MC542

Organização de Computadores
Teoria e Prática

2006

Prof. Paulo Cesar Centoducatte

ducatte@ic.unicamp.br

www.ic.unicamp.br/~ducatte

MC542

Arquitetura de Computadores

Sistemas de Memória

"Computer Organization and Design:
The Hardware/Software Interface" (Capítulo 7)

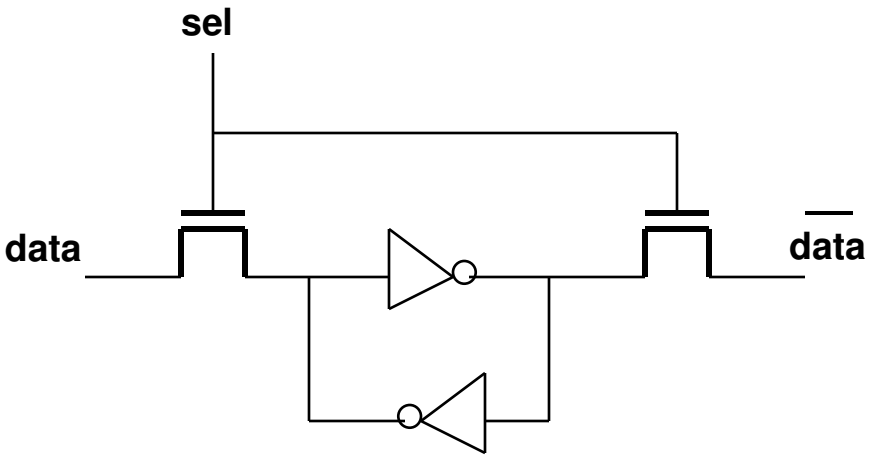
Sumário

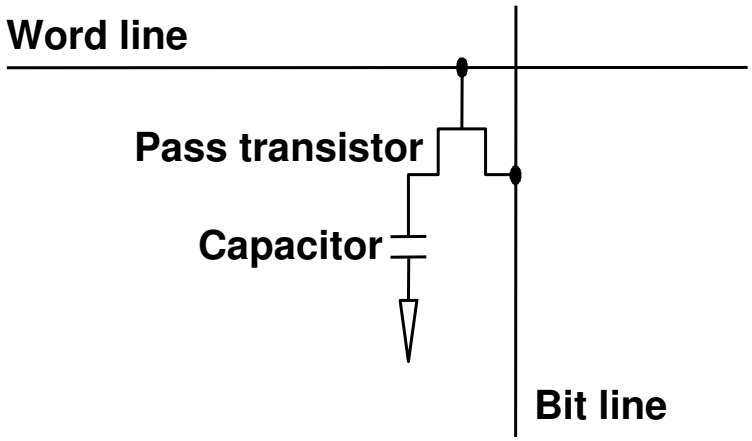
- Revisão
- Explorando Hierarquia de Memória
  - Hierarquia de Memória
  - Custo e Velocidade
  - Principio da Localidade
    - Localidade Temporal
    - Localidade Espacial
  - Definições
- Visão em Dois Níveis
- Cache
  - Direct Mapped Cache
- Via de Dados com Pipeline
- Tamanho da Cache em bits
- Tratamento de Cache Misses

Memórias: Revisão

- **SRAM:**
  - valor armazenado em um par de portas inversoras
  - mais rápida porém usa mais espaço do que DRAM (4 a 6 transistores)

- **DRAM:**
  - valor armazenado como carga de um capacitor (deve ser feito refresh)
  - menor porém mais lenta do que SRAM (fator de 5 a 10)



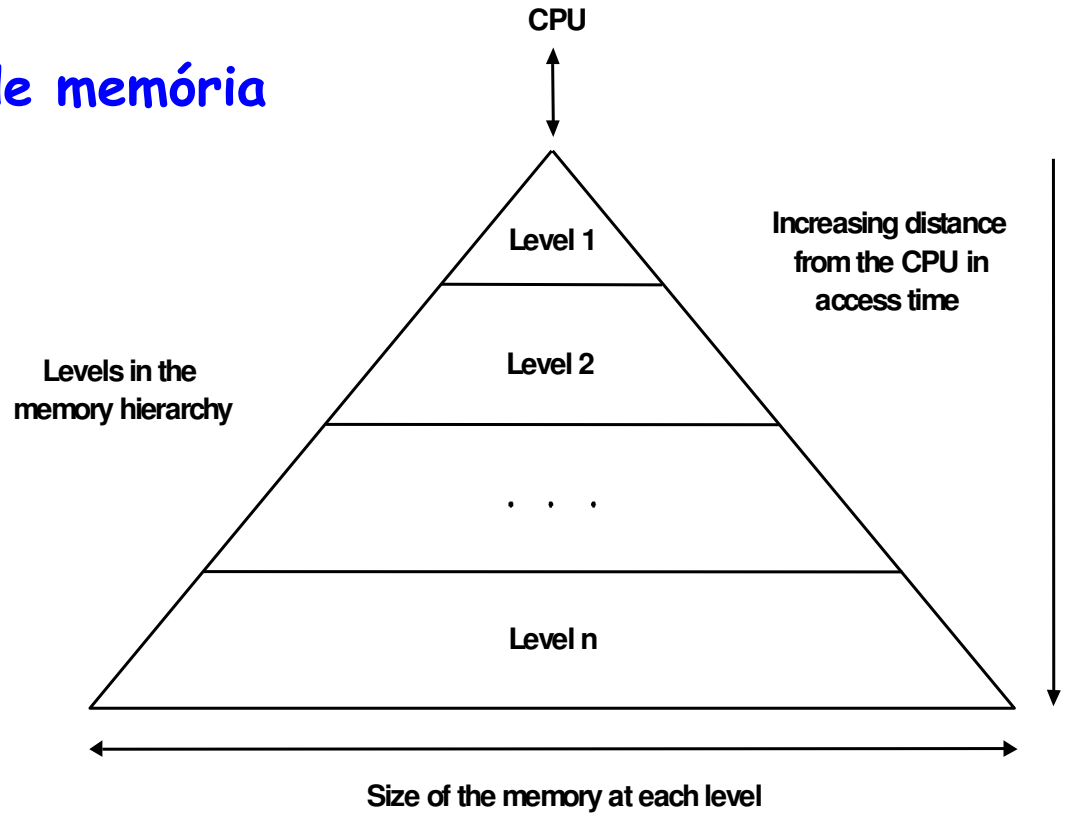


Explorando Hierarquia de Memória

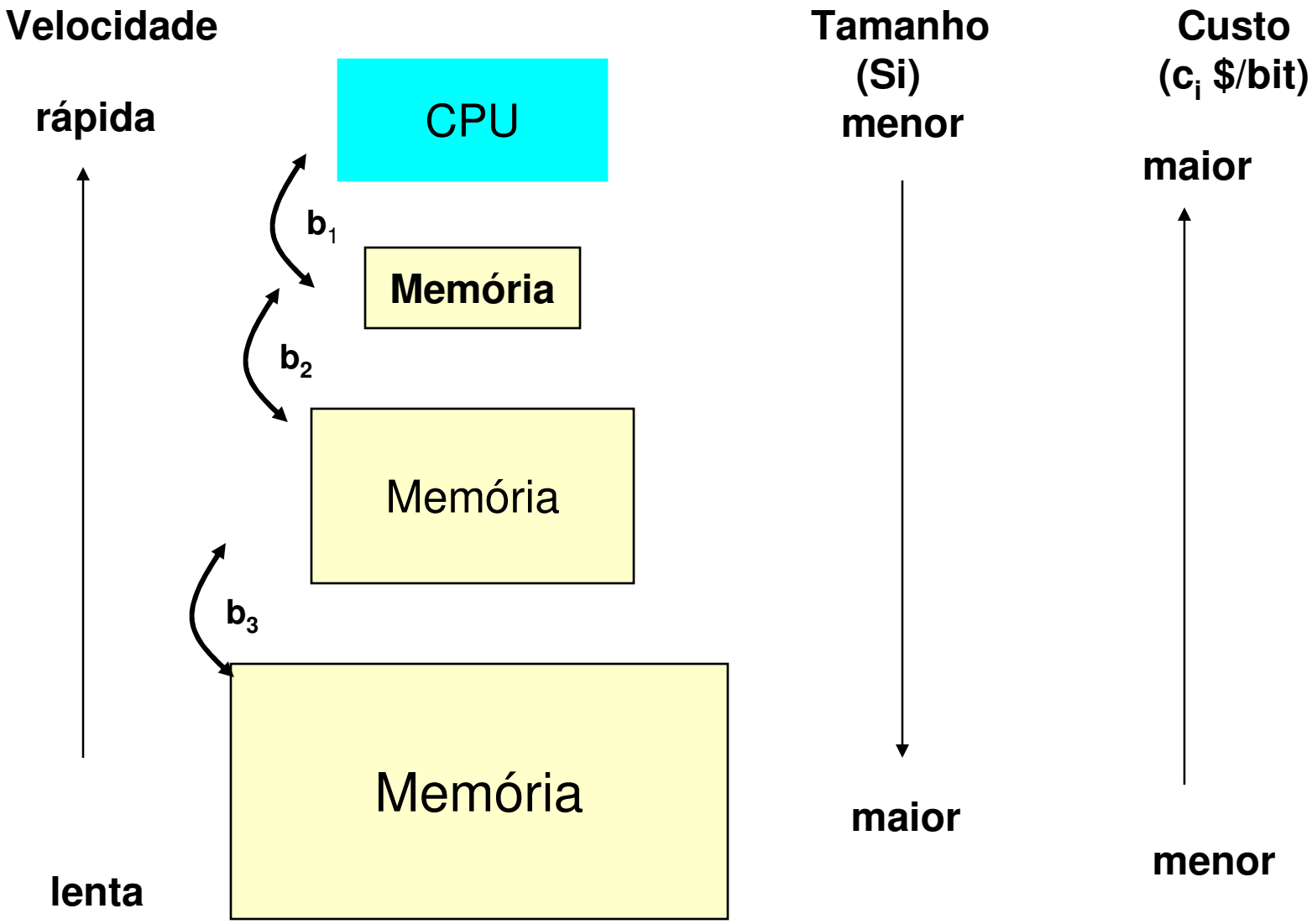
- Usuários querem memórias grandes e rápidas
  - SRAM access times: 2 - 25ns; custo de $100 a $250 por Mbyte.
  - DRAM access times: 60-120ns; custo de $5 a $10 por Mbyte
  - Disk access times: 10 a 20 milhões de ns; custo de $.10 a $.20 por Mbyte.

