UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO
DEPARTAMENTO DE CIÊNCIAS EXATAS E NATURAIS
CURSO DE CIÊNCIA DA COMPUTAÇÃO

Arquitetura e Organização de Computadores

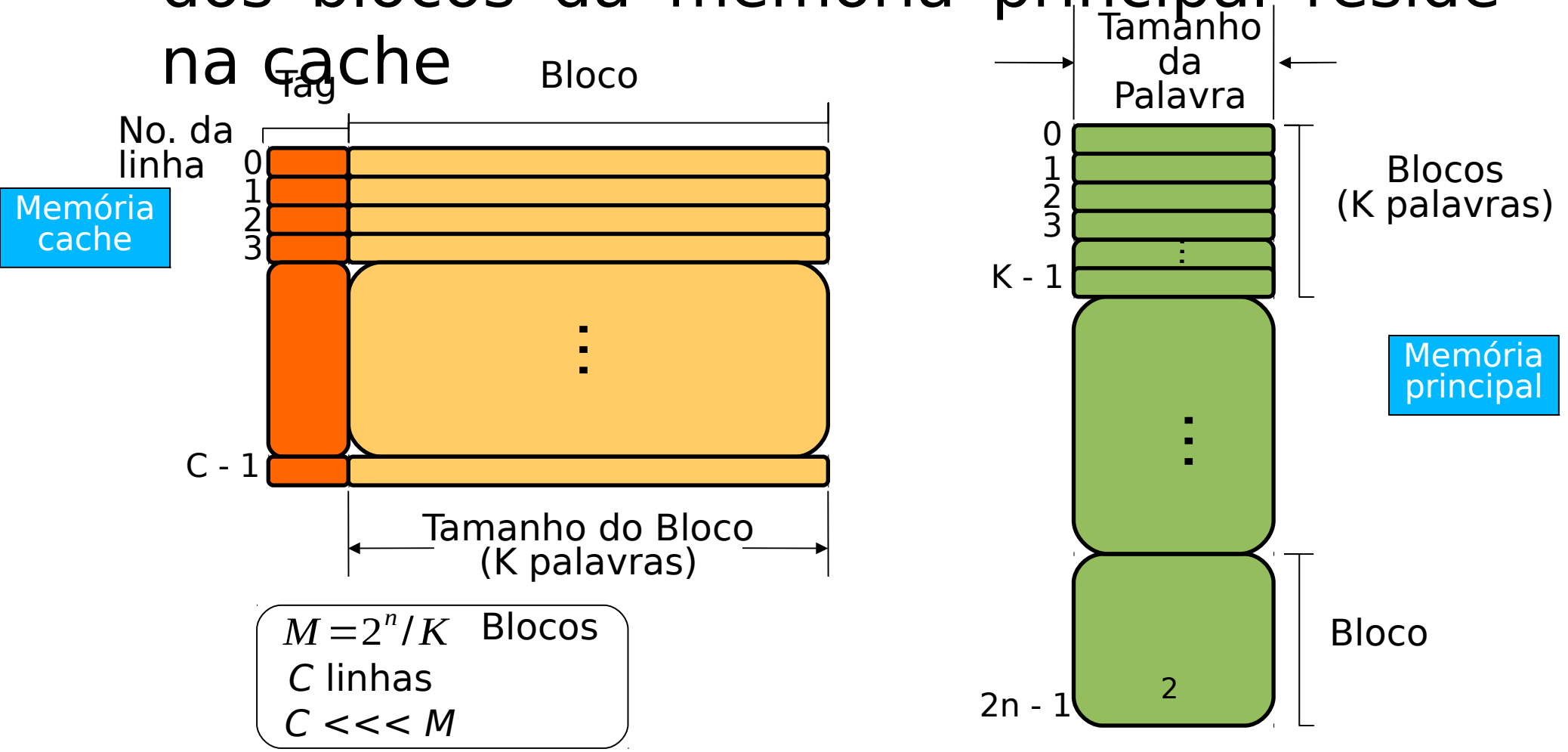
3- Sistemas de Memória Interna

Parte II

Prof. Sílvio Fernandes

Memória Cache

- Em qualquer instante, um subconjunto dos blocos da memória principal reside na cache



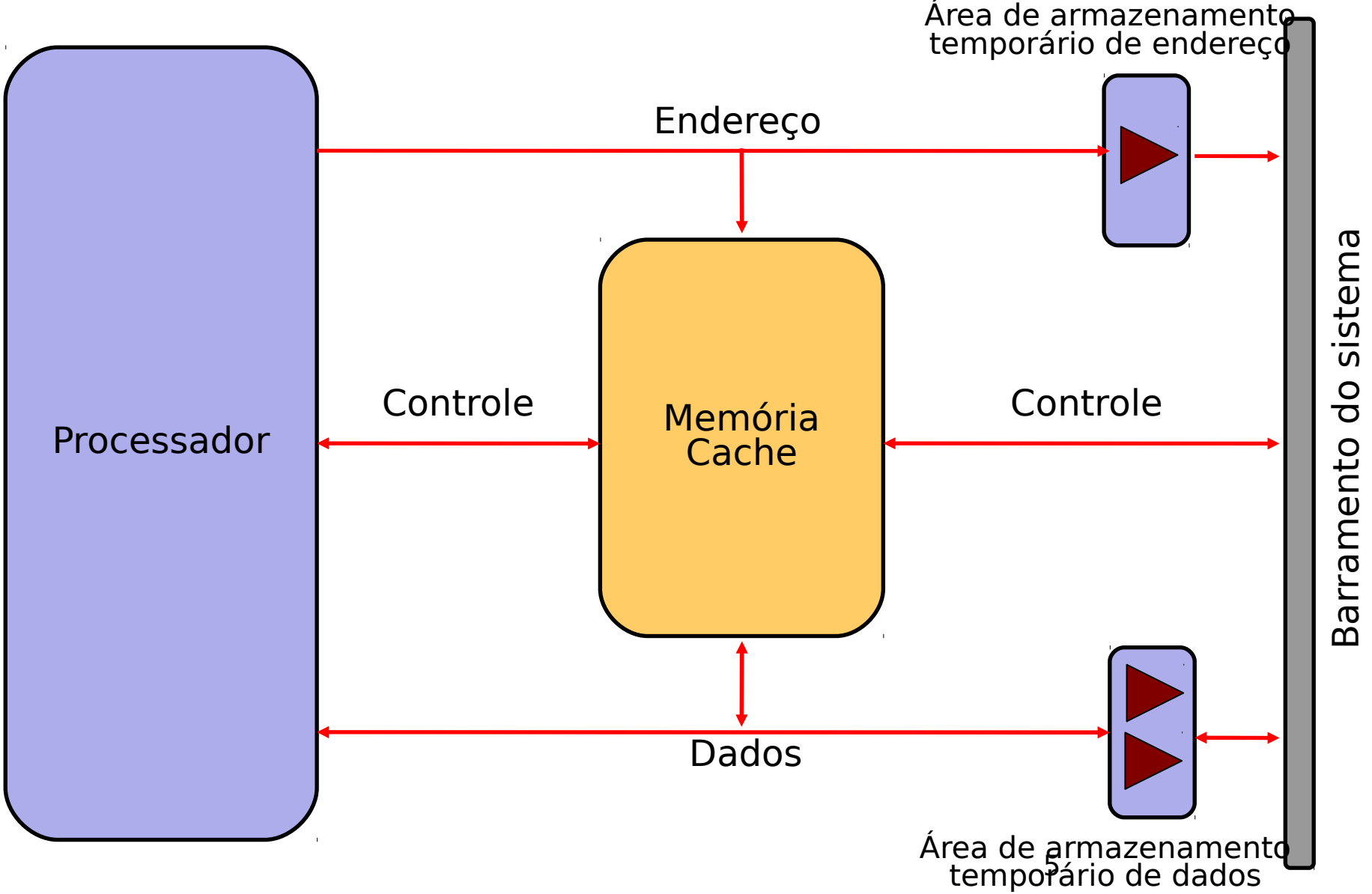
Memória Cache

- A mem. principal consiste em até 2n palavras endereçáveis, com cada palavra tendo um endereço distinto de **n** bits.
- Essa mem. é considerada como sendo uma série de blocos de tamanho fixo com **K** palavras cada (M blocos na memória principal)
  - M = 2n/K
- A cache consiste em **m** blocos, chamados **linhas**, cada uma contendo K palavras mais um tag de alguns bits

