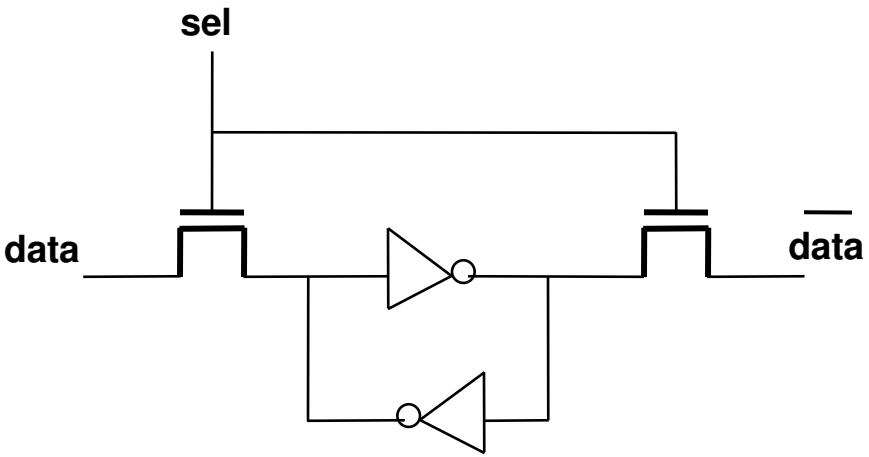
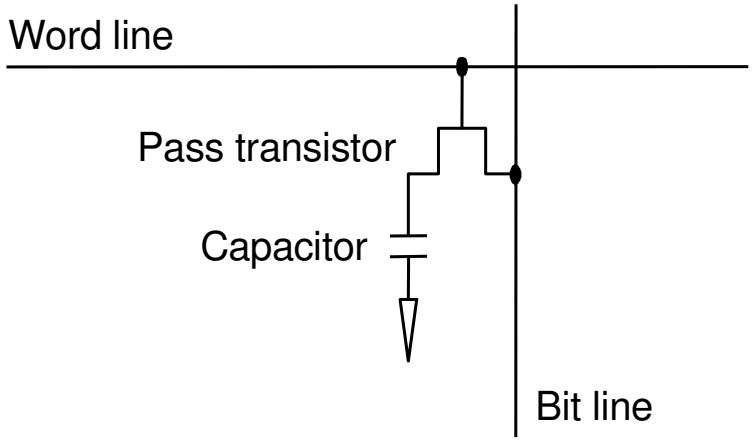
Capítulo 7
Sistemas de Memória

Memórias: Revisão

- **SRAM (Static RAM):**
  - **Valor é armazenado por meio da interligação de um par de inversores**
  - **Rápido, mas consome mais espaço que DRAM (4 a 6 transistores)**

- **DRAM (Dynamic RAM):**
  - **Valor é armazenado como carga em um capacitor (deve ser atualizado)**
  - **Pequeno, mas mais lento que SRAM**





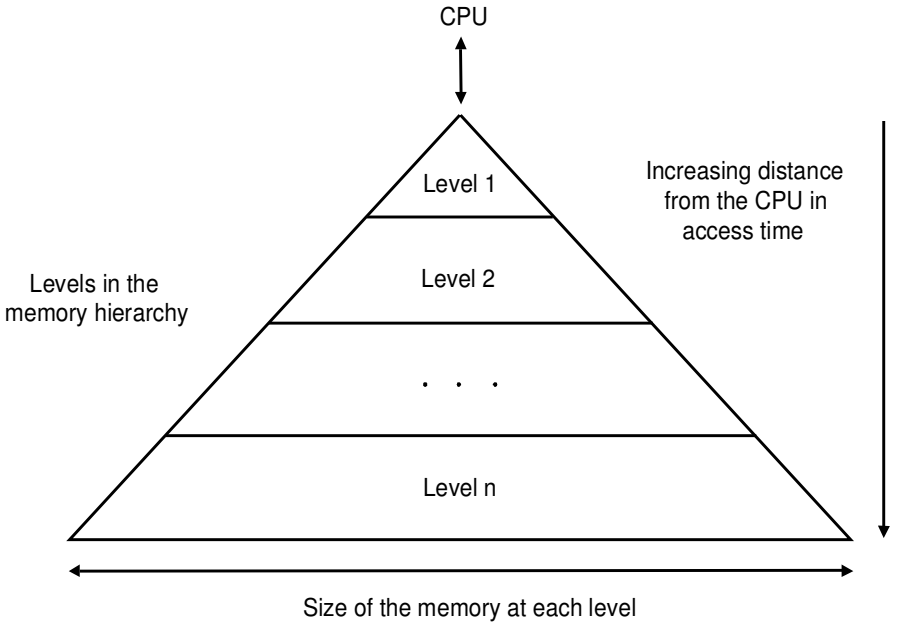
Hierarquia de Memória

- **Usuários querem memória suficiente e rápida**

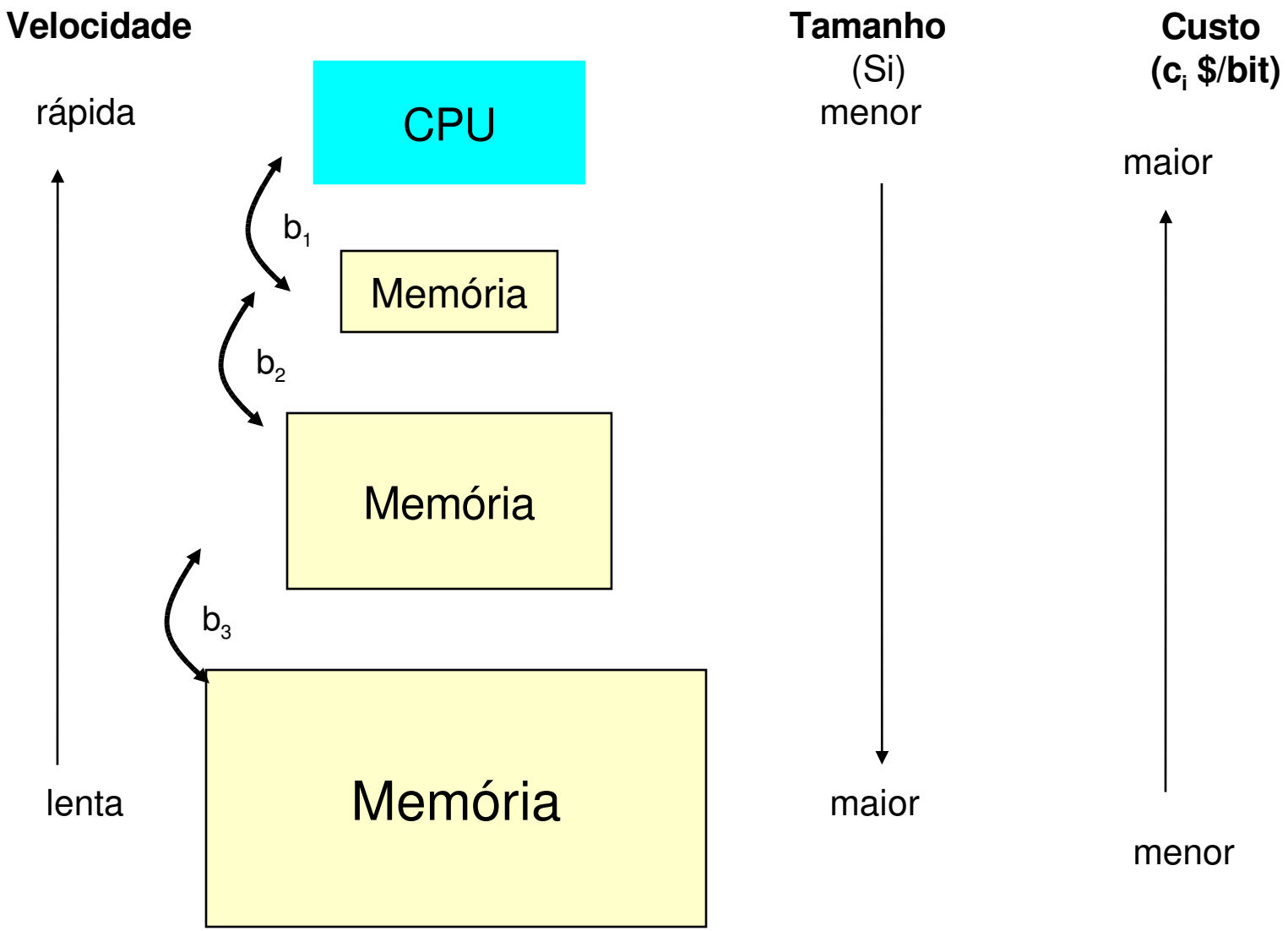
  Tempo de acesso SRAM 2 - 25ns no custo de $100-$250 por Mbyte
  Tempo de acesso DRAM 60-120ns no custo de $5-$10 por Mbyte
  Tempos de acesso ao disco são de 10-20 milhões de ns no custo de $.10-$.20 por Mbyte

