Eletrônica Digital

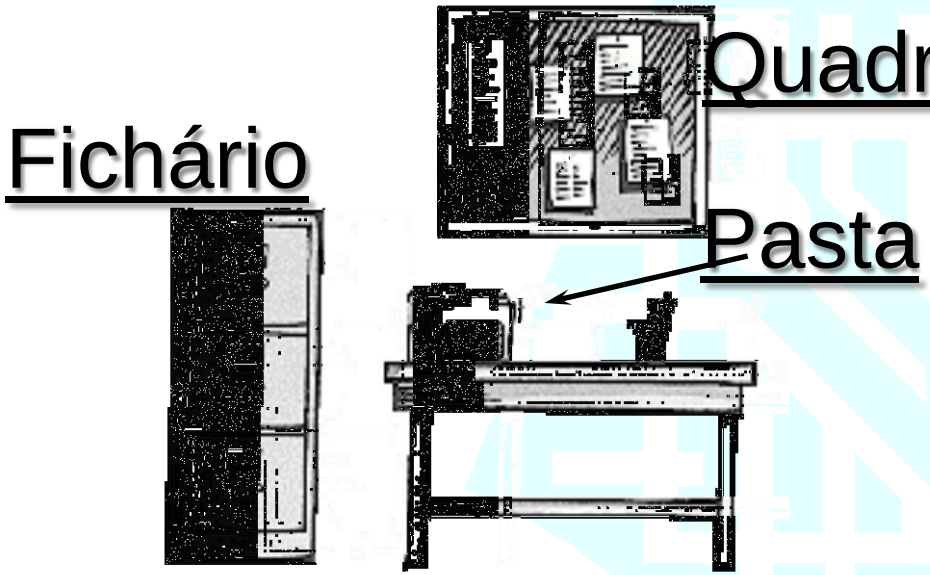
Memória Cache

Prof. Rômulo Calado Pantaleão Camara

Carga Horária: 60h

Memória Cache

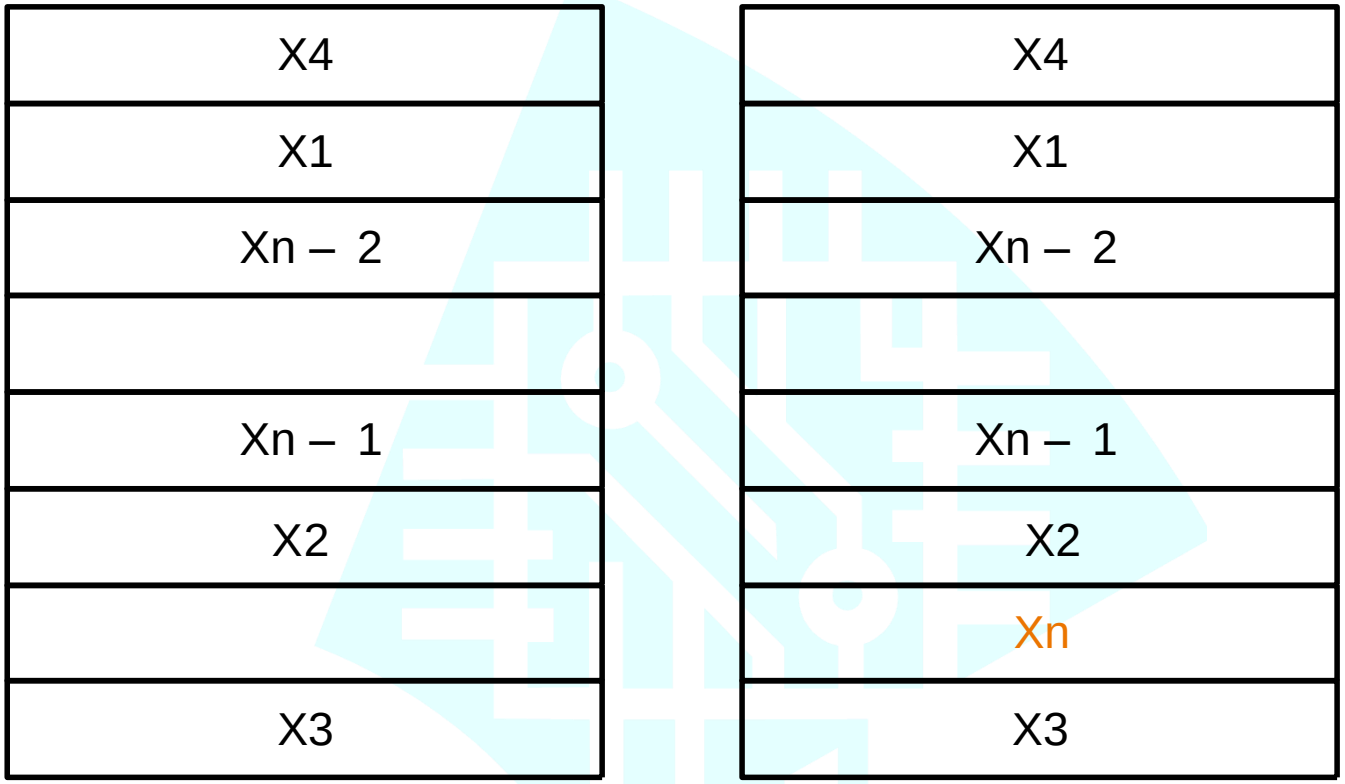
Memória Principal Vs. Cache



- ✓ 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

Cache Simples

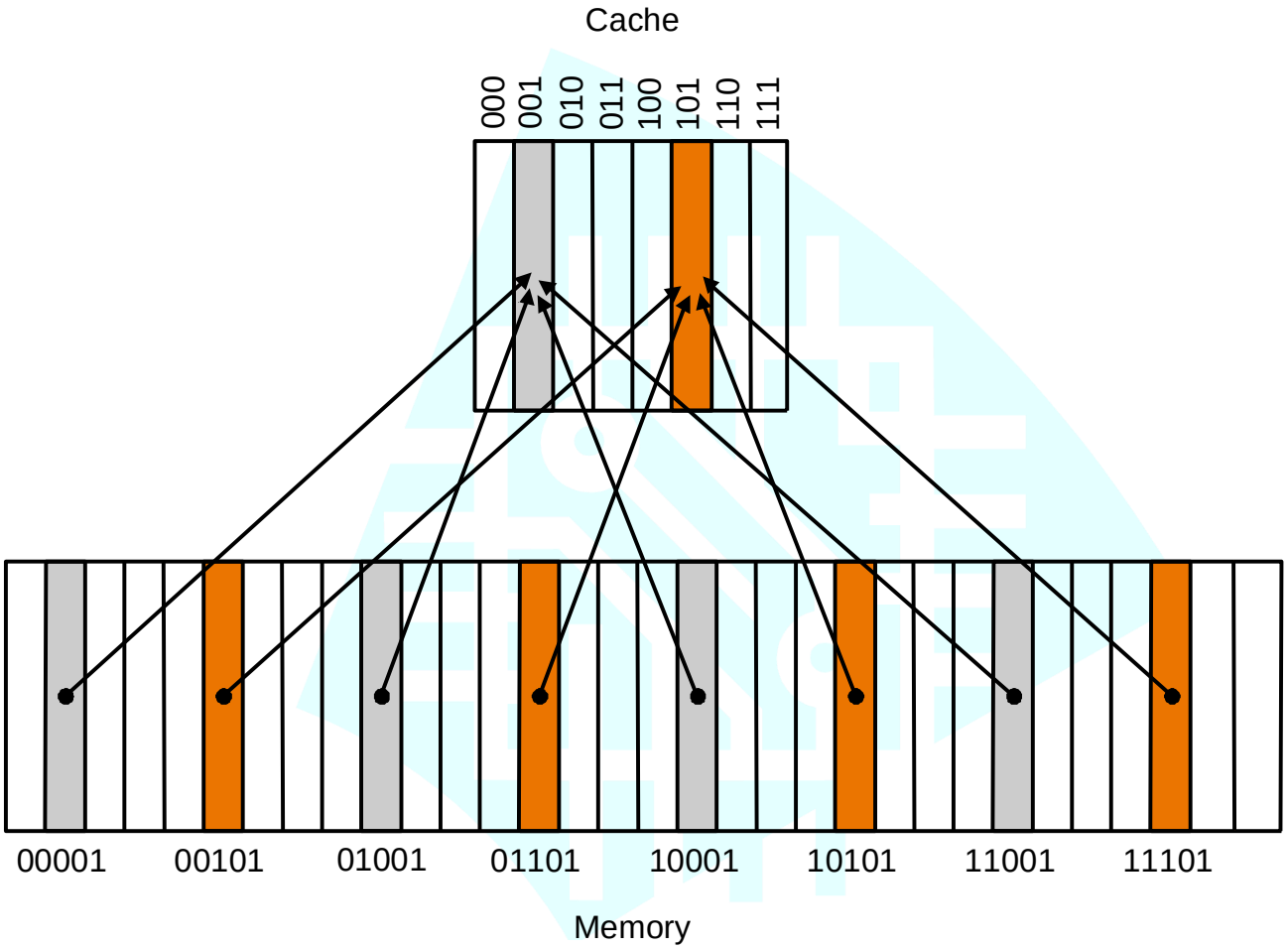
