Relatório do Trabalho 3 Organização e Arquit. de Computadores

Willian Magnum Albeche,Arthur Schossler Dutra
Escola Politécnica - PUCRS

Resumo

Este documento tem como objetivo apresentar um relatório de desenvolvimento do trabalho 3 da disciplina de Organização e Arquitetura de Computadores ministrada pelo Prof.Dr.Sergio Johann Filho em 2020/2 para a turma 127.

Introdução

Neste trabalho foi proposto no enunciado 3 exercícios sobre o conteúdo de hierarquia de memória, com um cofo na estrutura e no funcionamento de cache, que é encontrado no CPU(processador) e como o mesmo se relaciona com a memória RAM, que no caso deste trabalho será a DRAM(*Dynamic Random Access Memory).*

1.Tipos de mapeamentos utilizados:

Mapeamento direto:

Tag = 9 bits

Linha = 3bits

Palavra = 3bits

Bit de seleção = 1bit

TAG									LINHA			PALAVRA			BIT DE SELEÇÃO
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Tag = 9 bits

Linha = 4bits

Palavra = 2bits

Bit de seleção = 1bit

TAG									LINHA				PALAVRA		BIT DE SELEÇÃO
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Willian.Albeche@edu.pucrs.br
Arthur.Dutra@edu.pucrs.br

Mapeamento associativo:

TAG												PALAVRA			BIT DE SELEÇÃO
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Tag = 12bits

Palavra = 2bits

Bit de seleção = 1bit

TAG													PALAVRA		BIT DE SELEÇÃO
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Tag = 13bits

Palavra = 2bits

Bit de seleção = 1bit

2. Resolução

Para nossa segunda resolução, decidimos criar um programa em java, que iria realizar o cálculo de Hits e Miss da memória cache, para sua entrada, temos um arquivo .txt, onde contém todos os endereços, na sua saída, lista os endereços de memória, com hit ou miss(dependendo do resultado) a quantidade de hits e sua porcentagem referente ao conjunto todo

Implementação:

O código foi feio em uma classe só para ser algo simples porém funcional, separado por métodos e suas funcionalidades.

A seguir iremos destacar os principais métodos.

Main:

Na main, temos o "menu" que foi feito visando facilitar a interação com o usuário ou com quem estiver usando, na main temos validações para garantir a integridade do sistema, e a entrada dos valores do arquivo txt, com os endereços que deseja ser acessado e no o menu temos as opções para que se possa escolher o tipo de mapeamento que se deseja utilizar

direto:

Neste método, realizamos o mapeamento direto usando duas formas, um com a linha sendo composta por 3 bits, e o outro com a mesma sendo composta por 4 bits, ele envia isso para o método que iremos falar a seguir, após isso, seu resultado retorna para este método que já imprime o resultado formatado, com números de Hits e Miss e sua porcentagem.

addicionarAoCache:

Como citado anteriormente, este método realiza tanto o mapeamento com o a linha tendo 3 bits quanto a linha tendo 4 bits, comparando as Tags e linhas e retornando um Hit e Miss para o método que irá imprimir o resultado

associativo:

neste método, realizamos o mapeamento associativo, usando as duas formas, um com a tag sendo composta por 16 bit, e a com a tag de 12 bits, enviando os endereços para o método que cuida de conferir as tags e retorna o Hit e Miss para sua contagem, imprimindo assim, os números de Hits e Miss e sua porcentagem, com o resultado formatado.

addicionarAoCacheAssociativo:

