Programação em Sistemas Computacionais

Hierarquia de memória

Centro de Cálculo
Instituto Superior de Engenharia de Lisboa

Pedro Pereira palex@cc.isel.ipl.pt

Localidade

- **Localidade temporal**

  Um programa que acede a uma zona de memória, provavelmente irá aceder novamente a essa zona num curto espaço de tempo, uma ou mais vezes.

- **Localidade espacial**

  Um programa que acede a uma zona de memória, provavelmente irá aceder
  a vizinhanças dessa zona.

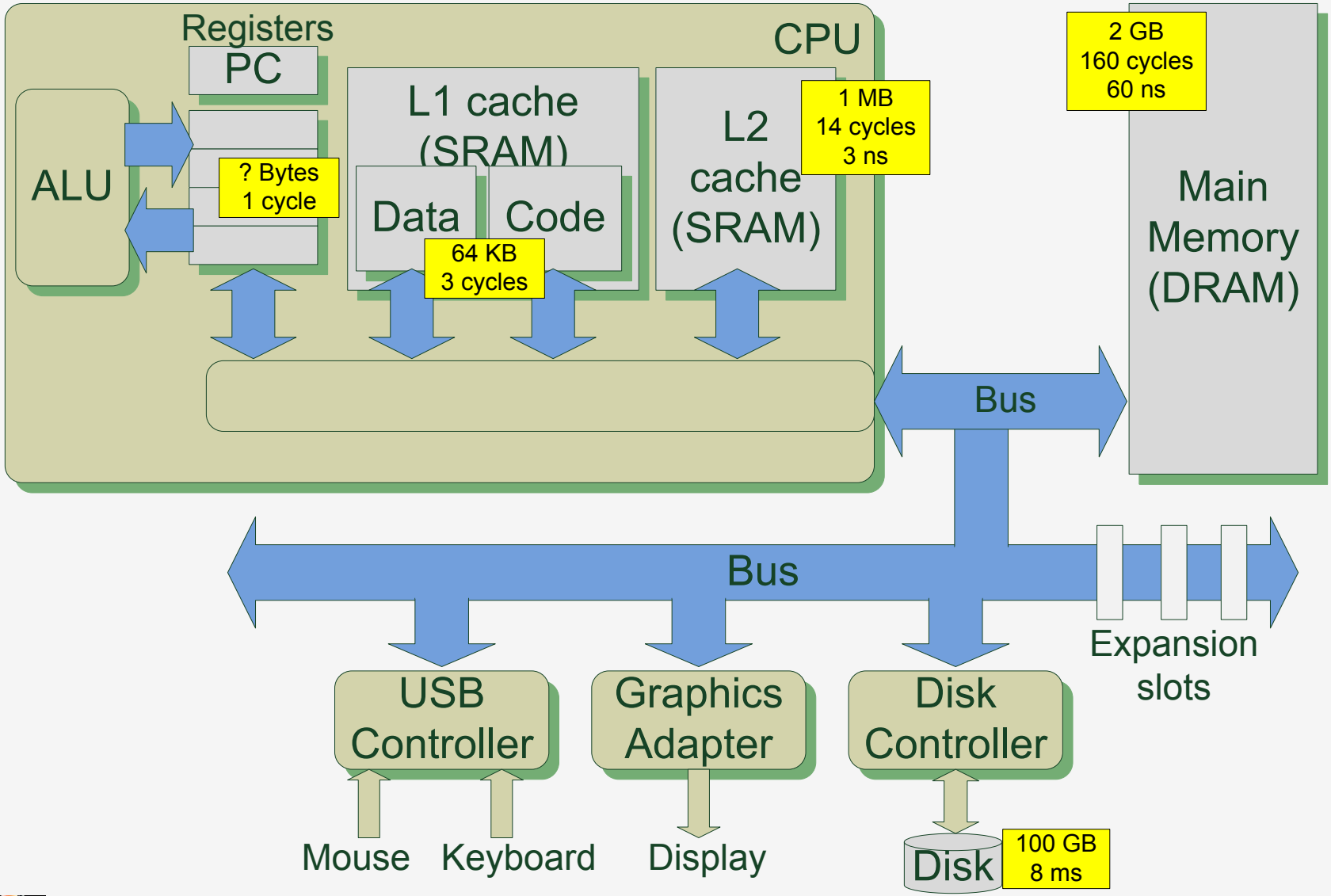
```
int sum(byte a[LINS][COLS]) {
  int l,c, res=0;
  for(c=0 ; c<COLS ; ++c)
    for(l=0 ; l<LINS ; ++l)
      res += a[l][c];
  return res;
}
```

