Memórias caches

Marcos Monteiro Junior

Princípio da localidade

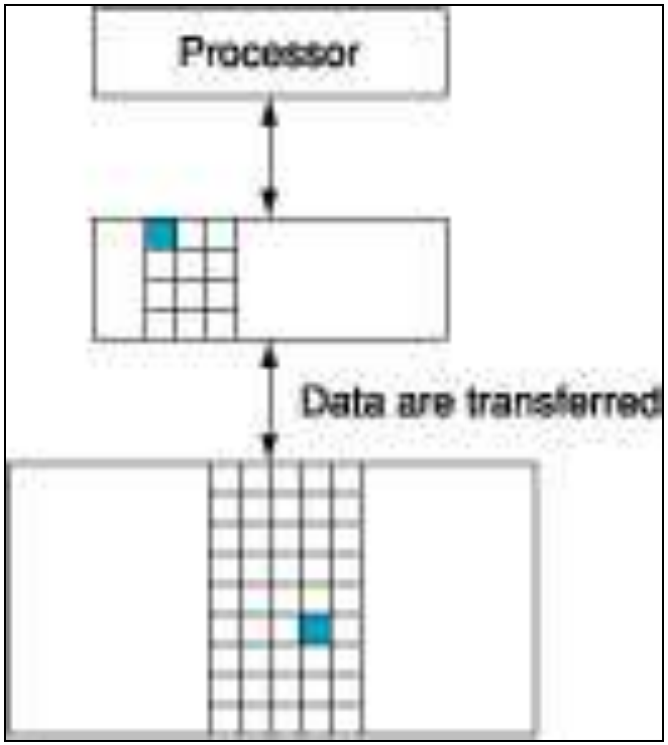
- Localidade temporal: (localidade no tempo) se um item é referenciado, ele tenderá ser referenciado novamente em breve.

- Localidade espacial: se um item é referenciado , os itens próximos deveram ser referenciados em breve.

