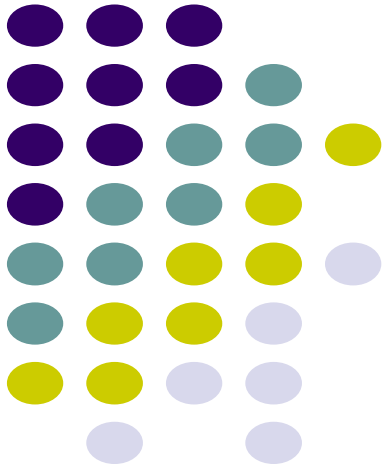
UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO
DEPARTAMENTO DE CIÊNCIAS EXATAS E NATURAIS
CURSO DE CIÊNCIA DA COMPUTAÇÃO

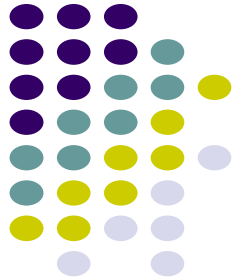
Arquitetura e Organização de Computadores

Hierarquia de Memória - Parte II



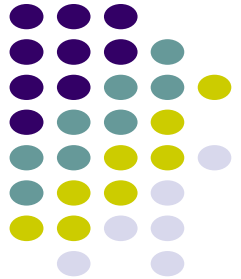
Prof. Sílvio Fernandes

Reduzindo falhas de cache com um posicionamento de blocos flexível



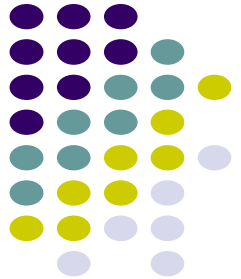
- Em um extremo está o mapeamento direto, onde um bloco só pode ser posicionado exatamente em um local
- No outro extremo está um esquema em que um bloco pode ser posicionado em *qualquer* local na cache – **totalmente associativo**
  - Para encontrar um determinado bloco em uma cache assim, todas as entradas precisam ser pesquisadas
- A faixa intermediária é chamada **associativa por conjunto**

Reduzindo falhas de cache com um posicionamento de blocos flexível



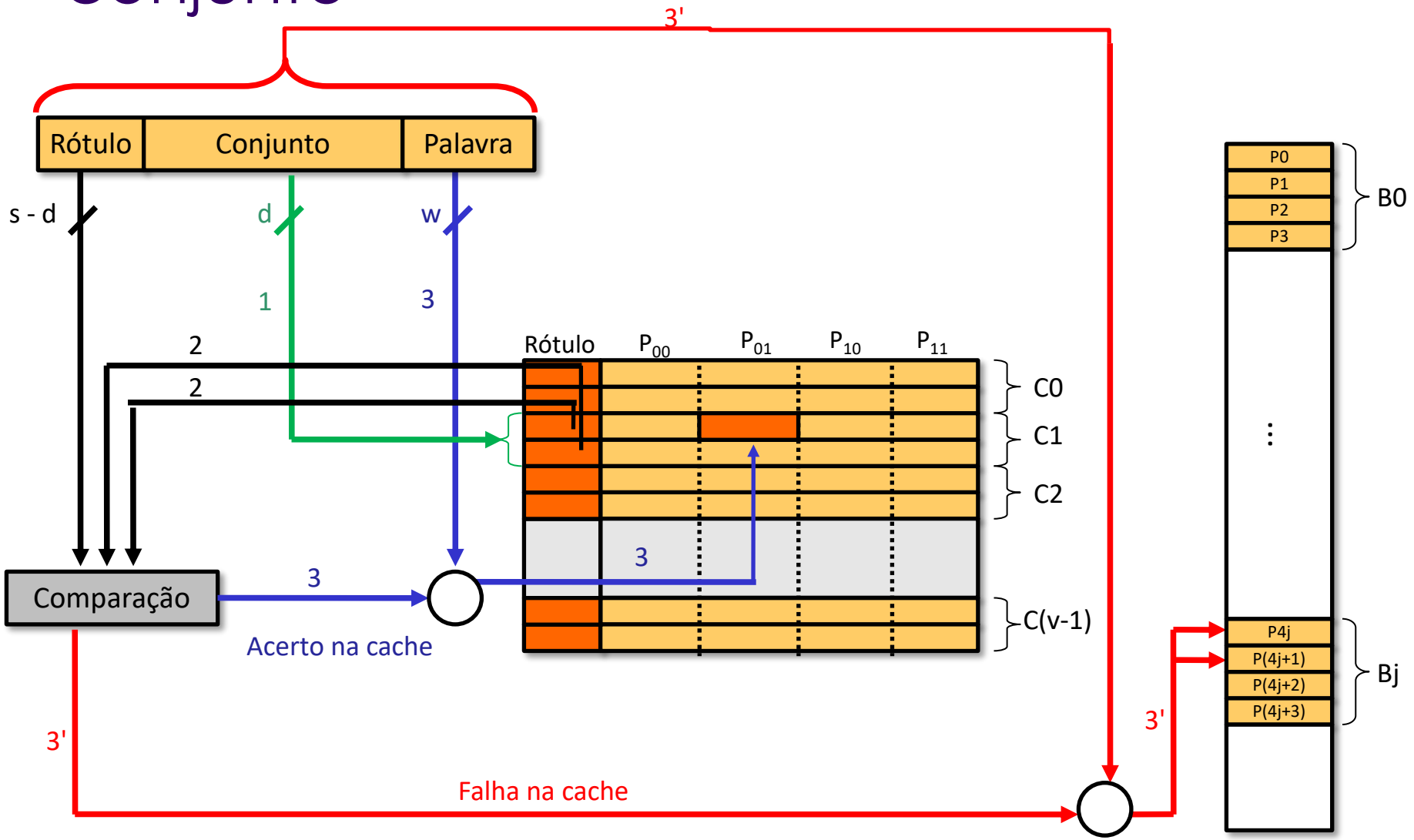
- Cache associativa por conjunto
  - Existe um número fixo de locais (pelo menos 2) onde cada bloco pode ser colocado
  - Com **n** locais para um bloco é chamado de cache associativa por conjunto de **n vias**
  - Cada bloco é mapeado para um *conjunto* único na cache, determinado pelo campo índice, e um bloco pode ser colocado em *qualquer* elemento desse conjunto

