Arquitetura de Computadores

MIEI – 2021/22

DI-FCT/UNL

Memória Cache

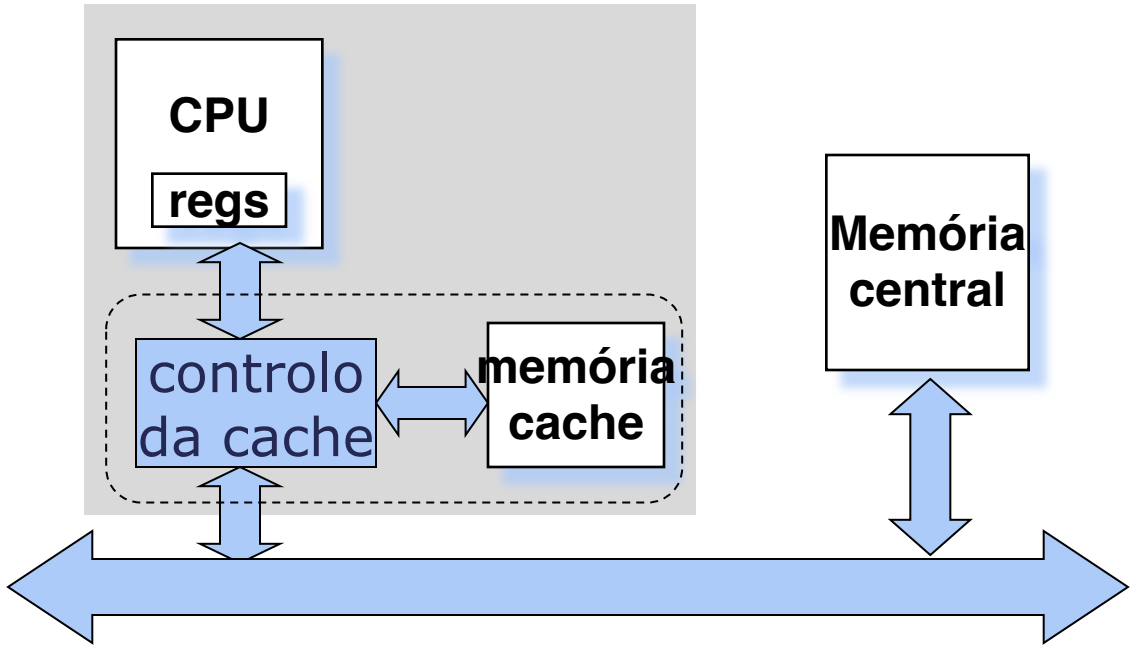
Sumário

- Memória Cache

Bibliografia:

Bryant, O'Hallaron Computer Systems: A Programmer's Perspective, 2nd ou 3rd Ed, secções 6.4

Cache de memória



- Funcionamento

CPU referencia memória:

se *está em cache*

acede à cache (mais rápido)

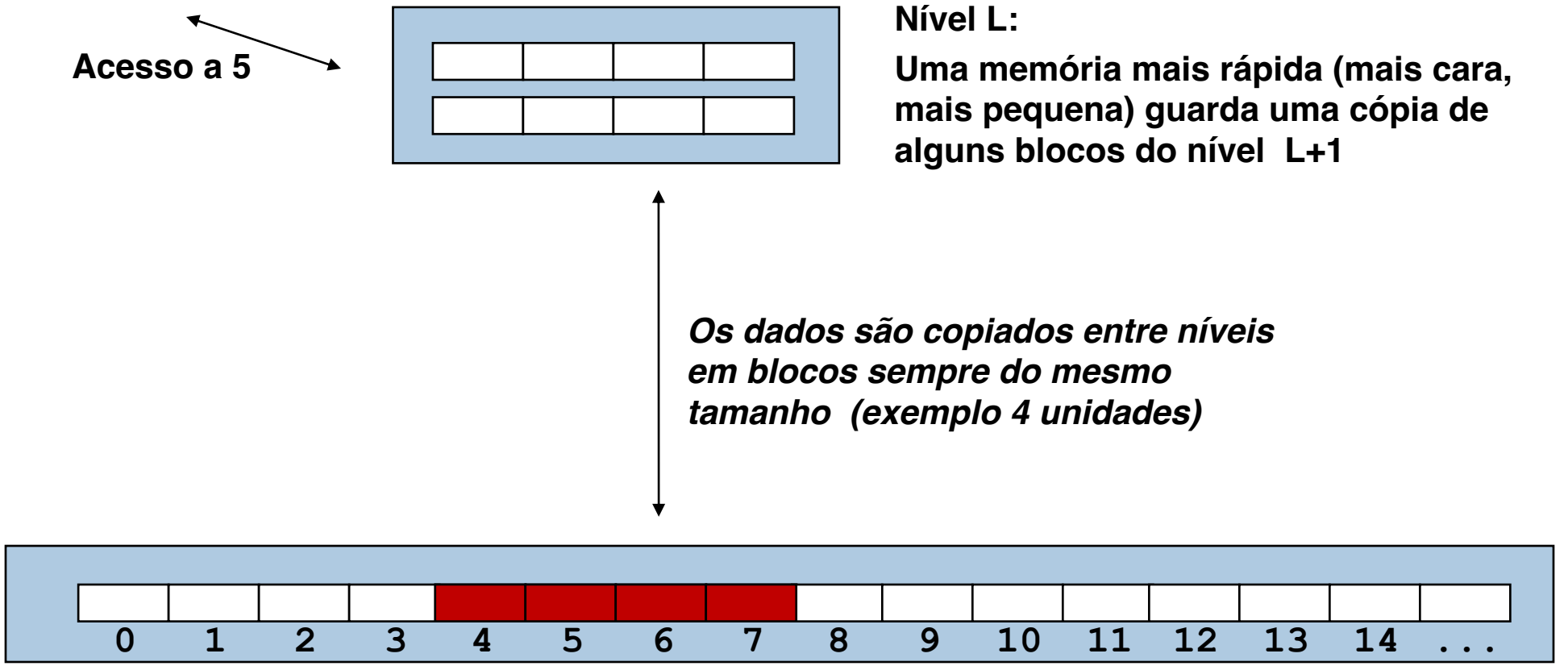
senão

acede à memória

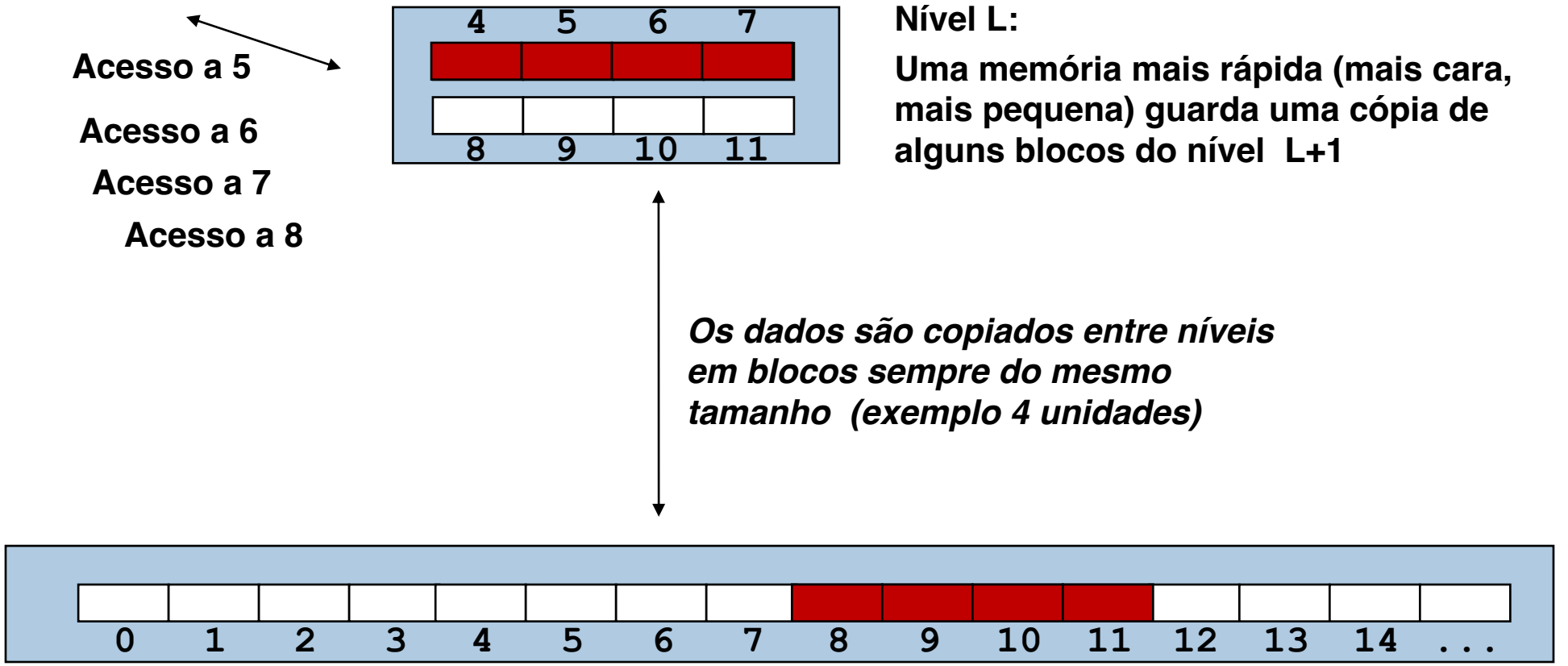
e atualiza cache

A localidade espacial leva a trazer logo um bloco da memória

Caching na hierarquia de memória



Caching na hierarquia de memória



Conceitos gerais de funcionamento

- A cache (L) tem uma cópia de uma pequena parte da memória (L+1) e respetivos endereços
- A cada acesso procura-se o endereço na cache e…
- **Cache hit**
  - Este está na cache → tempo de acesso rápido (nível L)
  - Exemplo anterior: acesso aos endereços 6 e 7
- **Cache miss**
  - Este não está na cache. Deve obtê-lo no nível inferior → tempo de acesso lento (nível L+1)
  - Exemplo anterior: primeiro acesso, ao endereço 5

Tempo médio de acesso à memória (L+1)

- Taxa de sucesso (hit ratio)
  - h = percentagem de vezes que o dado pretendido está na cache (L):

    h = núm. cache hit / núm. total acessos

- Tempo médio de acesso:
  - TL é o tempo de acesso no nível L
  - TL+1 é o tempo de acesso no nível L+1
  - Tempo médio de acesso

    $Ta = h * TL + (1 – h) * TL+1$

Exemplo

- Exemplo: um nível de cache e memória central
  - *T cache* = 2 ns
  - *T mem* = 8 ns
  - *Hit ratio* = 80% numa determinada execução

