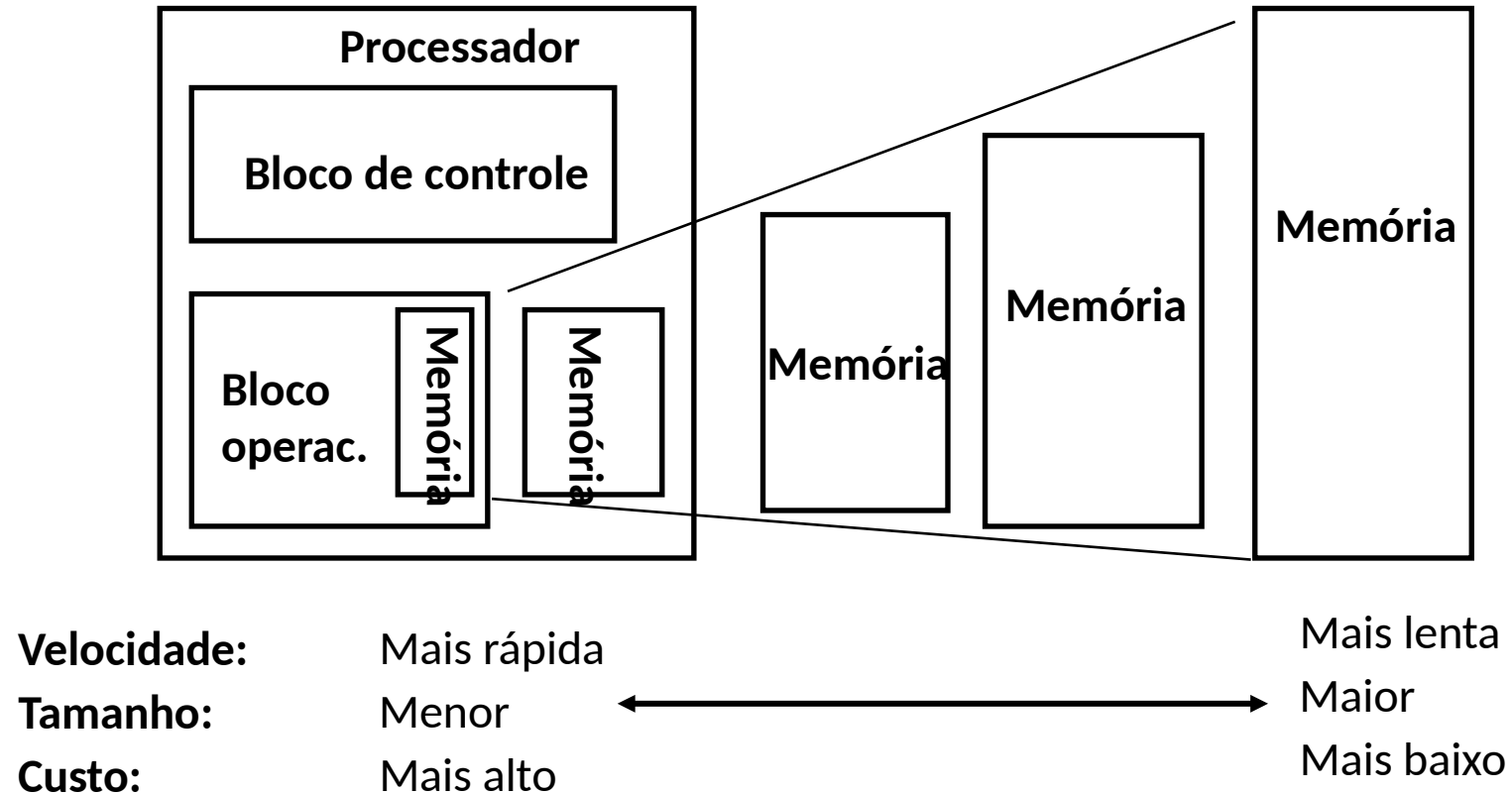


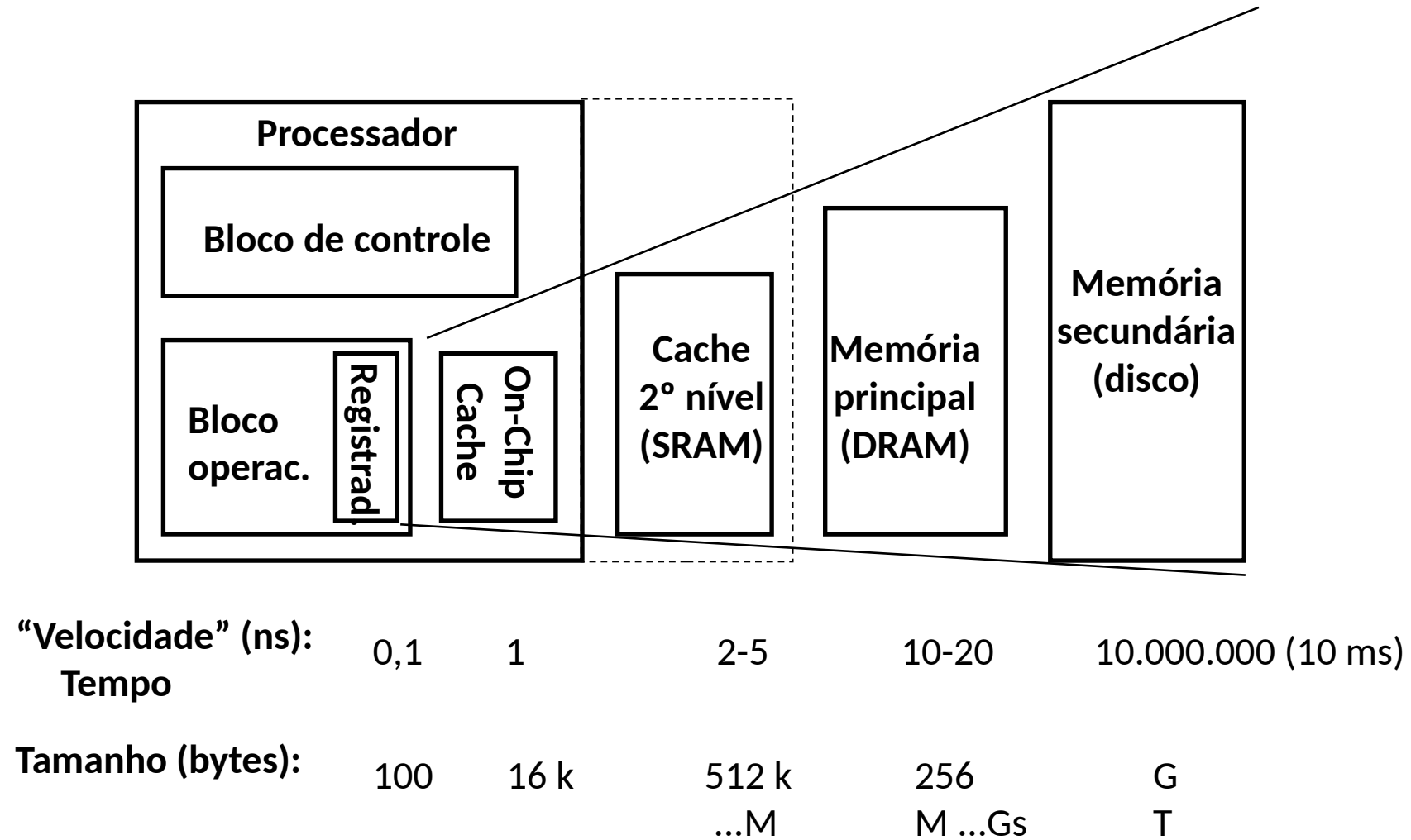
Arquitetura de Computadores III

Hierarquia de memória Caches

Hierarquia de Memória



Hierarquia de Memória



Hierarquia de Memória

Princípio da Localidade

- Espacial: se um dado é referenciado, seus vizinhos tendem a ser referenciados logo.
- Temporal: um dado referenciado, tende a ser referenciado novamente.