- ✓ Unidade de transferência: palavra



a. Before the reference to Xn

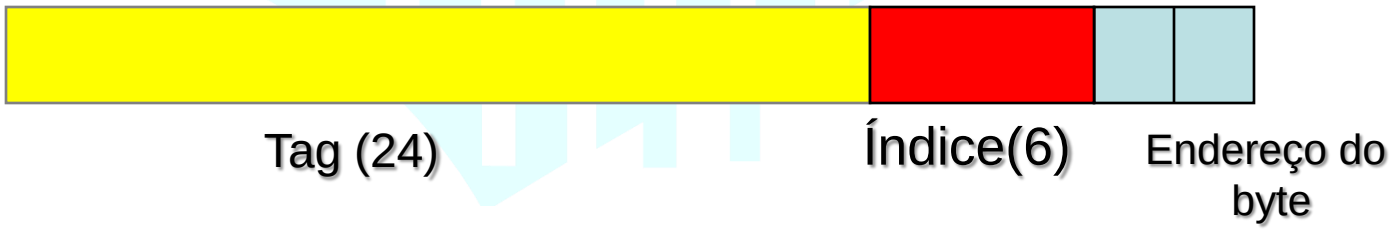
b. After the reference to Xn

Mapeamento direto



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

Endereço decimal	*Endereço binário*	*Hit ou miss na cache*	*Bloco na cache*
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

