Instituto de Matemática
Departamento de Ciência da Computação

Arquitetura de Computadores

Memória cache

Prof. Marcos E Barreto

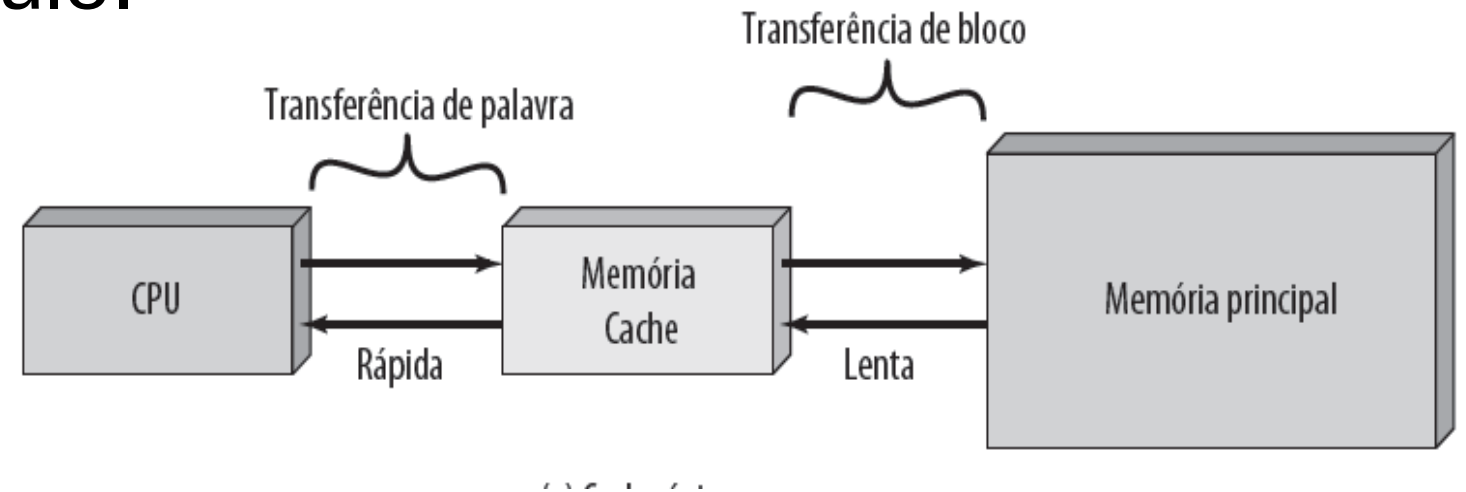
Tópicos

- Princípios da memória cache
- Elementos de projeto
- Estudo de casos
  - Pentium e ARM

- Referência:
  - Stallings. Arquitetura e organização de computadores. Cap. 4.

Memória cache

- Pequena quantidade de memória rápida.
- Fica entre a memória principal normal e a CPU.
- Pode estar localizada no chip da CPU ou módulo.

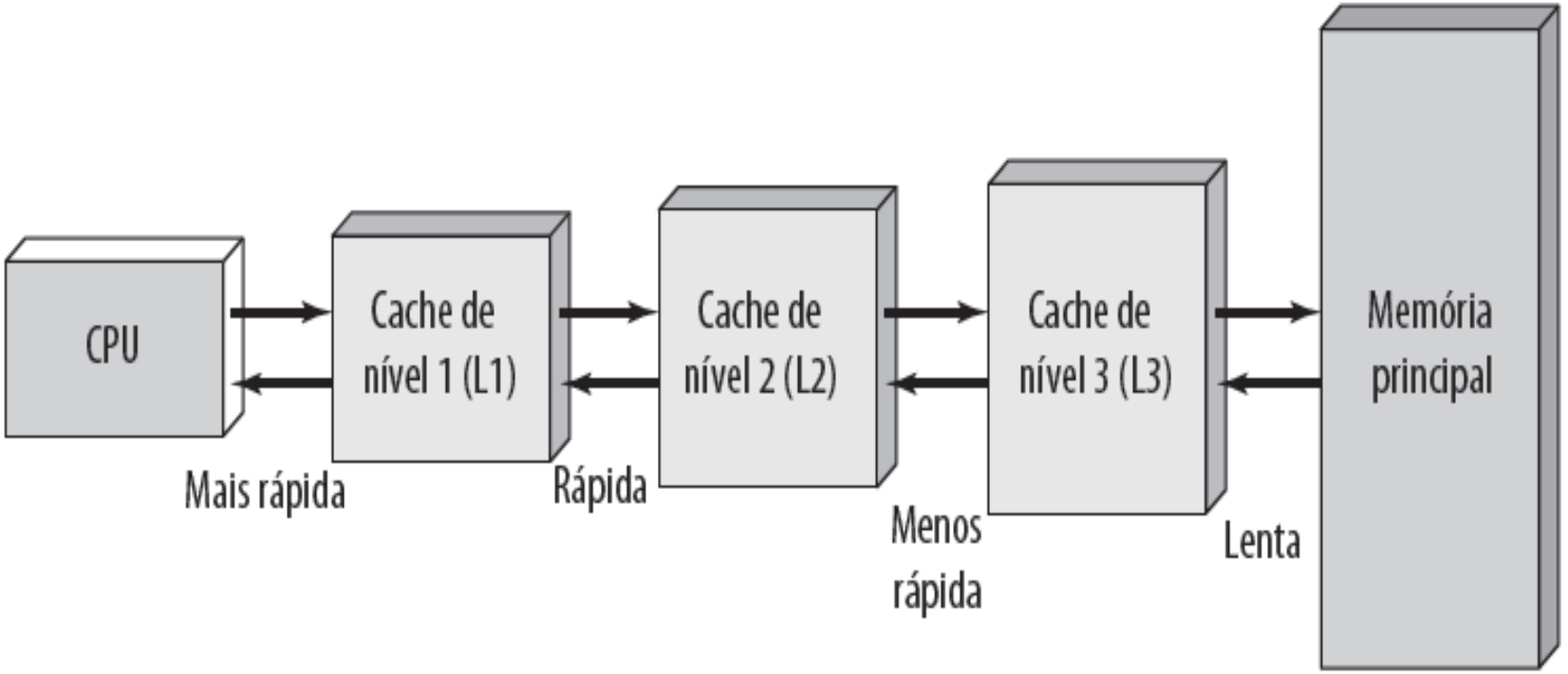


Conceito importante relacionado:

localidade de referência

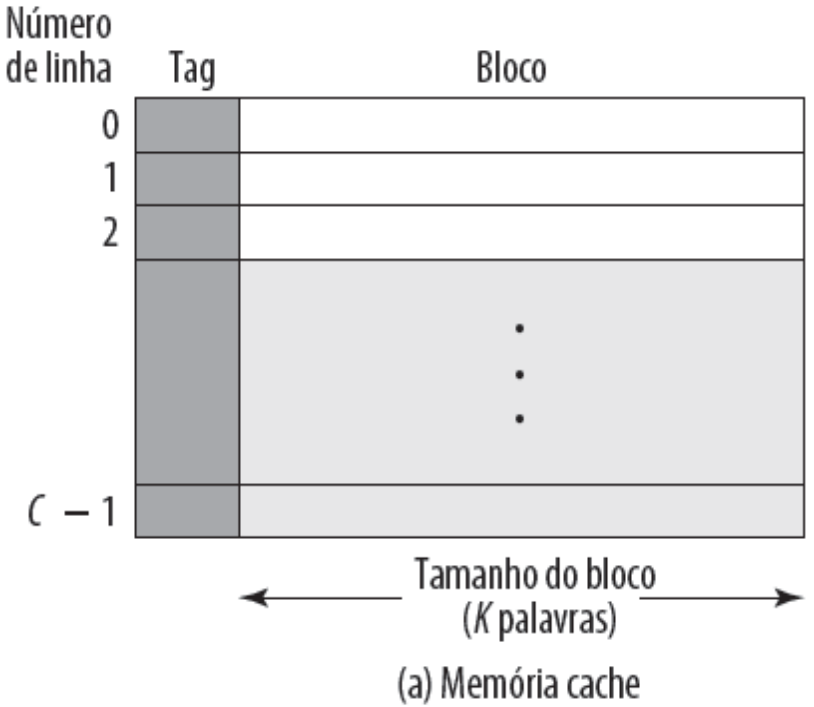
Memória cache

- Múltiplos níveis de cache

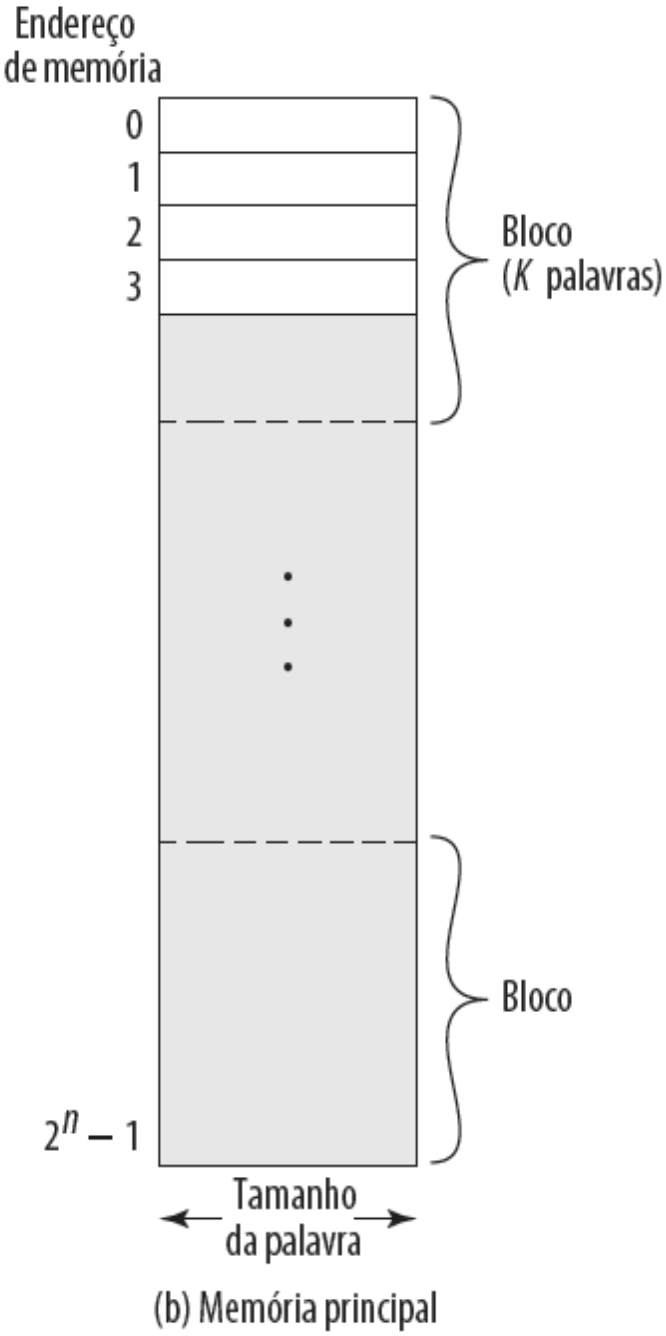


(b) Organização de cache em três níveis

Memória cache



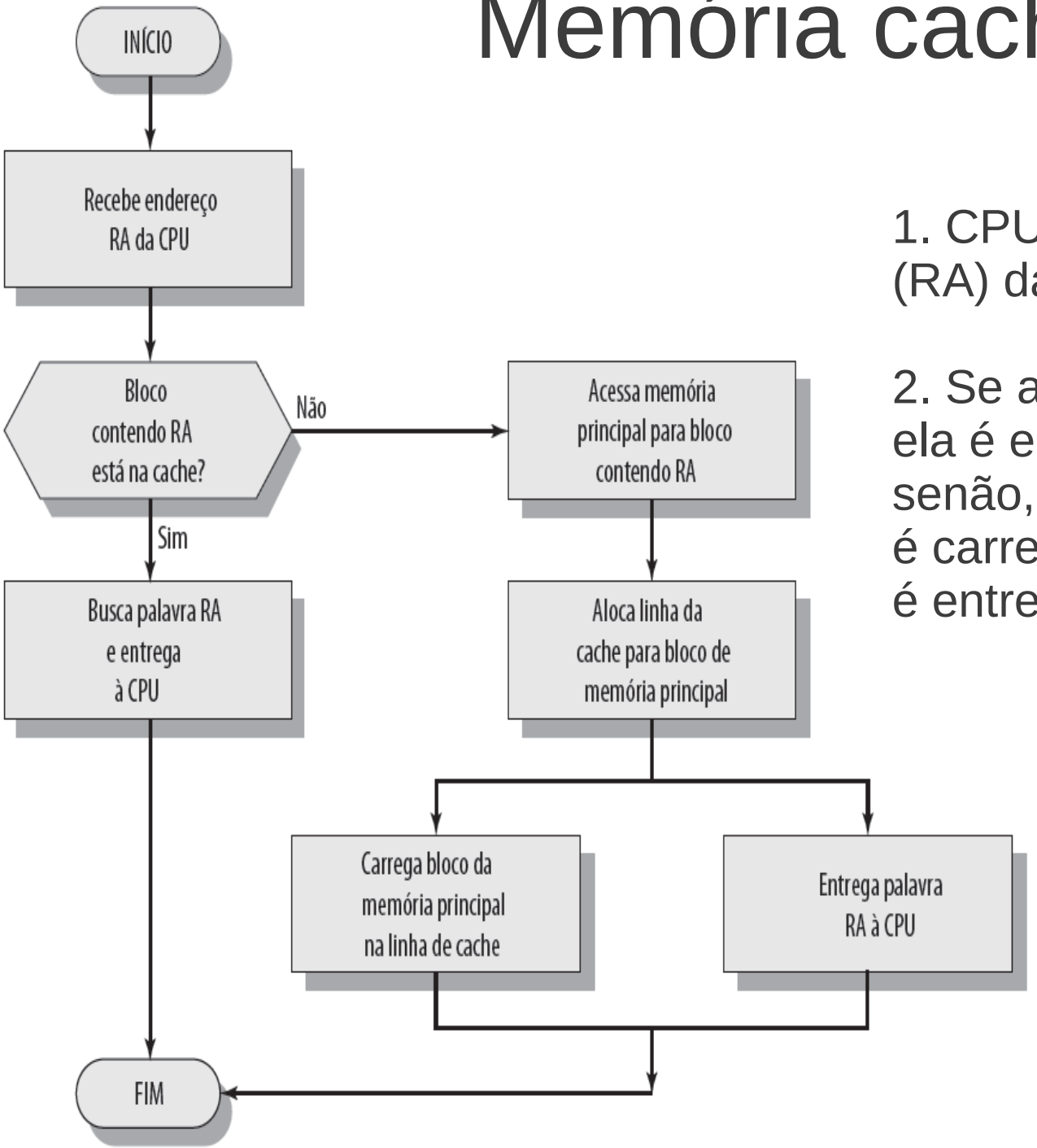
(a) Memória cache



(b) Memória principal

- Organização cache / MP
  - MP tem M = 2^n/K blocos
  - Cache tem m blocos (linhas)
  - Como m << M, deve haver um tag que que indique qual bloco da MP está armazenado na linha da cache.

Memória cache



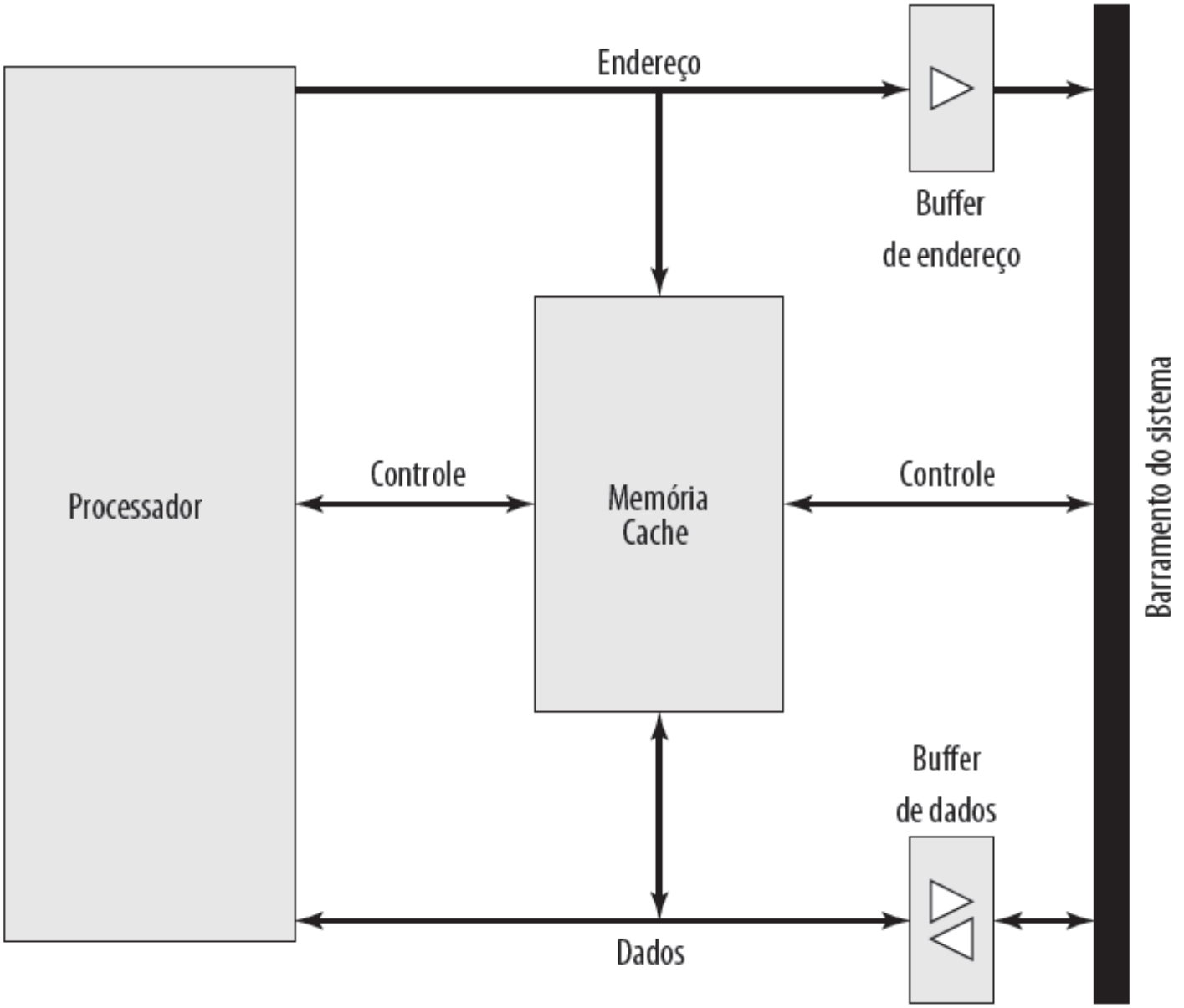
1. CPU gera endereço de leitura (RA) da palavra a ser lida.