Boa localidade relativamente à memória de código (instruções)?

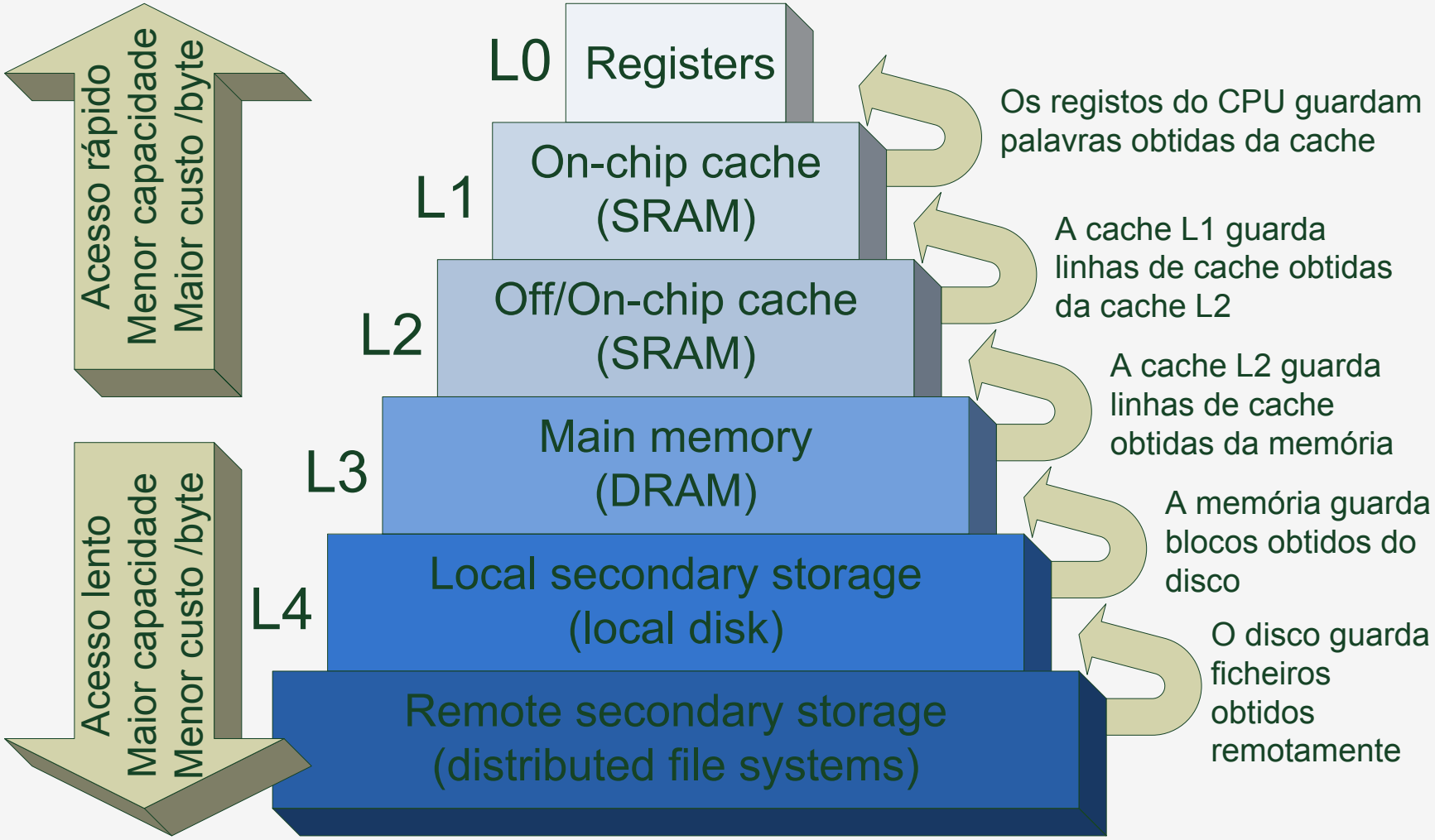
Boa localidade relativamente à memória de dados (variáveis)?