Como explorar o princípio de localidade numa hierarquia de memória?

- Localidade Temporal
 - => Mantenha itens de dados mais recentemente acessados nos níveis da hierarquia mais próximos do processador
- Localidade Espacial
 - => Mova blocos de palavras contíguas para os níveis da hierarquia mais próximos do processador

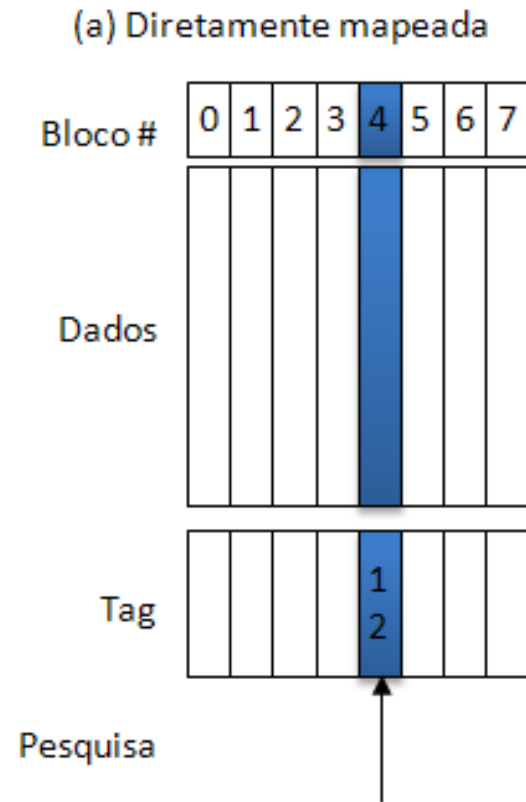
Organizações de Memória Cache

- Processador gera endereço de memória e o envia à cache
- Cache deve
 - verificar se tem cópia da posição de memória correspondente
 - se tem, encontrar a posição da cache onde está esta cópia
 - se não tem, trazer o conteúdo da memória principal e escolher posição da cache onde a cópia será armazenada
- Mapeamento entre endereços de memória principal e endereços de cache resolve estas 3 questões
 - deve ser executado em hardware
- Estratégias de organização (mapeamento) da cache
 - mapeamento completamente associativo
 - mapeamento direto
 - mapeamento set-associativo

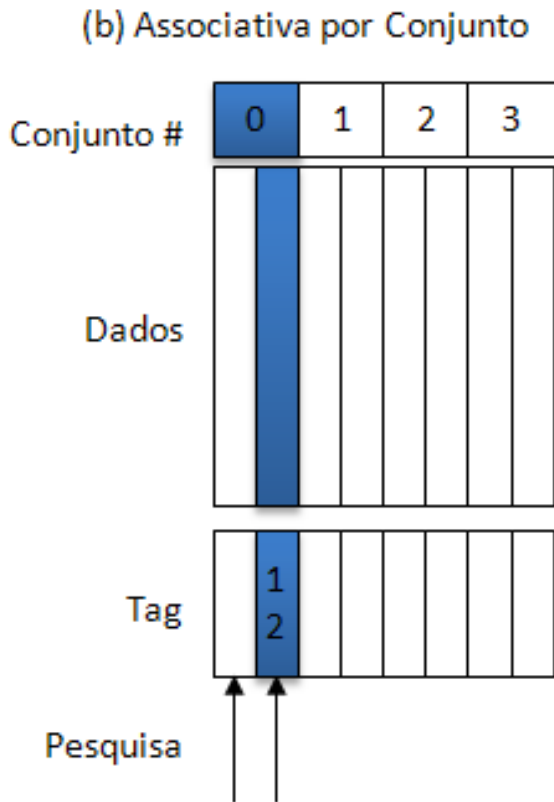
Memórias Cache

- Características
 - Pouco espaço de armazenamento
 - Alto custo financeiro
 - Baixo tempo de acesso
- Conceitos
 - Palavra: conjunto de um ou mais bytes.
 - Bloco: conjunto de uma ou mais palavras (unidade da cache)
 - Bit de Válido: indica se o dado ou bloco está válido
 - Tag ou rótulo: parte do endereço de uma palavra na memória principal
 - Slot: cada linha de uma cache, que pode armazenar um ou mais blocos dependendo da organização da cache.
 - Comparador: compara a tag de um endereço de uma palavra, com as tags dos endereços armazenados na cache