Reduzindo falhas de cache com um posicionamento de blocos flexível

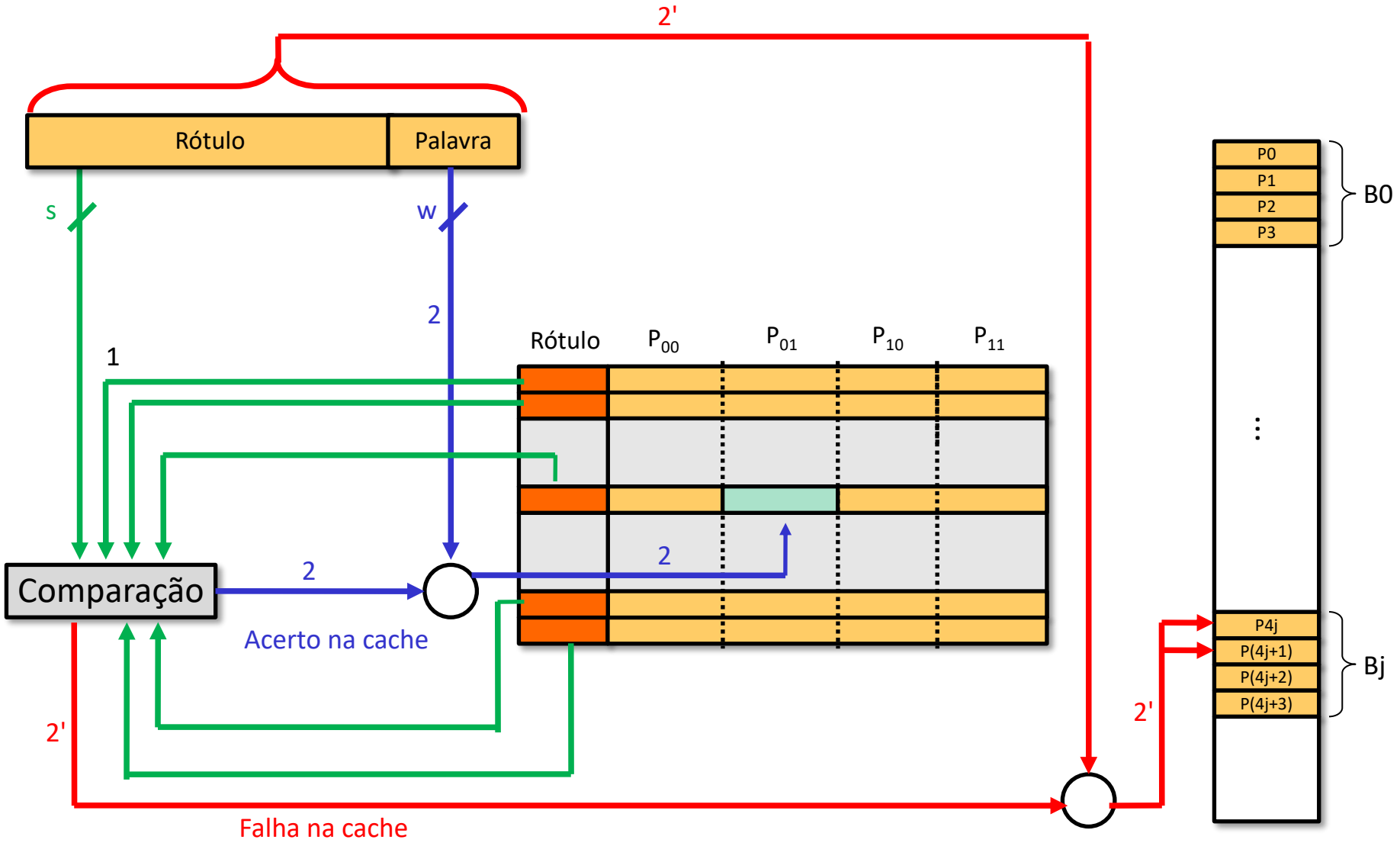


- Cache associativa por conjunto
  - O conjunto contendo um bloco de memória é determinado por
    - (número do bloco) módulo (número de conjuntos na cache)
  - Como o bloco pode ser colocado em qualquer elemento do conjunto, *todas as tags de todos os elementos do conjunto* precisam ser pesquisadas

