



Universidade Federal de Pelotas

Instituto de Física e Matemática

Departamento de Informática

Bacharelado em Ciência da Computação

Arquitetura e Organização de Computadores II

Aula 21

**Memória Cache: medidas de desempenho,
caches associativas e tamanho de rótulos,
seleção do bloco a ser substituído.**

Prof. Luciano Volcan Agostini

agostini@ufpel.edu.br

http://www.ufpel.edu.br/~agostini/disciplinas/aoc2_2007_2

3. Hierarquia de Memória: memória cache

► Medidas do Desempenho da Cache

$$\text{Tempo de processador} = \left[\begin{array}{l} \text{Ciclos de relógio} \\ \text{gastos na execução} \\ \text{normal do programa} \end{array} + \begin{array}{l} \text{Ciclos de relógio gastos à} \\ \text{espera do acesso ao} \\ \text{sistema de memória} \end{array} \right] \times \text{Tempo do ciclo} \\ \text{de relógio}$$

Inclui os acessos à cache que resultam em acerto

Devem-se às faltas na cache

Supondo um modelo simplificado de sistema de memória:

$$\text{Ciclos de relógio gastos à} \\ \text{espera do acesso ao} \\ \text{sistema de memória} = \text{Ciclos de relógio} \\ \text{para tratar faltas de} \\ \text{leitura na cache} + \text{Ciclos de relógio} \\ \text{para tratar faltas de} \\ \text{escrita na cache}$$

3. Hierarquia de Memória: memória cache

► Medidas do Desempenho da Cache

Nas leituras:

$$\text{Ciclos de relógio para tratar faltas de leitura na cache} = \frac{\text{Leituras}}{\text{Programa}} \times \text{Taxa de faltas de leitura} \times \text{Penalidade da falta de leitura}$$

Nas escritas (considerando *write-through*):

$$\text{Ciclos de relógio para tratar faltas de escrita na cache} = \left[\frac{\text{Escritas}}{\text{Programa}} \times \text{Taxa de faltas de escrita} \times \text{Penalidade da falta de escrita} \right] + \text{Ciclos parados devido ao } \textit{buffer}$$

Se o buffer estiver bem dimensionado

3. Hierarquia de Memória: memória cache

► Medidas do Desempenho da Cache

Então, combinando as duas equações anteriores, vem:

$$\text{Ciclos de relógio para tratar faltas na cache} = \frac{\text{Acessos à memória}}{\text{Programa}} \times \text{Taxa de faltas} \times \text{Penalidade por faltas}$$

Que pode ser reescrita como:

$$\text{Ciclos de relógio para tratar faltas na cache} = \frac{\text{Instruções}}{\text{Programa}} \times \frac{\text{Faltas}}{\text{Instrução}} \times \text{Penalidade por faltas}$$

3. Hierarquia de Memória: memória cache

► Medidas do Desempenho da Cache

Exemplo:

Na execução do programa gcc, suponha que a taxa de faltas devidas a instruções seja de 2% e que a taxa de faltas devidas a dados seja de 4%. Se a máquina tem CPI de 2,0, sem qualquer parada, e a penalidade por faltas for de 40 ciclos de relógio para qualquer das faltas, determine quanto a máquina vai rodar mais rápido se a equiparmos com uma cache perfeita, que nunca gere faltas. Use as frequências de instruções do gcc, apresentadas no capítulo 4.

3. Hierarquia de Memória: memória cache

► Medidas do Desempenho da Cache

Solução:

$$\text{Ciclos devido a faltas de instruções} = 1 \times 2\% \times 40 = 0,80 \text{ I}$$

A frequência de loads e stores no gcc é de 36%. Assim, o número de faltas devidas a referência a dados será:

$$\text{Ciclos devido a faltas de dados} = 1 \times 36\% \times 4\% \times 40 = 0,56 \text{ I}$$

O Número total de ciclos de relógio referentes à memória parada é de

$$0,80 \text{ I} + 0,56 \text{ I} = 1,36 \text{ I}$$

Isto significa que estamos gastando mais de um ciclo de relógio por instrução, devido às faltas geradas na cache. Por este motivo, a CPI com as paradas devidas às faltas é de $2,0 + 1,36 = 3,36$