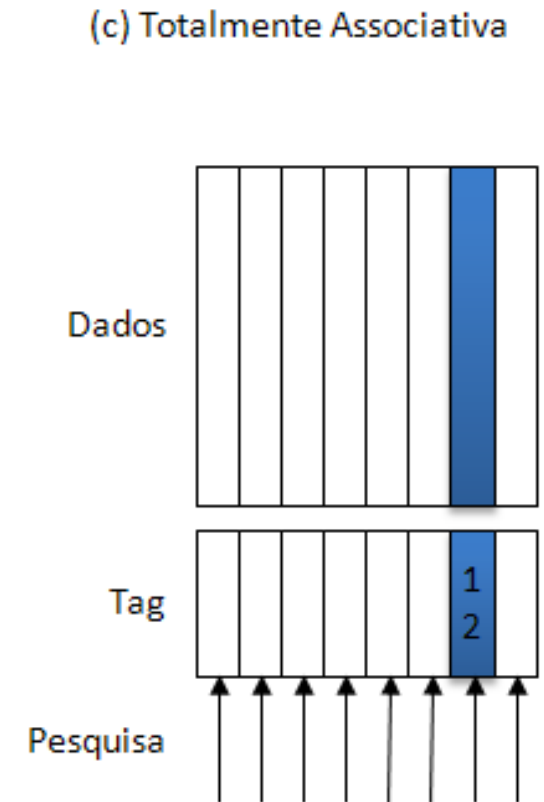
Modos de Mapeamento



$12 \text{ modulo } 8 = 4$
Bloco 4

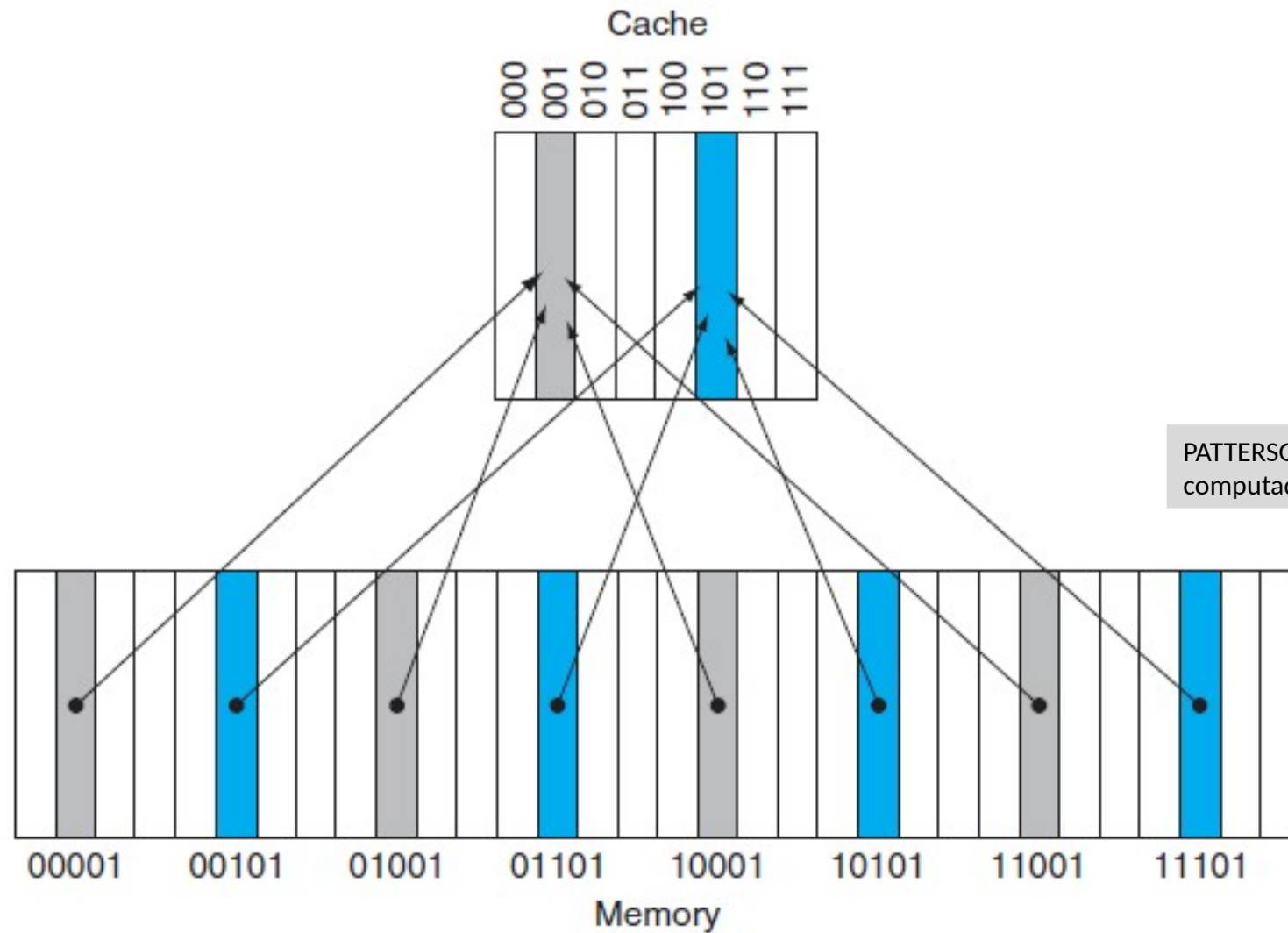


$12 \text{ modulo } 4 = 0$
Bloco 0



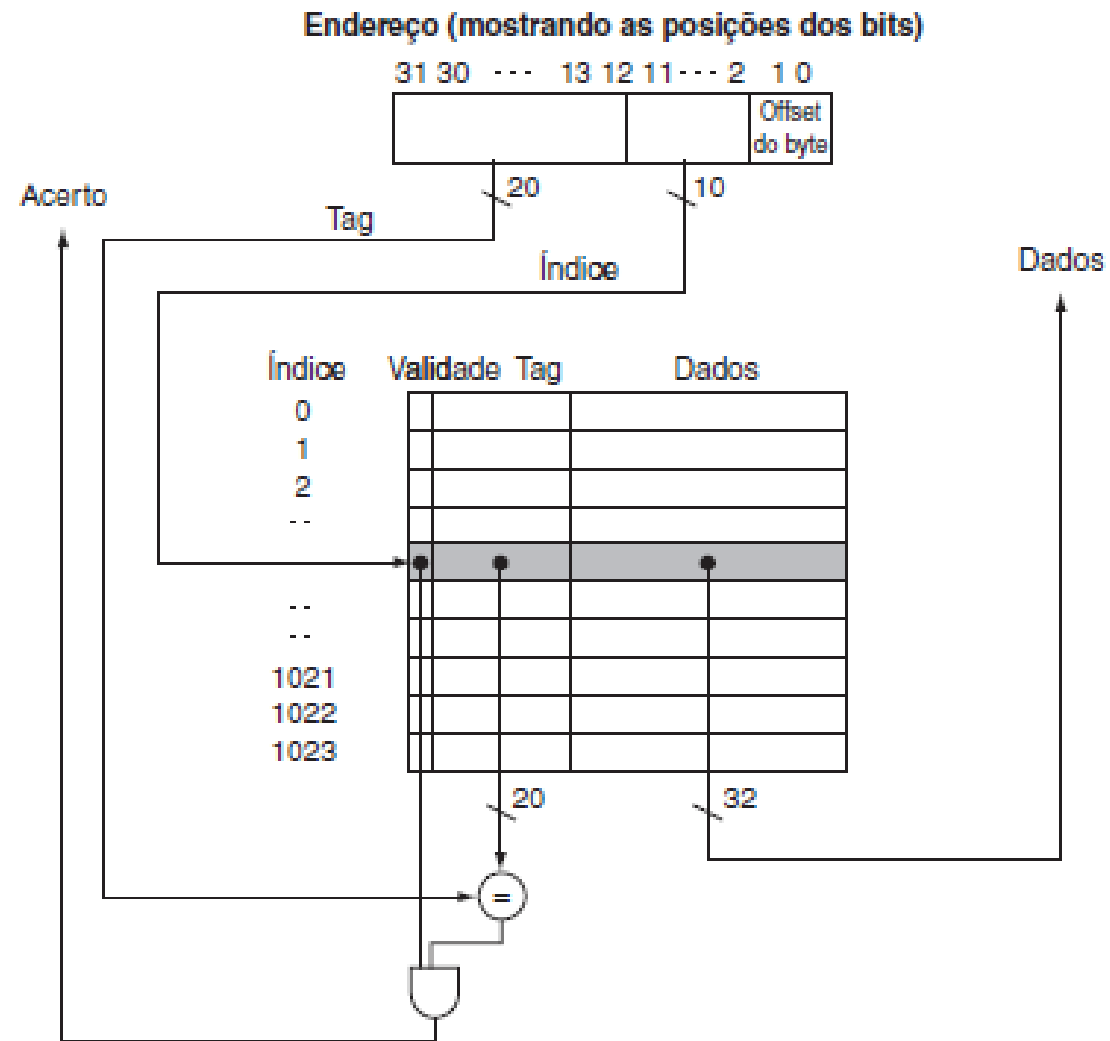
Qualquer end.