1997

- Solução: hierarquia de memória



Hierarquia de Memória



Hierarquia de Memória (Custo e Velocidade)

- Custo médio do sistema ($/bit)

$$\frac{S_1 C_1 + S_2 2_3 + \ldots\ldots + S_n C_n}{S_1 + S_2 + \ldots\ldots + S_n}$$

- Objetivos do sistema
  - Custo médio ≈ custo do nível mais barato (disco)
  - Velocidade do sistema ≈ velocidade do mais rápido (cache)

- Assumindo disco 40 GB e memória de 256 MB
  - Calcular o custo médio por bit

Localidade

- Principio que torna possível o uso de hierarquia de memória

- Um programa acessa uma porção relativamente pequena do espaço endereçável em um instante de tempo qualquer.

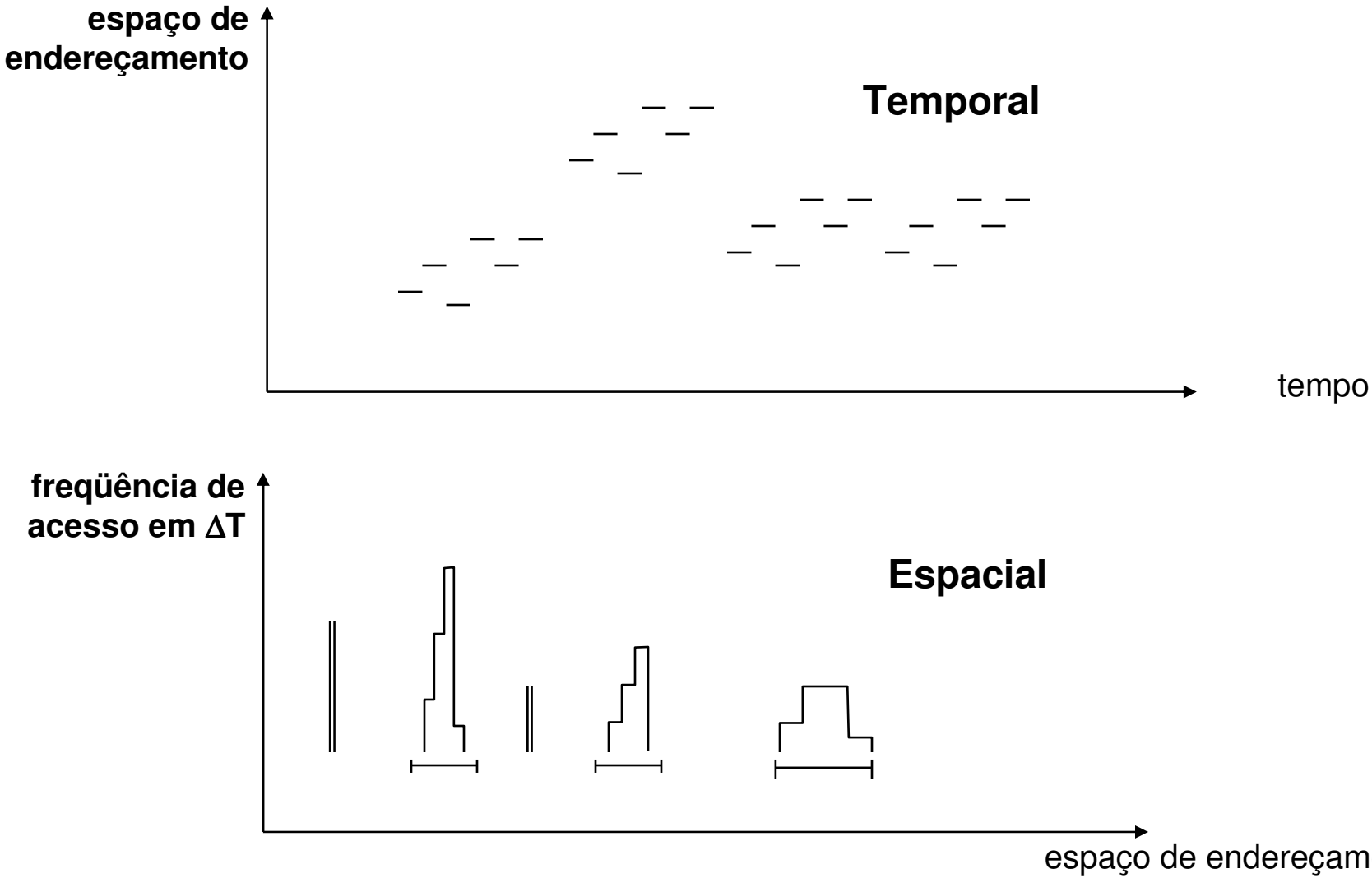
  - Localidade temporal: Se um item é referenciado, ele tende a ser referenciado novamente.
    - Exemplo → loops (instruções e dados).

  - Localidade Espacial: Se um item é referenciado, itens cujos endereços são próximos a este, tendem a ser referenciados também.
    - Exemplo → acesso a dados de um array.

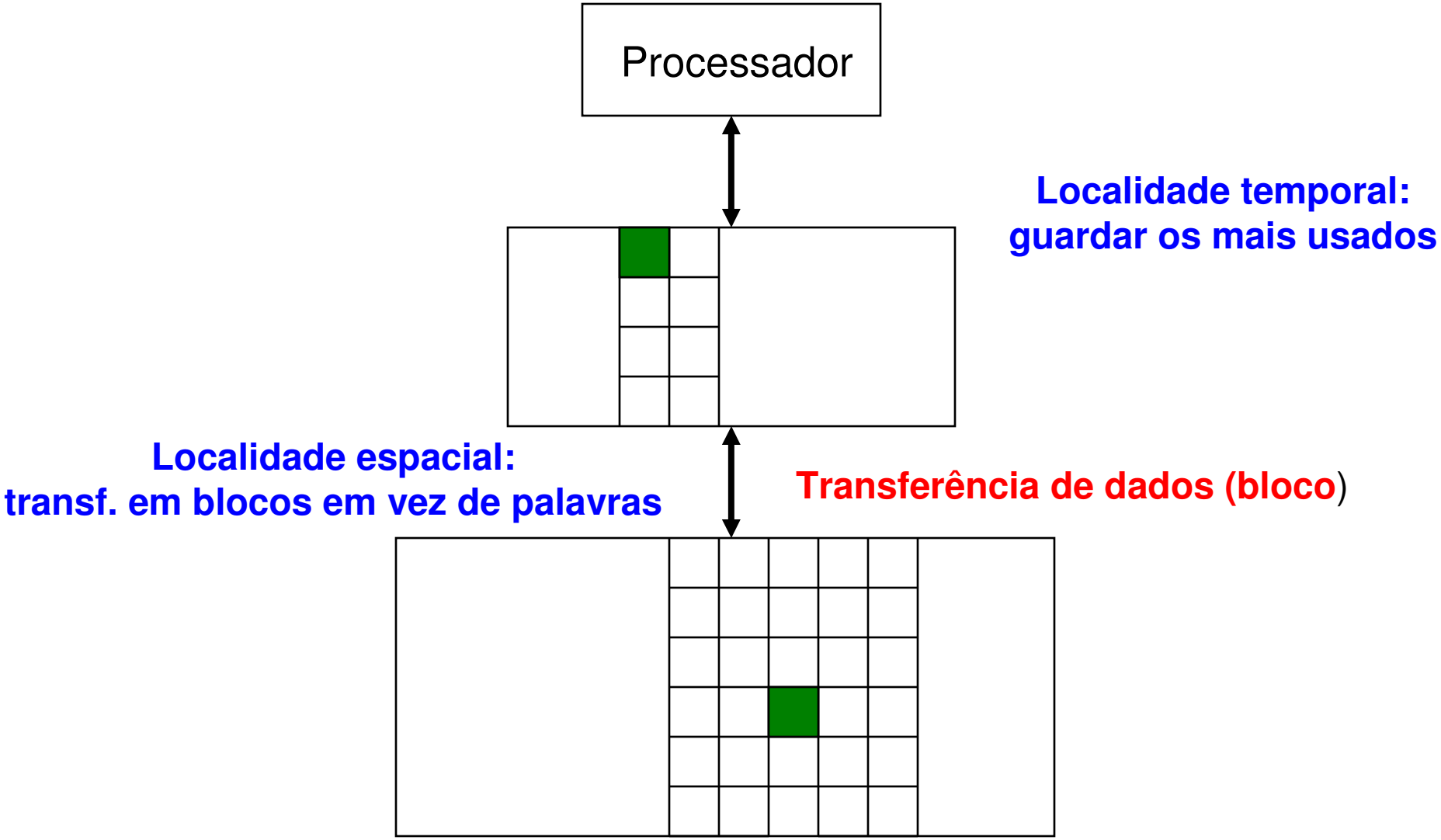
Localidade - Definições

- Bloco ➔ mínima unidade de informação que pode ou não estar presente em dois níveis de hierarquia de memória.
- Hit ➔ se o dado acessado aparece em algum bloco no nível superior.
- Miss ➔ se o dado acessado não aparece em algum bloco do nível superior.
- Hit ratio (hit rate) ➔ razão hits pelo número total de acessos ao nível superior.
- Miss ratio (miss rate) ➔ razão de misses pelo número total de acessos ao nível superior ➔ miss ratio = 1 – hit ratio.
- Hit time ➔ tempo de acesso ao nível superior da hierarquia de memória, que inclui o tempo necessário para saber se no acesso ocorrerá um hit ou um miss.
- Miss penalty ➔ tempo para recolocar um bloco no nível superior e enviá-lo ao processador, quando ocorre um miss. O maior componente do miss penalty é o tempo de acesso ao nível imediatamente inferior da hierarquia de memória.