	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 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

	V	Tag	Dado
000	N		
001	N		
010	N		
011	N		
100	N		
101	N		
110	Y	10	Memória(101110)
111	N		

Endereço 1011

➡

Endereço decimal	*Endereço binário*	*Hit ou miss na cache*	*Bloco na cache*
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

	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

Endereço decimal	*Endereço binário*	*Hit ou miss na cache*	*Bloco na cache*
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

	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(101110)
111	N		

Endereço 10000

Endereço decimal	*Endereço binário*	*Hit ou miss na cache*	*Bloco na cache*
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

	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(101110)
111	N		

Endereço 00011

Endereço decimal	*Endereço binário*	*Hit ou miss na cache*	*Bloco na cache*
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

	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(101110)
111	N		

Endereço 10010

Endereço decimal	*Endereço binário*	*Hit ou miss na cache*	*Bloco na cache*
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

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

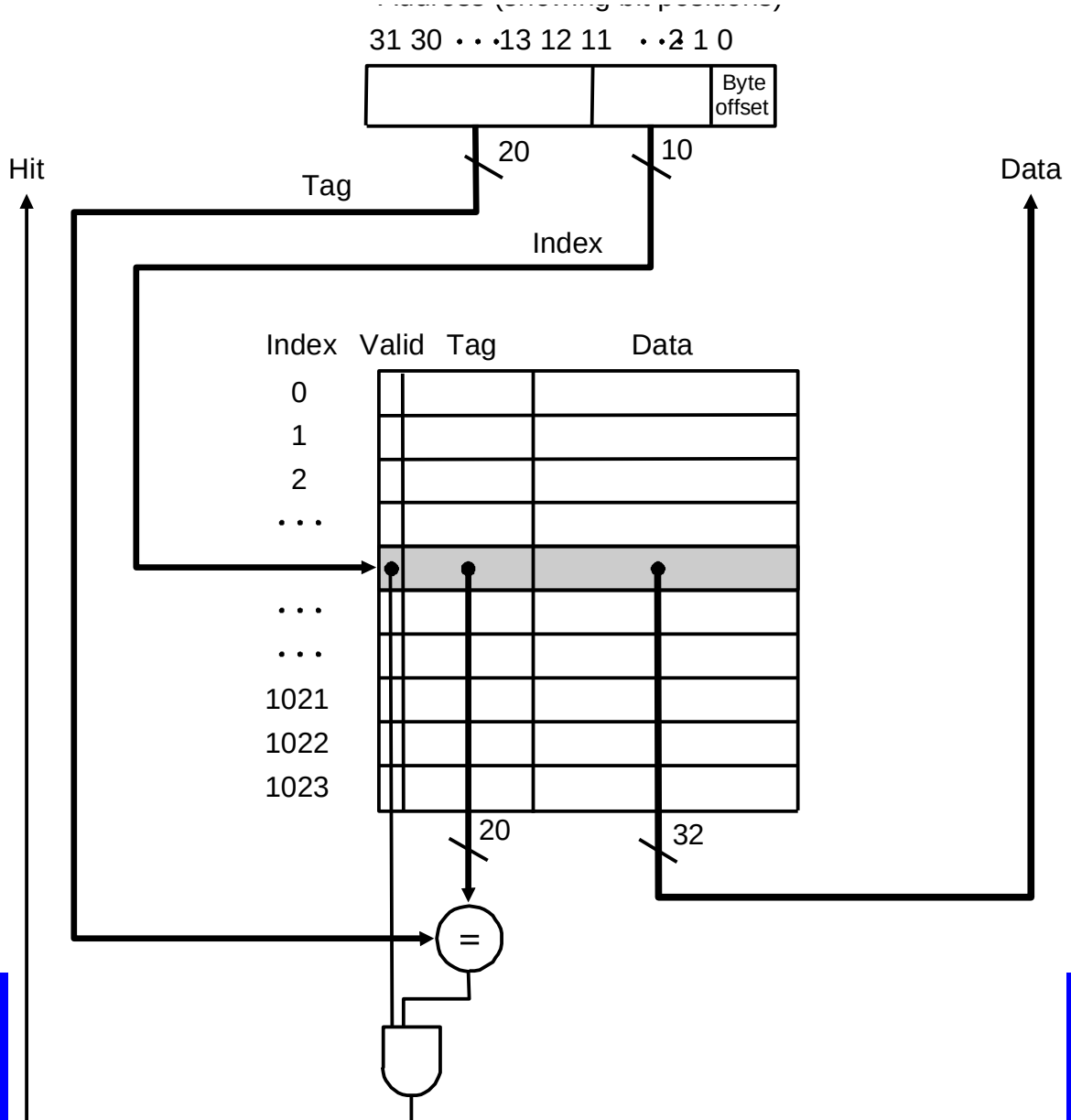
- ✓ Caches do tipo Write through
  - Cache e memória são atualizadas simultaneamente
- ✓ Caches do tipo Write back
  - Memória principal é atualizada quando bloco é substituído
  - Usa dirty bit para marcar linhas alteradas na cache.