2. Se a palavra estiver na cache, ela é entregue ao processador; senão, o bloco contendo a palavra é carregado na cache e a palavra é entregue à CPU.

Memória cache

Acerto de cache
(cache hit)

Falha de cache
(cache miss)



Elementos de projeto

- Embora haja um grande número de implementações de memória cache, existem alguns elementos básicos de projeto usados para classificar as memória cache.

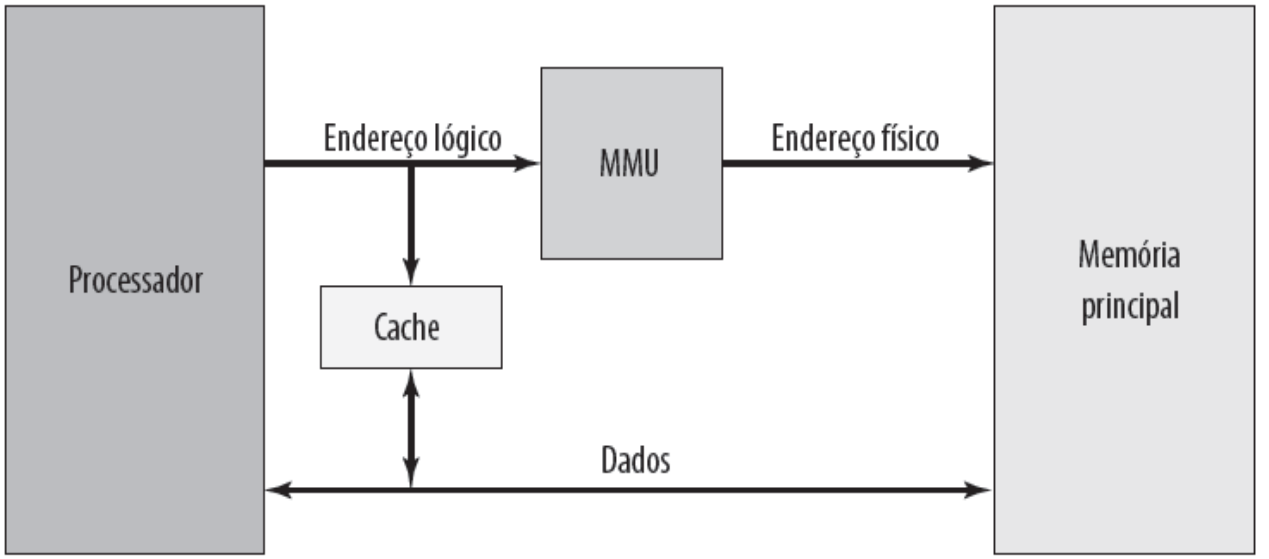
Endereços de cache	**Política de escrita**
Lógicos	*Write-through*
Físicos	*Write-back*
Tamanho de cache	*Write once*
Função de mapeamento	**Tamanho de linha**
Direta	**Número de caches**
Associativa	Um ou dois níveis
Associativa em conjunto (*set associative*)	Unificada ou separada
Algoritmo de substituição	
Usado menos recentemente (LRU, do inglês *least recently used*)	
Primeiro a entrar, primeiro a sair (FIFO, do inglês *first-in-first-out*)	
Usado menos frequentemente (LFU, do inglês *least frequently used*)	
Aleatório	

Elementos de projeto

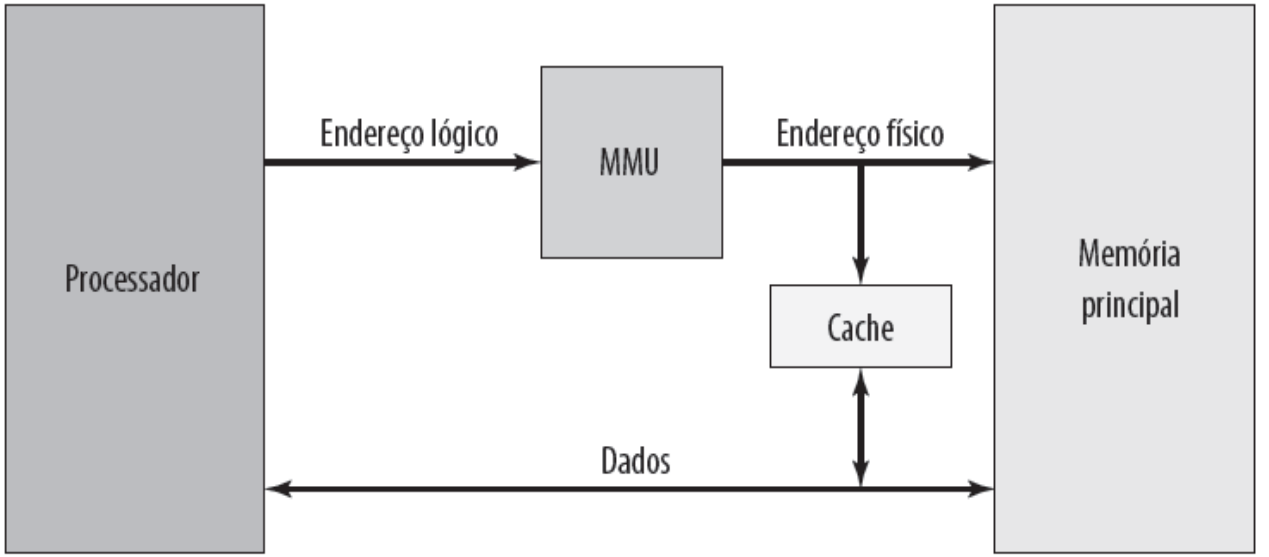
- **Endereços de cache**
  - **lógico/virtual X físico**
  - Relacionado ao conceito de memória virtual e ao emprego de MMU (*memory management unit*)
    - Memória virtual: programas endereçam a memória a partir de um ponto de vista lógico.
    - MMU: traduz endereços lógicos para endereços físicos.

Elementos de projeto

- **Cache lógica X cache física**



(a) Cache lógica



(b) Cache física

Elementos de projeto

- **Tamanho da cache**
  - IDEAL
    - Pequeno suficiente para que o custo médio por bit seja próximo do custo médio por bit da memória principal.
    - Grande suficiente para que o tempo médio de acesso seja satisfatório.
  - PRÁTICA
    - Por quê minimizar o tamanho da cache?
      - Caches grandes tendem a ser lentas
      - Área disponível no chip e na placa-mãe
    - Desempenho da cache é sensível à carga de trabalho
      - Impossível determinar um tamanho único (ideal) para cache

Tabela 4.3 Tamanhos de memória cache de alguns processadores

Processador	Tipo	Ano de introdução	Cache L1[a]	Cache L2	Cache L3
IBM 360/85	Mainframe	1968	16 a 32 KB	—	—
PDP-11/70	Minicomputador	1975	1 KB	—	—
VAX 11/780	Minicomputador	1978	16 KB	—	—
IBM 3033	Mainframe	1978	64 KB	—	—
IBM 3090	Mainframe	1985	128 a 256 KB	—	—
Intel 80486	PC	1989	8 KB	—	—
Pentium	PC	1993	8 KB/8 KB	256 a 512 KB	—
PowerPC 601	PC	1993	32 KB	—	—
PowerPC 620	PC	1996	32 KB/32 KB	—	—
PowerPC G4	PC/servidor	1999	32 KB/32 KB	256 KB a 1 MB	2 MB
IBM S/390 G4	Mainframe	1997	32 KB	256 KB	2 MB
IBM S/390 G6	Mainframe	1999	256 KB	8 MB	—
Pentium 4	PC/servidor	2000	8 KB/8 KB	256 KB	—
IBM SP	Servidor avançado/ Supercomputador	2000	64 KB/32 KB	8 MB	—
CRAY MTA[b]	Supercomputador	2000	8 KB	2 MB	—
Itanium	PC/servidor	2001	16 KB/16 KB	96 KB	4 MB
SGI Origin 2001	Servidor avançado	2001	32 KB/32 KB	4 MB	—
Itanium 2	PC/servidor	2002	32 KB	256 KB	6 MB
IBM POWER5	Servidor avançado	2003	64 KB	1,9 MB	36 MB
CRAY XD-1	Supercomputador	2004	64 KB/64 KB	1 MB	—
IBM POWER6	PC/servidor	2007	64 KB/64 KB	4 MB	32 MB
IBM z10	Mainframe	2008	64 KB/128 KB	3 MB	24 a 48 MB

a Dois valores separados por uma barra referem-se a caches de instrução e dados.

