AULA 2 – MEMÓRIA CACHE E MEMÓRIA PRINCIPAL

OBJETIVO DA AULA

Abordar o relacionamento entre memória principal e memória secundária e sua influência no desempenho do computador.

APRESENTAÇÃO

Olá!

Nesta aula vamos conhecer melhor o relacionamento entre as memórias cache e principal.

Veremos aqui de que forma a cache, que é mais rápida, contribui para a melhoria do desempenho do computador.

Mas antes disso vamos entender um pouco do funcionamento de cada uma delas, bem como suas vantagens e limitações.

Mãos à obra!

1. MEMÓRIA CACHE E MEMÓRIA PRINCIPAL

Na última aula, estudamos a hierarquia de memórias e vimos a importância de dividir o sistema de armazenamento do computador em diferentes níveis hierárquicos para garantir uma relação custo/benefício viável para os computadores.

Agora vamos ver como funcionam a memória cache e a principal, já que são nelas que ficam os programas que executamos em nosso computador.

Antes de mais nada, comentaremos aqui um problema chamado *Gargalo de Von Neumann*, que está presente em todos os nossos computadores, já que são Arquiteturas de Von Neumann.

De forma simples, o Gargalo de Von Neumann é o problema que acontece pelo fato de que processador ter uma velocidade muito maior do que as memórias. Sendo assim, sempre que precisa acessá-las, ele passa muito tempo aguardando a conclusão de seus pedidos de leitura e escrita.

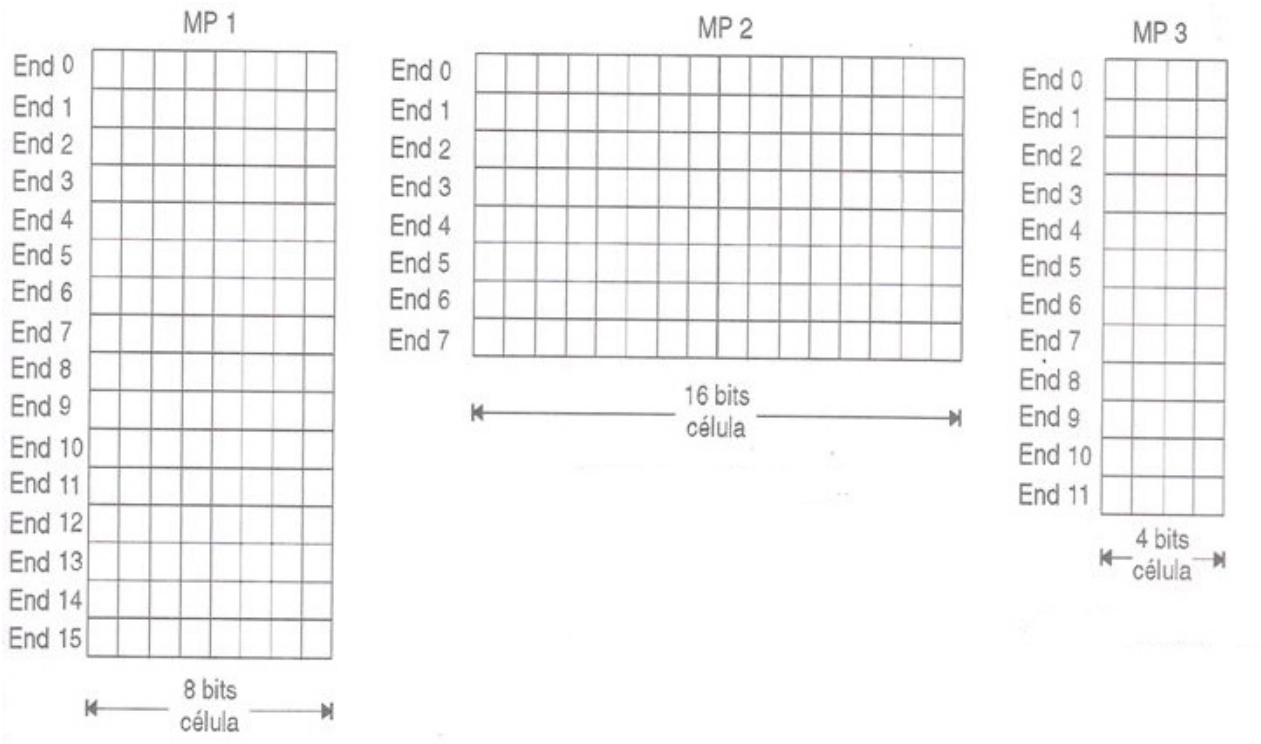
Portanto, esse problema afeta diretamente a performance do computador, uma vez que o processador tem seu trabalho prejudicado.