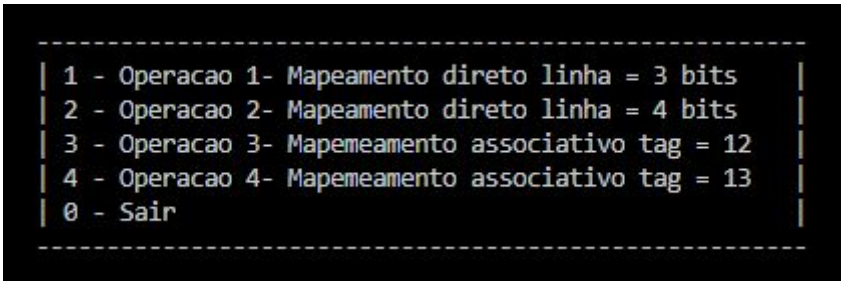
Da mesma forma que o método de mapeamento direto, este método realiza as operações para as duas configurações de mapeamento associativo utilizadas, comparando as tags e retornando os seus Hits para o método que imprime, ele é muito parecido com o método para o mapeamento direto, entretanto neste, só analisamos a tag, pois não usamos a linha para o associativo.

Startup:

Este método lê o arquivo contendo o endereço em hexadecimal e envia cada endereço para um método que os converte em binário e seu resultado retorna para o método, armazenado os mesmo em um ArrayList.

Menu:

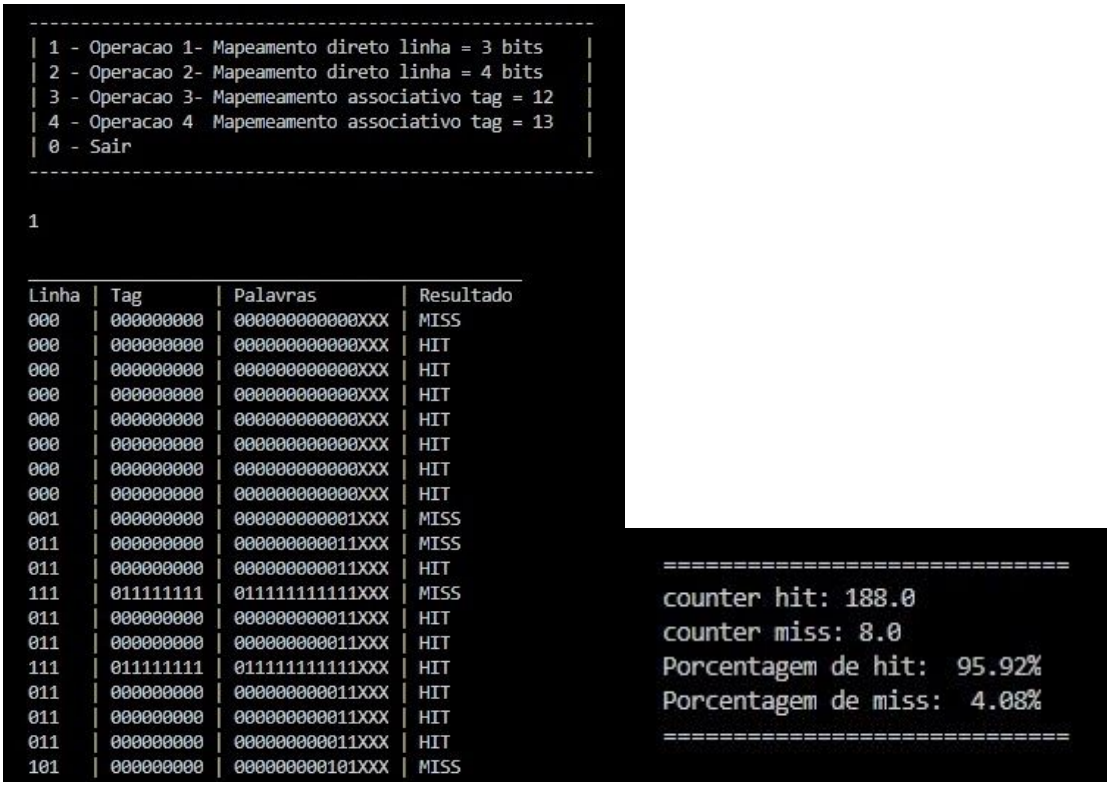
Criamos o menu para a seleção do tipo de mapeamento, utilizando switch case dentro de um loop “do while” para realizar todas as operações necessárias sem precisar fechar e abrir o programa várias vezes.