3. Hierarquia de Memória: memória cache

► Medidas do Desempenho da Cache

Continuação da solução:

Considerando que tanto o número de instruções executadas quanto o ciclo de relógio não mudam quando consideramos a cache sem faltas, podemos calcular a razão dos tempos de execução:

$$\frac{\text{Tempo do processador com paradas}}{\text{Tempo do processador com cache perfeita}} = \frac{I \times \text{CPI}_{\text{paradas}} \times \text{ciclo de relógio}}{I \times \text{CPI}_{\text{perfeitas}} \times \text{ciclo de relógio}} = \frac{I \times \text{CPI}_{\text{paradas}}}{I \times \text{CPI}_{\text{perfeitas}}}$$
$$= \frac{3,36}{2} = 1,68$$

3. Hierarquia de Memória: memória cache

► Mapeamentos de Cache Mais Flexíveis (Visando Reduzir a Taxa de Faltas)

Mapeamento Direto (estudado até aqui):

- Um bloco só pode ser colocado **em exatamente um lugar na cache**
- “Existe um caminho, mapeado previamente, de qualquer endereço de bloco da memória principal para uma única posição da cache”
- Na verdade, existem diversos tipos de mapeamento
- O mapeamento direto é um dos extremos

3. Hierarquia de Memória: memória cache

► Mapeamentos de Cache Mais Flexíveis

Mapeamento Totalmente Associativo

- É o outro extremo dos tipos de mapeamento
- Um bloco da memória principal pode ser colocado em qualquer posição da cache
- Um bloco de memória pode ser associado a qualquer “entrada” da cache
- **É preciso pesquisar todas as entradas da cache**, a fim de localizar um determinado bloco (pois ele pode estar em qualquer uma das entradas)
- Para reduzir o tempo da pesquisa, ela é feita em paralelo, usando **um comparador para cada entrada da cache**
- O custo do hardware é alto: tal esquema somente é atrativo para caches pequenas

3. Hierarquia de Memória: memória cache

▶ Mapeamentos de Cache Mais Flexíveis

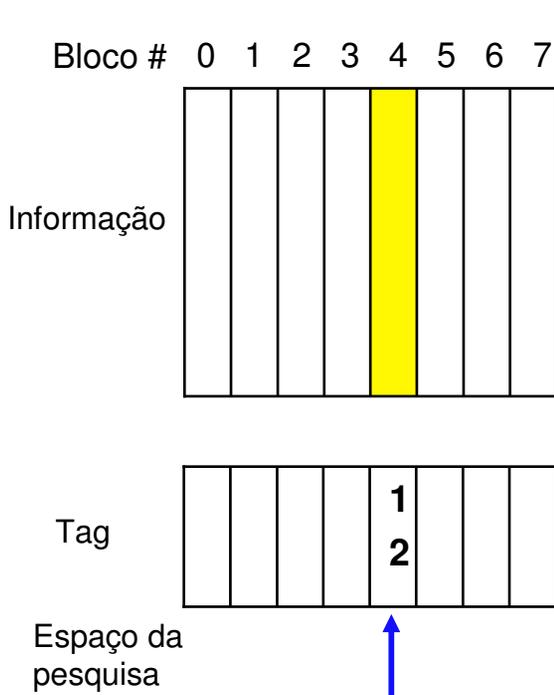
Mapeamento Associativo por Conjunto (ou Bloco Associativo)

- Fica entre os dois tipos de mapeamento anteriores
- Existe um número fixo de posições na cache (no mínimo duas) nas quais cada bloco pode ser colocado
- Cache associativa n : é uma cache associativa por conjunto com n posições possíveis para guardar um bloco
- Tal cache é composta por um certo número de conjuntos, cada conjunto com n blocos
- Cada bloco da memória principal é mapeado para um dos conjuntos da cache, determinado pelo campo de índice do endereço (sendo que o bloco pode ser colocado em qualquer dos elementos desse conjunto)
- O conjunto que contém o bloco de memória é dado por
(Número do bloco) módulo (Número de conjuntos da cache)