1997

- **Uma hierarquia de memória ficaria assim:**



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)**

- **Hoje, assumindo disco 40 GB e memória de 256 MB**
  - **Calcular o custo médio por bit**

Localidade

- **Princípio que faz a hierarquia de memória ser uma boa idéia!**

- **Se um item (variável, objeto) na memória é referenciado...**

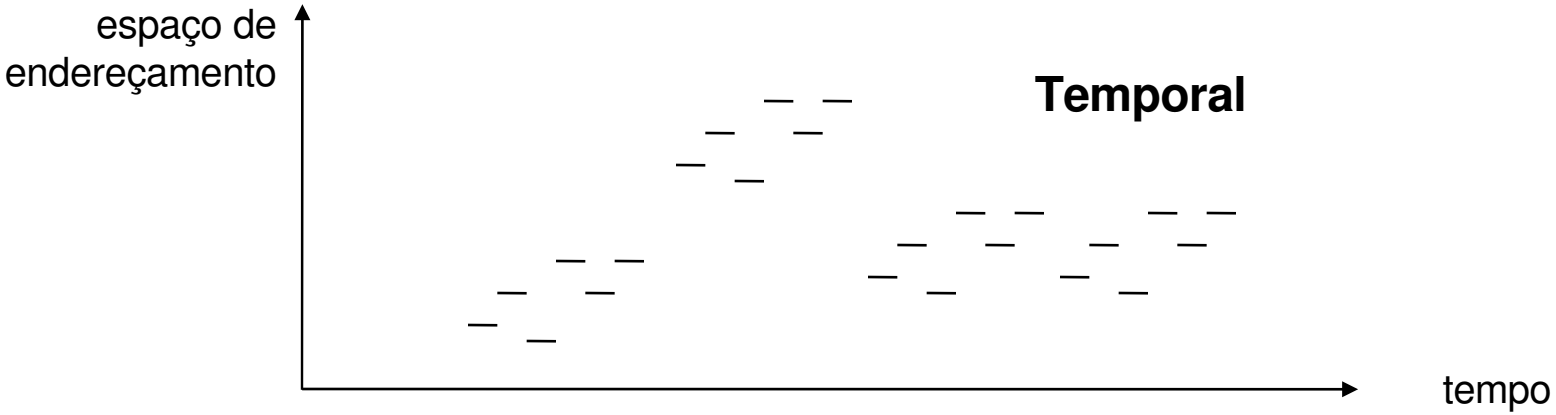
  Existe a tendência de ser referenciado novamente no futuro -> localidade temporal

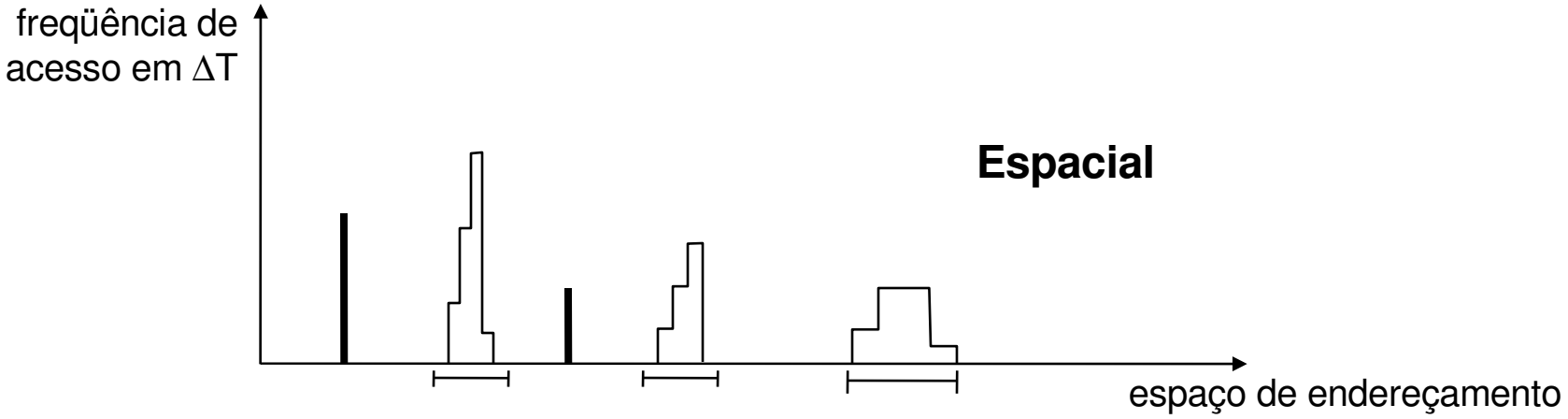
  Itens próximos tendem a ser referenciados -> localidade espacial

Entendendo a localidade: Considere dois níveis de memória (superior e inferior)

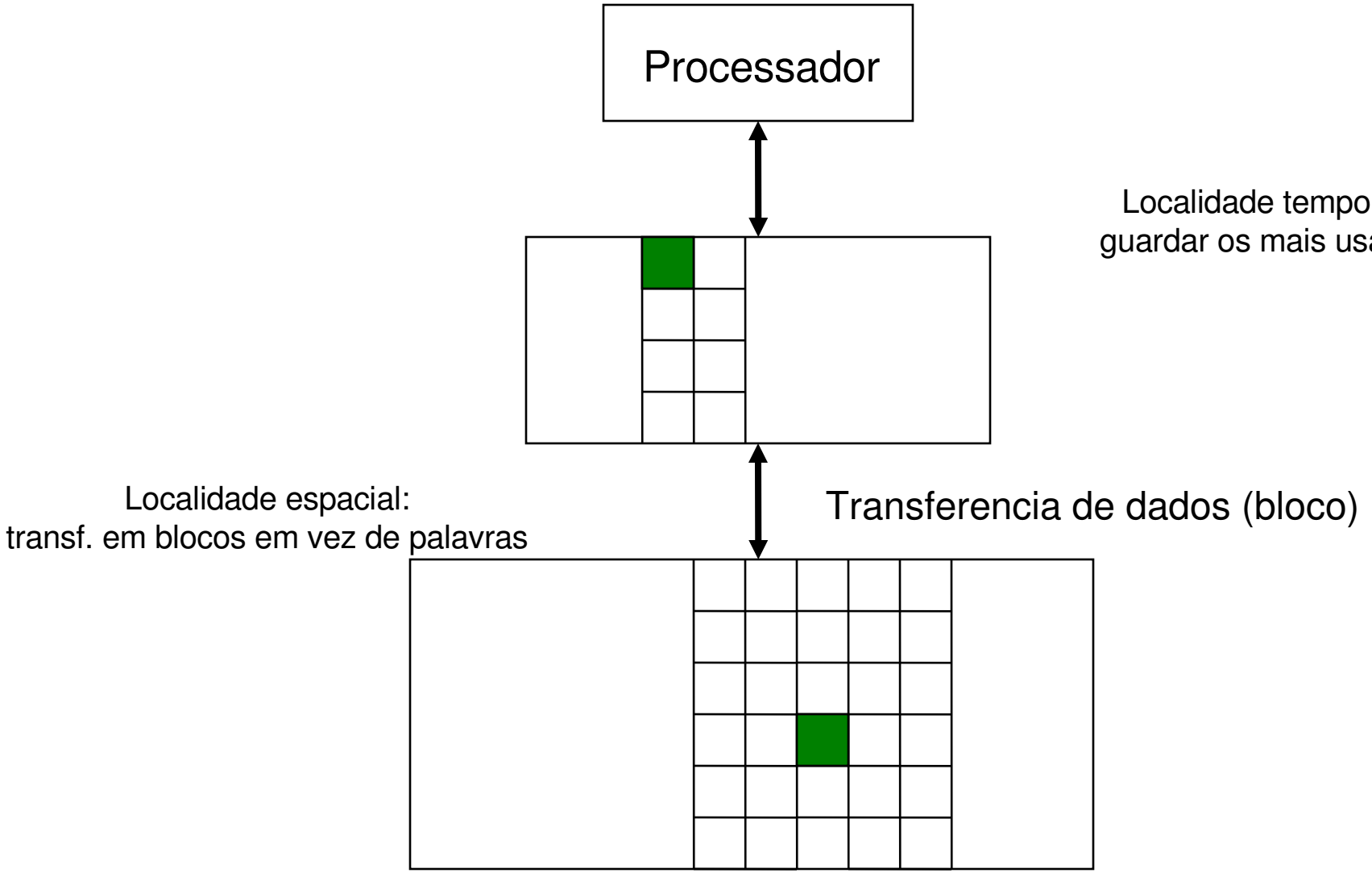
- **bloco: unidade de busca dos dados**
  - **hit: dado requisitado está no nível superior (hit ratio - hit time)**
  - **miss: dado requisitado não está no nível superior (miss ratio - miss penalty)**

