UNIVERSIDADE ESTADUAL DO PARANÁ

CAMPUS APUCARANA

Rafael Francisco Ferreira

Luís Fernando S. Gaspar

Simulador de Memória Cache

Apucarana

2016

**SUMÁRIO**

Tópico Página

1. Introdução ..............................................................................3
2. Objetivos/Justificativa.............................................................3
3. Funcionamento da memória cache........................................4
4. Funções de mapeamento.......................................................5
5. Algoritmos de substituição......................................................8
6. Política de escrita write-back................................................11
7. Decisões de projeto para a implementação: .......................12

* Estruturas de dados utilizadas
* Tamanho da memória cache
* Escolha dos algoritmos de substituição
* Escolha da linguagem

1. Conclusão.............................................................................13
2. Referências...........................................................................14
3. **INTRODUÇÃO**

Este trabalho tem como finalidade expor o desenvolvimento e funcionamento de um simulador de memória cache – parte integrante da disciplina de Arquitetura e Organização de Computadores do curso de Ciência da Computação.

1. **OBJETIVOS/JUSTIFICATIVA**

O simulador foi criado com o fim didático de mostrar o funcionamento da memória cache em conjunto com a memória principal, bem como suas funções de mapeamento, métodos de acesso e modificação e a politica write-back.

1. **FUNCIONAMENTO DA MEMÓRIA CACHE**

No simulador, a cache atua alocando um bloco da memória principal, formado por dois valores, em uma de suas quatro linhas utilizando o Mapeamento Associativo, ou seja, um bloco pode ser alocado em qualquer linha da cache, mesmo se ela já estiver armazenando algum bloco naquela posição.

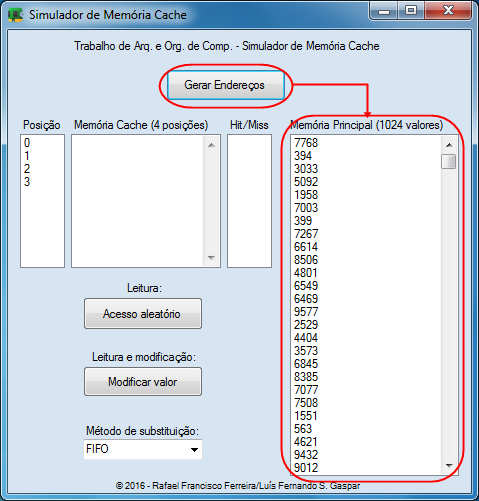
Após fazer o Mapeamento Associativo até preencher as quatro posições da cache, ela estará cheia. Nesse momento o usuário precisará selecionar um dos métodos de substituição (Aleatório ou FIFO) para que os novos valões de entrada sejam alocados em uma linha da cache, substituindo o valor que está nela.

Se o novo valor de entrada for igual a algum valor já alocado na cache, ocorre o chamado Cache HIT, e não é utilizada a politica de Write-Back na memória principal. Se o novo valor de entrada ainda não existir na cache, ocorrerá um Cache MISS, e o valor que foi substituído na cache terá seu bloco correspondente substituído na memória principal, assim sendo feita a política de write-back.

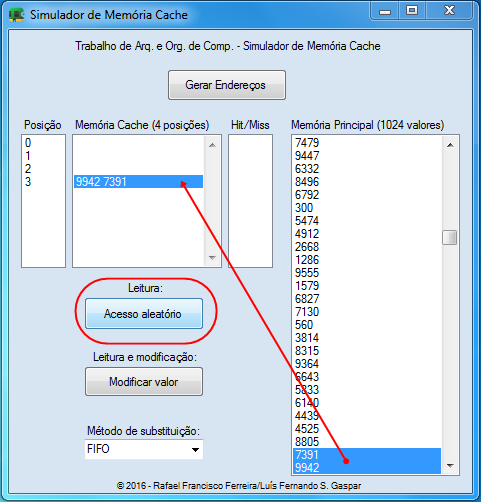
O processo de modificação se repete quantas vezes for necessário enquanto a cache estiver ativa.

