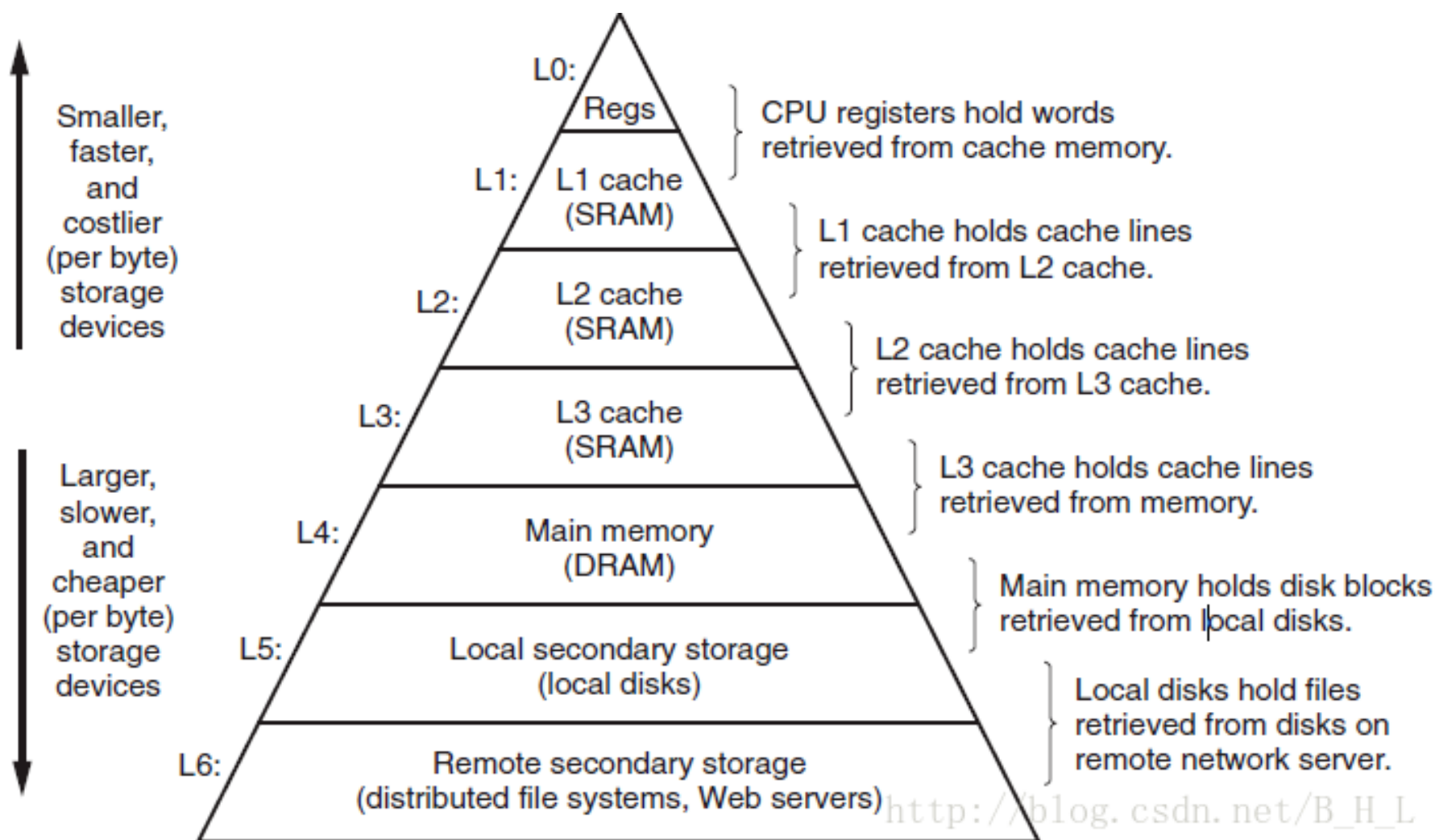


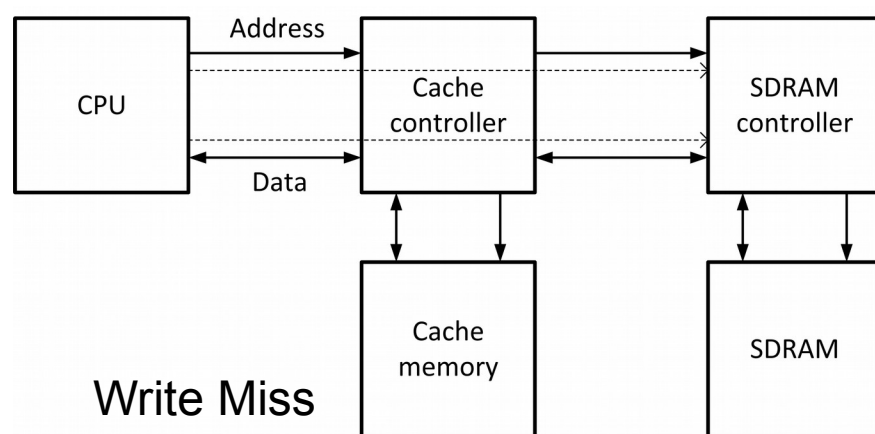
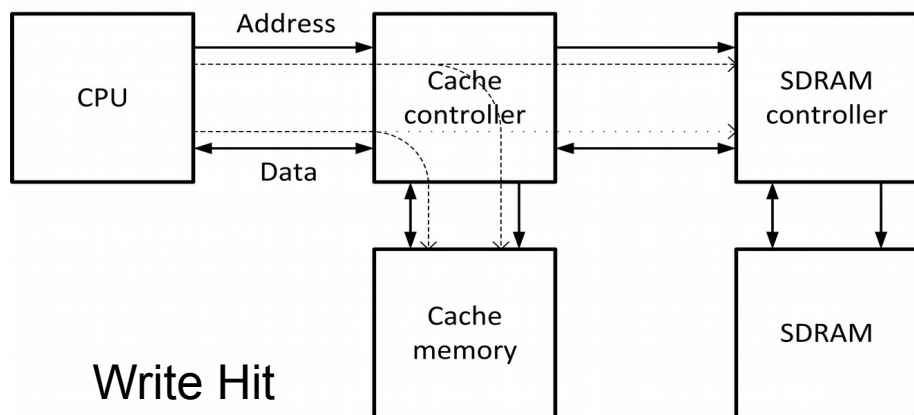