Memória Cache: escrita

Write through	*Write back*
facilidade de implementação ***consistência da memória principal***	***redução de acessos à memória***

- Para se evitar espera durante escrita:
  - Write buffers

Exemplo: DECStation3100



Tamanho da cache

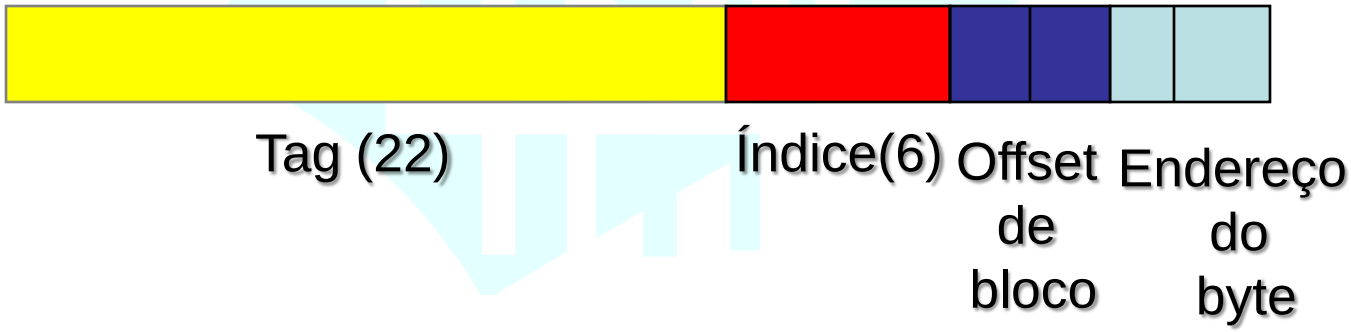
- ✓ Quantos bits tem uma cache de mapeamento direto com 64K bytes de dados e blocos de uma palavra? Assuma endereços de 32 bits.

Usando a localidade espacial

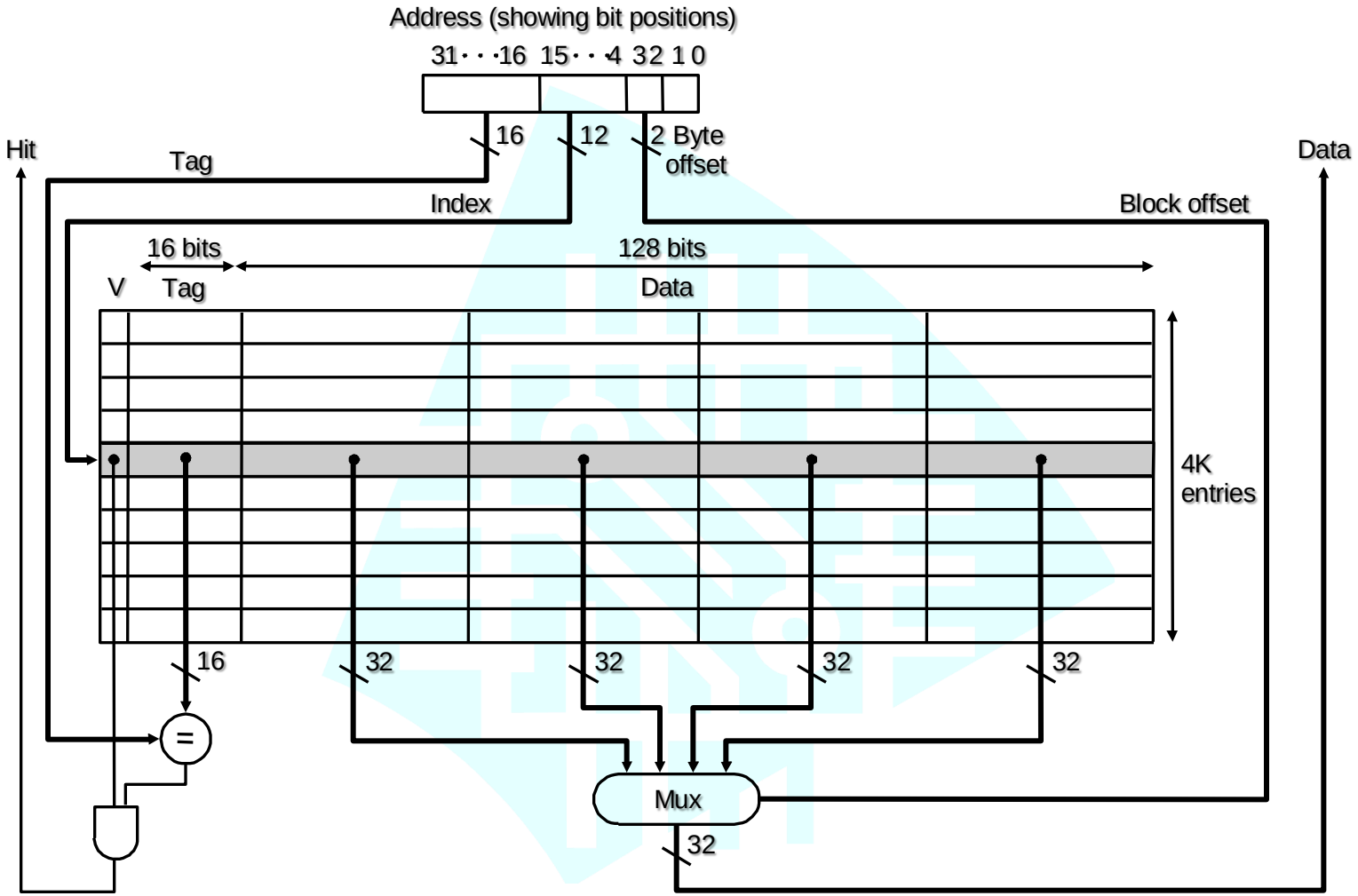
- ✓ Acessando a cache por blocos de palavras
- ✓ Composição do endereço:
  - Tag
  - Índice
  - Offset de bloco
  - Endereço de byte