Há diversas formas de tentar mitigar esse problema e uma delas é a introdução das memórias cache.

Como vimos na pirâmide da aula passada, a memória cache é um intermediário entre o processador e a memória principal, sendo mais veloz, porém com menor capacidade de armazenamento.

A memória principal é estruturalmente organizada em células, cada uma identificada por um endereço que identifica a localização física da célula na memória.

FIGURA 1 | **Exemplos de organização de memória**



Fonte: http://www.dpi.inpe.br/~carlos/Academicos/Cursos/ArqComp/aula_4.html.

Na Figura 1 temos três tipos de memória. Na primeira as células possuem oito bits cada temos dezesseis endereços (0 a 15) para localizar cada célula. Na segunda são células de dezesseis bits e temos oito endereços (0 a 7). Finalmente, na terceira, temos doze células de quatro bits.

O conteúdo da memória é acessado através dos endereços das células.

A memória principal é formada por capacitores e estes têm como uma de suas propriedades físicas o fato de "perderem" os dados depois de um tempo. Isso torna necessária uma operação chamada *refresh*.

O *refresh* é uma varredura, bit a bit, na memória principal para "lembrar" a cada um deles o seu valor (0 ou 1). Essa operação ocorre várias vezes por segundo e é em razão dela que a memória principal é chamada de DRAM (Dynamic RAM).

Por ocorrer várias vezes por segundo, o *refresh* contribui para o aumento do tempo de acesso da memória principal, tornando-a mais lenta.

Já a memória cache é formada por circuitos que dispensam o *refresh*, porém geram mais calor e por isso não permite uma alta densidade de integração, consumindo também mais energia. Ela é chamada de SRAM (*Static RAM*).

Observe o quadro abaixo.

Quadro 1 | **Comparação entre memórias DRAM e SRAM**

	DRAM (principal)	SRAM (cache)
Vantagens	Alta densidade de integração; Menor consumo de energia; Menos geração de calor; Custo menor.	Alta velocidade; Não precisa de *refresh*.
Desvantagens	Baixa velocidade; Necessidade de *refresh*.	Baixa densidade de integração; Maior consumo de energia; Mais geração de calor; Custo maior.

Fonte: elaboração própria.

Pela Tabela 1 podemos entender porque a cache é mais rápida (não precisa de *refresh*) e porque tem menor capacidade de armazenamento (baixa densidade de integração, ou seja, menos bits por espaço físico).

Agora vamos ver de que forma a cache contribui para a melhoria do desempenho do processador.

Como sabemos, a cache é mais rápida e menor do que a memória principal. A utilização da cache, portanto, se dará da seguinte forma: nela ficará somente o que for mais importante para o processador a cada momento. Isto significa que o processador terá quase sempre aquilo de que precisa no meio de armazenamento mais rápido.

Mas como saber o que é mais provável de ser necessário ao processador?

Isso é possível devido a uma característica de comportamento previsível dos programas, o que permite deduzirmos o *princípio de localidade*, que é subdividido em:

Princípio da localidade espacial, segundo o qual, sempre que o processador acessar um endereço, ele provavelmente acessará também os endereços vizinhos.