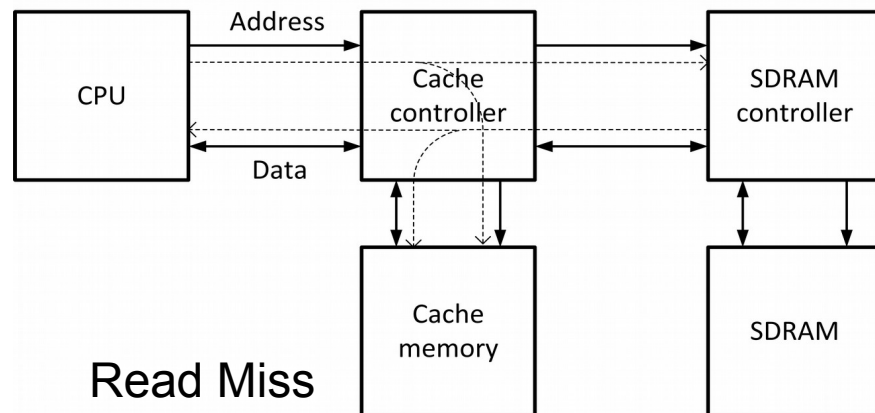
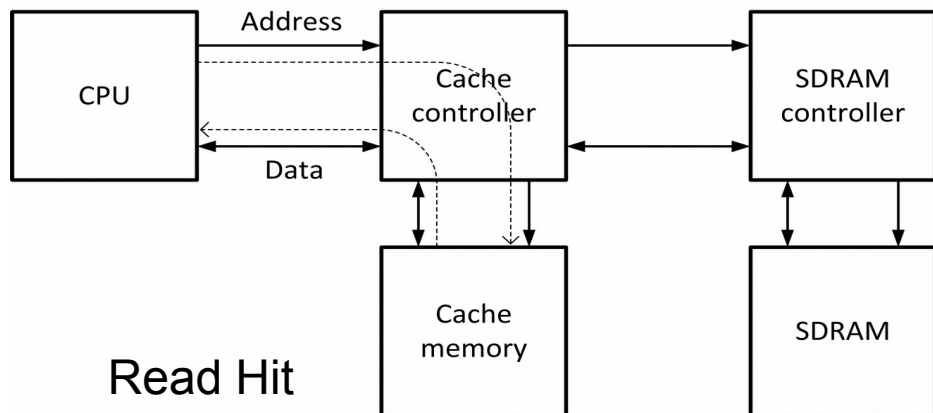
Cache