- ***Static* RAM (SRAM)**
  - 6 transístores por célula (bit)
  - Estável
  - Acesso rápido (10x)
  - Usada para cache (KBytes..MBytes)
- ***Dynamic* RAM (DRAM)**
  - 1 transístor e 1 condensador por célula
  - Custo baixo (100x)
  - Pequena dimensão
  - Memória principal (GBytes)

Estrutura típica



Hierarquia de memória



Conceito geral de cache

- **Cache *Hits* (acesso rápido)**

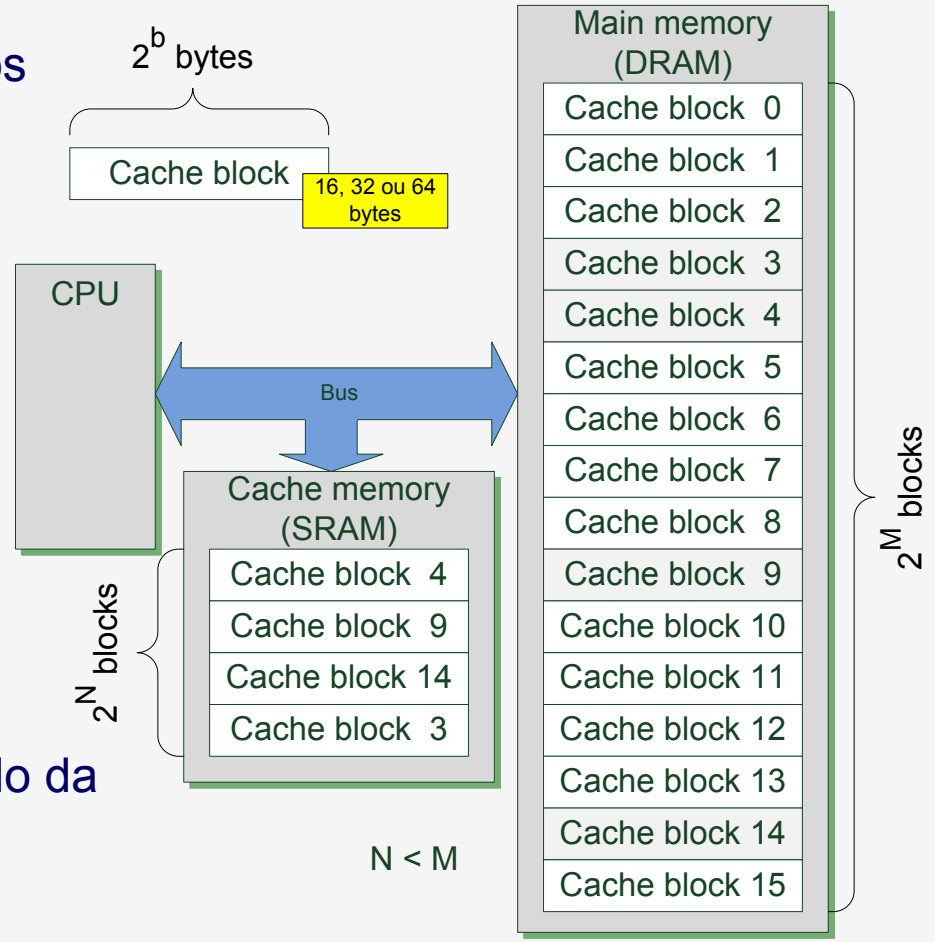
  Se os dados procurados estão num dos blocos da cache.