Mapeamento associativo por conjunto

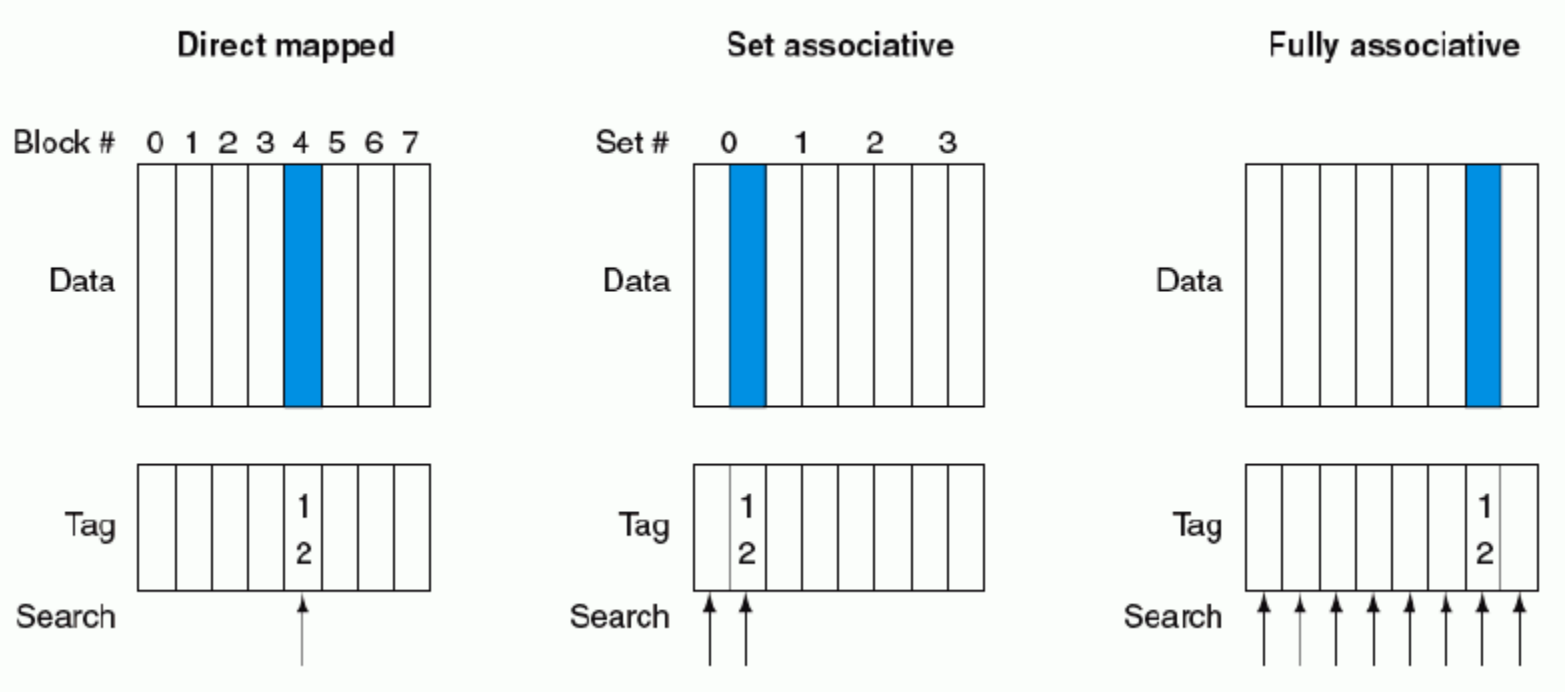


Mapeamento totalmente associativo

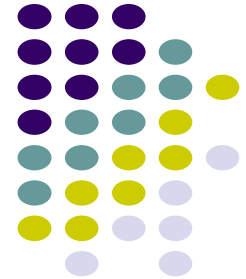