Ezequiel Conde
ezequiel.conde@isel.pt

Hierarquia de memória

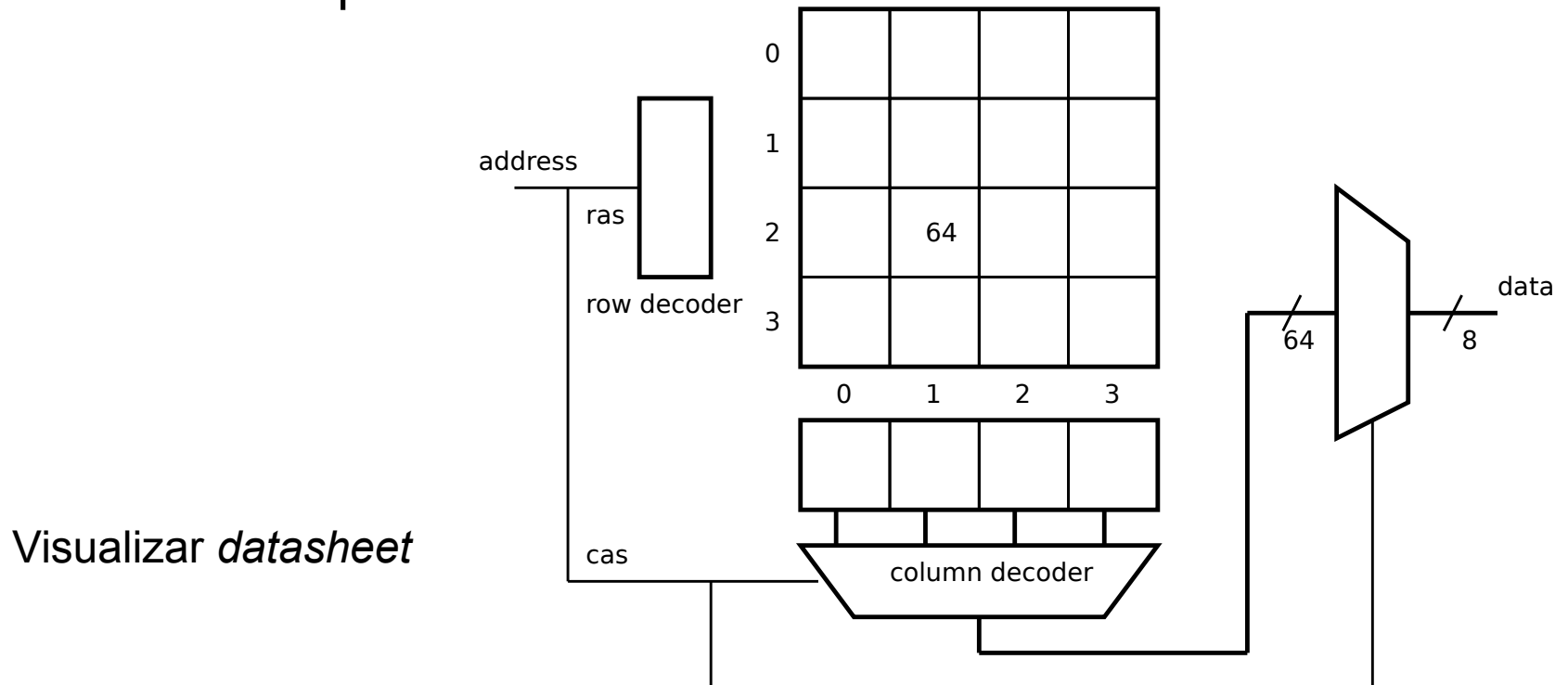


Arquitetura do sistema com *cache*

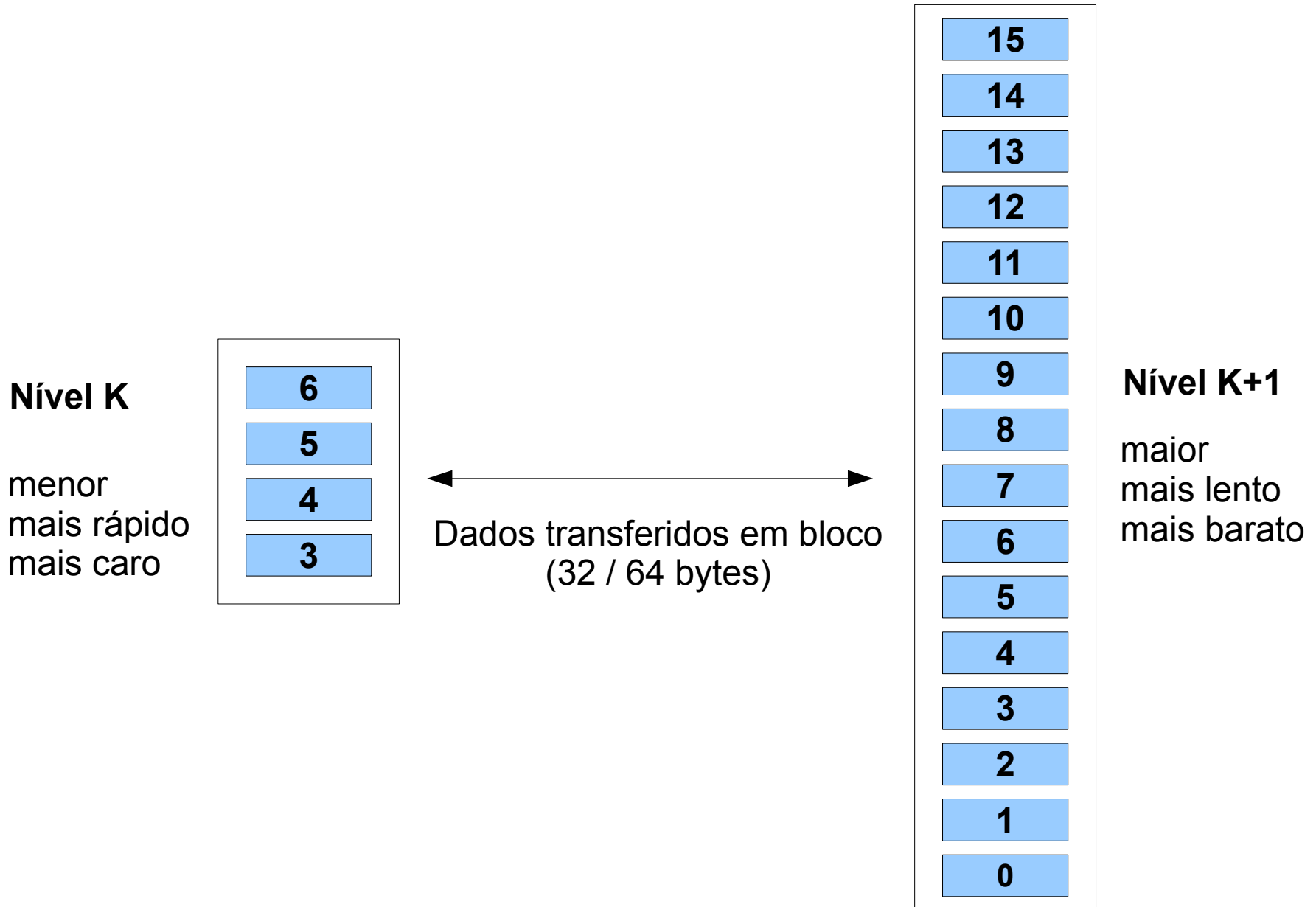


RAM dinâmica

- A DRAM é organizada em super-células (contém 32/64/128 bits).
- As super-células são organizadas em matriz.