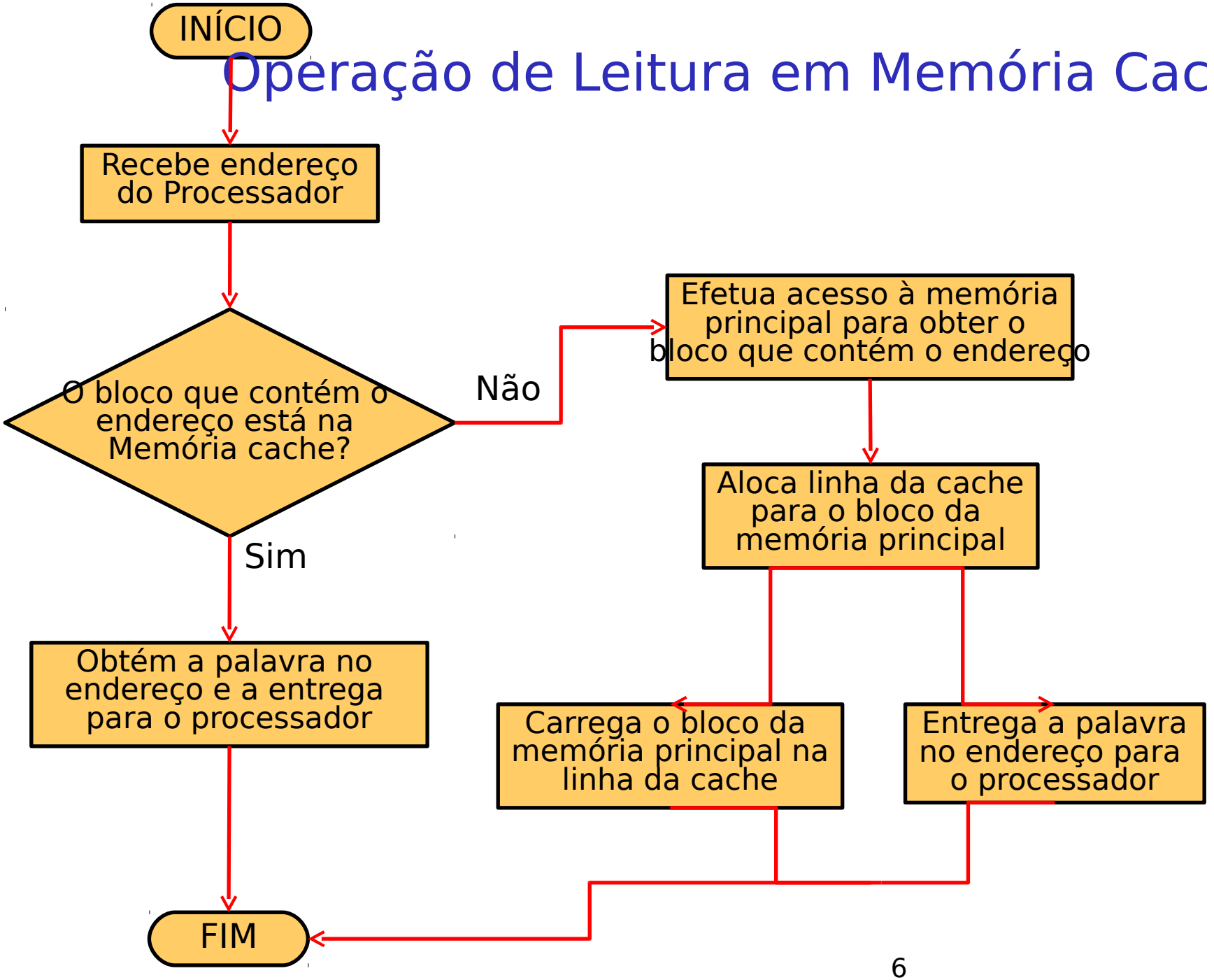
Memória Cache

- Cada linha contém K palavras, mais um tag de alguns bits
- A largura de uma linha, sem incluir tag e bits de controle, é o tamanho da linha
- A qualquer momento, algum subconjunto dos blocos de memória reside nas linhas na cache
- Se uma palavra em um bloco de memória for lida, esse bloco é transferido para uma das linhas da cache
- Cada linha inclui uma tag que identifica qual bloco em particular está armazenado

Memória Cache



Operação de Leitura em Memória Cache



Elementos do projeto da memória cache

Tabela 4.2 Elementos do projeto de 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	

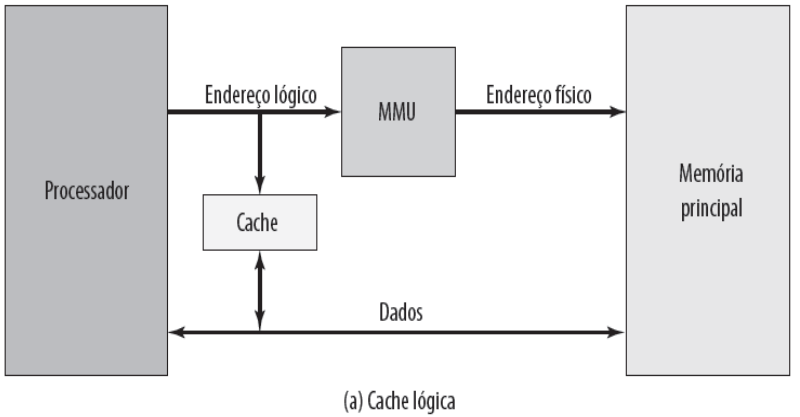
Elementos do projeto da memória cache

- Endereços de cache
  - Quase todos os processadores admitem memória virtual
    - Permitem que programas enderecem a memória a partir de um ponto de vista lógico, sem considerar a quantidade de memória principal disponível fisicamente
    - Os campos das instruções contém endereços virtuais
    - Para leitura e escrita, uma unidade de gerenciamento da memória (MMU – *memory management unit*) traduz cada endereço virtual para um endereço físico na mem. principal
    - O projetista escolhe colocar a cache entre o processador e a MMU ou entre a MMU e a memória principal

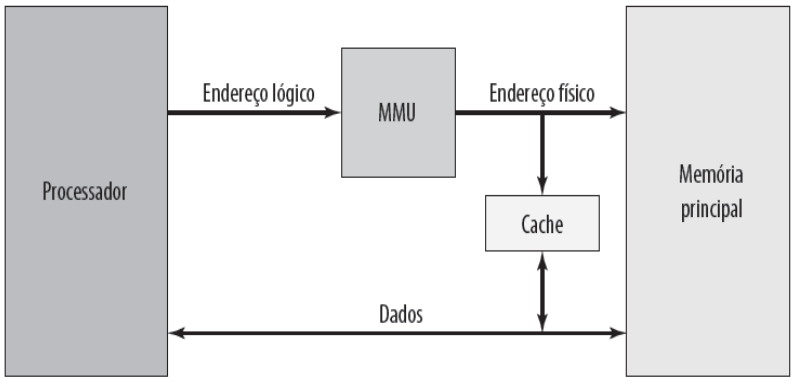
Elementos do projeto da memória cache

- Enderecos de cache

Figura 4.7 Caches lógicas e físicas



(a) Cache lógica



(b) Cache física

Elementos do projeto da memória cache

- Endereços de cache
  - Vantagem da cache lógica é a velocidade de acesso com relação a cache física
  - O processador acessa a cache direto sem tradução de endereços pela MMU
  - Desvantagem
  - Sistemas de mem. virtual oferecem o mesmo espaço de endereçamento de mem. virtual (cada aplicação começa no endereço 0)
  - A cache precisa ser completamente esvaziada a cada troca de contexto de aplicação

Elementos do projeto da memória cache

- Tamanho da memória cache
  - Deve ser grande para que o tempo médio de acesso à memória total seja próximo ao tempo de acesso da memória cache
  - Deve ser pequena para que o custo total por bit seja próximo do custo por bit da memória principal
  - Outros motivos para minimização da cache:
  - Quanto maior a cache, maior o número de pinos – e mais lento o endereçamento
  - O espaço limitado na placa de circuitos

Elementos do projeto da memória cache

- Tamanho da memória 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 do projeto da memória cache

- Tamanho da memória 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

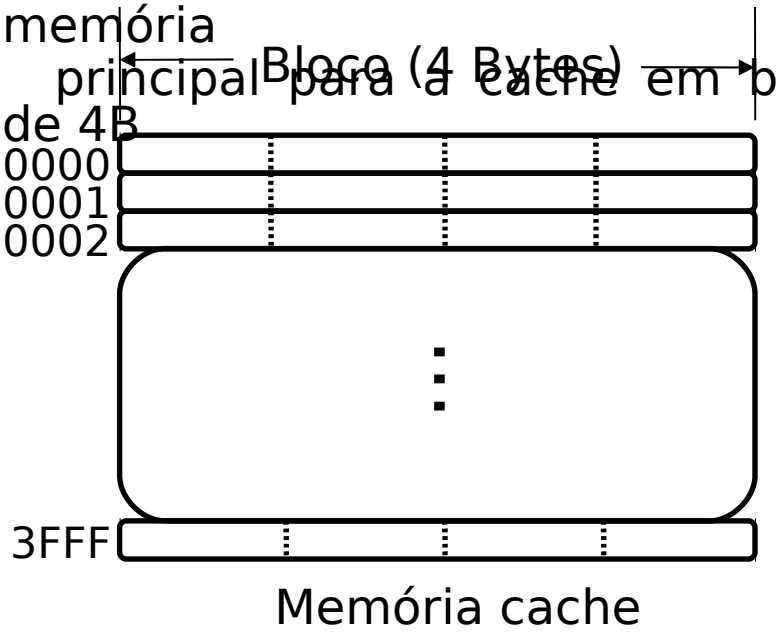
a Dois valores separados por uma barra referem-se a caches de instrução e dados.
b As duas caches são apenas de instrução; não há caches de dados.

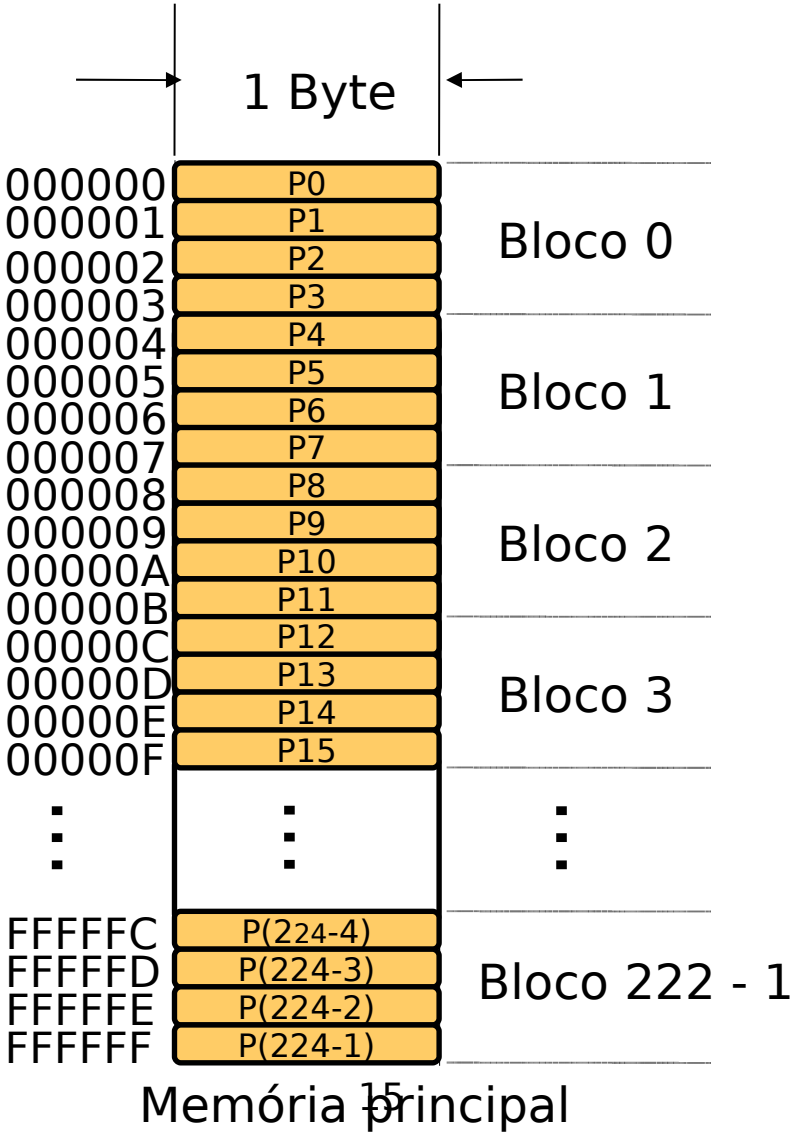
Elementos do projeto da memória cache

- Função de mapeamento
  - Necessidade
  - O número de linhas da cache é menor do que o número de blocos da memória principal
  - Técnicas utilizadas
  - Mapeamento Direto
  - Mapeamento Associativo
  - Mapeamento Associativo por Conjuntos
  - Exemplo a seguir