  1. Os dados são lidos/escritos apenas da/na cache.

- **Cache *Misses* (acesso lento)**

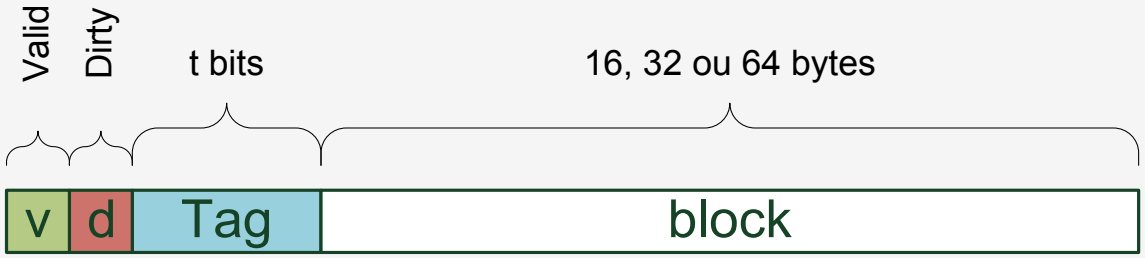
  Se os dados procurados não estão num dos blocos da cache.

  1. É escolhido um bloco da cache para ser substituído.
  2. Caso este bloco tenha sido alterado é escrito na memória principal.
  3. Todo o bloco do byte pretendido é lido da memória principal para a cache.
  4. Os dados são lidos/escritos apenas da/na cache.



Organização da cache

- **Organização em linhas:**



Bloco: Cópia de um bloco de dados (ex: 32 bytes)