- O constrangimento temporal mais relevante é o acesso à linha.
- Depois da linha estar acessível, o acesso a células da mesma linha é mais rápido.



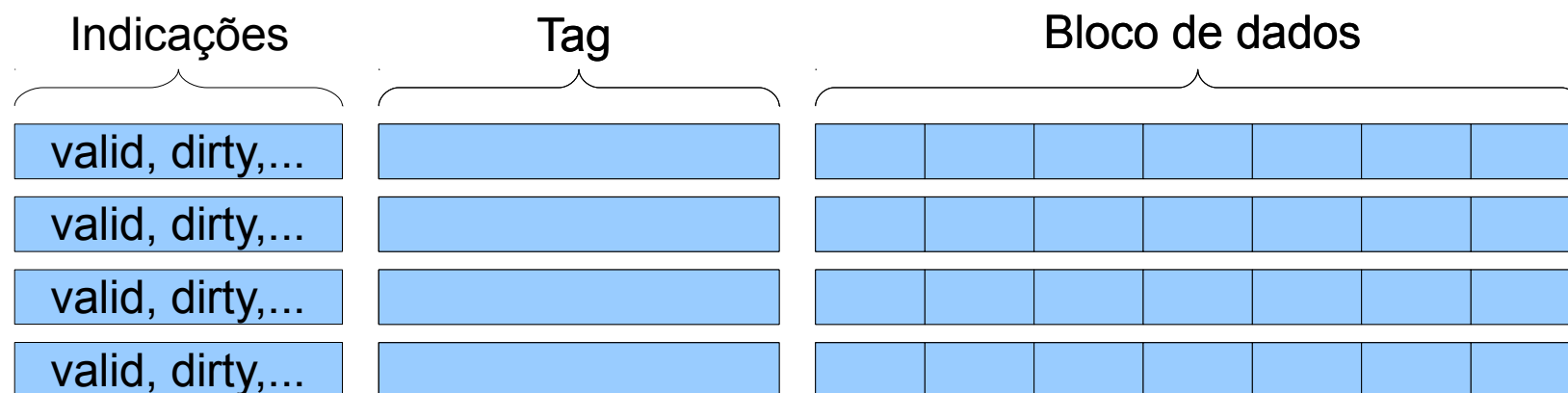
Princípio de funcionamento da cache



Organização da Cache

Como saber se os dados estão na cache?