- Ta = h * T_L + $(1 - h)$ * T_{L+1}
  - Ta = 0,8 x 2 + 0,2 x 8 = **3,2 ns**

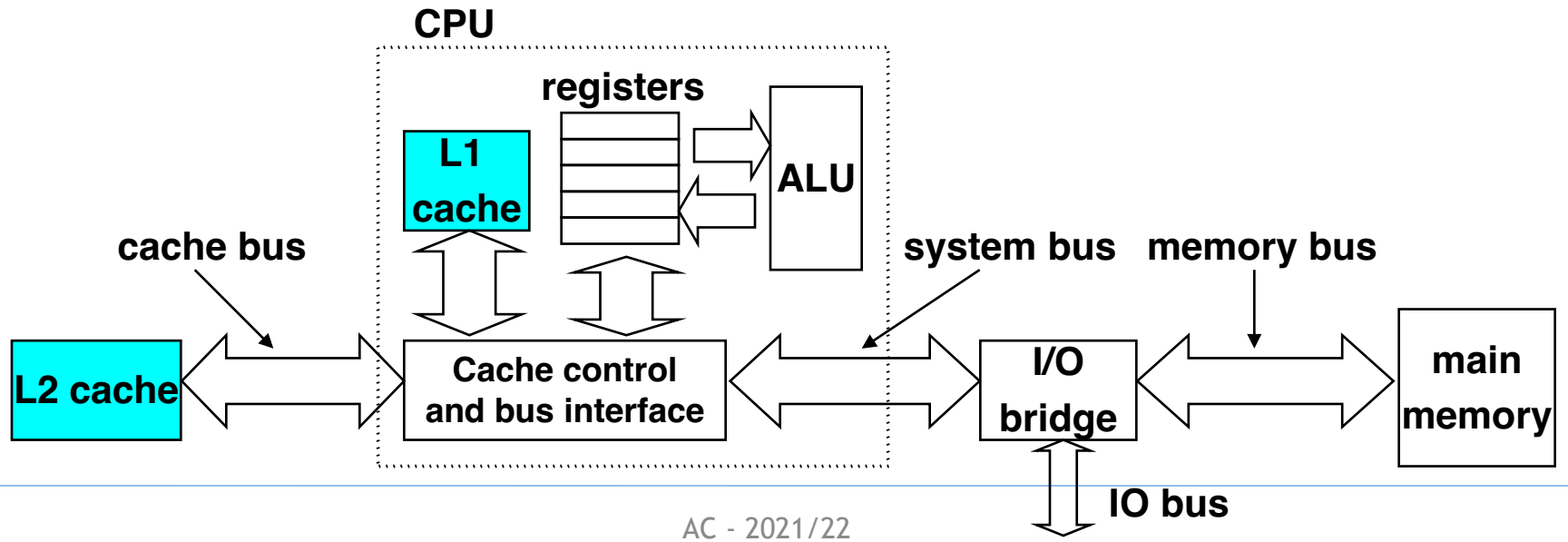
- Nesta execução a percepção do tempo de acesso à memória é de 3,2 ns

Tratamento do miss

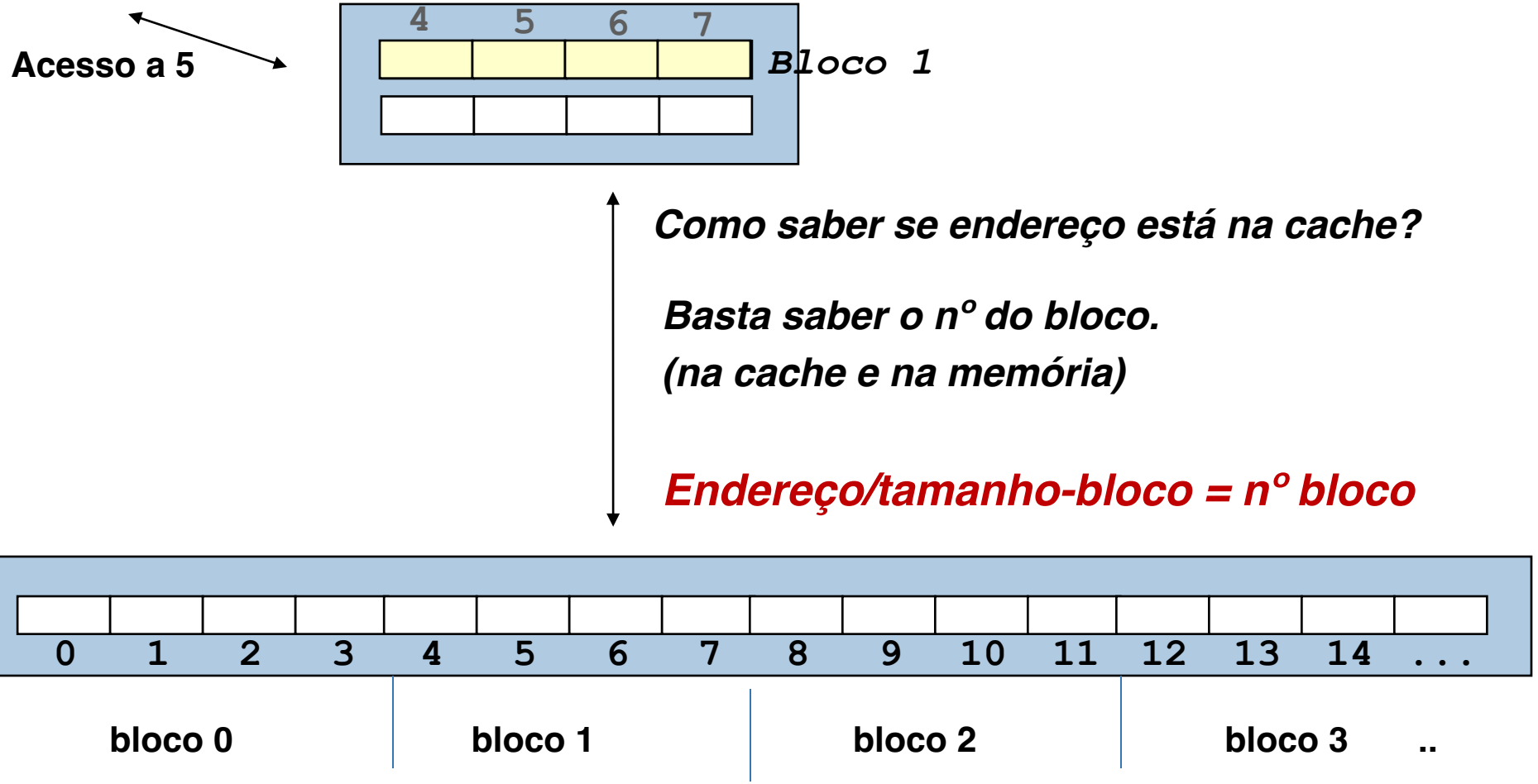
- **Cache miss**
  - Não encontra o que pretende no nível L, então vamos ao nível L+1 (e assim sucessivamente)
  - Guardar uma cópia no nível L, para futuros acessos
  - Mas se o nível L está cheio, então é preciso arranjar espaço:
    - Que bloco escolher como vítima? Ao acaso? O menos usado? …?
    - Se o bloco vítima não foi alterado, carregar o novo bloco por cima
    - Se o bloco vítima está alterado (foi escrito), é preciso garantir que no nível L+1 também está alterado antes de o substituir
    - **Vamos discutir soluções para estes problemas nas próximas aulas**

Cache – Exemplo com dois níveis

- Cache são memórias rápidas (SRAM) geridas automaticamente pelo hardware.
- Exemplo de arquitetura com L2 externa ao chip do CPU:
- CPU procura em L1, depois em L2, finalmente na memória central
  - Podem introduzir algum overhead, mas desprezável

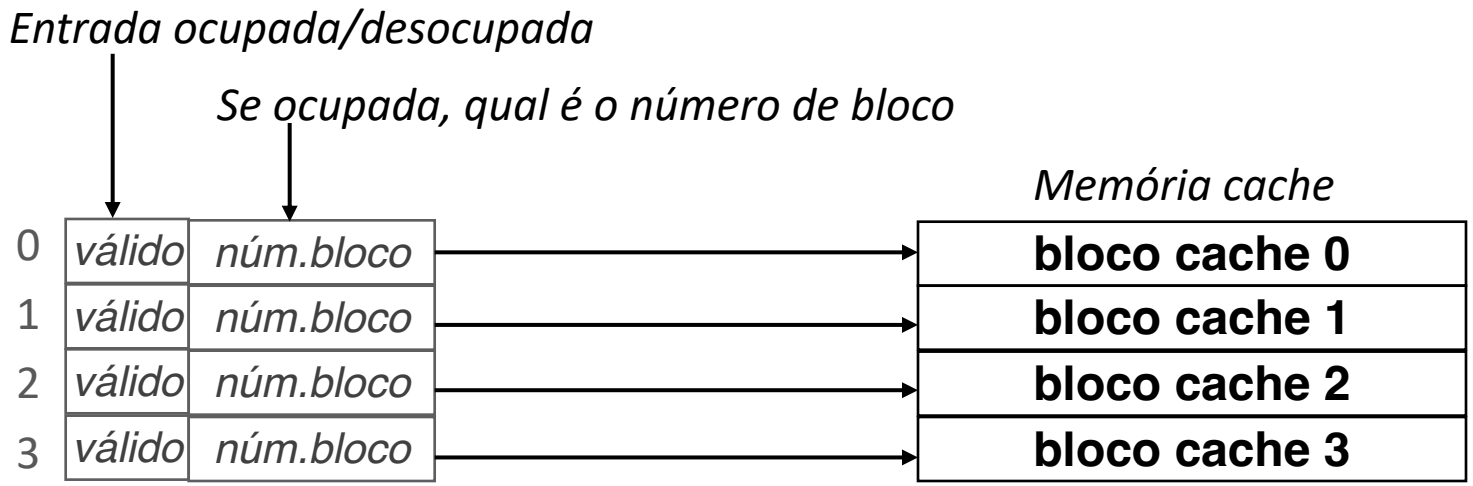


Como encontrar endereço na cache?



Gestão da Cache

- É necessário saber, de forma eficiente, que blocos da cache estão ocupados e com que blocos de memória
  - **Exemplo** de cache com capacidade para 4 blocos de memória:



Endereços e blocos de memória

Exemplo:

endereços de 8 bits

blocos de 4 bytes

dado um endereço *E*:

E/4 = número do bloco

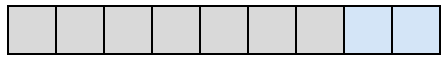
E%4 = byte dentro do bloco

ou seja:

4 bytes por bloco = 2^2 → 2 bits

os 2 bits menos significativos indicam bytes dentro do mesmo bloco

os restantes correspondem ao número do bloco



núm. bloco *deslocamento no bloco*

```
0= 00000000
1= 00000001
2= 00000010
3= 00000011
4= 00000100
5= 00000101
6= 00000110
7= 00000111
8= 00001000
9= 00001001
10= 00001010
11= 00001011
12= 00001100
13= 00001101
14= 00001110
15= 00001111
16= 00010000
17= 00010001
18= 00010010
19= 00010011
20= 00010100

. . .
```

bloco 0

bloco 1

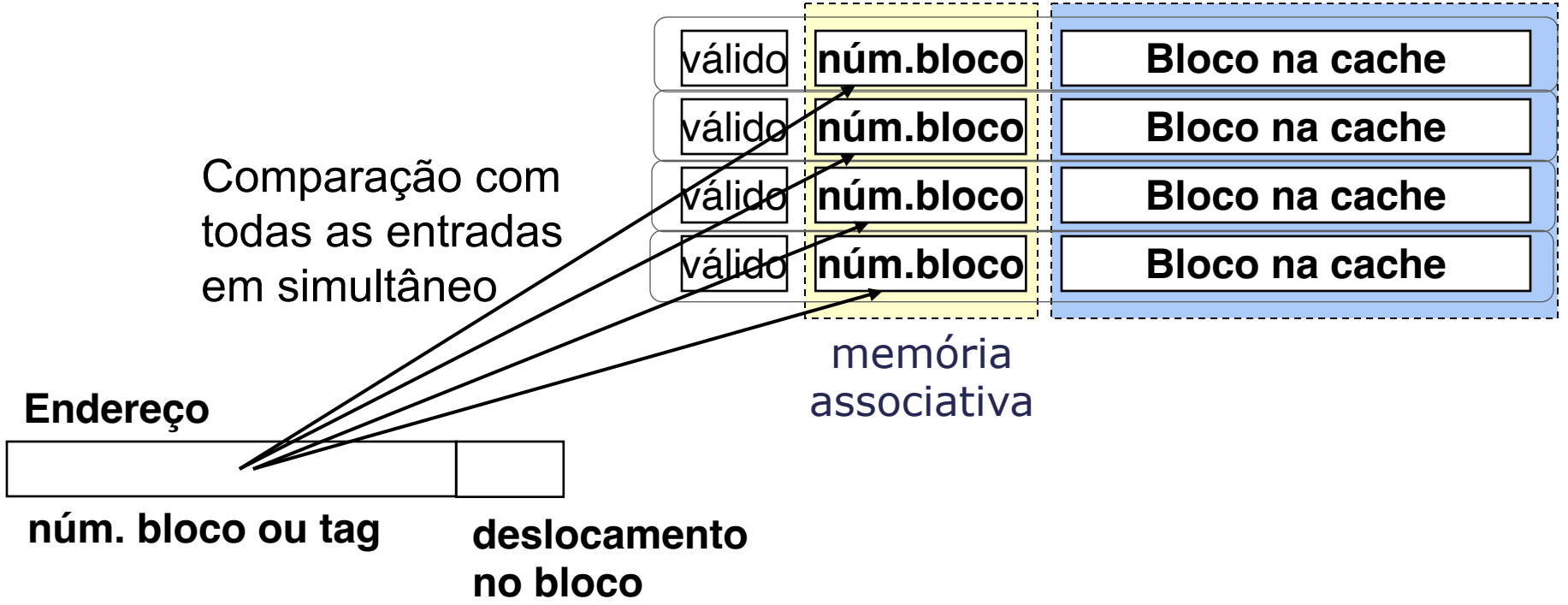
bloco 2

bloco 3

bloco 4

Cache associativa pura

- Usa o número do bloco como **chave** (ou *tag*) na memória associativa para procurar o bloco na cache