1. **FUNÇÕES DE MAPEAMENTO**

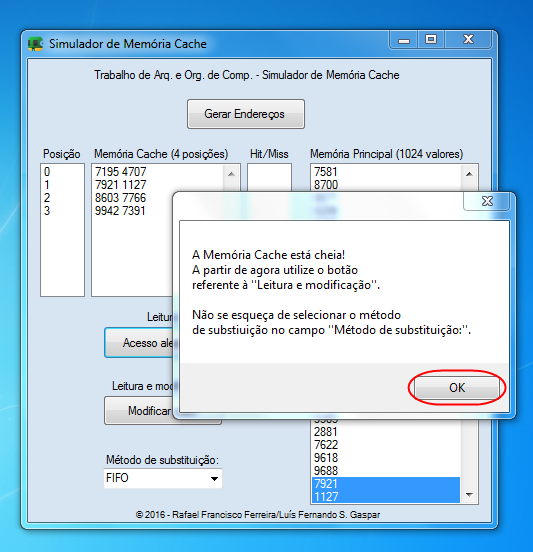
No simulador, a memória cache só entra em ação após serem gerados os valores da memória principal. Para isso, clique no botão “Gerar Endereços”.



Após os valores serem gerados, você deve clicar no botão “Acesso aleatório”. Este executará a função de Mapeamento Associativo, ou seja, irá alocar um bloco da memória principal em uma linha aleatória da memória cache.

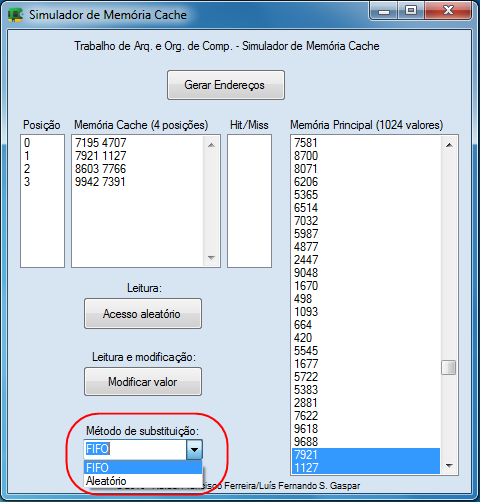


Repita o passo acima até que a seguinte mensagem apareça. Após isso, clique em “OK”.

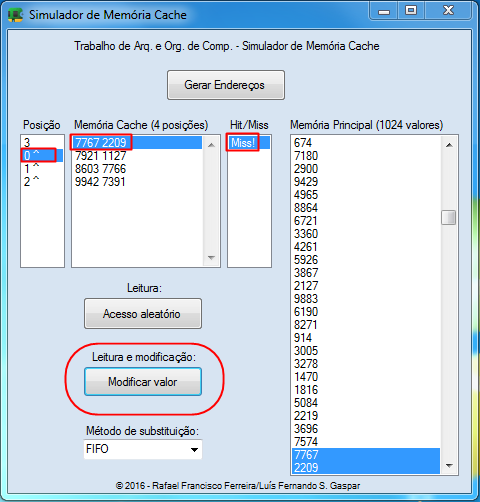


1. **ALGORITIMOS DE SUBSTITUIÇÃO**

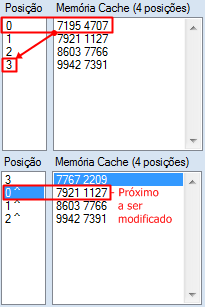
Após ter completado os passos acima, você será instruído a utilizar um dos métodos de substituição, sendo as opções: **FIFO** e **Aleatório**.



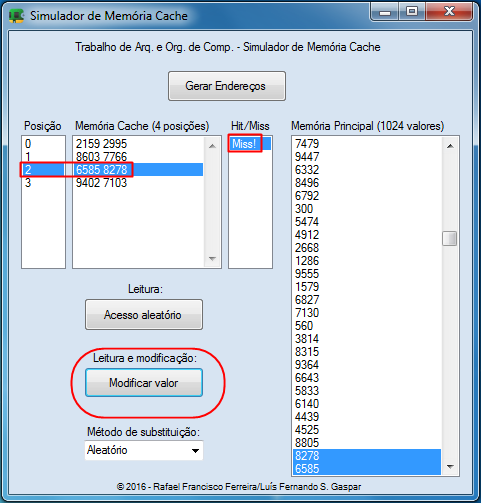
A começar pelo **FIFO**, clique no botão “Modificar valor” e o valor encontrado na posição 0 (zero) será modificado.



Note que, no caso do FIFO, quando um valor substitui o valor da posição zero, ele na verdade acabou de entrar na cache, então será movido para a posição três. Sendo assim, o valor que estava na posição 1 passa para a posição 0 para que ele seja o próximo a ser modificado (sendo ele o novo “First In”). E assim ocorre para todos os valores.