Princípio da localidade



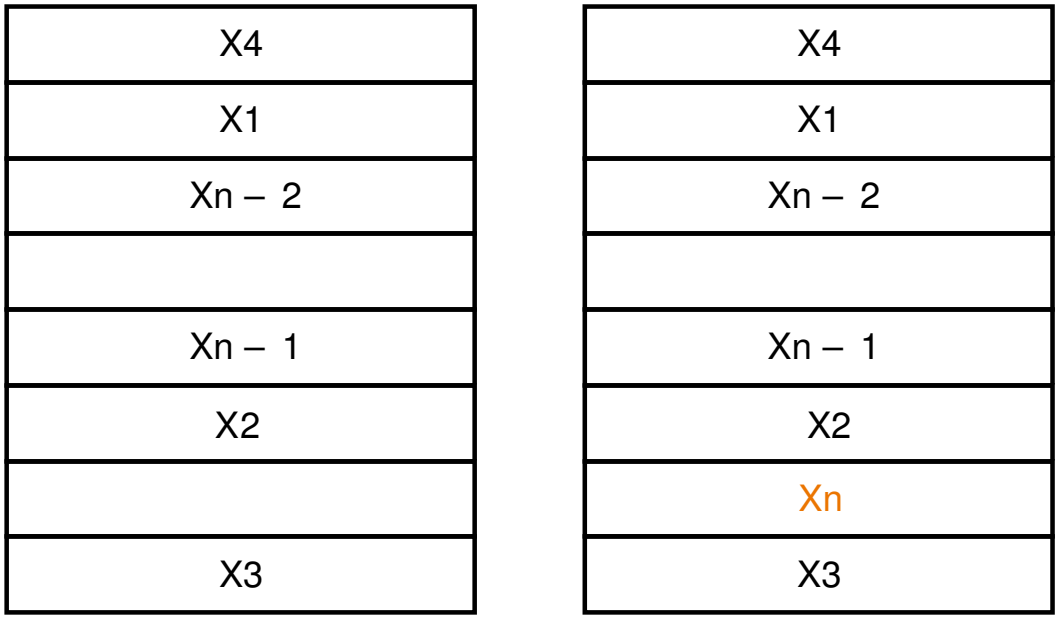


Visão em dois níveis



Cache

Referência à posição Xn



a. Before the reference to Xn

b. After the reference to Xn

Memória Cache

- **Questões:**
  - **Como saber se um dado x está na cache?**
  - **Se está na cache, como encontrá-lo?**
- **Considere que:**
  - **Tamanho do bloco é de uma palavra de dados**
  - **Mapeamento direto (*direct mapped*)**

Para cada item de dado do nível inferior, existe uma e somente uma localização na cache onde esse item pode estar.

Assim, muitos items de dados do nível inferior compartilham localizações no nível superior

  - **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

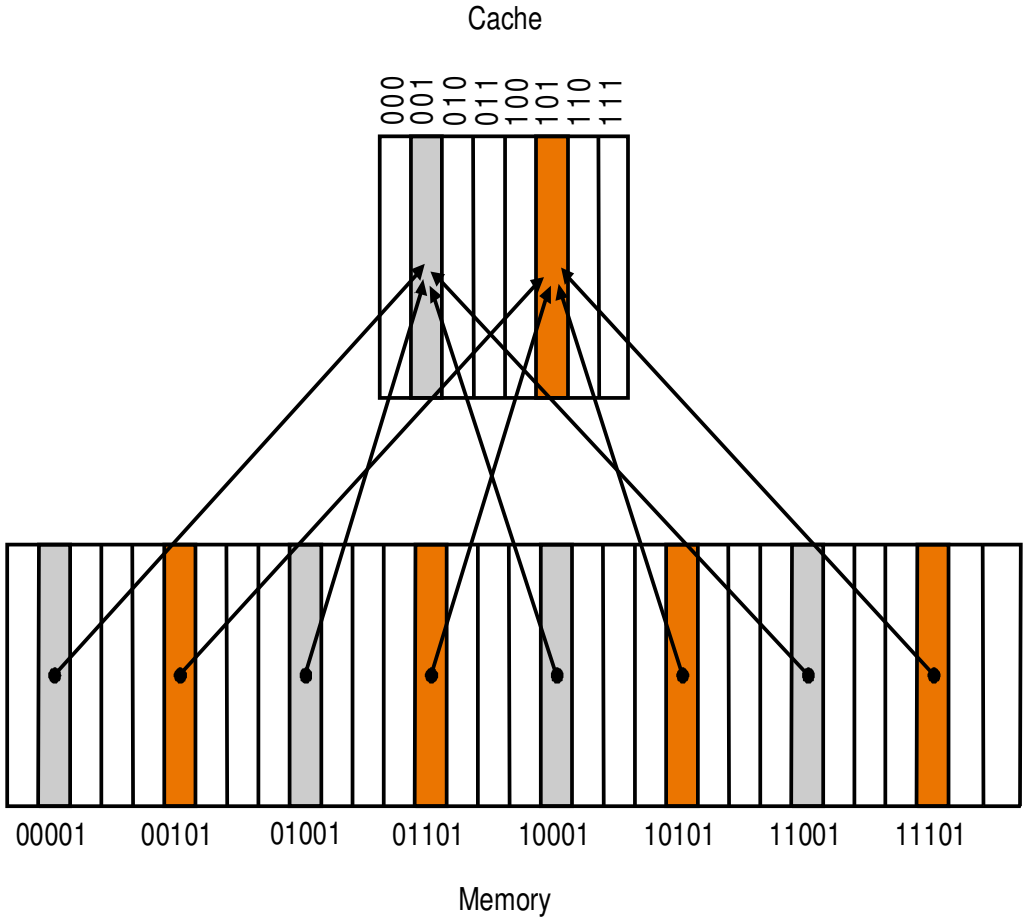
Cache com Mapeamento Direto

- **Mapeamento: endereço da cache consiste na operação de módulo entre o endereço da memória e o número de blocos da cache**

