ARQUITETURA DE COMPUTADORES

Memória Cache

1

Tipos de memória

- Acesso randômico (direto)
  - tempo de acesso é o mesmo para todas as posições
  - DRAM: Dynamic Random Access Memory
    - alta densidade, baixa potência, preço reduzido, lenta
    - dinâmica: precisa de um "refresh" regular
  - SRAM: Static Random Access Memory
    - baixa densidade, alta potência, preço elevado, rápida
    - estática: conteúdo dura "para sempre" (enquanto houver alimentação)
- Acesso "não-tão-randômico"
  - tempo de acesso varia de posição para posição e de tempos em tempos
  - exemplos: disco magnético, CD-ROM
- Acesso seqüencial
  - tempo de acesso varia linearmente com a posição (ex. fita)

2

Tendências tecnológicas

	Capacidade	Velocidade (latência)
Lógica:	2x em 3 anos	2x em 3 anos
DRAM:	4x em 3 anos	2x em 10 anos
Disco:	4x em 3 anos	2x em 10 anos

DRAM

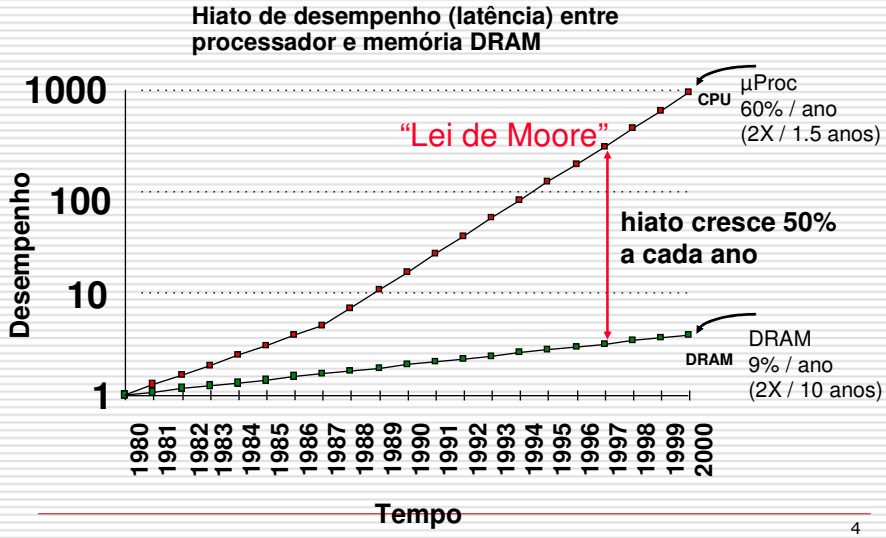
Ano	Tamanho	Tempo acesso
1980	64 Kb	250 ns
1983	256 Kb	220 ns
1986	1 Mb	190 ns
1989	4 Mb	165 ns
1992	16 Mb	145 ns
1995	64 Mb	120 ns

1000:1!

2:1!