Princípio da localidade temporal, segundo o qual, sempre que o processador acessar um endereço, ele provavelmente voltará a acessar esse endereço novamente em um curto espaço de tempo.

Graças ao princípio da localidade, é possível saber na maioria das vezes o que é mais provável de ser necessário ao processador a cada instante. Isto porque os programas apresentam um comportamento parecido na maioria das vezes.

Então, como a cache armazena o que é mais provável de ser necessário, o processador, toda vez que precisa buscar algo na memória, tenta primeiro encontrar aquilo que precisa na cache. Caso não consiga, ele vai procurar na memória principal.

Quando o processador consegue o que precisa na cache, chamamos esse evento de *cache hit* (hit de cache – sucesso na cache) e quando não consegue, chamamos esse evento de *cache miss* (miss de cache – falta na cache).

Porém, na ocorrência de um *cache miss*, o processador tem um trabalho, que envolve os seguintes passos:

1) Ir à memória principal (mais lenta) achar o que não achou na cache;

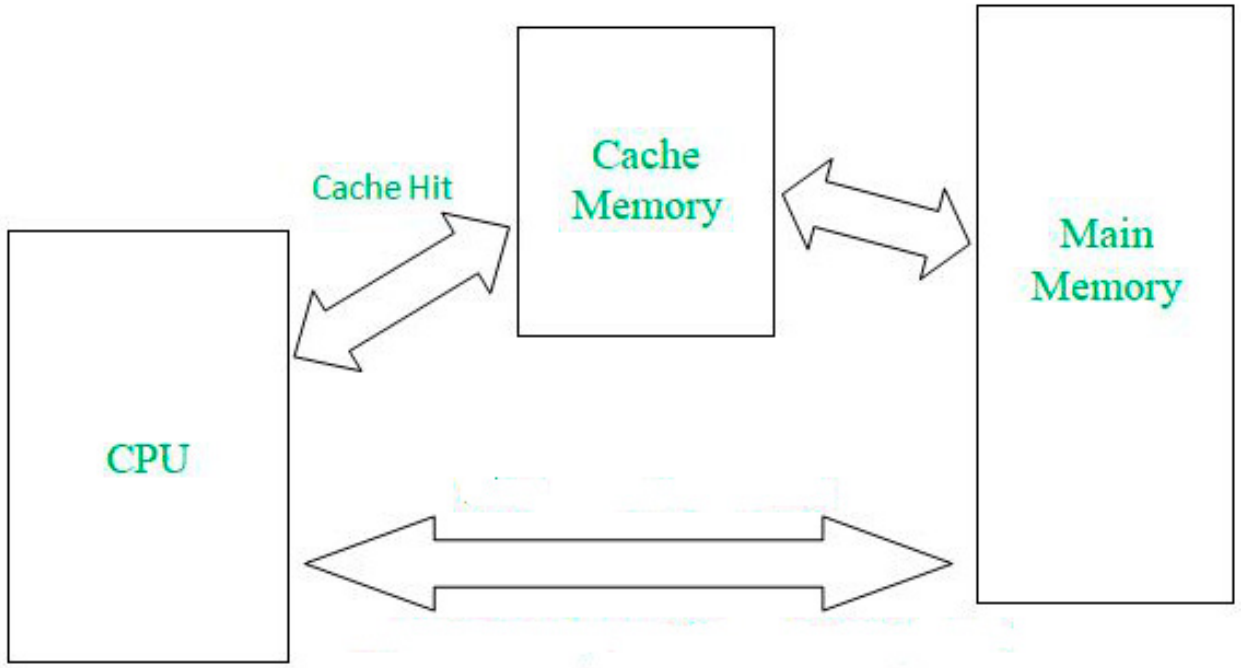
2) Trazer o que pegou na memória principal para a memória cache, já que, pelo princípio de localidade, provavelmente precisará acessar essas informações novamente num breve espaço de tempo;

3) Caso a cache esteja cheia, escolher qual dos blocos presentes nela dará lugar àquele que acabou de chegar;

4) Efetuar essa substituição.