Elementos do projeto da memória cache

- Memória principal com 16 MB (224 B)
- Cada Byte é endereçável diretamente
- Memória principal pode ser vista como
  4M (222) blocos de 4 B cada
- Memória cache de 64 KB, organizada
  em 16 K (214) linhas de 4 B
- Os dados são transferidos da memória
  principal para a cache em blocos de 4B





Elementos do projeto da memória cache

- Função de mapeamento(Mapeamento Direto)
  - Cada bloco da memória principal é mapeado em uma única linha da cache
  - O mapeamento é expresso pela equação:

$i = j$ módulo m,

Em que:

i : número da linha da memória cache
j : número do bloco da memória principal
m : número de linhas da memória cache

Elementos do projeto da memória cache

- Função de mapeamento(Mapeamento Direto)
  - Segundo $i = j$ módulo m, cada bloco da memória principal é assim mapeado em

Linha da memória cache	Blocos da MP mapeados na linha
0	*0, m, 2m, ..., 2S - m*
1	*1, m + 1, 2m + 1, ..., 2S – m + 1*
⋮	⋮
m-1	*m – 1, 2m – 1, 3m – 1, ..., 2S - 1*

Elementos do projeto da memória cache

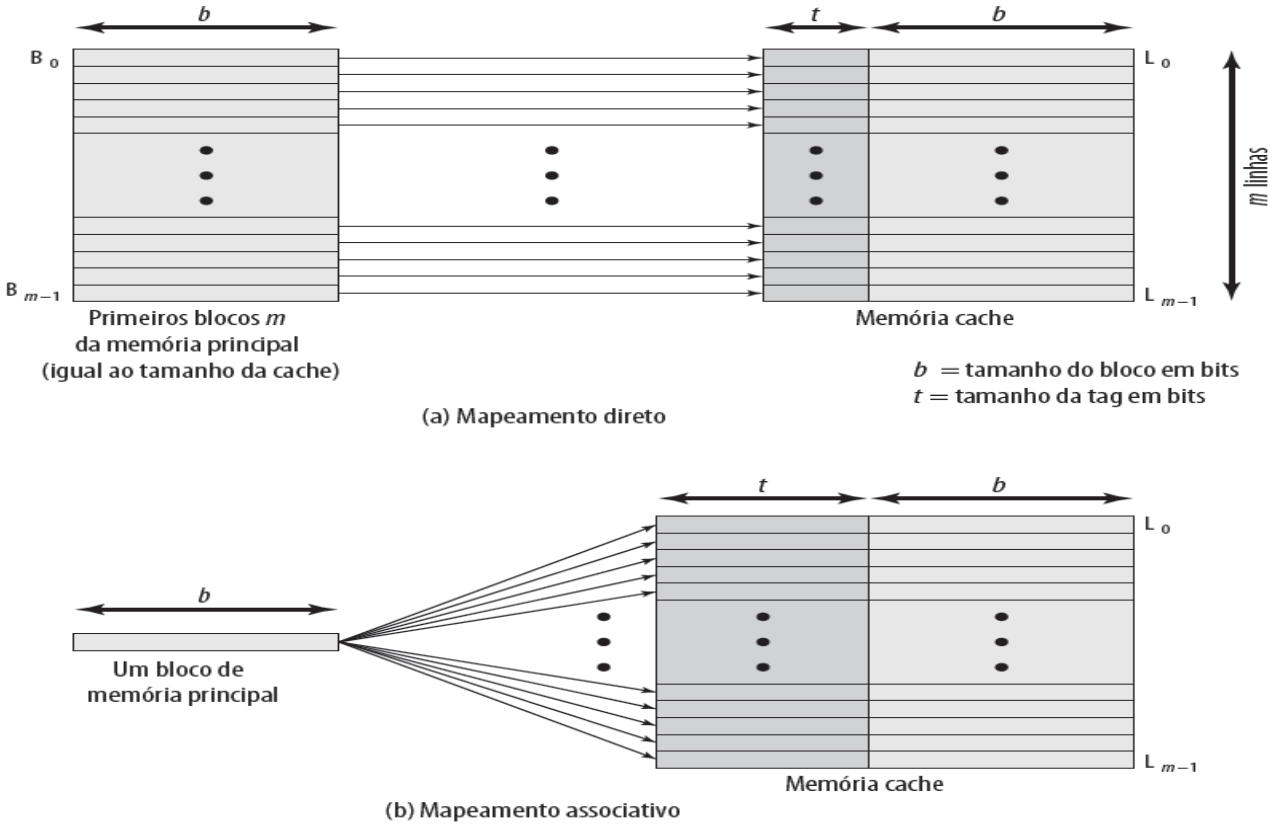
- Função de mapeamento(Mapeamento Direto)

Linha da memória cache	Blocos da MP mapeados na linha
0	000000, 010000, ..., FF0000
1	000004, 010004, ..., FF0004
⋮	⋮
214 - 1	00FFFC, 01FFFC, ..., FFFFFC

Elementos do projeto da memória cache

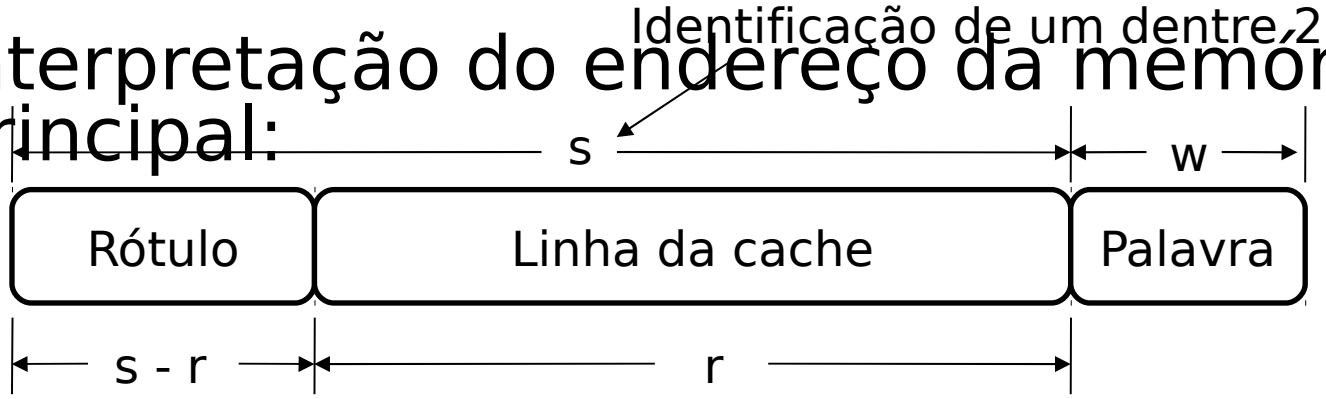
- Função de mapeamento(Mapeamento

Figura 4.8 Mapeamento da memória principal para a cache: direto e associativo



Elementos do projeto da memória cache

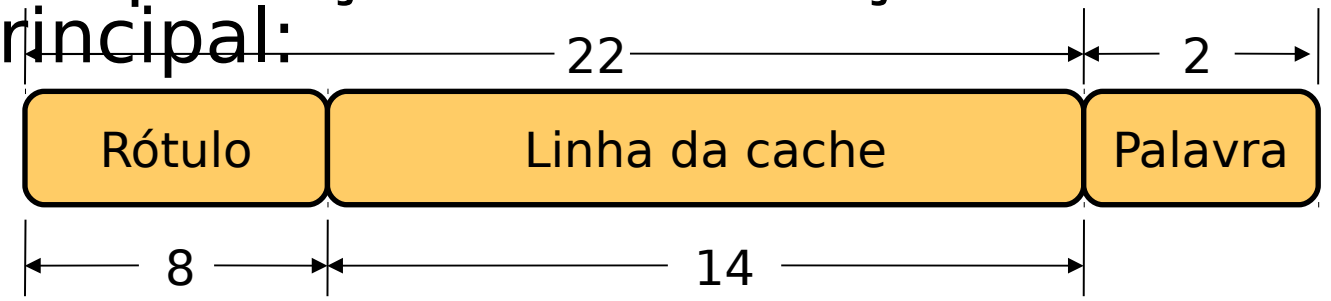
- Função de mapeamento(Mapeamento Direto)
  - Interpretação do endereço da memória principal:



  - No nosso exemplo
  - Se temos palavras de 22 bytes, logo **w = 2**
  - Se temos 222 blocos, precisamos de 22 bits para identificar o bloco, logo **s = 22**
  - Se temos 214 linhas de cache, precisamos de 14 bits, **r = 14**
  - Podemos dividir os 222 blocos da MP nas 2^{20} 214 linhas da cache, dando um total de 8 possibilidades,