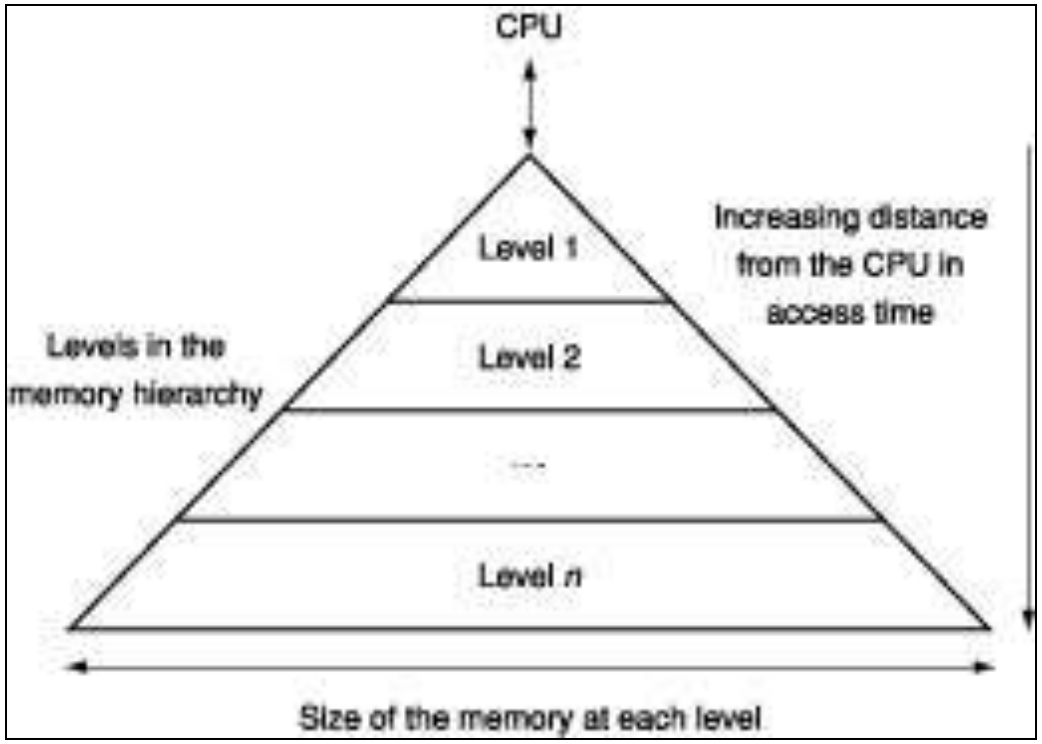
Falhas e acertos

- Bloco ou linha: é a unidade mínima de informação que pode estar presente ou ausente em uma cache.
- Se os dados requisitados pelo processador aparecem em algum bloco do nível superior, isso é chamado **acerto**
  - Se os dados não forem encontrados, isso é chamado de **falha**
  - A **taxa de acertos** é a fração dos acessos a memória encontrados no nível superior
  - A **taxa de falhas** (1 – taxa de acertos) é a proporção de acessos à memória não encontrados no nível superior
  - O **tempo de acerto** é o tempo para ter acesso o nível adjacente da hierarquia
  - A **penalidade de falha** é o tempo para substituir um bloco do nível superior pelo bloco correspondente do nível inferior
  - Tempo de acerto << tempo para acesso ao próximo nível

Falhas e acertos



Transferência de blocos entre os níveis



Hierarquia de memória

Gerencia da hierarquia de memória

- Registradores ↔ memória
  - Pelo compilador/programador
- Cache ↔ memória principal
  - Pelo controlador de cache (hardware)
- Memória principal ↔ disco
  - Pelo sistema operacional, através da memória virtual
  - O mapeamento de endereços virtuais para endereços físicos é assistido pelo hardware (TLB)
  - Pelo programador (arquivos)

Caches

- Na hierarquia de memória existe praticamente três tipos de memória: Caches, DRAM, Discos magnéticos
  - Cache, é formada por memoria SDRAM cujo o custo em 2008 é de 2000 a 5000 dólares por GB
  - DRAM tem custo de 20 a 75 por GB
  - E o disco de 0,2 a 2 por GB.

Caches

- Para facilitar cada requisição do processador é uma palavra e cada bloco da cache também é.

- Após feita um requisição pelo processador, e caso ocorra uma falha a cache busca nos níveis inferiores os dados

Caches

- O hardware deve ser implementado para responder duas questões
  - Q1: Como saber se os itens procurados estão na memória cache?
  - Q2: Se estiver, como encontrá-lo
  - A implementação da memória cache pode ser realizada de diferentes formas para responder essa pergunta
    - A forma de implementaçao afeta diretamente o desempenho
- Mapeamento direto
  - Para cada item de dados no nível mais baixo, existe apenas uma localização na cache onde ele pode estar
  - Uma quantidade de itens de dados do nível inferior deve **compartilhar** localizações no nível mais alto
  - Mapeamento de endereço

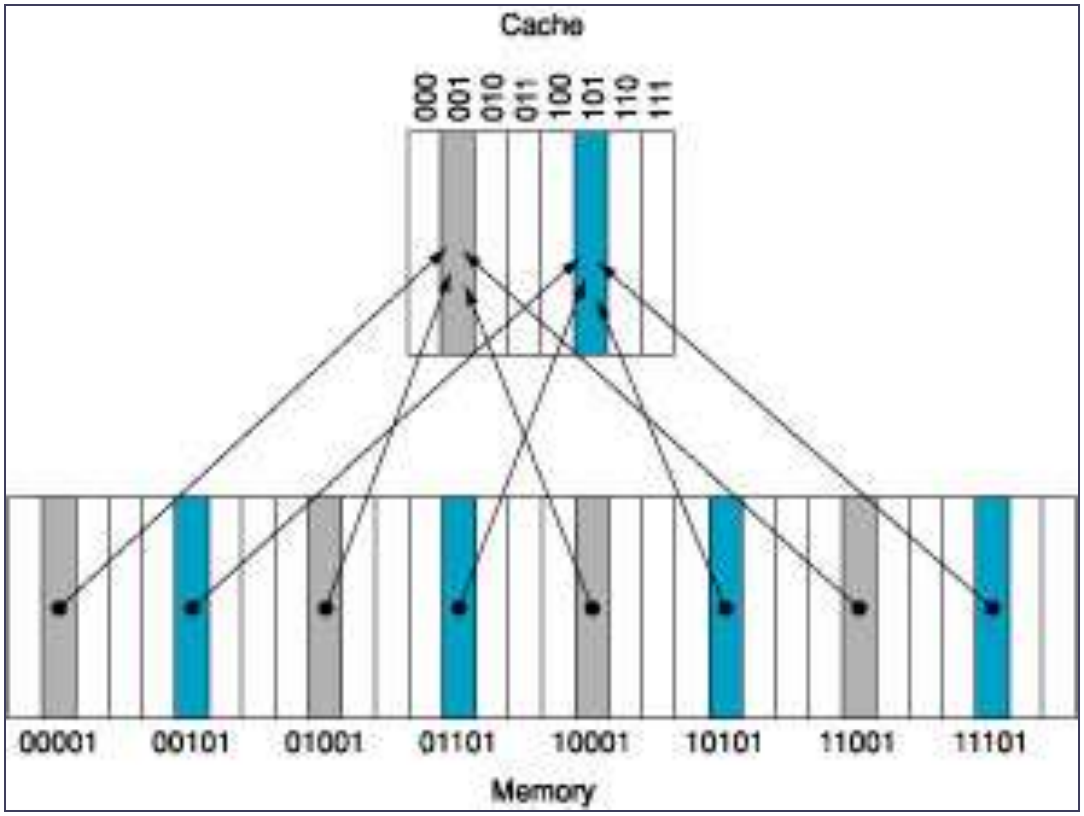
    (endereço do bloco) modulo (número de blocos na cache)