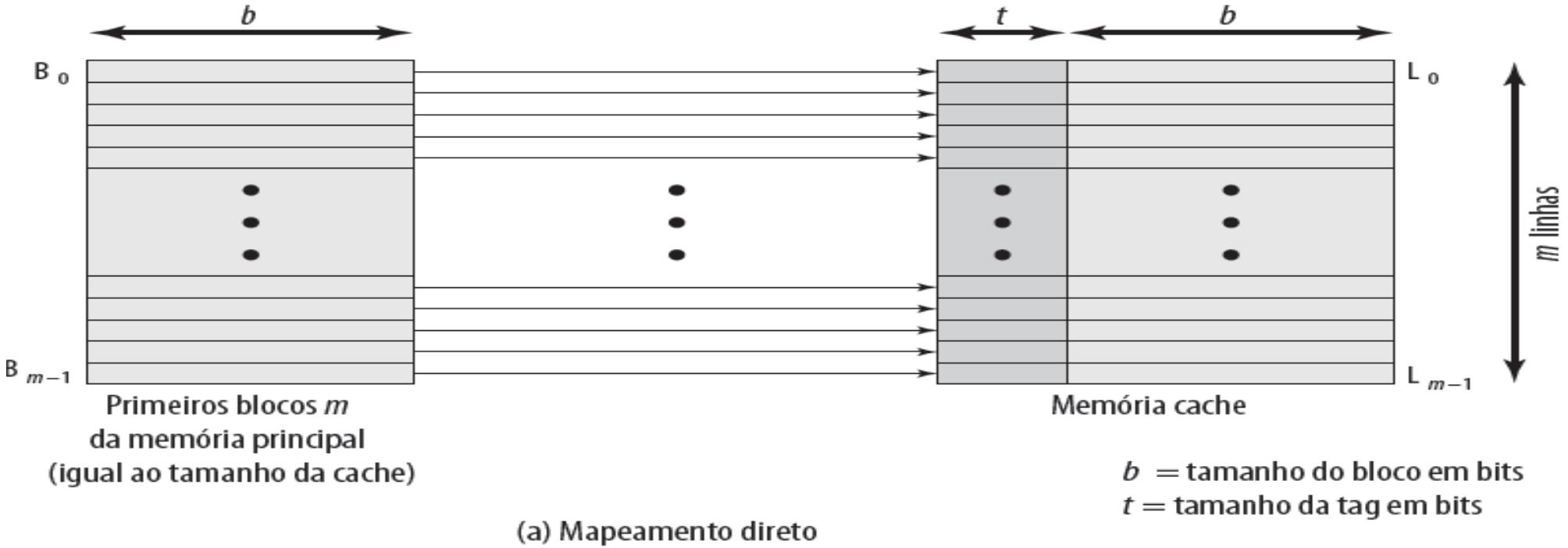
b As duas caches são apenas de instrução; não há caches de dados.

Elementos de projeto

- **Função de mapeamento**
  - Usada para mapear os blocos da memória principal às linhas da cache, uma vez que existem menos linhas de cache do que blocos de memória principal.
  - Determina qual bloco da memória principal ocupa uma determinada linha da cache.
  - Três técnicas:
    - Mapeamento direto
    - Mapeamento associativo
    - Mapeamento associativo em conjunto

Elementos de projeto

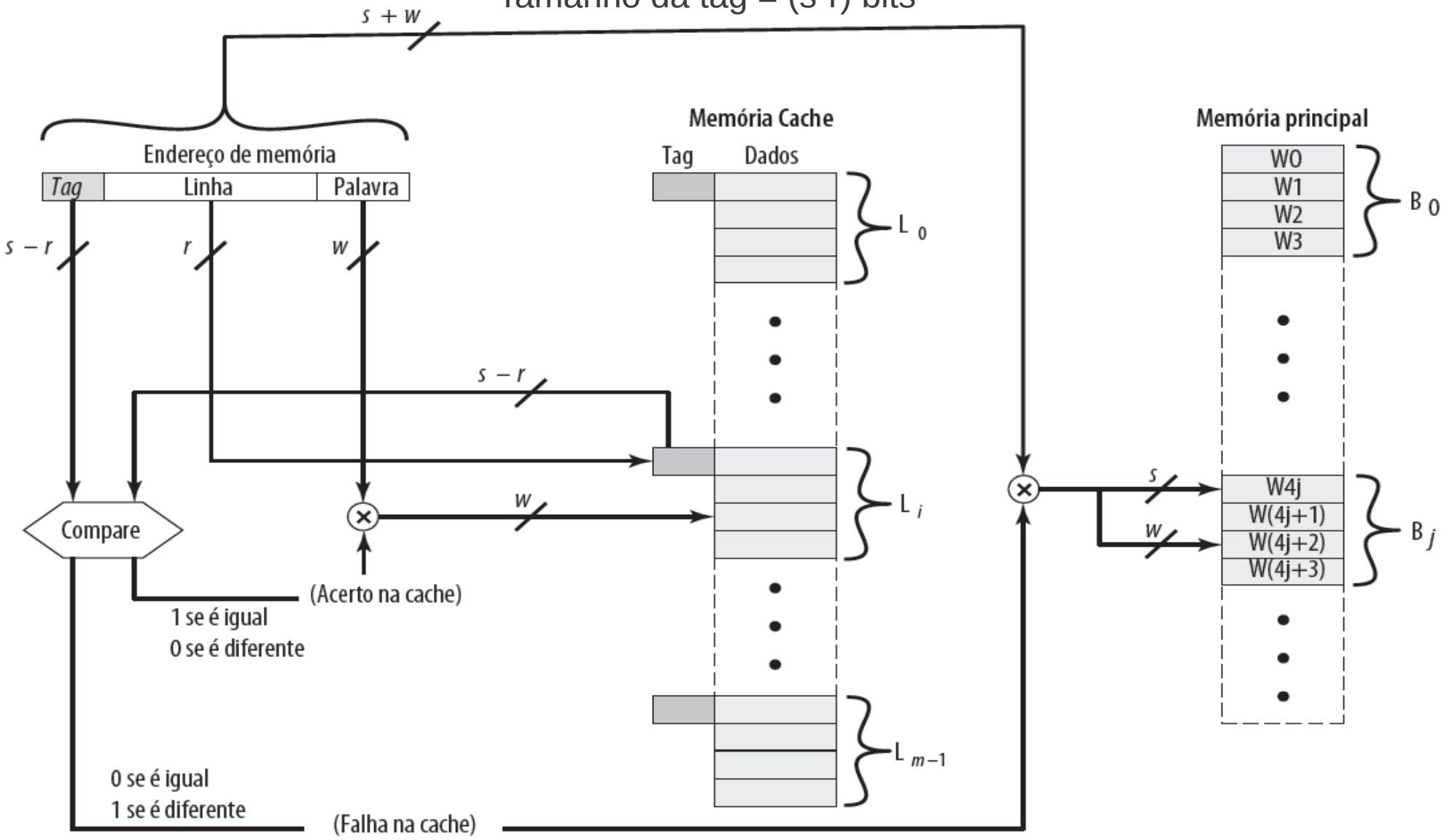
- **Mapeamento direto**
  - Cada bloco da memória principal é mapeado a apenas uma linha possível da cache
  - i = j módulo m
    - i = número de linhas da cache
    - j = número do bloco na memória principal
    - m = número de linhas da cache



(a) Mapeamento direto

Mecanismo geral de implementação do mapeamento direto

Tamanho do endereço = (s+w) bits
Número de unidades endereçáveis = 2^{s+w} palavras ou bytes
Tamanho do bloco = tamanho da linha = 2^{w} palavras ou bytes
Número de blocos na memória principal = 2^{s}
Número de linhas na cache = 2r
Tamanho da cache = 2^{r+w} palavras ou bytes
Tamanho da tag = (s-r) bits