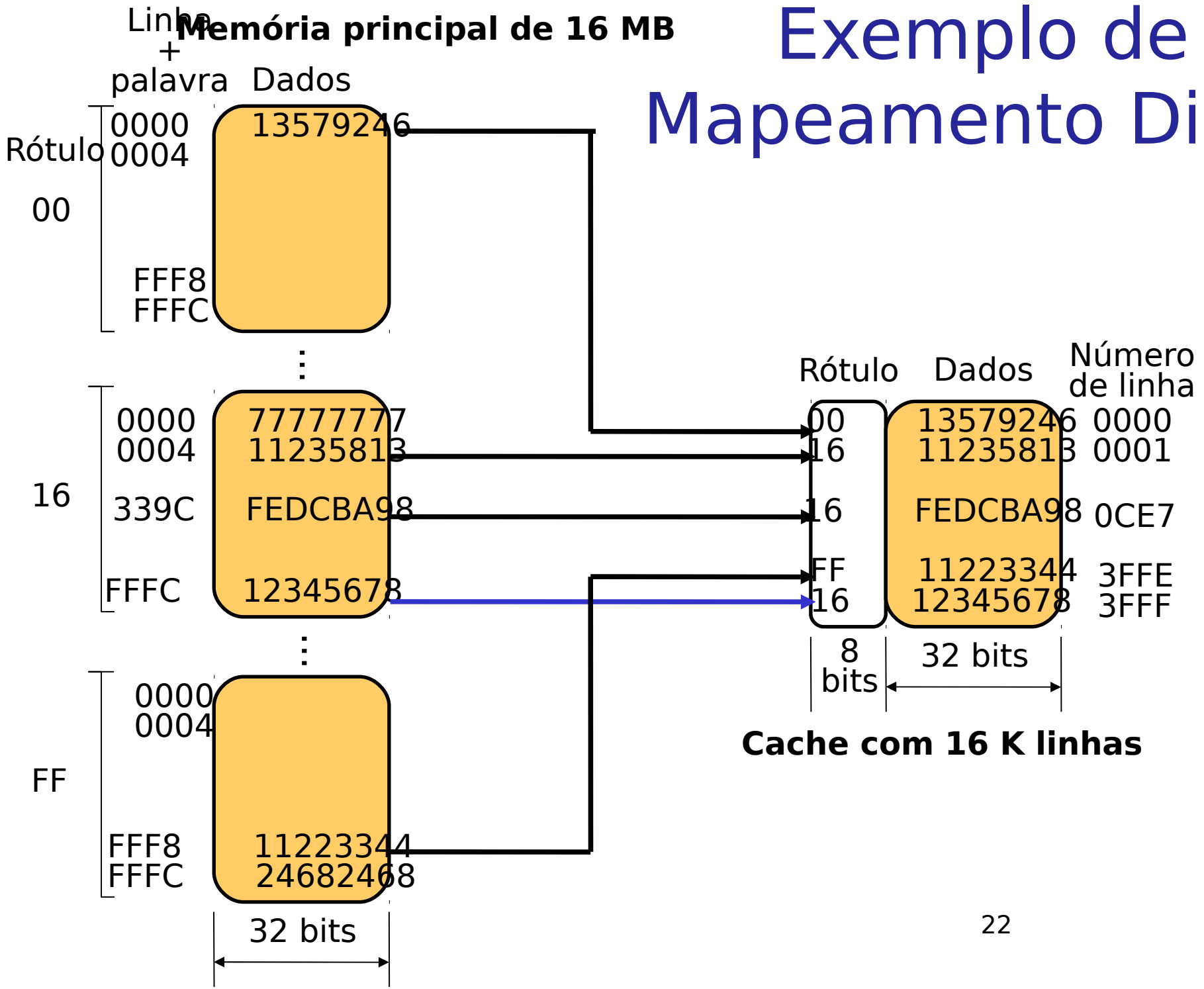
Elementos do projeto da memória cache

- Função de mapeamento(Mapeamento Direto)
  - Interpretação do endereço da memória principal:

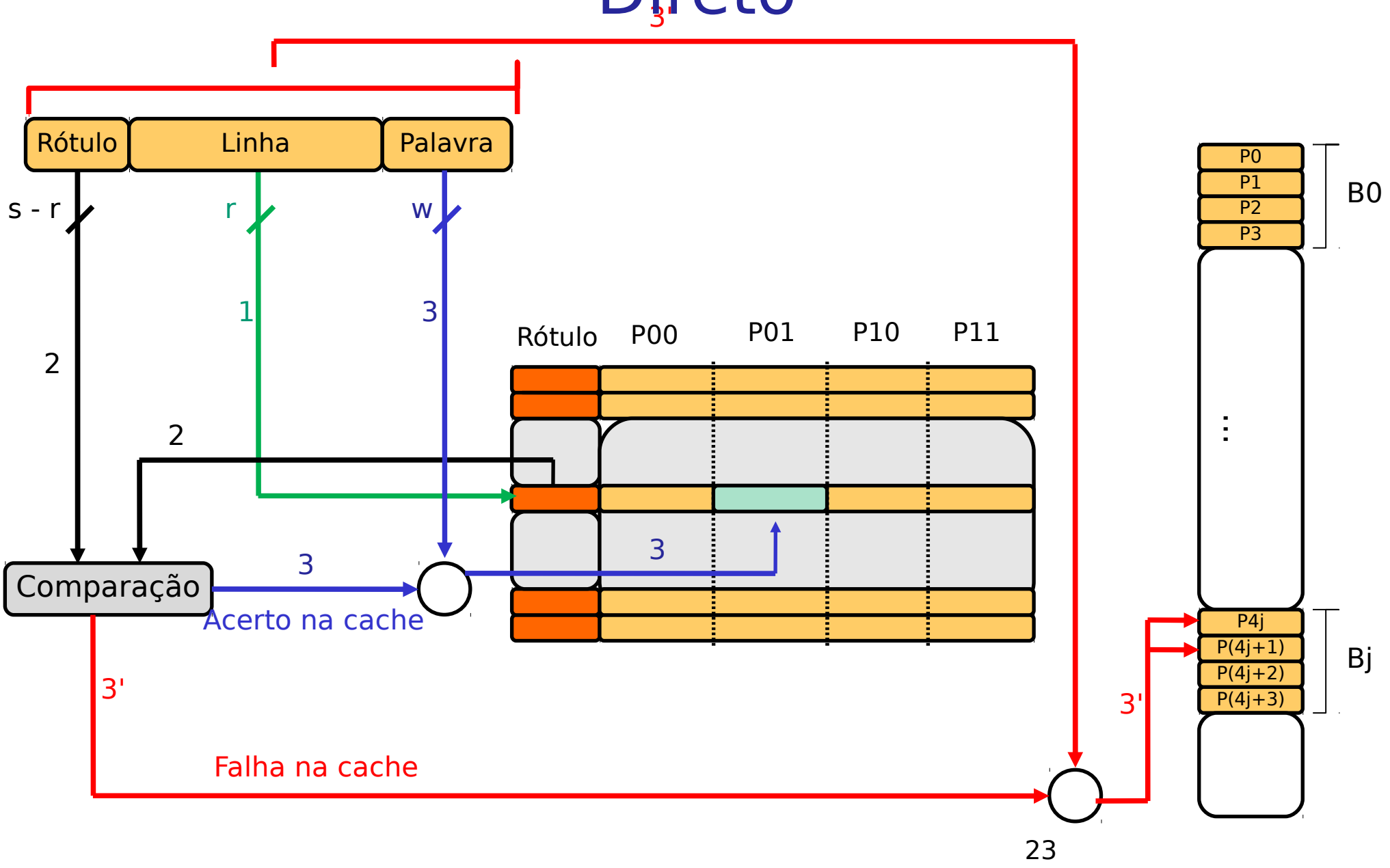


  - No nosso exemplo
  - Se temos palavras de 22 bytes, logo **w = 2**
  - Se temos 222 blocos, precisamos de 22 bits para identificar o bloco, logo **s = 22**
  - Se temos 214 linhas de cache, precisamos de 14 bits, **r = 14**
  - Podemos dividir os 222 blocos da MP nas 214 linhas da cache, dando um total de 8 possibilidades,

Exemplo de Mapeamento Direto



Exemplo de Leitura no Mapeamento Direto



Elementos do projeto da memória cache

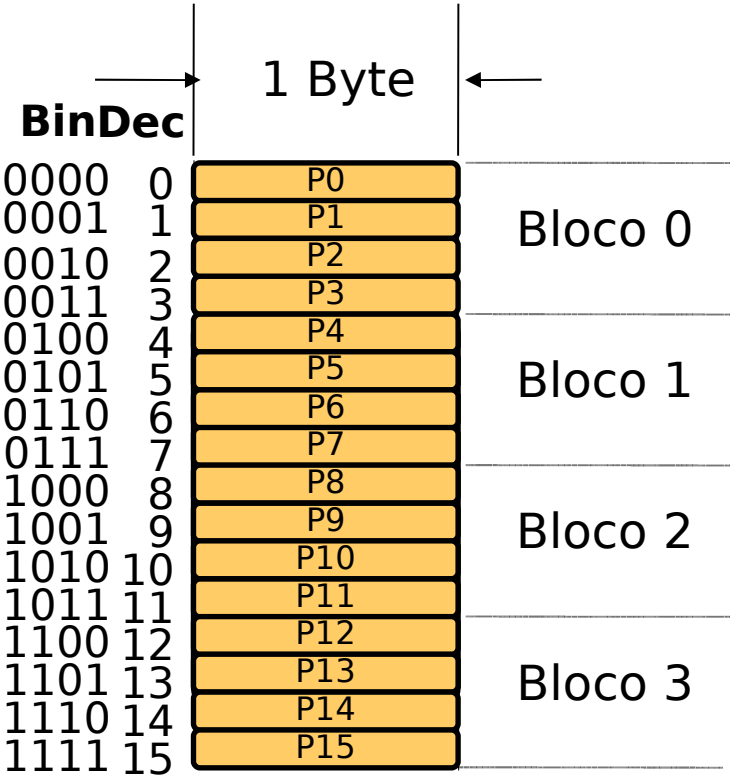
- Função de mapeamento(Mapeamento Direto)
  - r: indica a linha da cache
  - w: indica a palavra dentro da linha da cache
  - s-r: é o rótulo que vai diferenciar os blocos da memória principal que podem está mapeados na linha da cache

Elementos do projeto da memória cache

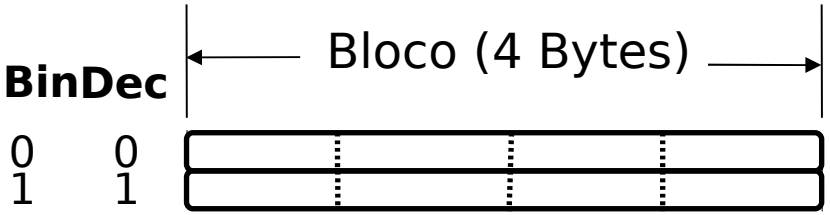
- Função de mapeamento(Mapeamento Direto)
  - Vantagens:
  - Simplicidade
  - Custo baixo de implementação
  - Desvantagem:
  - Se um programa fizer repetidas referências a palavras em dois blocos distintos, mapeados em uma mesma linha, esses blocos serão trocados continuamente na cache – e a taxa de acertos será baixa

Elementos do projeto da memória cache (Exemplo)

- Memória principal com 16 B (24 B)
- Cada Byte é endereçável diretamente
- Memória principal pode ser vista como
  4 (22) blocos de 4 B cada
- Memória cache de 8 B, organizada em 2 (21) linhas de 4 B
- Os dados são transferidos da memória principal para a cache em blocos de 4B