Mapeamento direto

- Se o número de entradas da cache for uma potência de dois, então o modulo pode ser calculado simplesmente usando log na base 2

- Se a memora cache possui 8 palavras, e você deseja referenciar os valores 0001 e o valor 11101, em que posição ficaria esses valores?

Mapeamento direto



Mapeamento direto

- Como cada local da cache pode armazenar o conteúdo de diversos locais de memória diferentes, como podemos saber se os dados na cache correspondem a uma word requisitada?
  - Inclusão de um conjunto de **tags**
  - **Tags** contém informações de endereço necessárias para identificar se uma *word* na cache corresponde a uma word requisitada
  - Precisamos apenas ter os dois bits mais significativos do endereço que seleciona o bloco no campo **tag**
    - exclui-se os demais bits pois eles são redundantes (ver figura)
  - Deve-se verificar também se o bloco da cache não tem informações não válidas
    - Mesmo após executar várias instruções, algumas entradas na cache podem ter informações não válidas (vazias)
    - Método mais comum é a inserção de um **bit de validade**

Índice	V	Tag	Dados
000	N		
001	N		
010	N		
011	N		
100	N		
101	N		
110	N		
111	N		

a. Estado inicial da cache após a inicialização

Índice	V	Tag	Dados
000	N		
001	N		
010	N		
011	N		
100	N		
101	N		
110	S	10_{bin}	Memória (10110_{bin})
111	N		

b. Após tratar uma falha no endereço (10110_{bin})

Índice	V	Tag	Dados
000	N		
001	N		
010	S	11_{bin}	Memória (11010_{bin})
011	N		
100	N		
101	N		
110	S	10_{bin}	Memória (10110_{bin})
111	N		

c. Após tratar uma falha no endereço (11010_{bin})

Índice	V	Tag	Dados
000	S	10_{bin}	Memória (10000_{bin})
001	N		
010	S	11_{bin}	Memória (11010_{bin})
011	N		
100	N		
101	N		
110	S	10_{bin}	Memória (10110_{bin})
111	N		

d. Após tratar uma falha no endereço (10000_{bin})

Índice	V	Tag	Dados
000	S	10_{bin}	Memória (10000_{bin})
001	N		
010	S	11_{bin}	Memória (11010_{bin})
011	S	00_{bin}	Memória (00011_{bin})
100	N		
101	N		
110	S	10_{bin}	Memória (10110_{bin})
111	N		

e. Após tratar uma falha no endereço (00011_{bin})

Índice	V	Tag	Dados
000	S	10_{bin}	Memória (10000_{bin})
001	N		
010	S	10_{bin}	Memória (10010_{bin})
011	S	00_{bin}	Memória (00011_{bin})
100	N		
101	N		
110	S	10_{bin}	Memória (10110_{bin})
111	N		