3

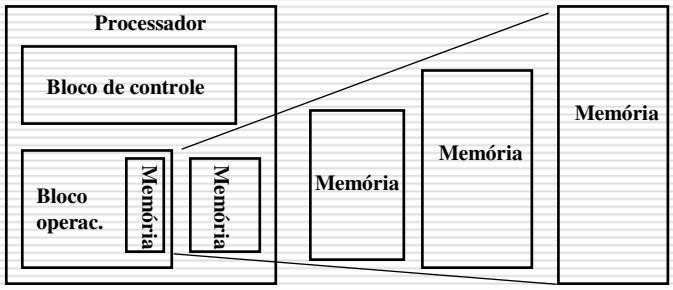
Tendências tecnológicas



4

Hierarquia de memória

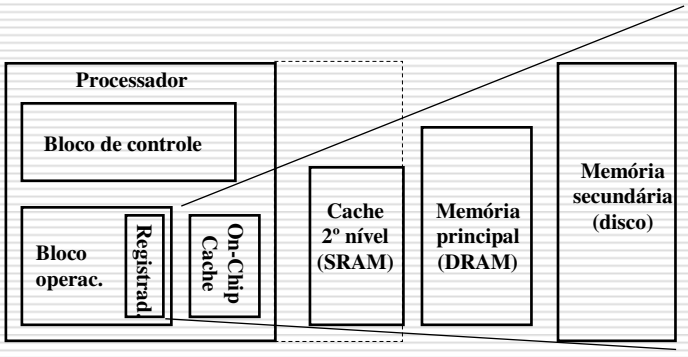
- Objetivo: oferecer ilusão de máximo tamanho de memória, com mínimo custo e máxima velocidade
- Cada nível contém cópia de parte da informação armazenada no nível superior seguinte



Velocidade:	Mais rápida	Mais lenta
Tamanho:	Menor	Maior
Custo:	Mais alto	Mais baixo

5

Hierarquia de memória



Velocidade (s):	0,1n	1n	2-5n	10-20n	10 m
Tamanho (bytes):	100	16 K	512 K	256 M	G

6

Hierarquia de memória

Como a hierarquia é gerenciada?

- Registradores <-> memória
  - pelo compilador
- cache <-> memória principal
  - pelo hardware
- memória principal <-> disco
  - pelo hardware e pelo sistema operacional (memória virtual)
  - pelo programador (arquivos)

7

Princípio de localidade

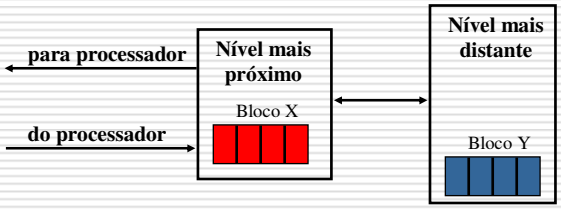
- Hierarquia de memória funciona devido ao princípio de localidade
  - todos os programas repetem trechos de código e acessam repetidamente dados próximos
- localidade temporal: posições de memória, uma vez acessadas, tendem a ser acessadas novamente no futuro próximo
- localidade espacial: endereços em próximos acessos tendem a ser próximos de endereços de acessos anteriores

8

Princípio de localidade

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



Hit (acerto) e *Miss* (falha)

- Hit: dado aparece em algum bloco no nível superior (junto ao processador)
  - Hit Ratio: a fração de acessos à memória resolvidos no nível superior
  - Hit Time: tempo de acesso ao nível superior, que consiste em:
    tempo para determinar hit/miss + tempo de acesso à memória (cache)
- Miss: dado precisa ser buscado de um bloco no nível inferior
  - Miss Ratio = 1 – (Hit Ratio)
  - Miss Penalty: tempo para determinar hit/miss + tempo gasto para substituir um bloco no nível superior + tempo para fornecer o bloco ao processador
- Hit Time << Miss Penalty

Localidade temporal

- Usualmente encontrada em laços de instruções e acessos a pilhas de dados e variáveis
- É essencial para a eficiência da memória cache
- Se uma referência é repetida N vezes durante um laço de programa, após a primeira referência a posição é sempre encontrada na cache

Tc = tempo de acesso à cache
Tm = tempo de acesso à memória principal
Tce = tempo efetivo de acesso à cache
Determine o Tce em função de N, Tc e Tm

$$Tce = \frac{N\,Tc + Tm}{N} = Tc + \frac{Tm}{N}$$

Para um exemplo onde N=10 e Tm = 20*Tc, qual seria o Tce?