Reduzindo falhas de cache com um posicionamento de blocos flexível

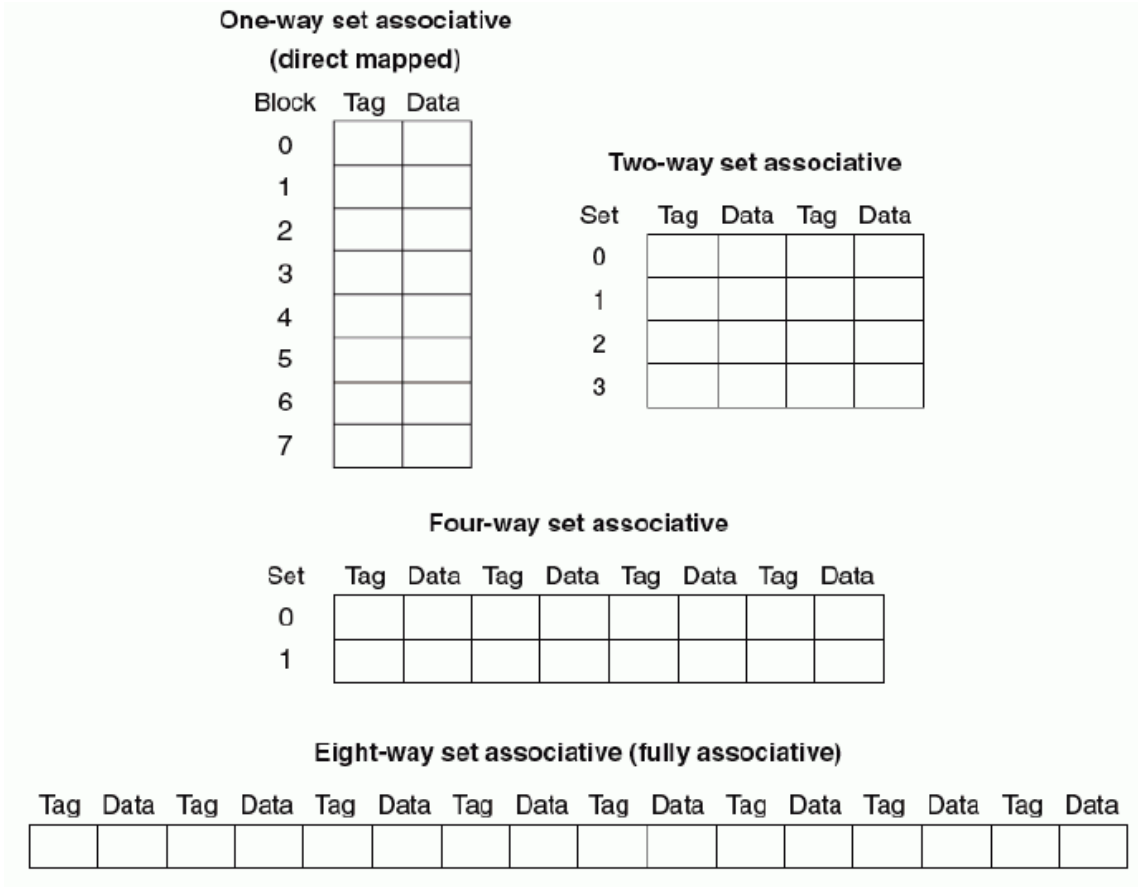
- O mapeamento do bloco 12 nos diferentes esquemas