Ec=Em mod |Bc|

cache:
8 posições
3 bits de endereço

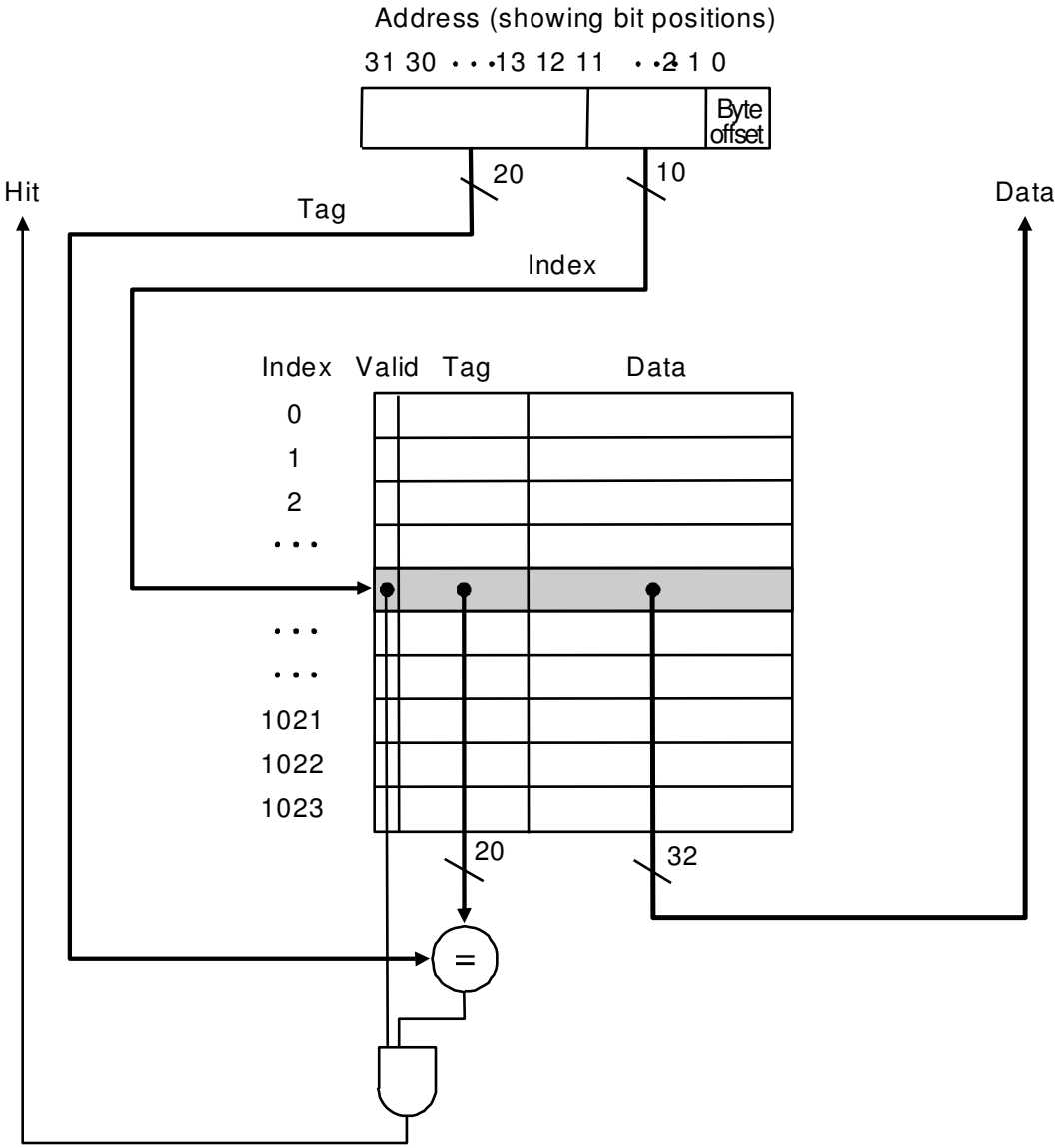
memória:
32 posições
5 bits de endereço



Cache de Mapeamento Direto

- 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)



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		

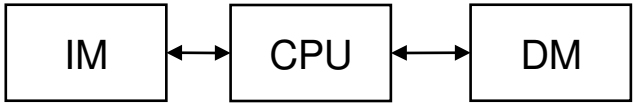
Index	V	Tag	Data
000	Y	10	M(10000)
001	N		
010	Y	11	M(11010)
011	Y	00	M(00011)
100	N		
101	N		
110	Y	10	M(10110)
111	N		

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	

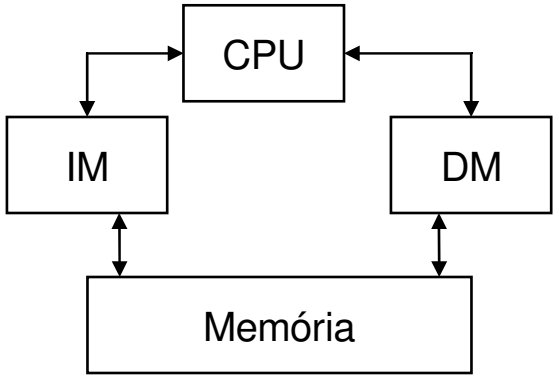
Cache na Via de dados com pipeline

- **Memória de Dados = cache de dados**
- **Memória de Instruções = cache de instruções**
- **Arquiteturas de memória:**
  - **de Harvard**
  - **ou Harvard modificada**