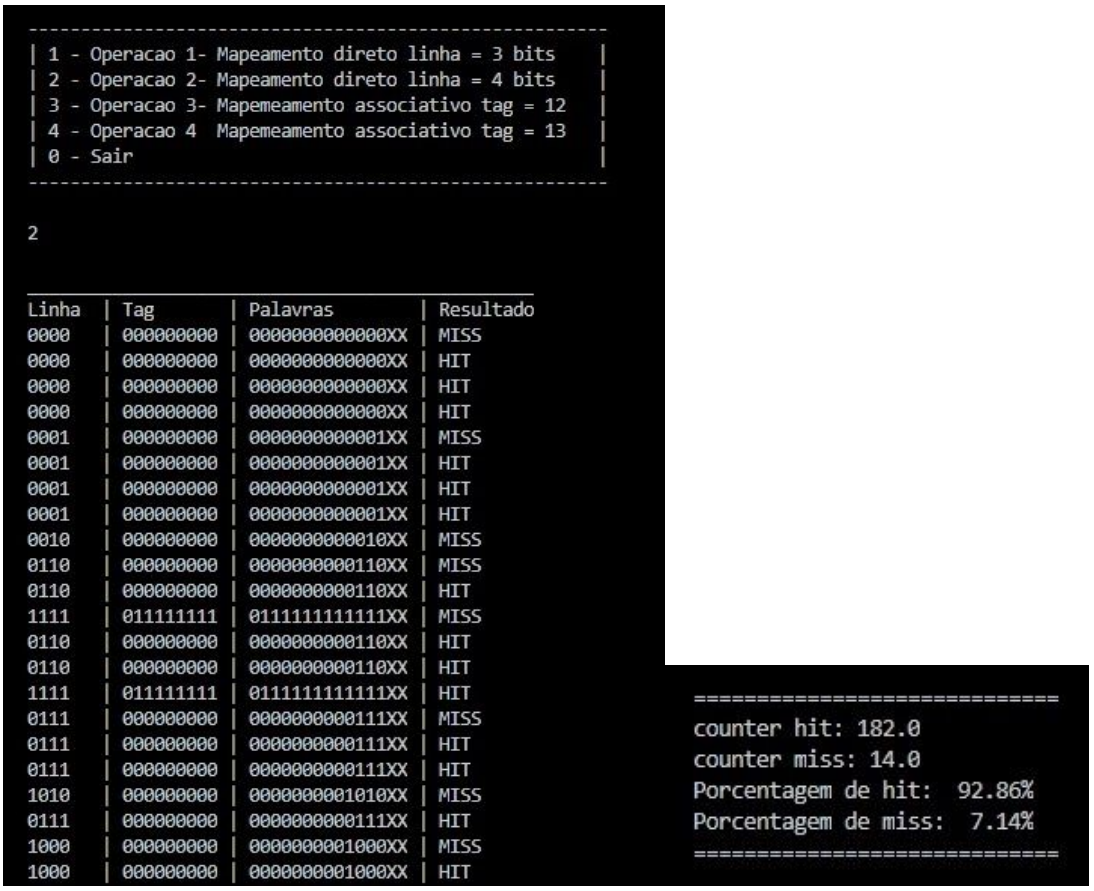
Mapeamento direto:

Para o mapeamento direto, temos um campo chamado linha, onde cada linha tem o valor final da referência de memória. Neste modelo cada espaço é dividido por um número fixo de referências na memória RAM.