Princípio da localidade



Visão em Dois Níveis



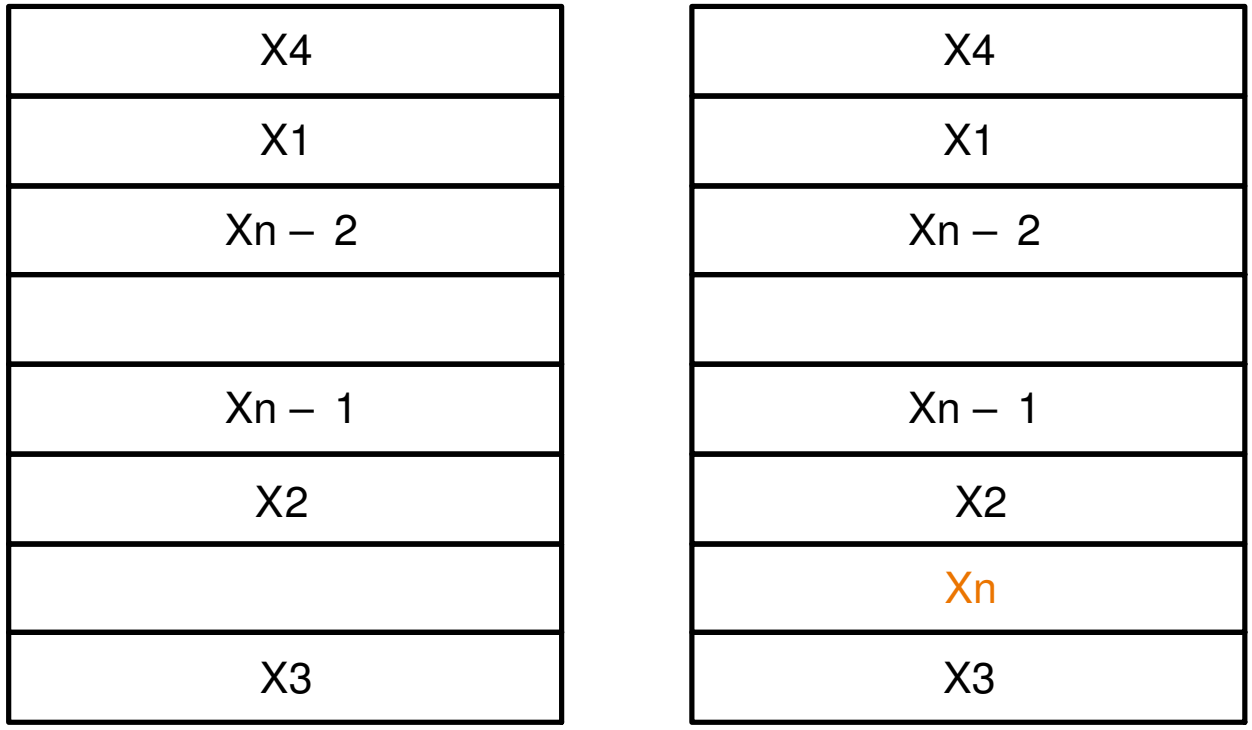
- **Memória Cache → nível da hierarquia entre CPU e Memória Principal ou qualquer espaço de armazenamento usado para tirar vantagem da localidade de acesso.**

- **Supondo uma cache simples onde um bloco corresponde a uma palavra e o processador acessa a uma palavra.**

- **Supor um bloco Xn que inicialmente não esteja na cache:**

Cache

Referência à posição Xn



a. Before the reference to Xn

b. After the reference to Xn

Cache

- Duas perguntas no acesso à cache:
  - Como saber se o dado está na cache?
  - Se estiver, como encontra-lo?

- Se cada palavra tiver um lugar, fixo, na cache ➔ saberemos como encontra-la.

- A maneira mais simples de assinalar uma posição da cache para uma palavra de memória é através de seu endereço na memória ➔ direct mapped
  - (Endereço do bloco) mod (Número de blocos na cache)

Cache

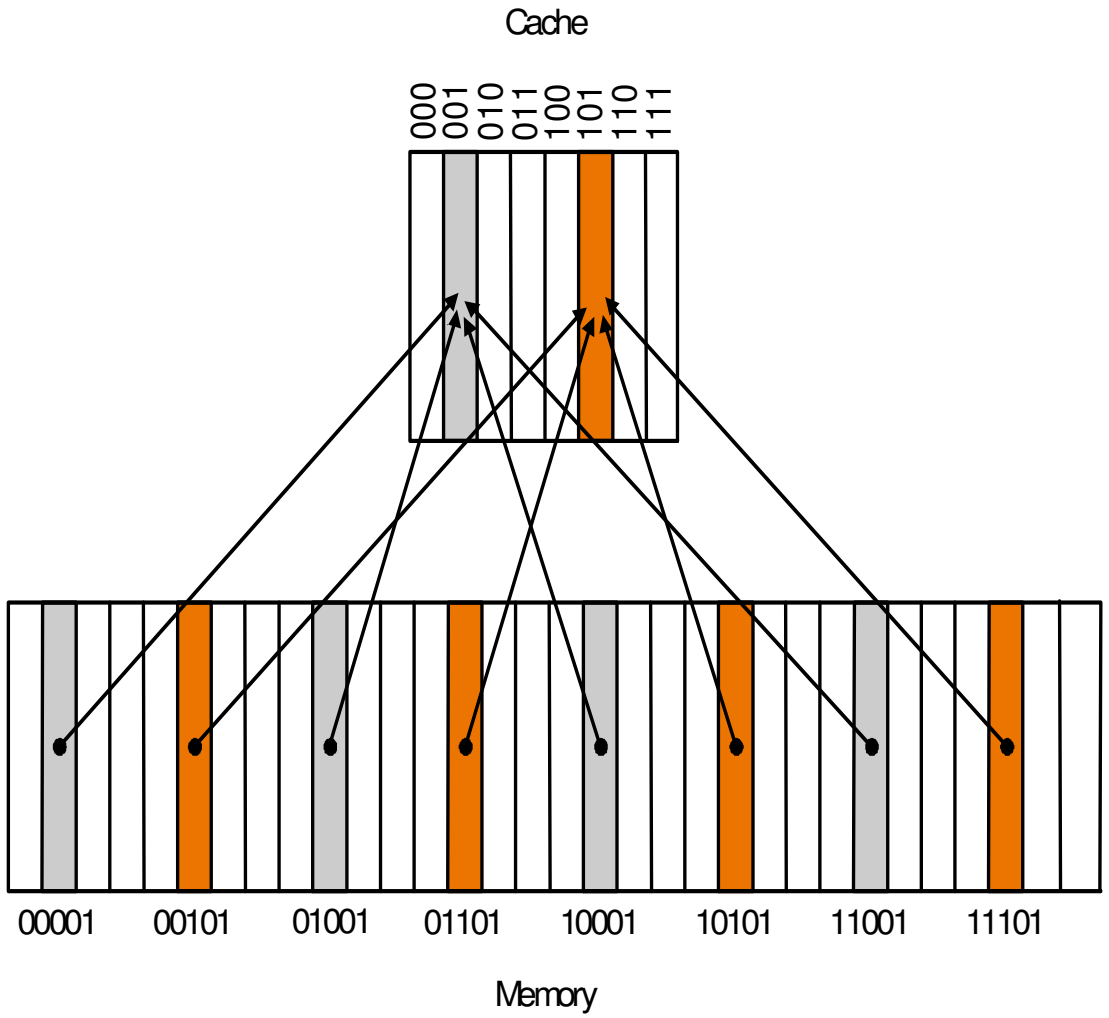
- Políticas:
  - mapeamento de endereços entre cache e memória
  - escrita: como fazer a consistência de dados entre cache e memória
  - substituição: qual bloco descartar da cache

Direct Mapped Cache

- Mapping: address is modulo the number of blocks in the cache

cache:
8 posições
3 bits de endereço

memória:
32 posições
5 bits de endereço



Preenchimento da Cache a Cada Miss

Index	V	Tag	Data
000	N		
001	N		
010	N		
011	N		
100	N		
101	N		
110	N		
111	N		

Index	V	Tag	Data
000	N		
001	N		
010	N		
011	N		
100	N		
101	N		
110	Y	10	M(10110)
111	N		