se Tc = 1 ns, Tm = 20 ns, N = 10 ⇨ Tce = 3 ns
N = 100 ⇨ Tce = 1,2 ns

11

Cache

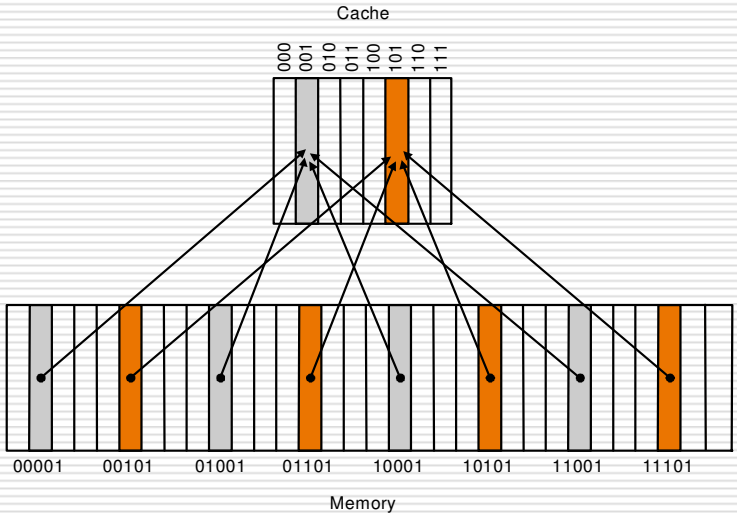
- Como sabemos se um dado está na cache?
- Caso ele esteja, como encontrá-lo?
- A maneira mais simples...

Mapeamento direto:

- Cada local da memória (nível inferior) é mapeado exatamente para um local na cache

12

Isso se chama mapeamento direto



13

Cache: um exemplo simples

armazenado como parte do "estado" da cache

Cache

Índice	Valid	Tag	Dado
00			
01			
10			
11			

Memória principal

0000xx
0001xx
0010xx
0011xx
0100xx
0101xx
0110xx
0111xx
1000xx
1001xx
1010xx
1011xx
1100xx
1101xx
1110xx
1111xx

Q1: Está lá?

Compara a cache tag com os 2 bits mais significativos do endereço de memória prá saber se o bloco de memória está na cache

Q2: Como encontrá-lo?

Use os próximos 2 bits menos significativos do endereço – o índice – para determinar qual bloco de cache (i.e., modulo número de blocos da cache)

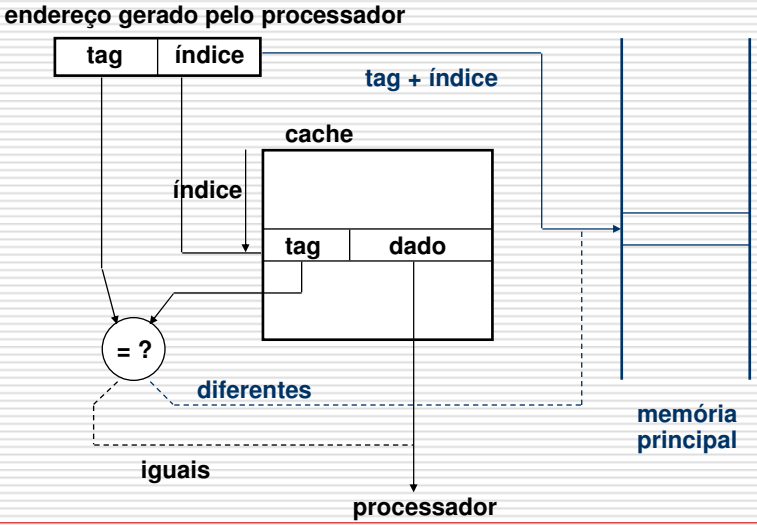
14

Mapeamento direto -resumo

- Endereço é dividido em 2 partes
  - parte menos significativa: índice, usado como endereço na cache onde será armazenada a palavra
  - parte mais significativa: tag, armazenado na cache junto com o conteúdo da posição de memória
- Quando acesso é feito, índice é usado para encontrar palavra na cache
  - se tag armazenado na palavra da cache é igual ao tag do endereço procurado, então houve hit
- Endereços com mesmo índice são mapeados sempre para a mesma palavra da cache (substituição simplificada)