- A cache é organizada como um conjunto de linhas.
- Uma linha contém:
 - um bloco de dados - 32 ou 64 *bytes*;
 - o endereço de memória desse bloco – *tag*;
 - indicações - *valid bit*; *dirty bit*; ...



Princípio da localidade

localidade temporal

- acesso repetido à mesma posição de memória
- código dos ciclos; variáveis muito usadas

localidade espacial

- o código executa em endereços consecutivos (até aos jump)
- variáveis localizadas proximamente

Exemplo clássico de localidade

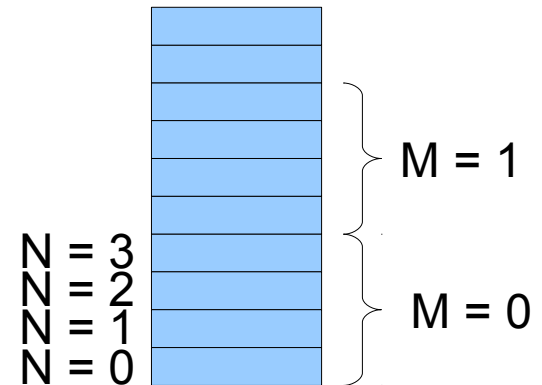
```
int array[M][N];  
  
for (int i = 0; i < M; ++i)  
    for (int j = 0; j < N; ++j)  
        x[i][j] = x[i][j] + K;
```

$x[0][0], x[0][1], x[0][2], x[0][3]$

$x[1][0], x[1][1], x[1][2], x[1][3]$

$x[2][0], x[2][1], x[2][2], x[2][3]$

$x[3][0], x[3][1], x[3][2], x[3][3]$



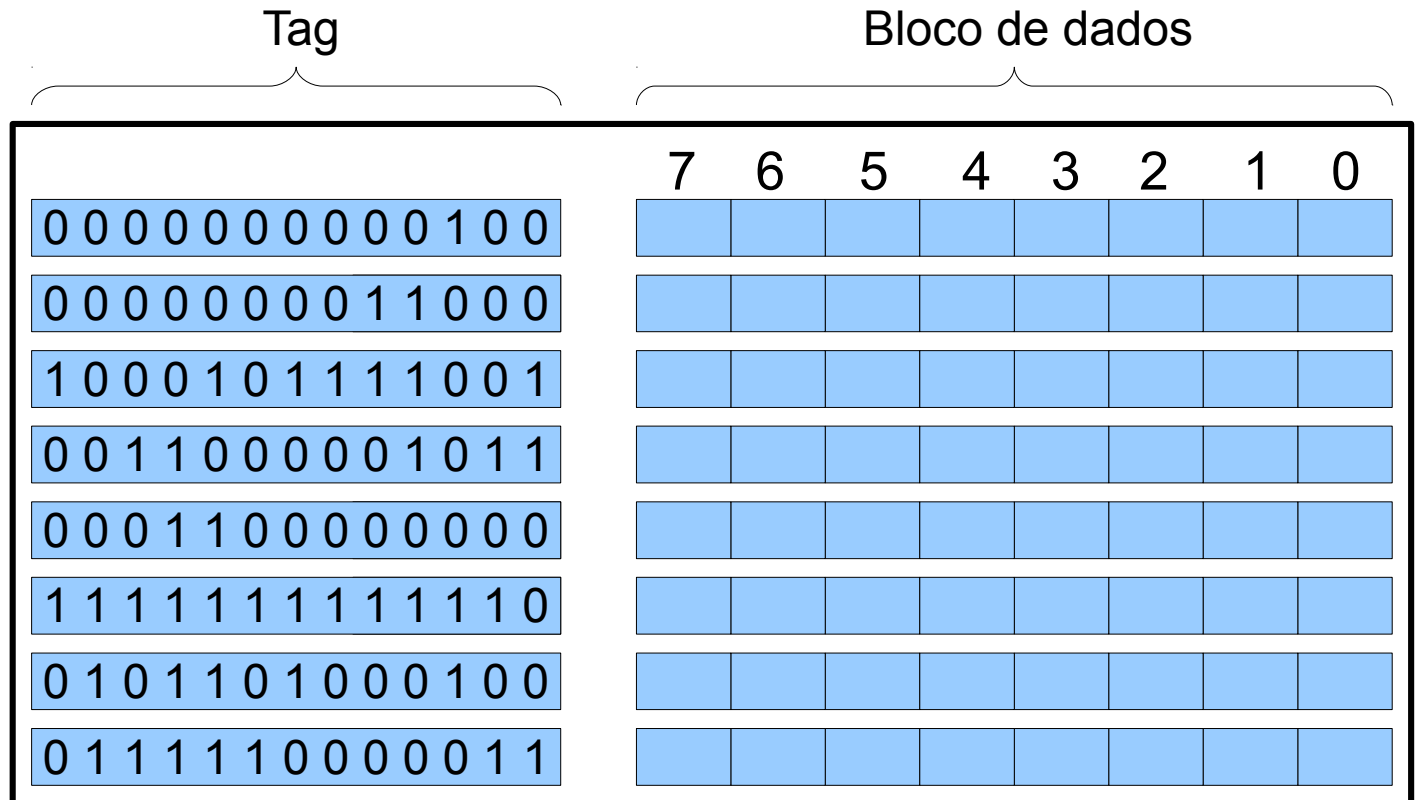
Organização da cache (2)