Memória principal

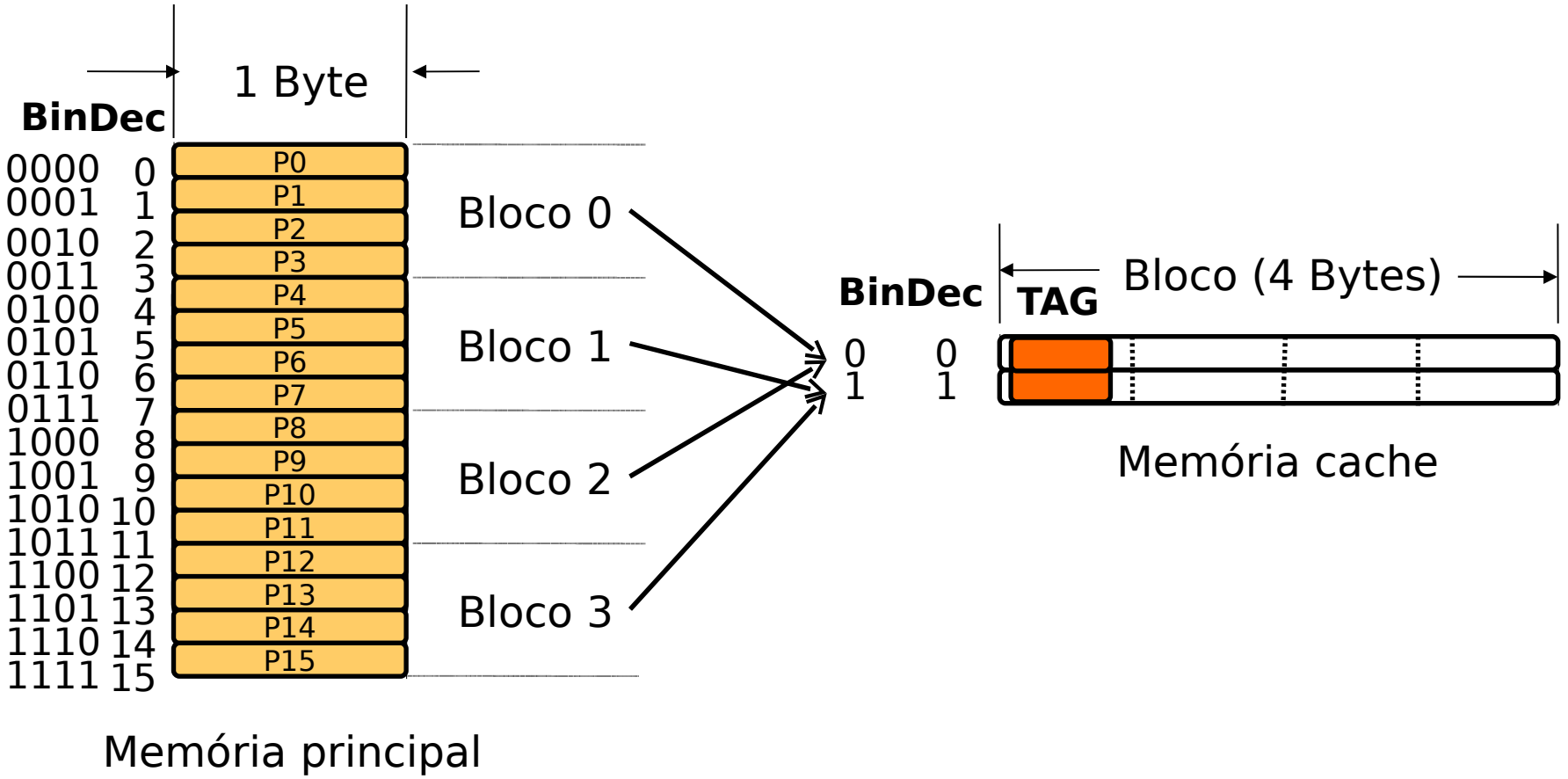


Memória cache

Elementos do projeto da memória cache (Exemplo)

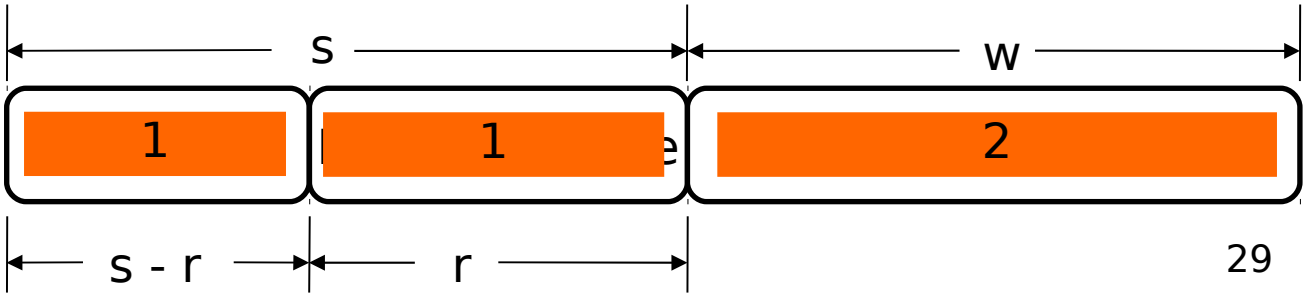
- Matematicamente, em que linha da cache cada bloco pode está?
  - L = id bloco MOD quant. linhas da cache
  - L = 0 MOD 2
  - L = 0
  - L = 1 MOD 2
  - L = 1
  - L = 2 MOD 2
  - L = 0
  - L = 3 MOD 2
  - L = 1

Elementos do projeto da memória cache (Exemplo)

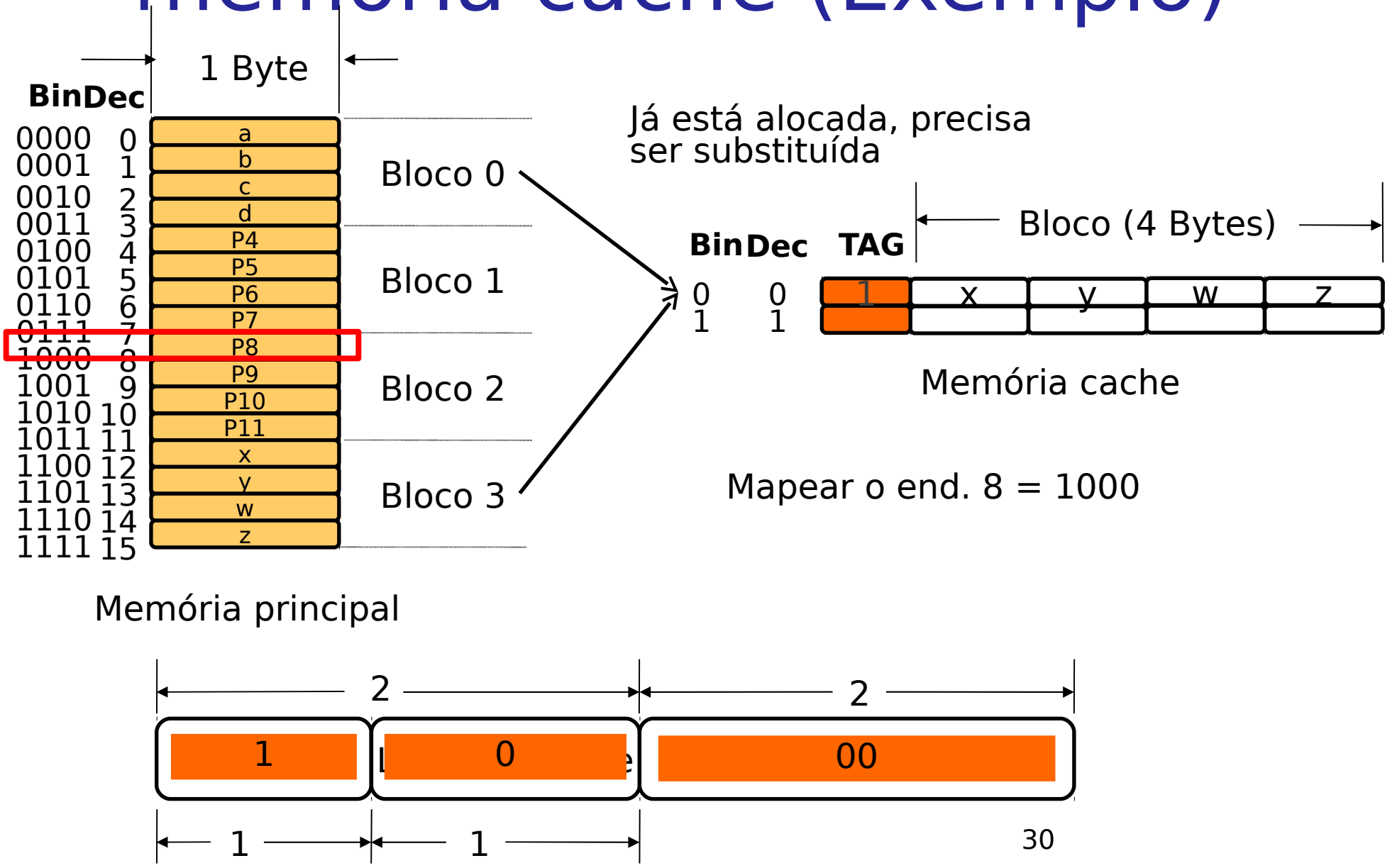


Elementos do projeto da memória cache (Exemplo)

- Como dividir o endereço de memória?
  - Temos 4 bytes em cada bloco
  - 4 = 22, logo w = 2
  - Temos 2 linhas na cache
  - 2 = 21, logo r = 1
  - Temos 4 blocos na memória principal
  - 4 = 22, logo s = 2 => s-r = 1



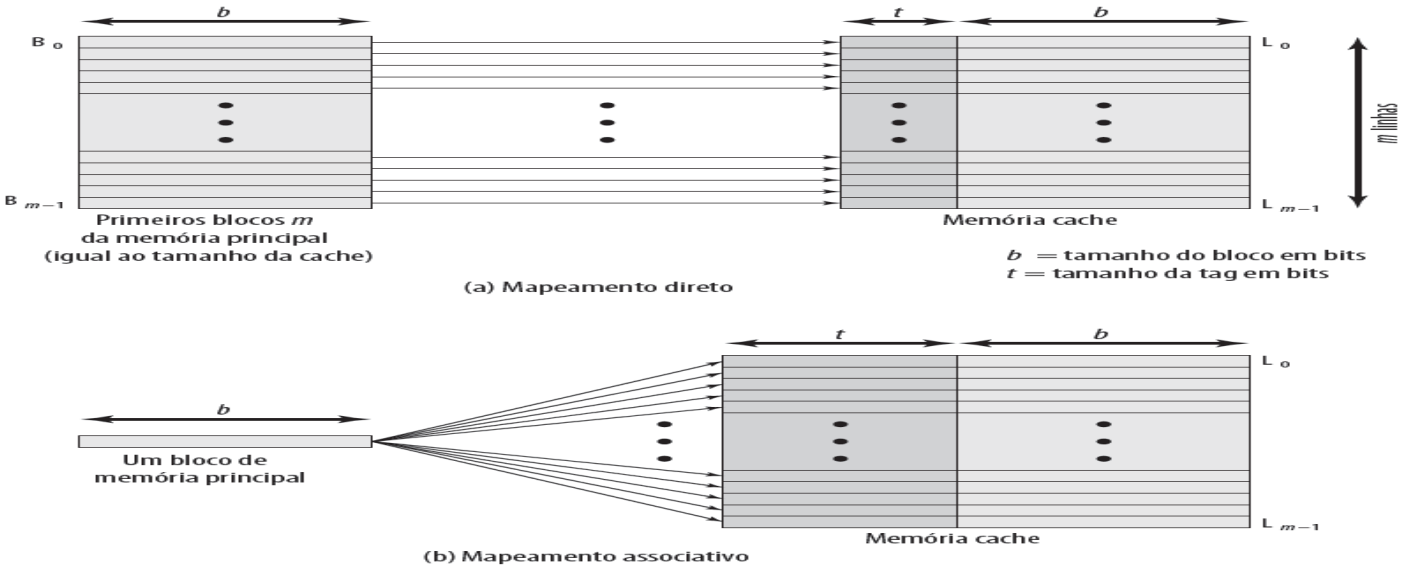
Elementos do projeto da memória cache (Exemplo)