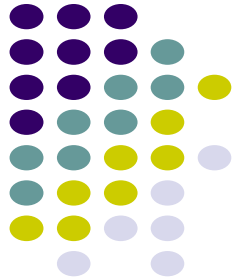
Reduzindo falhas de cache com um posicionamento de blocos flexível



- Possíveis estruturas de associatividade para uma cache de 8 blocos

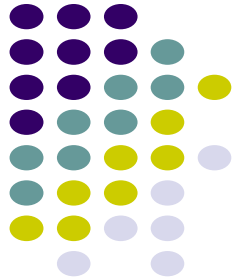


Reduzindo falhas de cache com um posicionamento de blocos flexível



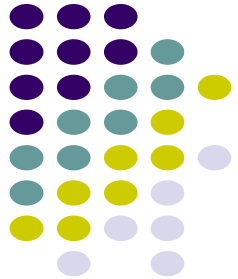
- A vantagem de aumentar o grau da associatividade é que ela normalmente diminui a taxa de falhas. A principal desvantagem é um aumento no tempo de acerto

Reduzindo falhas de cache com um posicionamento de blocos flexível



- **Exemplo:** Considere 3 caches pequenas, cada uma consistindo em 4 blocos de uma word cada. Uma cache é totalmente associativa, uma 2ª é associativa por conjunto de 2 vias e a 3ª é diretamente mapeada. Encontre o número de falhas para cada organização de cache, dada a seguinte sequência de endereços de blocos: 0, 8, 0, 6, 8.

Reduzindo falhas de cache com um posicionamento de blocos flexível

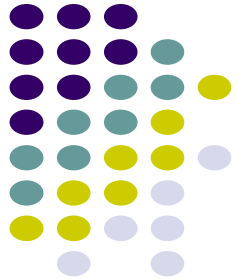


- **Resposta:**
  - O caso diretamente mapeado é mais fácil. Primeiro vamos determinar para qual bloco cada endereço de bloco é mapeado:

Block address	Cache block
0	(0 modulo 4) = 0
6	(6 modulo 4) = 2
8	(8 modulo 4) = 0

  - Agora podemos preencher o conteúdo da cache após cada referência, usando uma entrada em branco para indicar que o bloco é inválido

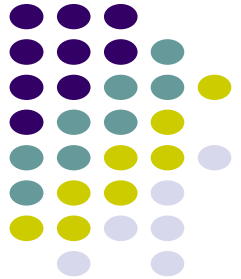
Reduzindo falhas de cache com um posicionamento de blocos flexível



- **Resposta:**
  - A cache diretamente mapeada gera 5 falhas para os cinco acessos

Address of memory block accessed	Hit or miss	Contents of cache blocks after reference			
		0	1	2	3
0	miss	Memory[0]			
8	miss	Memory[8]			
0	miss	Memory[0]			
6	miss	Memory[0]		Memory[6]	
8	miss	Memory[8]		Memory[6]	

Reduzindo falhas de cache com um posicionamento de blocos flexível

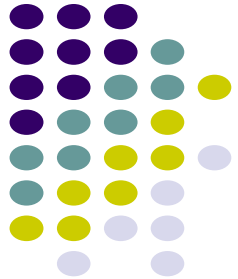


- **Resposta:**
  - A cache associativa por conjunto possui 2 conjuntos (com índices 0 e 1) com 2 elementos por conjunto

Block address	Cache set
0	(0 modulo 2) = 0
6	(6 modulo 2) = 0
8	(8 modulo 2) = 0

  - Já que temos uma escolha de qual entrada em um conjunto substituir em um falha, precisamos de uma regra de substituição. Usaremos a regra que o bloco usado há mais tempo é substituído

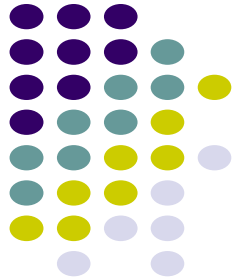
Reduzindo falhas de cache com um posicionamento de blocos flexível