3. Hierarquia de Memória: memória cache

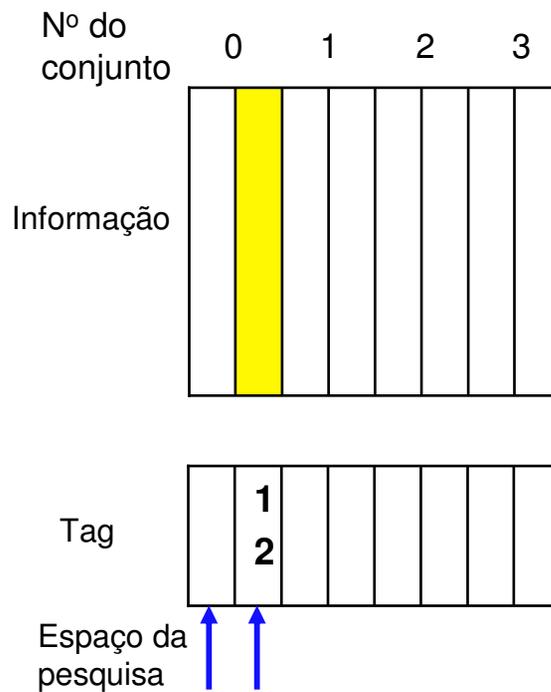
► Mapeamentos de Cache Mais Flexíveis

Cache Mapeada Diretamente



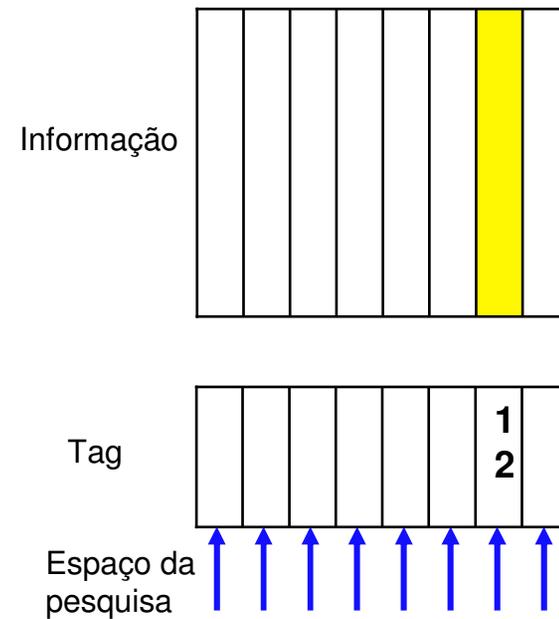
$$12 \text{ Modulo } 8 = 4$$

Cache Associativa por Conjunto (c/ 2 posições)



$$12 \text{ Modulo } 4 = 0$$

Cache Totalmente Associativa



3. Hierarquia de Memória: memória cache

► Mapeamentos de Cache Mais Flexíveis

- Podemos encarar as estratégias (políticas) de colocação de blocos na cache como variações da estratégia associativa por conjunto

3. Hierarquia de Memória: memória cache

► Mapeamentos de Cache Mais Flexíveis

Associativa por conjunto, de uma posição (Direta)

bloco	Tag	Info
0		
1		
2		
3		
4		
5		
6		
7		

Associativa por conjunto, de 2 posições

Con-junto	Tag	Info	Tag	Info
0				
1				
2				
3				

Associativa por conjunto, de 4 posições

Con-junto	Tag	Info	Tag	Info	Tag	Info	Tag	Info
0								
1								

Associativa por conjunto, de 8 posições (neste caso, Totalmente Associativa)

Tag	Info														

3. Hierarquia de Memória: memória cache

▶ Associatividade nas Caches

- O aumento do grau de associatividade resulta, em geral, na **redução da taxa de faltas**
- A desvantagem é o **aumento do tempo de tratamento do acerto**

Exemplo:

Temos a nossa disposição três pequenas caches, cada qual com quatro blocos de uma palavra:

- uma mapeada diretamente,
- outra associativa por conjunto de 2 posições
- E outra associativa por conjunto de 4 posições (totalmente associativa)

Encontre o número de faltas em cada uma delas, considerando a seguinte seqüência de endereços de blocos: 0, 8, 0, 6, 8.