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



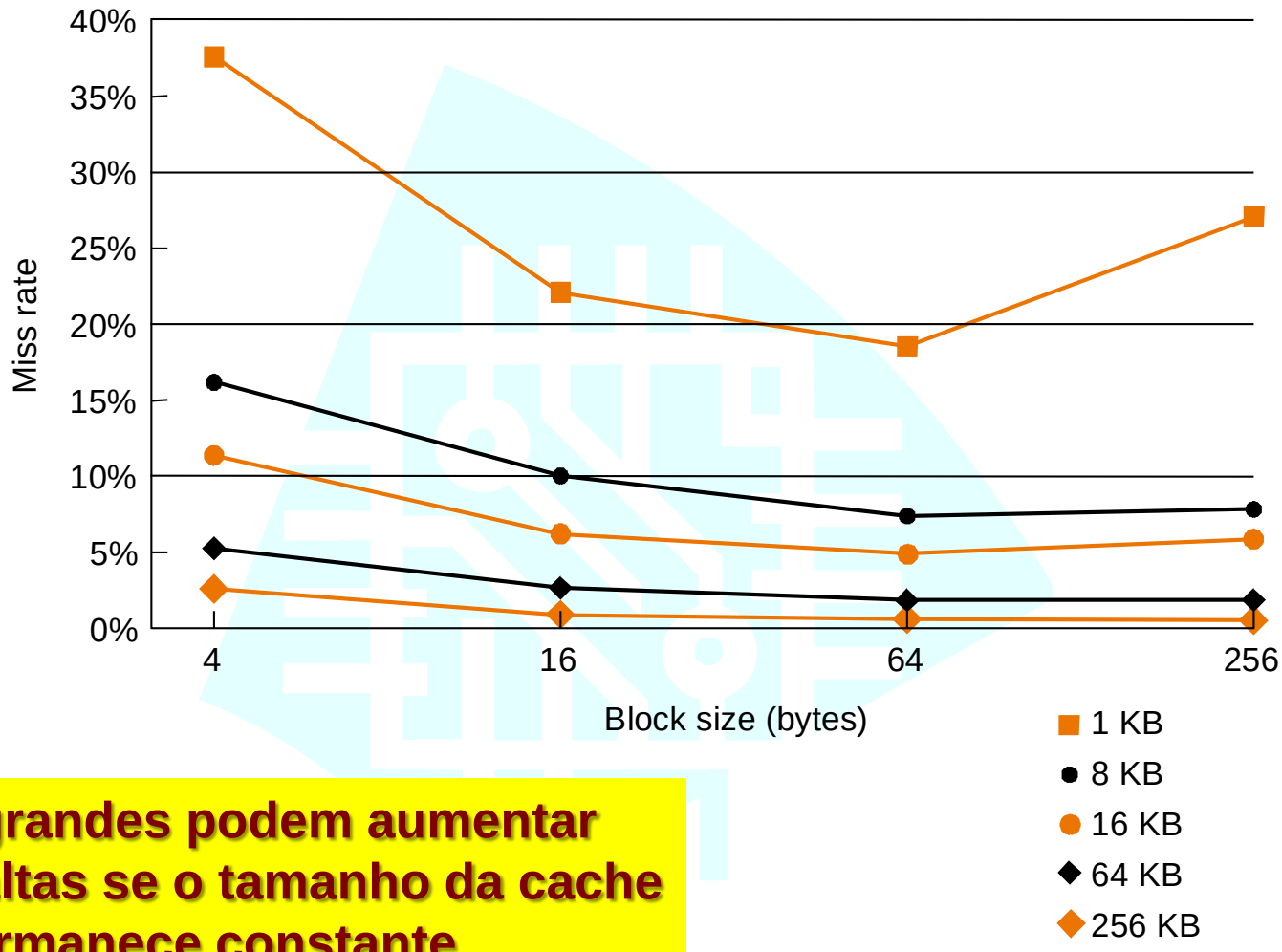
Mapeamento Direto – bloco > 1 palavra



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

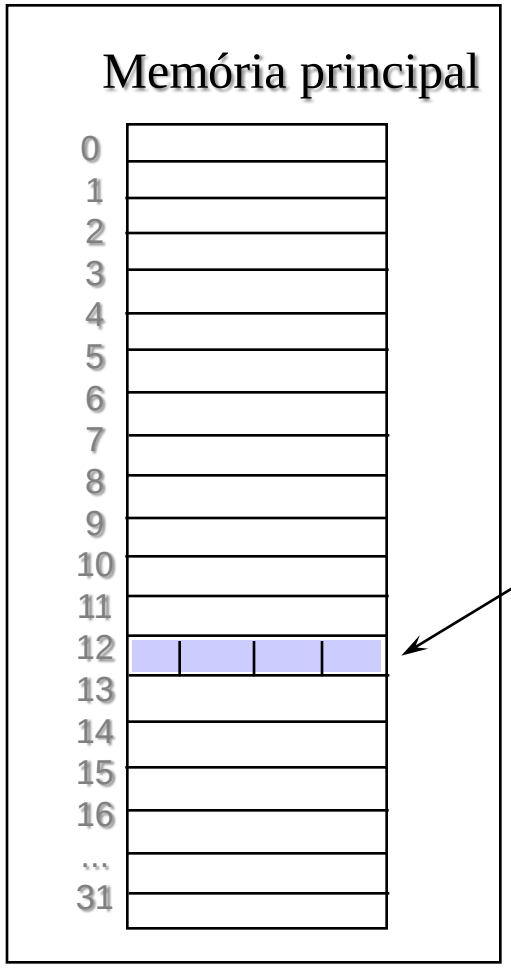


Blocos grandes podem aumentar a taxa de faltas se o tamanho da cache Permanece constante

Reduzindo a taxa de faltas

