BCC266 - Organização de Computadores

Aula 05: Memória Cache

Pedro Silva

Universidade Federal de Ouro Preto, UFOP
Departamento de Computação, DECOM
Email: **silvap@ufop.edu.br**

Conteúdo

Hierarquia de memória

Hierarquia de memória

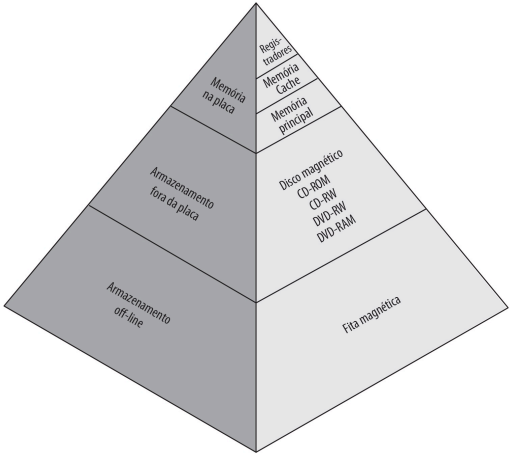
- Todas as memórias são iguais?
- O que varia entre os tipos de memórias?
  - Capacidade de armazenamento.
  - Custo (preço por bit).
  - Velocidade de acesso.
- Qual a memória ideal? barata, grande capacidade e acesso rápido.
- *Tradeoff*:
  - Memória rápida $\Rightarrow$ cara e de pequena capacidade.
  - Memória de grande capacidade $\Rightarrow$ mais barata e baixa velocidade de acesso.

Hierarquia de memória

Há diferentes tipos de memória em um computador:

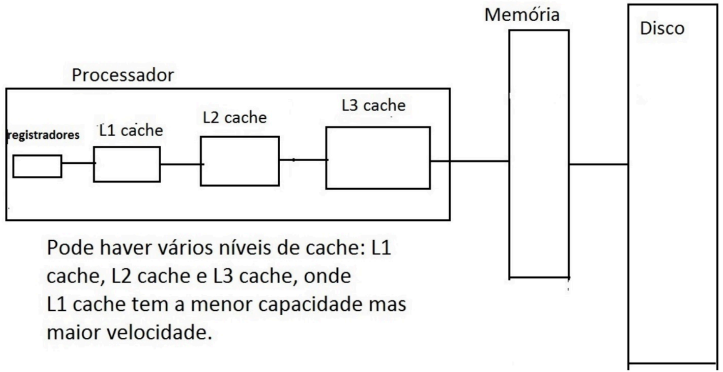
- Registradores (mais rápidas dentro do computador).
- Memórias Cache (L1, L2, etc).
- Memória interna ou principal (RAM).
- Memórias externas ou secundárias (discos, fitas).
- Outros armazenamentos remotos (arquivos distribuídos, servidores web).

Hierarquia de memória



Memória Cache

Organização das memórias



Pode haver vários níveis de cache: L1 cache, L2 cache e L3 cache, onde L1 cache tem a menor capacidade mas maior velocidade.

Memória Cache

- O mesmo dado (ou instrução) em disco pode estar copiado na memória e no processador (registrador ou memória cache).
- Se o processador precisar de um dado e ele estiver na cache, tem-se um *cache hit*.
- Se o processador precisar de um dado e ele **não** estiver na cache, tem-se um *cache miss*. Nesse caso busca-se o dado na RAM (ou até mesmo no disco);
- Ao buscar um dado na memória, um bloco com várias **palavras** é salvo na cache.

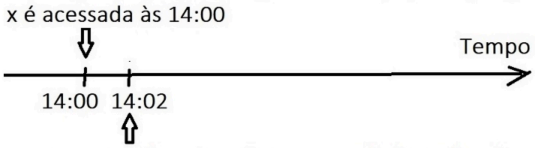
Memória Cache

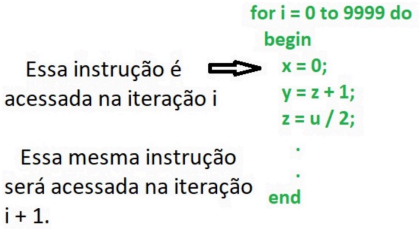
- Há estratégias para otimizar o uso da memória cache: ***prefetching***.
  - Blocos vizinhos são trazidos juntos para a cache para um possível uso futuro.
  - Fenômeno de localidade: dados nas vizinhanças de uma palavra referenciada provavelmente serão referenciados no futuro próximo.
- Nas próximas instruções, o dado (ou algum vizinho) referenciado já pode estar na cache, a qual tem um acesso muito mais rápido.
- Se a instrução ou dado não estiver no processador (registradores ou memória cache), deve-se buscar na memória e atualiza-se a RAM (acesso lento).