3. Hierarquia de Memória: memória cache

▶ Faltas e Associatividade nas Caches

Solução para a seguinte seqüência de referências à memória: 0, 8, 0, 6, 8

Na cache mapeada diretamente, os blocos serão mapeados para as seguintes entradas da cache:

Endereço do Bloco	Bloco da Cache
0	(0 módulo 4)= 0
6	(6 módulo 4)= 2
8	(8 módulo 4)= 0

Acompanhando o comportamento da cache nas referências à memória...

Endereço do bloco de memória acessado	Acerto ou falta	Conteúdo dos Blocos da Cache Após Referência			
		0	1	2	3
0	falta	Mem[0]			
8	falta	Mem[8]			
0	falta	Mem[0]			
6	falta	Mem[0]		Mem[6]	
8	falta	Mem[8]		Mem[6]	

Conclusão: 5 faltas para 5 referências

3. Hierarquia de Memória: memória cache

▶ Faltas e Associatividade nas Caches

Na cache associativa por conjunto de 2 posições, os blocos serão mapeados para os seguintes conjuntos da cache:

Endereço do Bloco	Conjunto da Cache
0	$(0 \text{ módulo } 2) = 0$
6	$(6 \text{ módulo } 2) = 0$
8	$(8 \text{ módulo } 2) = 0$

Mem[8] foi substituído por ser o bloco menos usado recentemente (esquema LRU)

Acompanhando o comportamento da cache nas referências à memória...

Endereço do bloco de memória acessado	Acerto ou falha	Conteúdo dos Blocos da Cache Após Referência			
		Conjunto 0		Conjunto 1	
0	falta	Mem[0]			
8	falta	Mem[0]	Mem[8]		
0	acerto	Mem[0]	Mem[8]		
6	falta	Mem[0]	Mem[6]		
8	falta	Mem[8]	Mem[6]		

Conclusão: 4 faltas para 5 referências

3. Hierarquia de Memória: memória cache

► Faltas e Associatividade nas Caches

Na cache associativa por conjunto de 4 posições (i.e., totalmente associativa), os blocos podem ser mapeados para qualquer posição da cache

Acompanhando o comportamento da cache nas referências à memória...

Endereço do bloco de memória acessado	Acerto ou falha	Conteúdo dos Blocos da Cache Após Referência			
		Bloco 0	Bloco 1	Bloco 2	Bloco 3
0	falta	Mem[0]			
8	falta	Mem[0]	Mem[8]		
0	acerto	Mem[0]	Mem[8]		
6	falta	Mem[0]	Mem[8]	Mem[6]	
8	acerto	Mem[0]	Mem[8]	Mem[6]	