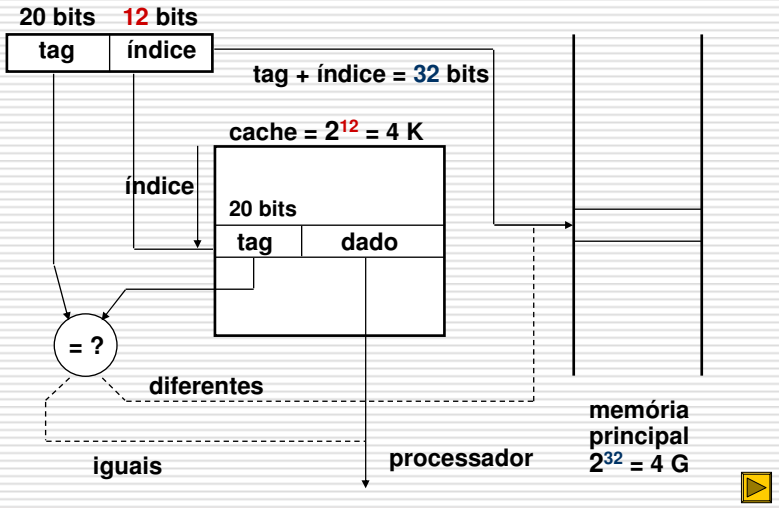
15

Exemplo de implementação de mapeamento direto



16

Mapeamento direto – exemplo numérico



17

Cache com mapeamento direto

- Considere os seguinte acesso à memória:

Iniciando com a cache vazia – todos os blocos são marcados inválidos

0 1 2 3 4 3 4 15

Tag

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

	4 miss	3 hit	4 hit	15 miss
	01 ~~00~~ Mem(~~0~~) 4	01 Mem(4)	01 Mem(4)	01 Mem(4)
	00 Mem(1)	00 Mem(1)	00 Mem(1)	00 Mem(1)
	00 Mem(2)	00 Mem(2)	00 Mem(2)	00 Mem(2)
	00 Mem(3)	00 Mem(3)	00 Mem(3)	11 ~~00~~ Mem(~~3~~) 15

- 8 requisições, 6 misses (falhas)

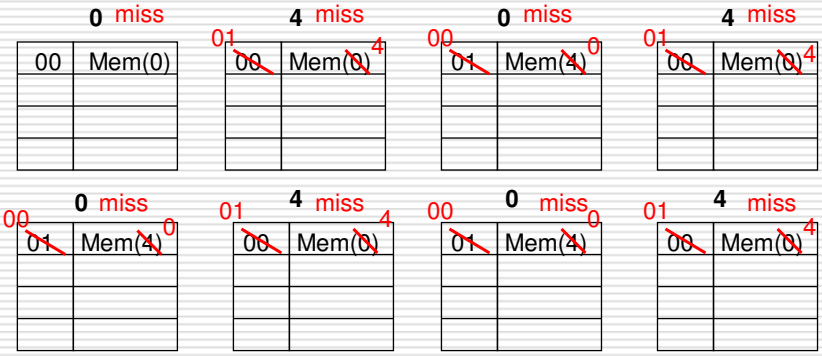
18

Cache com mapeamento direto: outro exemplo

- Considere os seguinte acesso à memória:

Iniciando com a cache vazia – todos os blocos são marcados inválidos

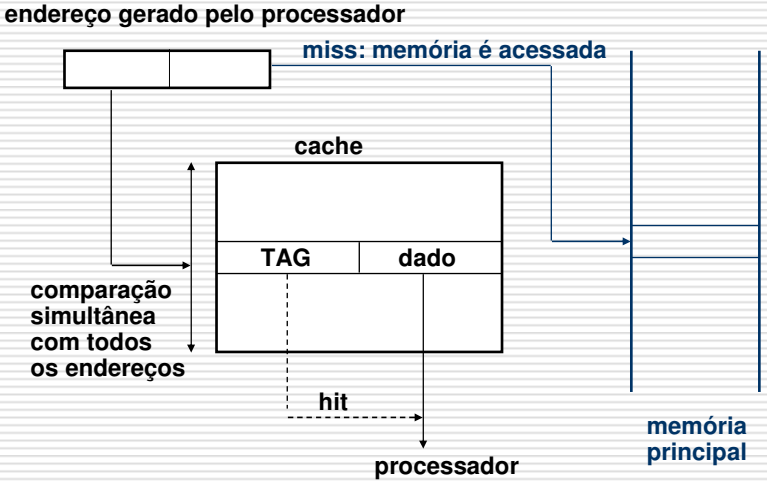
0 4 0 4 0 4 0 4



Efeito ping-pong devido à falha de conflito – duas localidades de memória que são mapeadas para o mesmo bloco de cache