Mapeamento Direto



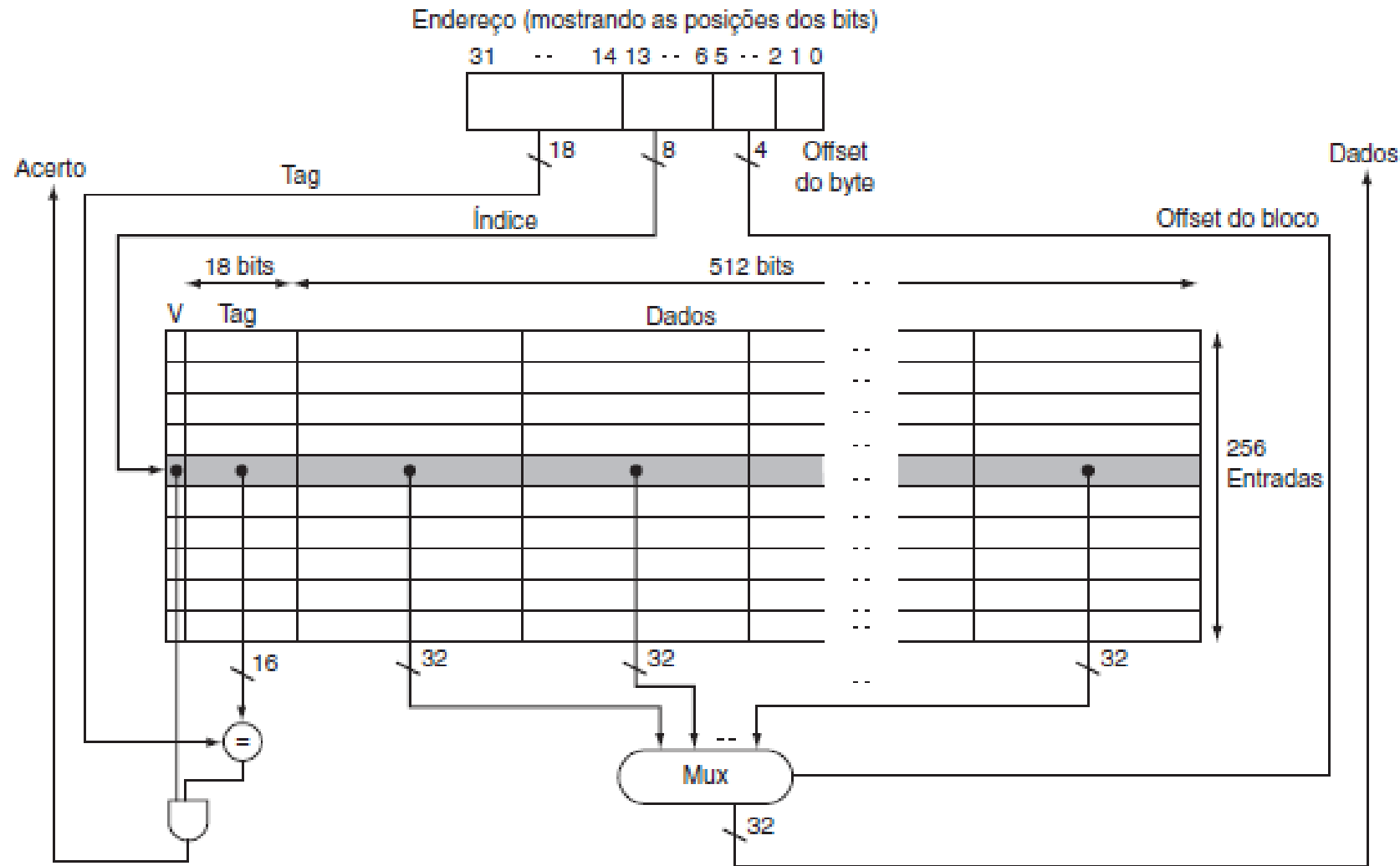
PATTERSON, David A. Organização e projeto de computadores a interface hardware/software

Tamanho da linha



PATTERSON, David A. Organização e projeto de computadores a interface hardware/software

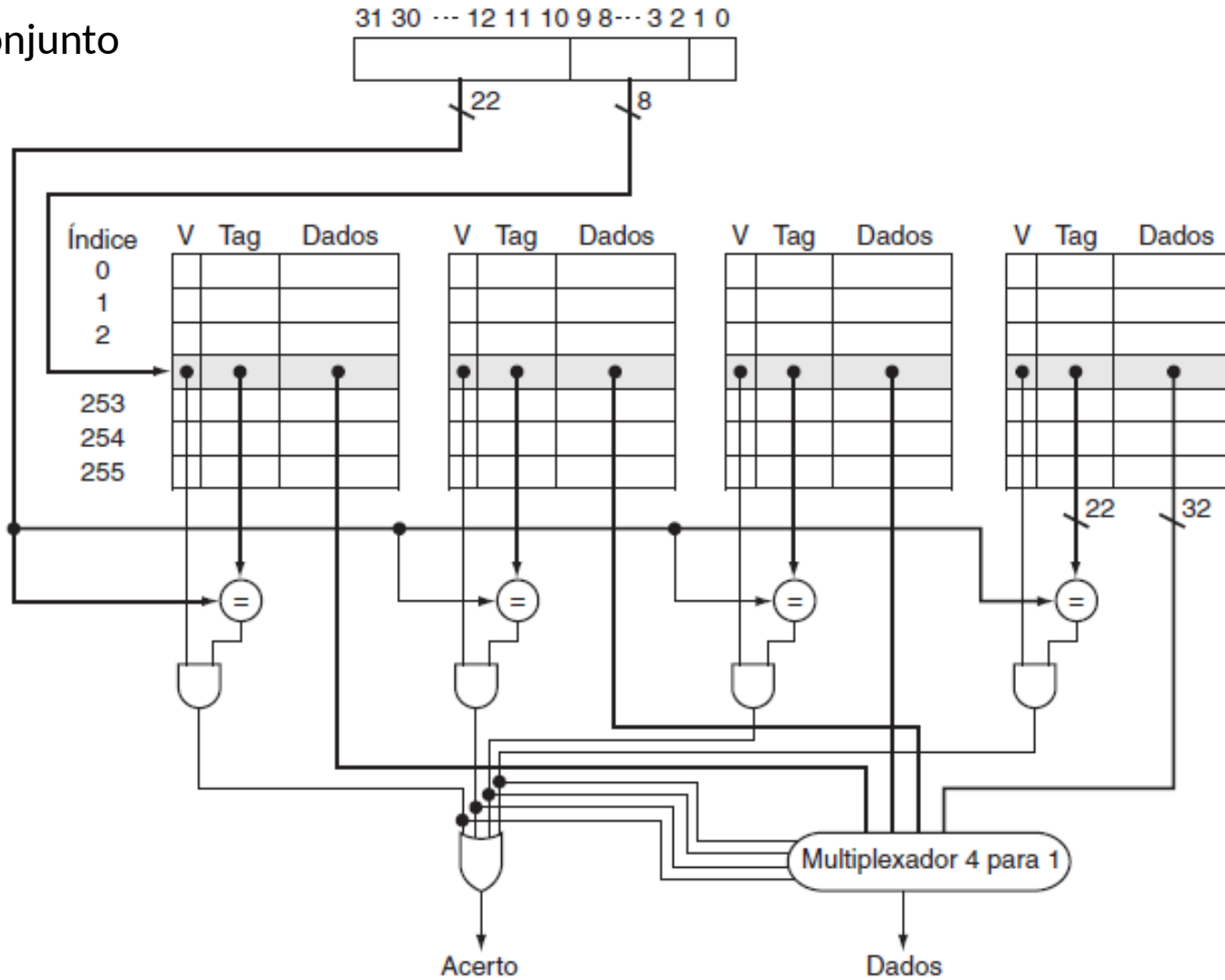
Tamanho da linha tirando vantagem da localidade espacial