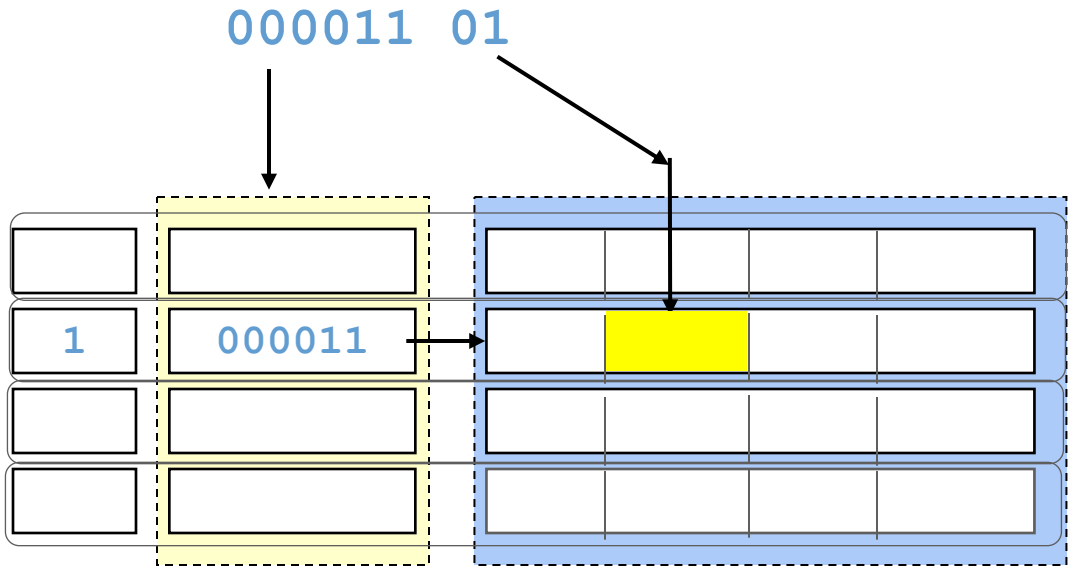
Endereços e a cache

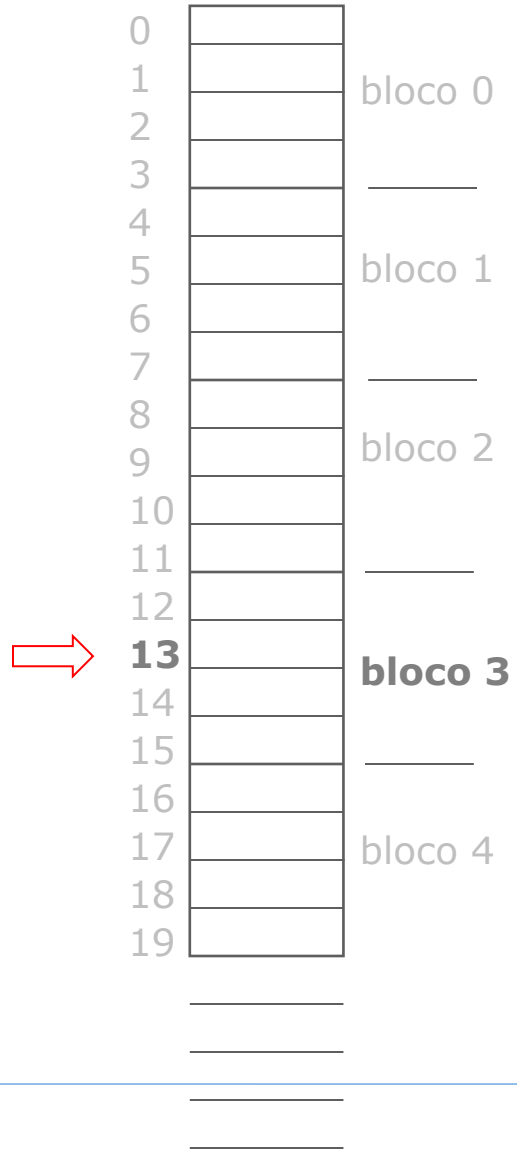
Exemplo: cache com 16bytes, 4 linhas de 4 bytes

Ler endereço 13

Bloco memória = 13/4 = 3

Byte nesse bloco = 13%4 = 1



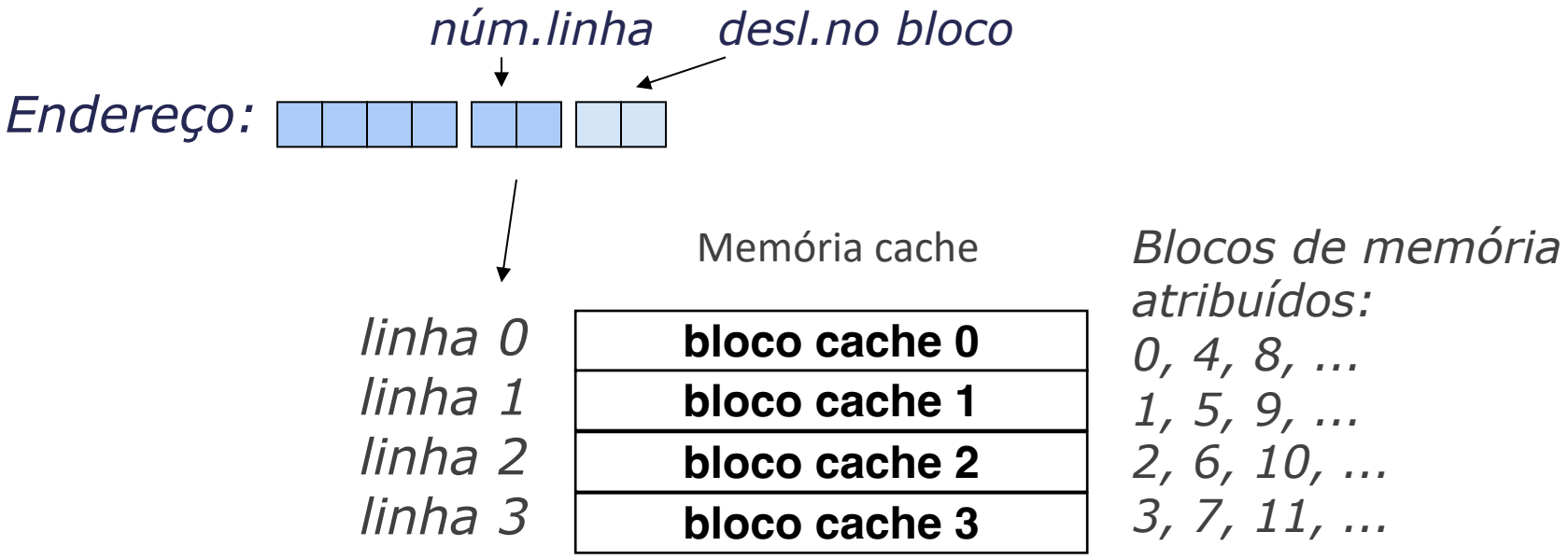


Caches associativas puras

- As Caches Associativas são muito eficientes:
  - a pesquisa do bloco faz-se em paralelo, com tantos comparadores quantos os blocos que cabem na cache
  - qualquer bloco de memória pode estar em qualquer linha da cache
- Mas são complexas e muito caras:
  - muitos comparadores, um para cada linha da cache
  - cada comparador é de tantos bits quantos os bits no número do bloco
  - cada linha da cache deve guardar todos os bits do número do bloco nessa posição

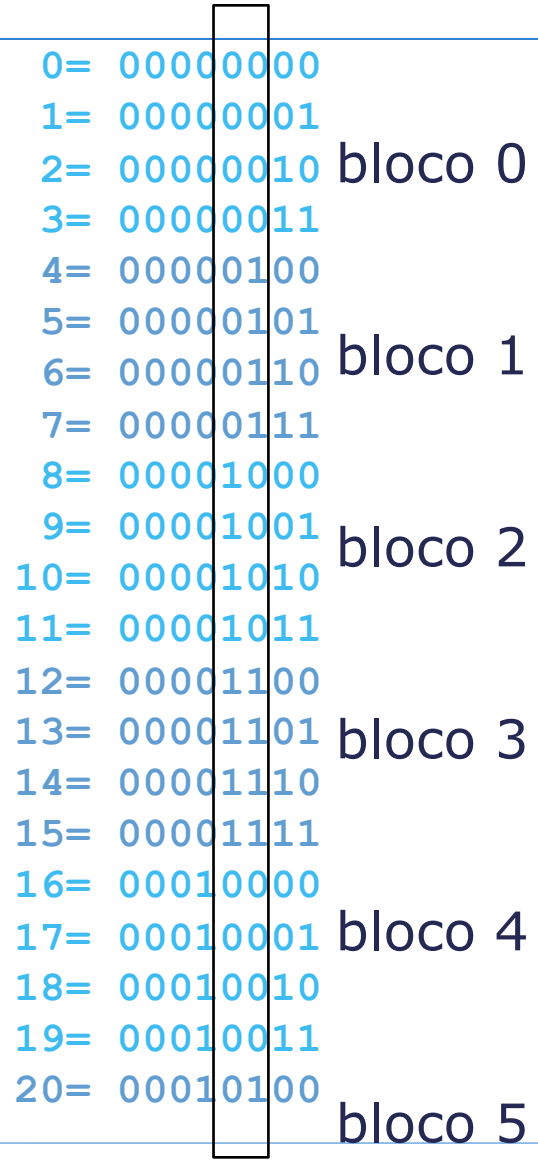
Simplificando

- Pré-definindo uma linha da cache para cada bloco de memória:
  - Linha = nºbloco % nº de linhas da cache
  - Exemplo: 4 linhas --> 2bits



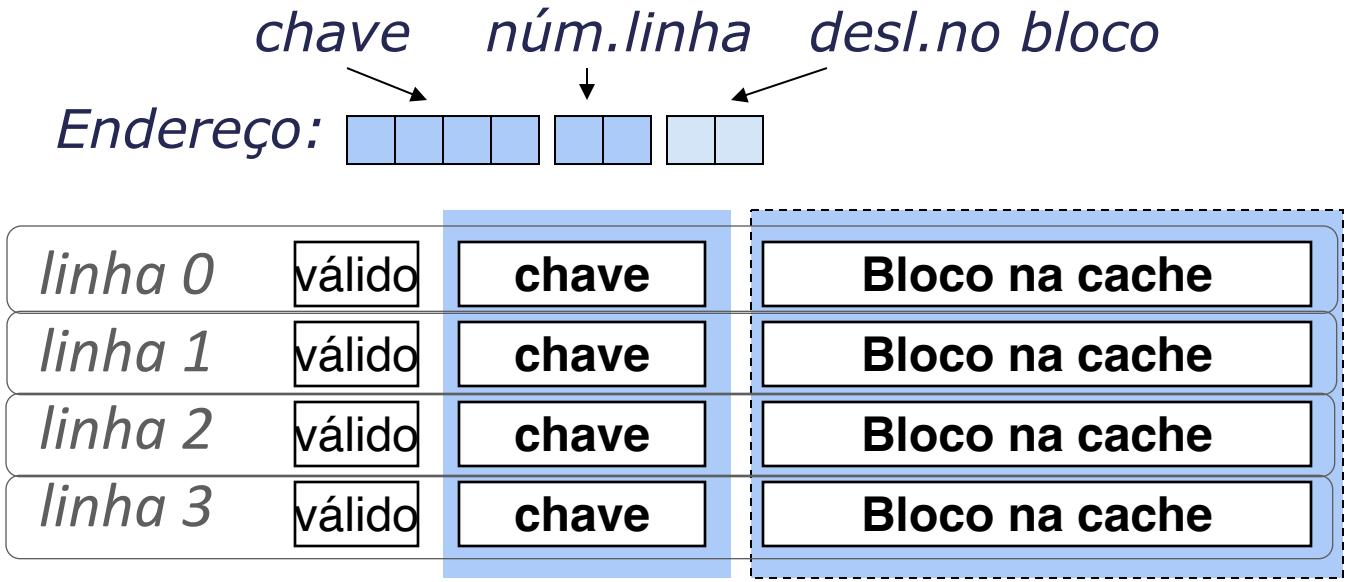
Simplificando

- Pré-definindo uma linha da cache para cada bloco de memória:
  - Linha = nºbloco % nº de linhas da cache
  - Exemplo: 4 linhas → 2bits