19

Mapeamento completamente associativo

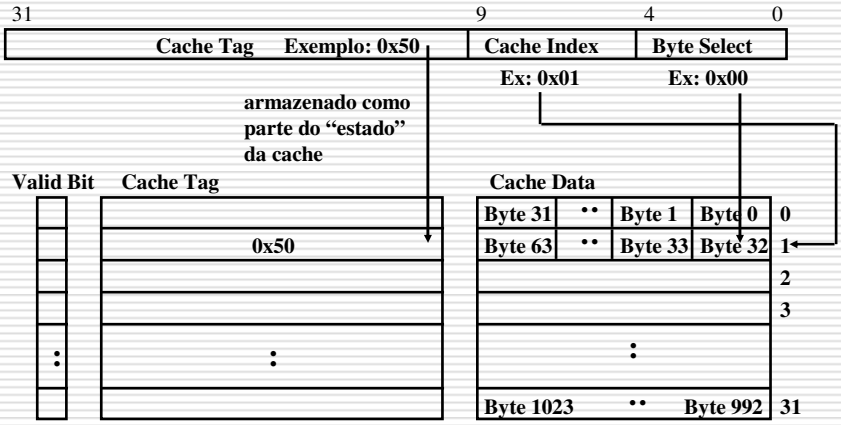


20

Mapeamento completamente associativo - resumo

- Vantagem: máxima flexibilidade no posicionamento de qualquer palavra (ou linha) da memória principal em qualquer palavra (ou linha) da cache
- Desvantagens
  - custo em hardware da comparação simultânea de todos os endereços armazenados na cache
  - algoritmo de substituição (em hardware) para selecionar uma linha da cache como conseqüência de um miss
- Utilizado apenas em memórias associativas de pequeno tamanho
  - tabelas

Mapeamento direto – uso de linhas



Tirando vantagem da localidade espacial

- Considere os seguinte acesso à memória:

Iniciando com a cache vazia – todos os blocos são marcados inválidos

0 1 2 3 4 3 4 15

0 miss

00	Mem(1)	Mem(0)

1 hit

00	Mem(1)	Mem(0)

2 miss

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

3 hit

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

4 miss

~~00~~ 01	~~Mem(1)~~ Mem(5)	~~Mem(0)~~ Mem(4)
00	Mem(3)	Mem(2)

3 hit

01	Mem(5)	Mem(4)
00	Mem(3)	Mem(2)

4 hit

01	Mem(5)	Mem(4)
00	Mem(3)	Mem(2)

15 miss

~~01~~ 11	Mem(5)	Mem(4)
~~00~~ 15	Mem(3)	~~Mem(2)~~ 14

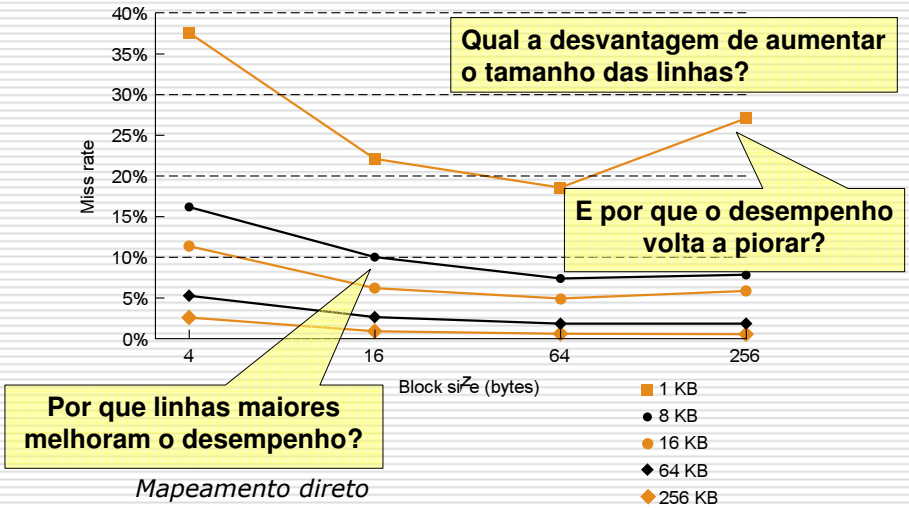
- 8 requisições, 4 misses (falhas)

Quantos bits tem a cache no total?