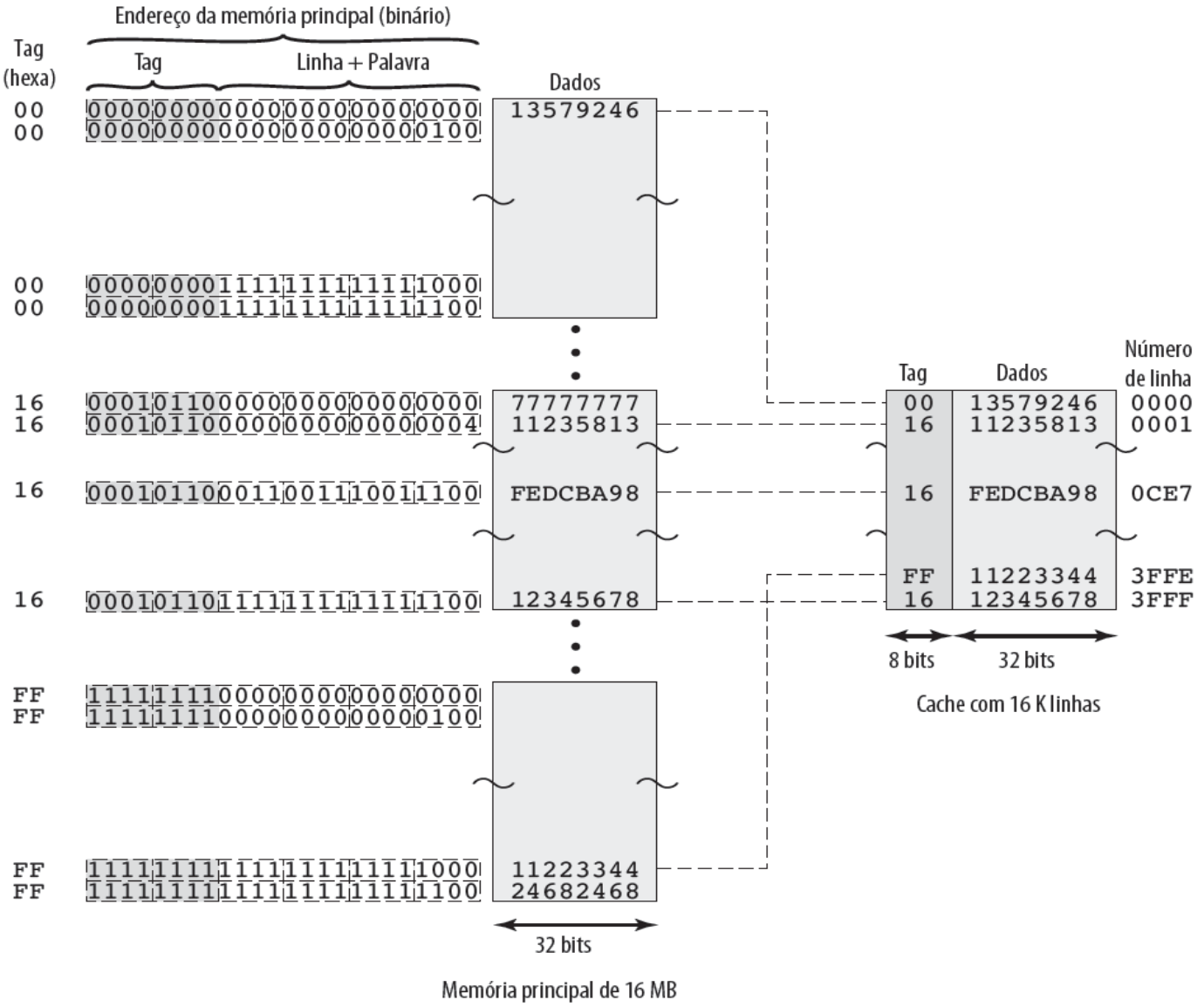
Mecanismo geral de implementação do mapeamento direto

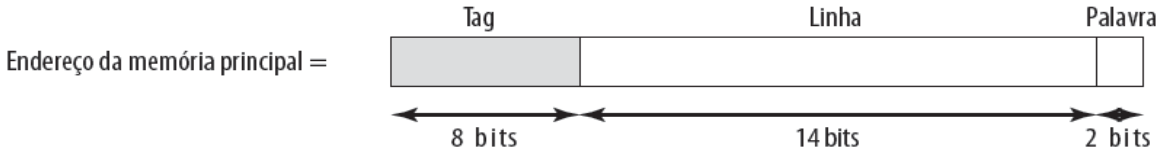
- O uso de uma parte do endereço como número da linha garante um mapeamento exclusivo de cada bloco à uma linha da cache.

Tabela 4.4 Mapeamento dos blocos da memória principal nas linhas da cache

Linha de cache	Blocos de memória principal mapeados
0	$0, m, 2m, \ldots, 2^s - m$
1	$1, m+1, 2m+1, \ldots, 2^s - m + 1$
⋮	⋮
$m - 1$	$m - 1, 2m - 1, 3m - 1, \ldots, 2^s - 1$

Exemplo de mapeamento direto





Nota: Valores de endereço de memória estão em binário; outros valores em hexadecimal.

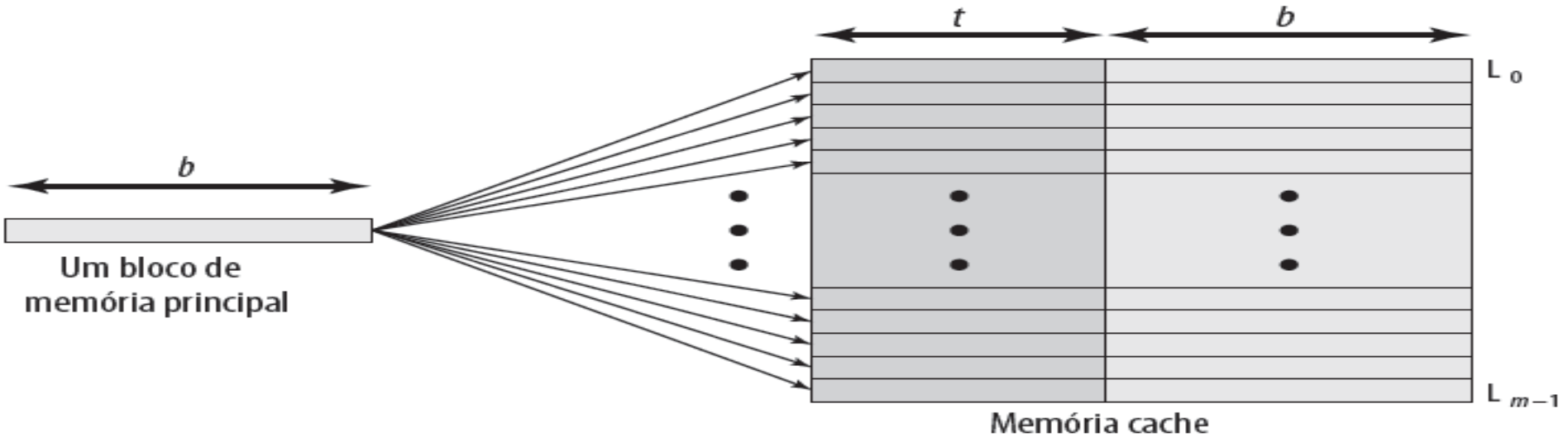
Elementos de projeto

- **Problema do mapeamento direto**
  - Problema: **thrashing**
    - Existe um local fixo na cache para cada bloco.
    - Se um programa referenciar repetidamente palavras alocadas em dois blocos que são mapeados para a mesma linha da cache, haverá muitas trocas de blocos de MP para a linha da cache, diminuindo a eficiência e o desempenho.
  - Solução: **victim cache**
    - Cache associativa, de tamanho entre 4 e 16 linhas de cache, colocada entre a cache e a memória principal
    - Armazena dados que foram descartados recentemente e que podem ser lidos novamente, com custo baixíssimo.

      => ver material sobre *victim cache* (Apêndice D Livro Stallings)