Cache com 64 bytes de dimensão.

Espaço de memória de CPU 64 Kbyte.

Endereço do CPU

0 0 1 1 0 0 0 0 0 1 0 1 1 0 0 1



Organização da cache (3)

Endereço do CPU

0 0 1 1 0 0 0 0 0 1 0 1 1 0 0 1

Tag

Linha

#byte

Tag

Bloco de dados

Linha

0
1
2
3
4
5
6
7

0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	1	1	
2	1	0	0	0	1	0	1	1	1	1	
3	0	0	1	1	0	0	0	0	0	1	
4	0	0	0	1	1	0	0	0	0	0	
5	1	1	1	1	1	1	1	1	1	1	
6	0	1	0	1	1	0	1	0	0	0	
7	0	1	1	1	1	1	0	0	0	0	

7 6 5 4 3 2 1 0

Outro endereço do CPU

0 0 1 1 0 0 0 0 0 0 0 1 1 0 0 0

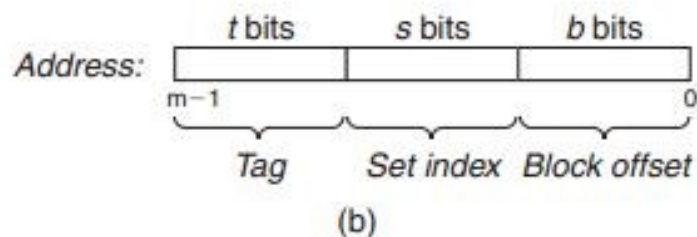
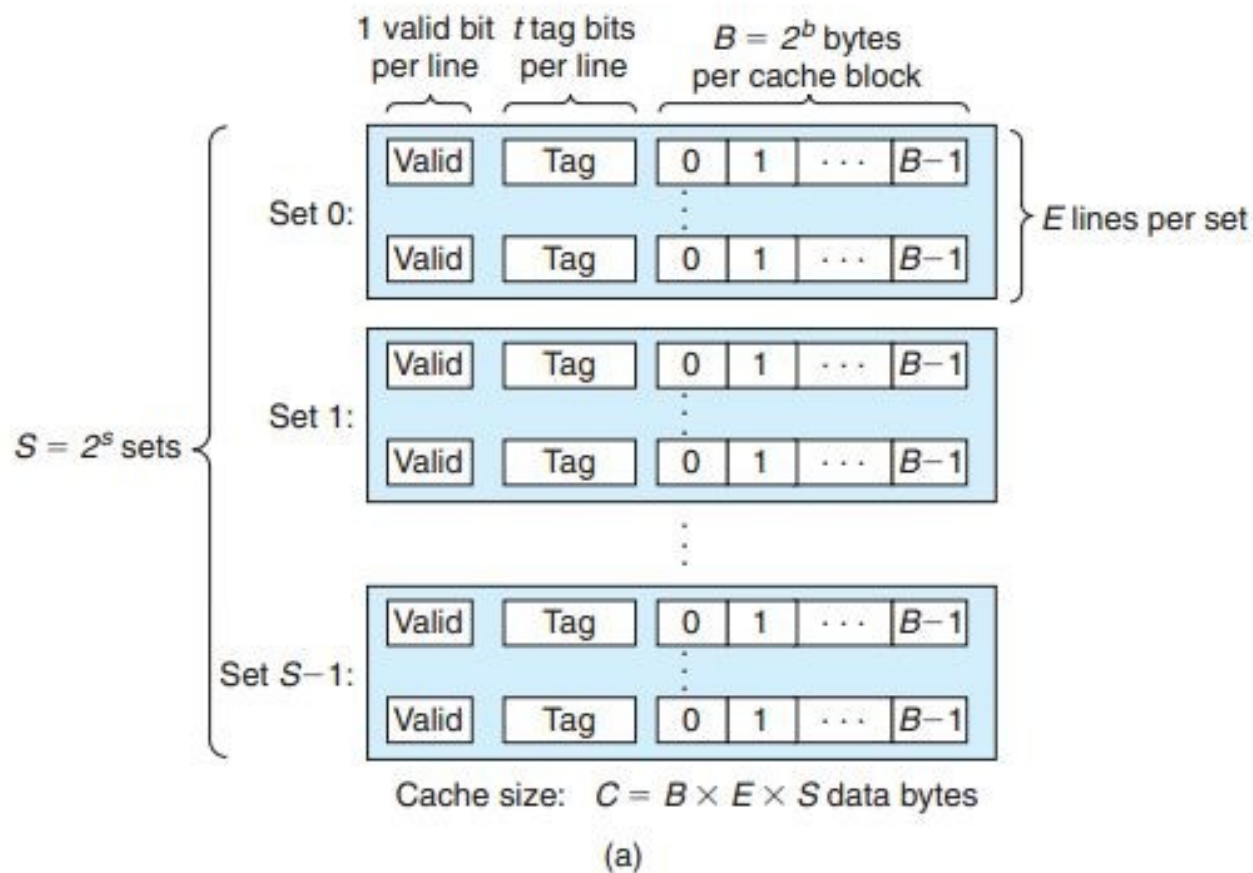
Colisão !

