

[cin.ufpe.br](http://cin.ufpe.br)



# Centro de Informática

U • F • P • E



UNIVERSIDADE FEDERAL DE PERNAMBUCO

# Memória e Hierarquia de Memória

# Roteiro da Aula



- Hierarquia de Memória
- Localidade temporal e espacial
- Memória cache
  - Tipos
  - Princípio de funcionamento
- Melhorando o desempenho
  - Aumento do bloco
  - Aumento da associatividade
  - Reduzindo a penalidade
- Memória principal
  - Melhorando a taxa de transferência

# Memória Vs. Armazenamento



Fichário



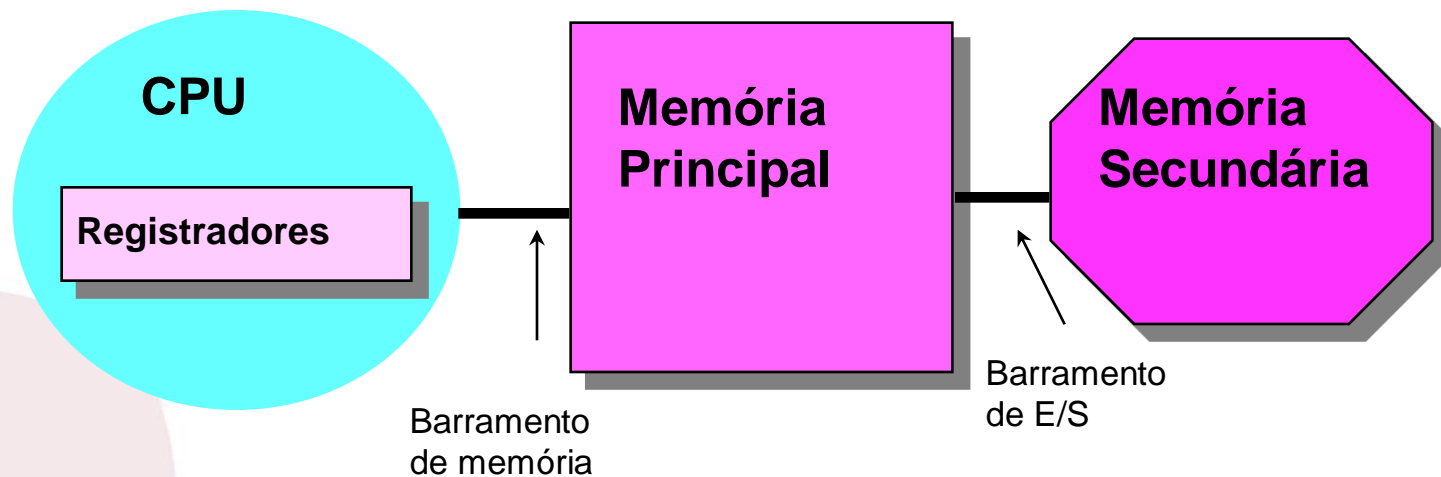
Pasta



- O fichário representa o disco rígido, com alta capacidade de armazenamento.
- A pasta sobre a mesa representa a memória, de acesso rápido e fácil
- Mesa e usuário são a CPU
- OBS: Memória é volátil e disco não.
  - Faxineira no final do expediente



# Sistema Hierárquico de Memória



# Nomenclatura Básica



- RAM = Random Access Memory
- SRAM = Static RAM
- DRAM = Dynamic RAM
- VRAM - Video RAM
- ROM = Read Only Memory
- PROM = Programmable ROM
- EPROM = Erasable PROM
- EEPROM = Electrically Erasable PROM (apagamento byte a byte)
- Flash EPROM = Fast erasable EPROM (apagamento por bloco)



# Tipos Básicos de Memória Semicondutora

<i>Tipo de Memória</i>	<i>Categoria</i>	<i>Apagamento</i>	<i>Escrita</i>	<i>Volatilidade</i>
Random-Access Mem. (RAM)	Read-Write	Elétrico byte a byte	Elétrica	Volátil
Read-Only Mem. (ROM)	Read-only	Impossível	Máscara	não-volátil
Programmable ROM (PROM)				
Erasable PROM (EPROM)	Read-mostly	Ultra-violeta	Elétrica	
Electrically EPROM (EEPROM)		Elétrico byte a byte		
Flash EPROM		Elétrico por bloco		

# RAM Dinâmica vs. Estática



- DRAM (Dynamic Random Access Memory)
  - **Grande capacidade de integração (baixo custo por bit)**
  - **Perda de informação após algum tempo: Necessidade de refreshing**
- SRAM (Static Random Access Memory)
  - **Pequeno tempo de acesso**
  - **Não existe necessidade de refreshing**
  - **Alto custo por bit (baixa integração)**







# Memory Technology

- Static RAM (SRAM)
  - 0.5ns – 2.5ns, \$2000 – \$5000 per GB
- Dynamic RAM (DRAM)
  - 50ns – 70ns, \$20 – \$75 per GB
- Magnetic disk
  - 5ms – 20ms, \$0.20 – \$2 per GB
- Memória Ideal
  - Tempo de acesso de uma SRAM
  - Capacidade e custo/GB de um disco

# Evolução Tecnológica

**1000:1** 

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

 **2:1**

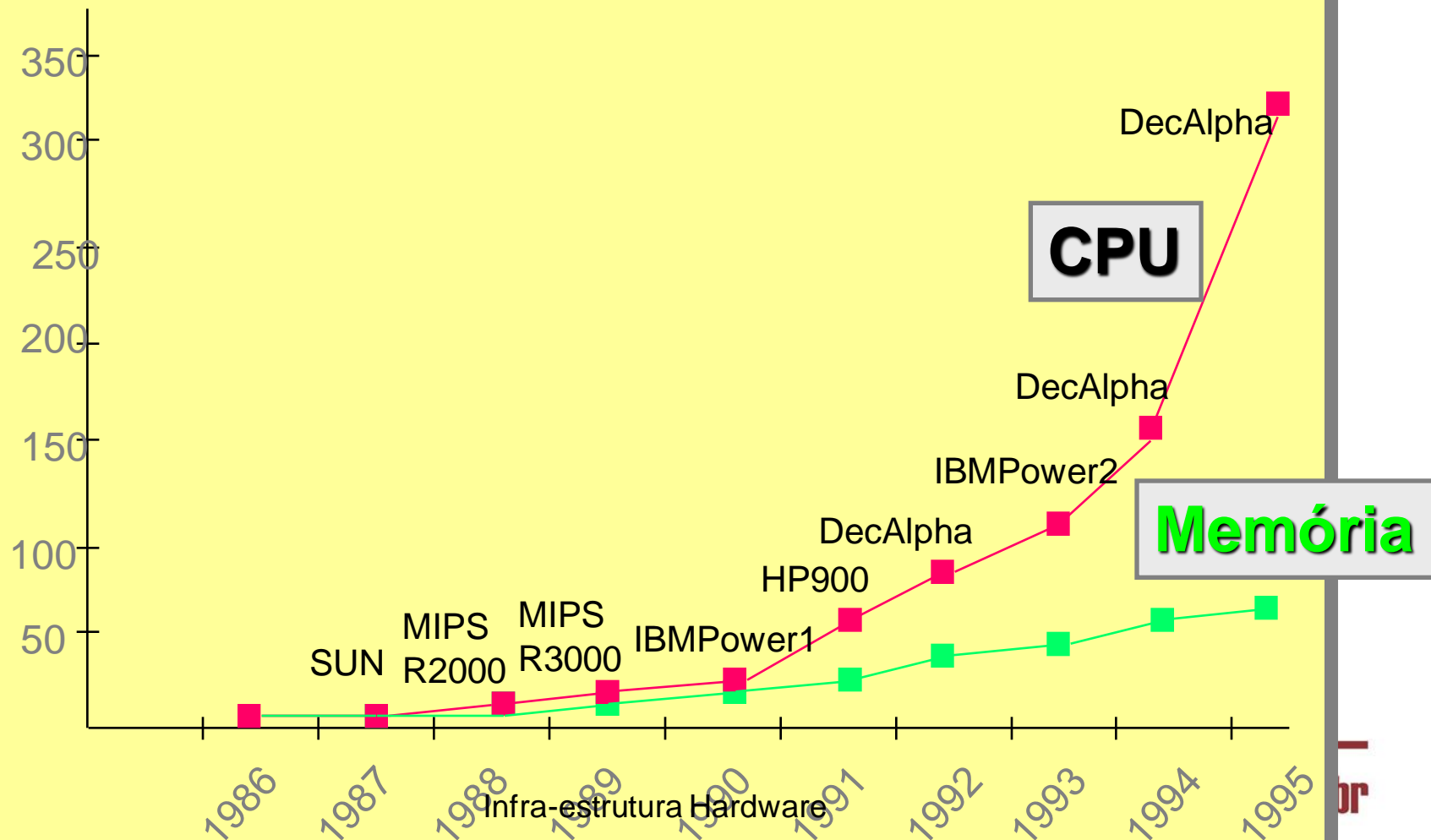
	Aumento da capacidade	Aumento da velocidade
SRAM	2x em 3 anos	2x em 3 anos
DRAM	4x em 3 anos	2x em 10 anos
Disco	4x em 3 anos	2x em 10 anos

# Memória Principal

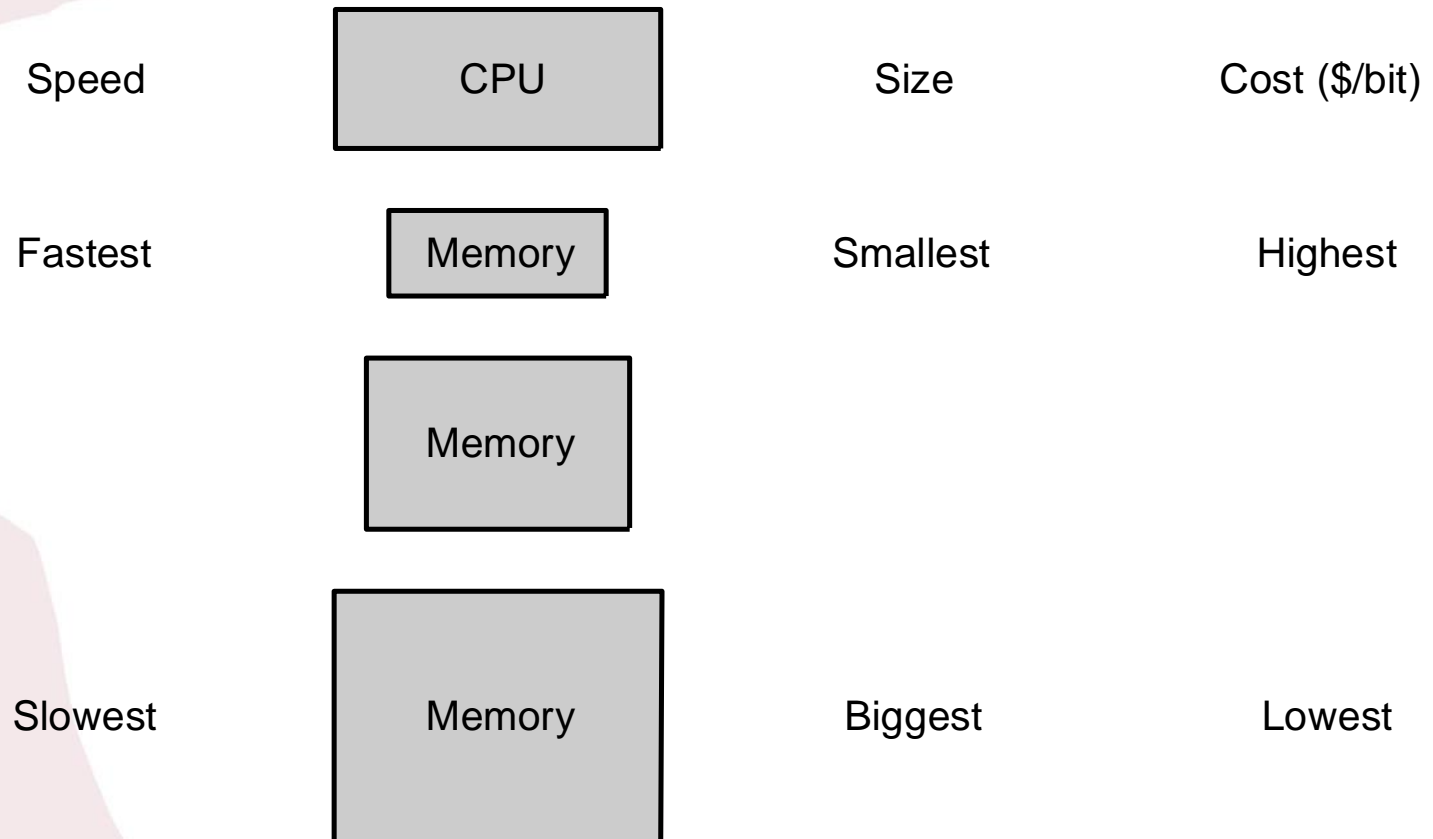
“640K ought to be enough for  
anybody.”

Bill Gates, 1981

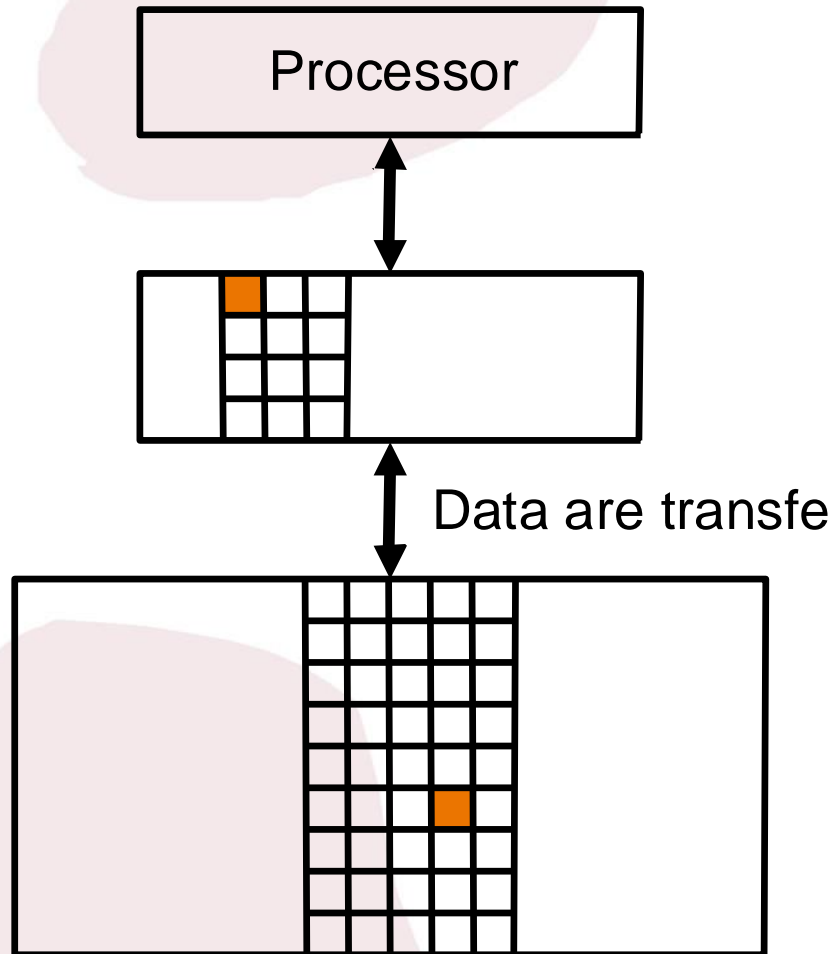
# Comparação da Performance da Mem. Principal e CPU