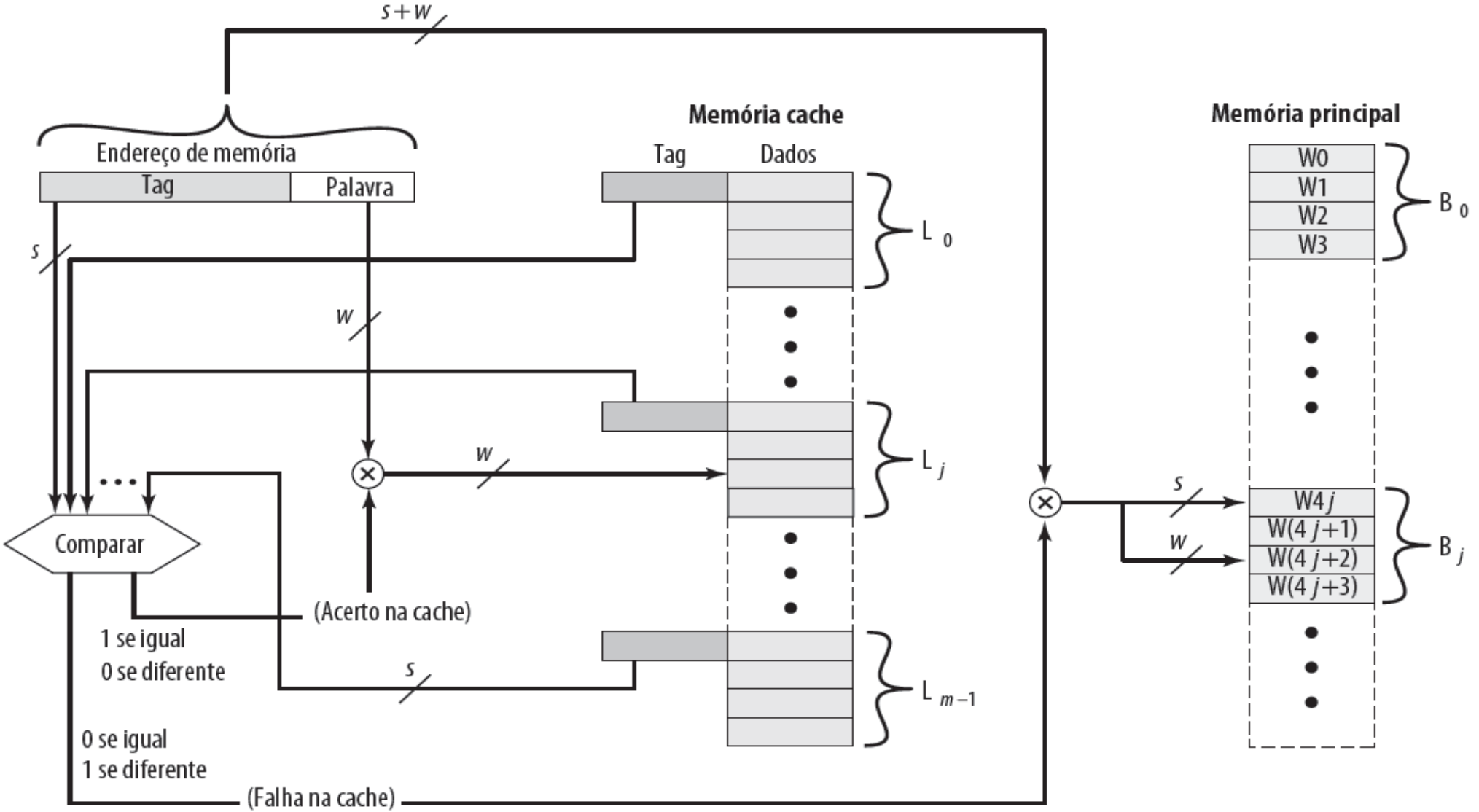
Elementos de projeto

- **Mapeamento associativo**
  - Permite que um bloco da memória principal seja mapeado para qualquer linha da cache.
  - Compensa a desvantagem do mapeamento direto.

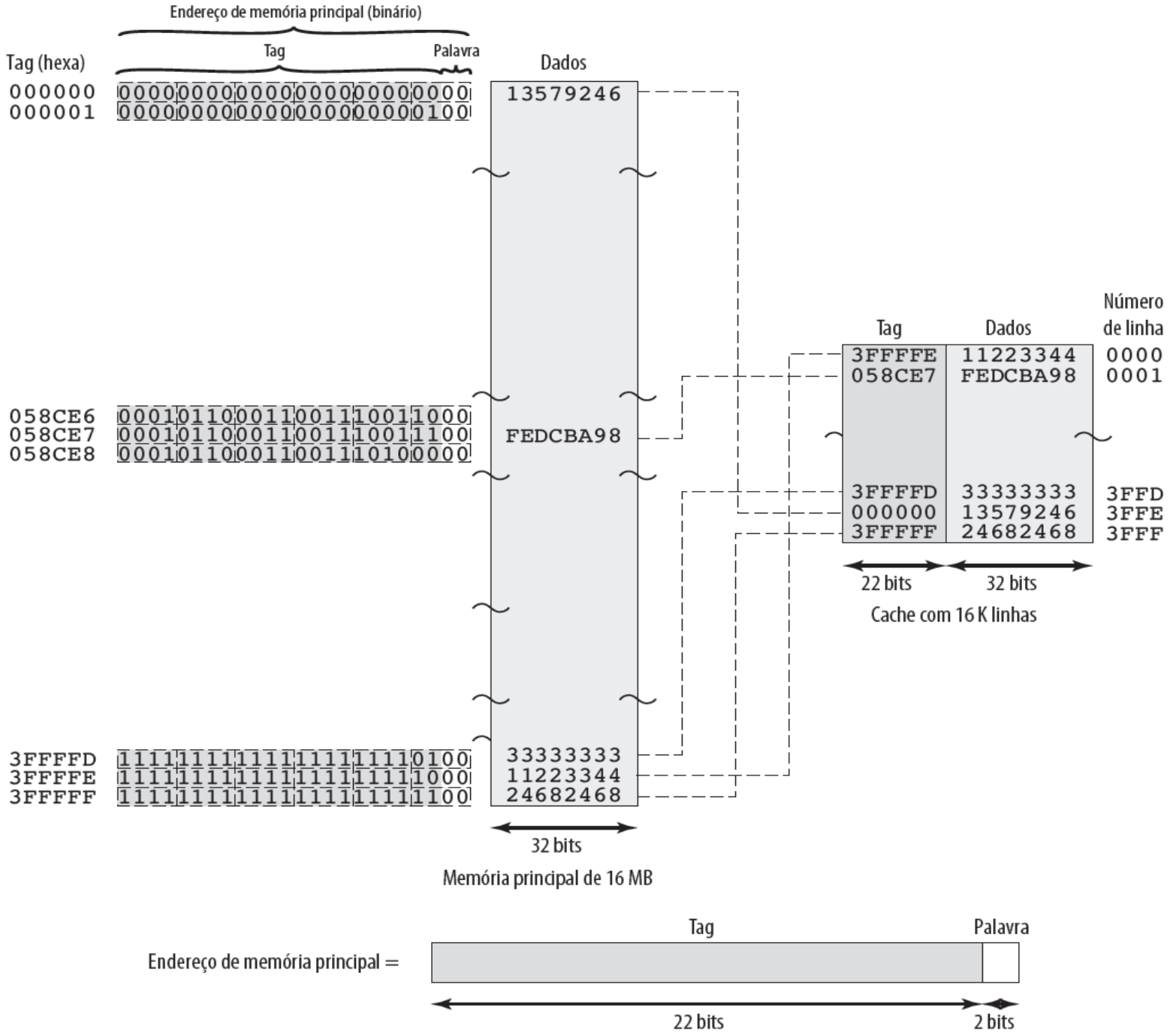


Mecanismo geral de implementação do mapeamento associativo

Tamanho do endereço = (s+w) bits
Número de unidades endereçáveis = 2^{s+w} palavras ou bytes
Tamanho do bloco = tamanho da linha = 2^{w} palavras ou bytes
Número de blocos na memória principal = 2^{s}
Número de linhas na cache = indeterminado
Tamanho da tag = s bits



Exemplo de mapeamento associativo



Nota: valores de endereço de memória em binário; outros, em hexadecimal

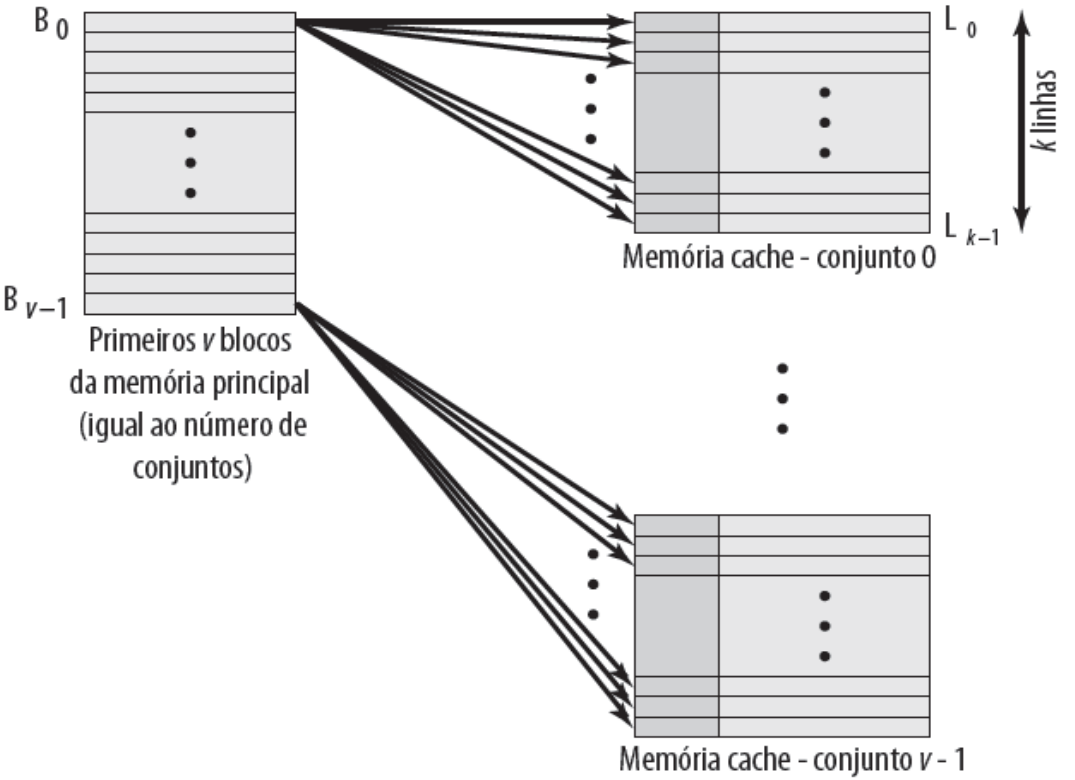
Elementos de projeto

- **Considerações sobre o mapeamento associativo**
  - Vantagem
    - Flexibilidade em relação a qual bloco substituir quando um novo bloco for lido para a cache.
  - Desvantagem
    - Complexidade do circuito necessário para comparar as tags em paralelo.