A essa sequência de passos damos o nome de *miss penalty*.

FIGURA 2 | **Relacionamento entre processador e sistema de memórias**



Fonte: https://acervolima.com/acessos-simultaneos-e-hierarquicos-ao-cache/.

Importante ressaltar que as informações são transferidas entre as memórias principal e cache em blocos por conta da localidade espacial.

Para o correto funcionamento do esquema cache/memória principal, é importante considerarmos as seguintes questões:

1) Como escolher o bloco a ser substituído quando é necessário trazer um novo bloco da memória principal para a cache?

2) Como fazer a correlação entre os endereços de memória principal e cache, já que se trata de duas memórias fisicamente distintas?

À primeira questão respondemos: através dos algoritmos ou políticas de substituição de blocos.

À segunda questão respondemos: através do mapeamento de cache.

As políticas de substituição de blocos são as seguintes:

- FIFO (*First In First Out*) – este algoritmo funciona basicamente como uma fila, ou seja, o bloco que está há mais tempo na cache (o primeiro que chegou) é escolhido para ser substituído;
- LFU (*Least Frequently Used* – menos frequentemente usado) – de acordo com esse algoritmo, o bloco a ser substituído é aquele que tiver a menor quantidade de acessos;
- LRU (*Least Recently Used* – menos recentemente usado) – nesse caso, o bloco a ser utilizado é aquele que tiver sido usado pela última vez há mais tempo, ou seja, aquele que está há mais tempo ocioso.

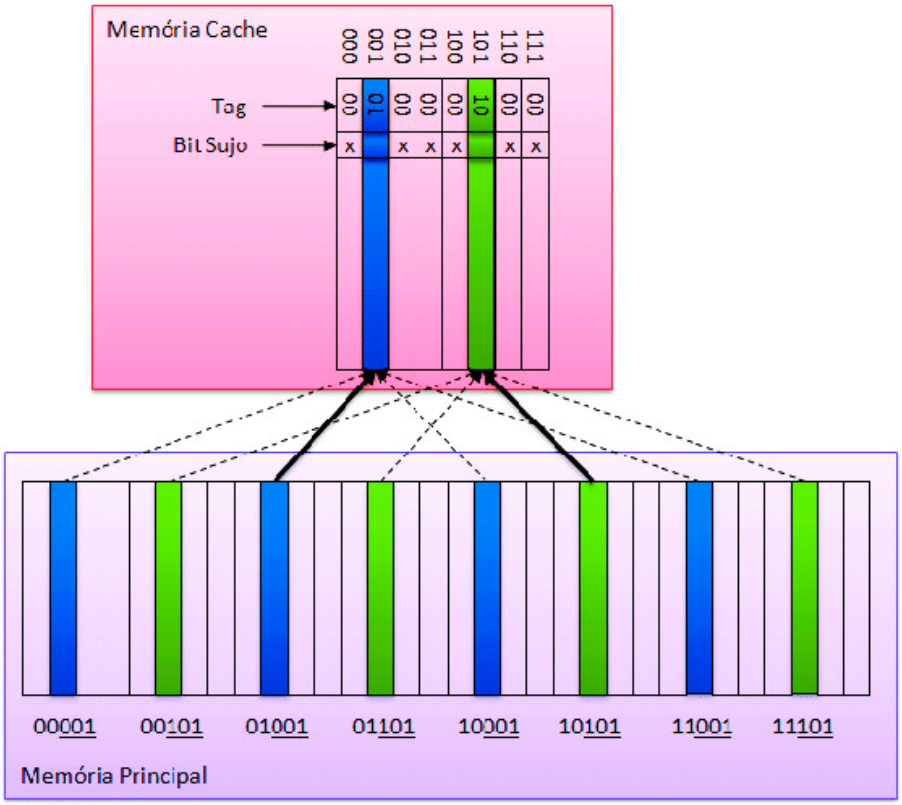
Quanto ao mapeamento da cache, também temos três tipos: o mapeamento direto, o mapeamento totalmente associativo e o mapeamento associativo por conjuntos.