# Sistema Hierárquico de Memória

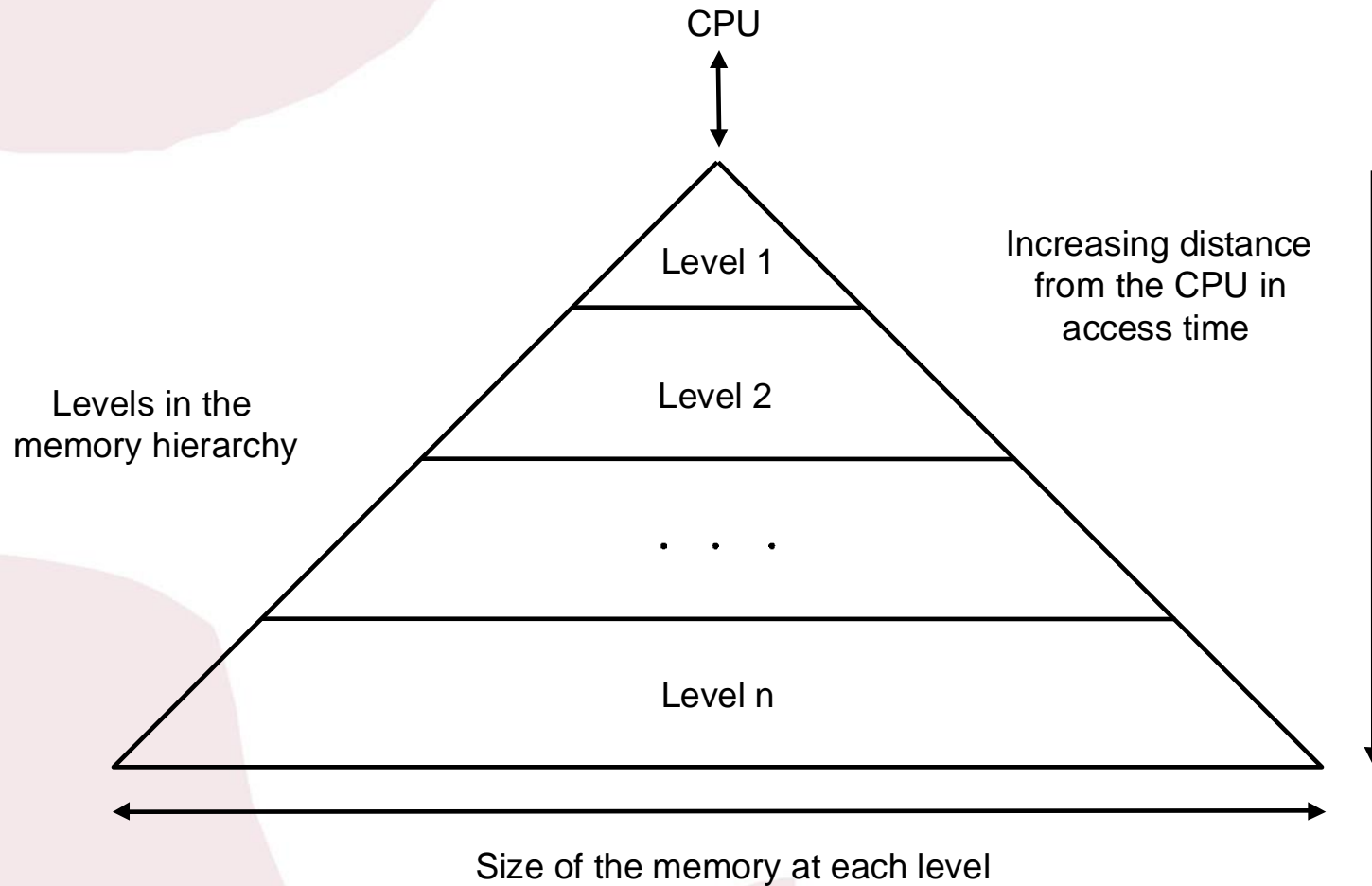


# Sistema Hierárquico de Memória



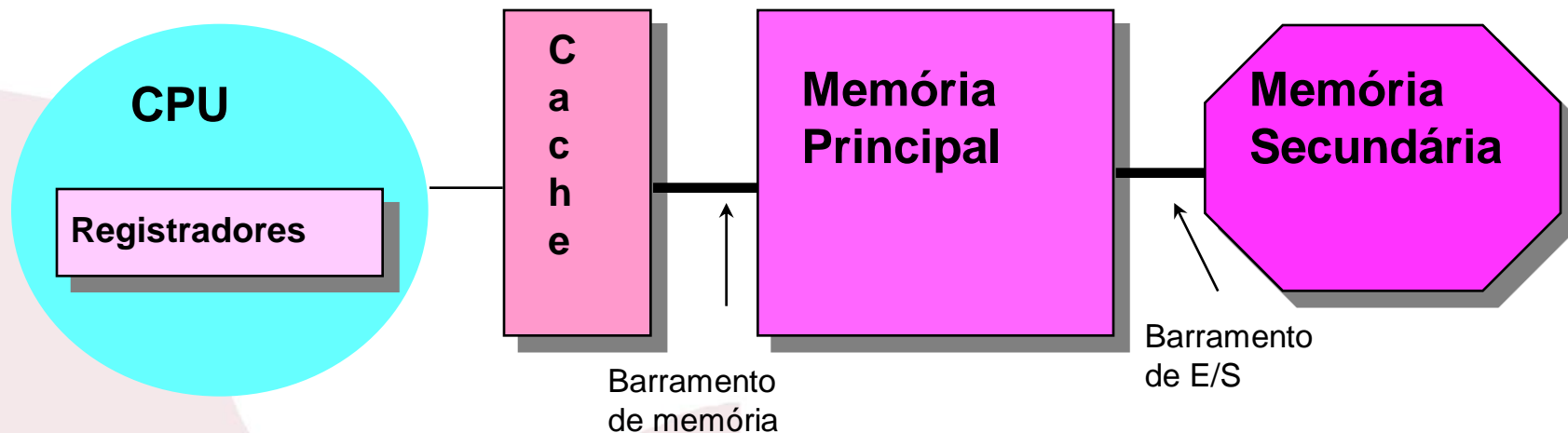
- Bloco (ou linha): unidade de cópia
  - Pode ser uma ou múltiplas palavras
- Se dado está presente no nível superior (mais perto da CPU)
  - Hit:
  - Hit ratio:  $\text{nr. hits} / \text{nr. acessos}$
- Se dado está ausente
  - Miss: bloco copiado do nível inferior
    - Tempo de acesso: miss penalty
    - Miss ratio:  $\text{misses} / \text{acessos} = 1 - \text{hit ratio}$
  - Então acessa dado do nível superior

# Sistema Hierárquico de Memória



# Motivação para hierarquia

- Princípio da localidade + Relação custo/desempenho das tecnologias
- Alto desempenho da CPU





# Princípio da Localidade



- **Localidade Temporal**

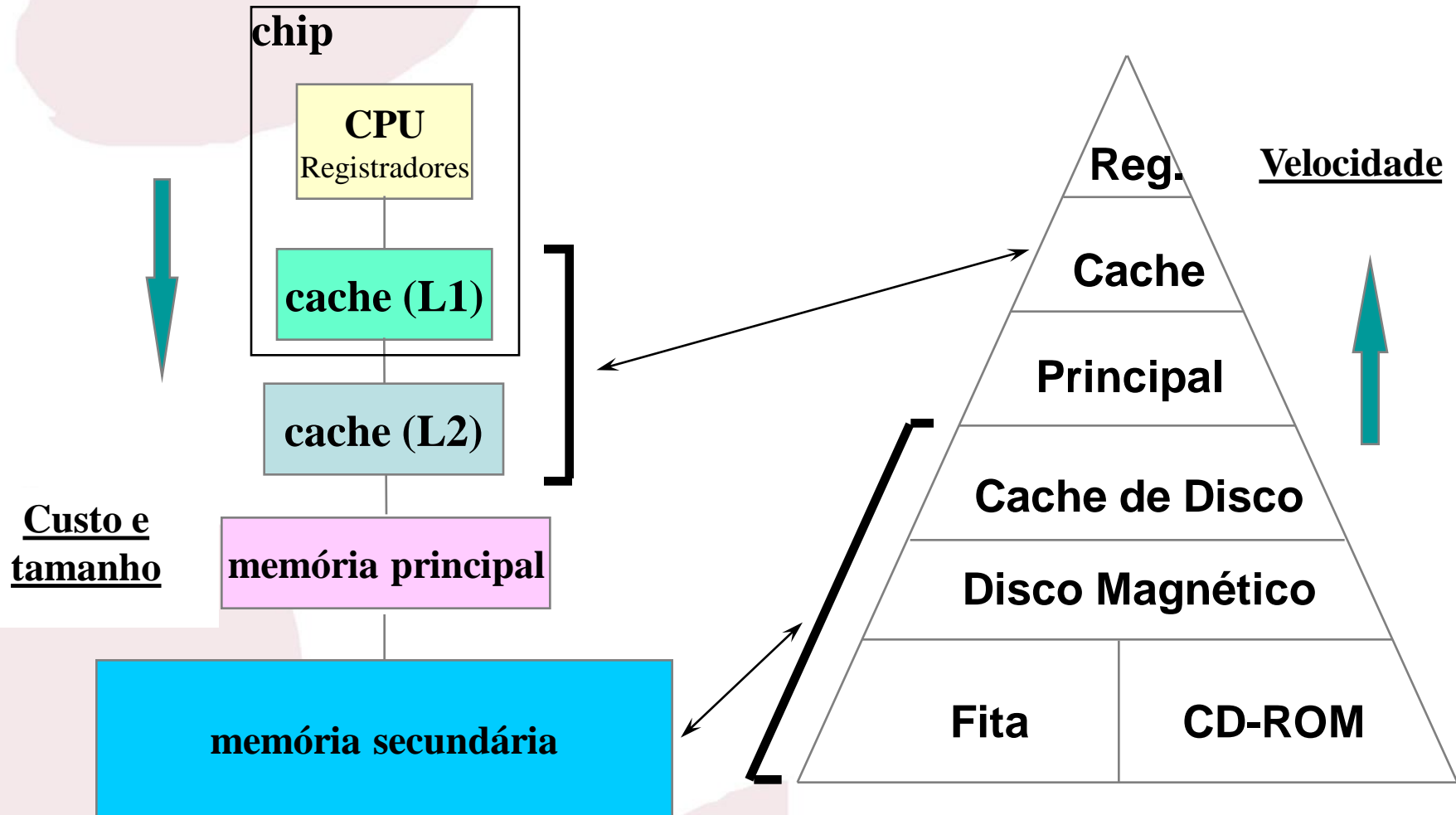
- Num futuro próximo, o programa irá referenciar as instruções e dados referenciados recentemente

- **Localidade Espacial**

- Num futuro próximo, o programa irá referenciar as instruções e dados que tenham endereços próximos das últimas referências.



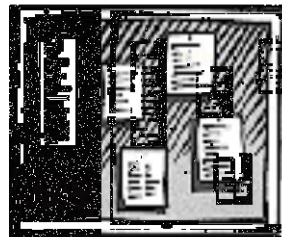
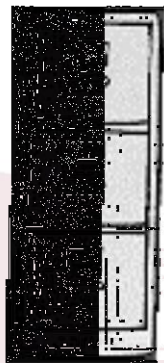
# Hierarquia de Memória



# Memória Cache

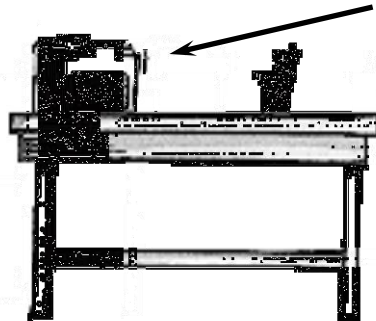
# Memória Principal Vs. Cache

Fichário



Quadro

Pasta



- O fichário representa o disco rígido.
- A pasta sobre a mesa representa a memória principal.
- No quadro de avisos se encontram informações que podem ser acessadas de forma muito rápida. O quadro representa a cache.
- Mesa e usuário são a CPU

# Memória Cache

- Memória Cache
  - Nível de memória mais próximo da CPU
- Dados os acessos  $X_1, \dots, X_{n-1}, X_n$

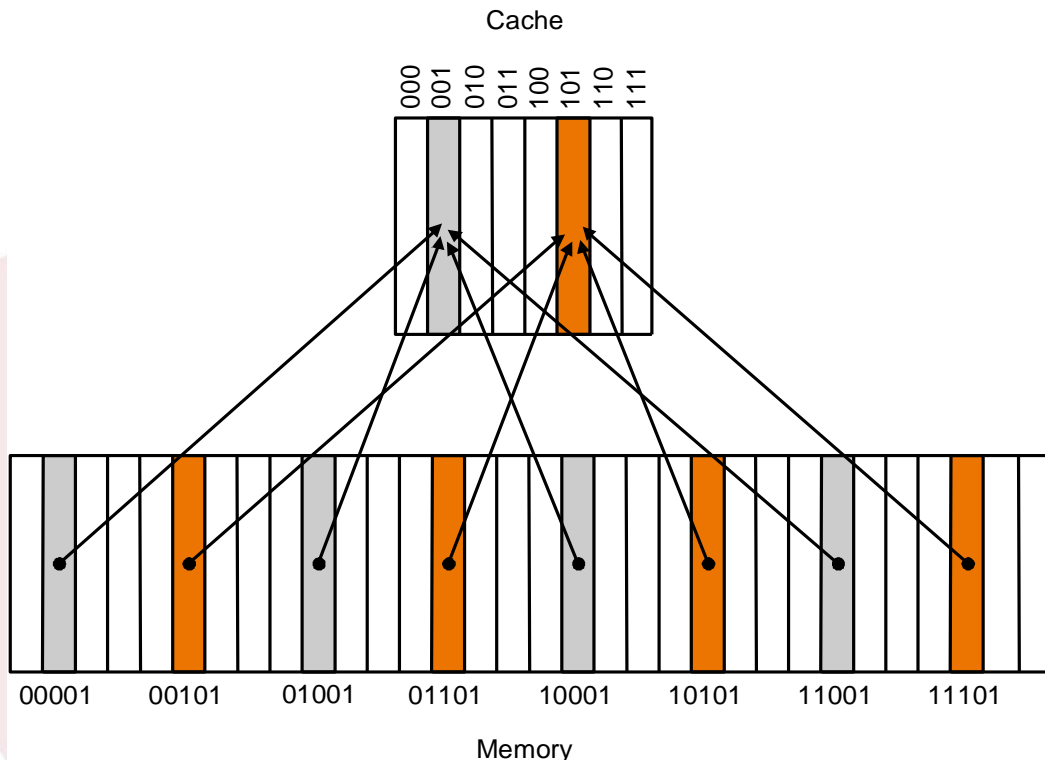
$X_4$
$X_1$
$X_{n-2}$
$X_{n-1}$
$X_2$
$X_3$

$X_4$
$X_1$
$X_{n-2}$
$X_{n-1}$
$X_2$
$X_n$
$X_3$

- Como saber se o dado está presente?
- Onde procurar o dado?