Organização geral da *cache*

Figure 6.27

General organization of cache (S, E, B, m).

(a) A cache is an array of sets. Each set contains one or more lines. Each line contains a valid bit, some tag bits, and a block of data. (b) The cache organization induces a partition of the m address bits into t tag bits, s set index bits, and b block offset bits.



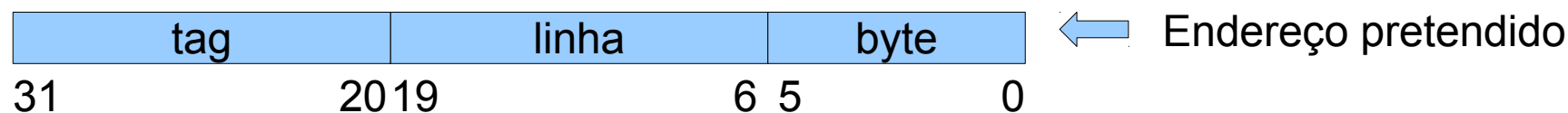
Mapeamento direto

A linha de *cache* é selecionada com uma função direta do endereço.

A parte do endereço usada para escolher a linha são os bits de menor peso a seguir ao endereço do *byte* dentro do bloco.

Considerando o caso de uma cache de 1 MByte com blocos de 64 Byte.

$$1\text{MB} / 64\text{B} = 16384 \text{ linhas } (\log_2 16384 = 14)$$



Um dado endereço só pode usar uma dada linha de *cache*.

Isso pode ser uma grande desvantagem se for necessário aceder a dois endereços que correspondam à mesma linha - *conflict* ou *collision misses*.

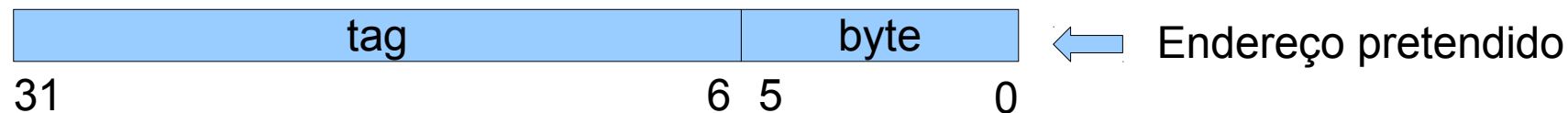
Mapeamento associativo

Qualquer linha de *cache* pode conter qualquer bloco.

É necessário pesquisar em todas as linhas por uma *tag* igual à do endereço pretendido.

Considerando o caso de uma *cache* de 1 MByte com blocos de 64 Byte:

todos os *bits* a partir do endereço do byte no bloco pertencem à *tag*.



Não há *conflict misses* mas a pesquisa é dispendiosa, em material ou em tempo.