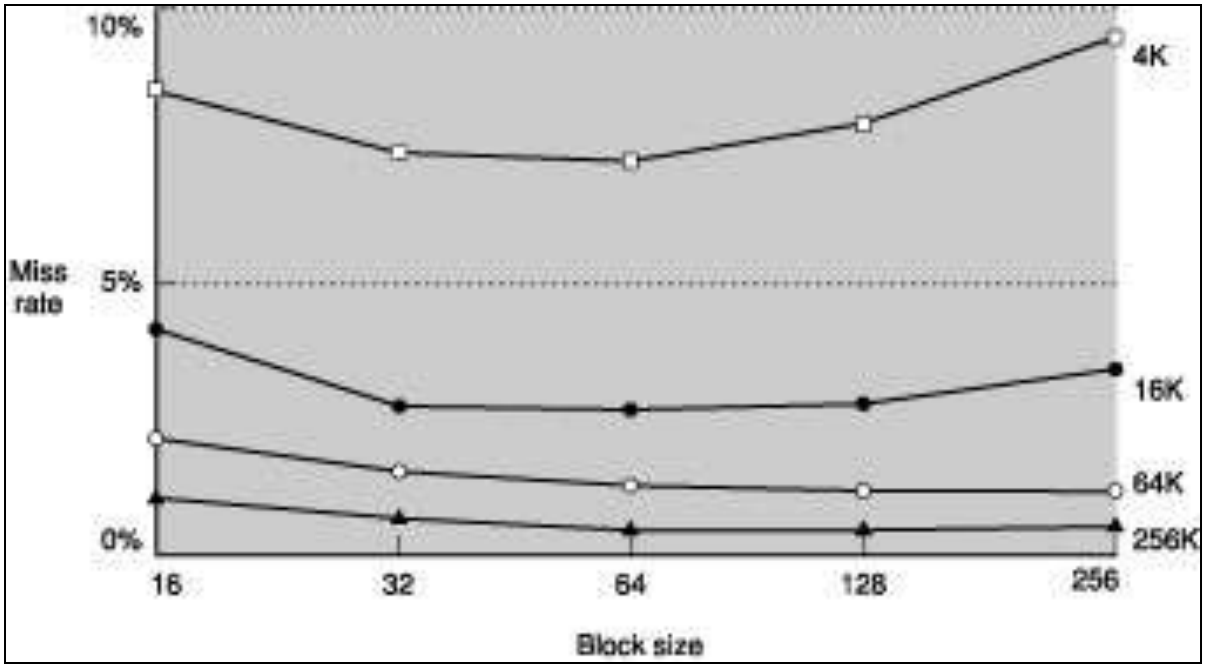
f. Após tratar uma falha no endereço (10010_{bin})

Mapeamento direto

Endereço decimal da referência	Endereço binário da referência	Acerto ou falha na cache	Bloco de cache atribuído (onde foi encontrado ou inserido)
22	10110_{bin}	falha (7.6b)	$(10110_{bin} \bmod 8) = 110_{bin}$
26	11010_{bin}	falha (7.6c)	$(11010_{bin} \bmod 8) = 010_{bin}$
22	10110_{bin}	acerto	$(10110_{bin} \bmod 8) = 110_{bin}$
26	11010_{bin}	acerto	$(11010_{bin} \bmod 8) = 010_{bin}$
16	10000_{bin}	falha (7.6d)	$(10000_{bin} \bmod 8) = 000_{bin}$
3	00011_{bin}	falha (7.6e)	$(00011_{bin} \bmod 8) = 011_{bin}$
16	10000_{bin}	acerto	$(10000_{bin} \bmod 8) = 000_{bin}$
18	10010_{bin}	falha (7.6f)	$(10010_{bin} \bmod 8) = 010_{bin}$

Cache

- Blocos maiores, vale a pena?
  - A taxa de falhas pode subir posteriormente se o tamanho de bloco se tornar uma fração significativa do tamanho da cache, uma vez que o número de blocos que pode ser armazenado na cache se tornará pequeno e haverá uma grande competição entre blocos (é claro, mantendo o tamanho total da cache)
  - O custo da falha aumenta na medida que aumenta o tamanho do bloco (maior tempo de transferência)
    - O tempo para buscar o bloco pode ser organizado em duas partes: a latência até a primeira *word* e o tempo de transferência para o restante do bloco
    - Segunda parte do tempo aumenta proporcionalmente ao tamanho do bloco



Cache

- Blocos maiores
  - A desvantagem principal de aumentar o tamanho do bloco é que a penalidade de falha aumenta
    - Solução: reinício precoce: retornar para o processador quando a palavra solicita estiver disponível, ao invés de esperar o bloco inteiro
    - Técnica de reinício precoce funciona bem para cache de instruções (execução sequencial é típica)
    - Não funciona bem para cache de dados (requisições são menos previsíveis que a na cache de instruções)
    - Técnica *word requisitada primeiro*: similar a reinício precoce, mas com acesso direto a *word* requisitada e então transfere o restante do bloco (duas partes: antes e depois da word requisitada)
      - sofre dos mesmos problemas