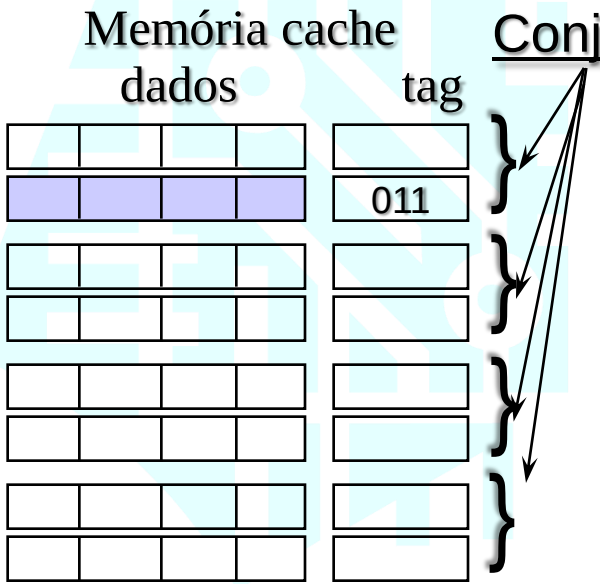
- ✓ Estratégias para posicionamento dos blocos:
  - Mapeamento direto: cada bloco possui posição única na cache
  - Associativa por conjunto: cada bloco pode ser colocado em algumas posições na cache
  - Completamente Associativa: cada bloco pode ser colocado em qualquer posição da cache