# Mapeamento direto

- Localização determinada pelo endereço
- Mapeamento direto: somente uma possibilidade
  - $(\text{Block address}) \bmod (\text{\#Blocks in cache})$



- #Blocks é uma potencia de 2
- Use bits menos significativos do endereço

# Bits: Tags Validade

- Como saber qual bloco está armazenado numa localização da cache?
  - Armazena parte do endereço (bits mais significativos) e o dado
  - Bits mais significativos: tag
- Como saber se o dado está armazenado?
  - Bit Validade: 1 = presente, 0 = não está present
  - Inicialmente: 0

# Endereçando a cache

- Composição do endereço
  - Tag
  - Índice
  - Endereço de byte
- Exemplo:
  - Memória: endereço de 32 bits, acesso por palavra(32 bits), endereçamento por byte
  - Cache: capacidade para armazenar 64 palavras





# Acessando a memória cache

<i><b>Endereço decimal</b></i>	<i><b>Endereço binário</b></i>	<i><b>Hit ou miss na cache</b></i>	<i><b>Bloco na cache</b></i>
22	10110	Miss	10110 mod 8 = 110
26	11010	Miss	11010 mod 8 = 010
22	10110	Hit	10110 mod 8 = 110
26	11010	Hit	11010 mod 8 = 010
16	10000	Miss	10000 mod 8 = 000
3	00011	Miss	00011 mod 8 = 011
16	10000	Hit	10000 mod 8 = 000
18	10010	miss	10010 mod 8 = 010

# Acessando a memória cache

Índice	V	Tag	Dado
000	N		
001	N		
010	N		
011	N		
100	N		
101	N		
110	N		
111	N		

Estado inicial  
da cache

Endereço decimal	Endereço binário	Hit ou miss na cache	Bloco na cache
22	10110	Miss	$10110 \bmod 8 = 110$
26	11010	Miss	$11010 \bmod 8 = 010$
22	10110	Hit	$10110 \bmod 8 = 110$
26	11010	Hit	$11010 \bmod 8 = 010$
16	10000	Miss	$10000 \bmod 8 = 000$
3	00011	Miss	$00011 \bmod 8 = 011$
16	10000	Hit	$10000 \bmod 8 = 000$
18	10010	miss	$10010 \bmod 8 = 010$

Acessos à  
memória

# Acessando a memória cache

Índice	V	Tag	Dado
000	N		
001	N		
010	N		
011	N		
100	N		
101	N		
110	Y	10	Memória(10110)
111	N		

Endereço 10110

<i>Endereço decimal</i>	<i>Endereço binário</i>	<i>Hit ou miss na cache</i>	<i>Bloco na cache</i>
22	10110	Miss	10110 mod 8 = 110
26	11010	Miss	11010 mod 8 = 010
22	10110	Hit	10110 mod 8 = 110
26	11010	Hit	11010 mod 8 = 010
16	10000	Miss	10000 mod 8 = 000
3	00011	Miss	00011 mod 8 = 011
16	10000	Hit	10000 mod 8 = 000
18	10010	miss	10010 mod 8 = 010

Acessos à  
memória

# Acessando a memória cache

Índice	V	Tag	Dado
000	N		
001	N		
010	Y	11	Memória(11010)
011	N		
100	N		
101	N		
110	Y	10	Memória(10110)
111	N		

Endereço 11010

<i>Endereço decimal</i>	<i>Endereço binário</i>	<i>Hit ou miss na cache</i>	<i>Bloco na cache</i>
22	10110	Miss	10110 mod 8 = 110
26	11010	Miss	11010 mod 8 = 010
22	10110	Hit	10110 mod 8 = 110
26	11010	Hit	11010 mod 8 = 010
16	10000	Miss	10000 mod 8 = 000
3	00011	Miss	00011 mod 8 = 011
16	10000	Hit	10000 mod 8 = 000
18	10010	miss	10010 mod 8 = 010

Acessos à  
memória

# Acessando a memória cache

Índice	V	Tag	Dado
000	Y	10	Memória(10000)
001	N		
010	Y	11	Memória(11010)
011	N		
100	N		
101	N		
110	Y	10	Memória(10110)
111	N		

Endereço 10000

<i>Endereço decimal</i>	<i>Endereço binário</i>	<i>Hit ou miss na cache</i>	<i>Bloco na cache</i>
22	10110	Miss	$10110 \bmod 8 = 110$
26	11010	Miss	$11010 \bmod 8 = 010$
22	10110	Hit	$10110 \bmod 8 = 110$
26	11010	Hit	$11010 \bmod 8 = 010$
16	10000	Miss	$10000 \bmod 8 = 000$
3	00011	Miss	$00011 \bmod 8 = 011$
16	10000	Hit	$10000 \bmod 8 = 000$
18	10010	miss	$10010 \bmod 8 = 010$

Acessos à  
memória

# Acessando a memória cache

Índice	V	Tag	Dado
000	Y	10	Memória(10000)
001	N		
010	Y	11	Memória(11010)
011	Y	00	Memória(00011)
100	N		
101	N		
110	Y	10	Memória(10110)
111	N		

Endereço 00011

<i>Endereço decimal</i>	<i>Endereço binário</i>	<i>Hit ou miss na cache</i>	<i>Bloco na cache</i>
22	10110	Miss	$10110 \bmod 8 = 110$
26	11010	Miss	$11010 \bmod 8 = 010$
22	10110	Hit	$10110 \bmod 8 = 110$
26	11010	Hit	$11010 \bmod 8 = 010$
16	10000	Miss	$10000 \bmod 8 = 000$
3	00011	Miss	$00011 \bmod 8 = 011$
16	10000	Hit	$10000 \bmod 8 = 000$
18	10010	miss	$10010 \bmod 8 = 010$

Acessos à  
memória

# Acessando a memória cache

Índice	V	Tag	Dado
000	Y	10	Memória(10000)
001	N		
010	Y	10	Memória(10010)
011	Y	00	Memória(00011)
100	N		
101	N		
110	Y	10	Memória(10110)
111	N		

Endereço 10010

<i>Endereço decimal</i>	<i>Endereço binário</i>	<i>Hit ou miss na cache</i>	<i>Bloco na cache</i>
22	10110	Miss	$10110 \bmod 8 = 110$
26	11010	Miss	$11010 \bmod 8 = 010$
22	10110	Hit	$10110 \bmod 8 = 110$
26	11010	Hit	$11010 \bmod 8 = 010$
16	10000	Miss	$10000 \bmod 8 = 000$
3	00011	Miss	$00011 \bmod 8 = 011$
16	10000	Hit	$10000 \bmod 8 = 000$
18	10010	miss	$10010 \bmod 8 = 010$

Acessos à  
memória

# O que acontece numa falta de cache?

- Informação deve ser lida da memória
- São inseridos ciclos de espera no pipeline até que a informação esteja disponível na cache
  - Penalidade
- Se o endereço de cache está ocupado, a informação é sobre-escrita



# Leitura/Escrita da Cache

---



- Leitura:
  - Mais frequentes, rápidas e fáceis de implementar
- Escrita:
  - Mais lentas e complicadas e consistência de dados com a memória principal deve ser mantida (se um bloco da cache foi alterado pela CPU, não pode ser descartado da cache sem garantir que foi copiado para a mem. principal)



# Tipos de acesso à cache

- Leitura
- Escrita
  - Dado e tag são atualizados na cache
  - Inconsistencia entre memória principal e cache!!
  - Como resolver?

# Políticas de Escrita e Consistência

---



- Caches do tipo Write through
  - Cache e memória são atualizadas simultaneamente
- Caches do tipo Write back
  - Memória principal é atualizada quando bloco é substituído
  - Usa dirty bit para marcar linhas alteradas na cache.

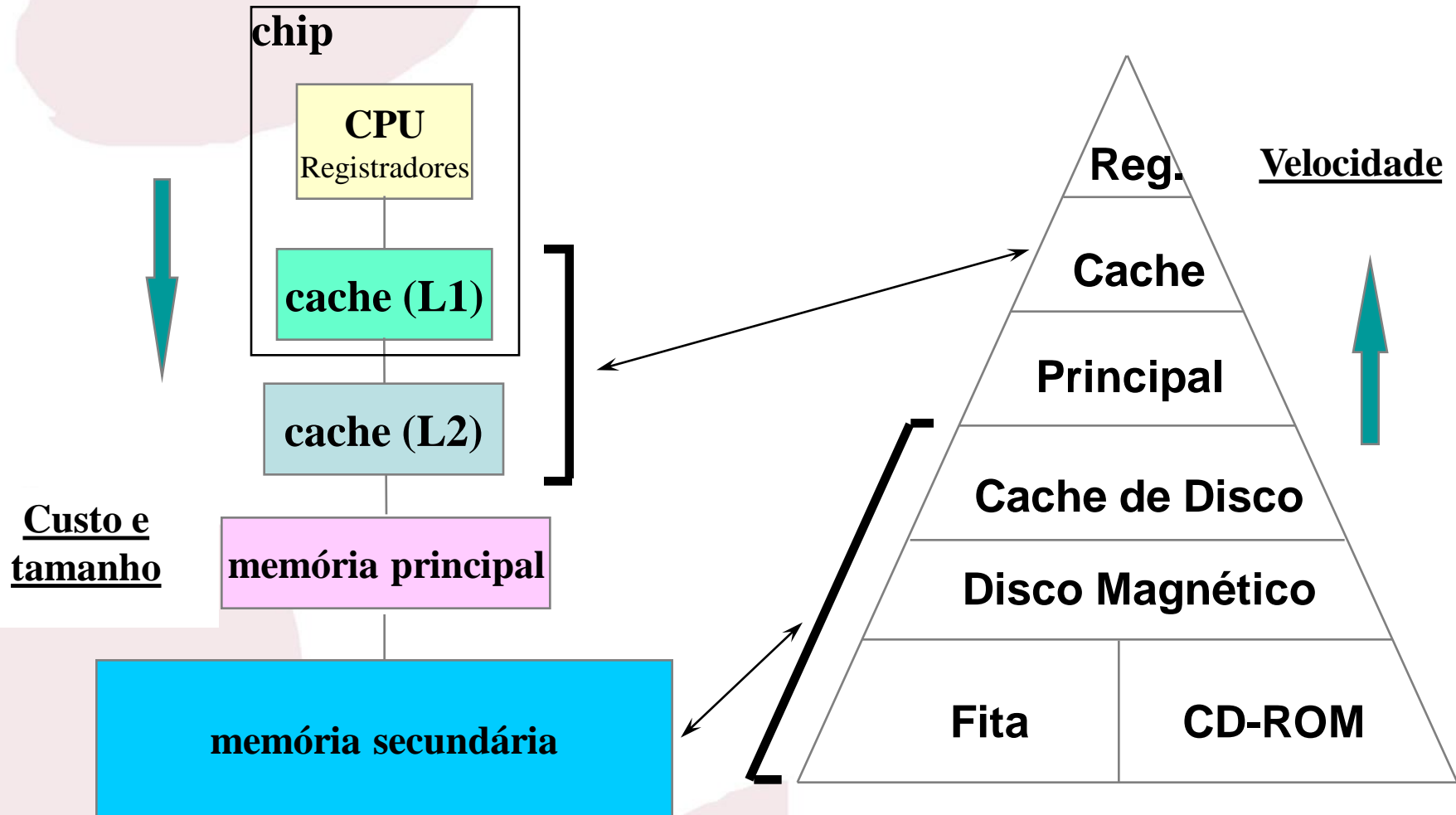


# Memória Cache: escrita

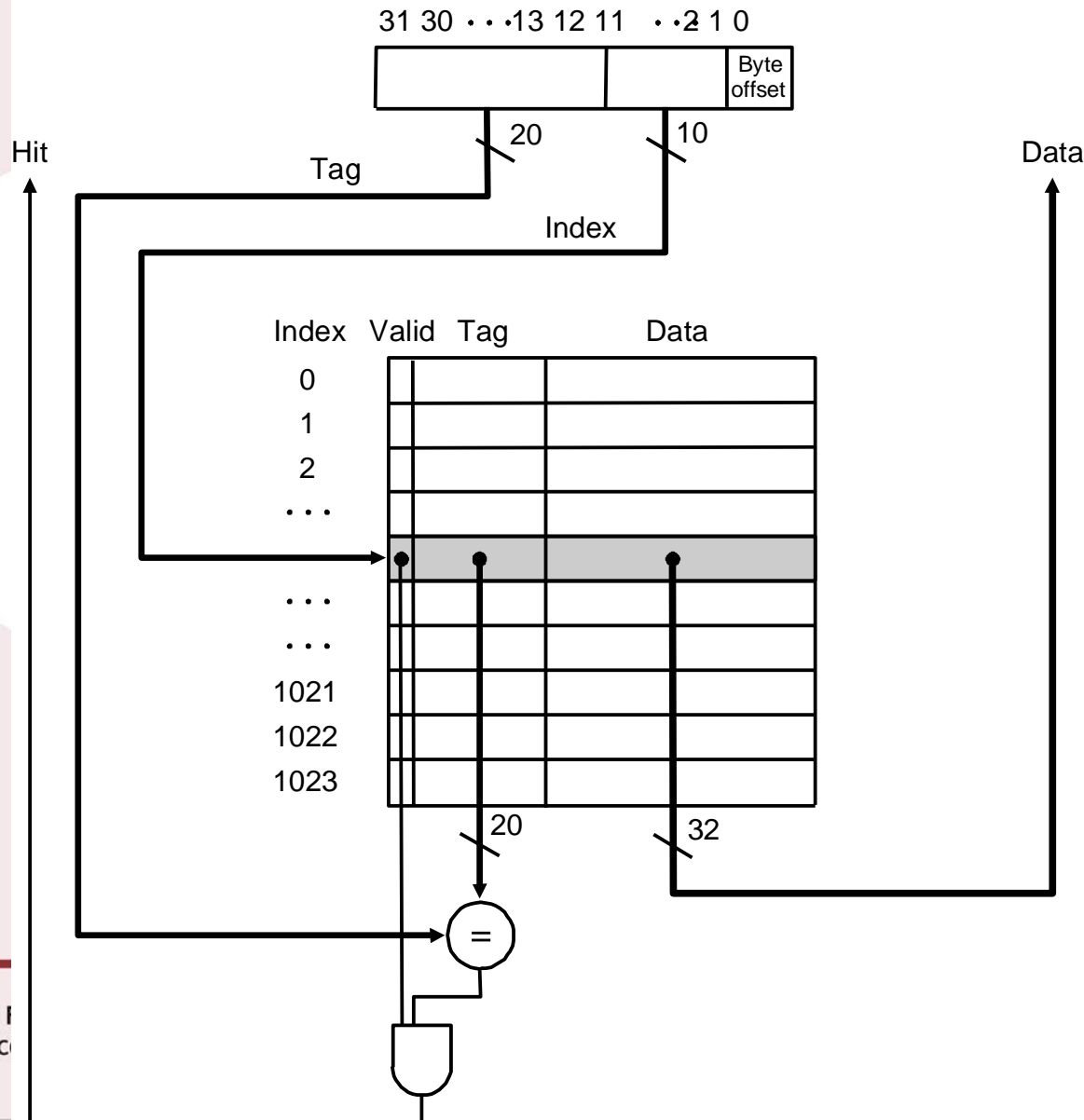
<b><i>Write through</i></b>	<b><i>Write back</i></b>
<b><i>facilidade de implementação</i></b>	<b><i>redução de acessos à memória</i></b>
<b><i>consistência da memória principal</i></b>	

- Para se evitar espera durante escrita:
  - Write buffers

# Hierarquia de Memória



# Exemplo: DECStation3100



# Tamanho da cache

- Quantos bits tem uma cache de mapeamento direto com 64K bytes de dados e blocos de uma palavra? Assuma endereços de 32 bits.

# Usando a localidade espacial



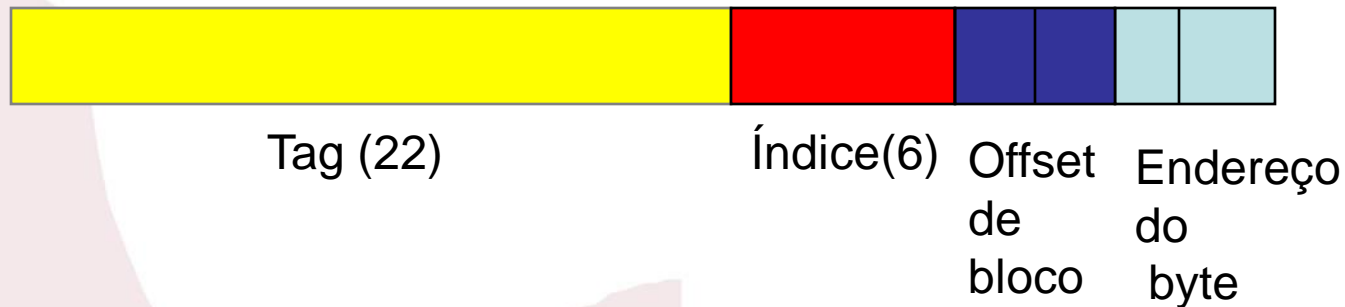
- Acessando a cache por blocos de palavras
- Composição do endereço:
  - Tag
  - Índice
  - Offset de bloco
  - Endereço de byte