Index	V	Tag	Data
000	N		
001	N		
010	Y	11	M(11010)
011	N		
100	N		
101	N		
110	Y	10	M(10110)
111	N		

Index	V	Tag	Data
000	Y	10	M(10000)
001	N		
010	Y	11	M(11010)
011	N		
100	N		
101	N		
110	Y	10	M(10110)
111	N		

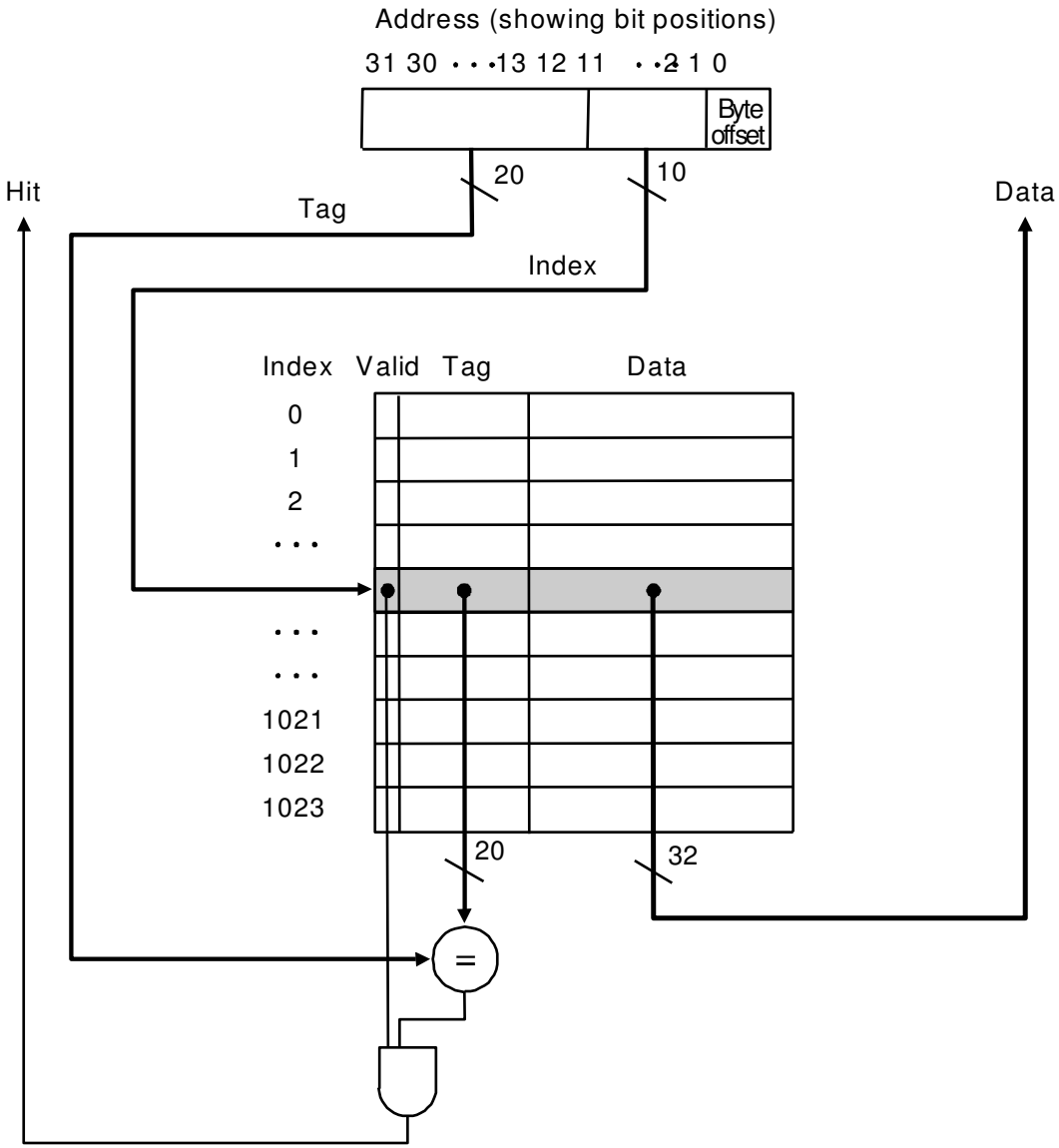
Index	V	Tag	Data
000	Y	10	M(10000)
001	N		
010	Y	11	M(11010)
011	Y	00	M(00011)
100	N		
101	N		
110	Y	10	M(10110)
111	N		

end_{10}	end_2	end_{cache}	Hit
22	10 110	110	
26	11 010	010	
22	10 110	110	H
26	11 010	010	H
16	10 000	000	
3	00 011	011	
16	10 000	000	H
18	10 010	010	

Direct Mapped Cache

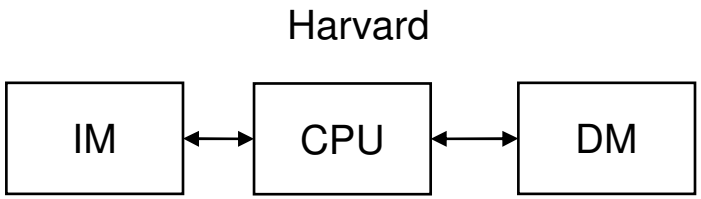
- mapeamento direto
- byte offset:
  - só para acesso a byte
- largura da cache: v+tag+dado

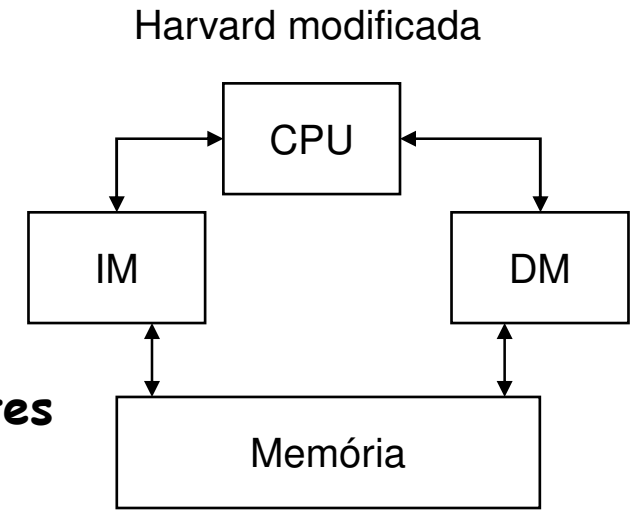
- cache de 2^n linhas:
- índice de n bits
- linha da cache: 1+(30-n)+32
  v tag dado
- tamanho da cache= 2^n*(63-n)



Via de Dados com Pipeline

- Data memory = cache de dados
- Instruction memory = cache de instruções
- Arquitetura
  - de Harvard
  - ou Harvard modificada





- Miss? semelhante ao stall
  - dados: congela o pipeline
  - instrução:
    - quem já entrou prossegue
    - inserir "bolhas" nos estágios seguintes
    - esperar pelo hit
- enquanto instrução não é lida, manter endereço original (PC-4)

Tamanho da Cache em bits

- Número de bits necessários para uma cache é função do tamanho da cache e do tamanho do endereço (dados + tags)

- Número de bits de uma cache
  - Endereço de 32 bits, cache com mapeamento direto de 2^n words com blocos de uma palavra (4 bytes) ➔ tag de 32 – (n + 2).

  - 2 bits usados para offset do byte e n para o índice. O número total de bits da cache ➔ 2^n X (32 + (32 – n –2) + 1) = 2^n X (63 – n).

- Exemplo: Quantos bits são necessários para uma cache com mapeamento direto com 64KB de capacidade para dados e bloco de uma palavra, assumindo endereço de 32-bit?

Solução:

64KB ➔ 16K palavras ➔ 2^{14} palavras ➔ 2^{14} blocos

Cada bloco tem 32 bits de dados mais o tag (32 – 14 – 2 = 16) mais o bit de validade

Total de bits da cache 2^{14} X (32 + 16+ 1) = 784 Kbits = 98 KB

Para esta cache, temos um overhead de 1.5, devido aos tag e aos bits de validade.

Tratamento de Cache Misses

- Quando a unidade de controle detecta um **miss**, ela busca o dado da memória. Se detecta um **hit**, continua o processamento como se nada tivesse acontecido.

- Mudança do datapath (cap. 5 e cap. 6) ➔ substituir as memórias por caches e alterar o controle para quando ocorrer **miss**.

- Alteração do controle ➔ atrasar (stall semelhante ao stall do pipeline ➔ diferença que para todas as unidades do pipeline) da CPU, congelando o conteúdo de todos os registradores. Um controlador separado trata o miss, lendo o dado da memória.