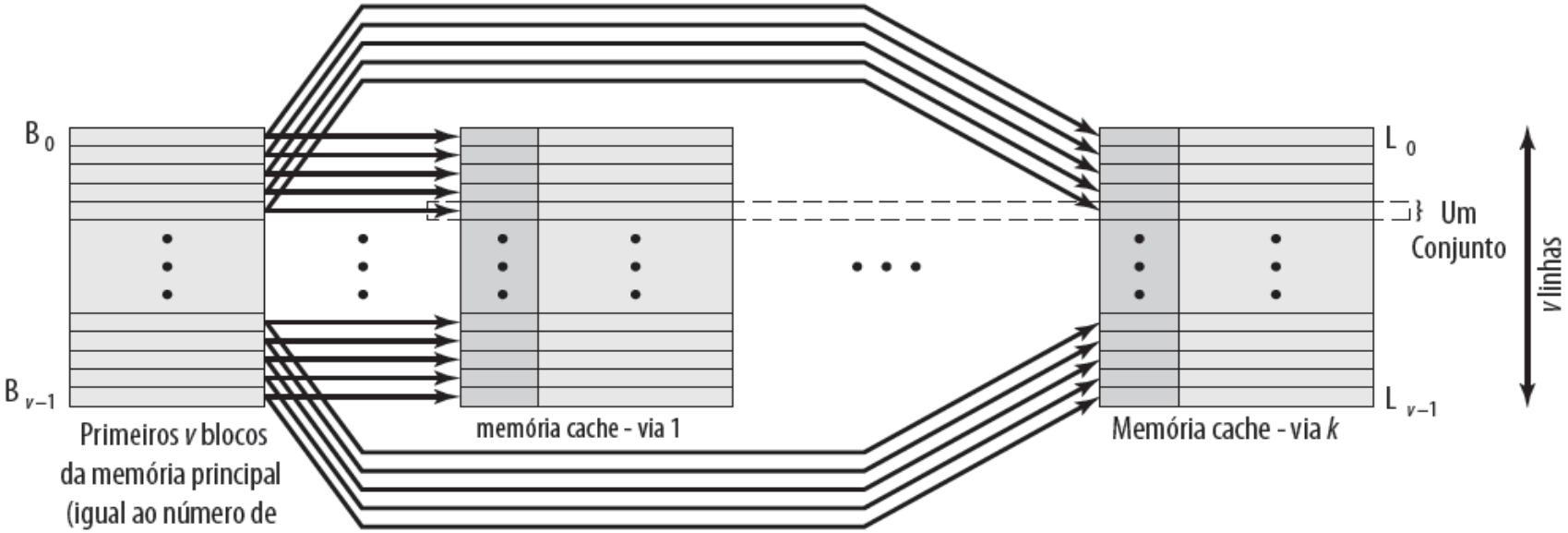
Elementos de projeto

- **Mapeamento associativo em conjunto**
  - Técnica intermediária entre mapeamento direto e associativo, que potencializa as vantagens e reduz as desvantagens.
  - Cache é uma série de conjuntos, cada um consistindo em uma série de linhas.
  - Relacionamentos
    - m = v * k e i = j módulo v onde
      - i = número do conjunto de cache
      - j = número de bloco da memória principal
      - m = número de linhas na cache
      - v = número de conjuntos
      - k = número de linhas em cada conjunto