Organizações da Cache

Associativa por conjunto

4 vias



256 conjuntos
1 possui 4 vias
1 via = $1 + 2^2 + 3^2 = 55$ bits

256 conjuntos x 4 = 1024 linhas

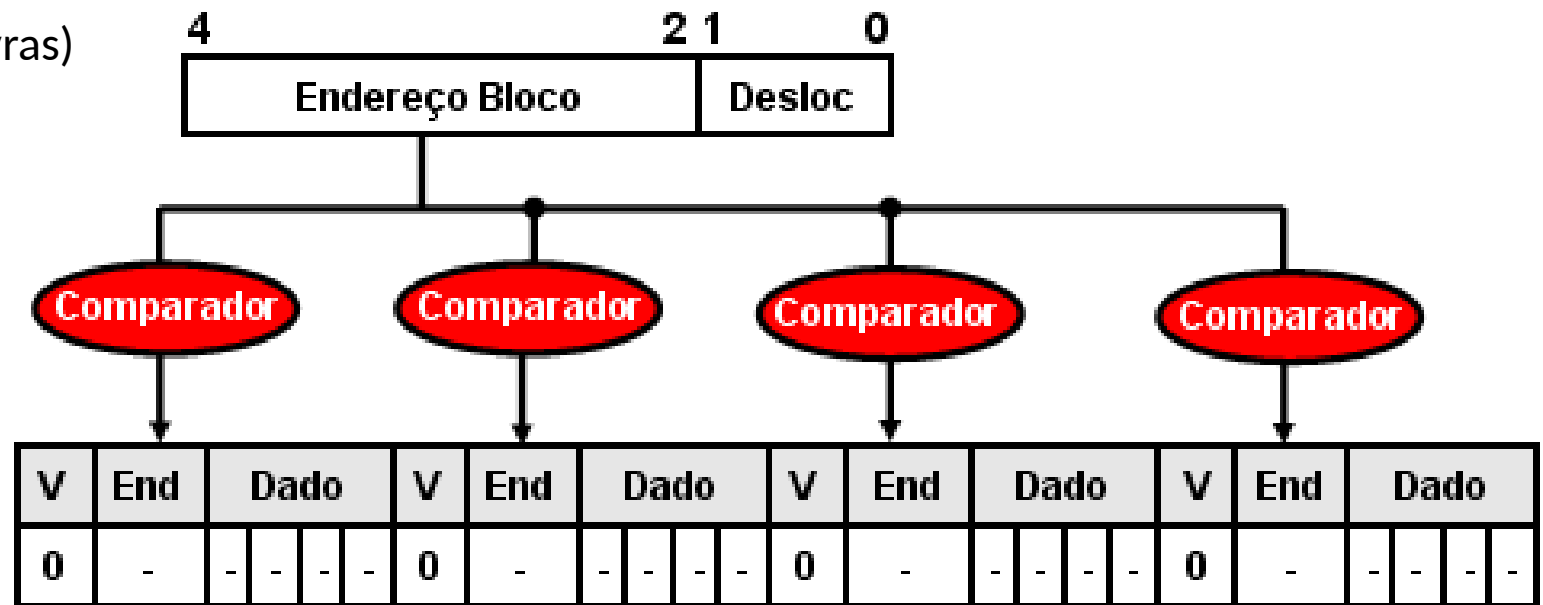
$$55 \times 1024 = 55 \text{ kb}$$

PATTERSON, David A. Organização e projeto de computadores a interface hardware/software

Organizações da Cache

Completamente Associativa

- Possui um slot com N blocos
- Necessita de N comparações
- Endereça o bloco diretamente
- Exemplo:
 - Endereços de 5 bits (32 palavras)
 - 1 slot (com 4 blocos)
 - Blocos de 4 palavras



Caches com o mesmo tamanho de dados

**One-way set associative
(direct mapped)**

Block	Tag	Data
0		
1		
2		
3		
4		
5		
6		
7		

Two-way set associative

Set	Tag	Data	Tag	Data
0				
1				
2				
3				

Four-way set associative

Set	Tag	Data	Tag	Data	Tag	Data	Tag	Data
0								
1								

Eight-way set associative (fully associative)

Tag	Data	Tag	Data	Tag	Data	Tag	Data	Tag	Data	Tag	Data	Tag	Data	Tag	Data

PATTERSON, David A. Organização e projeto de computadores a interface hardware/software

Acesso à Cache

- Quando a cache estiver sem espaço, qual bloco será substituído?
 - Mapeamento Direto: o bloco que estiver no slot
 - Associativa por conjunto e Completamente Associativa: usar uma política de substituição
 - LRU: substituir o bloco menos recentemente utilizado
 - Item (endereço de linha) vai para a frente da lista
 - LFU: substituir o bloco menos frequentemente utilizado
 - Contador incrementado quando bloco é acessado
 - FIFO: substituir o primeiro bloco que entrou na cache
 - Aleatório: escolher um bloco qualquer