Cache de mapa direto

- É necessário manter para cada linha uma chave (ou tag) que identifique qual o bloco na cache de entre os vários possíveis:
  - o que distingue os vários blocos são os bits mais significativos restantes do endereço
- **Exemplo**: end. de 8bits, cache de 4 linhas e 4 bytes p/bloco:



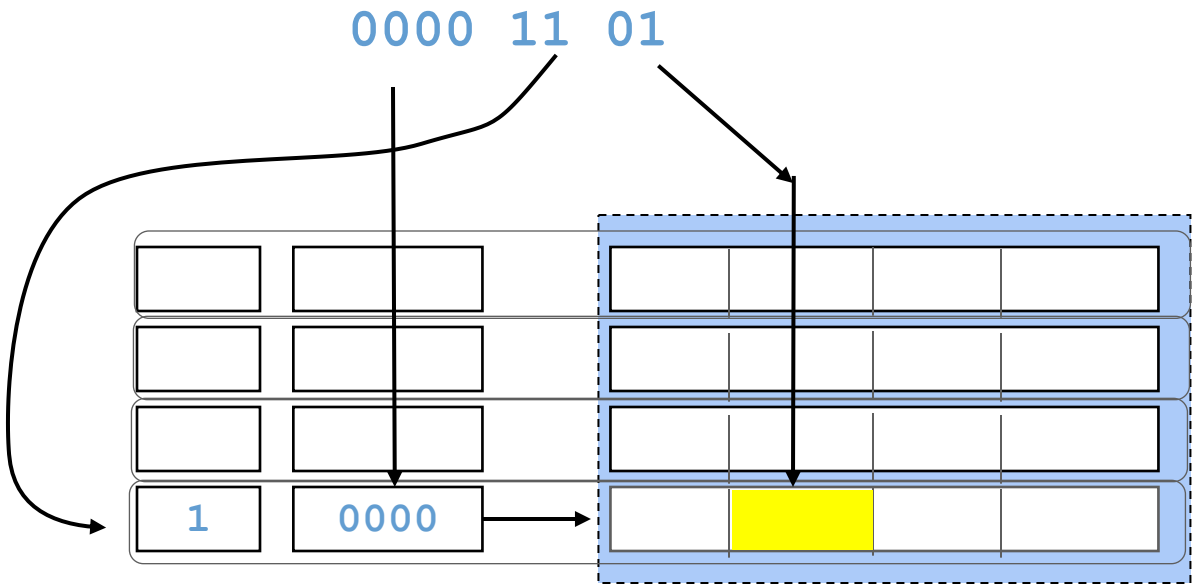
Endereços e a cache

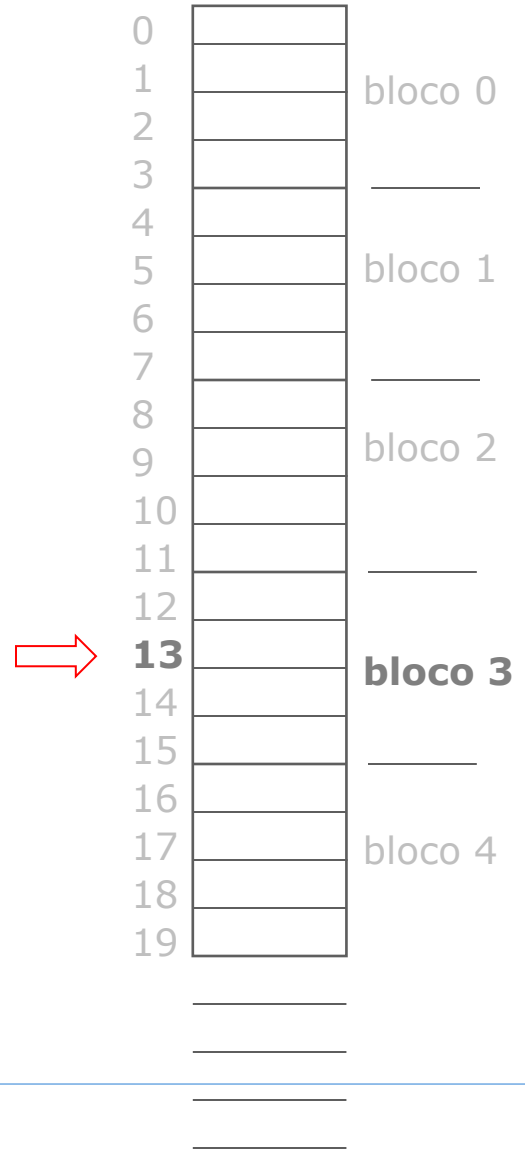
Exemplo: cache com 16bytes, 4 linhas de 4 bytes

Ler endereço 13

Bloco memória = 13/4 = 3

Byte nesse bloco = 13%4 = 1





Características da Cache de mapa direto

- As Caches de mapa direto são mais baratas do que as Associativas:
  - O mapa da Cache tem menos bits;
  - Não necessita de memória associativa e só precisa de 1 comparador (de tantos bits quantos os na chave)
- São eficientes na pesquisa em Cache:
  - usam bits do endereço como índice na Cache e restantes como chave.

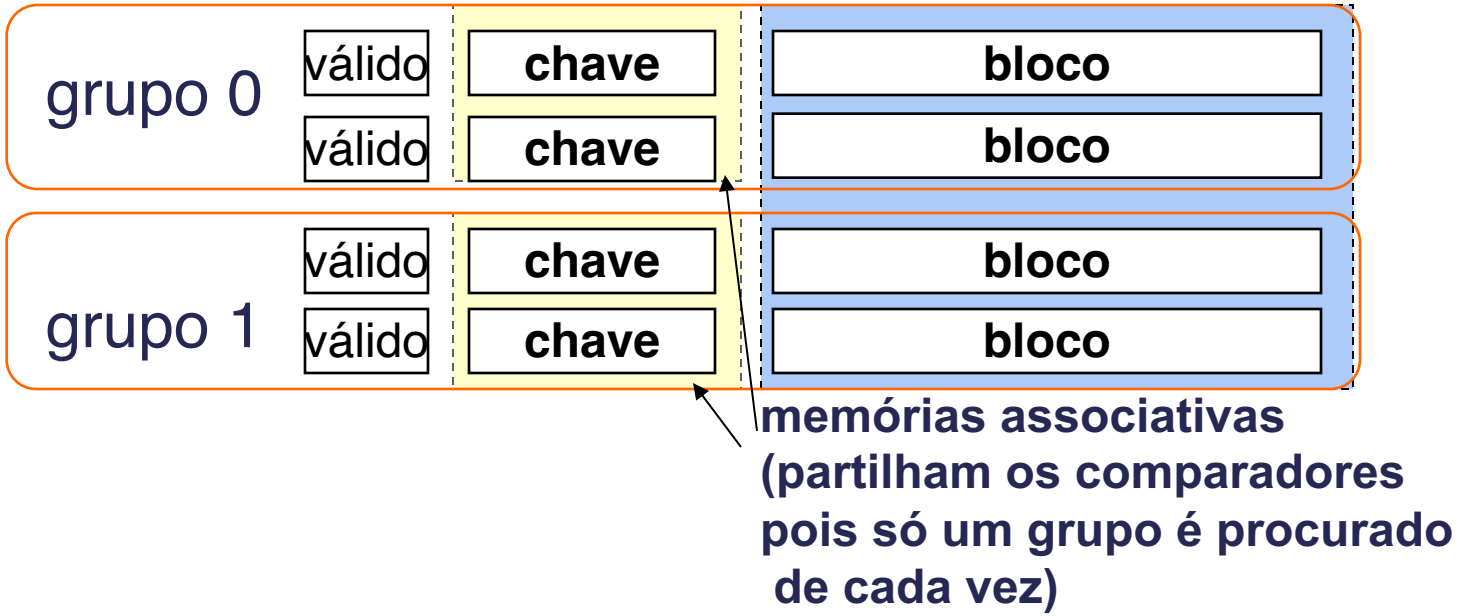
- **Problema**: levam a colisões, mesmo que existam linhas livres!

Cache associativa por grupos ou conjuntos (set associative)

- Solução híbrida de compromisso
- Procura-se responder às desvantagens das duas técnicas anteriores:
  - evitar as colisões no mapa direto
  - ter menos comparadores que a associativa pura
- Abordagem:
  - cada "linha" de mapa direto dá lugar a um grupo (conjunto) de blocos
  - cada bloco pode ficar em qualquer linha dentro do grupo
  - a busca no grupo é efectuada procurando a chave do bloco usando memória associativa

Cache associativa por grupos

- Os blocos de memória são mapeados diretamente num grupo (ou conjunto de linhas)
  - o núm. do grupo é dado pelo: núm.bloco%núm.grupos
- Dentro do grupo pode ficar em qualquer posição, como se o grupo fosse uma "mini cache associativa pura"

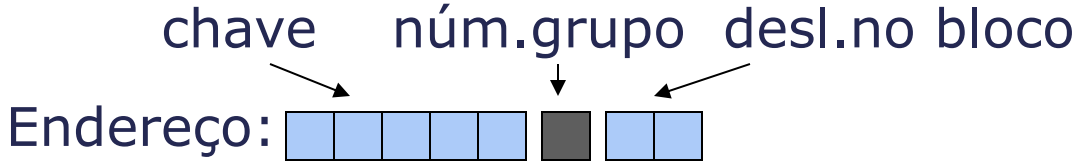


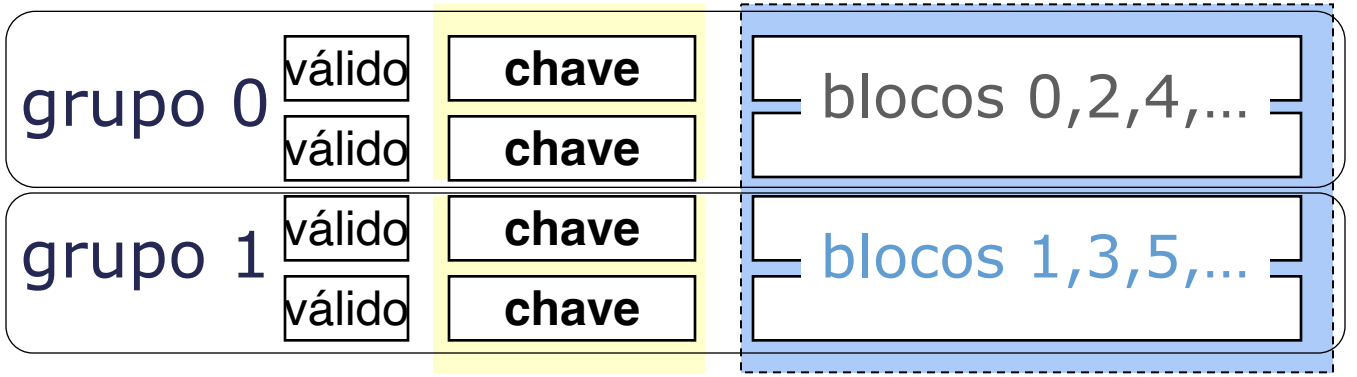
Cache associativa por grupos (ou conjuntos)