Mapeamento direto – Neste mapeamento, cada bloco da memória principal é mapeado para um quadro da cache. O quadro a ser usado é obtido pelo resto da divisão do endereço do bloco da memória principal pela quantidade de quadros da cache. Cada quadro da cache tem três campos: o índice, o *tag* e o endereço de memória. O *tag* é usado para validar se a linha procurada é a mesma que está na cache.

Na Figura 3 podemos observar que, para cada quadro da cache, teremos a possibilidade de quatro blocos da memória principal. Assim, precisamos de dois bits no *tag* para identificar qual dos blocos da memória principal está carregado ali.

A desvantagem deste tipo de mapeamento é que se dois blocos da memória tiverem endereços com o mesmo resto de divisão, eles nunca poderão estar simultaneamente na cache, já que devem ocupar o mesmo quadro.

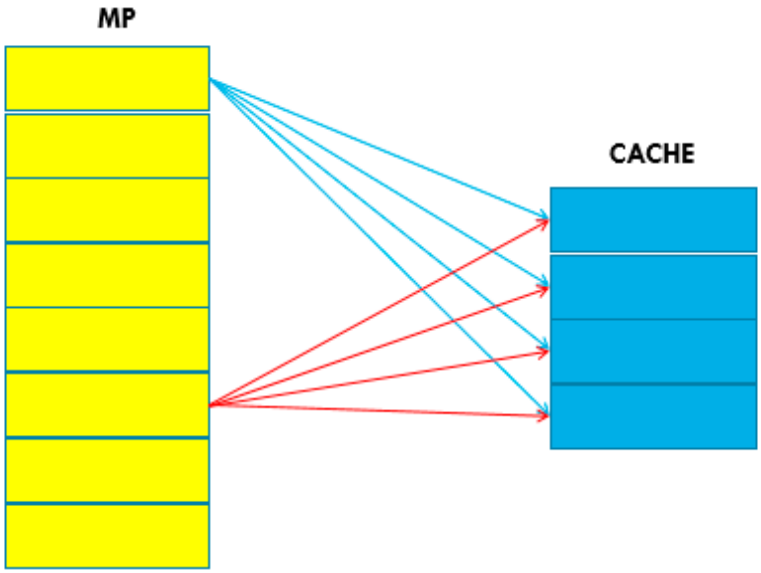
FIGURA 3 | **Mapeamento Direto**



Fonte: https://www.researchgate.net/figure/Figura-39-Diagrama-de-uma-memoria-cache-com-mapeamento-direto-apresentando-a-tag-e-o_fig5_256293104.

Mapeamento (totalmente) associativo – Neste mapeamento, um bloco de memória pode estar em qualquer quadro da cache. Neste caso, sempre que precisar encontrar algo, o processador deverá varrer toda a cache até encontrar (*se* encontrar) o que procura.