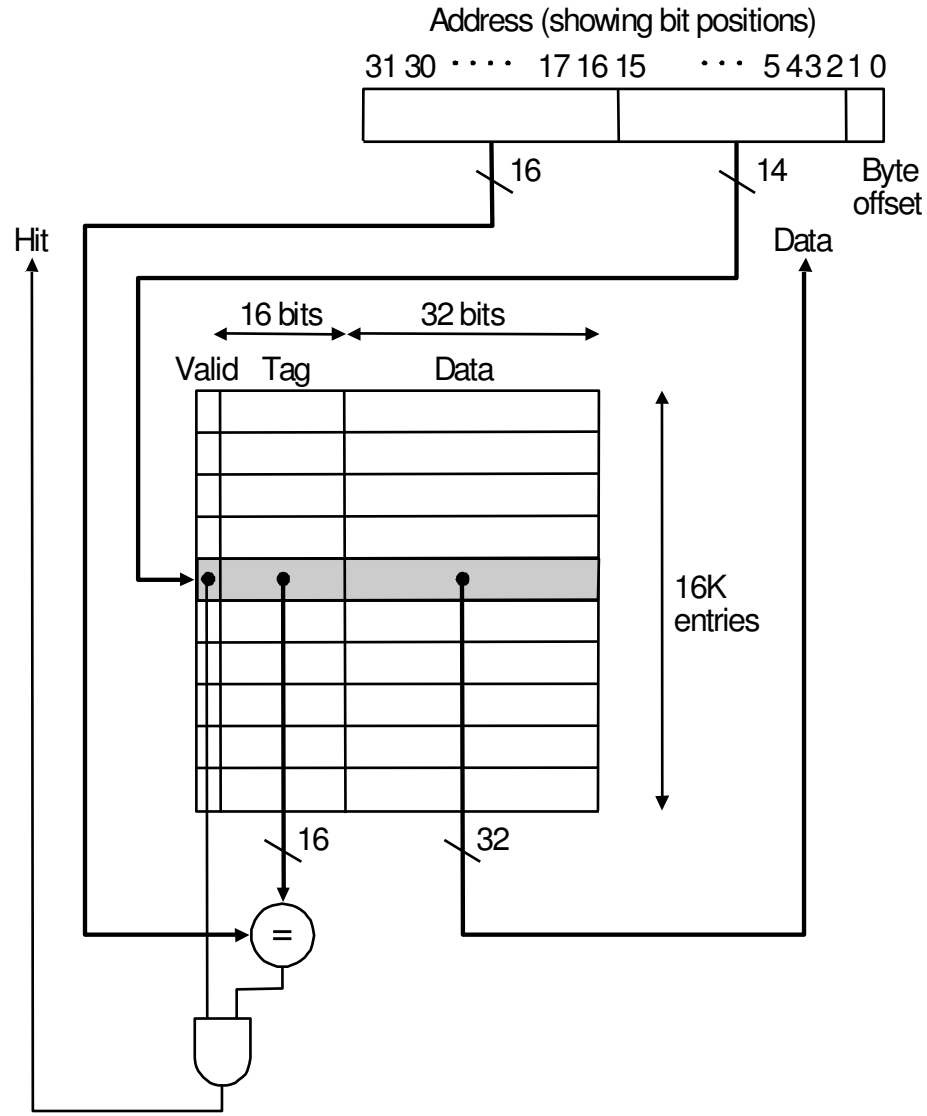
Etapas de um Cache Miss de uma Instrução

- Enviar o PC original (PC corrente – 4) para a memória
- Fazer a leitura da memória e esperar o conteúdo
- Escrever na cache o dado vindo da memória, escrevendo os bits mais significativos do endereço (da ULA) no campo de tag e *setando* o bit de validade.
- Reiniciar a execução da instrução.

Etapas de um Cache Miss de Dados

- stall no processador até a memória enviar o dado.

A Cache da DECStation 3100



Etapas para uma Leitura na Cache (de Dados ou de Instruções)

- Enviar o endereço para a cache (vem do PC para leitura de instruções ou da ULA para leitura de dados)

- Se existir o sinal hit, significa que a palavra desejada está disponível na linha de dados. Se existir o sinal de miss o endereço é enviado à memória principal, e quando o dado chega, é escrito na cache.

Etapas para uma Escrita na Cache

- Escrita → na escrita de uma instrução de store → o dado tem que ser escrito na cache → valores diferentes entre cache e memória principal → inconsistência → escrever também na memória principal → write-through.

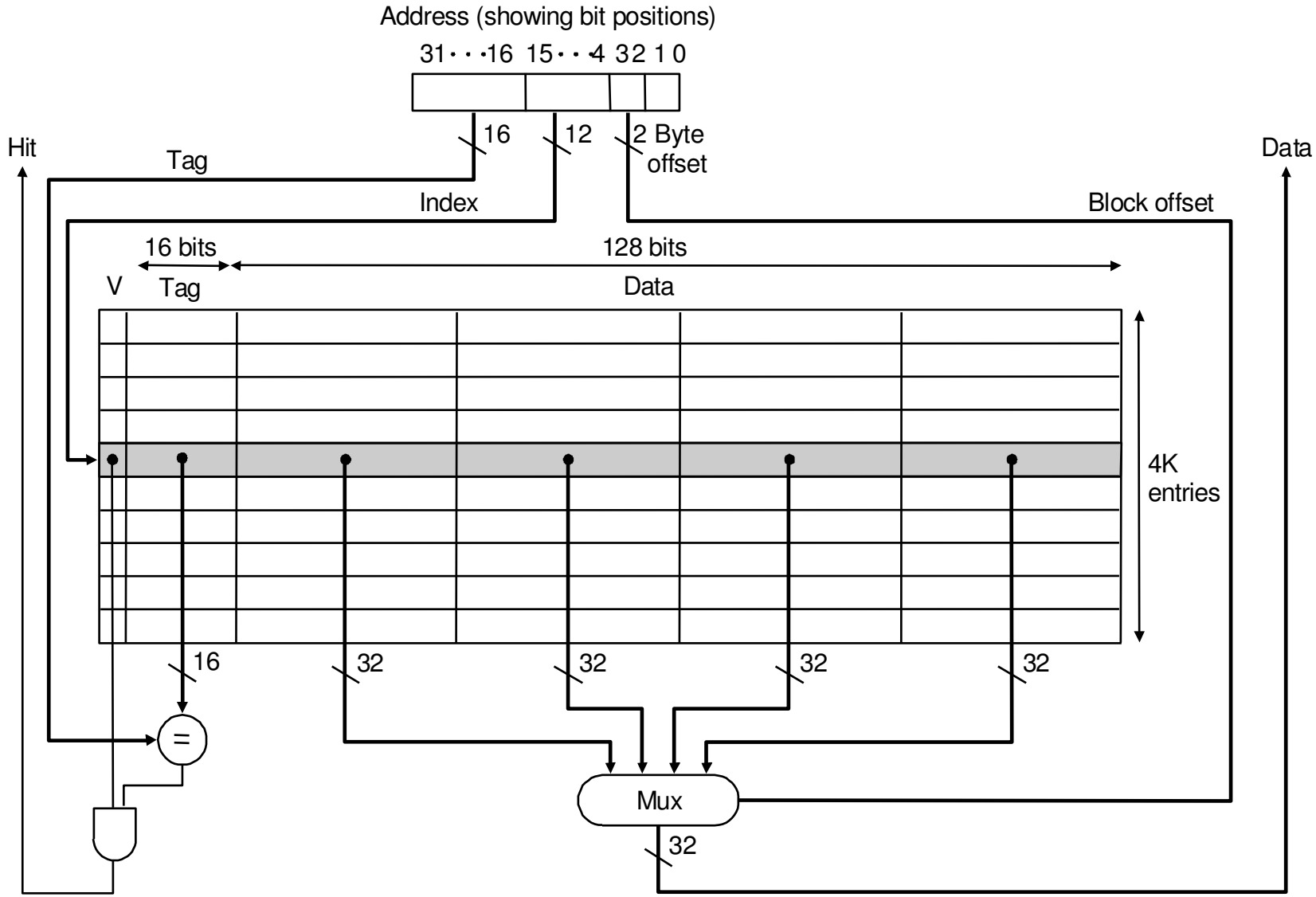
  - Performance com write-through → gcc tem 13% de instruções de store. Na DECStation 3100 a CPI para store é 1.2 e gasta 10 ciclos a cada escrita → nova CPI = 1.2+13% X 10 = 2.5 → reduz o desempenho por um fator maior que 2 → solução possível → write buffer.

Etapas para uma Escrita na Cache

- Outro esquema de atualização da memória ➔ write back ➔ a memória só é atualizada quando o bloco da cache que sofreu modificação for substituído por outro.

- Write miss ➔ dado escrito na cache pelo processador ➔ não há leitura da memória principal ➔ atualizar tag.

Localidade Espacial: Aumentando o Tamanho do Bloco



Hits vs. Misses (política de atualização ou escrita)

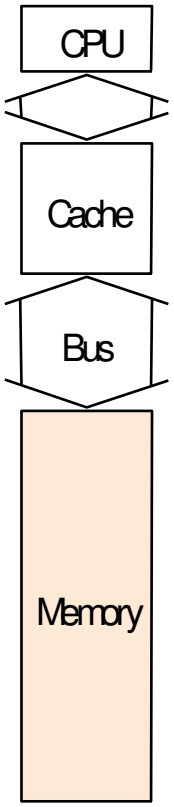
- **Read hits**
  - É o desejado
- **Read misses**
  - stall a CPU, fetch block da memória, preencher a cache
- **Write hits:**
  - atualiza o dado na cache e na memória (write-through)
  - atualiza o dado somente na cache (write-back the cache later)
    - também conhecida como copy-back
    - dirty bit

Hits vs. Misses (política de atualização ou escrita)