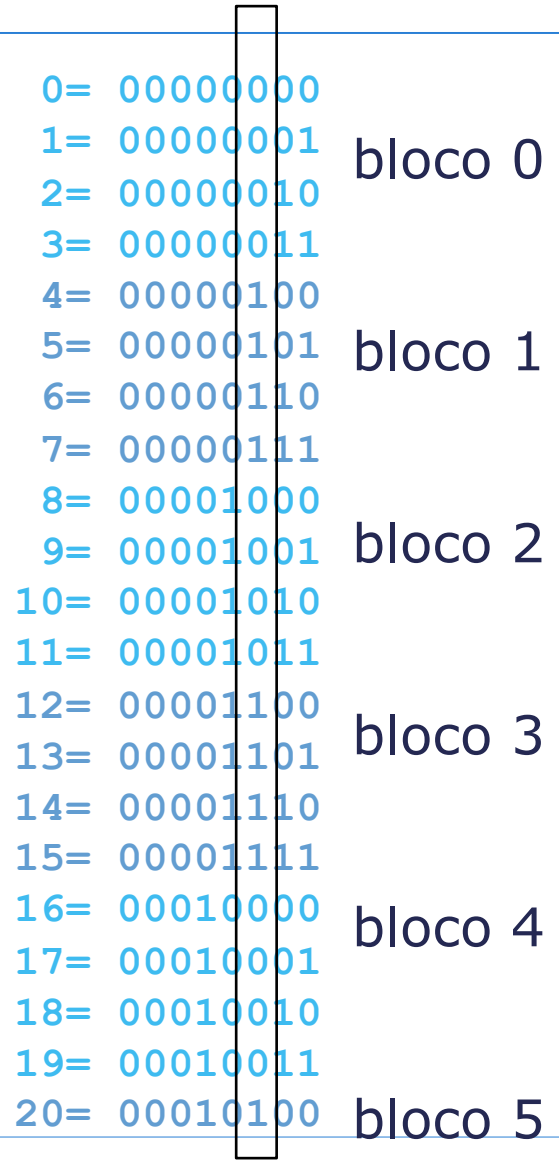
- **Exemplo, end. de 8 bits:**
  - **cache com 2 grupos**
  - **4 blocos de 4 bytes, 2 por grupo**
  - núm. do grupo é dado por: **núm.bloco%núm.grupos**

(como no mapa direto para obter a linha)

- ou seja:







Endereços e a cache

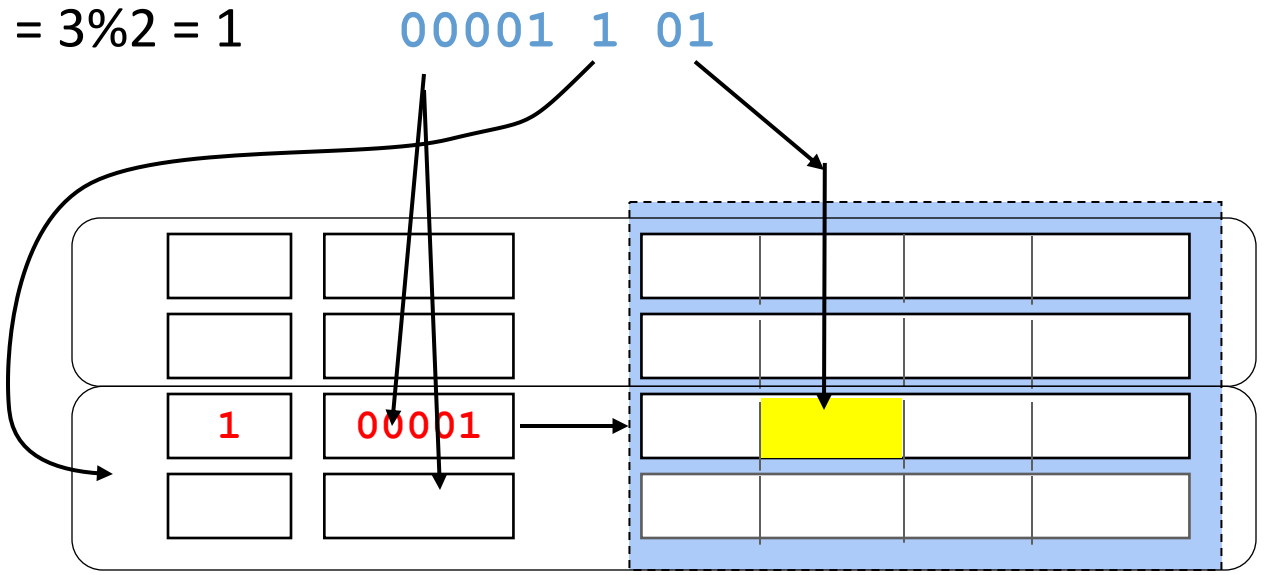
Exemplo: cache com 16bytes, 4 linhas de 4 bytes

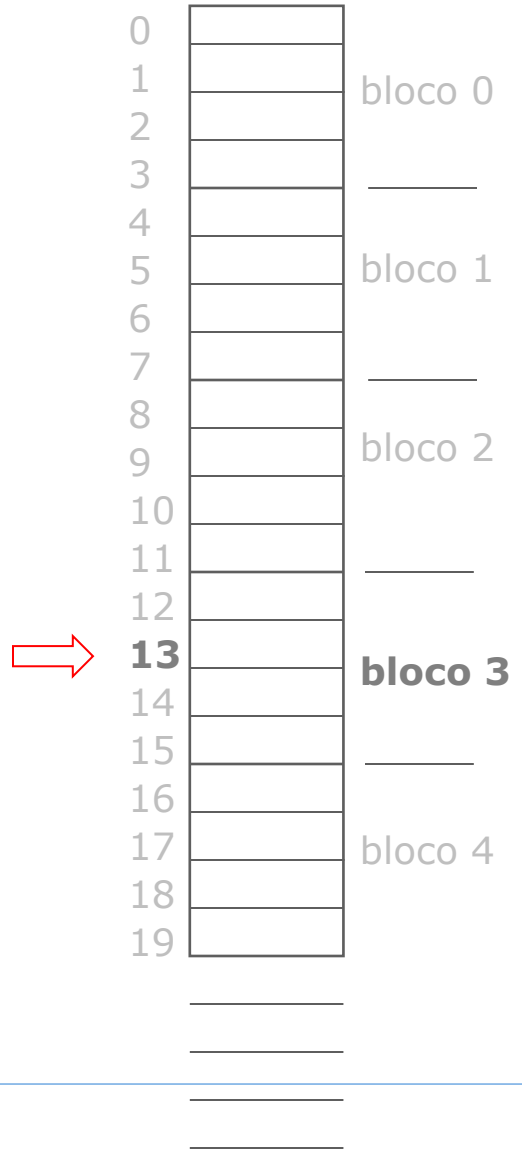
Ler endereço 13

Bloco memória = 13/4 = 3

Byte nesse bloco = 13%4 = 1

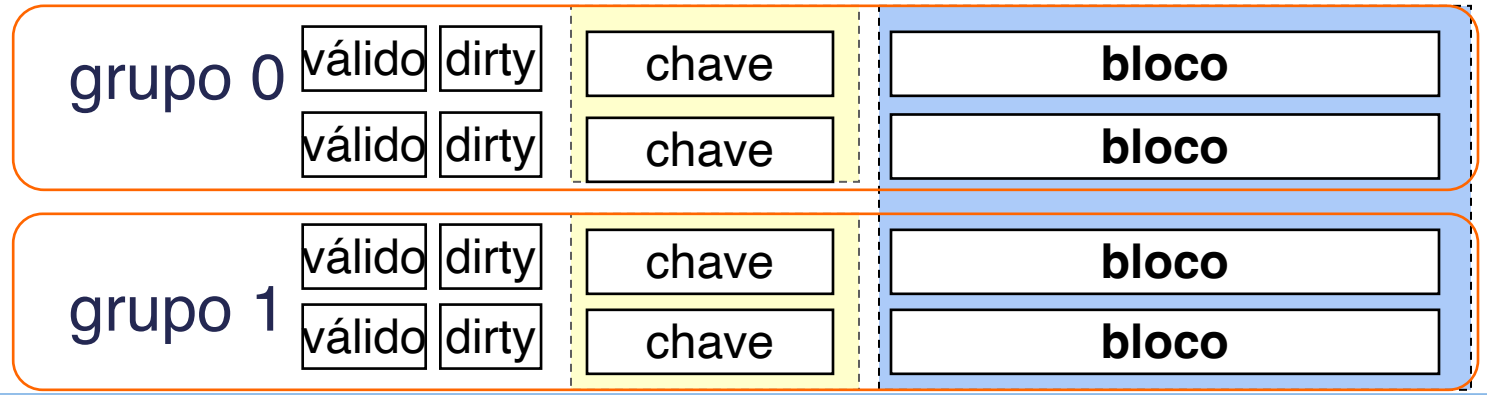
Grupo = 3%2 = 1





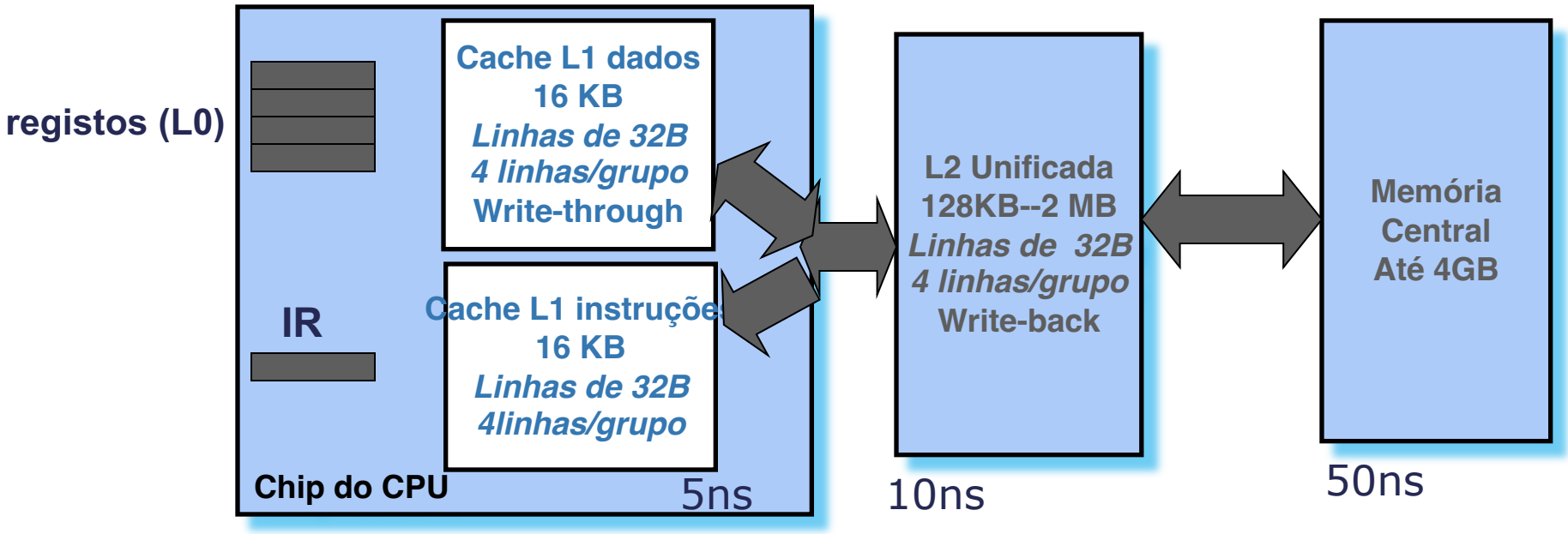
Política de Propagação de Escritas

- **Write through** – As escritas são efetuadas na cache e propagadas para o nível seguinte
- Write back – as escritas são feitas na cache e a propagação para os níveis seguintes deferida
- É necessário marcar que estão por escrever na memória
  - **Dirty bit** –colocado a 1 a cada escrita
  - **Exemplo** para associativa por grupos:

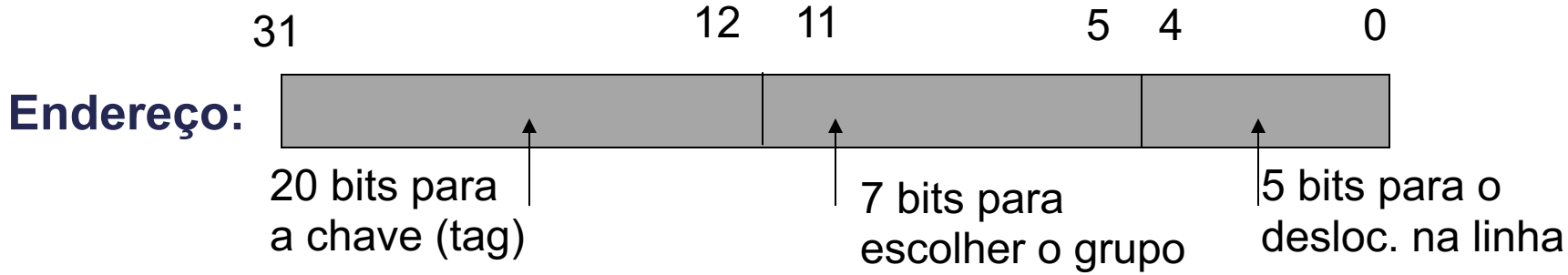


Exemplo de cache do Intel Pentium

As capacidades e tempos dependem do modelo

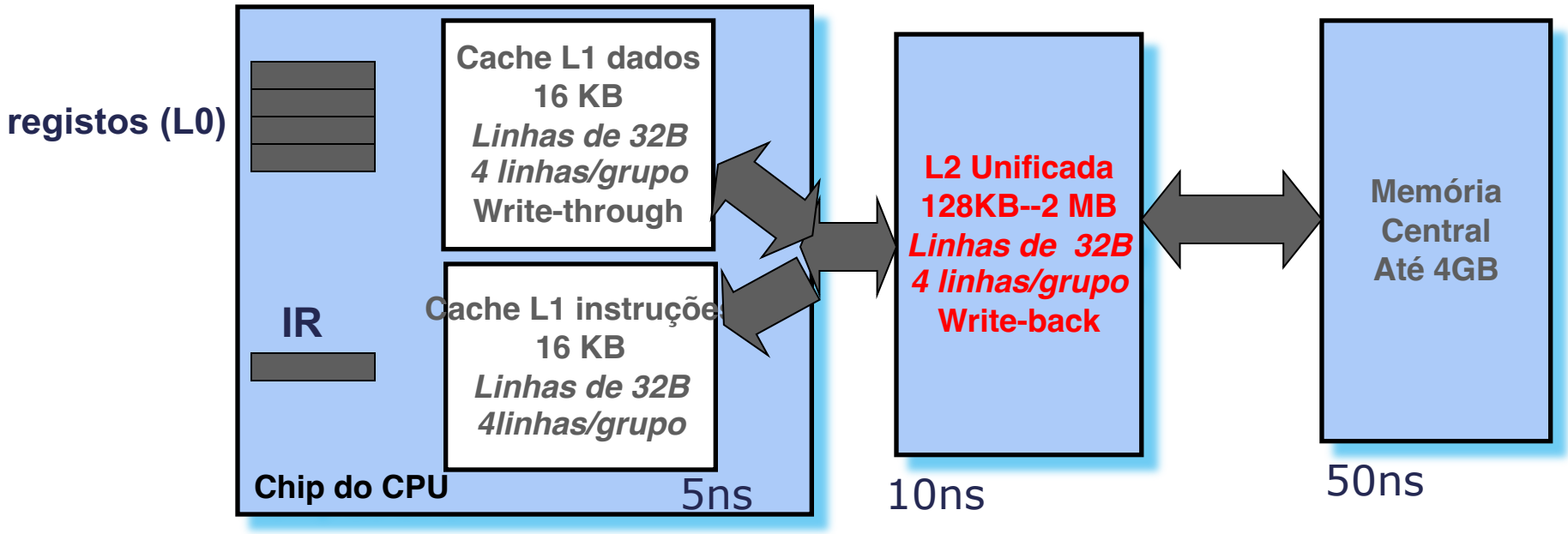


Exemplo Cache L1 (16KB) = 128 grupos * 4linhas/grupo * 32B/linha

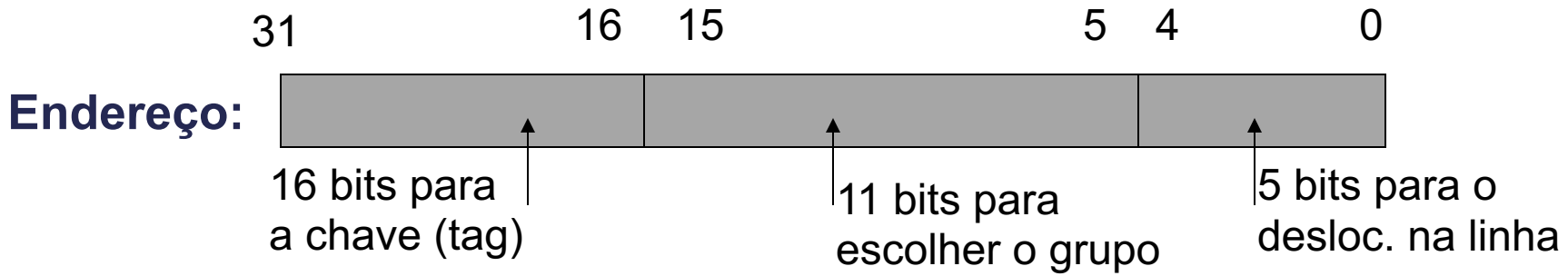


Exemplo de cache do Intel Pentium

As capacidades e tempos dependem do modelo



Exemplo de Cache L2 de 256KB = 2048 grupos*4linhas/grupo*32B/linha



Exemplo mov (393282),%eax

- Tratamento na L2 do endereço: 393282
  - em binário:

    0000 0000 0000 0110 0000 0000 0100 0010
  - dividindo de acordo com a cache L2:

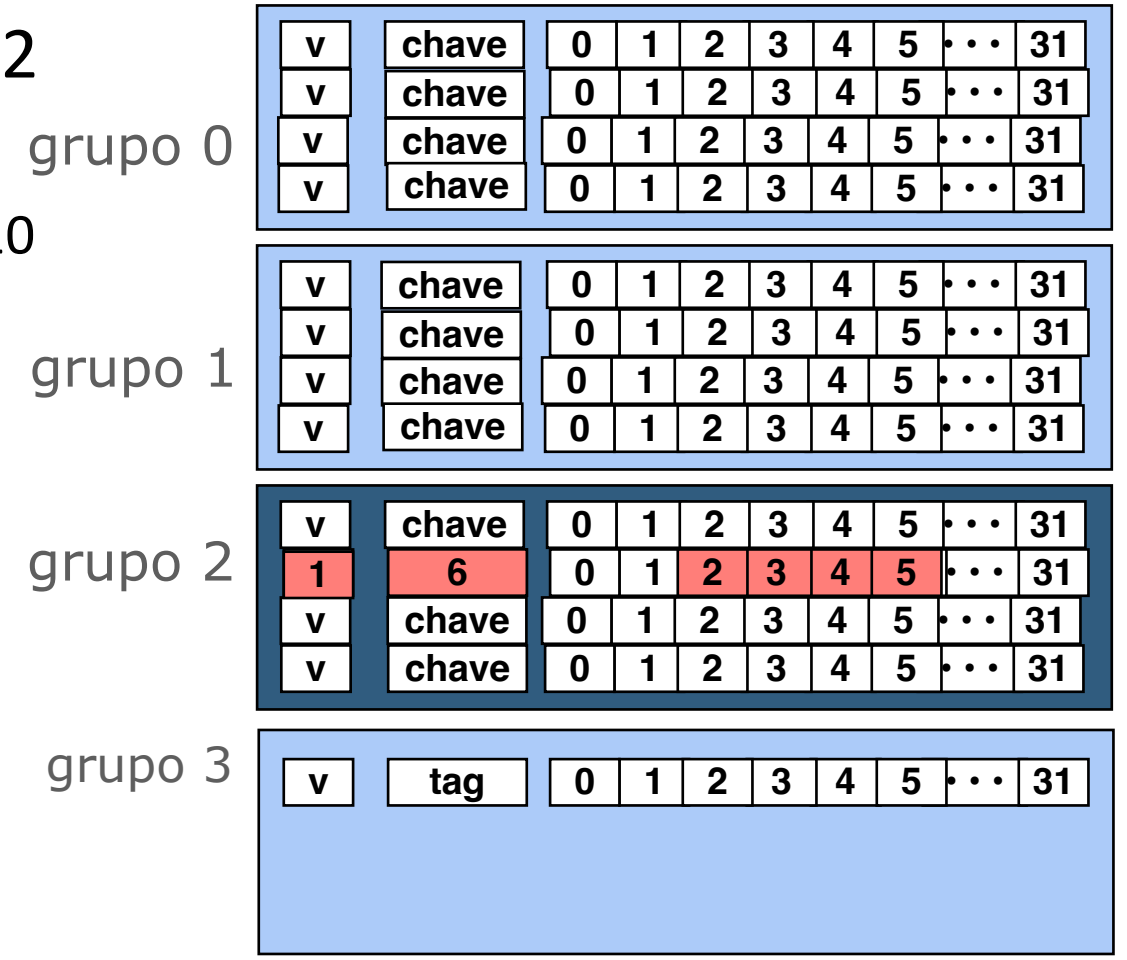
    00000000000000110 00000000010 00010

deve procurar no grupo 2

a chave= 6

se encontrar

lê a partir do byte 2 (4 bytes)



Quando falha a cache (miss)

- Existe *misses* quando as localidades não se verificam:
  - por acesso a dados não contíguos em memória
    - ou estruturas que não cabem na cache
  - muda o conjunto de instruções em execução (*working-set*): devido a jumps, calls,…
    - ou este conjunto não cabe todo na cache
  - o SO troca de programa em execução
    - muda de código e de dados…

Tratamento dos misses

- Se Cache miss: é preciso arranjar espaço para o novo bloco.
  - Se associativa pura pode ficar em qualquer linha da cache
  - Se mapa direto tem de ficar na linha respetiva
  - Se associativa por grupos tem de ficar no grupo respetivo

1. Se **existe linha livre**, marca como válida, atualiza a chave (tag) e copia o conteúdo do bloco

Tratamento dos misses (2)

2. Se **tudo ocupado**, qual a linha a usar?
   - É necessário escolher uma vítima para dar lugar ao novo bloco

     Uma qualquer? . . .
   - A do bloco que no futuro não vai ser necessário!

     Mas qual é essa?