Cache Hit / Cache Miss

Está na *cache* \longrightarrow *cache hit* (*hit rate*)

Não está na *cache* \longrightarrow *cache miss* (*miss rate*)

Cache Miss !

1. ler da memória;
2. guardar numa linha de *cache*.

Onde colocar o bloco?

- Depende da função de mapeamento;
- Se não existir lugar disponível há que substituir o bloco;
- Necessidade de política de substituição.

Políticas de substituição

Mapeamento direto não requer processamento.

Idealmente deve-se substituir a última a ser necessária no futuro próximo.

Como prever as futuras necessidades? Aplicar o princípio da localidade.

- Escolher a não usada há mais tempo (LRU)
- Escolher aleatoriamente (bons resultados para sets grandes)
- Escolher a mais antiga (FIFO, ring buffer)
- Escolher a usada com menos frequência (contador em cada linha)

Políticas de escrita

- **write-through**

A memória principal é atualizada em simultâneo com a escrita na cache.

- **write-back**

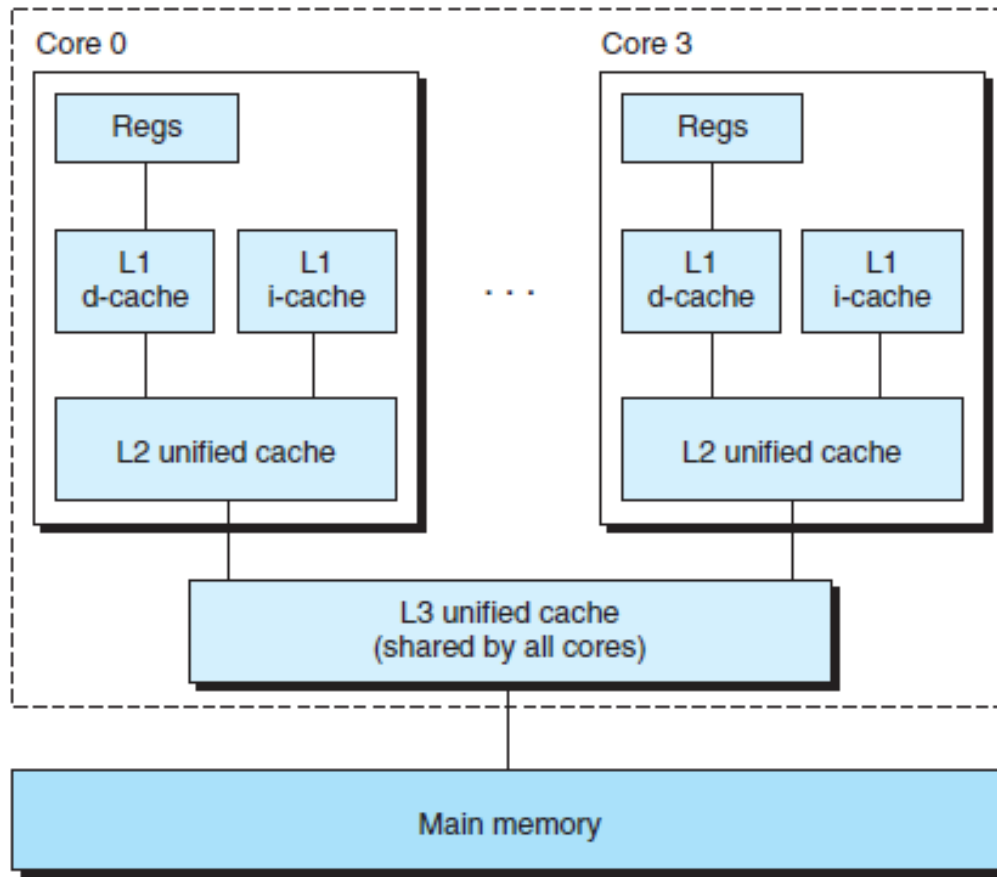
A escrita é realizada apenas na cache a atualização da memória principal é realizada posteriormente.

Quando atualizar a memória em *write-back*?

Se for na altura da substituição, torna esta operação mais demorada.

Para evitar escrever o bloco na memória principal, no caso do processador não ter realizado qualquer escrita no bloco, é acrescentado à linha o ***dirty bit***

Intel Core i7



Cache type	Access time (cycles)	Cache size (C)	Assoc. (E)	Block size (B)	Sets (S)
L1 i-cache	4	32 KB	8	64 B	64
L1 d-cache	4	32 KB	8	64 B	64
L2 unified cache	10	256 KB	8	64 B	512
L3 unified cache	40–75	8 MB	16	64 B	8,192

Referências

- Computer Systems – cap. 6 The Memory Hierarchy
- <http://www.cpu-world.com>