Mapeamento Direto

0000 **0** miss

00	Mem(0)

0001 **1** miss

00	Mem(0)
00	Mem(1)

0010 **2** miss

00	Mem(0)
00	Mem(1)
00	Mem(2)

0011 **3** miss

00	Mem(0)
00	Mem(1)
00	Mem(2)
00	Mem(3)

0100 **4** miss

01

00	Mem(0)
00	Mem(1)
00	Mem(2)
00	Mem(3)

4

0011 **3** hit

01	Mem(4)
00	Mem(1)
00	Mem(2)
00	Mem(3)

0100 **4** hit

01	Mem(4)
00	Mem(1)
00	Mem(2)
00	Mem(3)

1111 **15** miss

11

01	Mem(4)
00	Mem(1)
00	Mem(2)
00	Mem(3)

15

8 requests, 2 hits

Completamente Associativo

000 0

000	Mem(0)

010 2

000	Mem(0)
010	Mem(2)

001 1

000	Mem(0)
010	Mem(2)
001	Mem(1)

011 3

000	Mem(0)
010	Mem(2)
001	Mem(1)
011	Mem(3)

100 4

100

000	Mem(0)
010	Mem(2)
001	Mem(1)
011	Mem(3)

FIFO

011 3

100	Mem(4)
010	Mem(2)
001	Mem(1)
011	Mem(3)

100 4

100	Mem(4)
010	Mem(2)
001	Mem(1)
011	Mem(3)

111 15

111

100	Mem(4)
010	Mem(2)
001	Mem(1)
011	Mem(3)

15

FIFO

8 requests, 2 hits

Acesso à Cache

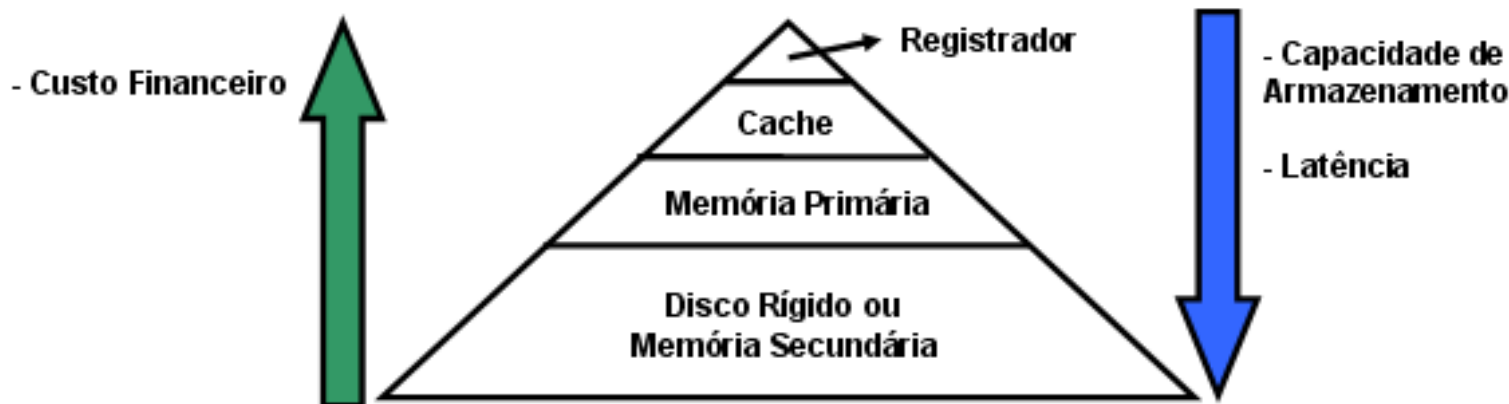
- Quando ocorrer uma escrita, como manter a coerência com a memória principal?
 - Write-through: a palavra é escrita tanto no bloco da cache, quanto no bloco da memória principal.
 - Write-back: a palavra é escrita somente no bloco da cache. Quando este bloco for substituído, então a palavra será escrita na memória principal.

Cache hit e Cache miss

- Cache Hit: acerto na cache, ou seja, o dado procurado já se encontra carregado na cache
 - **Hit Time**: tempo de acesso ao nível **superior**, que consiste de tempo de acesso + tempo para determinar hit/miss
- Cache Miss: falta na cache, ou seja, o dado procurado ainda tem que ser buscado na memória principal.
 - **Miss Penalty**: tempo gasto para substituir um bloco no nível **superior** + tempo para fornecer o bloco ao processador

-Métrica

- Taxa de acerto: dado um número de acessos a cache, qual a porcentagem de cache hit.

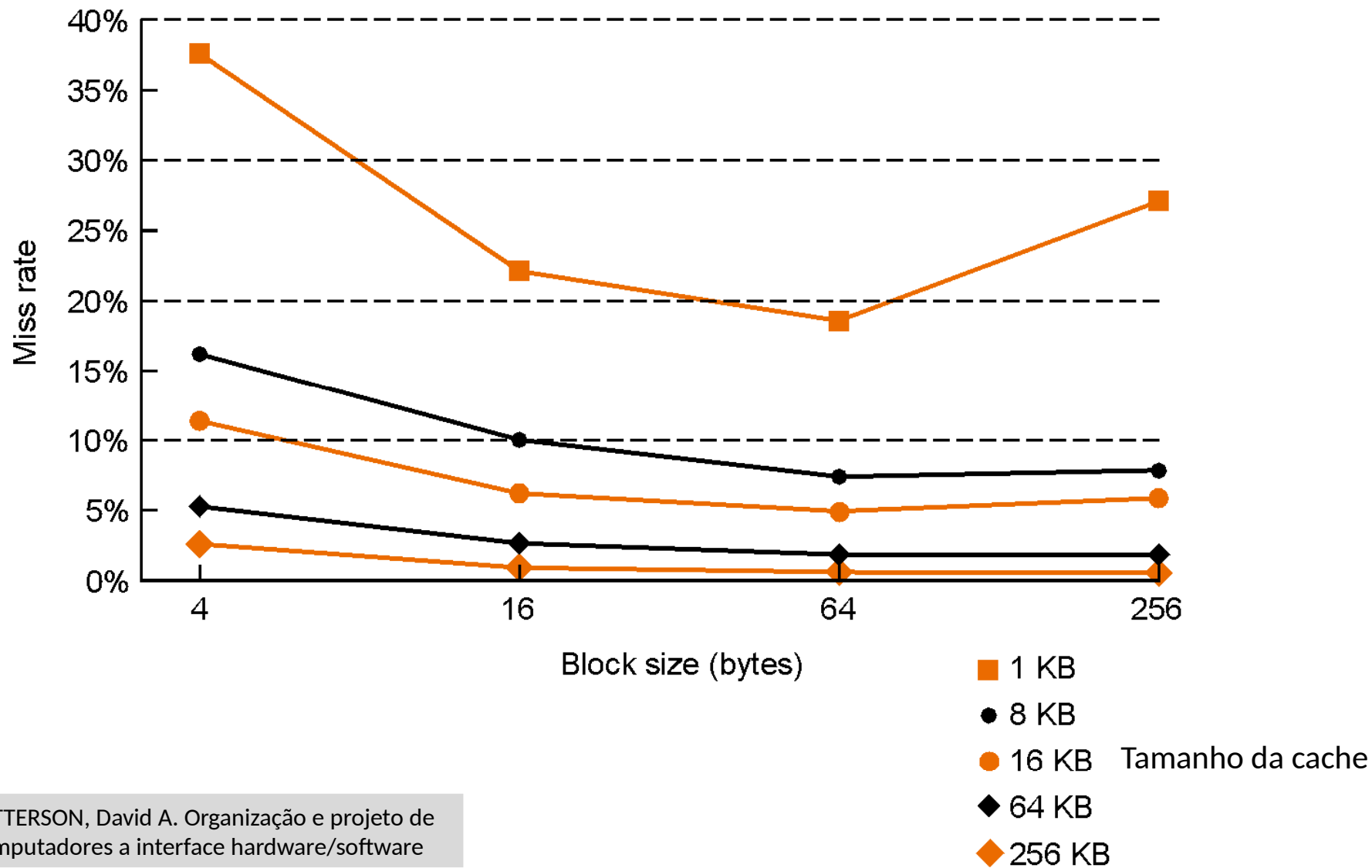


$$\text{Taxa de Acerto} = \frac{\text{Nº Cache Hit}}{\text{Nº de Acessos}}$$

Tipos de cache miss

- **compulsórios** (cold start ou chaveamento de processos, primeira referência): primeiro acesso a uma linha
 - é um “fato da vida”: não se pode fazer muito a respeito
 - se o programa vai executar “bilhões” de instruções, misses compulsórios são insignificantes
- de **conflito** (ou colisão)
 - múltiplas linhas de memória acessando o mesmo conjunto da cache conjunto-associativa ou mesma linha da cache com mapeamento direto
 - solução 1: aumentar tamanho da cache
 - solução 2: aumentar associatividade
- de **capacidade**
 - cache não pode conter todas as linhas acessadas pelo programa
 - solução: aumentar tamanho da cache
- **invalidação**: outro processo (p.ex. I/O) atualiza memória

Tamanho da linha vs. miss ratio



PATTERSON, David A. Organização e projeto de computadores a interface hardware/software