Elementos do projeto da memória cache

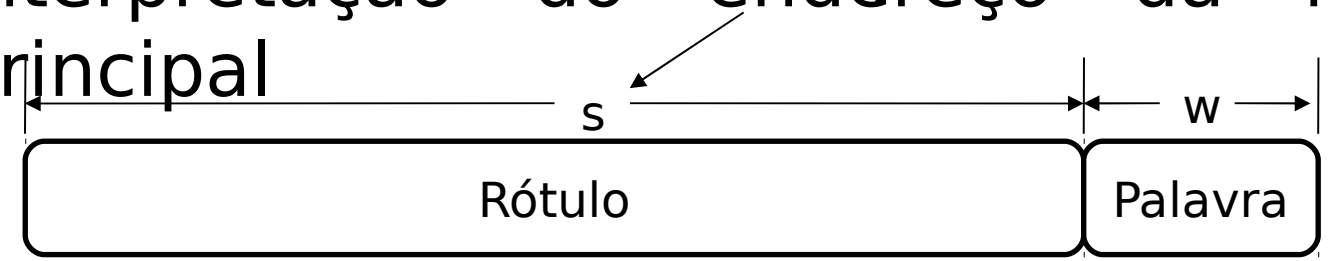
- Função de mapeamento(Mapeamento Associativo)
  - Permite que cada bloco da memória

Figura 4.8 Mapeamento da memória principal para a cache: direto e associativo

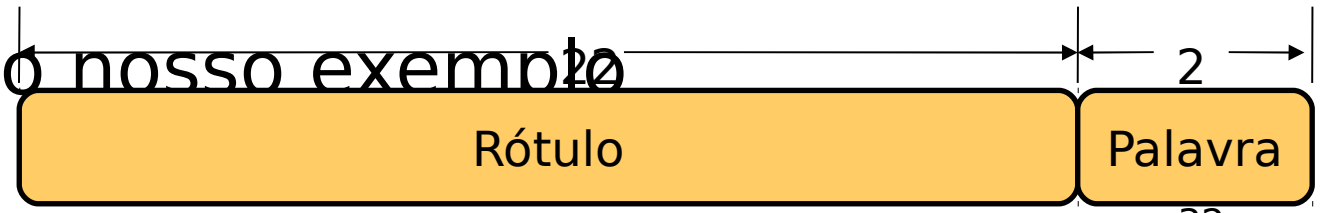


Elementos do projeto da memória cache

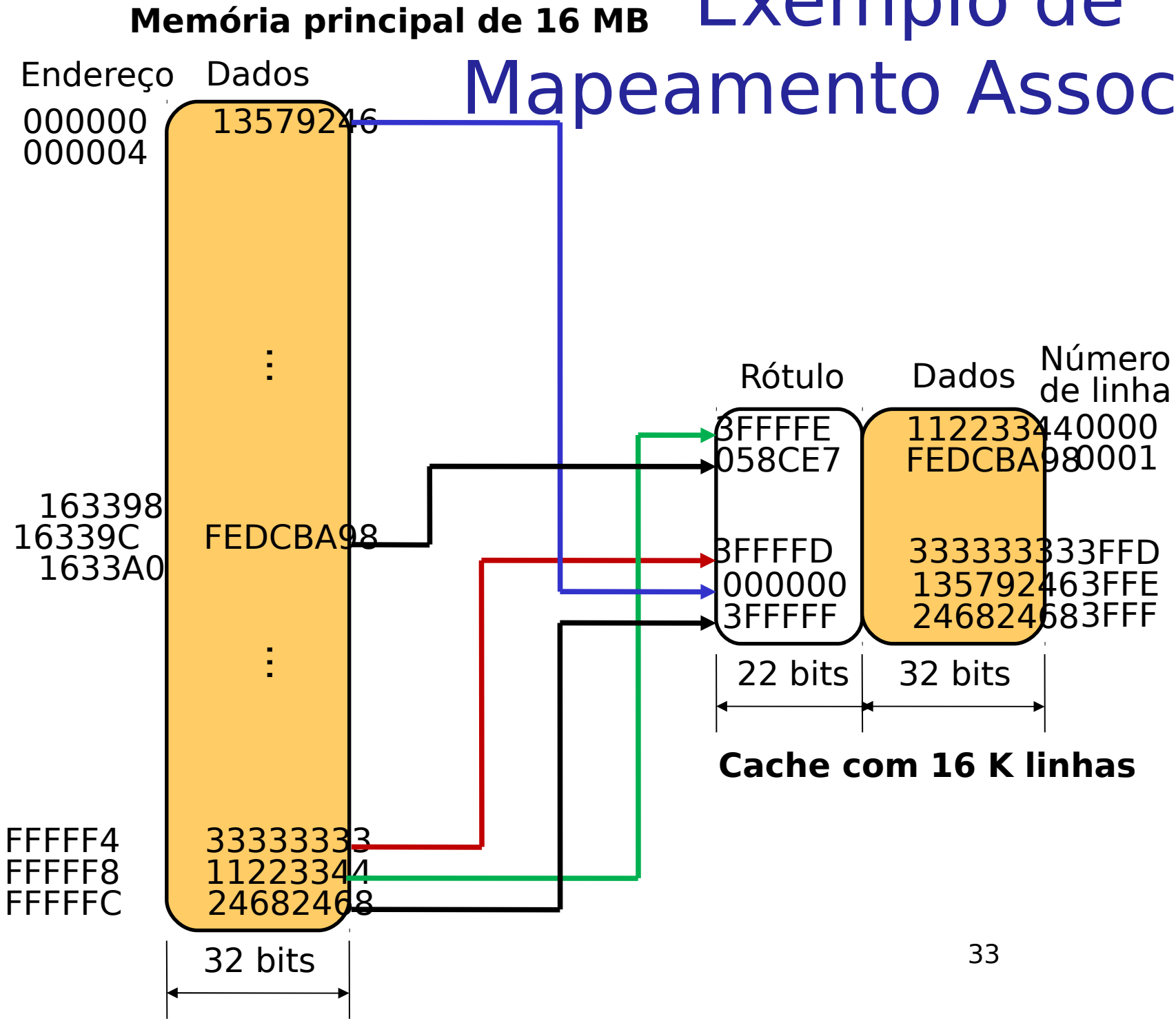
- Função de mapeamento(Mapeamento Associativo)
  - Interpretação do endereço da memória principal



  - No nosso exemplo



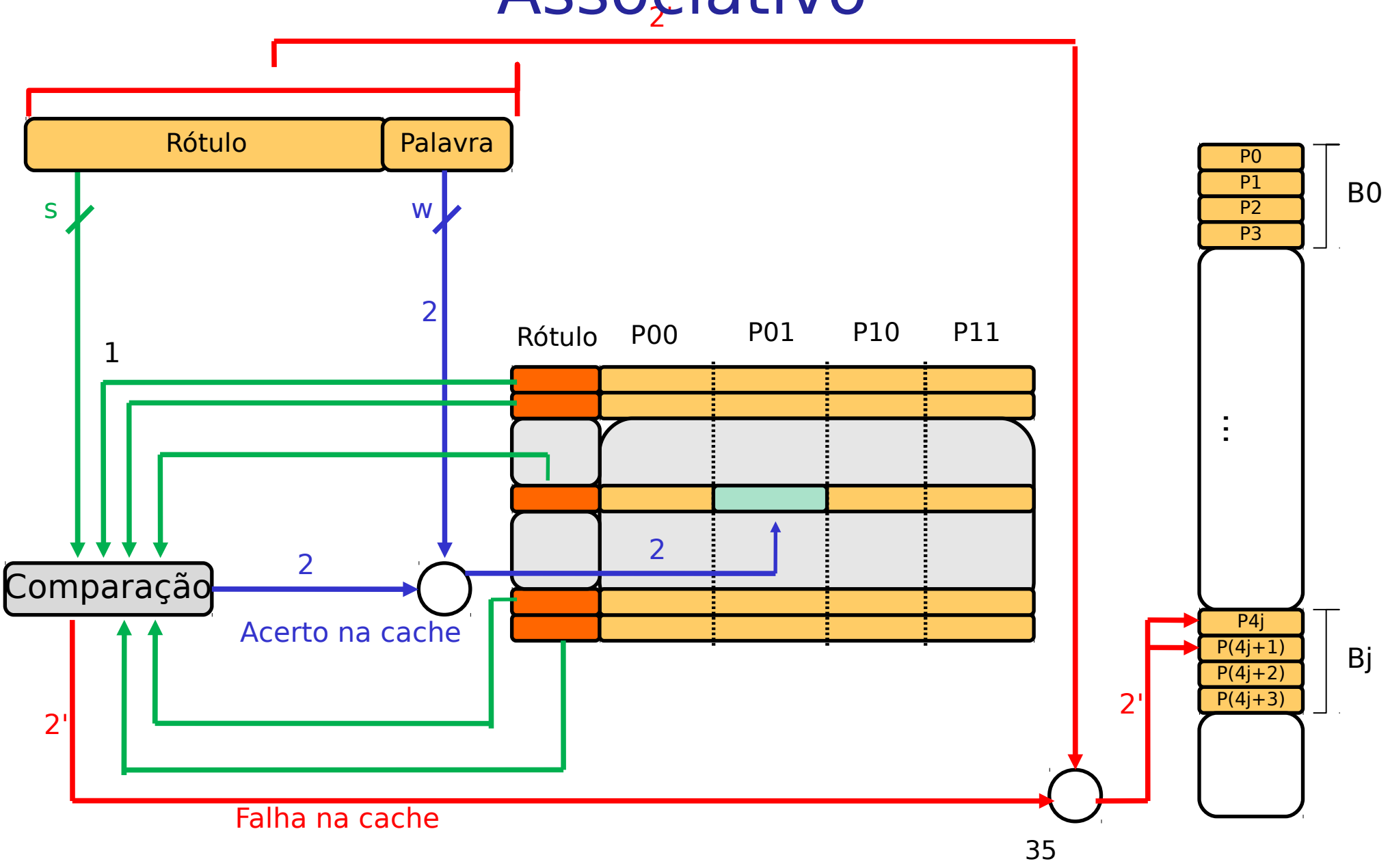
Exemplo de Mapeamento Associativo



Elementos do projeto da memória cache

- Função de mapeamento(Mapeamento Associativo)
  - O rótulo corresponde aos 22 bits mais significativos do endereço
  - Para determinar se um bloco está na cache, a lógica de controle da cache precisa comparar simultaneamente a tag de cada linha