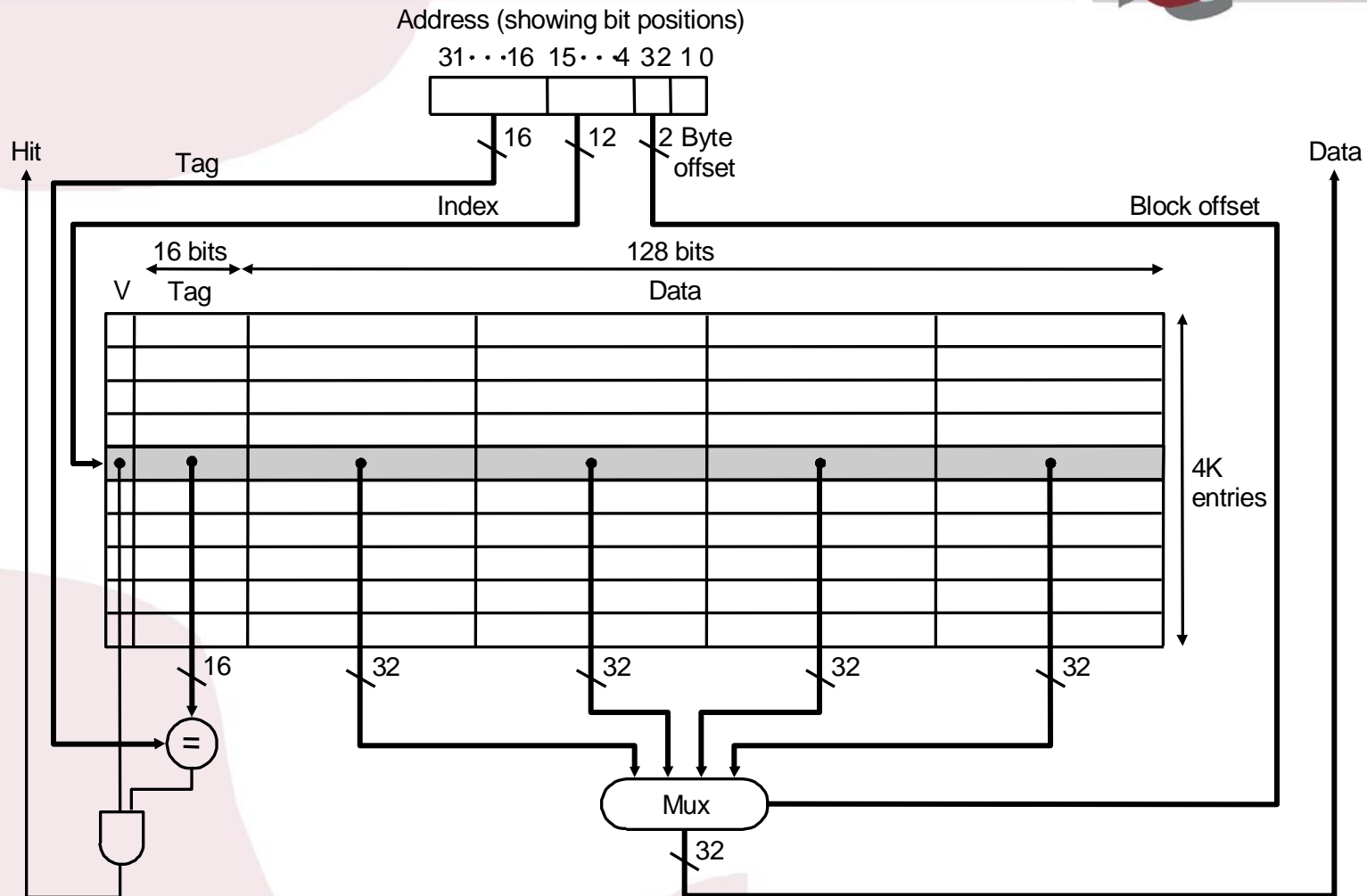


# Usando a localidade espacial

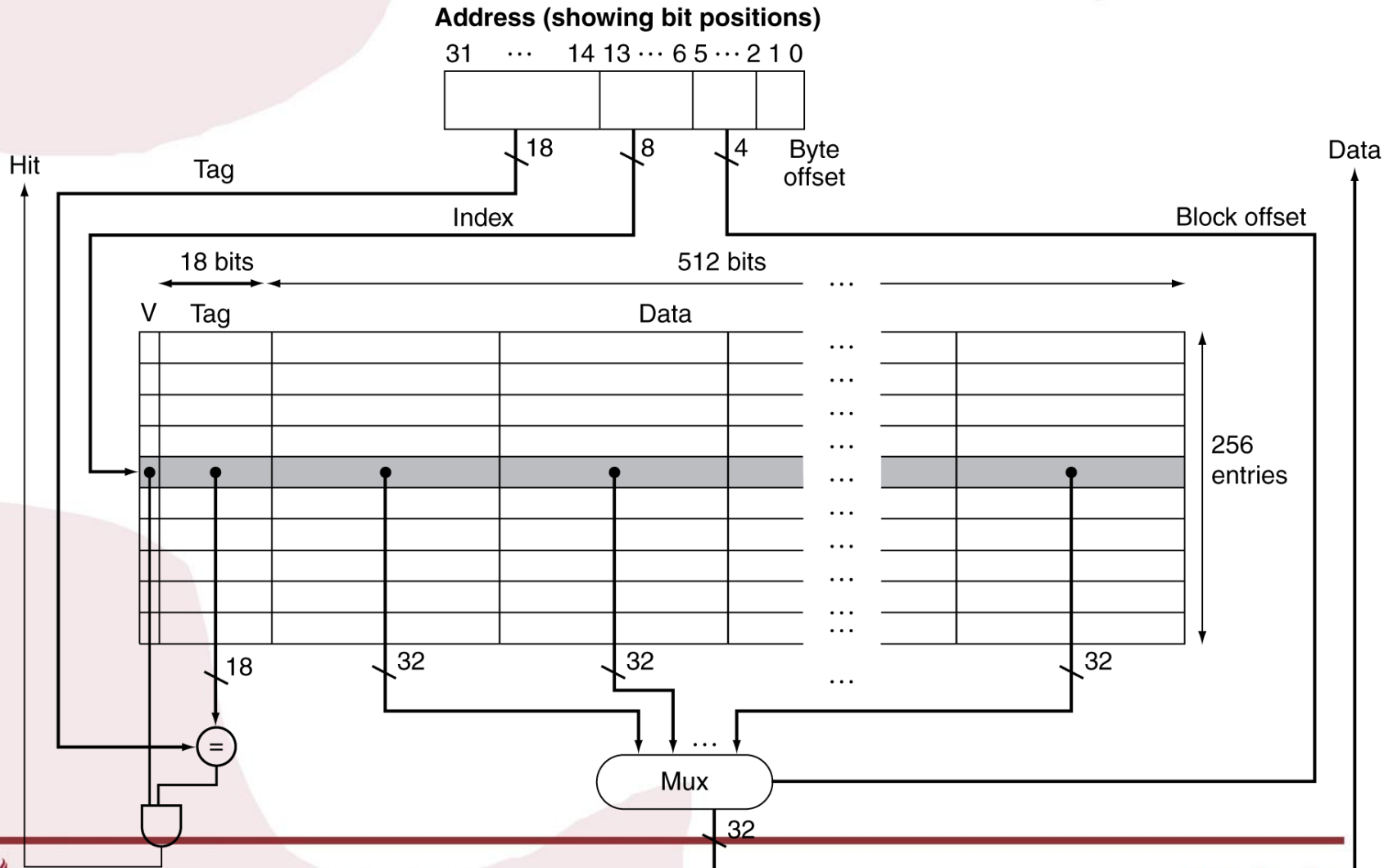
- Exemplo:
  - Memória: endereço de 32 bits, acesso por palavra(32 bits), endereçamento por byte
  - Cache: capacidade para armazenar 64 blocos de 4 palavras cada



# Mapeamento Direto-multiword



# Exemplo: Intrinsity FastMATH



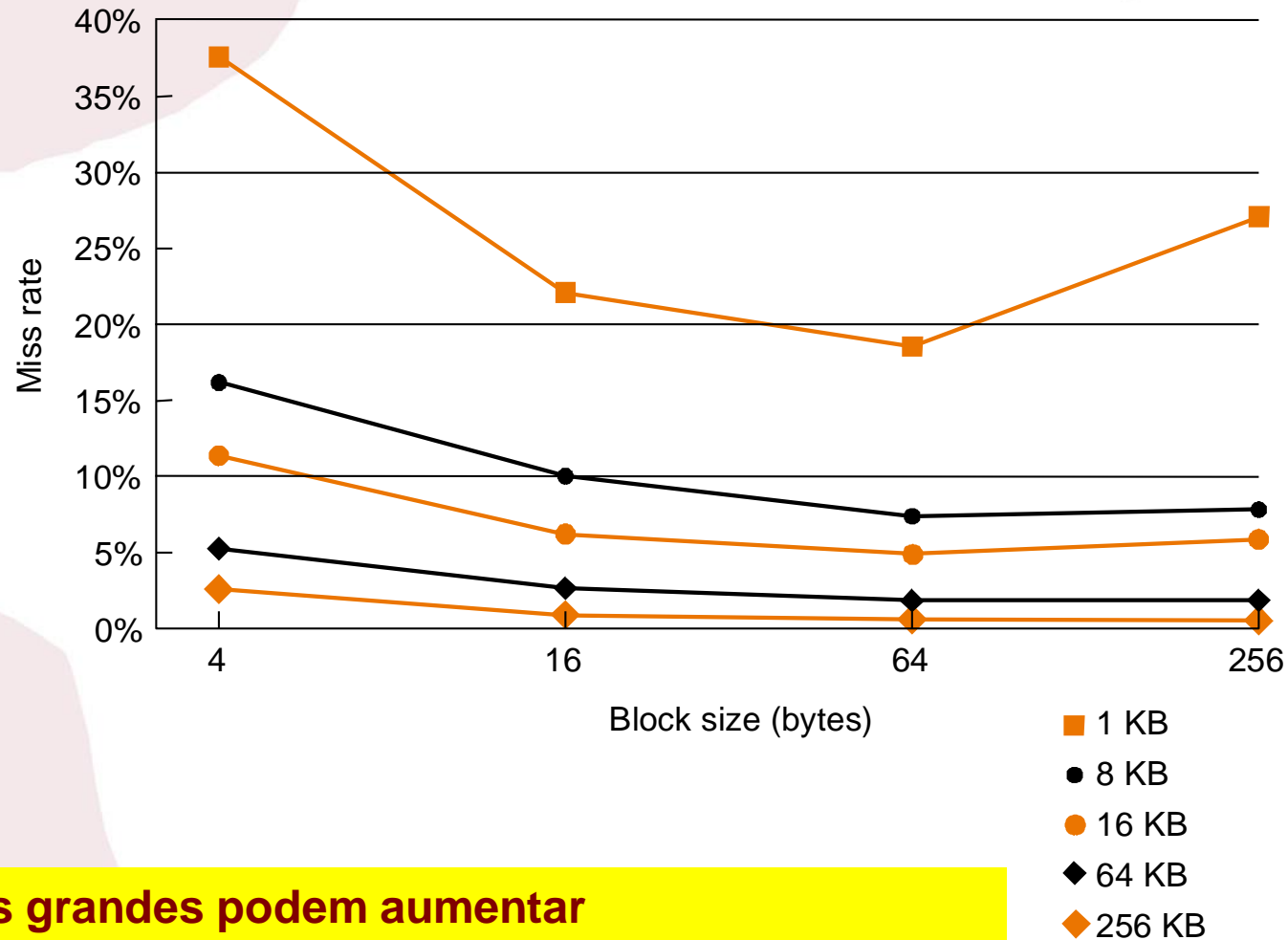
# Usando a localidade espacial



- O que acontece durante uma falta em acesso de leitura ou escrita?
  - Todo o bloco tem que ser carregado na cache
  - A escrita da palavra acontece
  - Cache write-through:
    - Todo o bloco é atualizado na memória



# Tamanho do bloco e taxa de faltas



**Blocos grandes podem aumentar  
a taxa de faltas se o tamanho da cache  
Permanece constante**

# Reduzindo a taxa de faltas



- Estratégias para posicionamento dos blocos:
  - Mapeamento direto: cada bloco possui posição única na cache
  - Associativa por conjunto: cada bloco pode ser colocado em algumas posições na cache
  - Completamente Associativa: cada bloco pode ser colocado em qualquer posição da cache

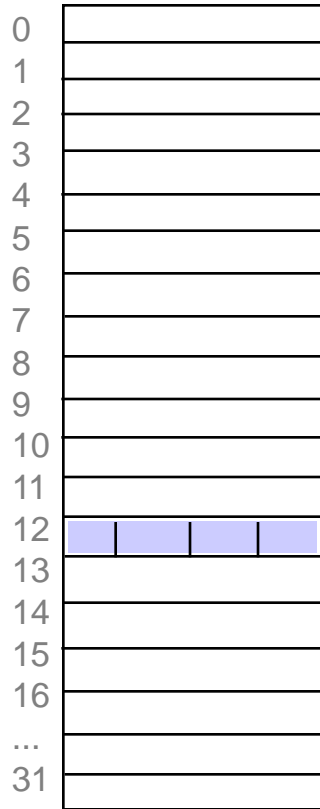


# Mapeamento Associativo por Conjunto



Um bloco na memória principal pode ocupar qualquer posição dentro de um conjunto definido de blocos da cache

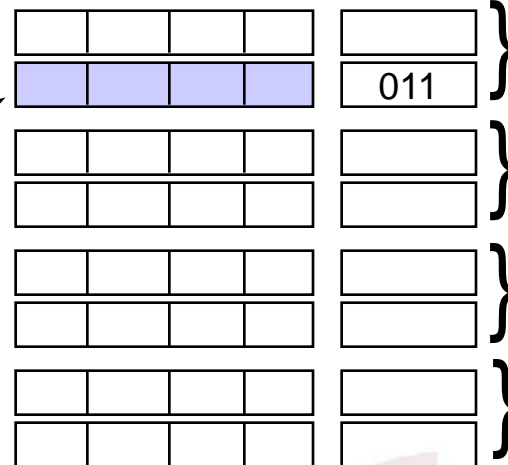
Memória principal



Memória cache

dados

tag



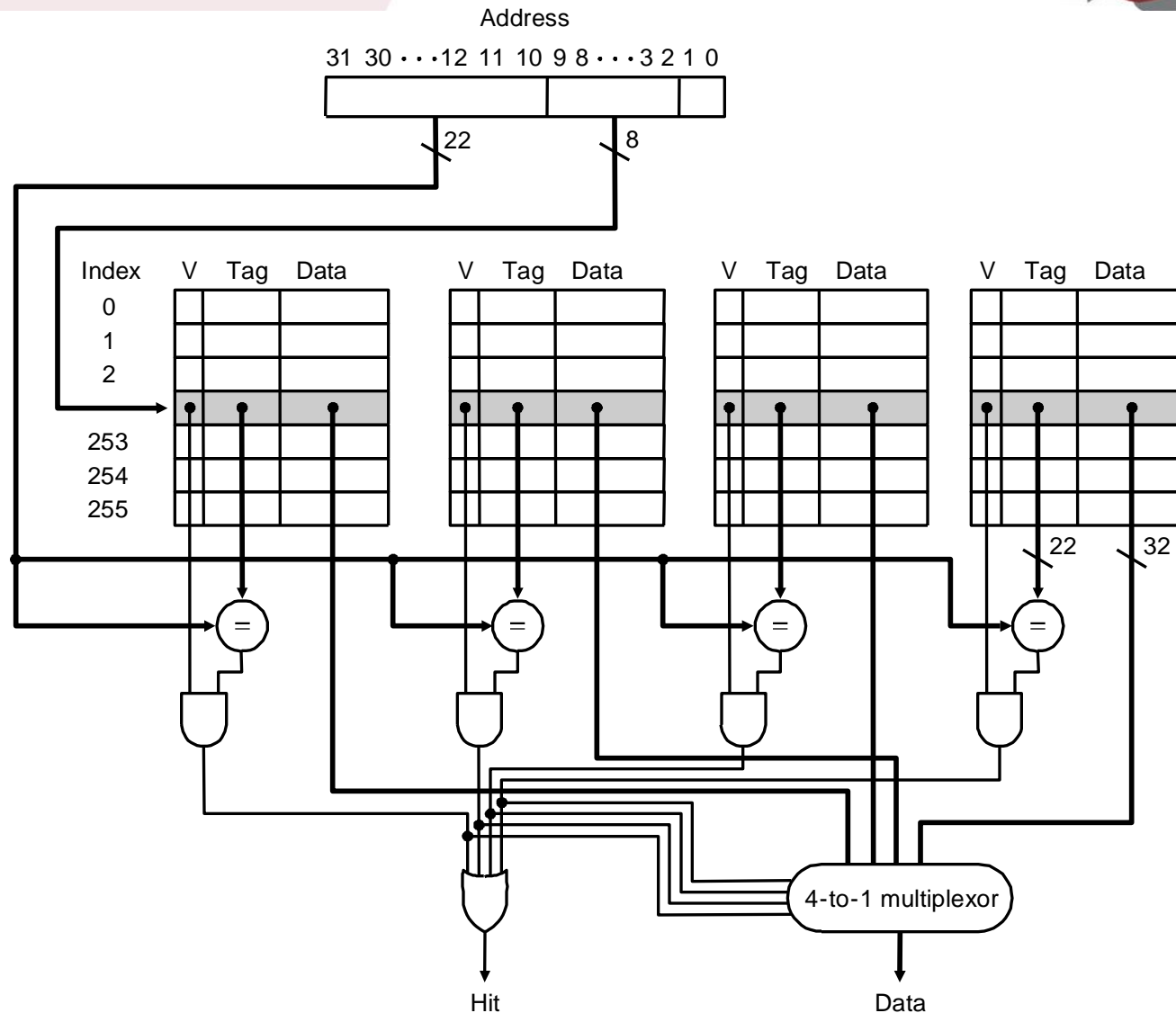
Conjuntos (sets)

Endereço da palavra

Tag	set	Offset
011	00	Offset



# Mapeamento Associativo por Conjunto





# Mapeamento Associativo

Memória principal

0	
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	
16	
...	
31	

Um bloco na memória principal pode ocupar qualquer posição na cache

- **Tag**  $\Rightarrow$  armazena na cache o end. Do bloco na mem.