Harvard

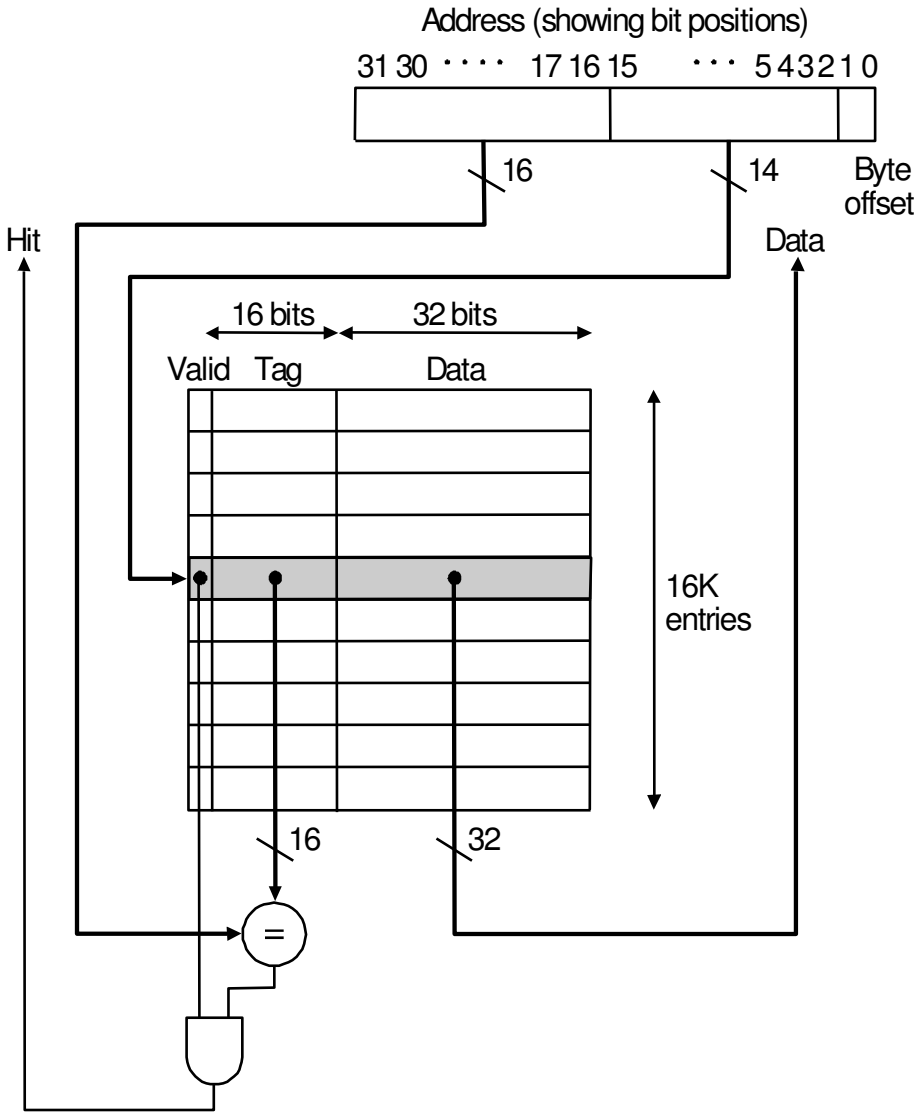


Harvard modificada

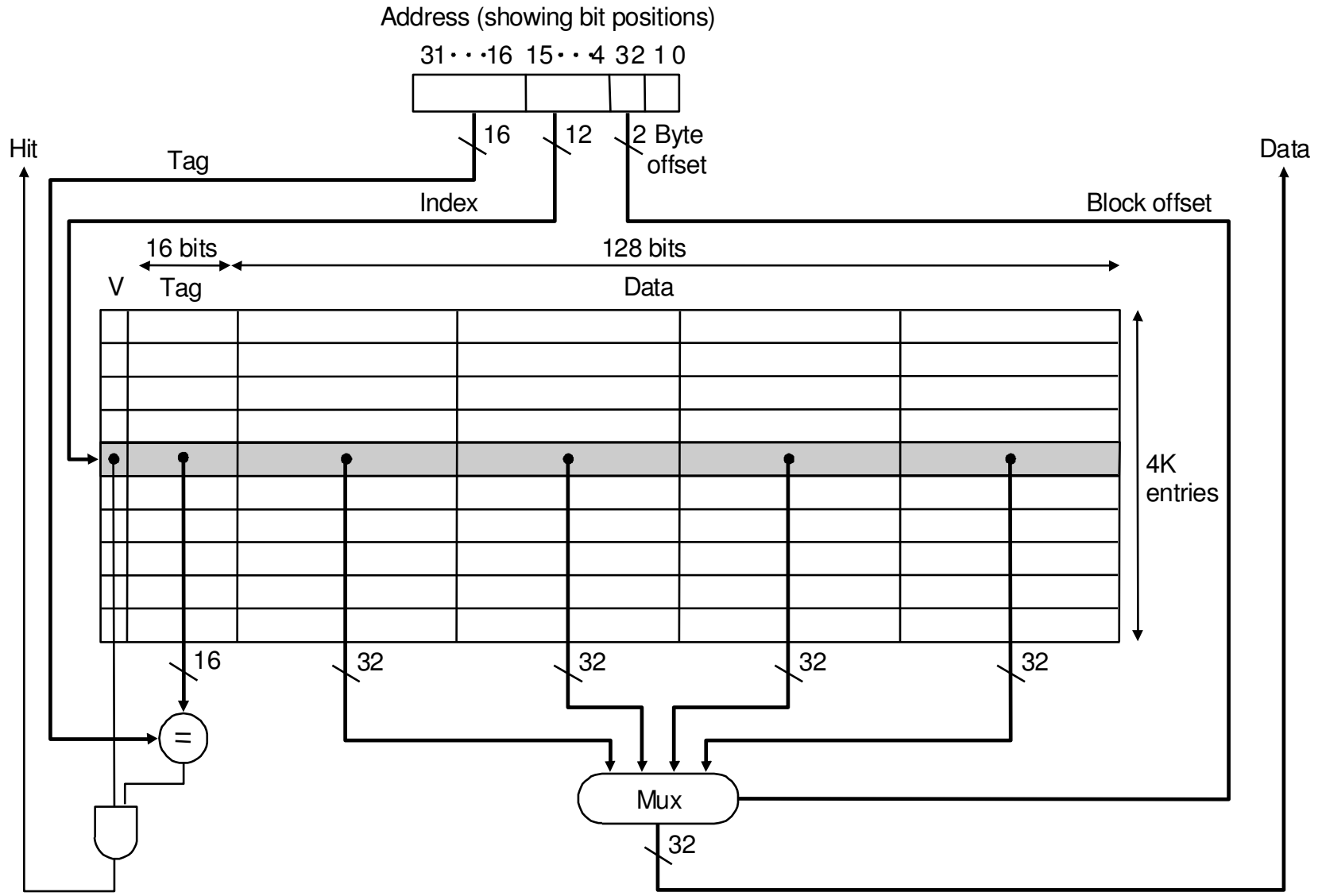


- **Qual é o efeito de um *Miss*?**
  - **semelhante ao stall!**
  - **dados: congela o pipeline**
  - **instrução:**
    - quem já entrou prossegue
    - inserir “bolhas” nos estágios seguintes
    - esperar pelo hit
  - **enquanto instrução não é lida, manter endereço original (PC-4)**

O esquema de caches no Processador DECStation 3100



Localidade espacial: aumentando o tamanho do bloco



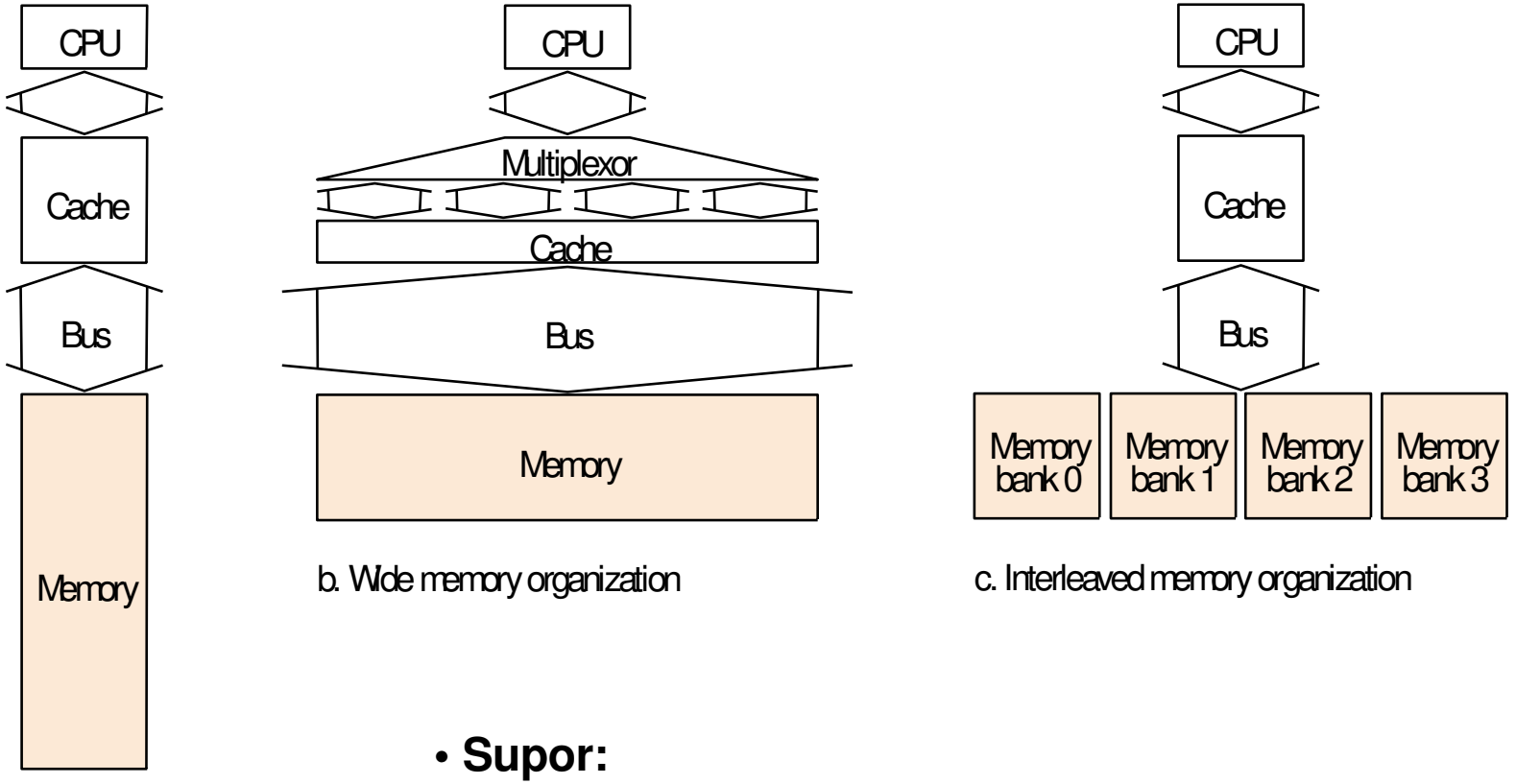
Hits vs. Misses (política de atualização ou escrita)

- **Hits na leitura: efeito desejado**

- **Misses na leitura: provocar um stall, buscar bloco de dados na memória, trazer esse bloco para a cache, reiniciar a leitura**