No caso do **Aleatório**, logo após você clicar em “Modificar valor”, o algoritmo substituiu um valor de uma posição aleatória na cache por um novo valor de entrada. Caso esse novo valor de entrada não exista na cache, ocorre um Cache MISS. É nesse ponto que o Write-Back ocorre.

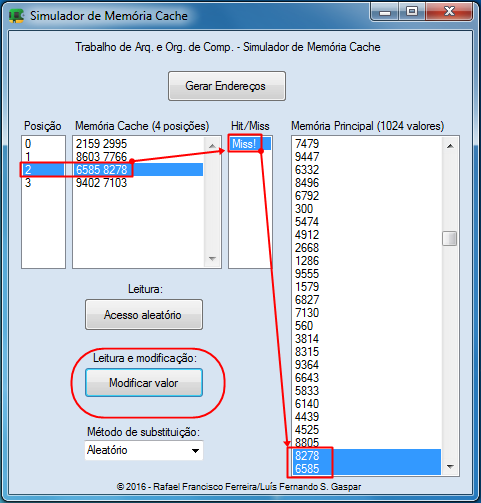


**NOTA:** Neste método, o programa tem um pequeno *bug*. Caso você clique em “Modificar valor” e o valor selecionado não for modificado, não se assuste, basta clicar novamente e o mesmo será modificado.

1. **POLÍTICA WRITE-BACK**

Para explicar a Politica de Escrita Write-Back, vamos tomar como exemplo o uso do algoritmo de substituição Aleatório.

Após detectar o Cache MISS, o valor será adicionado à linha aleatória escolhida pelo algoritmo, substituindo o valor encontrado naquela linha da cache. Logo após essa substituição, ocorrerá a substituição dos valores correspondentes àquela linha na memória principal, ou seja, como o bloco foi modificado na linha da cache, os valores desse mesmo bloco serão modificados na memória principal.



1. **DECISÕES PARA IMPLEMENTAÇÃO**

* **Estruturas de dados utilizadas:**

As estruturas escolhidas foram as **listas**. A partir delas, simulamos o funcionamento de outras estruturas, como a fila, no caso do método FIFO.

* **Tamanho da memória cache:**

O tamanho da cache é de 4 (quatro) posições. Este valor foi o escolhido pois facilita a visualização do funcionamento do mapeamento associativo, bem como o do método de substituição aleatório, também ajudando na hora da correção do trabalho.

* **Escolha dos algoritmos de substituição:**

Os algoritmos Aleatório e FIFO foram os escolhidos pois ambos facilitavam o trabalho de desenvolvimento utilizando as listas e ListBox do Windows Forms, e devido ao baixo nível de conhecimento em C#, acreditamos que seria benéfico para o funcionamento correto do software.

* **Escolha da linguagem:**

A linguagem escolhida foi C#, pois visualizamos a facilidade de se trabalhar com estruturas de dados na mesma.

Este foi o primeiro contato com desenvolvimento em C#, e acreditamos que seria ainda mais didático desenvolver o software em uma linguagem desconhecida.

1. **CONCLUSÃO**

Vimos que o funcionamento de uma memória cache depende de vários fatores, como suas funções de mapeamento, métodos de substituição e politicas de escrita, bem como sua comunicação com a CPU e a memória principal.

Como visto nas aulas, quando os valores da RAM são gerados, a cache precisa buscas seus blocos e aloca-los em uma de suas linhas, analisando se o valor buscado é par ou ímpar para formar o bloco. Após estar cheia, é necessário o uso de um método de substituição para que possam ser armazenados novos valores.

Os resultados obtidos por este trabalho estão em plena concordância com o conhecimento obtido nas aulas da disciplina. Tudo ocorre como na teoria, insto é, não encontramos discordância entre os resultados encontrados e aqueles previstos pela teoria existente.

Como esta é uma atividade relativamente simples e com resultados bastante coerentes, o software desenvolvido nela permite que qualquer pessoa entenda o funcionamento de uma memória cache, tornando-se assim também uma possível ferramenta didática.

Convêm deixar claro, entretanto, que o software ainda pode ser mais bem explorado e melhor elaborado, com melhor precisão de resultados e mais funcionalidades. E isso é perfeitamente natural, pois o conhecimento científico está em constante evolução e as necessidades se tornam cada vez mais exigentes.

1. **REFERÊNCIAS**

* Stallings, William; Arquitetura e organização de computadores; 8ª edição, São Paulo;

Pearson Pratice Hall, 2010.

* <https://msdn.microsoft.com/pt-br>