- Supondo cache com mapeamento direto, com 64 KB de dados, linha com palavras de 32 bits, 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} x (32 + 32 -14 -2 +1) = 2^{14} x 49 = 784 x 2^{10} = 784 Kbits
- 98 KB para 64 KB de dados, ou 50% a mais

Tamanho da linha x *miss ratio*



Mapeamento direto

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

Mapeamento direto - propriedades

- Vantagens
  - não há necessidade de algoritmo de substituição
  - hardware simples e de baixo custo
  - alta velocidade de operação
- Desvantagens
  - desempenho cai se acessos consecutivos são feitos a palavras com mesmo índice (ping-pong)
  - *hit ratio* inferior ao de caches com mapeamento associativo
- Demonstra-se no entanto que *hit ratio* aumenta com o aumento da cache, aproximando-se de caches com mapeamento associativo
  - tendência atual é de uso de caches grandes

27

Origens de falhas (misses) na cache

- 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 (conjunto associativo)
- De capacidade
  - cache não pode conter todas as linhas accessadas pelo programa
  - solução: aumentar tamanho da cache
- Invalidação: outro processo (p.ex. I/O) atualiza memória

28

Literatura

- *Stallings – cap 4.3 e 4.4*
- *Patterson – cap 7*

29

Resumo (1)

- 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 conjunto associativo

30

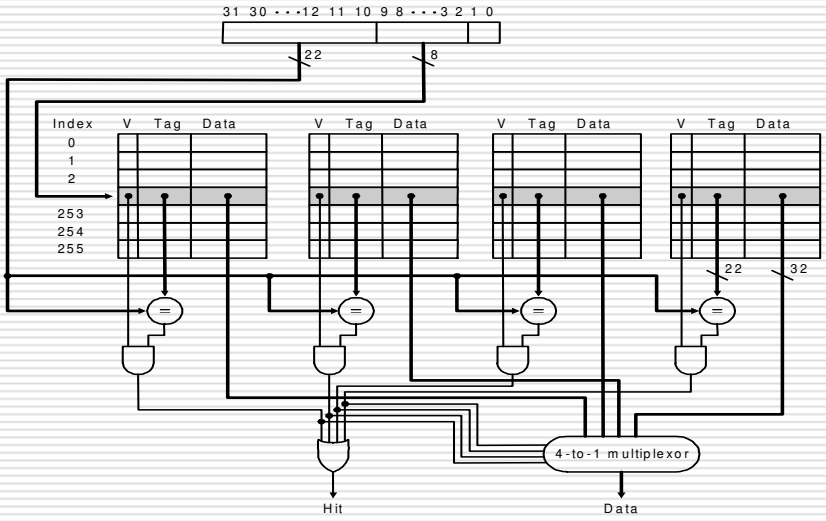
Resumo (2)

- Mapeamento direto
  - Cada local da memória é mapeado exatamente para um local na cache.
- Mapeamento completamente associativo
  - Um bloco pode ser posicionado em qualquer local da cache.
- Mapeamento set-associativo (associativo por conjunto)
  - A cache possui um número finito de locais (no mínimo 2) onde cada bloco pode ser colocado.

Questões adicionais...