- **Hits na escrita:**
  - **Pode substituir o dado na cache e memória (política *write-through*)**
  - **ou...substituir o dado apenas na cache (write-back)**
    - também conhecida como copy-back
    - dirty bit
- **Misses na escrita:**
  - **Buscar o bloco e trazê-lo para a cache, substituir o bloco**
- **Comparação**
  - **desempenho: write-back**
  - **confiabilidade: write-through**
  - **proc. paralelo: write-through**

Largura da comunicação Mem - Cache – CPU



a. One-word-wide memory organization

b. Wide memory organization

c. Interleaved memory organization

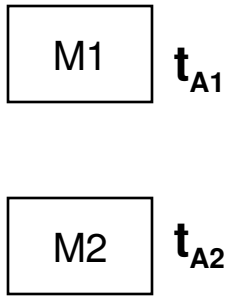
- **Supor:**
  - **1 clock para enviar endereço**
  - **15 clocks para ler DRAM**
  - **1 clock para enviar uma palavra de volta**
  - **linha da cache com 4 palavras**

Cálculo da 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**
- **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

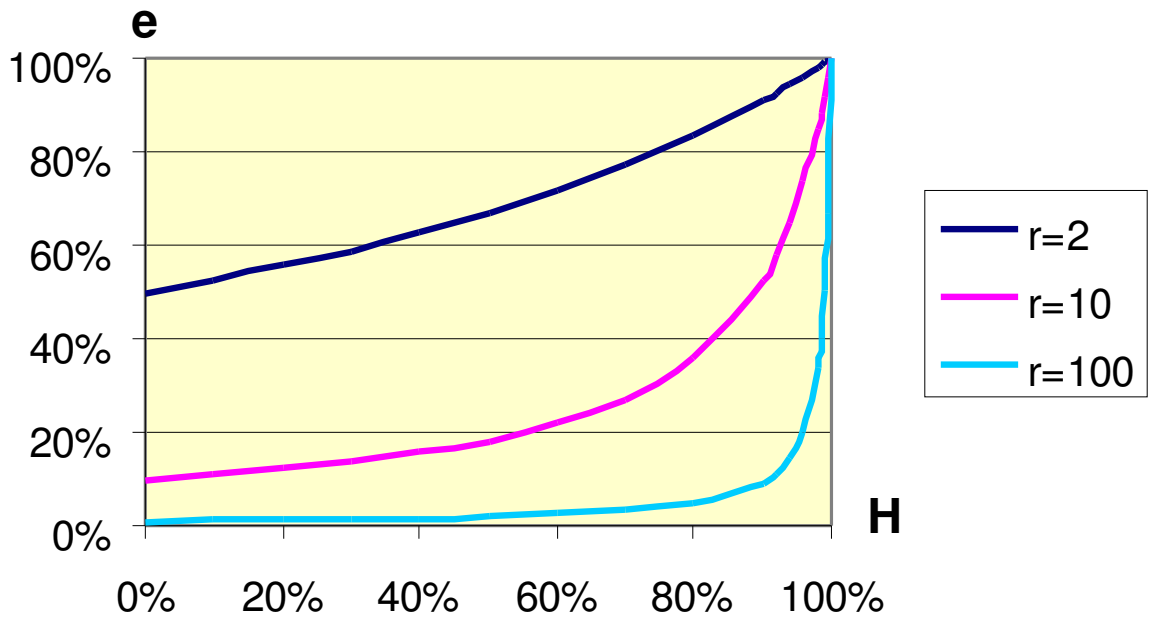
- objetivo:
  - tempo de acesso médio = estágio mais rápido
- supor dois níveis:
  - t_{A1} = tempo de acesso a M1
  - t_{A2} = tempo de acesso a M2 (M2+miss penalty)
  - t_A = tempo médio de acesso do sistema
  - $r = t_{A2} / t_{A1}$
  - e = eficiência do sistema = t_{A1} / t_A



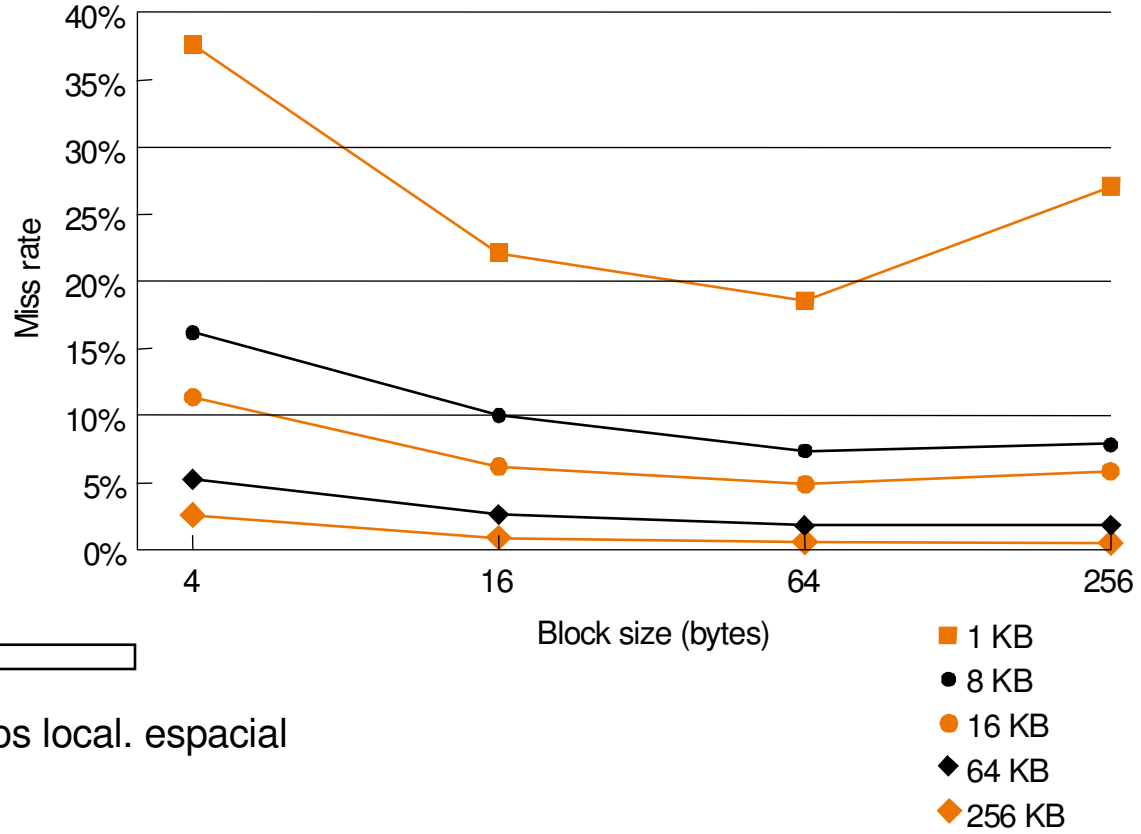
$$t_A = H * t_{A1} + (1-H) * t_{A2}$$

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

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



Taxa de misses (Miss Rate) vs Tamanho do Bloco



pior

menos local. espacial

pior

- fragmentação interna
- menos blocos
- miss penalty

Use split caches because there is more spatial locality in code:

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%

Como calcular o desempenho?