**principal**

Memória cache

dados

tag

				01100

Offset

**Exemplo: tag = 12 (01100<sub>2</sub>)**

Endereço da palavra	
Tag	Offset

000000...01100	Offset
----------------	--------

# Grau de associatividade

(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

# Comparação de Métodos de Mapeamento



- Mapeamento direto

- **Simples e Barata**

- **Lenta**

- **Mais faltas**

- Associativa

- **Rápida**

- **Menos falta**

- **Cara (comparação do endereço em paralelo)**

- Associativa por conjunto: combinação das anteriores

- Se  $NCC = NBC \Rightarrow$  Ass. por conjunto = Mapeamento Direto

- Se  $NCC = 1 \Rightarrow$  Ass. por conjunto = Associativa

**NBC = núm. blocos da cache**

**NCC = núm. conjuntos da cache**



# Políticas de Substituição de Blocos



- Randômica:
  - Simples e fácil de implementar
- FIFO (First-In-First-Out)
- LFU (Least-Frequently Used)
- LRU (least-recently used)
  - Menor taxa de faltas

Associatividade						
2 way			4-way		8-way	
Size	LRU	random	LRU	random	LRU	random
16 KB	5.18	5.69	4.67	5.29	4.39	4.96
64 KB	1.88	2.01	1.54	1.66	1.39	1.53
256KB	1.15	1.17	1.13	1.13	1.12	1.12



# Caches separadas

- Cache de dados e cache de instruções
  - Vantagens:
    - Melhor capacidade de otimizações
    - Evita hazard estrutural
  - Desvantagens:
    - maior taxa de falta

Programa	Miss rate (instr.)	Miss rate (dado)	Miss rate (sep.)	Miss rate (única)
Gcc	6.1 %	2.1 %	5.4 %	4.8 %
Spice	1.2 %	1.3 %	1.2 %	

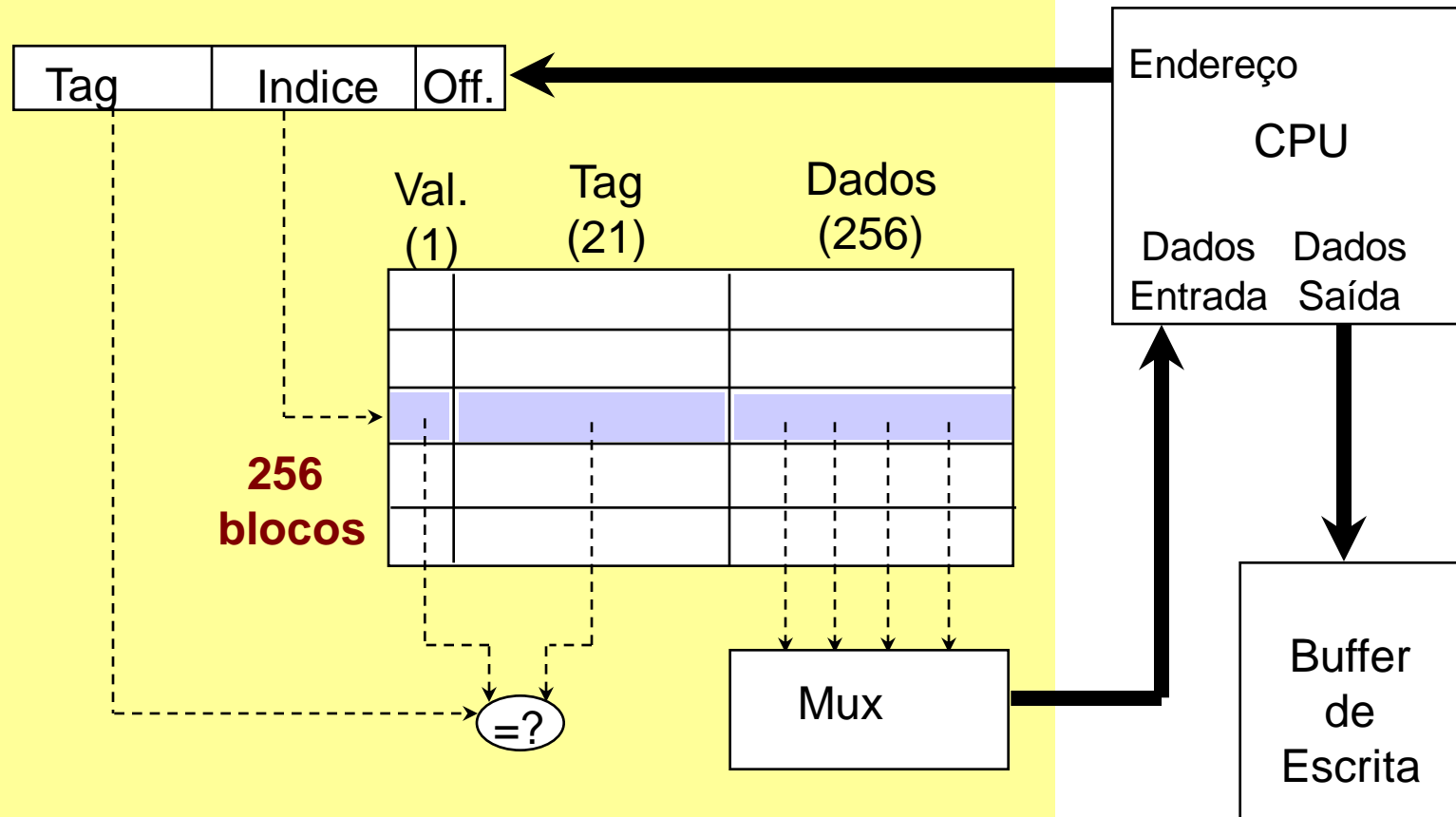
# Exemplo: Alpha AXP 21064



- Cache separadas de dados e de instruções
- Características:
  - Tamanho: 8192 bytes
  - Blocos de 32 bits
  - Mapeamento direto
  - Write through
  - Four buffer write-buffer



# Alpha AXP 21064- Cache Dados



# Exercício



Considere referências aos seguintes endereços de memória: 1,4,8,5,20,17,19,56, 9,11, 4,43,5,6,9, 17. Calcule o número de faltas para uma cache de 16 palavras com blocos de 1 palavra e mostre o estado final da cache. Compare os resultados para as seguintes organizações:

- (a) - mapeamento direto
- (b) - two-way set associativa,
- (c) - completamente associativa.

Suponha que a cache está inicialmente vazia e quando necessário use como política de substituição o algoritmo LRU.

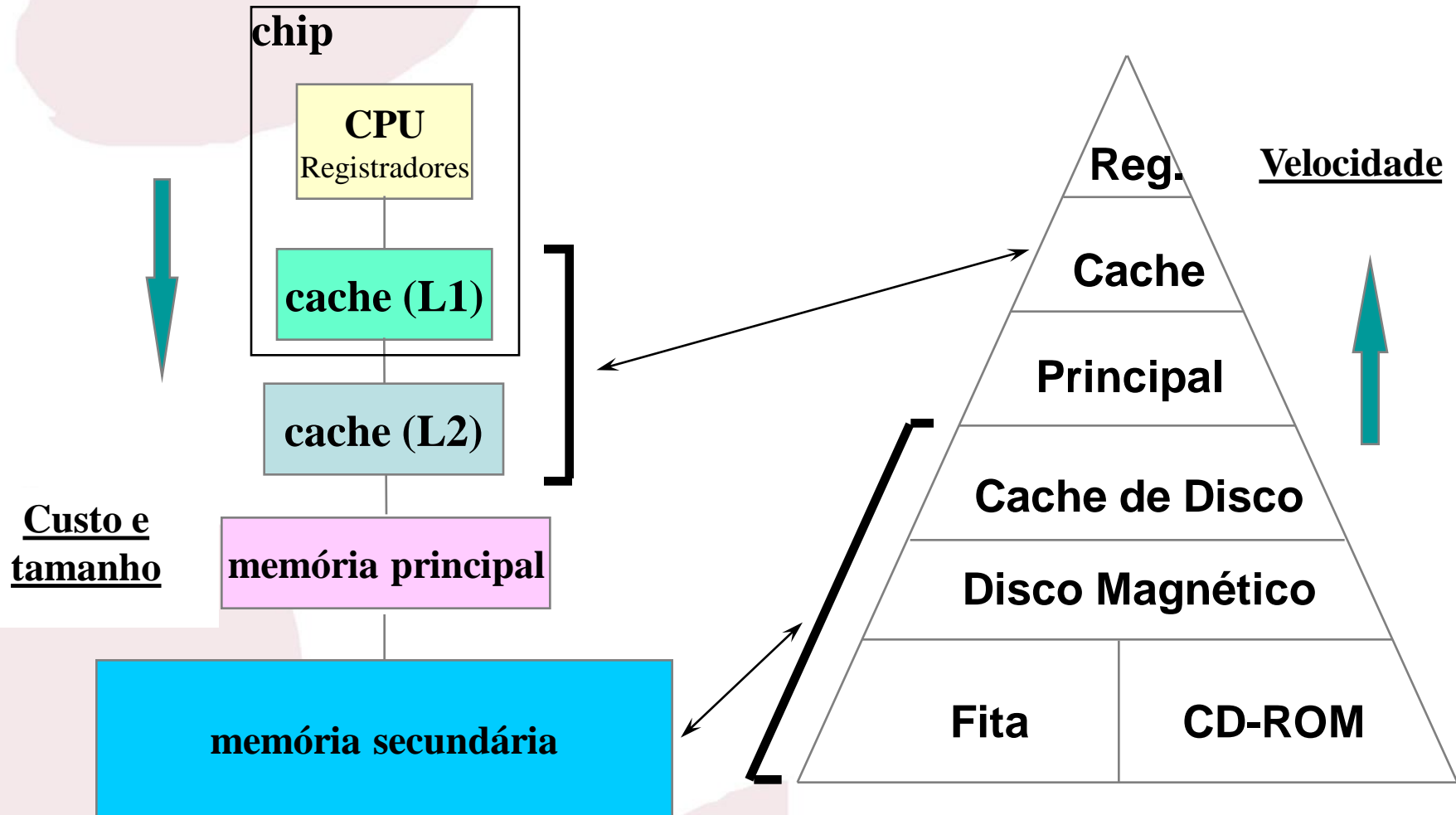




# Hierarquia de memória

## Melhorando o desempenho

# Hierarquia de Memória



# Desempenho de uma CPU

- $\text{CPU}_{\text{time}} = (\text{CPU\_Ciclos para execução} + \text{Memória\_ciclos-stall}) \times \text{CLK\_período}$
- $\text{CPU\_ciclos execução} = \# \text{instruções} \times \text{CPI}$
- $\text{Memória}_{\text{ciclos}} = \text{Leitura}_{\text{ciclos}} + \text{Escrita}_{\text{ciclos}}$
- $\text{Leitura}_{\text{ciclos}} = \text{Leituras} \times \text{Miss\_rate}_{\text{leitura}} \times \text{Penalty}_{\text{leitura}}$
- $\text{Escrita}_{\text{ciclos}} = \text{Escritas} \times \text{Miss\_rate}_{\text{escrita}} \times \text{Penalty}_{\text{escrita}}$

# Desempenho de uma CPU



- Exemplo:
  - $\text{Miss\_rate}_{\text{instr.}} = 2\%$ ,  $\text{Miss\_rate}_{\text{dado.}} = 4\%$ ,  $\text{CPI} = 2$ ,  $\text{Penalty} = 100$  ciclos
  - $\text{Taxa}(\text{Load}, \text{Store}) = 36\%$
  - Qual a degradação de desempenho devido aos acessos à memória?  $\text{CPI}_{\text{mem}}$ ?



# Desempenho de uma CPU

- $\text{Miss\_instr}_{\text{ciclos}} = I \times 2\% \times 100 = 2,00 \text{ I}$
- $\text{Miss\_dados}_{\text{ciclos}} = I \times 36\% \times 4\% \times 100 = 1,44 \text{ I}$
- $\text{CPU}_{\text{time}} = (\text{CPU}_{\text{ciclos-execução}} + \text{Memória}_{\text{ciclos-stall}}) \times \text{Clk}$
- $\text{Memória}_{\text{ciclos}} = 2 \text{ I} + 1,44 \text{ I} = 3,44 \text{ I}$
- $\text{CPI}_{\text{mem}} = 2,0 + 3,44 = 5,44$
- $\text{CPI}_{\text{stall}} / \text{CPI}_{\text{perf}} = 5,44/2 = 2,72$

# Processador mais rápido...



- Diminuindo CPI
  - $CPI_{\text{novo}} = 1,0$
  - $CPI_{\text{mem}} = 1,0 + 3,44 = 4,44$
  - Desempenho =  $4,44 / 1,0 = 4,44$ 
    - Quantidade de tempo com stalls sobre de 63% ( $3,44/5,44$ ) para 77% ( $3,44 / 4,44$ )
- Duplicando o clock
  - Penalty = 200 ciclos
  - $Miss_{\text{ciclos}}(\text{por instr.}) = 2\% \times 200 + 36\% (4\% \times 200) = 6,88$
  - $CPI_{\text{mem}} = 2,0 + 6,88 = 8,88$
  - $CI \times CPI_{\text{ck-lento}} \times \text{ciclo} / CI \times CPI_{\text{ck-rap.}} \times \text{ciclo} = 5,44 / 8,88 \times 0,5 = 1,23$



# Desempenho de uma cache

- CPI( #clocks por instrução)
  - Cache Perfeita  $\Rightarrow 2,0$
  - Cache (2% miss<sub>instr</sub>, 4% miss<sub>dado</sub>)  $\Rightarrow 5,44$
  - Sem cache  $\Rightarrow 68.5$
- Melhorando o processador
  - Diminuindo CPI
    - CPI  $\Rightarrow 4,44$  (em vez de 1,0)
  - Duplicando clock
    - CPI  $\Rightarrow 8,88$