Fenômeno de localidade

Localidade temporal

Um dado ou instrução acessado recentemente tem maior probabilidade de ser acessado novamente, do que um dado ou instrução acessado há mais tempo.

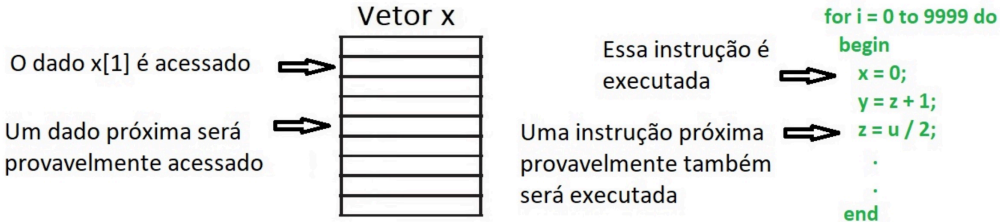




Fenômeno de localidade

Localidade espacial

Se um dado ou instrução é acessado recentemente, há uma probabilidade grande de acesso a dados ou instruções próximos.



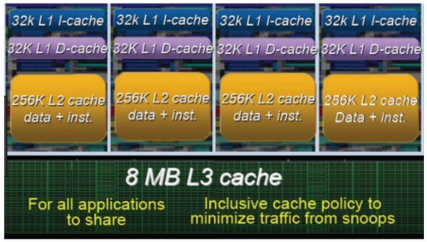
Analogia memória cache

Images source: Wikimedia Commons

Analogia

Se falta ovo (dado) na cozinha (processador), vai ao supermercado (memória) e compra uma dúzia (*prefetching*), deixando na geladeira (cache) para próximo uso.

Memória cache no Intel core i7



Intel core i7 cache (L3 cache também conhecida como LL ou Last Level cache)

Hierarquia memória	Latência em ciclos
registrador	1
L1 cache	4
L2 cache	11
L3 cache	39
Memória RAM	107
Memória virtual (disco)	milhões

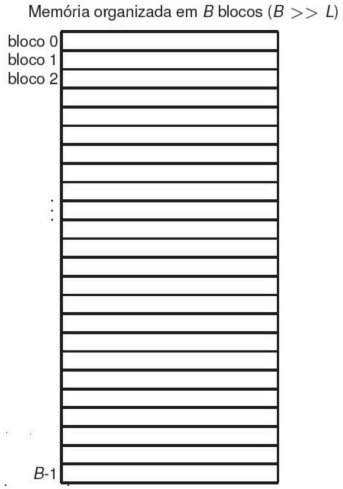
Estratégias da memória cache

Como salvar os dados na cache?

- Duplicar um dado da memória cache L1 também na cache L2.
- Caches L1 e L2 não compartilham o mesmo dado.
- Se não está na Cache L1 e nem na Cache L2, busca-se na L3, se não estiver, busca-se na memória principal.

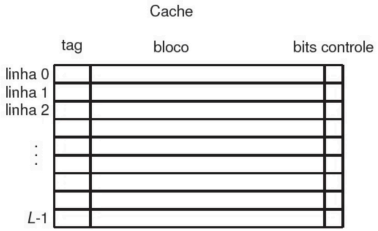
Memória Principal *vs.* Memória Cache

Organização Memória Principal



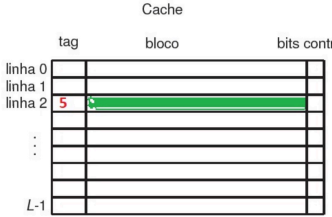
- Memória principal organizada em B blocos.
- Os blocos são numerados de 0 até *B - 1*.