Exemplo de Leitura no Mapeamento Associativo



Elementos do projeto da memória cache

- Função de mapeamento(Mapeamento Associativo)
  - Vantagens
  - Oferece maior flexibilidade para escolha do bloco a ser substituído quando um novo bloco é trazido para a memória cache
  - Desvantagens
  - Complexidade do conjunto de circuitos necessários para a comparação simultânea dos rótulos de todas as linhas da memória cache

Elementos do projeto da memória cache

- Função de mapeamento(Associativo por Conjuntos)
  - Combina as vantagens do mapeamento direto e do mapeamento associativo e diminui suas desvantagens
  - A memória cache é dividida em *v* conjuntos, cada qual com *k* linhas

$m = v \times k$

$i = j \text{ módulo } v$

Em que:

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

Elementos do projeto da memória cache

- Função de mapeamento(Associativo por Conjuntos)
  - Combina as vantagens do mapeamento direto e do mapeamento associativo e diminui suas desvantagens
  - A memória cache é dividida em *v* conjuntos, cada qual com *k* linhas

$m = v \times k$

$i = j \text{ módulo } v$

Em que:

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

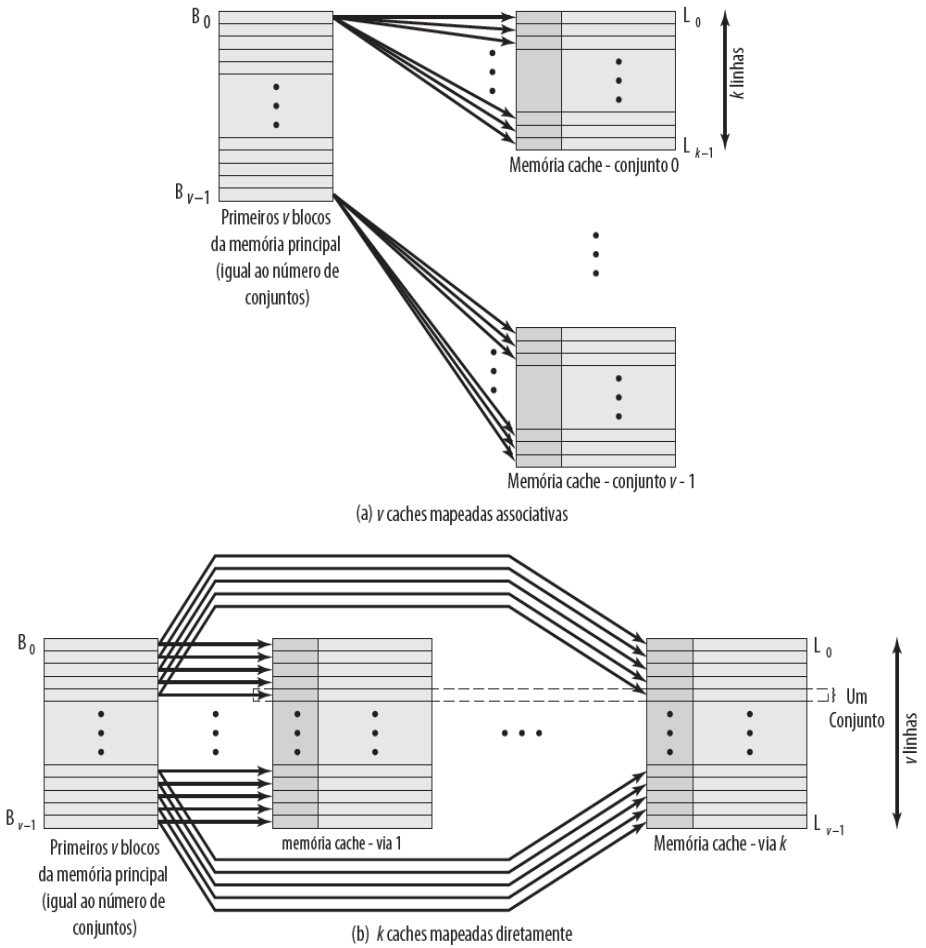
Elementos do projeto da memória cache

- Função de mapeamento(Associativo por Conjuntos)
  - Conhecido como mapeamento associativo em conjunto com **k** linhas por conjunto
  - Com o mapeamento associativo em conjunto, o bloco **Bj** pode ser mapeado para qualquer uma das linhas do conjunto **j**.
  - Pode ser implementado fisicamente como **v** caches associativas
  - Ou como **k** caches de mapeamento direto

Elementos do projeto da memória cache

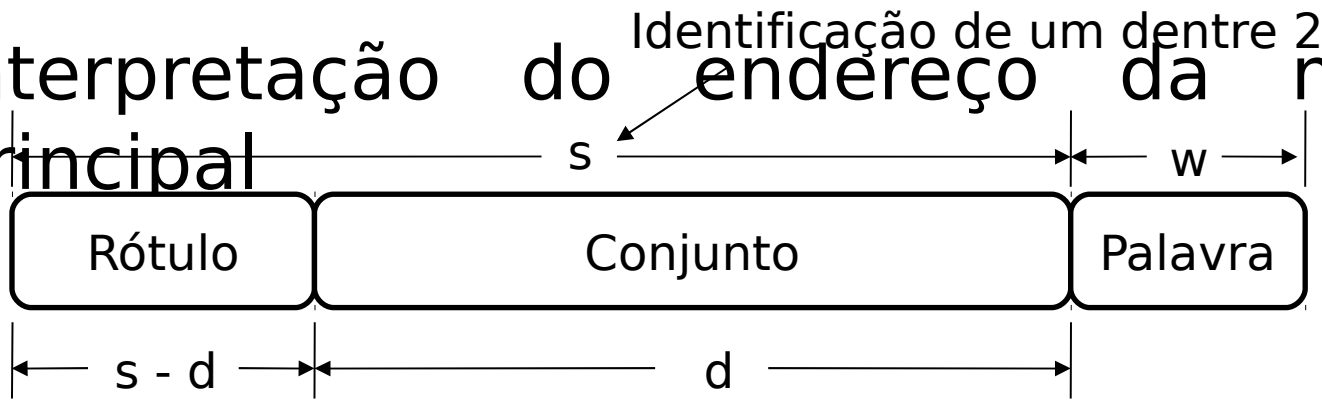
- Função de mapeamento(Associativo por Conju

Figura 4.13 Mapeamento da memória principal na memória cache: associativa em conjunto com *k* linhas por conjunto (*k-way*)



Elementos do projeto da memória cache

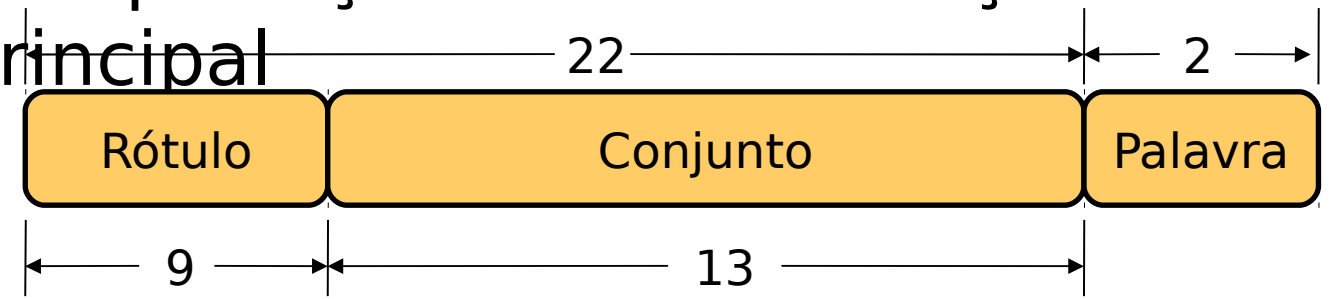
- Função de mapeamento(Associativo por Conjuntos)
  - Interpretação do endereço da memória principal



  - Considerando em exemplo conjuntos de duas linhas:
  - Temos palavras de 22 bytes, logo **w = 2**
  - Temos 222 blocos, logo **s = 22**
  - Agora as 214 linhas de cache precisam ser divididas em 2 conjuntos: 214/21 = 213, logo **d = 13**
  - Podemos dividir os 222 blocos da MP nos 213

Elementos do projeto da memória cache

- Função de mapeamento(Associativo por Conjuntos)
  - Interpretação do endereço da memória principal



  - Considerando em exemplo conjuntos de duas linhas:
  - Temos palavras de 22 bytes, logo **w = 2**
  - Temos 222 blocos, logo **s = 22**
  - Agora as 214 linhas de cache precisam ser divididas em 2 conjuntos: 214/21 = 213, logo **d = 13**
  - Podemos dividir os 222 blocos da MP nos 213

Elementos do projeto da memória cache

- Função de mapeamento(Associativo por Conjuntos)
  - Segundo $i = j$ módulo v, cada bloco da m

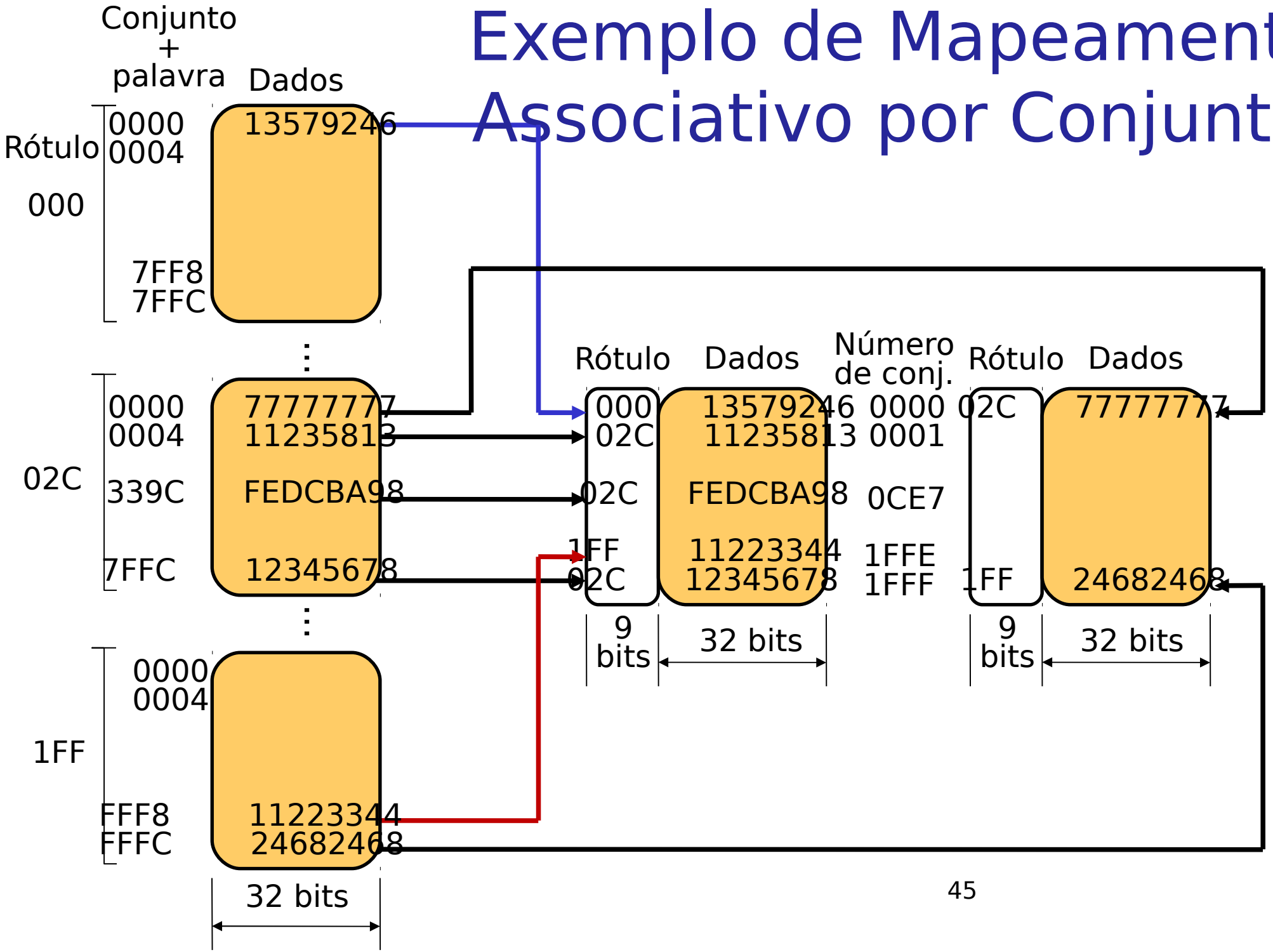
Conjunto da memória cache	Blocos da MP mapeados no conjunto
0	*0, v, 2v, ..., 2S - v*
1	*1, v + 1, 2v + 1, ..., 2S – v + 1*
⋮	⋮
v- 1	*v – 1, 2v – 1, 3v – 1, ..., 2S - 1*