Tag: Identifica o bloco

Uma parte do endereço em memória

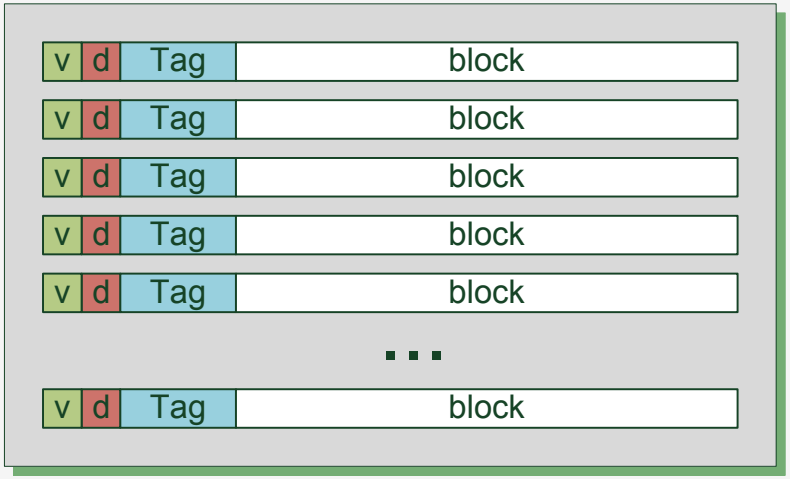
Valid bit: Cópia válida

Inicialmente, todas as linha estão inválidas

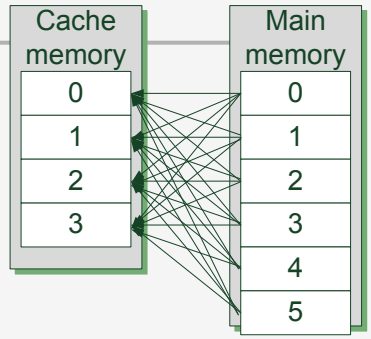
Dirty bit: Bloco alterado

Falta actualizar este bloco na memória

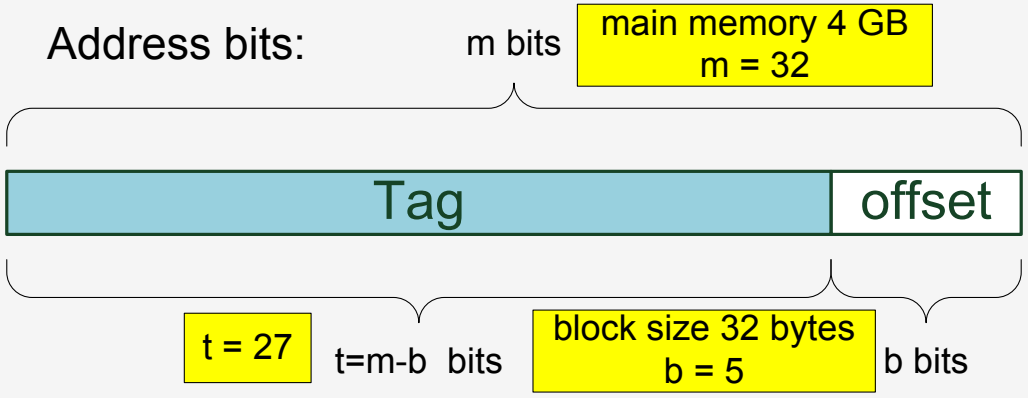
(não é usado em *write-through*)