Organização Memória Cache

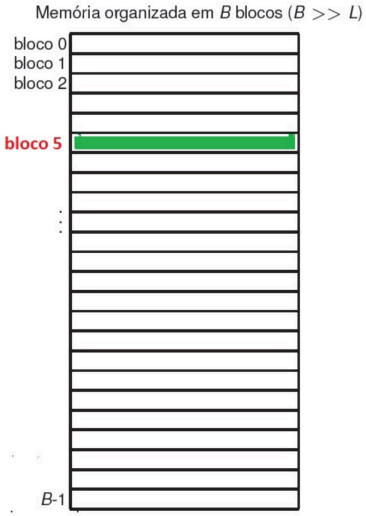


- Cache é organizada em L linhas.
- As linhas são numerados de 0 até *L - 1*.
- Tem-se um número de blocos muito maior do que linhas (B » L).
- Mas o que tem em uma linha?
  - *Tag*: endereço do bloco na RAM.
  - Bloco de memória: bloco de informações com palavras.
  - bits de controle: bits que podem auxiliar na manutenção dos blocos.

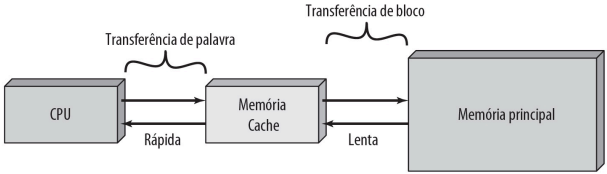
Exemplo da relação da memória principal e cache



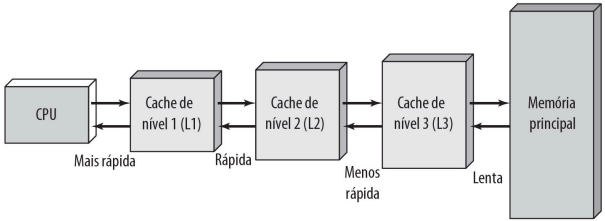
Se bloco 5 da memória foi colocado na linha 2 da cache, então *tag* contém 5.



Organização do processador, cache e RAM



(a) Cache única



(b) Organização de cache em três níveis

Memória Cache

- Em computadores multi-cores, as caches L1 e L2 podem ser específicas de cada core e a L3 compartilhada.
- Capacidade da memória cache é muito menor do que a da memória principal (se a memória é virtual, então parte dela fica no disco).
- Somente uma fração da memória principal está na cache.
- O primeiro lugar que o processador procura uma palavra da memória é na cache.
  - Se achar o bloco com a palavra na cache (*cache hit*), não precisa acessar a memória principal.
  - Se **não** achar o bloco com a plavra na cache (*cache miss*), lê-se o bloco da memória principal e o coloca na cache.
- O *hit ratio* relação entre o *cache hit* e o número de buscas na cache.

Mapeamento da memória cache

Mapeamento da memória Cache

- A memória cache tem muito menos linhas comparado ao número de blocos da memória principal.
- É necessário que linhas saiam para que outras entrem.
- A escolha de quem sai e entra define a organização da cache.
- Há três principais funções de mapeamento:
  - Mapeamento direto.
  - Mapeamento associativo.
  - Mapeamento associativo por conjunto.

Analogia para o mapeamento direto

Buscar um livro é de forma direta (você sabe onde está).

Analogia para a função de mapeamento

- Biblioteca com um acervo de 10.000 de livros.
- Livros consultados recentemente tem uma chance maior de ser buscado de novo.
- Há um escaninho de 100 posições: cada vez que um livro é buscado, um exemplar é deixado no escaninho.
- Pode faltar espaço no escaninho: remover um livro para dar espaço.
- Não precisa ir as estantes se o livro desejado já está no escaninho.
- Mas como organizar?

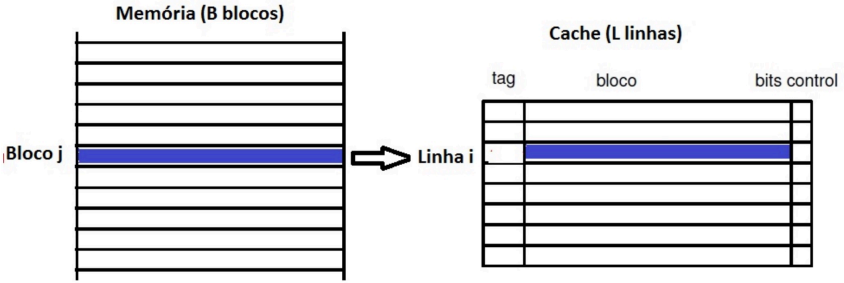
Mapeamento direto

- Exemplo com numeração binária, mas o PC utiliza numeração binária.
- Livros identificados entre 0000 e 9999.
- As posições do escaninho são identificadas de 00 a 99.
- Como mapear?