Elementos do projeto da memória cache

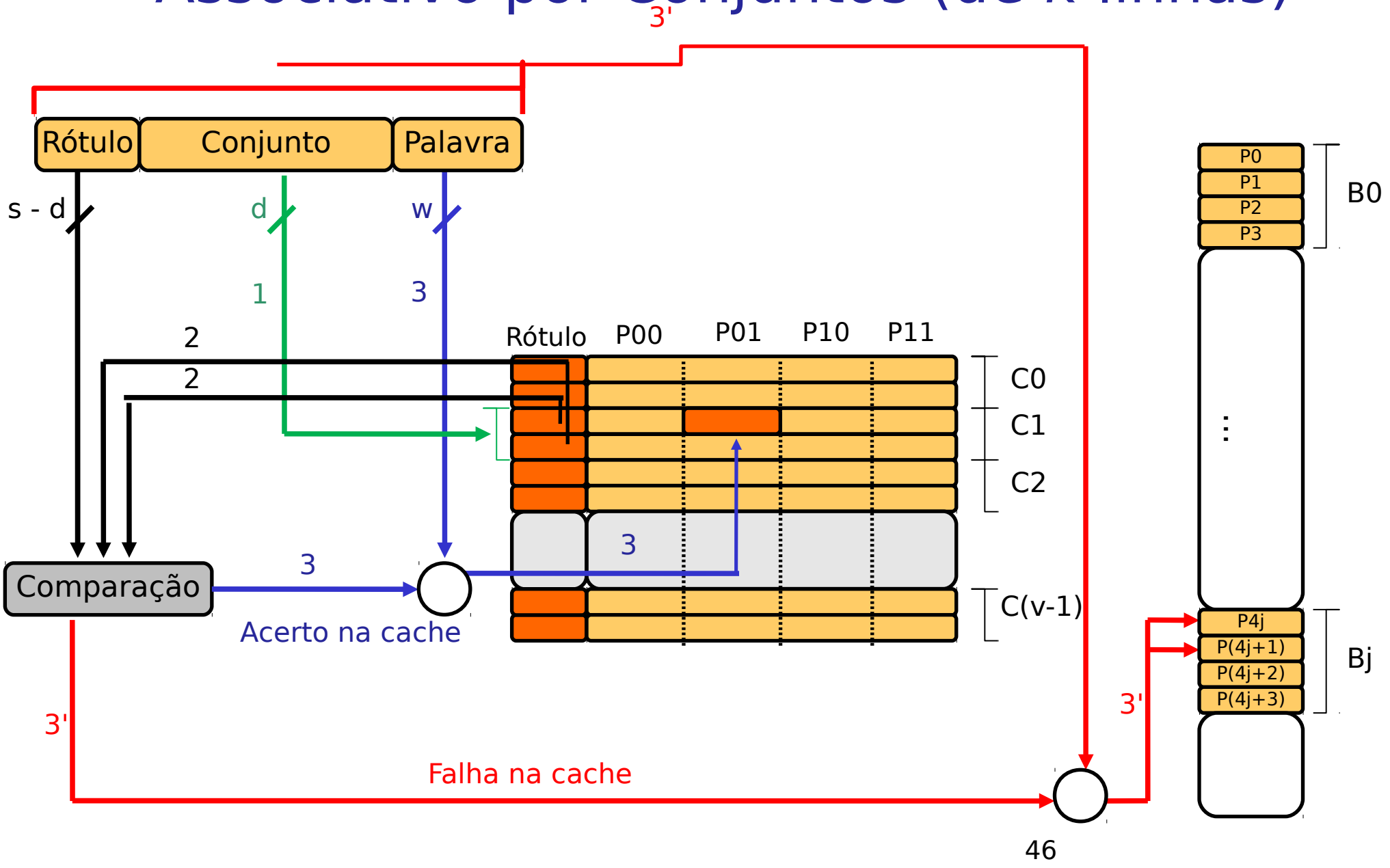
- Função de mapeamento(Associativo por Conjuntos)

Conjunto da memória cache	Blocos da MP mapeados no conjunto
0	000000, 008000, ..., FF8000
1	000004, 008004, ..., FF8004
⋮	⋮
213 - 1	007FFC, 00FFFC, ..., FFFFFC

Exemplo de Mapeamento Associativo por Conjunto



Exemplo de Leitura no Mapeamento Associativo por Conjuntos (de *k* linhas)



Elementos do projeto da memória cache

- Função de mapeamento(Associativo por Conjuntos)
  - Casos extremos:
  - $v = m$ e $k = 1$: mapeamento direto (m conjuntos de 1 linha)
  - $v = 1$ e $k = m$: mapeamento associativo (1 conj. de m linhas)
  - Configurações comuns:
  - $v = m/2$ e $k = 2$: taxa de acertos significantemente maior do que no mapeamento direto
  - $v = m/4$ e $k = 4$: pequena melhoria a um custo adicional relativamente pequeno
  - $K > 4$: sem melhoras significativas de desempenho

Elementos do projeto da memória cache

- Algoritmos de substituição
  - Quando um novo bloco é trazido para a cache, um dos blocos existentes deve ser substituído
  - No mapeamento direto, não há alternativa – cada bloco é mapeado em uma única linha
  - Para os mapeamentos associativo e associativo por conjuntos, é necessário um algoritmo de substituição
  - Recomenda-se a implementação em HW, por motivo de desempenho

Elementos do projeto da memória cache

- Algoritmos de substituição
  - LRU (Menos Recentemente Usado)
  - Implementação com bits de uso
  - FIFO (First In First Out)
  - Implementação com áreas de armazenamento circular
  - LFU (Menos Frequentemente Usado)
  - Implementação com contadores
  - Aleatório
  - Apresenta um desempenho apenas levemente inferior aos demais

Elementos do projeto da memória cache

- Política de escrita
  - Antes que um bloco residente na memória possa ser substituído, é necessário verificar se ele foi alterado na memória cache
  - Se isso não ocorreu, então o novo bloco pode ser escrito sobre o bloco antigo
  - Caso contrário, então a memória principal deve ser atualizada
  - Problema encontrado:
  - A memória principal pode ser utilizada tanto por outros processadores quanto por dispositivos de E/S

Elementos do projeto da memória cache

- Política de escrita
  - Escrita Direta (*write through)*
  - Todas as operações de escrita são feitas tanto na memória quanto na cache
  - Vantagem:
    - A memória principal está sempre atualizada
  - Desvantagem:
    - Geração de tráfego de memória considerável

Elementos do projeto da memória cache

- Política de escrita
  - Escrita de Volta *(write back)*
  - Escritas são feitas apenas na cache
  - Quando uma linha da cache é atualizada, um bit de atualização associado a ela é setado em 1
  - Quando um bloco for substituído, ele é escrito de volta na memória apenas se o seu bit de atualização for 1
  - Vantagem:
    - Minimiza o número de operações de escrita na memória
  - Desvantagem:
    - Partes da memória principal podem ficar inválidas

Elementos do projeto da memória cache

- Política de escrita
  - Escrita Uma Vez *(write once)*
  - Ideal para sistemas multiprocessados com memória principal compartilhada
  - É uma mistura de write through e write back
  - Cada µP escreve a memória principal sempre que o bloco correspondente na cache foi atualizado pela primeira vez (write through)
  - Os demais µP são alertados da alteração
  - Outras alterações naquele bloco são realizadas apenas na cache local e o bloco da memória só será atualizado quando o bloco for substituído na cache (write back)

Elementos do projeto da memória cache

- Tamanho da linha
  - Tamanho da linha = tamanho do bloco
  - À medida em que esse número aumenta, aumenta inicialmente a taxa de acertos
  - Entretanto, se esse número aumentar muito, a taxa de acertos diminuirá
  - Porque não usar blocos muito grandes:
  - Para uma dada capacidade, o número de linhas diminui
  - Cada palavra adicional estará mais distante da usada – e a chance de uso será menor
  - Tamanho ideal: 2 – 8 palavras

Elementos do projeto da memória cache

- Número de memórias cache
  - Anteriormente, a cache era externa ao processador
  - Com o avanço da eletrônica, tem-se:
  - Cache L1: interna ao processador
  - Cache L2: externa ao processador
    - Também podem ser internas e incluir uma cache L3
  - Unificadas ou Separadas?
  - Unificadas (instruções + dados)
  - Separadas (uma para instruções e outra para os dados)
  - Projetos atuais baseiam-se em caches separadas, por motivos de desempenho junto ao pipeline

Referências

- STALLINGS, W. Arquitetura e organização de computadores: projeto para o desempenho. 8. ed. Prentice Hall, 2009.
- DELGADO, J.; RIBEIRO, C. Arquitetura de Computadores. 2 ed. LTC, 2009.
- PATTERSON, D. A. ; HENNESSY, J.L. Organização e projeto de computadores – a interface hardware software. 3. ed. Editora Campus, 2005.