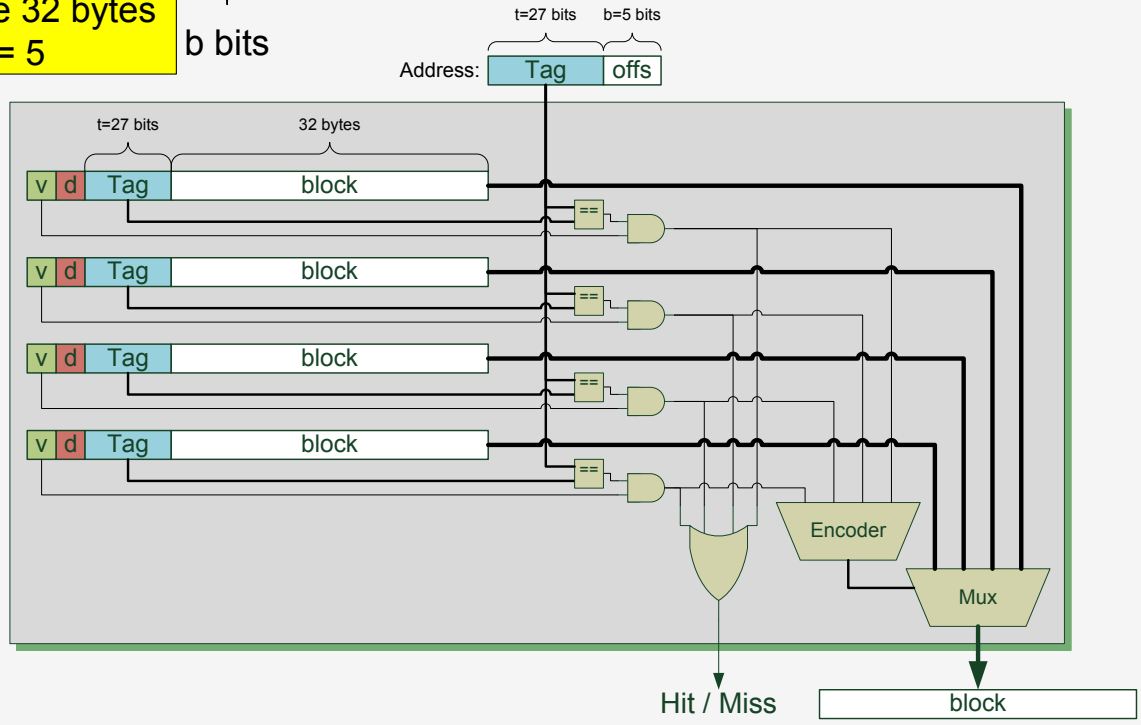
Fully Associative cache



- **Um bloco pode estar em qualquer linha da cache**

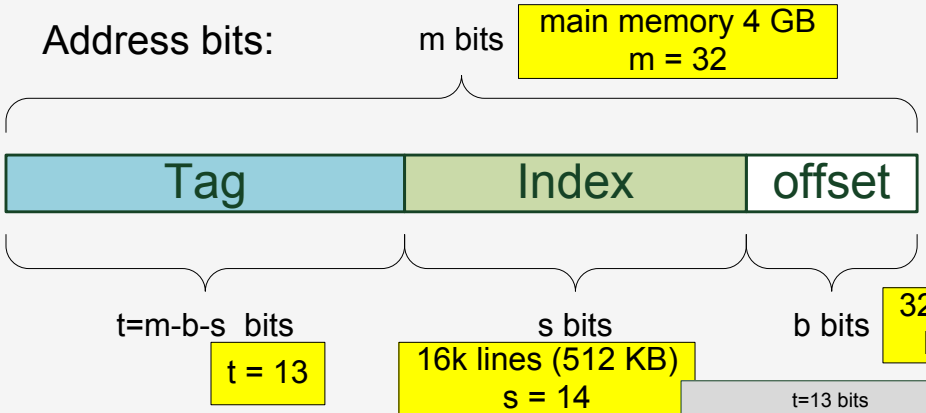


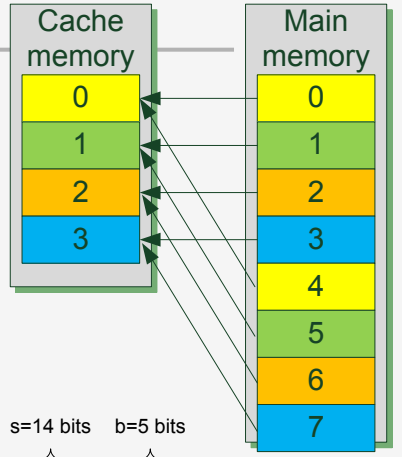
- **Um comparador para cada linha.**
- ***Enconder* e *Or* com lógica em cascata.**
- **Para médias e grandes dimensões tem acessos lentos e é dispendioso.**
- **Caches de pequena dimensão (TLBs)**