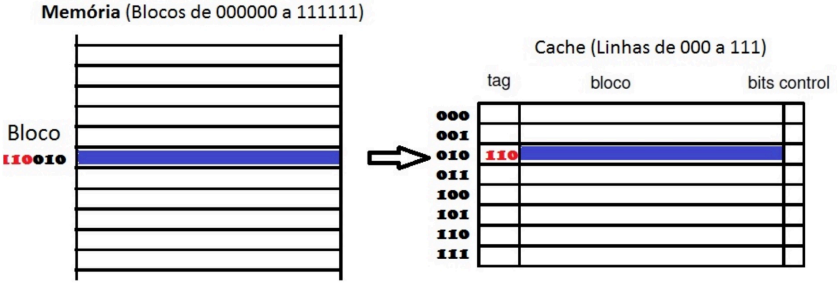
Mapeamento direto

- Buscar o livro 2513: já está no escaninho, não precisa ir a estante.
- Buscar o livro 5510: não está no escaninho, buscar na estante e deixar um exemplar no escaninho.
- Buscar o livro 8813: a posição no escaninho está ocupada, mas não é o livro, buscar na estante e substituir o exemplar.

Mapeamento direto



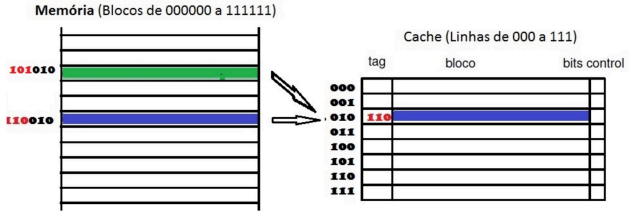
Mapeamento direto



Mapeamento direto

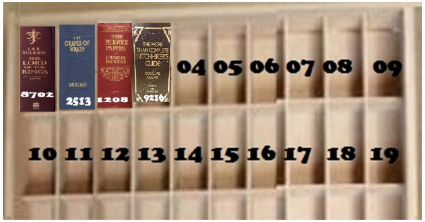
- Caso geral: $i = j \ mod \ L$ ($iinL$ e $j \in B$).
- Se B e L são potências de 2:
  - $B = 2^S$ e $L = 2^r$ ($s = 6$ e $r = 3$).
  - Basta pegar os três últimos bits ($s - r = 6 - 3 = 3$).

Análise mapeamento direto



- É simples.
- Desvantagem: se acessa repetida e alternadamente dois blocos mapeados para à mesma posição, então eles serão continuamente introduzidos e retirados da cache, mesmo que sejam muito utilizados.
- Fenômeno de *Thrashing*: alto *hit miss* e baixo *hit ratio*.

Mapeamento associativo

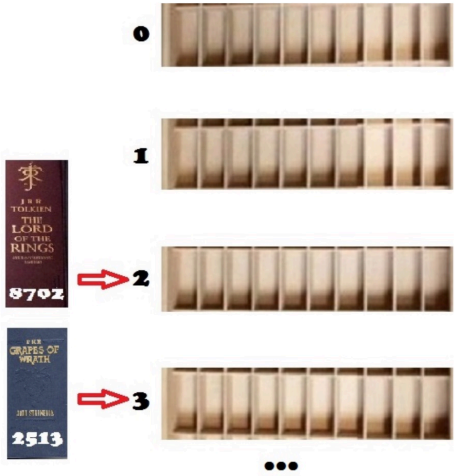


- Pode se colocar em qualquer lugar.
- Para buscar um livro, deve-se analisar todo o escaninho.
- Ficaria lento? Solução: busca em paralelo (busca associativa).
- Semelhante ao mapeamento direto: se o livro desejado não está no escaninho, então busca-se na estante e deixa um exemplar no escaninho. Não tem espaço, remove-se um livro.

Mapeamento associativo

- Um bloco de memória pode estar em qualquer linha da cache.
- Usa-se o campo *tag* de uma linha para informar qual o bloco está carregado.
- Para verificar se um bloco já está na cache, analisa-se os campos *tag* de todas as linhas simultaneamente.
- Desvantagem é a complexidade para a comparação de todos os *tags* em paralelo.