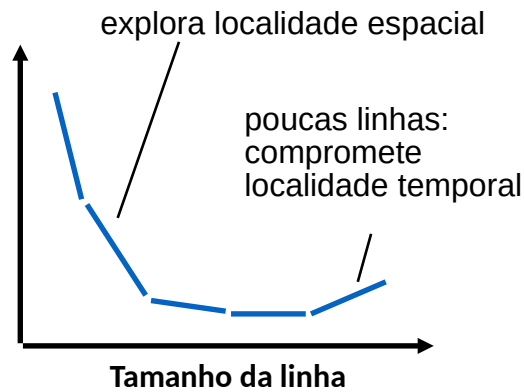
Tamanho da linha

- em geral, uma linha maior aproveita melhor a localidade espacial **MAS**
 - linha maior significa maior *miss penalty*
 - demora mais tempo para preencher a linha
 - se tamanho da linha é grande demais em relação ao tamanho da cache, *miss ratio* vai aumentar
 - muito poucas linhas
- em geral, tempo médio de acesso = Hit Time x (1 - Miss Ratio) + Miss Penalty x Miss Ratio

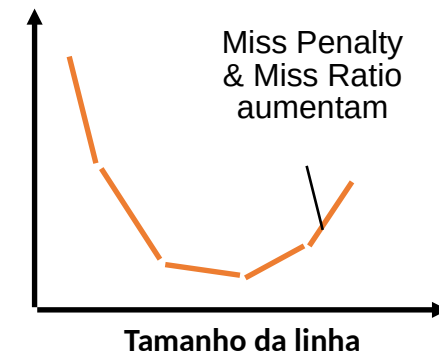
Miss Penalty



Miss Ratio




Tempo médio de acesso



Quantos bits tem a cache no total?

- supondo cache com mapeamento direto, com 64 kB de dados, linha com uma palavra de 32 bits (4 bytes), e endereços de 32 bits
- 64 kB -> 16 kpalavras, 2^{14} palavras, neste caso 2^{14} linhas
- cada linha tem 32 bits de dados mais um tag (32-14-2 bits) mais um bit de validade:
 $2^{14} \times (32 + 32 - 14 - 2 + 1) = 2^{14} \times 49 = 784 \times 2^{10} = 784 \text{ kbits}$
- 98 kB para 64 kB de dados, ou 50% a mais

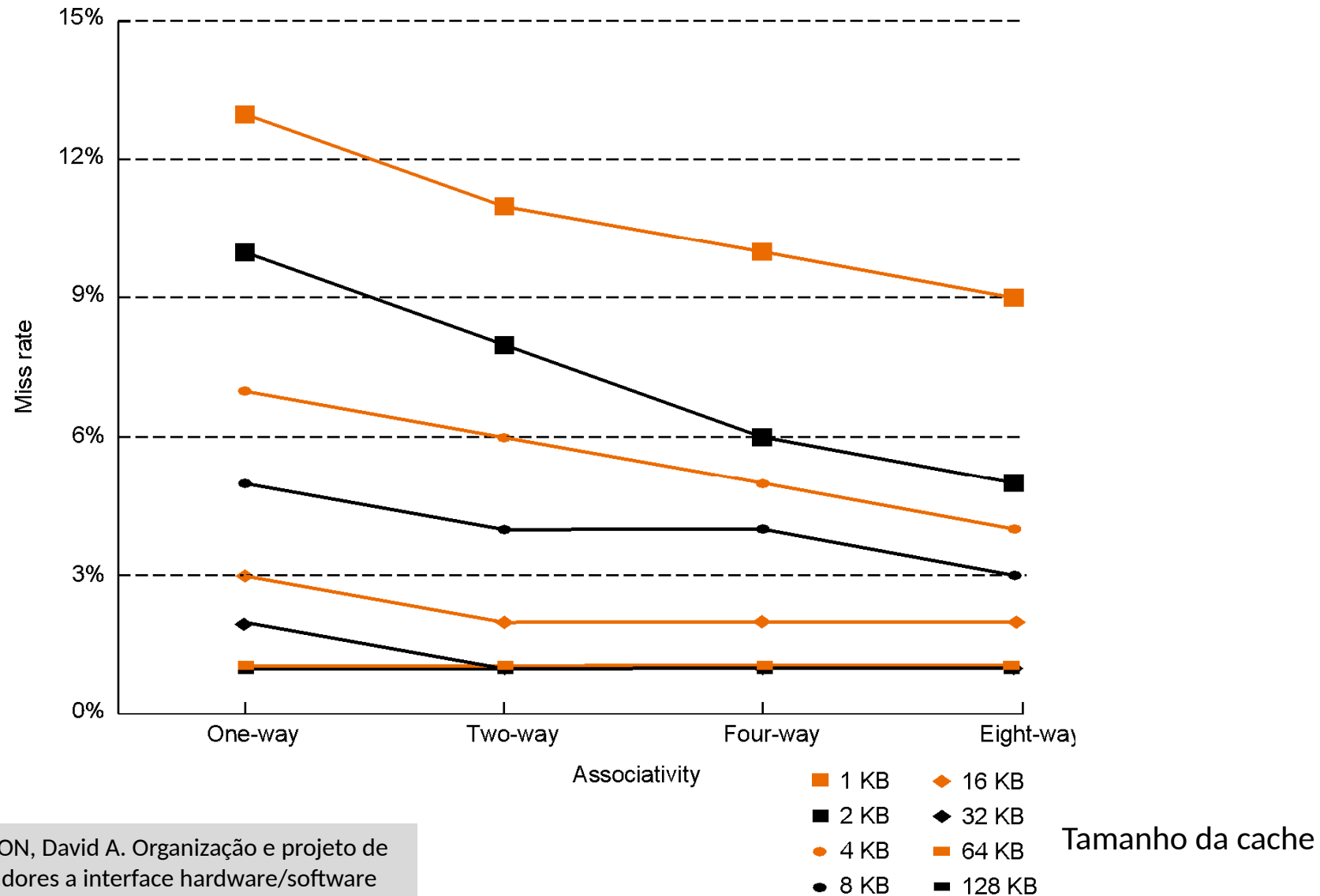
Índice e offset



Quantos bits tem a cache no total?

- supondo cache com mapeamento direto, com 16 kB de dados, blocos de 4 palavras, sendo cada palavra de 32 bits e endereços de 32 bits
- 16 kB -> 4 kpalavras, 2^{12} palavras
- Bloco de 4 palavras (2^2), 2^{10} blocos (linhas)
- cada bloco tem 32 bits x 4 = 128 bits de dados mais um tag (32-10-2-2 bits) mais um bit de validade:
$$2^{10} \times (128 + 32 - 10 - 2 - 2 + 1) = 2^{10} \times 147 = 147 \text{ kbits}$$
- 18.4 kB para 16 kB de dados, ou 15% a mais

Impacto da associatividade



PATTERSON, David A. Organização e projeto de computadores a interface hardware/software

Impacto no desempenho

- Medindo o impacto do hit ratio no tempo efetivo de acesso

T_c = tempo de acesso à memória cache

T_m = tempo de acesso à memória principal

T_{ce} = tempo efetivo de acesso à memória cache, considerando efeito dos misses

$$T_{ce} = h T_c + (1 - h) T_m$$

se $T_c = 1 \text{ ns}$, $T_m = 20 \text{ ns}$


h	$=$	0.85	0.95	0.99	1.0
		↓	↓	↓	↓
Então					
T_{ce}	$=$	3.85 ns	1.95 ns	1.19 ns	1 ns