- **Modelo simplificado:**

  tempo de execução = (ciclos de execução + ciclos de stall) × tempo de ciclo
  ciclos de stall = RD + WR ciclos de stall

  RD ciclos de stall = # de RDs × RD miss ratio × RD miss penalty
  WR ciclos de stall = # de WRs × WR miss ratio × WR miss penalty

- **Duas possíveis formas de melhorar o desempenho:**
  - **Reduzir a taxa de *miss***
  - **Reduzir a penalidade do *miss***

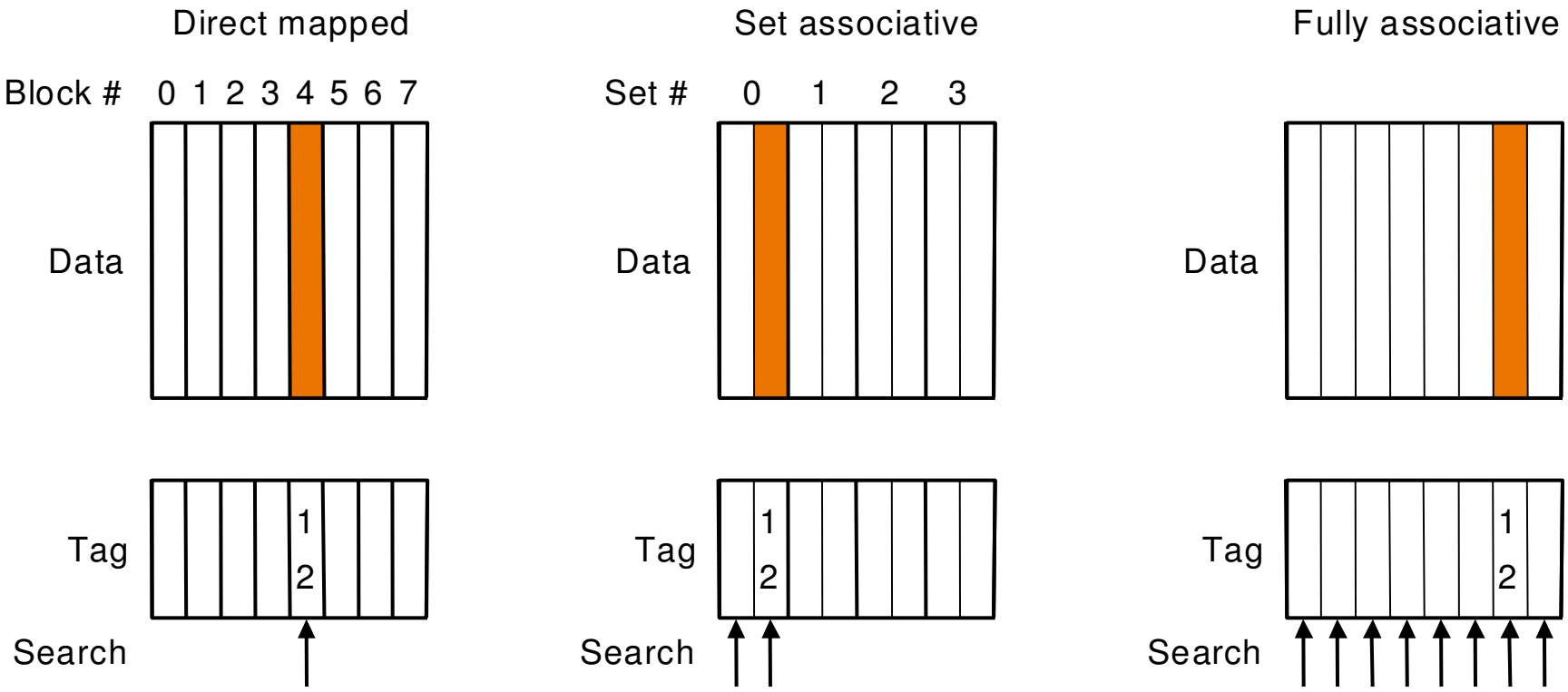
O que acontece se aumentar o tamanho do bloco?

Exemplo

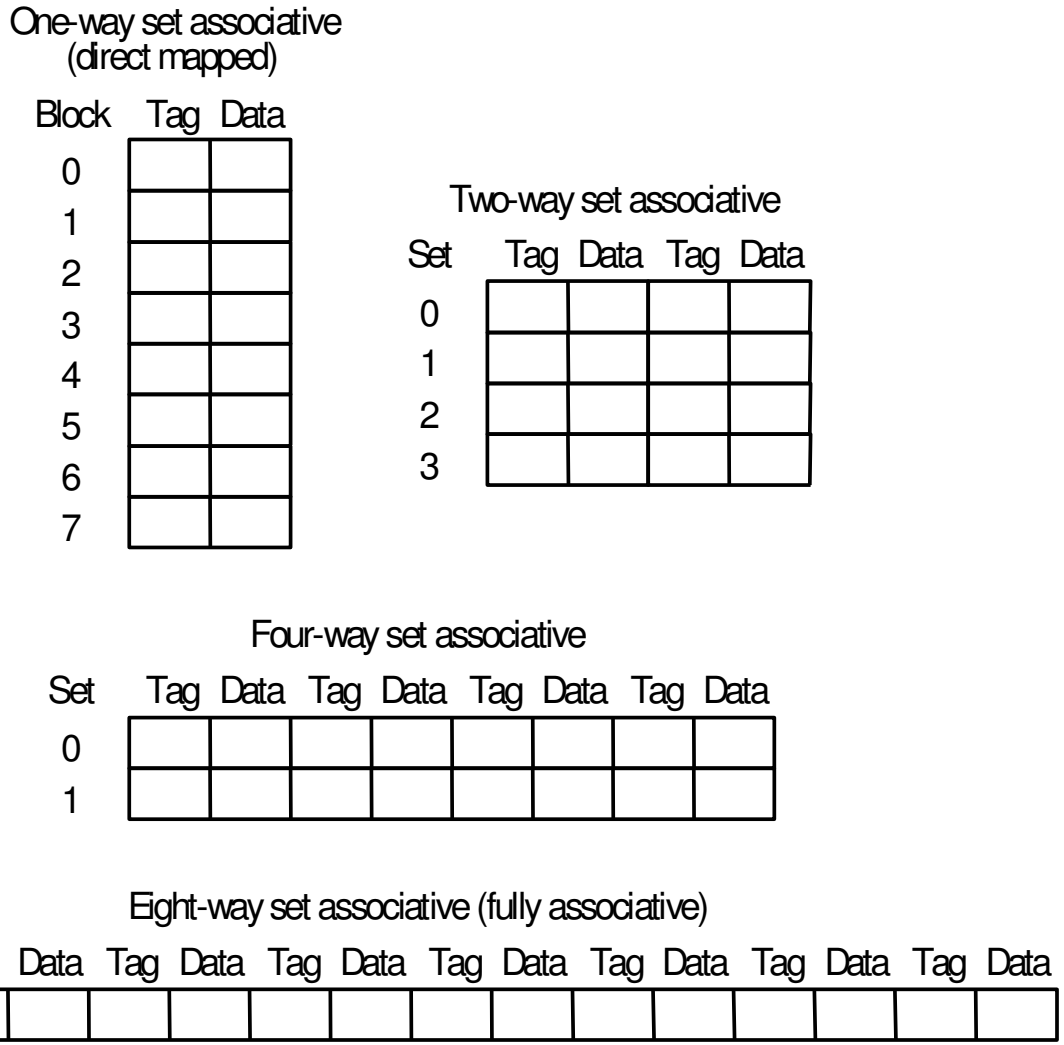
- **gcc: instruction miss ratio = 2%; data cache miss rate = 4%**
- **CPI = 2 (sem stalls de mem); miss penalty = 40 ciclos**
- **Ciclos de misses das instruções = I * 2% * 40 = 0.8 I**
- **Sabendo que lw+sw= 36%**
  - **Ciclos de misses de dados = I * 36% * 4% * 40 = 0.58 I**
- **Stalls de memória = 0.8 I + 0.58 I = 1.38 I**
  - **CPI total = 2 + 1.38 = 3.38**
- **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)**

