa ser igual a um expoente de 2.

2. Identificar na linha de traço qual conjunto a tag pertence. As tags vão ocupando em ordem o conjunto até que ele não possua mais espaços vazios.

3. Quando um conjunto estiver cheio e uma nova tag diferente for analisado nesse mesmo conjunto, um número aleatório entre a menor posição da cache a qual esse conjunto pertence **(0 + (conjunto_no_traço * numero_Linhas_conjunto))** até a maior posição da cache a qual esse conjunto pertence ***((numero_Linhas_conjunto – 1) + (conjunto_no_traço * numero_Linhas_conjunto))*** é gerado.

4. Analisar o caso de Instrução para identificar acréscimos de HIT, MISS e Acesso_Memória_Principal.

MAPEAMENTO ASSOCIATIVO POR CONJUNTOS

SUBSTITUIÇÃO DOS PRIMEIROS A ENTRAR

1. A divisão entre o número de linhas da cache e a número de linhas por conjunto precisa ser igual a um expoente de 2.

2. Identificar na linha de traço qual conjunto a tag pertence. As tags vão ocupando em ordem o conjunto até que ele não possua mais espaços vazios.

3. Quando um conjunto estiver cheio e uma nova tag diferente for analisado nesse mesmo conjunto, haverá uma variável responsável por analisar qual a foi a primeira tag adicionada no conjunto, e essa mesma variável será a posição na cache na qual ocorrerá a substituição de tag.

4. Em paralelo, o caso de instrução da linha de traço vai determinar o acréscimo de HIT, MISS e Acesso_Memória_Principal.

MAPEAMENTO ASSOCIATIVO POR CONJUNTOS

LFU

1. A divisão entre o número de linhas da cache e a número de linhas por conjunto precisa ser igual a um expoente de 2.

2. Identificar na linha de traço qual conjunto a tag pertence. As tags vão ocupando em ordem o conjunto até que ele não possua mais espaços vazios, além disso haverá um contador de frequência de cada linha do conjunto.

3. A substituição ocorrerá na posição do conjunto da cache em que a frequência for a mínima.

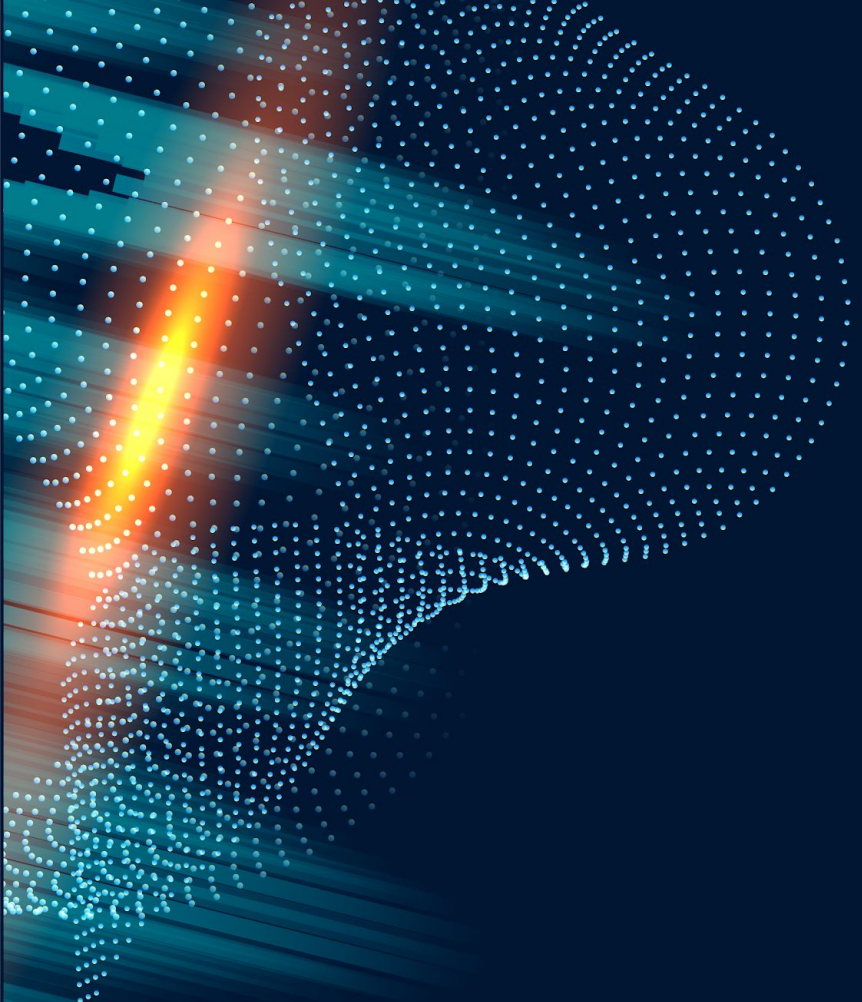
4. Em paralelo, o caso de instrução da linha de traço vai determinar o acréscimo de HIT, MISS e Acesso_Memória_Principal.

CONCLUSÃO

Ao final, conseguimos atingir o objetivo de criar o simulador dado e analisar o seu funcionamento. Com eles temos 7 tipos diferentes de combinações das características de uma memória cache. Apesar disso, nosso simulador tem uma limitação em relação a política de escrita Write Back, a partir do nosso conhecimento não foi possível implementar esse tipo de escrita na linguagem de Python.

BIBLIOGRAFIA

1. William Stallings, “Arquitetura e Organização de Computadores” Person, Ed. 10, 2018.



FIM.