Impacto no desempenho

- Supondo um processador que executa um programa com:
 - CPI = 1.1
 - 50% aritm/lógica, 30% load/store, 20% desvios
- Supondo que 10% das operações de acesso a dados na memória sejam misses. Cada miss resulta numa penalidade de 50 ciclos.

$$\begin{aligned}\text{CPI} &= \text{CPI ideal} + n^{\circ} \text{ médio de stalls por instrução} \\ &= 1.1 \text{ ciclos} + 0.30 \times 0.10 \times 50 \\ &= 1.1 \text{ ciclos} + 1.5 \text{ ciclos} = 2.6\end{aligned}$$

CPI ideal	1.1
Data misses	1.5
Instr.misses	0.5

- 58 % do tempo o processador está parado esperando pela memória!
- um miss ratio de 1% no  fetch de instruções resultaria na adição de 0.5 ciclos ao CPI médio

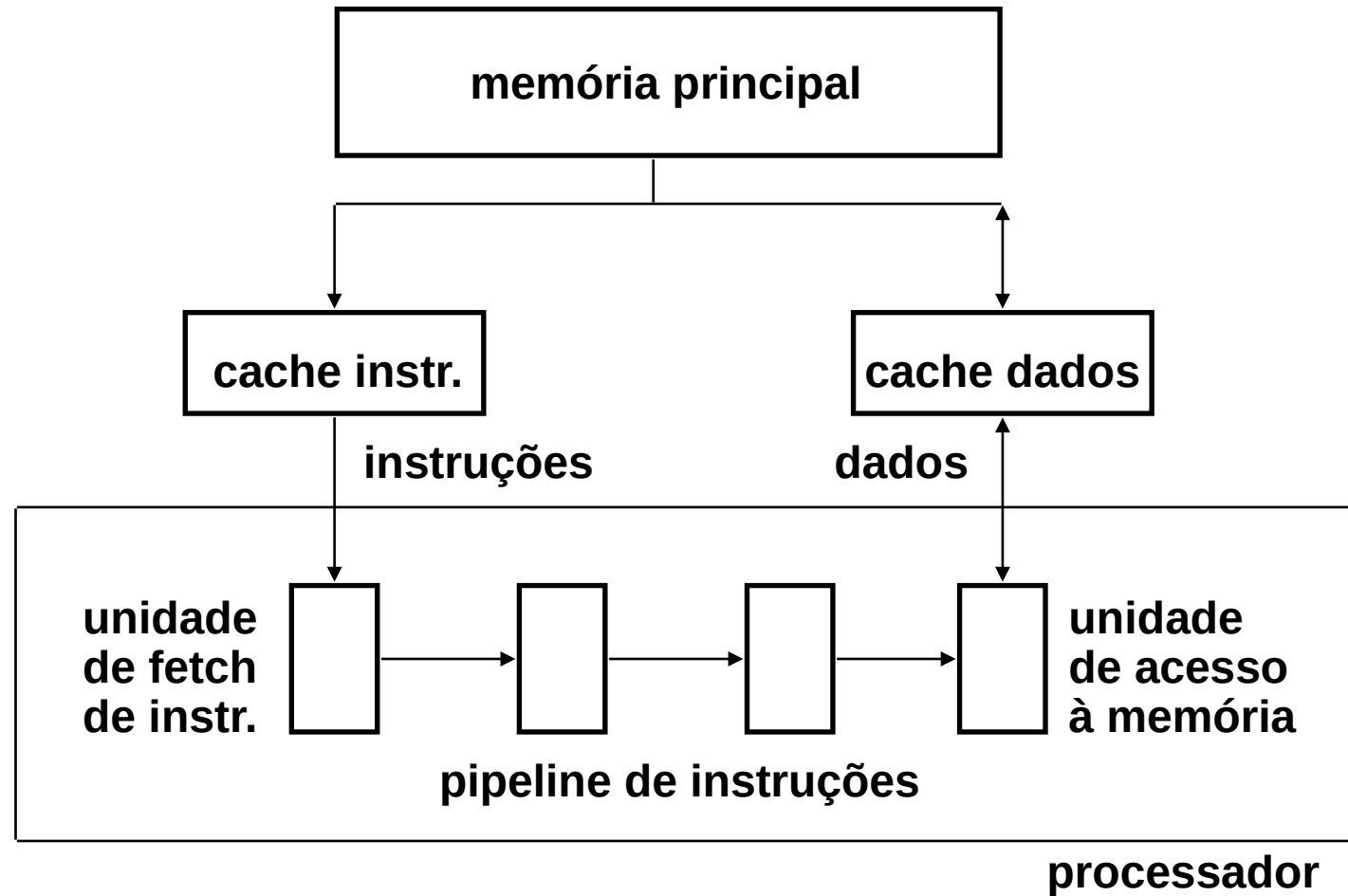
Hierarquia de caches

- caches integradas dentro de um processador têm limitação de tamanho
- miss penalty na cache é muito grande, pela diferença entre os tempos de acesso da cache e da memória principal
- solução: caches em dois ou três níveis
 - cache integrada (L1, de primeiro nível) é de tamanho pequeno, p.ex. 8 kbytes, e tempo de acesso menor
 - cache secundária (L2) tem tamanho maior, p.ex. 256 kbytes, e tempo de acesso maior
- cache de terceiro nível (L3)
 - cache L3 (*fora*) dentro do chip do processador, cache L2 dentro
- misses podem ocorrer em referências a qualquer nível de cache
- transferências entre níveis de cache apresentam mesmos problemas e possíveis soluções já discutidos

Caches de dados e instruções

- dados e instruções: cache unificada x caches separadas
- vantagens das caches separadas
 - política de escrita só precisa ser aplicada à cache de dados
 - caminhos separados entre memória principal e cada cache, permitindo transferências simultâneas (p.ex. num pipeline)
 - estratégias diferentes para cada cache: tamanho total, tamanho de linha, organização
- caches separadas são usadas na maioria dos processadores, no nível L1
- caches unificadas nos níveis L2 e L3

Caches de dados e instruções



Desempenho em caches multinível

- Suponha que o processador tenha um CPI de 1,0 e que todas as referências acertem na cache primária a uma velocidade de clock de 5GHz (0,2ns). O tempo de acesso à memória principal é de 100ns com todos os tratamentos de faltas. Taxa de falhas por instrução na cache primária é de 2%.
- O quanto mais rápido será o processador se acrescentarmos uma cache secundária com tempo de acesso de 5ns para um acerto ou uma falha e que seja grande o suficiente para que a taxa de falhas na L2 seja de 0,5%?

Desempenho em caches multinível

- Penalidade de falha para memória principal:
 - $100\text{ns} = 500$ ciclos de clock.
 - $0,2\text{ns/ciclo}$ de clock
- Para processador com apenas L1:
 - $\text{CPI total} = 1,0 + \text{ciclos de stall de memória por instrução} = 1,0 + 2\% \times 500 = 11,0$
- Em relação a L1, penalidade de falha para L2:
 - $5\text{ns} / 0,2\text{ns} = 25$ ciclos de clock
- Para cache de dois níveis:
 - $\text{CPI total} = 1 + \text{stall L1} + \text{stall L2} = 1 + 2\% \times 25 + 0,5\% \times 500 = 1 + 0,5 + 2,5 = 4,0$
- Portanto, com cache L2:
 - $11,0 / 4,0 = 2,8$ vezes mais rápido