- **Resposta:**
  - A cache associativa por conjunto gera 4 falhas

Address of memory block accessed	Hit or miss	Contents of cache blocks after reference			
		Set 0	Set 0	Set 1	Set 1
0	miss	Memory[0]			
8	miss	Memory[0]	Memory[8]		
0	hit	Memory[0]	Memory[8]		
6	miss	Memory[0]	Memory[6]		
8	miss	Memory[8]	Memory[6]		

Reduzindo falhas de cache com um posicionamento de blocos flexível



- **Resposta:**
  - A cache totalmente associativa possui 4 blocos de cache (em um único conjunto); qualquer bloco de memória pode ser armazenado em qualquer bloco da cache

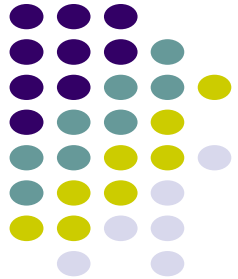
Address of memory block accessed	Hit or miss	Contents of cache blocks after reference			
		Block 0	Block 1	Block 2	Block 3
0	miss	Memory[0]			
8	miss	Memory[0]	Memory[8]		
0	hit	Memory[0]	Memory[8]		
6	miss	Memory[0]	Memory[8]	Memory[6]	
8	hit	Memory[0]	Memory[8]	Memory[6]	

  - Gera apenas 3 falhas

Localizado um bloco na cache

- Cada bloco em uma cache associativa por conjunto inclui uma tag de endereço que fornece o endereço do bloco
- A tag de cada bloco dentro do conjunto apropriado é verificada para ver se corresponde ao endereço do bloco vindo do processador