# Melhorando desempenho da cache



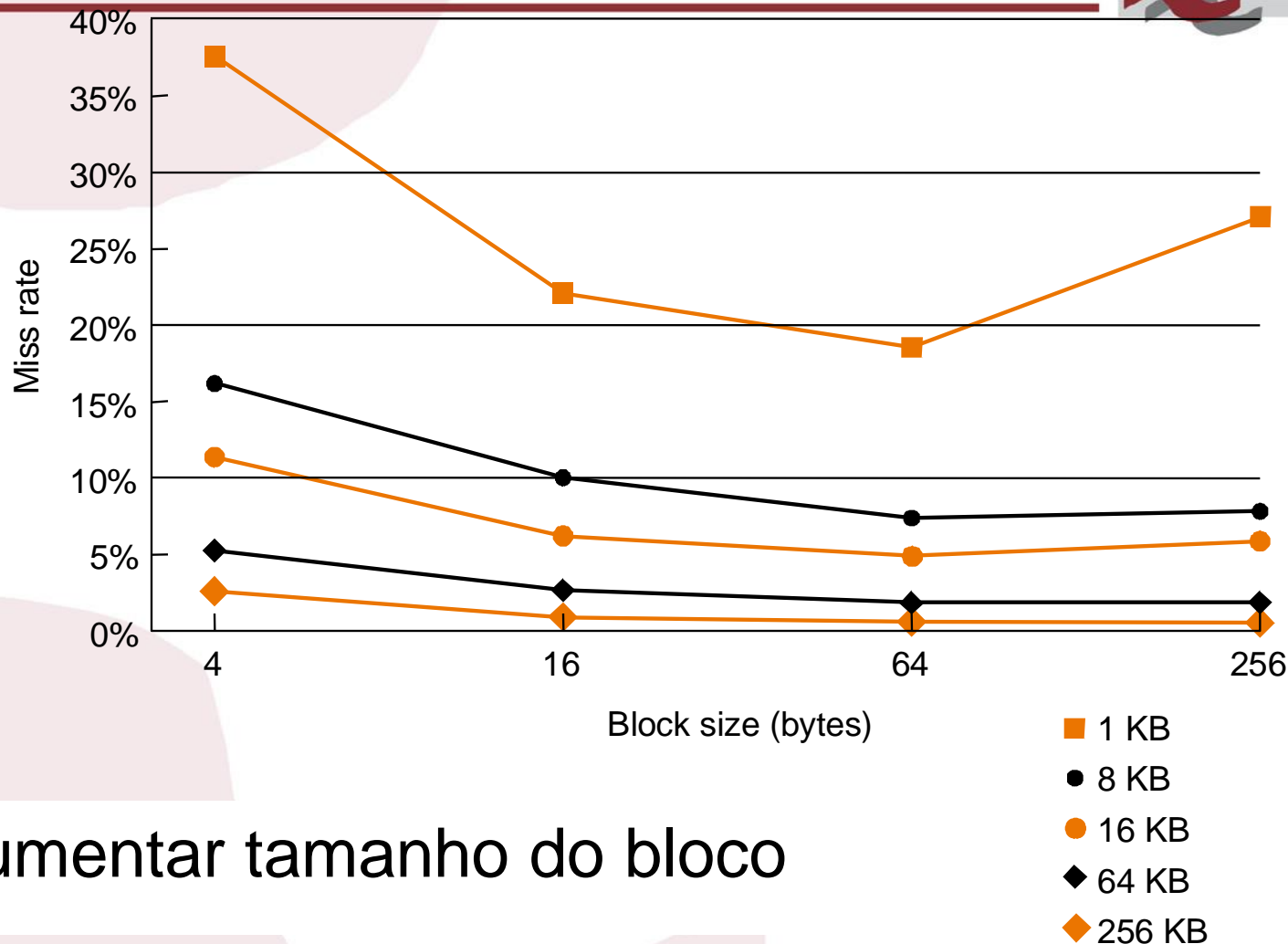
$$\text{Tempo\_acesso}_{\text{m\u00e9dio}} \equiv \text{Hit time} + \text{Miss rate} \times \text{Miss penalty}$$

- Estrat\u00e9gias:
  - Redu\u00e7\u00e3o de faltas
  - Redu\u00e7\u00e3o da penalidade
  - Redu\u00e7\u00e3o do tempo de acesso





# Reduzindo falta de cache



- Aumentar tamanho do bloco

# Reduzindo falta de cache



- Aumento da associatividade

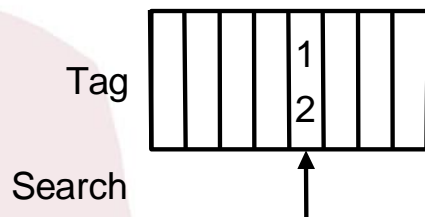
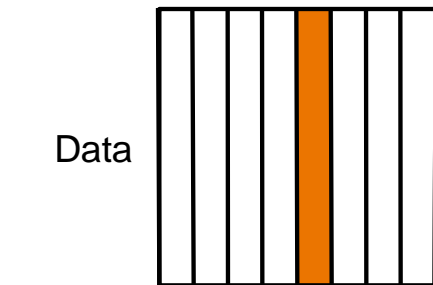
Tam. cache	one-way	two-way	four-way	eight-way
1	7.65	6.60	6.22	5.44
2	5.90	4.90	4.62	4.09
4	4.60	3.95	3.57	3.19
8	3.30	3.00	2.87	2.59
16	2.45	2.20	2.12	2.04
32	2.00	1.80	1.77	1.79
64	1.70	1.60	1.57	1.59
128	1.50	1.45	1.42	1.44



# Aumento da Associatividade

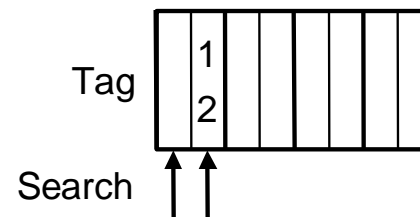
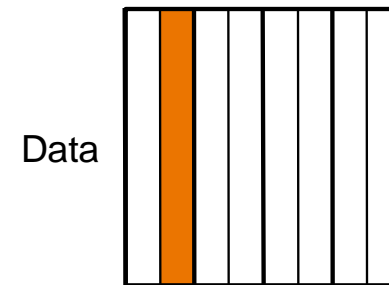
Direct mapped

Block # 0 1 2 3 4 5 6 7

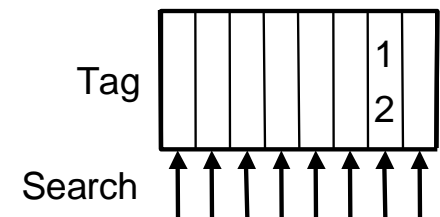
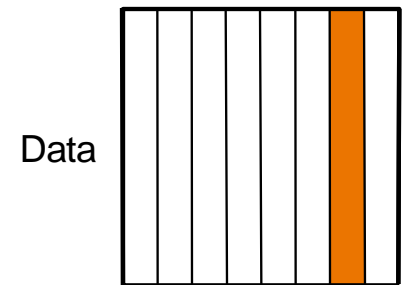


Set associative

Set # 0 1 2 3



Fully associative



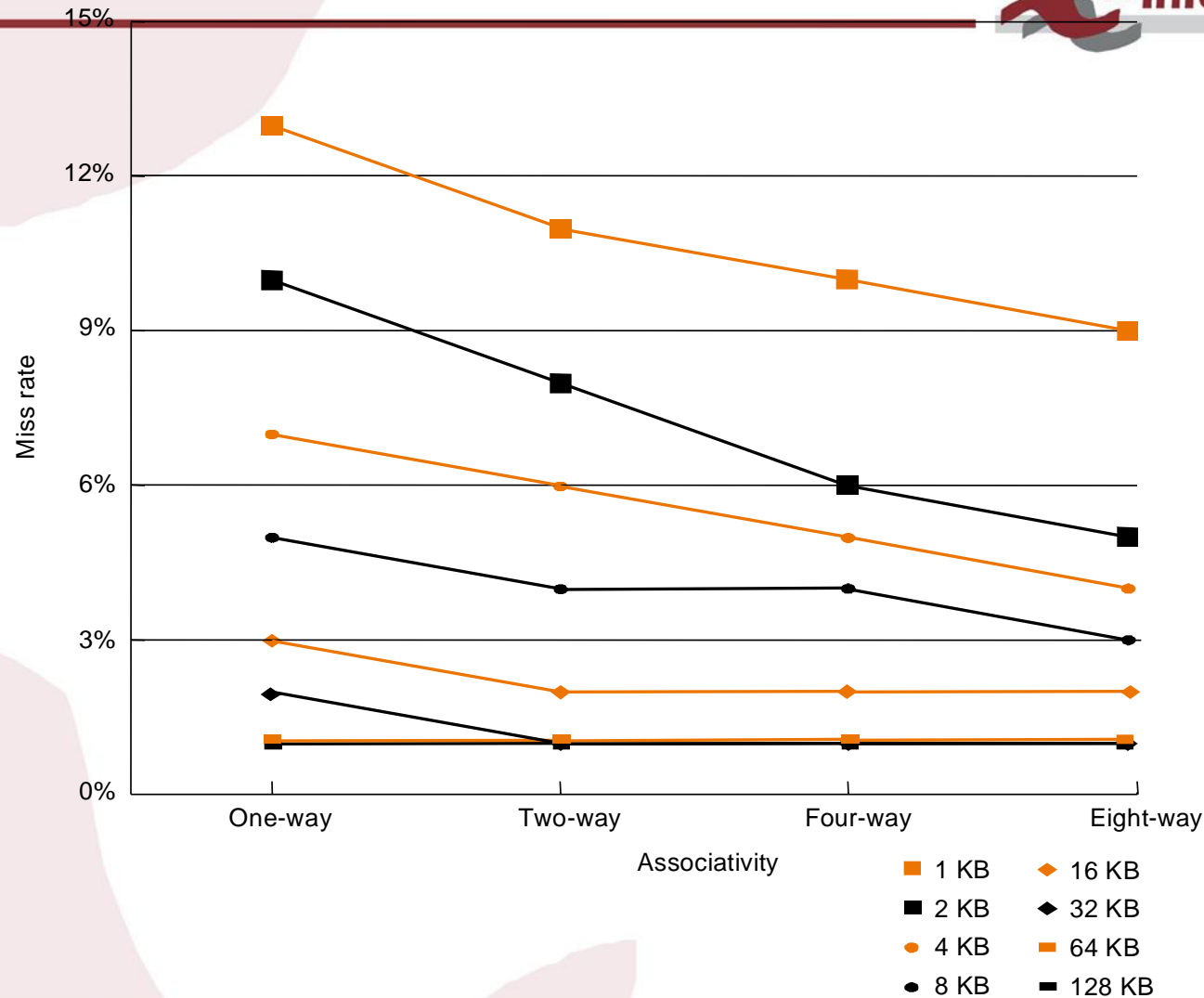
# Aumento da Associatividade



Programa	Associatividade	Miss instr.	Miss dado	Miss Total
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%

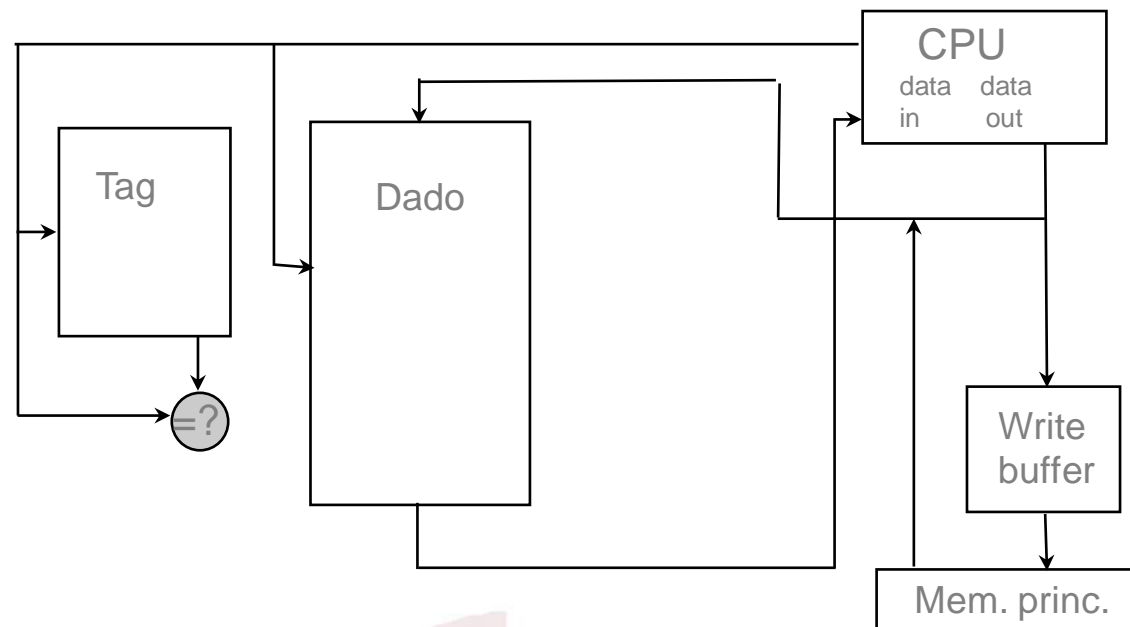


# Aumento da Associatividade



# Reduzindo penalidade de cache

- Write Buffers



# Reduzindo penalidade

---

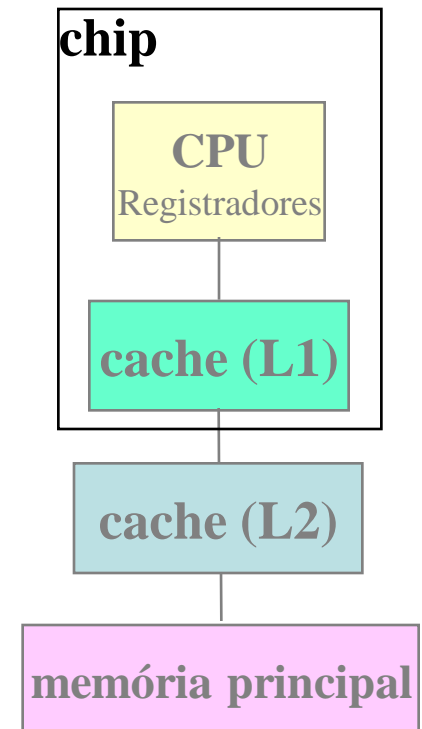


- Early Restart and Critical Word First:
  - Assim que palavra procurada foi carregada na cache esta é enviada para a CPU.
  - Requisita palavra procurada primeiro e a envia para a CPU assim que a mesma foi carregada.
  - Aplicável para grandes blocos



# Reduzindo a penalidade

- Dois níveis de cache:
  - primeiro nível:
    - menor tempo de acesso
    - menor capacidade
    - maior custo
  - segundo nível:
    - maior capacidade
    - menor custo
    - maior tempo de acesso





# Reduzindo penalidade



- Segundo nível de cache:
  - Desempenho:
    - $\text{Avrg.mem.acc.time} = \text{hit}_{L1} + \text{miss}_{L1} \times \text{pen}_{L1}$
    - $\text{Pen}_{L1} = \text{hit}_{L2} + \text{miss}_{L2} \times \text{Pen}_{L2}$
  - De quanto melhora o desempenho da máquina pela inclusão do 2. nível?



# Reduzindo a penalidade

- Exemplo:  $CPI_{base}=1.0$ ,  $Clk=500MHz$ ,  $Time_{mem}=200ns$ ,  $Miss-rate_{mem}=5\%$ .
- Segundo nível:  $Time_{L2}=20ns$ ,  $Miss-rate_{mem}=2\%$
- Qual o desempenho da máquina com 2. nível de cache?

# Reduzindo a penalidade

- Qual o desempenho da máquina com 2. nível de cache?
  - $\text{Penalty}_{\text{mem}} = 200\text{ns} / 2\text{ns}/\text{clk} = 100 \text{ ciclos}$
  - $\text{CPI}_{\text{total}} = \text{CPI}_{\text{base}} + \text{Mem}_{\text{ciclos}} / I = 1.0 + 5\% \times 100 = 6.0$
  - $\text{Penalty}_{L2} = 20 / 2 = 10 \text{ ciclos}$
  - $\text{CPI}_{\text{total}} = 1 + L1\text{-stalls} + L2\text{stalls} =$   
 $1 + ((5\% - 2\%) \times 10) + (2\% \times (10 + 100)) =$   
 $1 + 0.3 + 2.2 = 3.5$
  - $\text{Desempenho} = 6.0 / 3.5 = 1.7$

# Reduzindo penalidade

---



- Primeiro nível de cache:
  - Redução da penalidade
    - Redução do tempo de acesso
    - Uso de técnicas como early-restart e critical-word-first



# Reduzindo penalidade

---



- Segundo nível de cache:
  - Redução da taxa de falta
    - cache do segundo nível maior que a do primeiro nível
  - E quanto a duplicação de dados nos dois níveis?
    - Os dados devem ser duplicados (consistência)