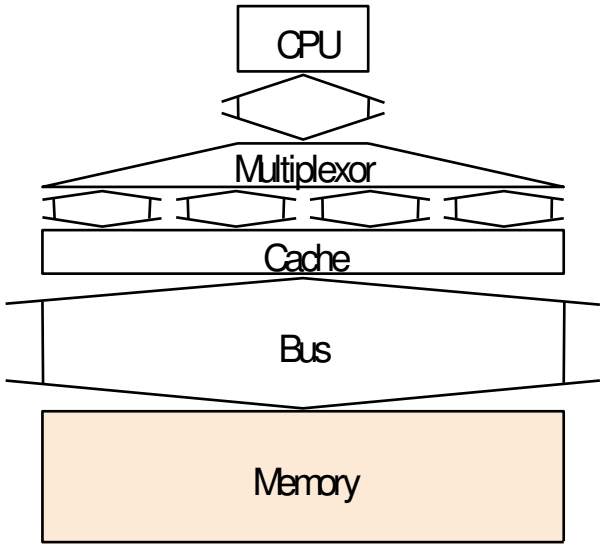
- **Write misses:**
  - ler o block e coloca-lo na cache, então escrever o dado

- **Comparação**
  - desempenho: write-back
  - confiabilidade: write-through
  - proc. paralelo: write-through

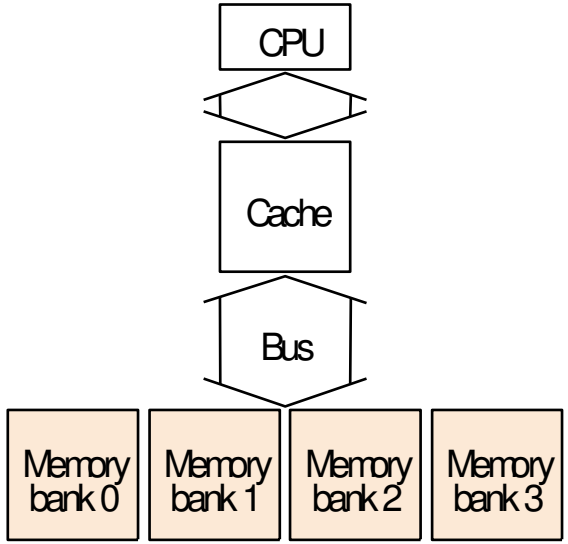
Largura da Comunicação Memória - Cache - CPU



a. One-word-wide memory organization



b. Wide memory organization



c. Interleaved memory organization

- **Supor:**
  - **1 clock para enviar endereço**
  - **15 clocks para ler DRAM**
  - **1 clock para enviar uma palavra de volta**
  - **linha da cache com 4 palavras**

Cálculo do Miss Penalty vs Largura Comunicação

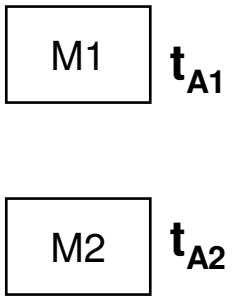
- Uma palavra de largura na memória:
  - 1 + 4*15 + 4*1 = 65 ciclos (miss penalty)
  - Bytes / ciclo para um miss: 4 * 4 / 65 = 0,25 B/ck
- Duas palavras de largura na memória:
  - 1 + 2*15 + 2*1 = 33 ciclos
  - Bytes / ciclo para um miss: 4 * 4 / 33 = 0,48 B/ck
- Quatro palavras de largura na memória:
  - 1 + 1*15 + 1*1 = 17 ciclos
  - Bytes / ciclo para um miss: 4 * 4 / 17 = 0,94 B/ck
  - Custo: multiplexador de 128 bits de largura e atraso

Cálculo do Miss Penalty vs Largura Comunicação

- Tudo com uma palavra de largura mas 4 bancos de memória interleaved (intercalada)
  - Tempo de leitura das memórias é paralelizado (ou superpostos)
    - Mais comum: endereço bits mais significativos
  - 1 + 1*15 + 4*1 = 20 ciclos
  - Bytes / ciclo para um miss: 4 * 4 / 20 = 0,8 B/ck
  - funciona bem também em escrita (4 escritas simultâneas):
    - indicado para caches com write through

Cálculo Aproximado da Eficiência do Sistema

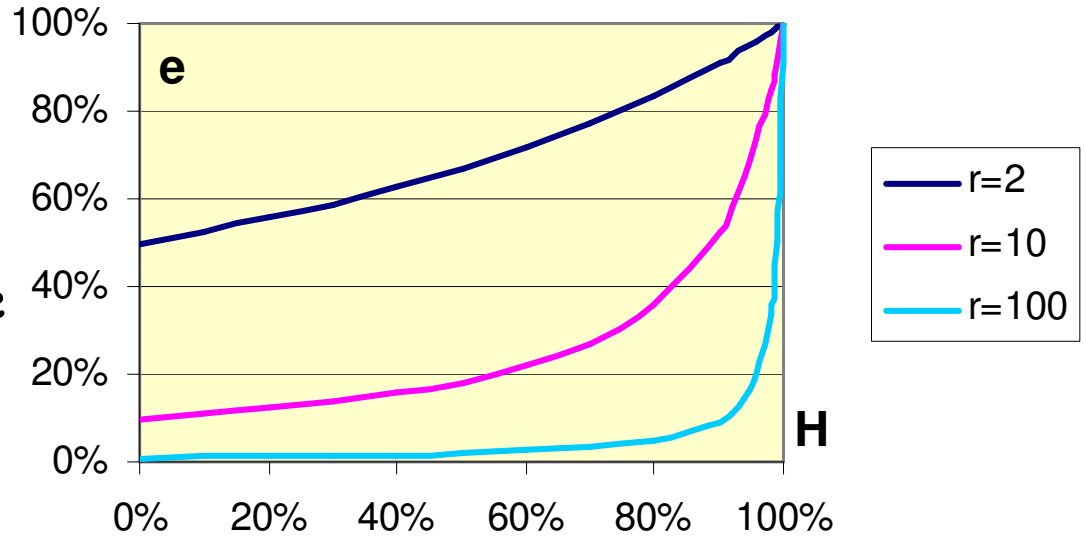
- **Objetivo:**
  - tempo de acesso médio = estágio mais rápido
- **Supor dois níveis:**
  - t_{A1} = tempo de acesso a M1
  - t_{A2} = tempo de acesso a M2 (M2+miss penalty)
  - t_A = tempo médio de acesso do sistema
  - $r = t_{A1} / t_{A2}$
  - e = eficiência do sistema = t_{A1} / t_A



$$t_A = H * t_{A1} + (1-H) * t_{A2}$$

$$t_A / t_{A1} = H + (1-H) * r = 1/e$$

$$e = 1 / [\, r + H * (1-r) \,]$$



Medida e Melhoria de Desempenho de Cache

- **Modelo simplificado de Desempenho**

execution time = (execution cycles + stall cycles) × cycle time

stall cycles = (RD + WR) stalls

RD stall cycles = # de RDs × RD miss ratio × RD miss penalty

WR stall cycles = # de WRs × WR miss ratio × WR miss penalty

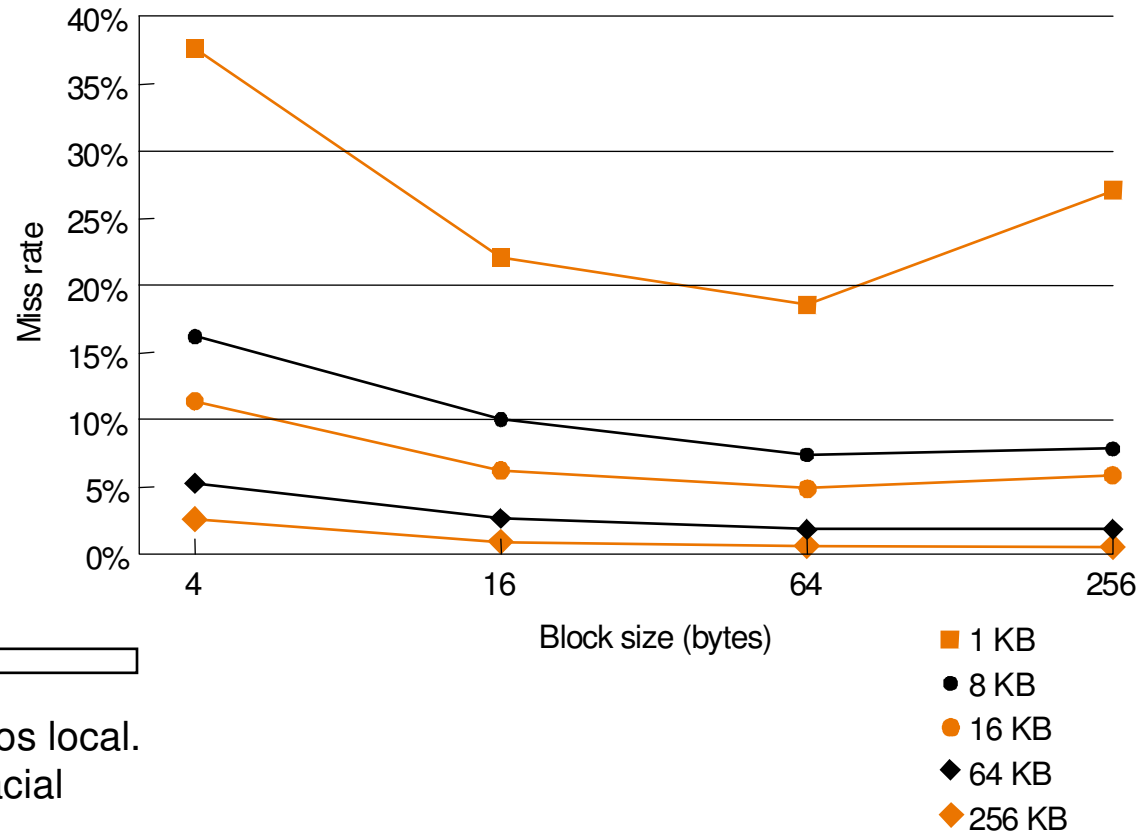
(mais complicado do que isto)

Medida e Melhoria de Desempenho de Cache

- **Melhoria de Desempenho**
  - Redução da probabilidade de dois blocos diferentes serem alocados na mesma linha de cache.
  - Redução do miss pela adição de mais um nível de cache na hierarquia (multilevel caching).