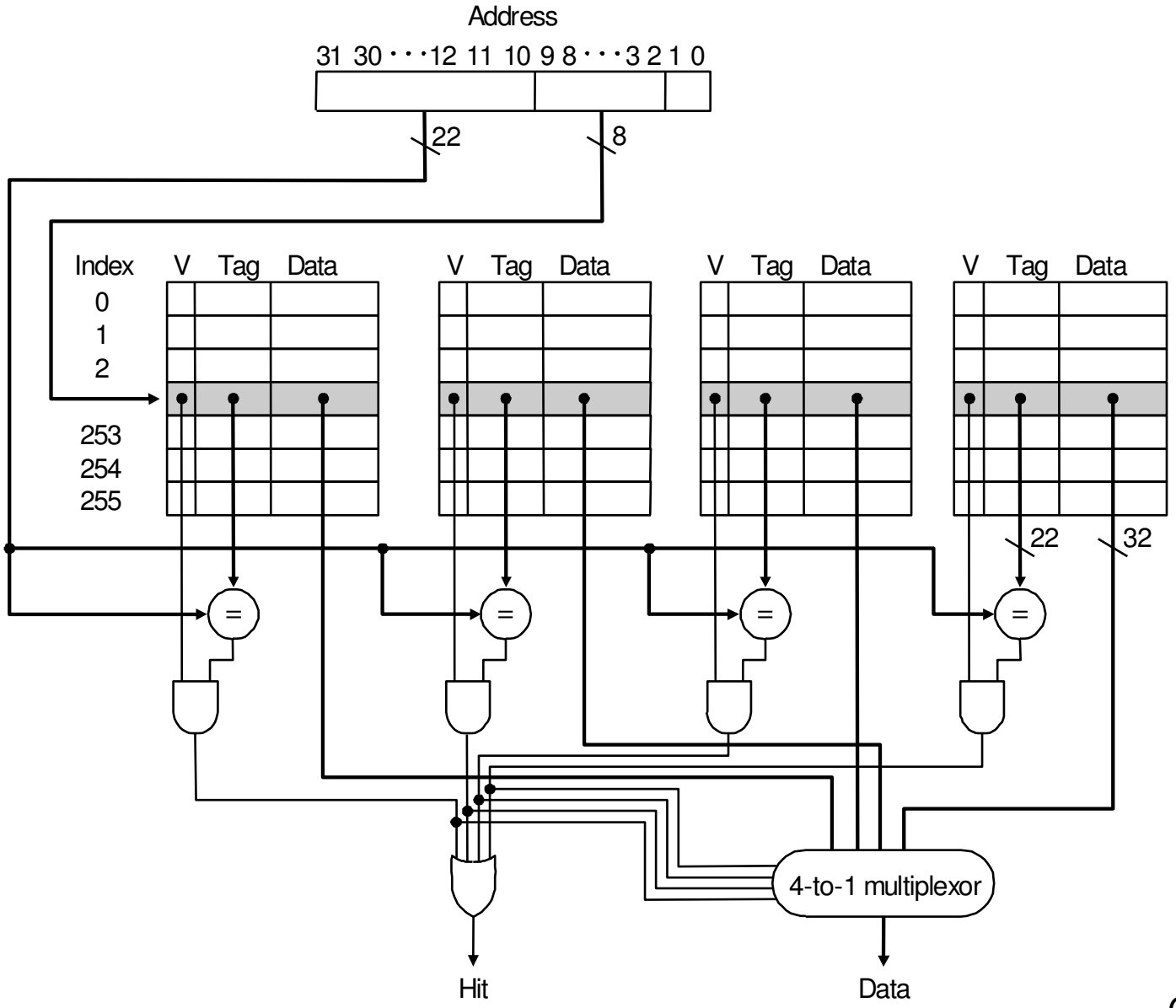
Reduzindo a taxa de miss com associatividade



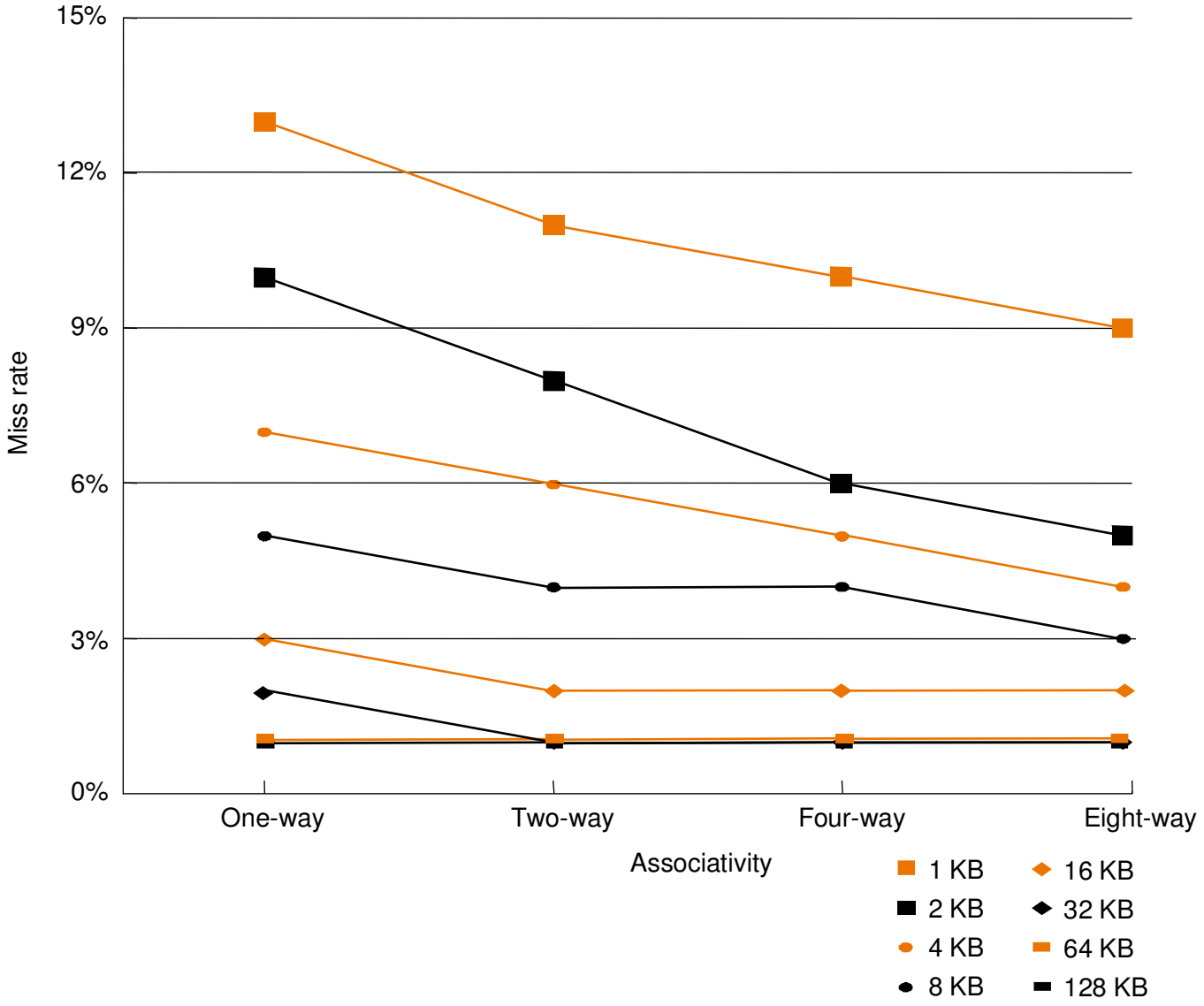
Reduzindo a taxa de miss com associatividade



Uma implementação de caches associativas



Desempenho


15%
12%
9%
6%
3%
0%
Miss rate
One-way
Two-way
Four-way
Eight-way
Associativity
1 KB
2 KB
4 KB
8 KB
16 KB
32 KB
64 KB
128 KB

Política de substituição

- **Qual item descartar?**
  - **FIFO**
  - **LRU**
  - **Aleatoriamente**

Reduzindo miss penalty com caches de múltiplos níveis

- **Adicionar um segundo nível de cache:**
  - **Penalidade do miss é reduzida pois o dados procurado pode estar na cache de 2o. Nível**

- **Exemplo :**
  - **CPI de 1.0 em uma máquina de 500MHz com 5% taxa de miss, 200ns acesso DRAM**
  - **Adicionar cache de 2nd nível com time de acesso de 20ns e miss rate de 2%**
  - **miss penalty (só L1) = 200ns/período = 100 ciclos**
  - **CPI (só L1)= CPIbase + clocks perdidos = 1 + 5% * 100 = 6**
  - **miss penalty (L2)= 20ns/período = 10 ciclos**
  - **CPI (L1 e L2)= 1 + stalls L1 + stalls L2 = 1 + 5% * 10 + 2% * 100 = 3.5**
  - **ganho do sistema em velocidade com L2 = 6.0 / 3.5 = 1.7**