- **Mapeamento associativo em conjunto**



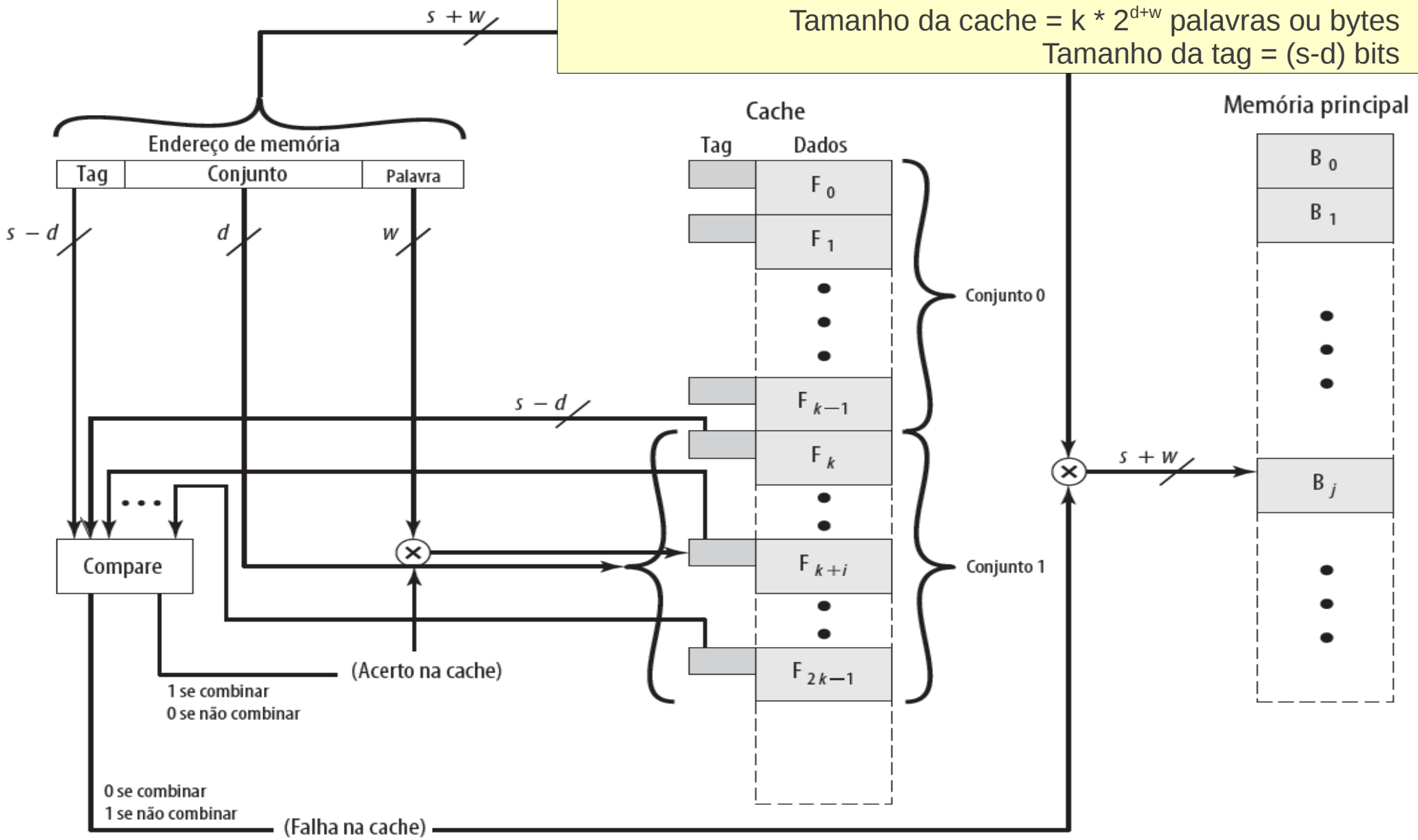
(a) *v* caches mapeadas associativas

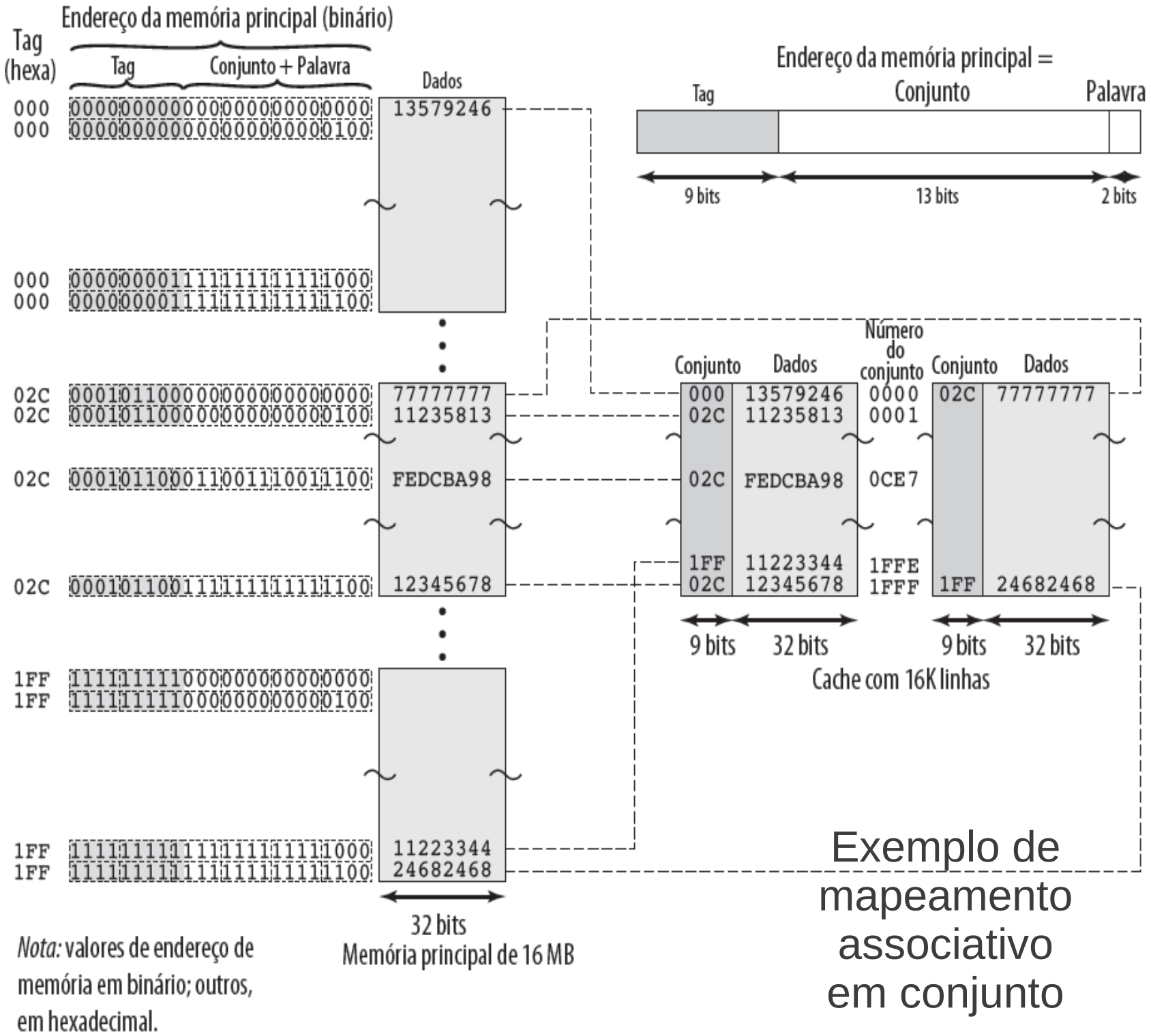


(b) *k* caches mapeadas diretamente

Implementação do mapeamento associativo em conjunto

Tamanho do endereço = (s+w) bits
Número de unidades endereçáveis = 2^{s+w} palavras ou bytes
Tamanho do bloco = tamanho da linha = 2^{w} palavras ou bytes
Número de blocos na memória principal = 2^{s}
Número de conjuntos = v = 2^{d}
Número de linhas na cache = m = kv = k * 2^{d}
Tamanho da cache = k * 2^{d+w} palavras ou bytes
Tamanho da tag = (s-d) bits





Exemplo de mapeamento associativo em conjunto

Elementos de projeto

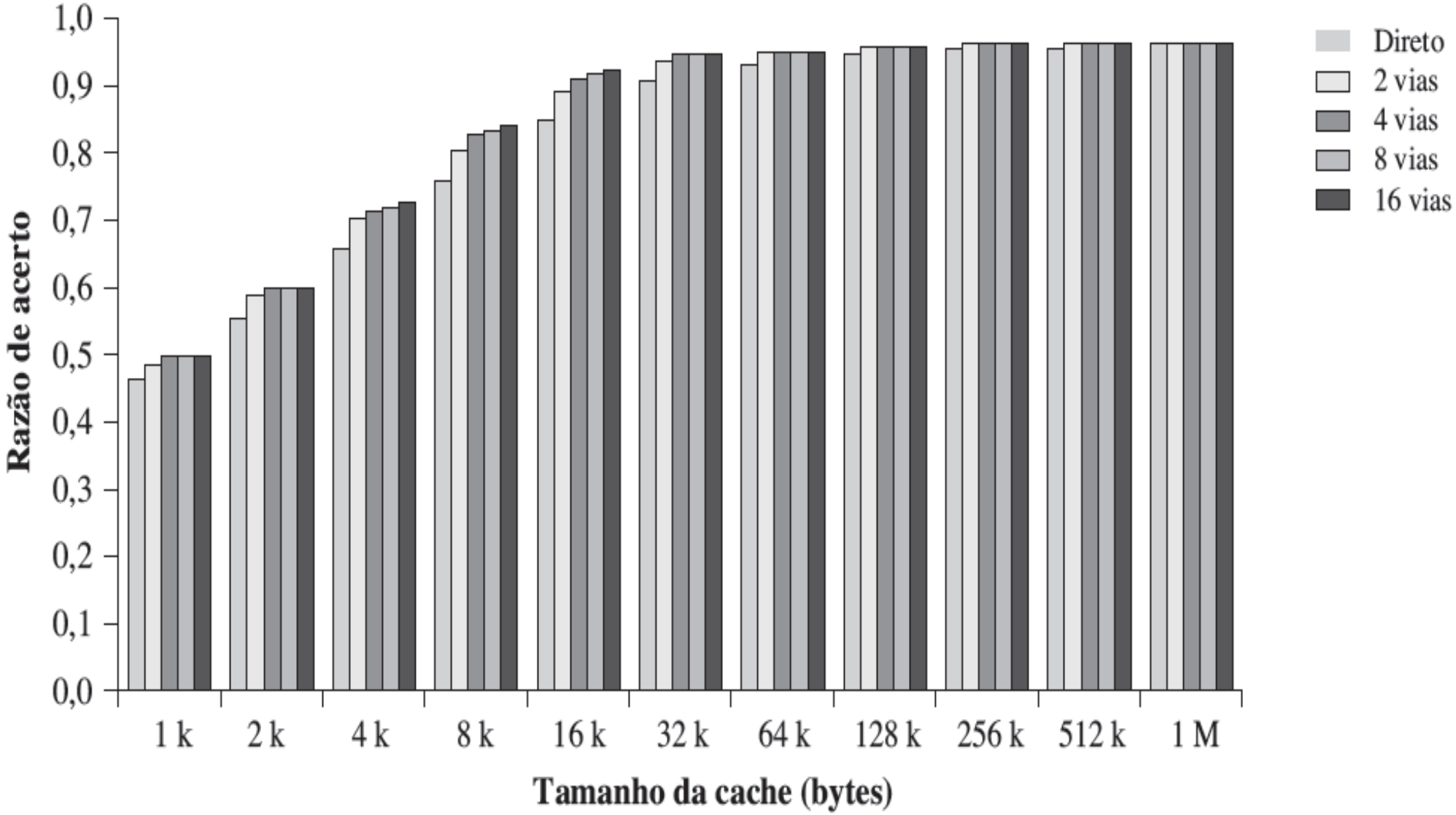
- **Considerações sobre o mapeamento associativo em conjunto**
  - Se v = m, k = 1 => mapeamento direto
  - Se v = 1, k = m => mapeamento associativo
  - Em geral, emprega-se duas linhas por conjunto

    (v = m/2, k = 2) como organização associativa em conjunto.
  - O uso de quatro linhas por conjunto (v = m/4, k = 4) provê uma melhoria adicional com custo relativamente baixo.

i = número do conjunto de cache
j = número de bloco da memória principal
m = número de linhas na cache
v = número de conjuntos
k = número de linhas em cada conjunto

Estudo de caso sobre desempenho de cache associativa

Simulação da execução de um compilador GCC.



Elementos de projeto

- **Algoritmos de substituição**
  - Usados para a substituição de blocos em cache, quando a cache estiver cheia e um novo bloco tiver que ser lido.
  - Se mapeamento direto, não há substituição!
  - Para mapeamentos associativos, existem várias alternativas de algoritmos, dentre os quais:
    - LRU (*least recently used*)
    - FIFO (*first-in, first-out*)
    - LFU (*least frequently used*)
    - Escolha de uma linha aleatória

Elementos de projeto

- **Política de escrita**
  - Como atualizar a memória principal com os blocos de dados que foram alterados na cache?
  - Escrita na memória X escrita na cache para a mesma palavra
  - Múltiplos processadores conectados a um barramento compartilhado, cada um com sua cache local.
  - Técnicas
    - Write-through
      - Escreve na cache e na memória
    - Write-back
      - Escreve somente na cache, marcando um bit (tag) na linha que foi modificada.

Elementos de projeto

- **Política de escrita (cont.)**
  - Outro problema ocorre quando um barramento é compartilhado por diversos dispositivos com cache local e que acessam uma mesma memória .

    => a alteração em uma palavra mapeada em uma das caches invalida a memória principal e também as outras caches.
  - Sistema pode impedir isso e prover coerência da cache:
    - Observação do barramento com write-through
    - Transparência do hardware
    - Memória não “cacheável”

Elementos de projeto

- **Tamanho da linha**
  - Quando um bloco de dados é recuperado e colocado na cache, não apenas a palavra desejada mas também algumas palavras adjacentes são armazenadas.
  - Razão de acerto X tamanho do bloco
    - Depende do princípio da localidade
  - Em geral:
    - Blocos maiores limitam o número de blocos que cabem em uma cache.
    - À medida que o bloco se torna maior, a nova palavra fica mais distante da palavra solicitada e, portanto, tem menos probabilidade de ser necessária num futuro próximo.

Elementos de projeto

- **Número de memórias cache**
  - Cache multinível
    - L1, L2 e L3
    - Atualmente, os três níveis são implementados no chip.
  - **Cache unificada X cache separada**
    - Uso de caches separadas para dados e endereços.