Mapeamento associativo por conjunto



- Consiste em dividir o escaninho em mais escaninhos (mapeamento direto).
- Dentro de cada escaninho, o livro pode ser colocado em qualquer lugar (mapeamento associativo).
- Combina ambos os mapeamentos: uni-se as vantagens.

Mapeamento associativo por conjunto

- Compromisso entre o mapeamento direto e associativo (une vantagens de ambos).
- Mais usado em processadores modernos.
- O mais comum é L = 4 ou 8 (*4-way* ou *8-way set-associative cache*).
- Exemplo de *2-way set-associative cache*:

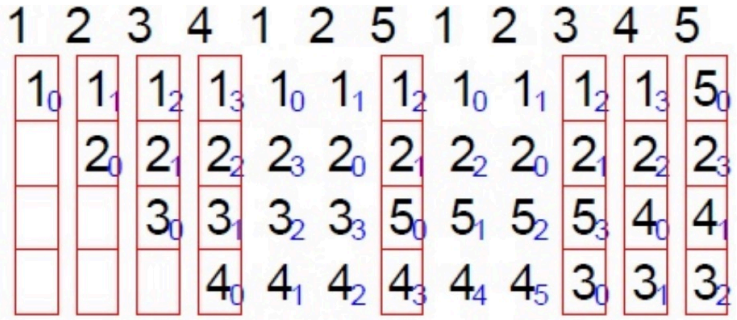
Algoritmos de substituição na cache

- Algoritmo que determina qual linha deve receber o novo bloco.
- No mapeamento direto não há escolha: há apenas uma linha possível para receber o novo bloco.
- No mapeamento associativo e associativo por conjunto há escolha e usa-se um algoritmo de substituição para fazer a escolha.

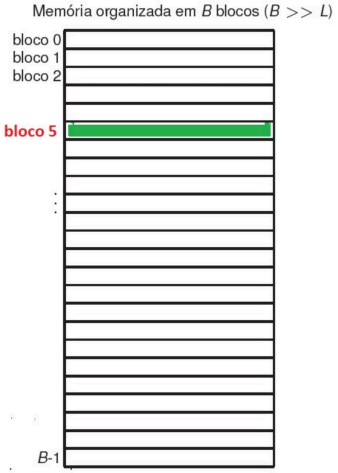
  Normalmente esse algoritmos são implementados em hardware para uma maior velocidade.
- Exemplos de algoritmos de substituição:
  - LRU (*Least Recently Used*): substitui o bloco que está na cache há mais tempo sem ser usado (um dos mais efetivos).
  - LFU (*Least Frequently Used*): substitui o bloco que menos foi usado na cache.
  - FIFO (*First-in First-out*): substitui a primeira linha a entrar na cache.

Algoritmos de substituição - LRU

- Mantêm-se uma lista de índices a todas as linhas da cache. Quando uma linha é referenciada, move-se à frente da lista.
- A linha no final da lista é substituída.
- Tem-se mostrado eficaz e com um bom *hit ratio*.



Cache *write through* e *write back*



- A cache está cheia e o alg. de substituição escolheu a linha 2 para ceder o seu lugar para outro bloco. A linha 2 pode ser alterada? Depende se a cache e RAM tem a mesma informação

Cache *write through* e *write back*

Técnica de *write through*

- Toda vez que um bloco muda de valor, o mesmo valor é escrito na RAM e na cache.
- Técnica conservadora.
- Prática segura.
- Ineficiente.
  - Mesmo que o bloco esteja na cache, é necessário acessar a memória.

Cache *write through* e *write back*

Técnica de *write back*

- Se o bloco mudar, somente a cache é alterada, a memória não é alterada.
- Gera uma inconsistência (*dirty bit* - bit de controle).
- Se a linha de cache vai ser alterada, então o conteúdo da cache é escrita na memória e a consistência é mantida.

Considerações Finais

Considerações Finais

- Hierarquia de memória.
- Memória Principal *vs.* Memória Cache.
- Métodos de mapeamento de memória cache.
- Algoritmos de substituição de cache.
- Técnicas de *write through* e *write back*.

Memórias.

Bibliografia

Bibliografia

Os conteúdos deste material, incluindo figuras, textos e códigos, foram extraídos ou adaptados do livro-texto indicado a seguir:

Stallings, William
Arquitetura e Organização de Computadores 8a Edição..
Pearson, 2010.

Baseado no material do Professor Siang Wu Song.