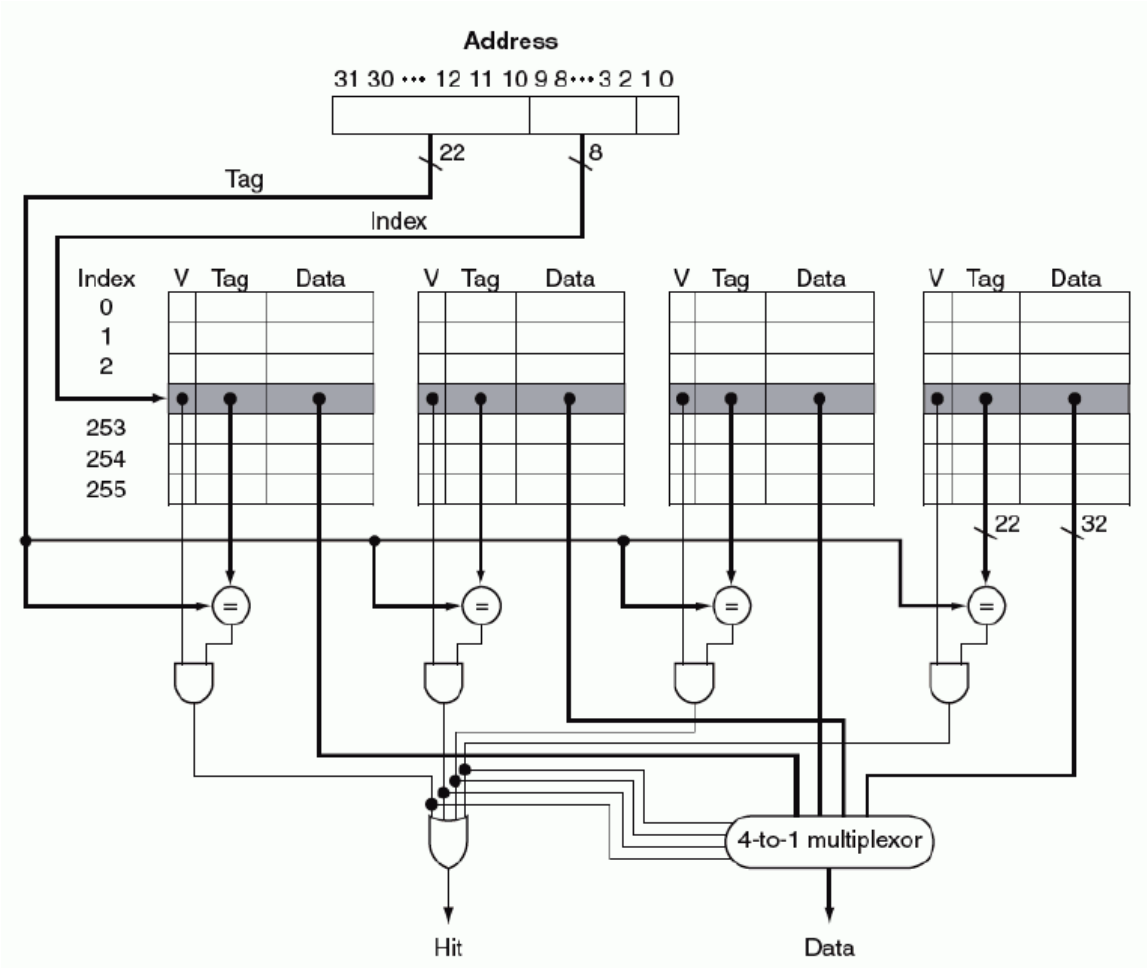
Localizado um bloco na cache



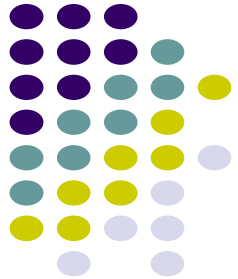
- O valor do índice é usado para selecionar o conjunto contendo o endereço de interesse, e as tags de todos os blocos no conjunto selecionado são pesquisadas em paralelo
- Se o tamanho da cache for mantido igual:
  - Aumentar a associatividade aumenta o número de blocos por conjunto
  - Cada aumento por um fator de 2 na associatividade dobra o número de blocos por conjunto e divide por 2 o número de conjuntos
  - Cada aumento pelo dobro na associatividade diminui o tamanho do índice em 1 bit e aumenta o tamanho da tag em 1 bit

Localizado um bloco na cache

- Cache associativa por conjunto de 4 vias

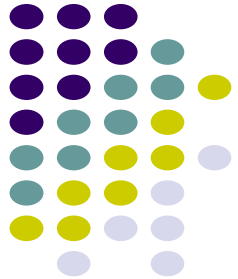


Localizado um bloco na cache



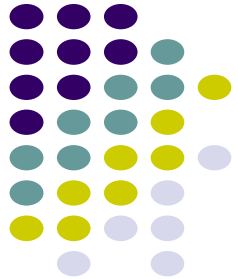
- **Exemplo:** O acréscimo da associatividade requer mais comparadores e mais bits de tag por bloco de cache. Considerando uma cache de 4K blocos, um tamanho de bloco de 4 words e um endereço de 32 bits, encontre o número total de conjuntos e o número total de bits de tag para caches que são diretamente mapeadas, associativa por conjunto de 2 vias e 4 vias e totalmente associativas.

Localizado um bloco na cache



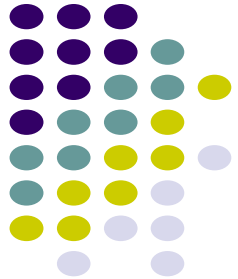
- **Resposta:**
  - Como existem 16 (2^4) bytes por bloco, um endereço de 32 bits produz 32-4 = 28 bits para serem usados para índice e tag.
  - A cache diretamente mapeada possui um mesmo número de conjuntos e blocos e, portanto, 12 bits de índice, já que $\log_2(4K) = 12$; logo, o número total de bits de tag é (28-12) x 4K = 16 x 4K = 64 Kbits.

Localizado um bloco na cache



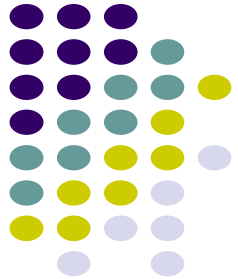
- **Resposta:**
  - Cada grau de associatividade diminui o número de conjuntos por um fator de 2, diminui o número de bits para indexar por 1 e aumenta os bits de tag por 1
    - Para uma cache associativa por conjunto de 2 vias, existem 2K de conjuntos
    - Bits para tag é (28-11) x 2 x 2K = 34 x 2K = 68 Kbits
    - Para uma cache associativa por conjunto de 4 vias:
      - (28-10) x 4 x 1K = 72 x 1K = 72 Kbits
    - Para uma cache totalmente associativa, há apenas 1 conjunto com blocos de 4K, e a tag possui 28 bits, produzindo um total de
      - 28 x 4K x 1 = 112K de bits de tag