3. E depois de escolher vítima, se usar write-back, pode ter de atualizar memória com o conteúdo da vítima

Escolha da vítima

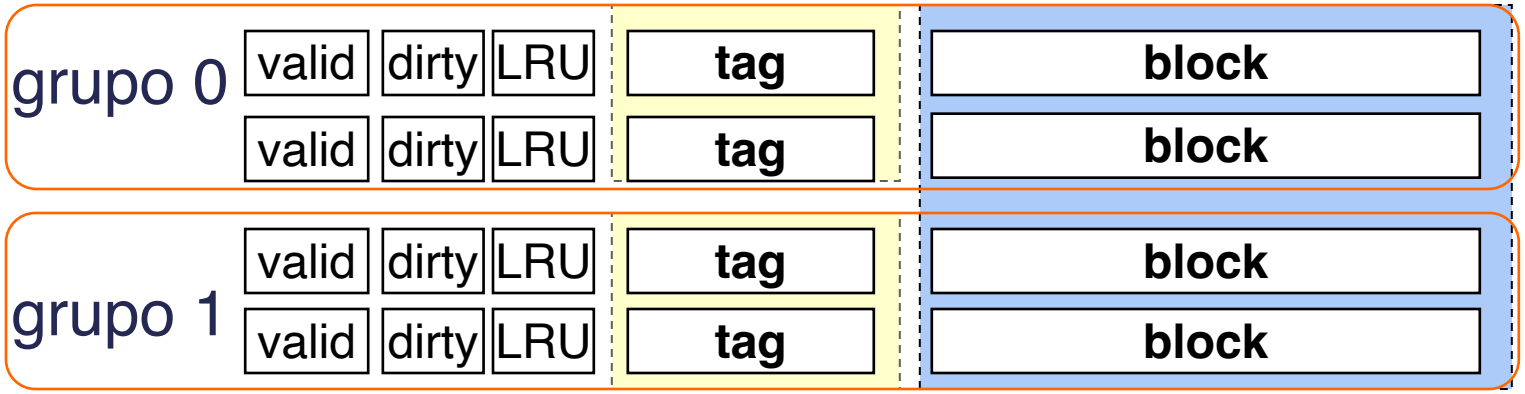
- Tentar prever o que será ou não necessário no futuro:
  - **LRU** – *Least Recently Used* – eliminar o bloco que há mais tempo não é usado (exige contar tempo)
  - **LFU** – *Least Frequently Used* – eliminar o bloco que foi referenciado menos vezes (exige contar acessos)
  - **FIFO** – *First In First Out* – eliminar o bloco mais antigo (exige manter lista ou tempo do 1º acesso)
  - **Aleatório** – escolhe-se um bloco de forma aleatória (exige um gerador de pseudo-aleatórios)

Pseudo-LRU de 1 bit

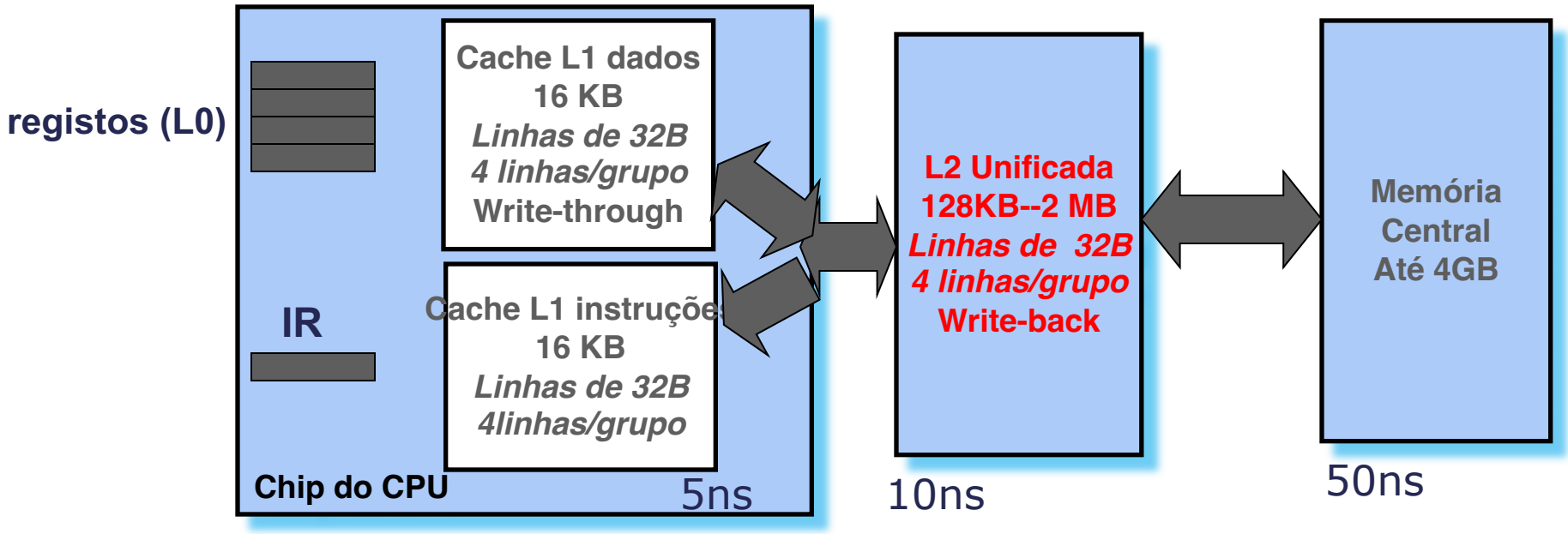
- A cada acesso a uma linha, passa a 1 um bit indicando o acesso
  - as linhas não acedidas recentemente têm o valor a 0
- Quando todas as linhas (do grupo) forem acedidas, todos os bits são mudados para 0 (estão em igualdade)
- Espera-se que, em caso de miss, deve haver pelo menos uma linha com bit de acesso a 0
- Se todas as linhas de um grupo têm o bit a 0, considera-se que os acessos são aproximadamente semelhantes e qualquer uma pode ser substituída

Completando a cache ass por grupos

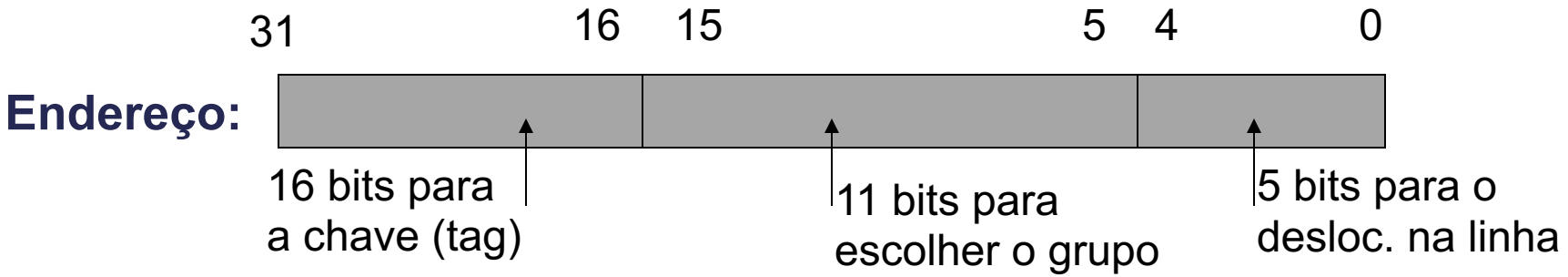
- Cada linha pode ter, além dos dados:
  - Valid bit – a 1 se entrada válida
  - LRU bit – a 1 se acedida recentemente
  - Dirty bit – a 1 se escrita (apenas se escritas write-back)
  - Tag (ou chave) – para distinguir os blocos
- Exemplo para associativa por grupos:



Exemplo de cache do Intel Pentium



Exemplo de Cache L2 de 256KB = 2048 grupos*4linhas/grupo*32B/linha



As capacidades e tempos dependem do modelo

Exemplo mov (393282),%eax

- Tratamento na L2 do endereço: 393282
  - em binário:

    0000 0000 0000 0110 0000 0000 0100 0010

  - dividindo de acordo com a cache L2:

    0000000000000110 00000000010 00010

deve procurar no grupo 2

a chave= 6

se não encontra tem de

colocar no grupo 2

→ pode copiar para a linha desocupada

Exemplo **mov (393282),%eax**

- Tratamento na L2 do endereço: 393282
  - em binário:

0000 0000 0000 0110 0000 0000 0100 0010

  - dividindo de acordo com a cache L2:

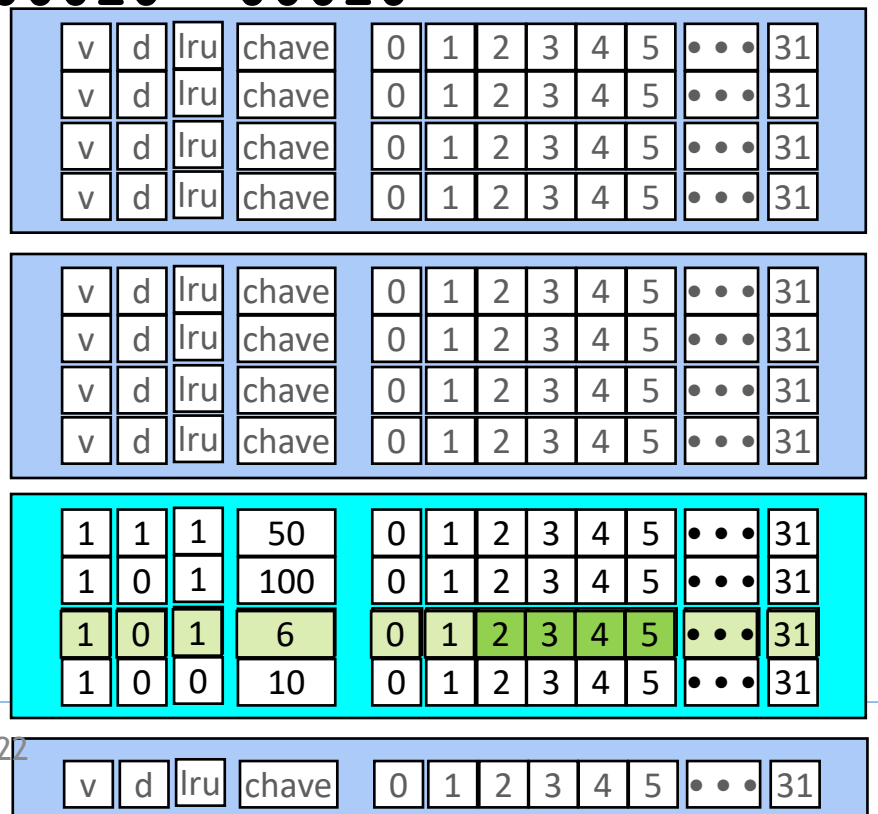
00000000000110 00000000010 00010

deve procurar no grupo 2

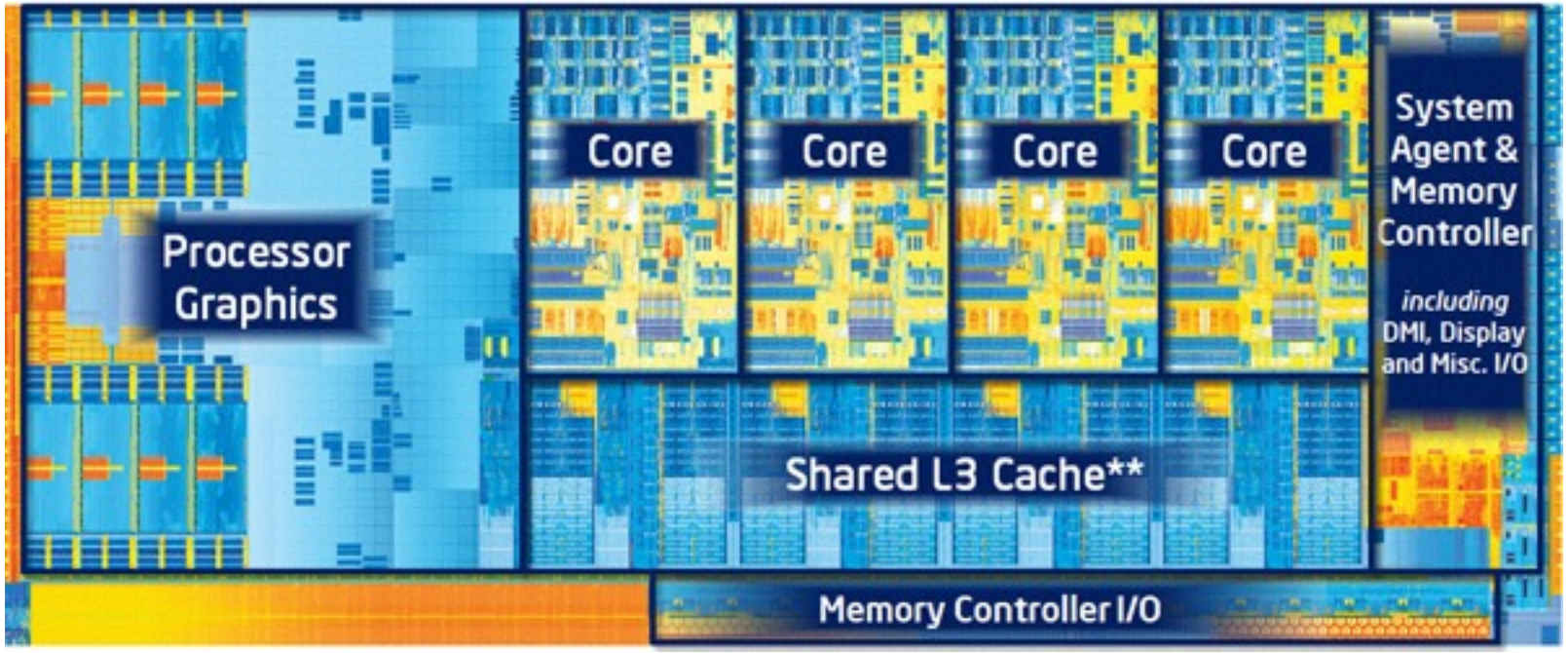
a chave= 6

se não encontra tem de colocar no grupo 2

->tem de escolher vítima

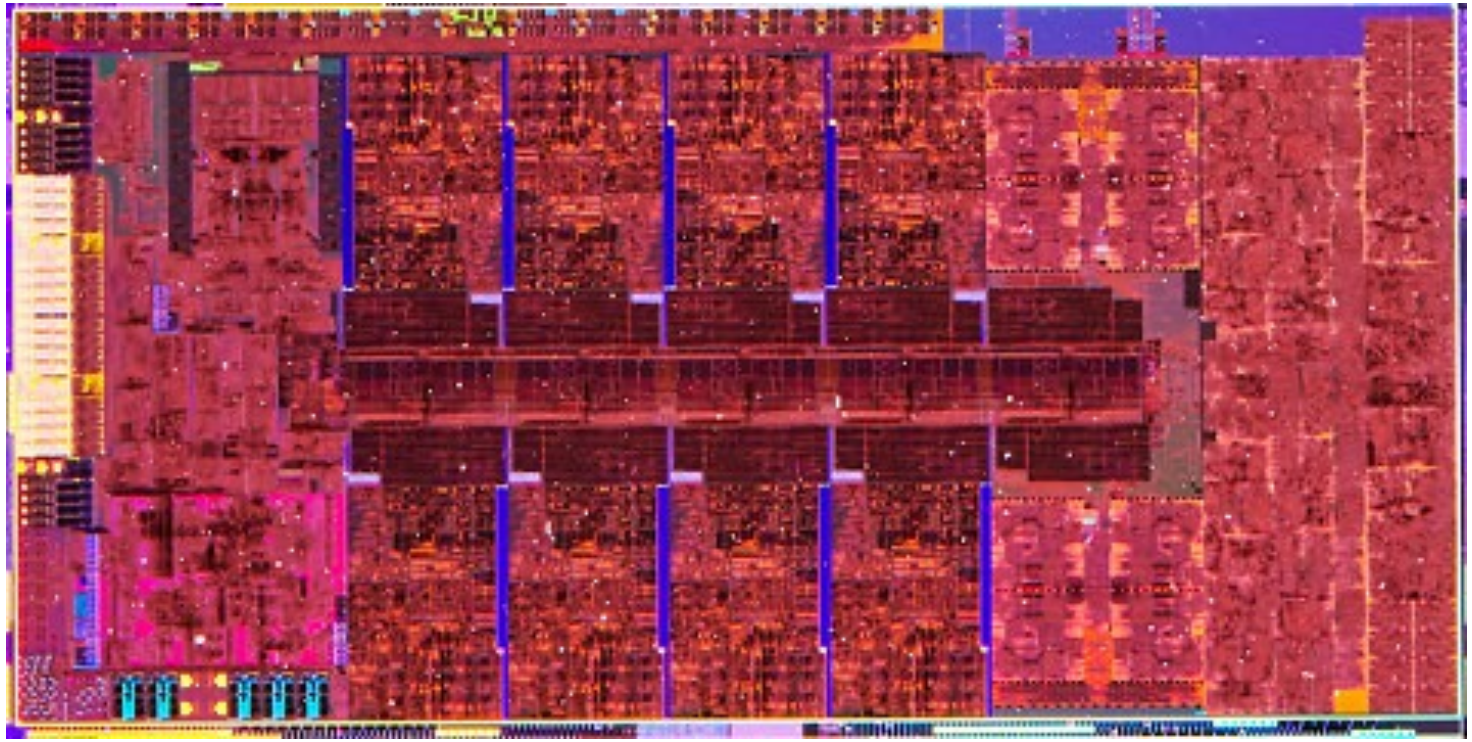


Intel Core i7-3770K (2014)

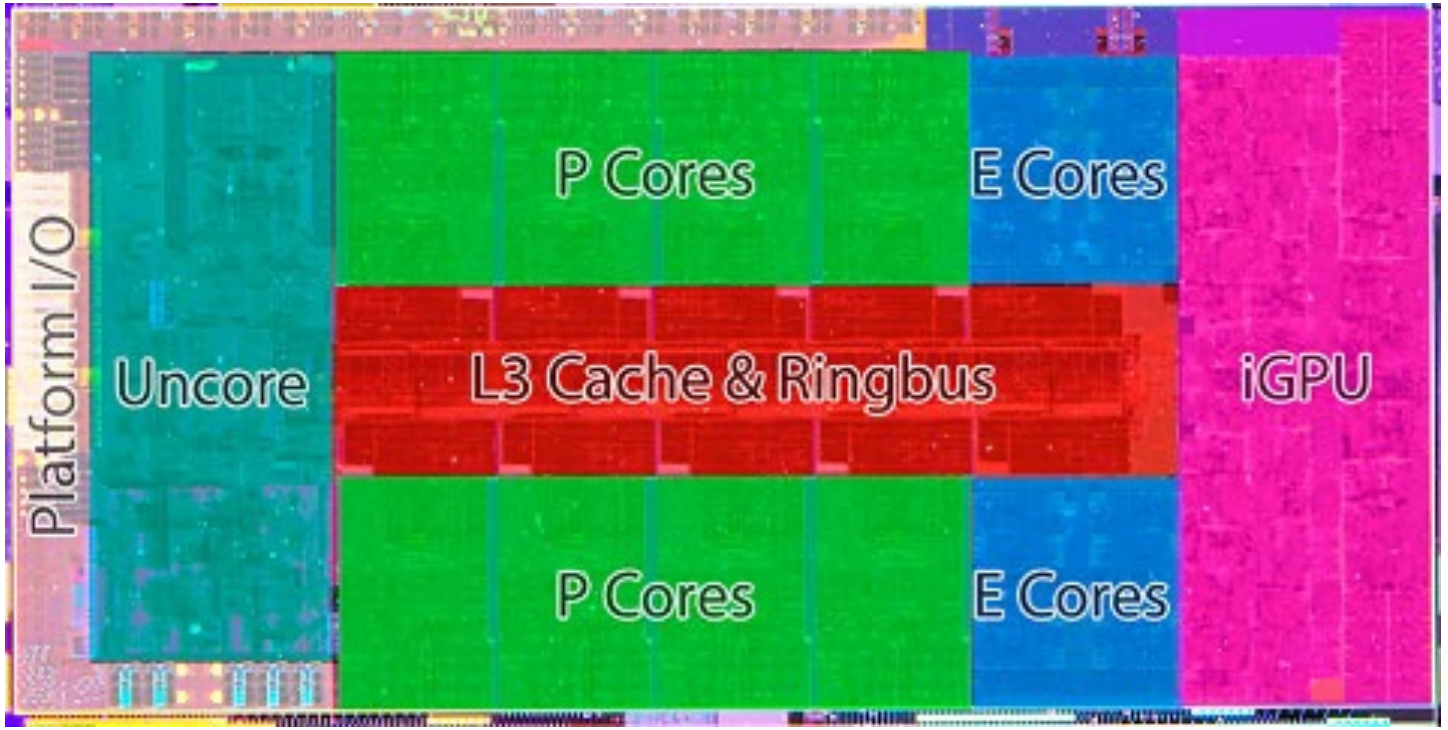


L1/core 32 KB 8-way set associative instruction caches
32 KB 8-way set associative data caches
L2/core 256 KB 8-way set associative caches
L3 shared 8 MB 16-way set associative cache
Max RAM 32 GB

Intel Core I9-12900K (2021)



Intel Core I9-12900K (2021)



P cores L1/core 64* KB 8*-way set associative instruction caches

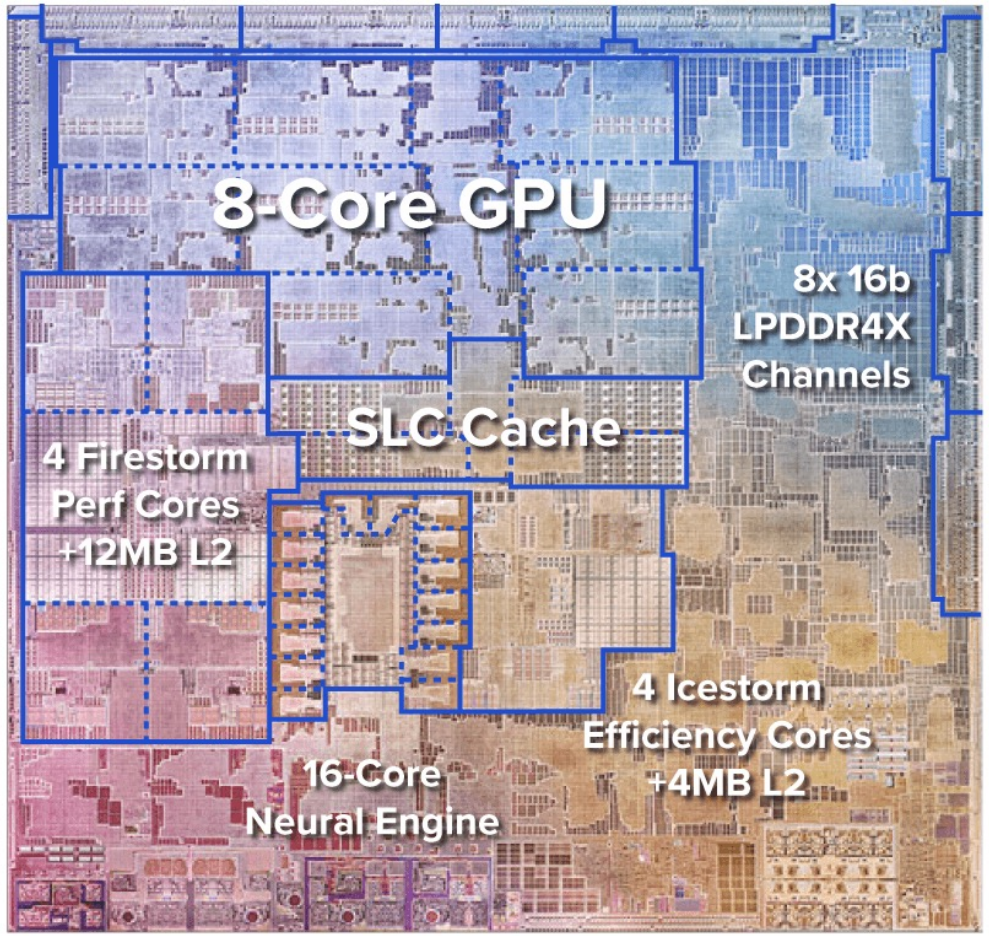
64* KB 8*-way set associative data caches

L2/core 1.25 MB 4*-way set associative caches

L3 shared 24 to 30 MB 16*-way set associative cache

* Deduced from previous generations of the processor

ARM M1 (2020)



Perf Cores

L1/core 192 KB 8-way set associative instruction caches

192 KB 8-way set associative data caches

L2 shared 12 MB ?-way set associative caches

L3 (SLC) shared with GPU 16MB ?-way set associative cache

? - Information could not be found