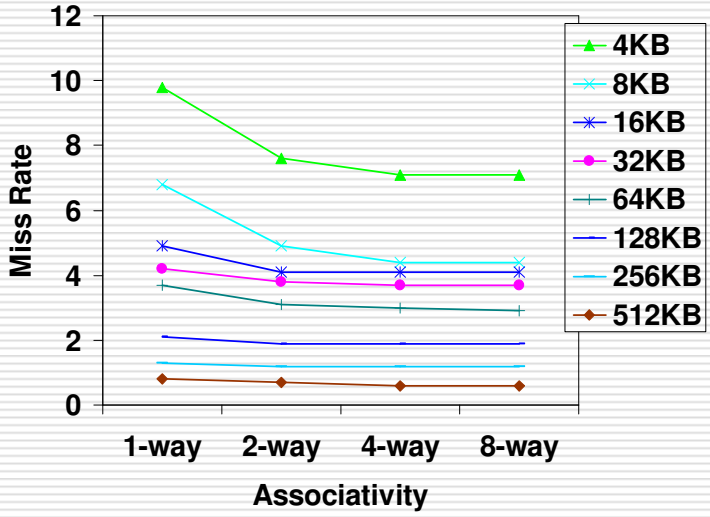
- Investigue o funcionamento do mapeamento de cache associativo por conjunto.
- Pesquise o tipo de mapeamento de processadores comerciais (Pentium, PowerPC).

Mapeamento associativo por conjunto



33

Conjunto associativo - desempenho



Hennessy & Patterson, *Computer Architecture*, 2003

34

Processadores comerciais

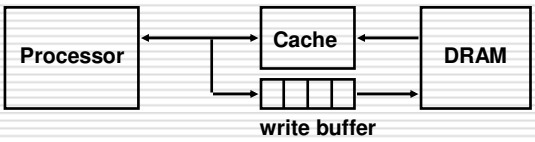
Characteristic	Intel Pentium P4	AMD Opteron
L1 cache organization	Split instruction and data caches	Split instruction and data caches
L1 cache size	8 KB for data, 96 KB trace cache for RISC instructions (12K RISC operations)	64 KB each for instructions/data
L1 cache associativity	4-way set associative	2-way set associative
L1 replacement	Approximated LRU replacement	LRU replacement
L1 block size	64 bytes	64 bytes
L1 write policy	Write-through	Write-back
L2 cache organization	Unified (instruction and data)	Unified (instruction and data)
L2 cache size	512 KB	1024 KB (1 MB)
L2 cache associativity	8-way set associative	16-way set associative
L2 replacement	Approximated LRU replacement	Approximated LRU replacement
L2 block size	128 bytes	64 bytes
L2 write policy	Write-back	Write-back

FIGURE 7.35 First-level and second-level caches in the Intel Pentium P4 and AMD Opteron. The primary caches in the P4 are physically indexed and tagged; for a discussion of the alternatives, see the Elaboration on page 527.

35

Escrevendo na cache [1]

- *Write-through*
  - A informação é escrita tanto no bloco da cache quanto no próximo nível da hierarquia de memória
  - Esta técnica usualmente é combinada com um buffer de escrita de modo a eliminar o tempo de espera pela escrita na memória mais lenta (desde que o buffer não encha)



36

Escrevendo na cache [2]

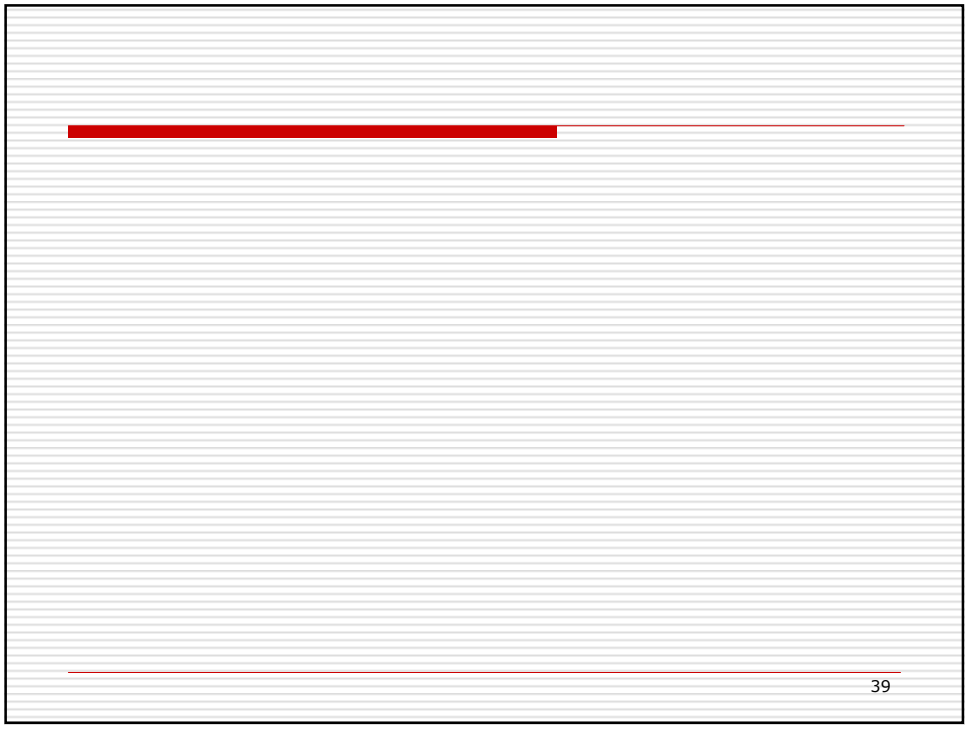
- *Write-back*
  - A informação é escrita apenas no bloco de cache. O bloco modificado é escrito no próximo nível da hierarquia de memória somente quando ocorre uma substituição
  - Precisa de um *dirty bit* para informar se o bloco foi alterado