2.1



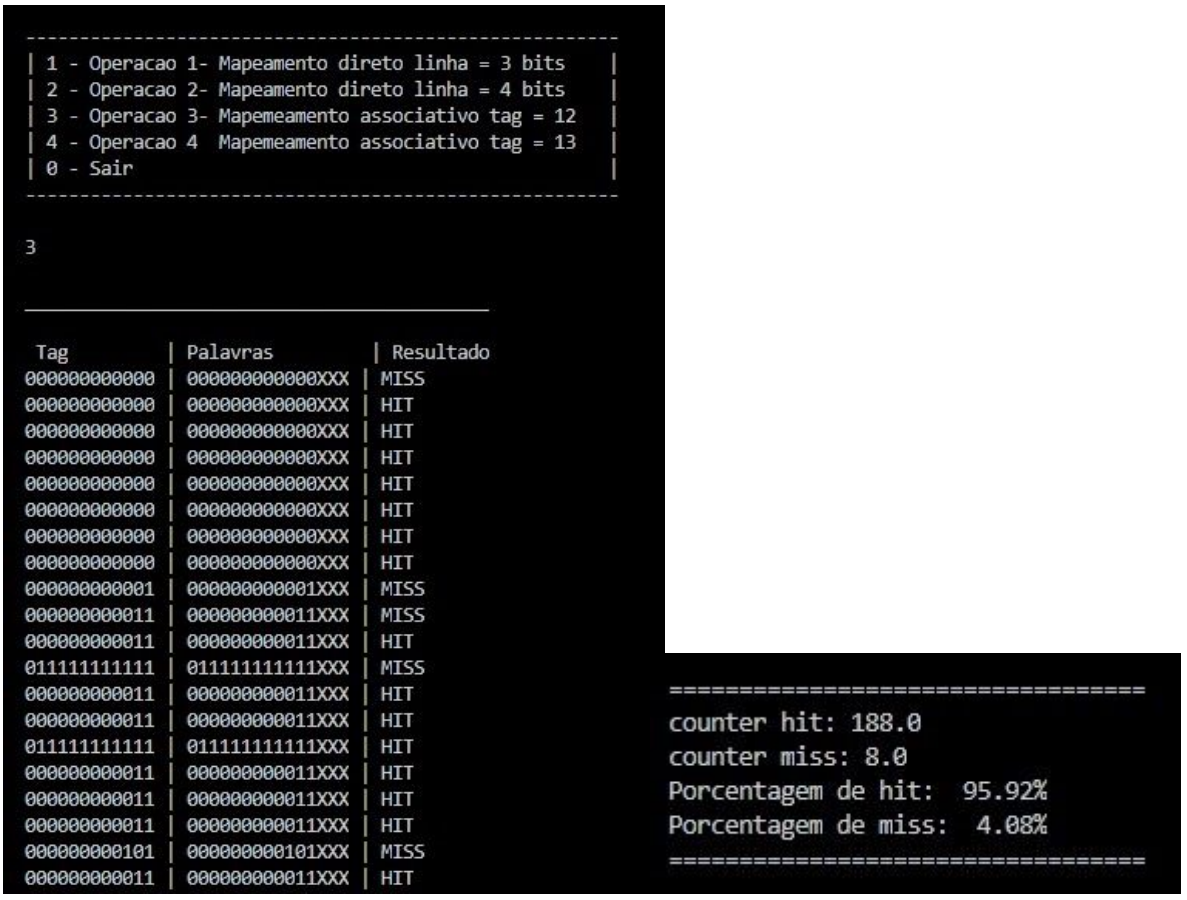
Mapeamento direto, com 9 bits para tag, 3 bits para linha