Conclusão: 3 faltas para 5 referências

3. Hierarquia de Memória: memória cache

▶ Taxa de Faltas e Associatividade nas Caches

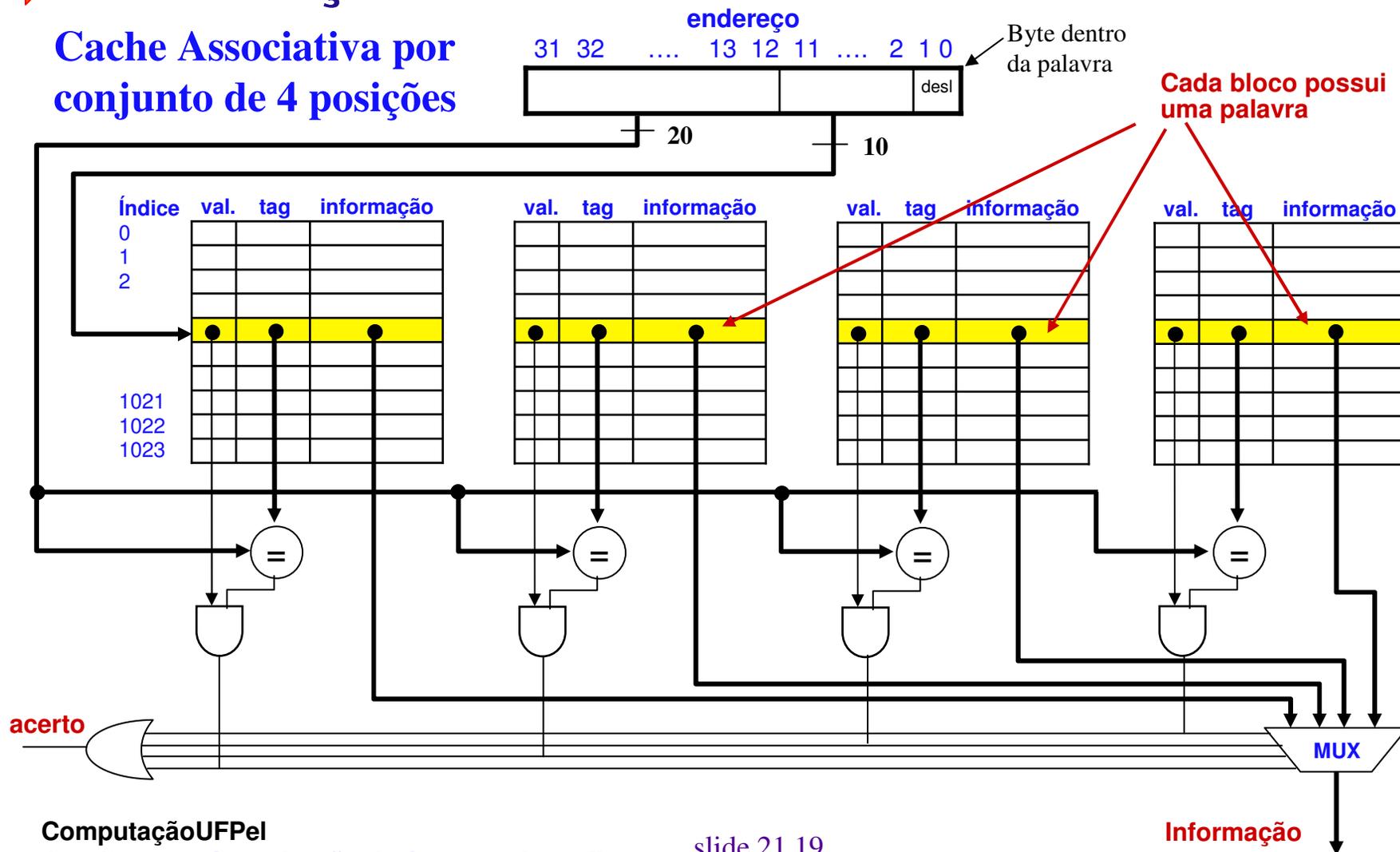
Taxa de Faltas para o gcc e o spice em uma máquina que possui cache de dados de 64Kb e cache de instruções de 64KB (e blocos de quatro palavras)

Programa	Associatividade	Taxa de Faltas devidas a instruções	Taxa de Faltas devidas a dados	Taxa de Faltas combinada
gcc	1	2,0%	1,7%	1,9%
gcc	2	1,6%	1,4%	1,5%
gcc	4	1,6%	1,4%	1,5%
spice	1	0,3%	0,6%	0,4%
spice	2	0,3%	0,6%	0,4%
spice	4	0,3%	0,6%	0,4%

3. Hierarquia de Memória: memória cache

Localização de um Bloco na Cache

Cache Associativa por conjunto de 4 posições



acerto

3. Hierarquia de Memória: memória cache

▶ Localização de um Bloco na Cache

- **Total de bits do endereço: 32**
- **Memória endereçada a bytes: isto quer dizer que:**
 - usando-se **32 bits** no acesso à memória, obtém-se um byte (pois cada endereço será o endereço de um byte)
 - usando-se os **30 bits** mais significativos no acesso à memória, obtém-se uma palavra de **32 bits** (pois cada endereço será o endereço de uma palavra)