# Memória principal

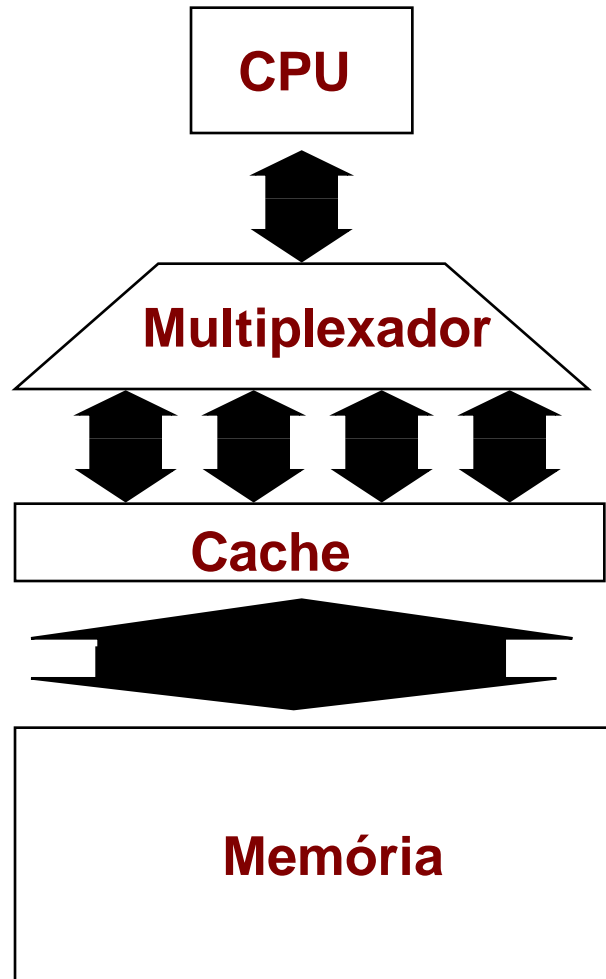
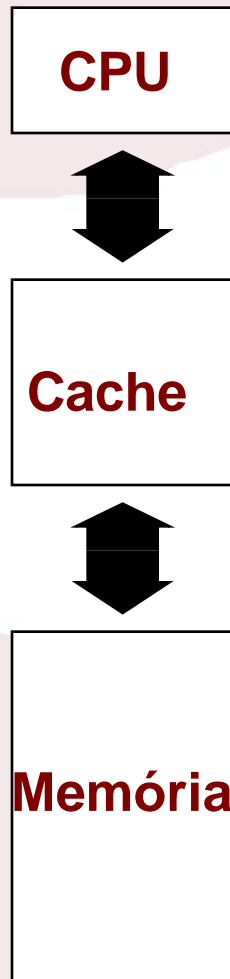
---



- Duplo papel:
  - satisfazer a demanda da cache
  - servir como interface para E/S
- Medidas de performance:
  - latência -> cache
  - Largura de banda -> E/S



# Memórias mais largas



# Memórias mais largas

---

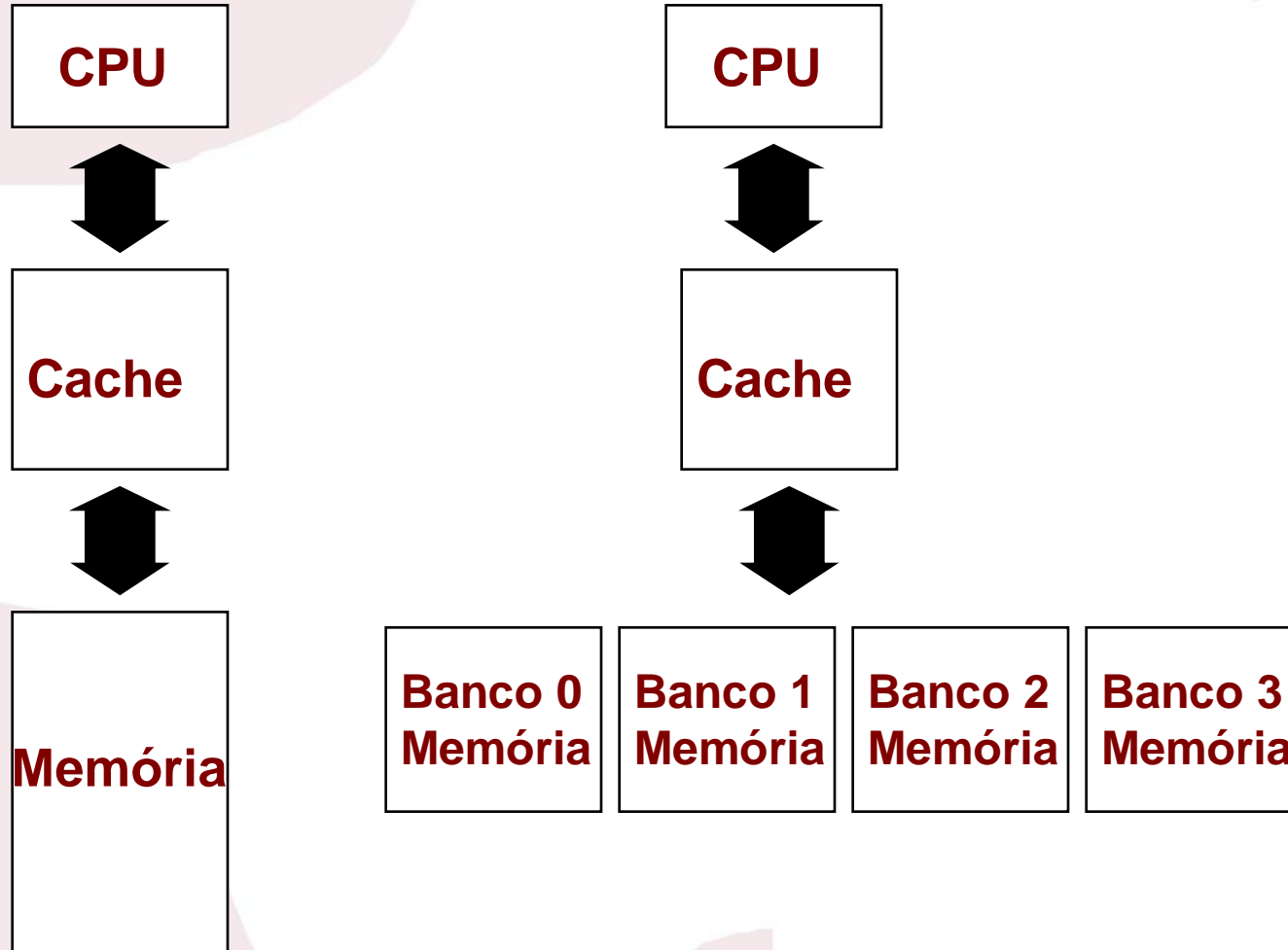


- Redução da penalidade de cache
- Necessidade de barramento e multiplexadores
- Expansão condicionada a largura
- Dificuldade em corrigir erros
- Ex: Alpha :
  - cache e mem. principal => 256 bits





# Memória “Interleaved”



# Memória Interleaved

---



- Bancos de memória para escrita/leitura de múltiplas palavras
- Reduz penalidade
- Necessita pouco hardware adicional



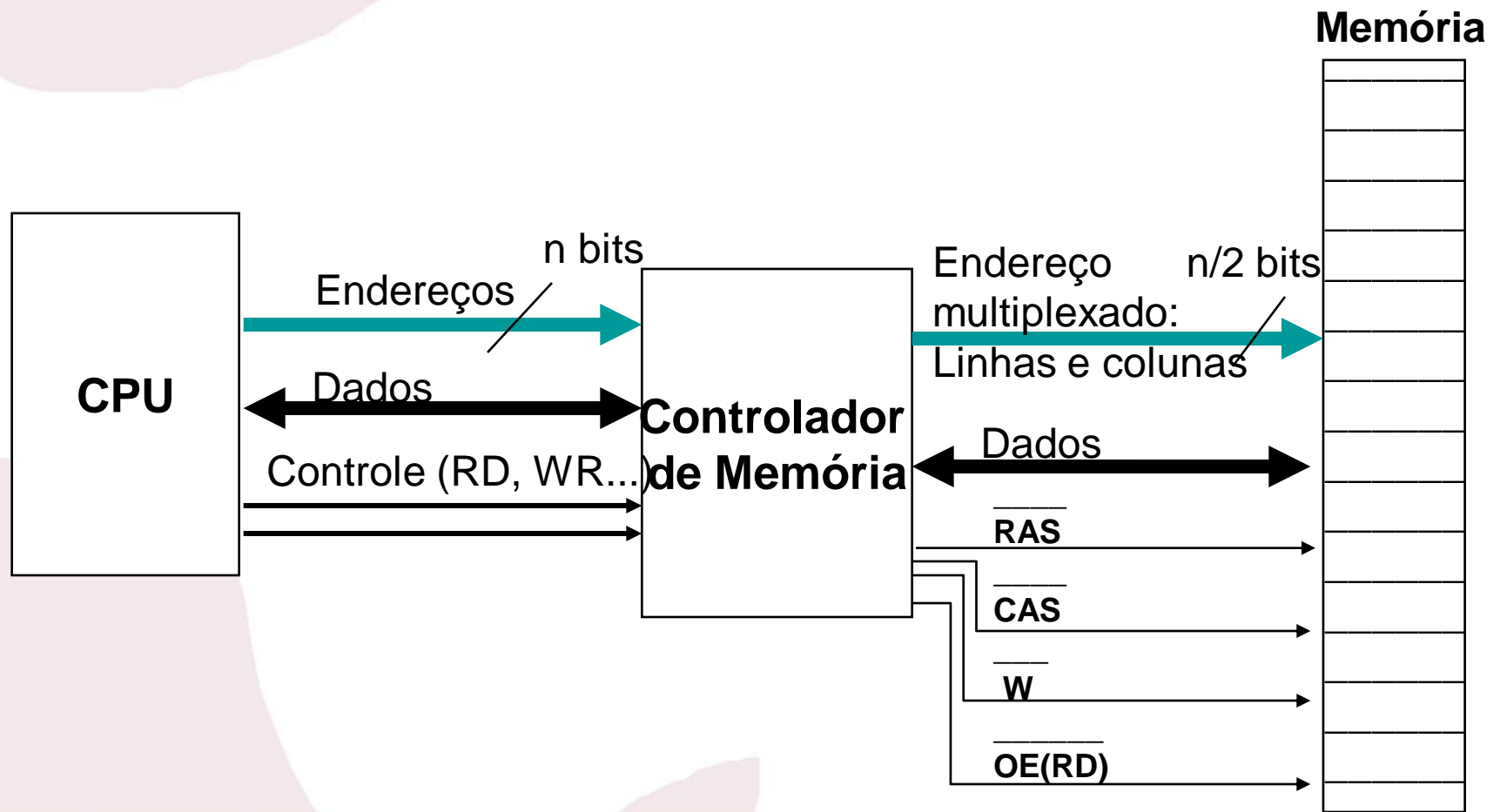
# Memória “larga” vs. Interleaved

- CPI (# clocks por instrução)
  - 32 bits, sem interleaving=3.54
  - 32 bits, interleaving=2.86



# MEMÓRIA PRINCIPAL

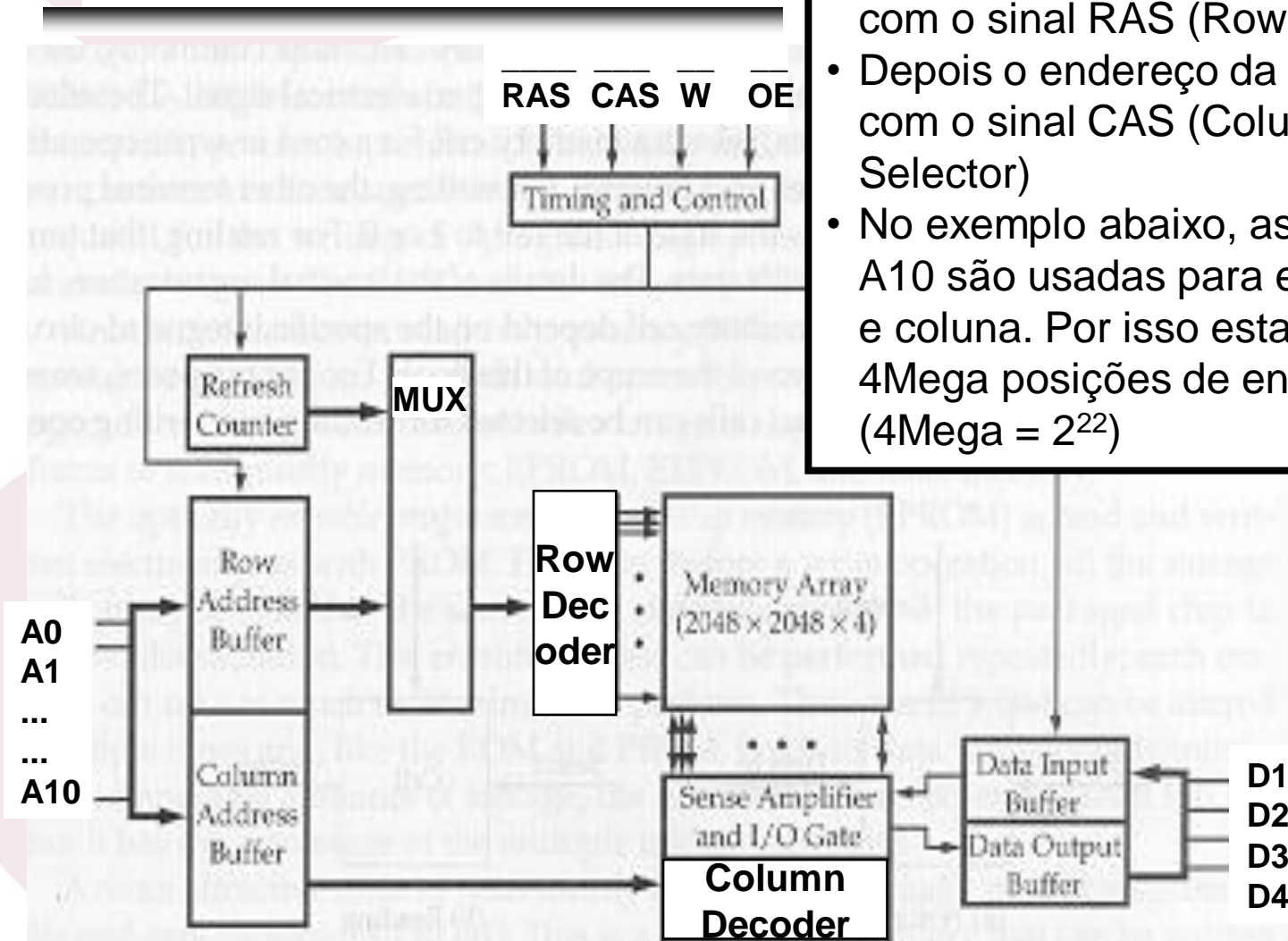
# Acesso a memória principal (DRAM)



# DRAM

## Organização Interna e Método de Acesso

- O endereço das linhas e colunas são enviados separadamente (prim. linha, depois coluna).
- Para acessar uma posição (leitura ou escrita), o endereço da linha é posto junto com o sinal RAS (Row Address Selector).
- Depois o endereço da coluna vai junto com o sinal CAS (Column Address Selector)
- No exemplo abaixo, as 11 linhas de A0 a A10 são usadas para enviar end. de linha e coluna. Por isso esta memória tem 4Mega posições de endereçamento ( $4\text{Mega} = 2^{22}$ )



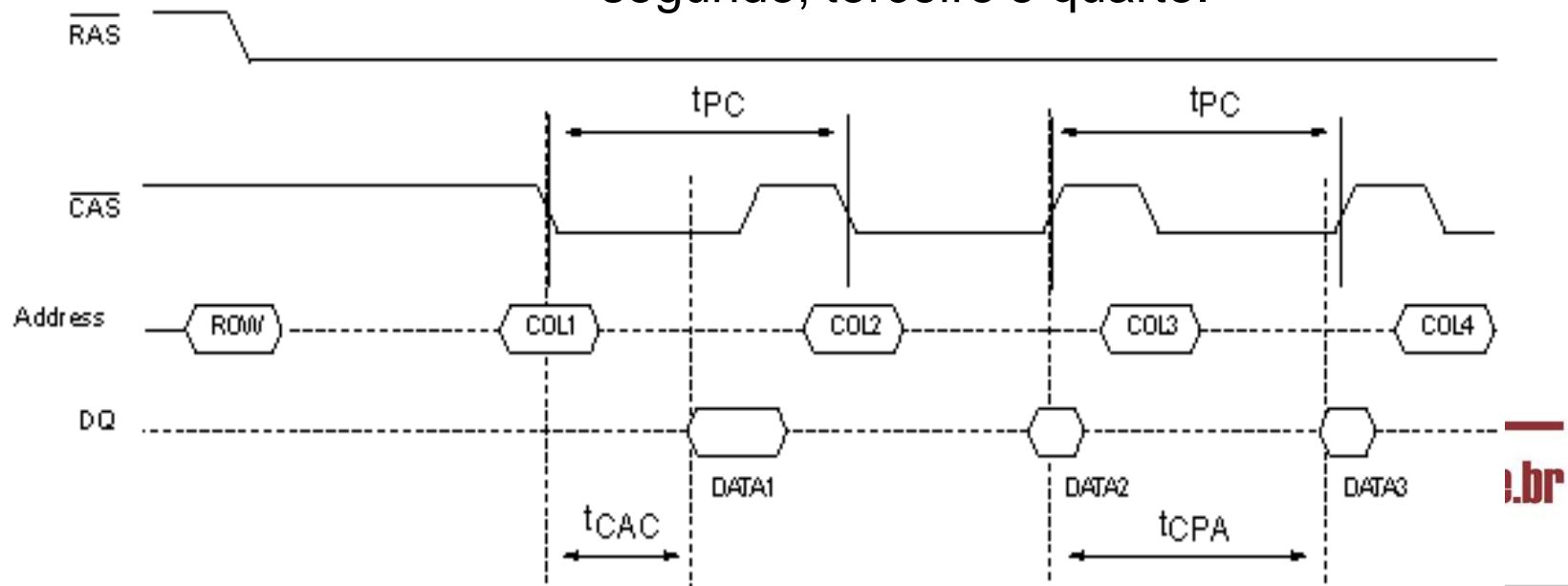
# Modo de Acesso

Ex: FPM RAM

- Ativa uma fila (RAS)
- Acessos sequenciais a colunas (vários pulsos de CAS)

- No Page Mode, o controlador de memória faz até 4 acessos em sequência à DRAM. É comum designar o núm. de pulsos de clock de cada acesso para cada tipo de memória.
- Ex. a FPM RAM tem acesso 5/3/3/3 (a 66MHz) em page mode, o que significa 5 pulsos de clock para obter o primeiro dado, e 3 para o segundo, terceiro e quarto.