O que acontece se aumentarmos o tamanho do bloco?

Miss Rate vs Block Size



pior

menos local. espacial

pior

- fragmentação interna
- menos blocos
- miss penalty

Program	Block size in words	Instruction miss rate	Data miss rate	Effective combined miss rate
gcc	1	6.1%	2.1%	5.4%
	4	2.0%	1.7%	1.9%
spice	1	1.2%	1.3%	1.2%
	4	0.3%	0.6%	0.4%

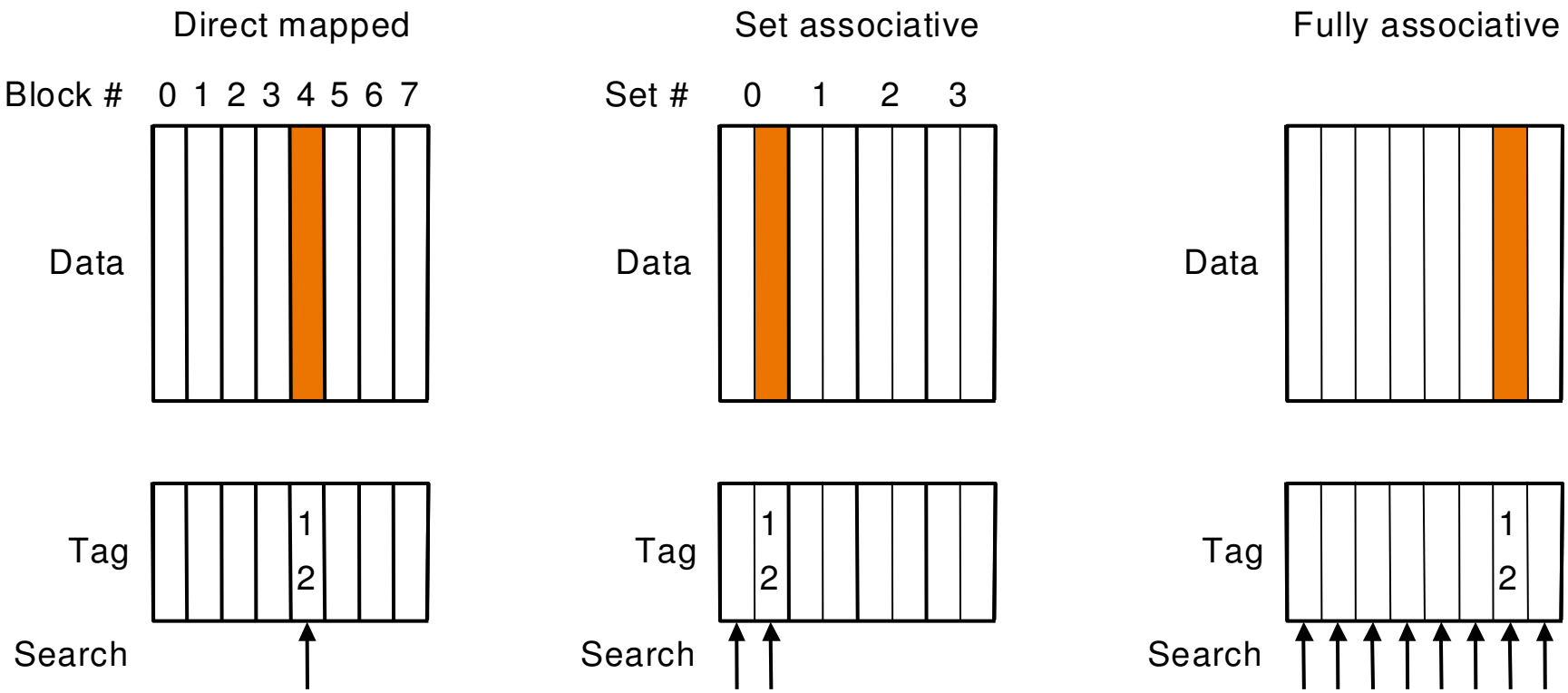
Exemplo pag 565 - 566

- gcc: instruction miss ratio = 2%; data cache miss rate = 4%
  - CPI = 2 (sem stalls de mem); miss penalty = 40 ciclos
  - Instructions misses cycles = I * 2% * 40 = 0.8 I
- Sabendo que lw+sw = 36%
  - data miss cycles = I * 36% * 4% * 40 = 0.58 I
- No. de stalls de mem = 0.8 I + 0.58 I = 1.38 I
  - CPI total = 2 + 1.38 = 3.38

Exemplo pag 565 - 566

- Relação de velocidades com ou sem mem stalls = rel de CPIs
  - 3.38 / 2 = 1.69
- Se melhorássemos a arquitetura (CPI) sem afetar a memória
  - CPI = 1
  - relação = 2.38 / 1 = 2.38
  - efeito negativo da memória aumenta (Lei de Amdhal)
- Ver exemplo da pag 567: aumento do clock tem efeito semelhante

Reduzindo o Miss Ratio com Associatividade



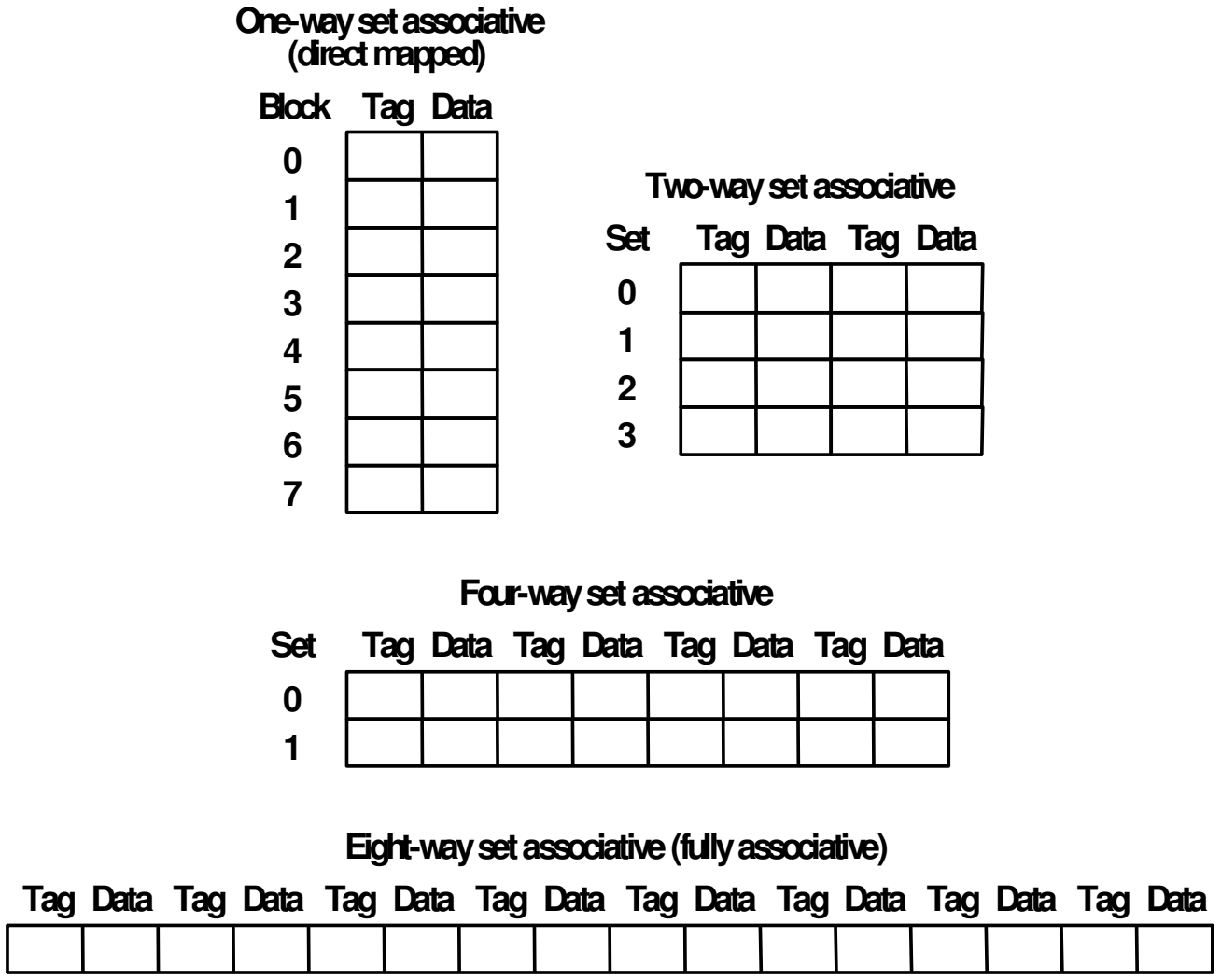
memory block position =

(Block number) mod (Number of cache blocks)

set que contém o memory block =

(Block number) mod (Number of sets in the cache)

Reduzindo o miss ratio com Associatividade



Exemplo

- Existem 3 pequenas caches de 4 blocos de uma palavra. Uma é fully associative, a segunda two-way set associative e a terceira é direct mapped. Encontre o número de misses para cada uma, dado a seguinte seqüência de endereços de blocos: 0, 8, 0, 6, 8