Escolhendo que bloco substituir



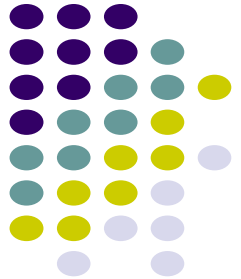
- Em caches associativas, tem uma escolha de onde colocar o bloco requisitado e, portanto, uma escolha de qual bloco substituir
- Em uma cache totalmente associativa, todos os blocos são candidatos à substituição
- Em uma cache associativa por conjunto, precisamos escolher entre os blocos do conjunto selecionado
- O esquema mais comum é o **LRU (Least Recently Used – usado menos recentemente)**

Escolhendo que bloco substituir



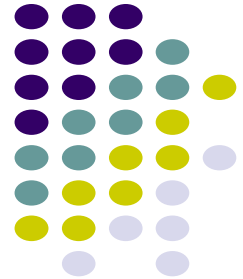
- No LRU, o bloco substituído é aquele que não foi usado há mais tempo
- É implementada monitorando quando cada elemento em um conjunto foi usado em relação aos outros elementos do conjunto
- Para caches associativas por conjunto de 2 vias, o controle de quando os 2 elementos foram usados pode ser implementado mantendo um único bit em cada conjunto e definindo o bit para indicar um elemento sempre que o elemento é referenciado

Reduzindo a penalidade de falhas usando caches multiníveis



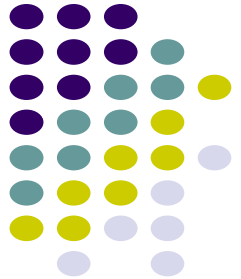
- Para diminuir a diferença entre as rápidas velocidades de clock dos processadores modernos e o tempo para acessar as DRAMs, um nível adicional de cache pode ser incluída

Reduzindo a penalidade de falhas usando caches multiníveis



- **Exemplo:** Suponha que tenhamos um processador com um CPI básico de 1,0, considerando que todas das referências acertem na cache primária e uma velocidade de clock de 5GHz. Considere um tempo de acesso à memória principal de 100ns, incluindo todo o tratamento de falhas. Suponha que a taxa de falhas por instrução na cache primária seja de 2%. O quanto mais rápido será o processador se acrescentarmos uma cache secundária que tenha um tempo de acesso de 5ns para um acerto ou uma falha e que seja grande o suficiente para reduzir a taxa de falhas para a memória principal para 0,5%?

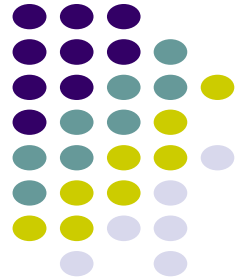
Reduzindo a penalidade de falhas usando caches multiníveis



5Ghz = $5*10^9$ ciclos em 1 s
1 ciclo = x s
X = $0,2*10^{-9}$ s = 0,2 ns

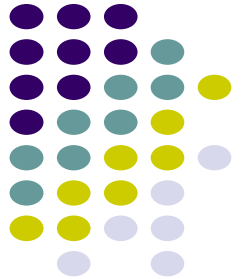
- **Respostas:**
  - A penalidade de falhas para a memória principal é
    - $$\frac{100\ ns}{0{,}2\frac{ns}{ciclos\ de\ clock}} = 500\ ciclos\ de\ clock$$
  - O CPI efetivo com 1 nível de cache é dado por:
    - CPI total = CPI básico + Ciclos de stall de memória por instrução
  - Para o processador com um nível de cache
    - CPI total = 1,0 + 2% x 500 = 11,0
  - Com 2 níveis de cache, uma falha na cache primária pode ser satisfeita pela cache secundária ou pela memória principal.

Reduzindo a penalidade de falhas usando caches multiníveis



- **Respostas:**
  - A penalidade no 2º nível de cache é:
    - $\frac{5\ ns}{0{,}2\frac{ns}{ciclos\ de\ clock}} = 25\ ciclos\ de\ clock$
  - Se a falha for satisfeita na cache secundária, essa será toda a penalidade
  - Se a falha precisar ir à memória principal, a penalidade será a soma do tempo de acesso à cache secundária e o tempo de acesso à memória principal
    - CPI total = 1 + Stalls primários por instrução + Stalls secundários por instrução
    - = 1 + 2% x 25 + 0,5% x 500 = 1 + 0,5 + 2,5 = 4,0

Reduzindo a penalidade de falhas usando caches multiníveis



- **Respostas:**
  - Portanto, o processador com cache secundária é mais rápido por um fator de
    - $\frac{11,0}{4,0} = 2,8$

Referências

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