## Fast Page Mode:



# FPM DRAM - Fast Page Mode DRAM

## EDO DRAM - Extended Data-Output DRAM

### FPM DRAM

- DRAM mais simples
- Tempo de acesso  
70ns and 60ns.
- Acesso page mode  
= 5/3/3/3 a 66MHz

### EDO DRAM

- Tempo de acesso  
70ns, 60ns and  
50ns  
Para barramento  
de 66 MHz use 60  
ns ou melhor.
- Acesso page mode  
= 5/2/2/2 a 66MHz





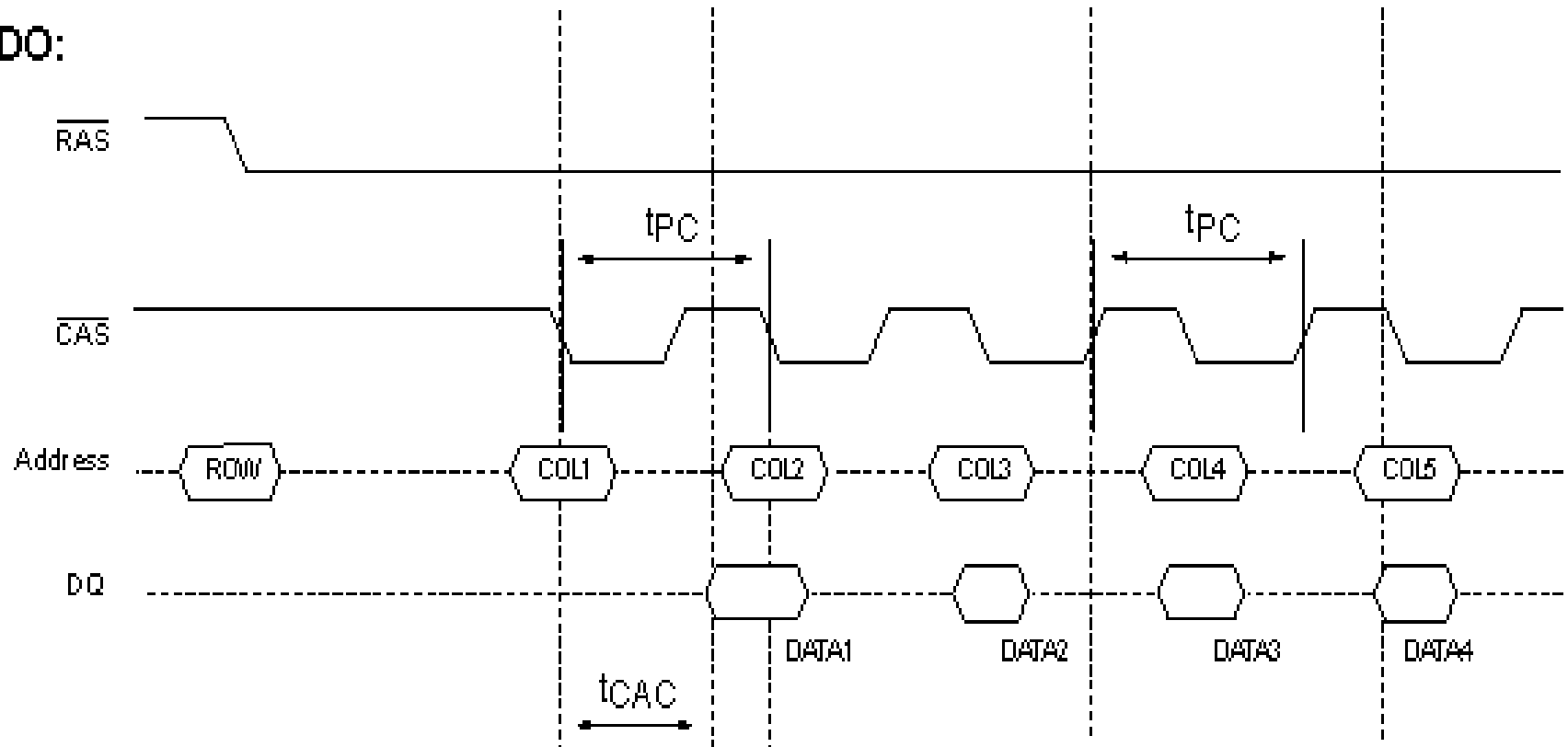
# EDO DRAM

## Método de Acesso



- No page mode, um latch na saída de dados permite o acesso simultâneo a novas posições de memória enquanto os dados estão sendo lidos na saída.
- Isso permite a diminuição do tempo entre pulsos de CAS

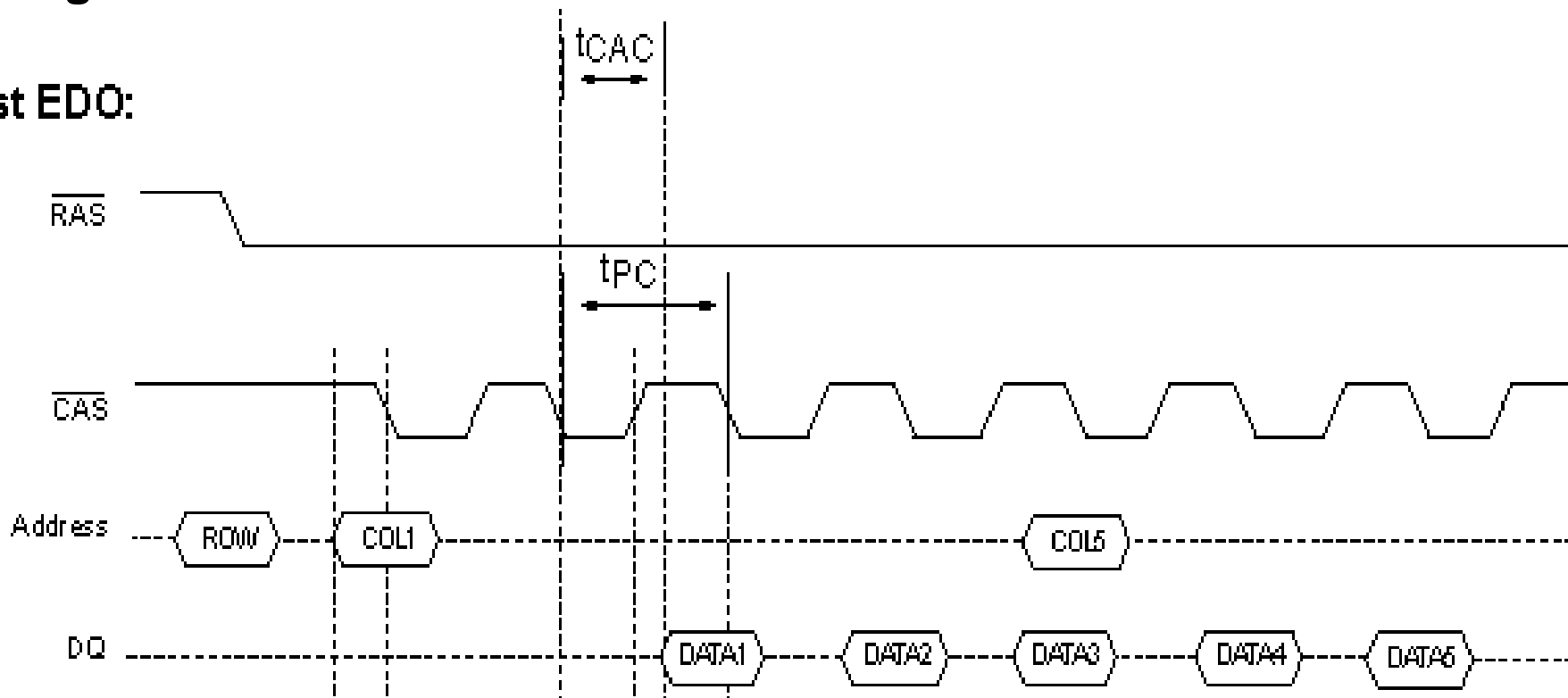
EDO:



# BEDO - Burst EDO DRAM

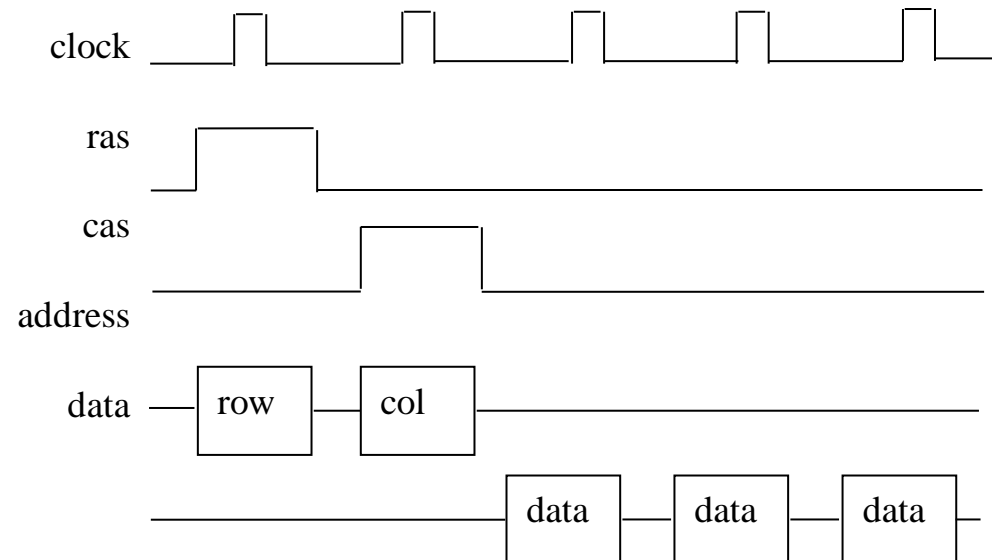
- Page (burst) mode = 5/1/1/1 a 66MHz
- Tempo de acesso randômico é igual ao FPM ou EDO RAM
- Possui registrador e gerador interno de ender. sequenciais

## Burst EDO:



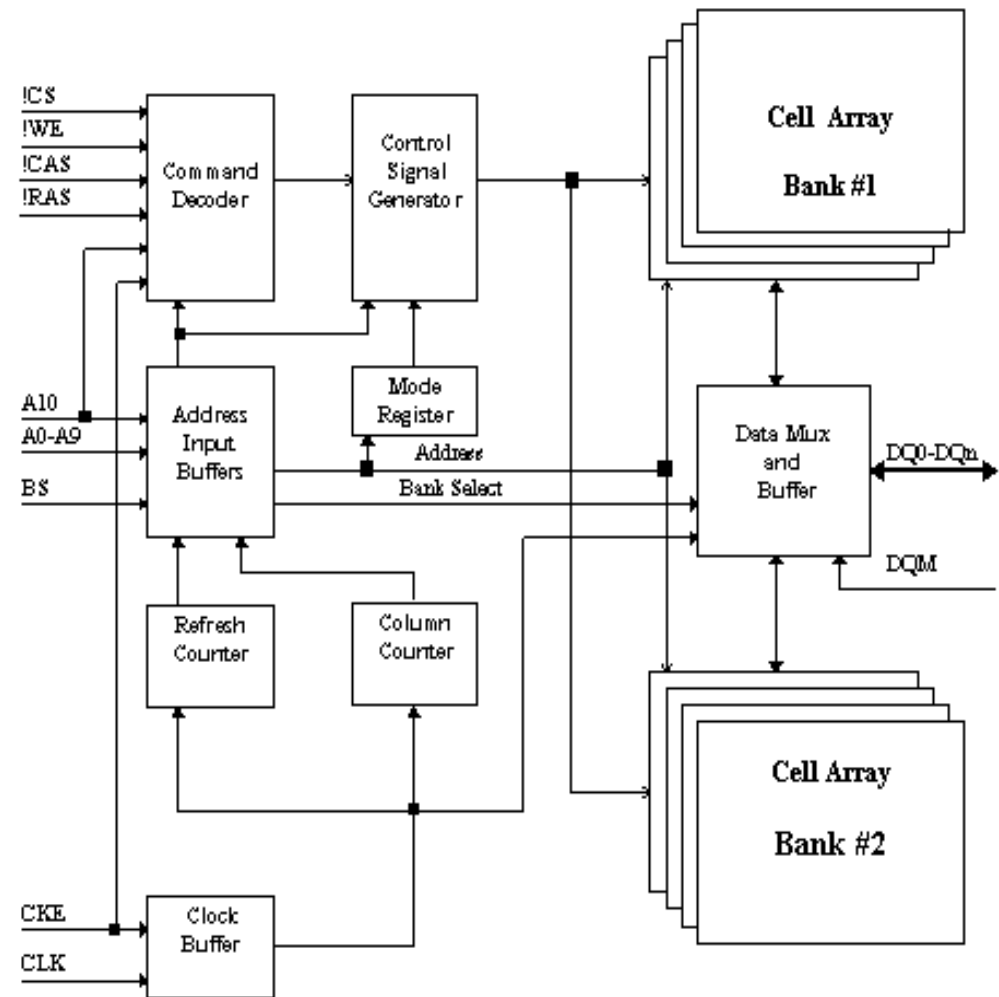
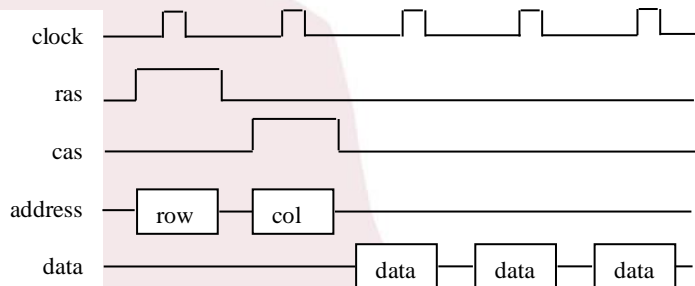
# SDRAM - Synchronous DRAM

- Page (burst) mode = 5/1/1/1 a 100MHz
- Tempo de acesso randômico é igual à FPM ou EDO RAM.
- Trabalha na velocidade do bus, por isso o nome.



# SDRAM - Synchronous DRAM

- Page (burst) mode  
= 5/1/1/1 a 100MHz
- Interleaved
- Uso de Serial  
Presence Detect  
para Plug-and-Play



# DDR SDRAM – Double Data Rate SDRAM

- Transfere dados na subida e descida do clock (compensa o uso de barramentos lentos)
- Uso de Serial Presence Detect para Plug-and-Play



- Hierarquia de Memória
  - Memórias com diferentes características:
    - Tempo acesso
    - Capacidade
    - Custo
  - Uso da localidade temporal e espacial
  - Primeiro nível da hierarquia
    - Memória cache
  - Tipos de Cache
    - Mapeamento direto
    - Associativa
    - Associativa por conjunto

# Resumo



- Melhorando o desempenho
  - Aumentar tamanho do bloco
  - Aumentar a associatividade
  - Redução da penalidade
  - Cache multinível
- Memória Principal
  - Memórias largas
  - Bancos de memória
- Memória DRAM:
  - Acessos de endereços consecutivos mais eficiente
    - SDRAM ou DDR SDRAM