- Direct mapped

Block address	Cache block
0	0 mod 4 = 0
6	6 mod 4 = 2
8	8 mod 4 = 0

Exemplo

- Direct mapped
  - 5 misses

Endereço do bloco de memória acessado	Hit or miss	Conteúdo do bloco da cache após a referência			
		0	1	2	3
0	miss	mem[0]			
8	miss	mem[8]			
0	miss	mem[0]			
6	miss	mem[0]		mem[6]	
8	miss	mem[8]		mem[6]	

Exemplo

- 2-way set associative
  - 4 misses

Block address	Cache set
0	0 mod 2 = 0
6	6 mod 2 = 0
8	8 mod 2 = 0

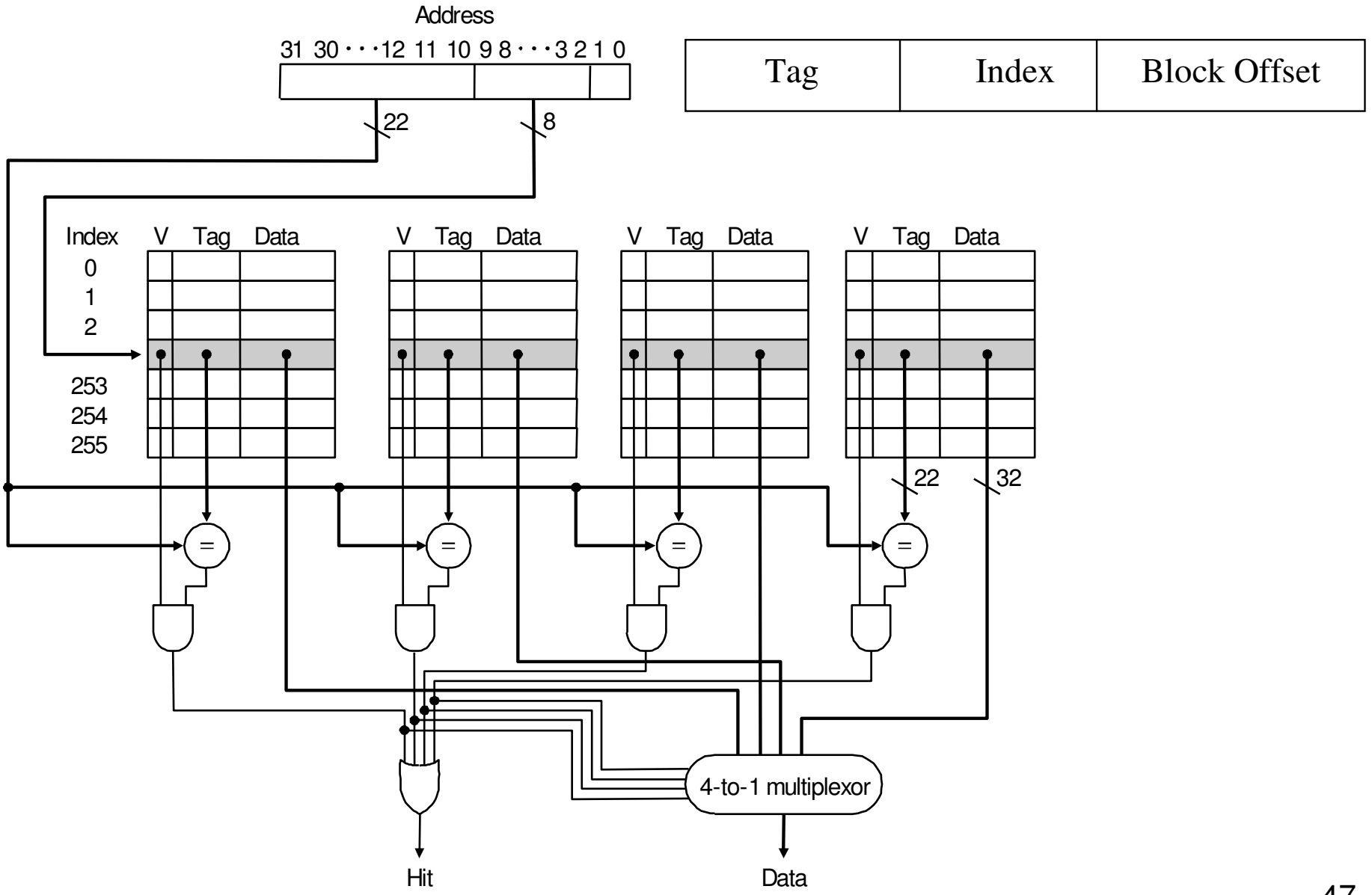
Endereço do bloco de memória acessado	Hit or miss	Conteúdo do bloco da cache após a referência			
		Set 0	Set 0	Set 1	Set 1
0	miss	mem[0]			
8	miss	mem[0]	mem[8]		
0	hit	mem[0]	mem[8]		
6	miss	mem[0]	mem[6]		
8	miss	mem[8]	mem[6]		

Exemplo

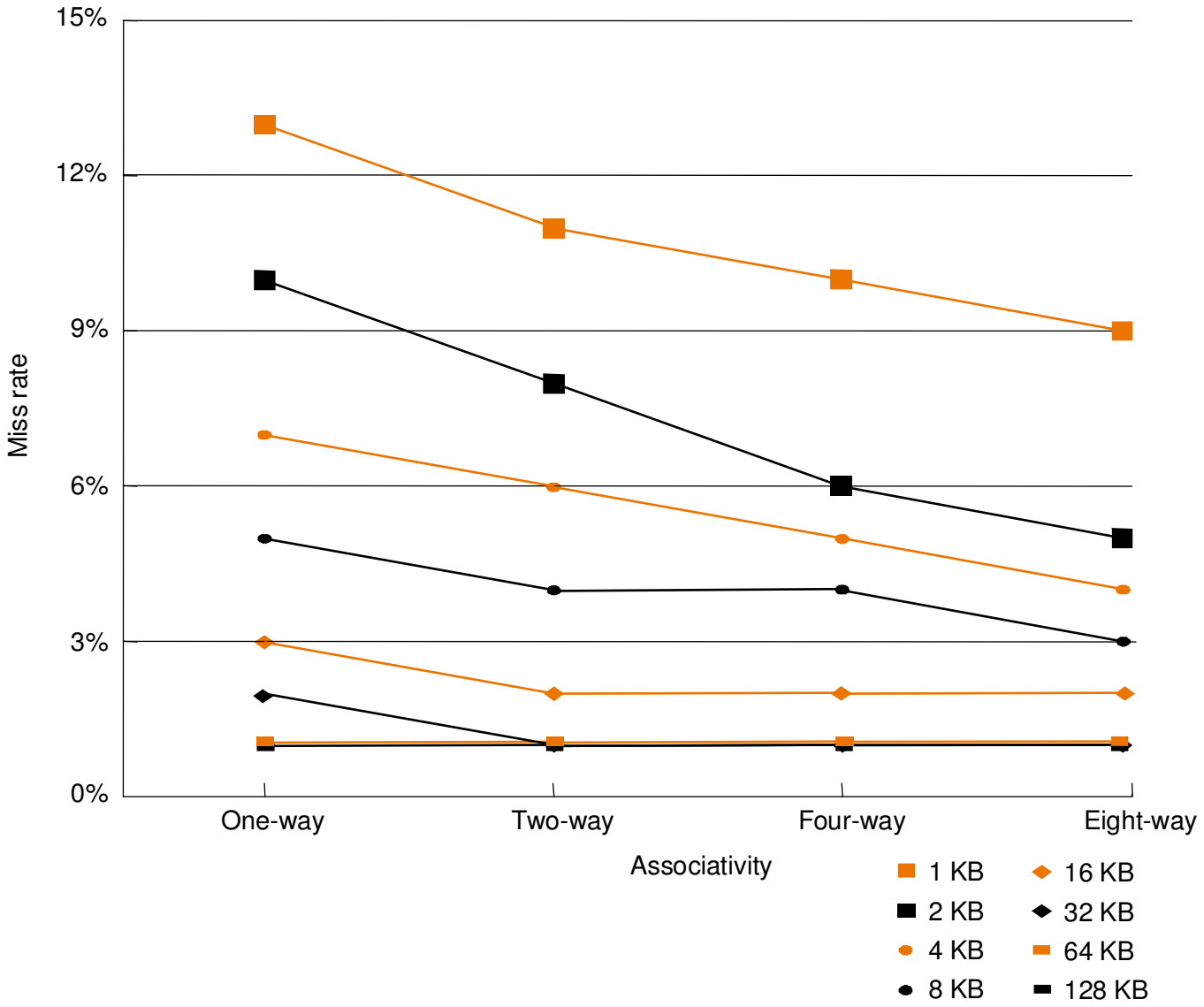
- Fully associative
  - 3 misses

Endereço do bloco de memória acessado	Hit or miss	Conteúdo do bloco da cache após a referência			
		bloco 0	bloco 1	bloco 2	bloco 3
0	miss	mem[0]			
8	miss	mem[0]	mem[8]		
0	hit	mem[0]	mem[8]		
6	miss	mem[0]	mem[8]	mem[6]	
8	hit	mem[0]	mem[8]	mem[6]	

Uma Implementação



Desempenho



Política de Substituição

- Qual item Descartar?
  - FIFO
  - LRU
  - Aleatoriamente

- ver seção 7.5