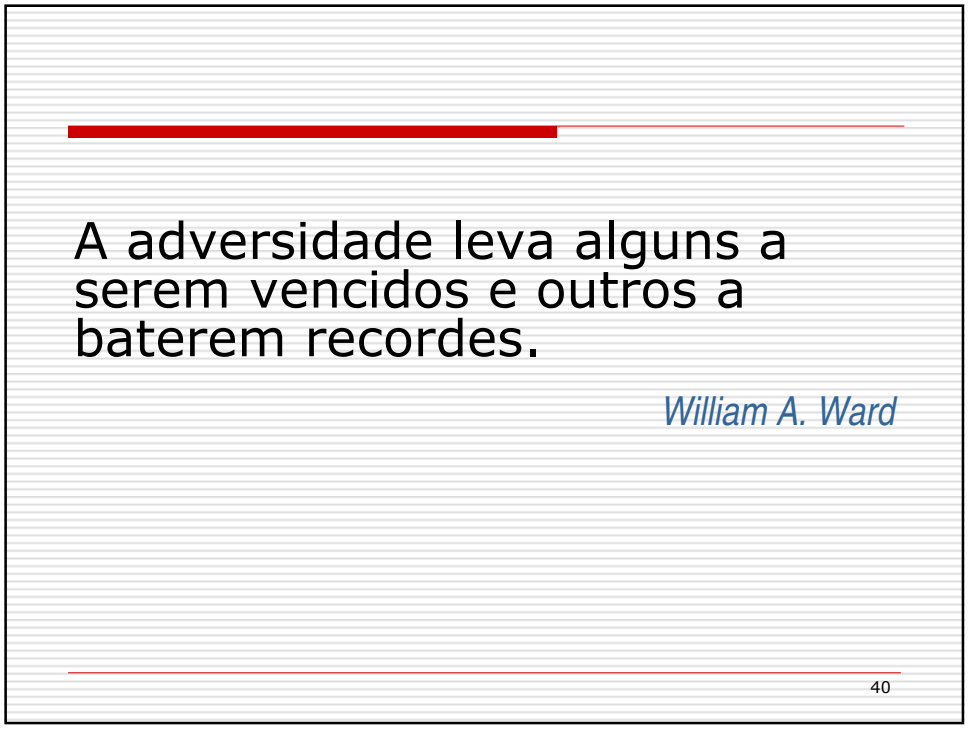
37

Escrevendo na cache [3]

- Comparando ...
  - *Write-through*: mais barato e mais simples (sem *dirty bit*)
  - *Write-back*: escritas sucessivas no mesmo endereço requerem apenas uma escrita na memória mais lenta

38


39


A adversidade leva alguns a serem vencidos e outros a baterem recordes.
William A. Ward
40

There is only one boss: The Customer. And he can fire everybody in the company, from the chairman on down, simply by spending his money somewhere else.

Sam Walton

41