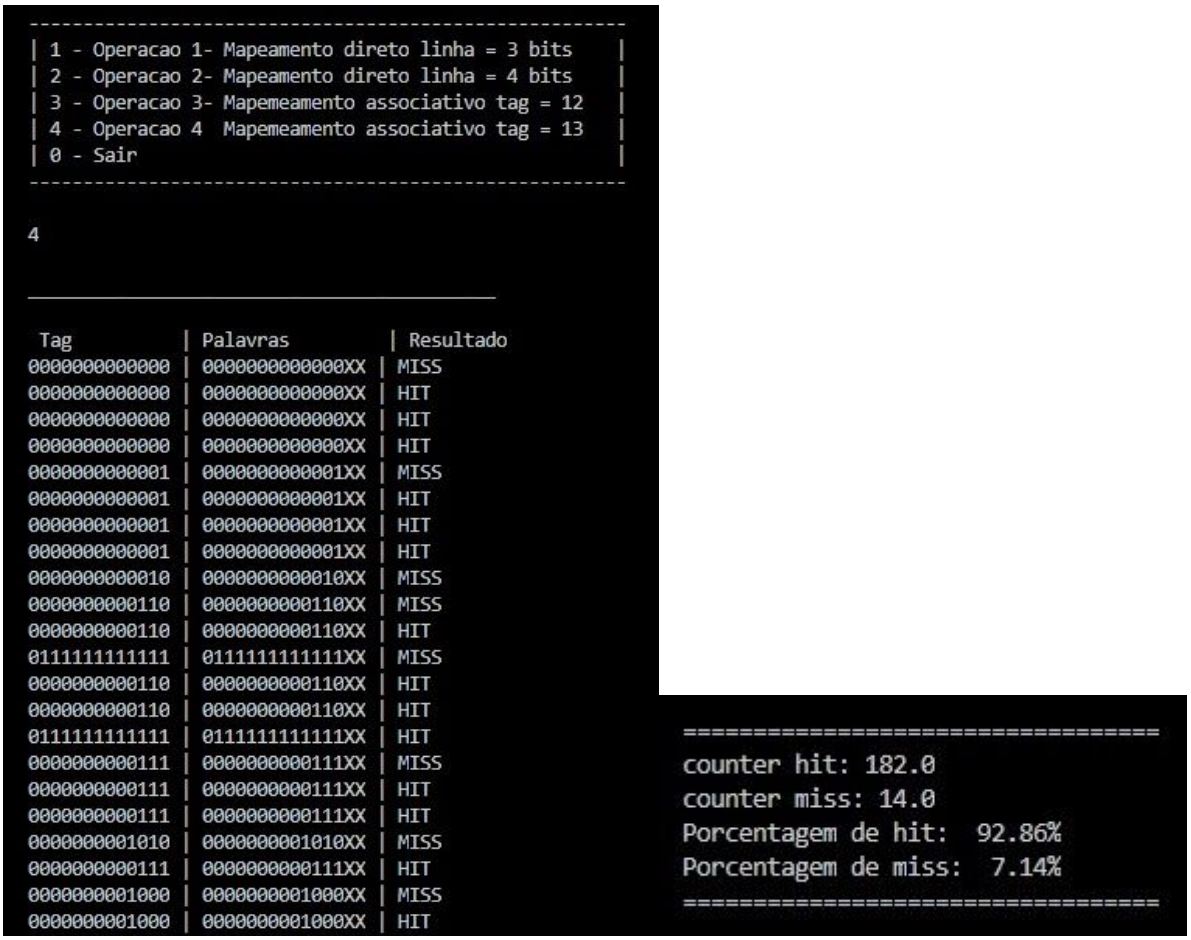
Mapeamento direto, com 9 bits para tag, 4 bits para linha

Mapeamento associativo:

O mapeamento associativo ele é mais flexível, pois qualquer linha tem apenas uma tag, que é atribuída livremente para qualquer referência desejada na memória, é possível também manter o conteúdo muito requisitado e liberar o menos requisitado, assim diminuindo o custos de atribuições com conteúdos repetidos.



Mapeamento associativo, com 12 bits para tag



Mapeamento associativo, com 13 bits para tag

Conclusão:

Após analisarmos os resultados, conseguimos perceber que a porcentagem de acerto, era reduzida conforme aumentamos o número de linhas, de 3 bits para 4 bits, essa redução não é proporcional e muitas vezes pouco perceptível, girando em torno de 2%. Então pode não ser muito proveitoso ter um grande espaço disponível de cache, visto que não temos um aumento de performance que justifique esse aumento de cache, que muitas vezes pode até acarretar em gastos desnecessários, não tendo uma boa relação custo x benefício, em relação a questão monetária.

Vale ressaltar que o mapeamento associativo seja mais vantajoso em alguns casos, visto que não tem um método de divisão a variação de endereço por linha, é menor e mais eficiente, com isso, usa todo o espaço disponível inicialmente.