3. Hierarquia de Memória: memória cache

► Localização de um Bloco na Cache

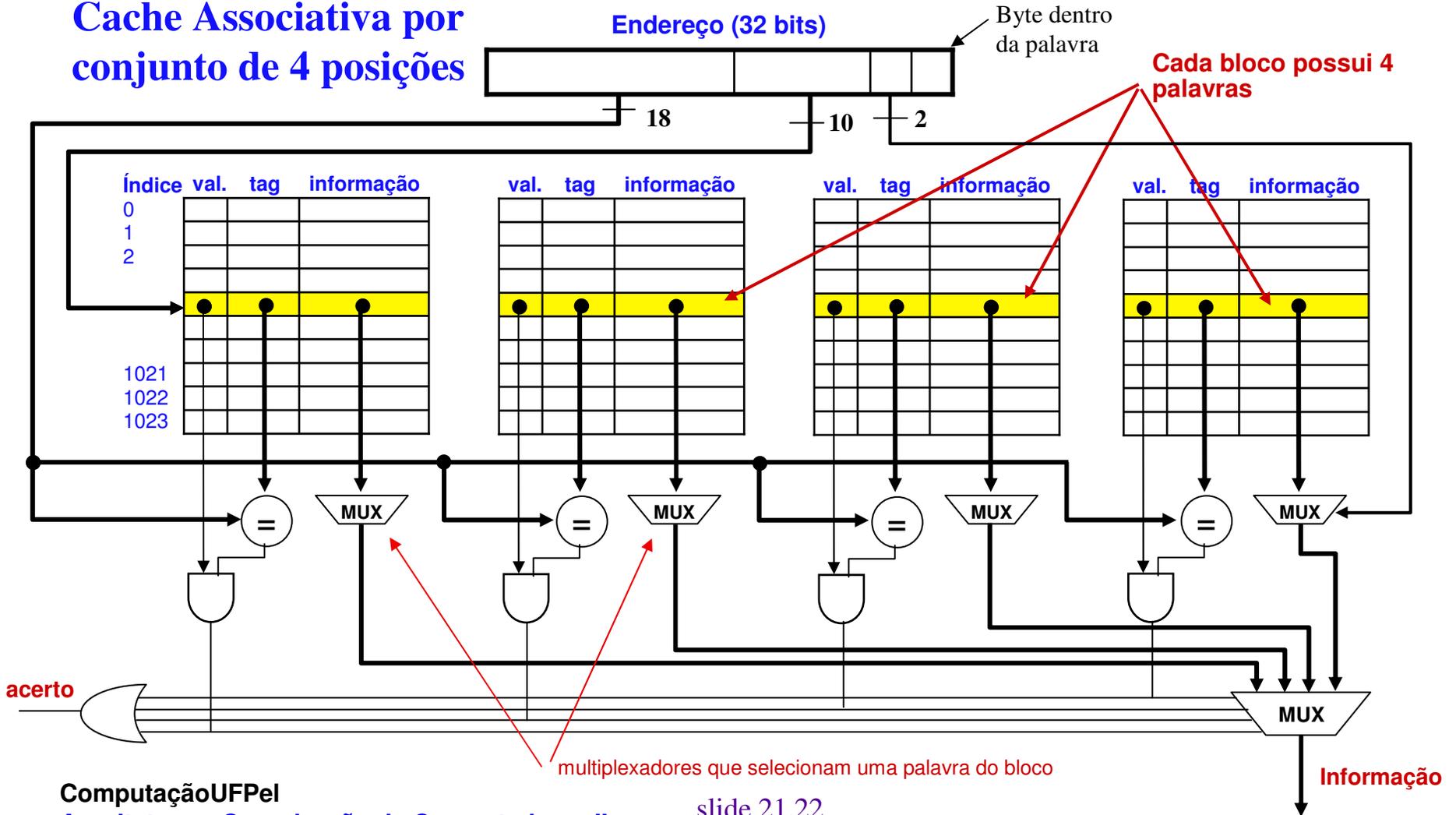
Na cache mostrada na transparência 15.19:

- 2 bits (bits 0-1) → byte dentro da palavra
- 10 bits (bits 2-11) → índice da cache, ou seja, número de entradas na cache
- 20 bits (bits 12-31) → rótulo (tag): precisa ser armazenado na cache
- Tamanho da cache:
 - Cada linha de um conjunto tem: 32 bits (1 bloco possui uma palavra) + 20 (tag) + 1 (bit de validade) = 53 bits
 - A cache tem $2^{10} = 1024$ entradas = 1K entradas
 - 4 conjuntos
 - **Tamanho total= 53 x 4 x 1K** (dos quais 32 x 4 x 1K para informação)