FIGURA 4 | **Mapeamento Totalmente Associativo**

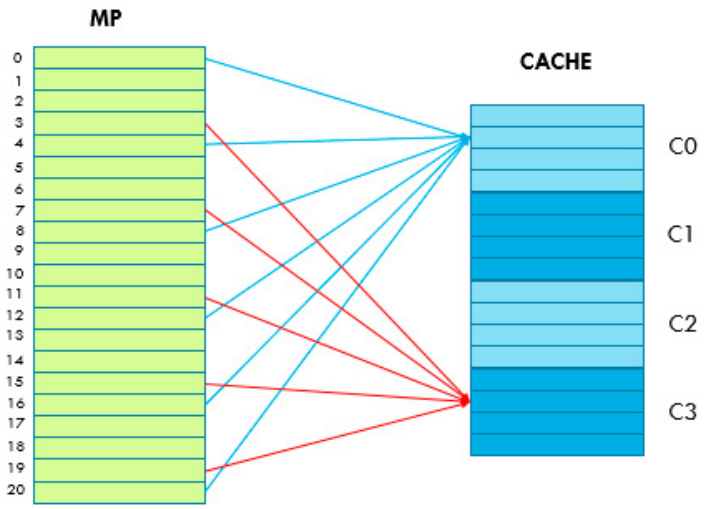


Fonte: elaborado pelo autor.

Na Figura 4 podemos ver que um bloco da memória principal pode estar em qualquer quadro da cache.

Mapeamento associativo por conjuntos – Este mapeamento é um híbrido entre os mapeamentos direto e totalmente associativo. Nele, os quadros da cache são divididos em conjuntos. O bloco de memória usará então o quadro conforme o resto da divisão entre o endereço da memória principal e a quantidade de conjuntos da cache (como no mapeamento direto). Dentro do conjunto, o bloco da memória principal poderá estar em qualquer um dos quadros (como no mapeamento totalmente associativo).

FIGURA 5 | **Mapeamento Associativo por Conjuntos**



Fonte: elaborado pelo autor.

Na Figura 5 podemos ver que os blocos da memória principal cujo resto da divisão por quatro (quantidade de conjuntos da cache) é igual a 0, podem ser alocados em qualquer quadro da cache do conjunto 0 (setas azuis) e os blocos da memória principal cujo resto da divisão por quatro é igual a 3 podem ser alocados em qualquer quadro do conjunto 3.

CONSIDERAÇÕES FINAIS

Vimos nesta aula um pouco mais sobre o subsistema de armazenamento do computador, entrando em mais detalhes sobre o relacionamento entre as memórias principal e cache.

A cache, que é uma memória mais rápida do que a memória principal, guarda o que é mais provável do processador usar. E isso é possível graças ao princípio da localidade, subdividido em localidade espacial e localidade temporal.

Vimos também que, para que a memória cache possa realmente melhorar a performance do computador, é importante que os *hits* de cache sejam superiores aos *misses* de cache, caso contrário, o *miss penalty* acabaria prejudicando em vez de ajudar o computador.

Finalmente, vimos que o correto e eficiente funcionamento da cache passa por políticas de substituição de quadros e mapeamento entre blocos de memória principal e quadros da cache.

Vamos falar um pouco sobre processadores? Hora de irmos para nossa próxima aula.

Até lá!

MATERIAIS COMPLEMENTARES

https://www.youtube.com/watch?v=qTvsyk-phEo – Neste vídeo você poderá aprofundar seus conhecimentos sobre mapeamento de memória cache.

https://www.youtube.com/watch?v=vVK6ffd9Aw4 – Conheça mais sobre os algoritmos de substituição de quadros da cache.

REFERÊNCIAS

STALLINGS, William. *Arquitetura e Organização de Computadores: projeto para o desempenho*. 8ª edição. Editora Pearson. Livro. (642 p.). ISBN 9788576055648. Disponível em: <https://middleware-bv.am4.com.br/SSO/iesb/9788576055648>. Acesso em: 16 out. 2022.

TANENBAUM, Andrew S. *Sistemas Operacionais Modernos*. 3ª edição. Editora Pearson. Livro. (674 p.). ISBN 9788576052371. Disponível em: <https://middleware-bv.am4.com.br/SSO/iesb/9788576052371>. Acesso em: 16 out. 2022.

TANENBAUM, Andrew S. *Organização estruturada de computadores*. 6ª edição. Editora Pearson. Livro. (628 p.). ISBN 9788581435398. Disponível em: <https://middleware-bv.am4.com.br/SSO/iesb/9788581435398>. Acesso em: 16 out. 2022.