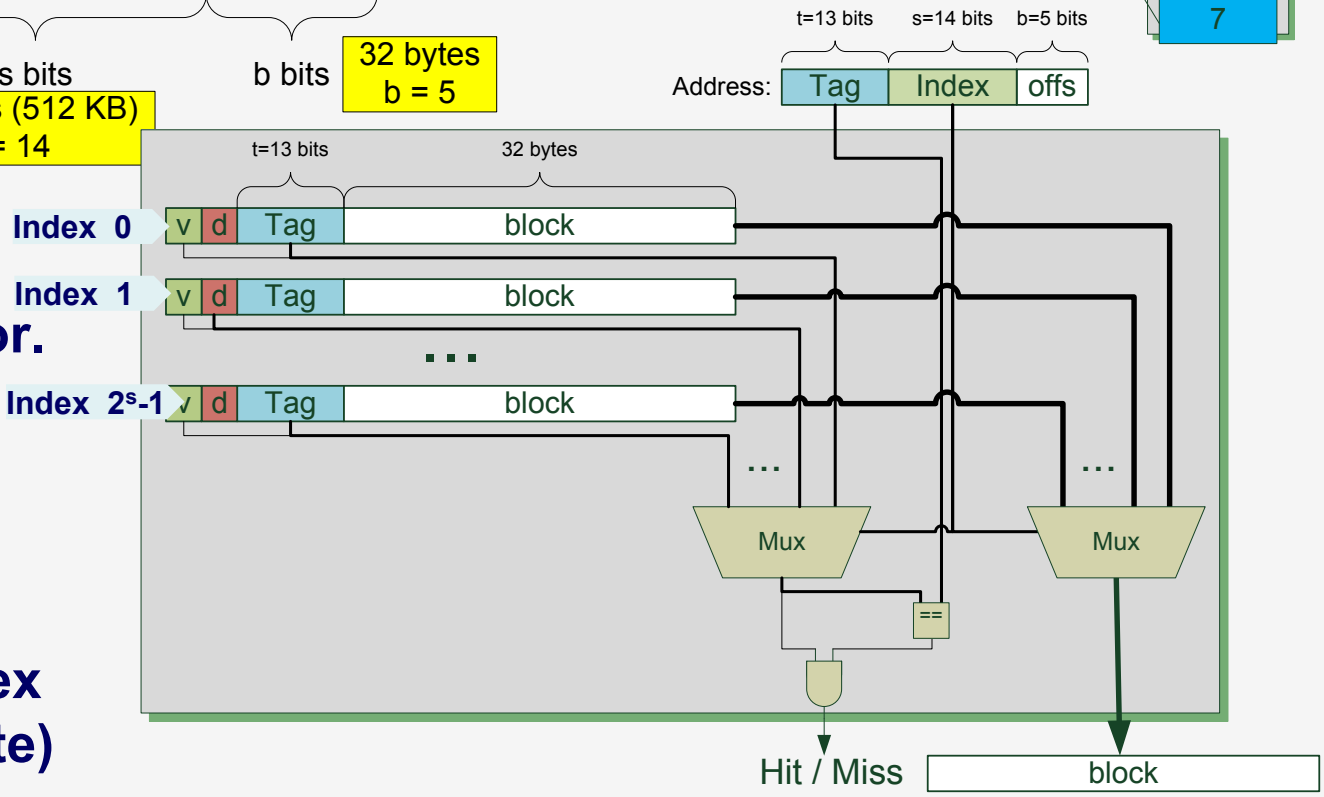


Direct-Mapped cache

- **Cada bloco só pode estar numa linha da cache**



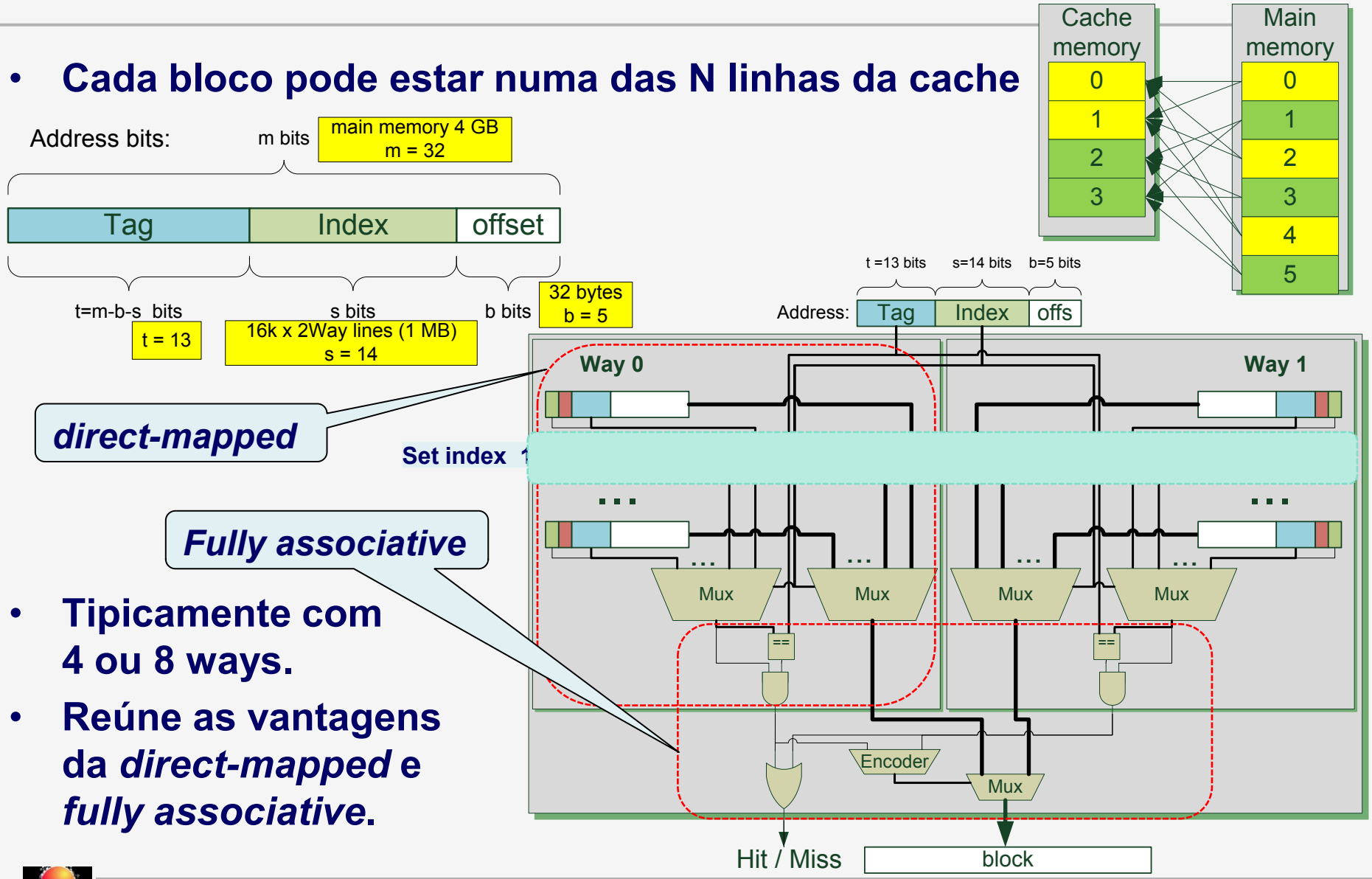




- **Um só comparador.**
- **Económico e com acessos rápidos.**
- **Cache *miss* se 2 blocos usados têm o mesmo index (situação frequente)**

Set Associative cache

- **Cada bloco pode estar numa das N linhas da cache**



- **Tipicamente com 4 ou 8 ways.**
- **Reúne as vantagens da *direct-mapped* e *fully associative*.**

Política de substituição por Miss

- **Direct-mapped**
  - Não tem. É sempre a mesma linha da cache
- **Set associative**
  - Sem *validate*
    - Escolher linhas vazias em vez de ocupadas. Só no arranque?
  - Sem *dirty*
    - As linhas não alteradas podem ser removidas. Será justo?
  - *Least-Frequently-Used* (LFU)
    - Escolher a linha usada menos vezes. Desde quando?
  - *Least-Recently-Used* (LRU)
    - Escolher a usada há mais tempo. Como implementar?
  - Random
    - Escolher aleatoriamente tem custo reduzido.

Estratégia de write

- **Write-through**

O bloco é escrito quando a linha for alterada.
A escrita do bloco é realizada em paralelo.

  - Não usa bit dirty.
  - Tempo de miss reduzido.
  - A memória é usada durante mais tempo.
  - Mais conflitos no acesso por DMA.

- **Write-back**

O bloco só é escrito quando a linha for escolhida para substituição.

  - Necessita bit dirty.
  - Tempo de Miss mais elevado.
  - Minimiza a utilização da memória.
  - Menos conflitos no acesso por DMA.