3. Hierarquia de Memória: memória cache

Localização de um Bloco na Cache

Cache Associativa por conjunto de 4 posições



3. Hierarquia de Memória: memória cache

► Localização de um Bloco na Cache

Na cache mostrada na transparência anterior:

- **2 bits (bits 0-1) → byte dentro da palavra**
- **2 bits (bits 2-3) → para selecionar uma palavra dentro do bloco**
- **10 bits (bits 4-13) → índice da cache, ou seja, número de entradas na cache**
- **18 bits (bits 14-31) → rótulo (tag): precisa ser armazenado na cache**
- **Tamanho da cache:**
 - **Cada linha de um conjunto tem: 128 bits (1 bloco possui 4 palavras) + 18 (tag) + 1 (bit de validade) = 147 bits**
 - **A cache tem $2^{10}=1024$ entradas = 1K entradas**
 - **4 conjuntos**
 - **Tamanho total= 147 x 4 x 1K (dos quais 128 x 4 x 1K para informação)**

3. Hierarquia de Memória: memória cache

▶ Tamanho dos Rótulos versus Associatividade

Exemplo:

Supondo processador com endereços de 32 bits e cache com 4K blocos. Encontrar o número total de conjuntos e o número total de bits do tag (rótulo) considerando:

- 1. Mapeamento direto**
- 2. Associativo por conjunto de 2 e por conjunto de 4**
- 3. Totalmente associativo**

3. Hierarquia de Memória: memória cache

▶ Tamanho dos Rótulos versus Associatividade

1. Mapeamento direto

Em uma cache mapeada diretamente cada conjunto tem exatamente um bloco.

Logo, são 12 bits para o índice ($2^{12} = 4K$) e o número total de bits para o tag é $(32 - 12) \times 4K = 80$ Kbits.

3. Hierarquia de Memória: memória cache

► Tamanho dos Rótulos versus Associatividade

2. Associativo por conjunto de 2 e por conjunto de 4

- Cada grau de associatividade a mais diminui o número de conjuntos de um fator de 2
- Portanto, diminui em 1 unidade o número de bits necessários para indexar a cache, aumentando em 1 unidade o número de bits do rótulo

Cache associativa por conjunto de 2, existem 2K conjuntos. O número total de bits para o tag é $(32 - 11) \times 2 \times 2K = 84$ Kbits.

Cache associativa por conjunto de 4, existem 1K conjuntos. O número total de bits para o tag é $(32 - 10) \times 4 \times 1K = 88$ Kbits.

3. Hierarquia de Memória: memória cache

▶ Tamanho dos Rótulos versus Associatividade

3. Totalmente associativo

Em uma cache totalmente associativa existe um único conjunto, no caso com 4K blocos, com tags (rótulos) de 32 bits.

O número total de bits para o tag é $32 \times 4K \times 1 = 128$ Kbits.

A escolha entre um mapeamento direto, associativo por conjunto ou totalmente associativo irá depender do custo de uma falta versus o custo de implementação da associatividade, tanto em termos de tempo quanto em termos de hardware.

3. Hierarquia de Memória: memória cache

▶ Seleção do Bloco a Ser Substituído

Cache com Mapeamento Direto:

- Não há seleção: o bloco novo ocupa uma entrada pré-definida, sobrescrevendo o bloco que lá se encontrava...

Caches com Mapeamento Associativo

- É preciso selecionar o bloco que será sobrescrito (“substituído”)
- LRU (*Least Recently Used*) é o algoritmo mais utilizado. Ele escolhe o bloco que não é usado há mais tempo

3. Hierarquia de Memória: memória cache

► Caches Multinível

- A maioria dos computadores possuem ao menos dois níveis de cache:
 - L1 (level 1), integrada no microprocessador
 - L2 (level 2), fora do processador, em *chips* de memória DRAM
 - capacidade(L2) > capacidade(L1)
 - tempo de acesso(L2) > tempo de acesso(L1)
- Atualmente, computadores com alto desempenho podem possuir 3 níveis de cache:
 - L1 (level 1) e L2 (level 2), integradas no microprocessador
 - L3 (level 3), fora do processador, em *chips* de memória DRAM
 - capacidade(L3) > capacidade(L2) > capacidade(L1)
 - tempo de acesso(L3) > tempo de acesso(L2) > tempo de acesso(L1)