Mapeamento Associativo por Conjunto



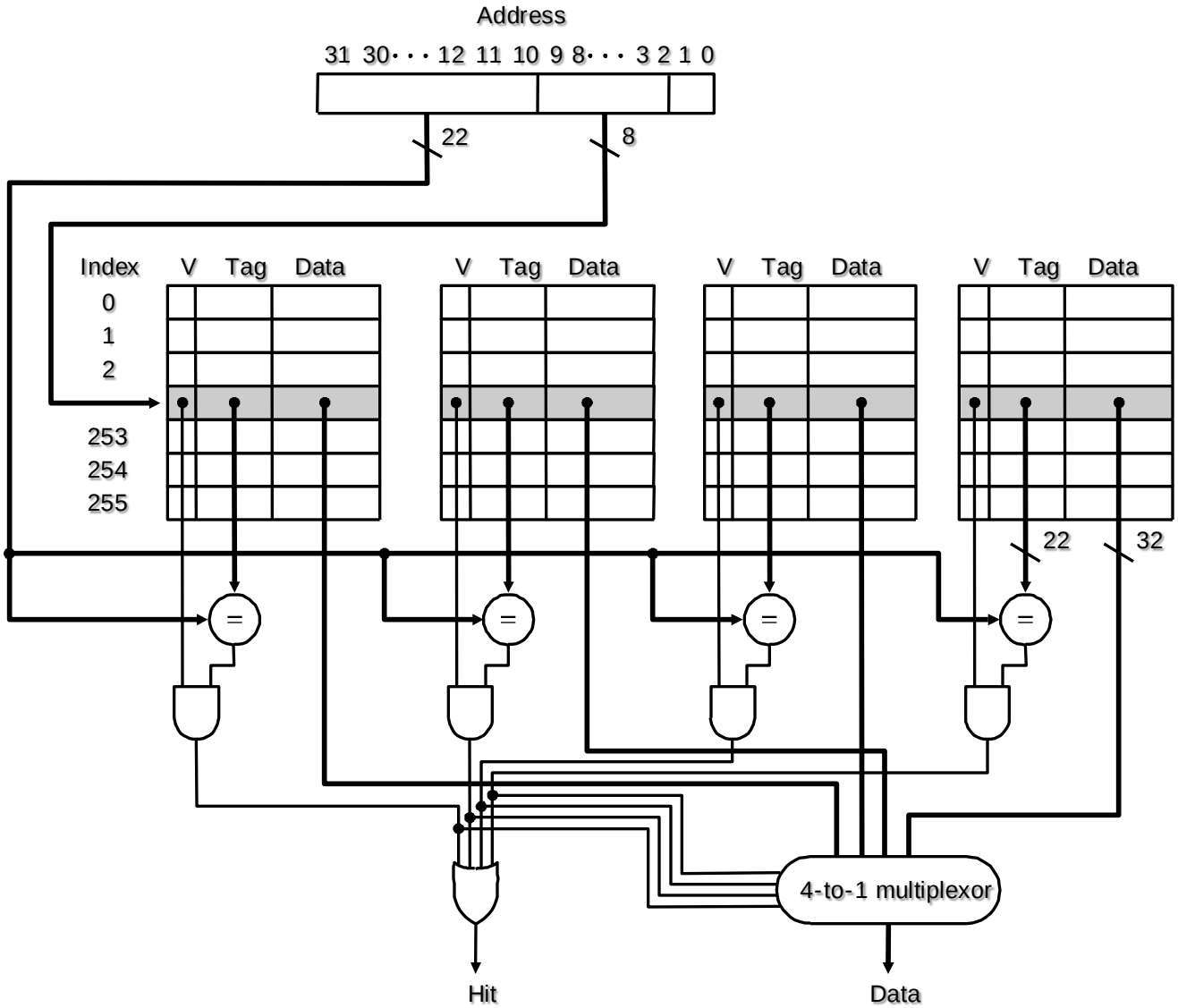
Um bloco na memória principal

pode ocupar qualquer posição dentro de um conjunto de blocos da cache

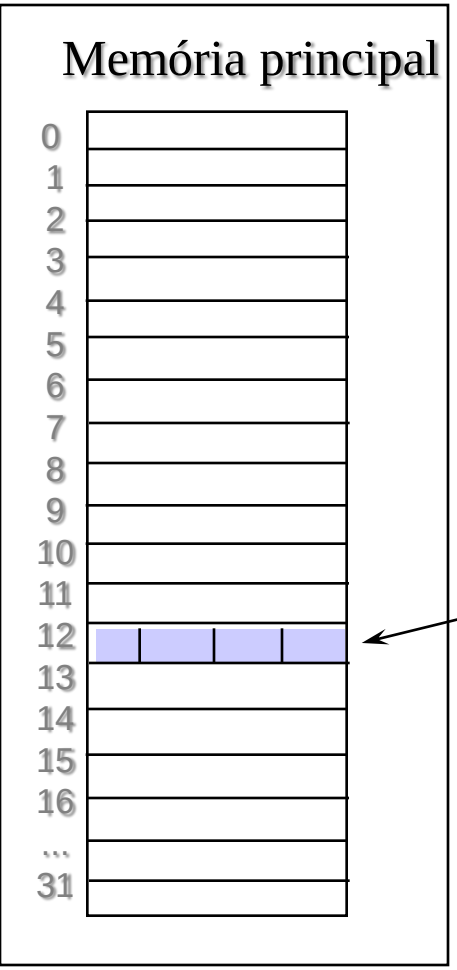


Endereço da palavra		
Tag	set	Offset
011	00	Offset

Mapeamento Associativo por Conjunto

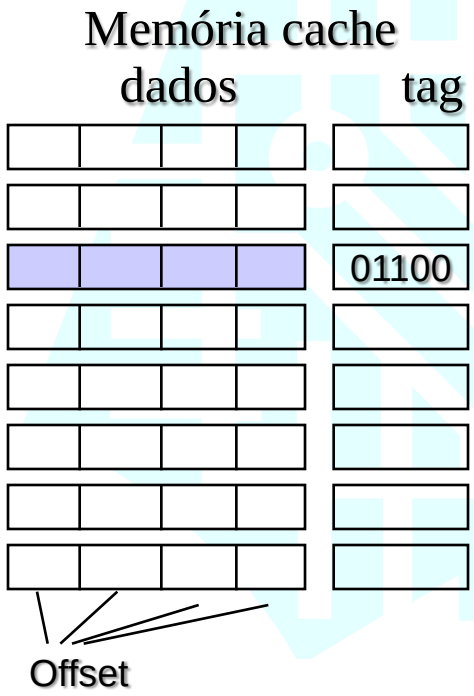


Mapeamento Associativo

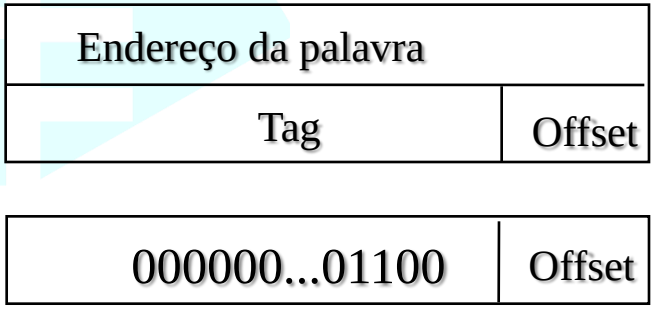


Um bloco na memória principal pode ocupar qualquer posição na cache

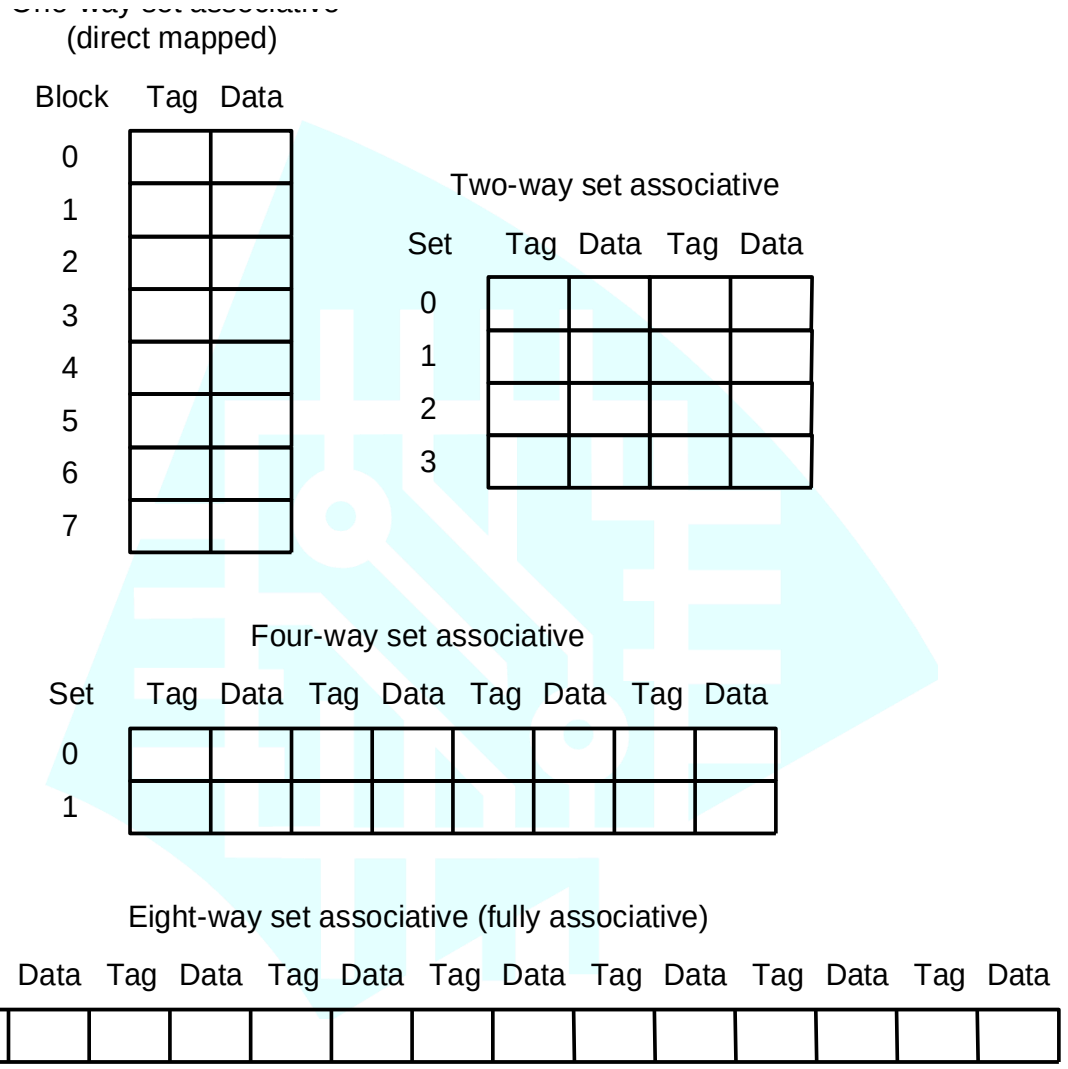
- **Tag ⇒ armazena na cache o end. do bloco na mem. principal**



Exemplo: tag = 12 (01100_2)



Grau de associatividade



Comparação de Métodos de Mapeamento

- ✓ Mapeamento direto
  - – **Simples e Barata**
  - – **Lenta**
  - – **Mais faltas**

NBC = núm. blocos da cache
NCC = núm. conjuntos da cache

- ✓ Associativa
  - —Rápida
  - —Menos falta
  - —Cara (comparação do endereço em paralelo)

- Associativa por conjunto: combinação das anteriores

• Se NCC = NBC ⇒ Ass. por conjunto = Mapeamento Direto

• Se NCC = 1 ⇒ Ass. por conjunto = Associativa

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

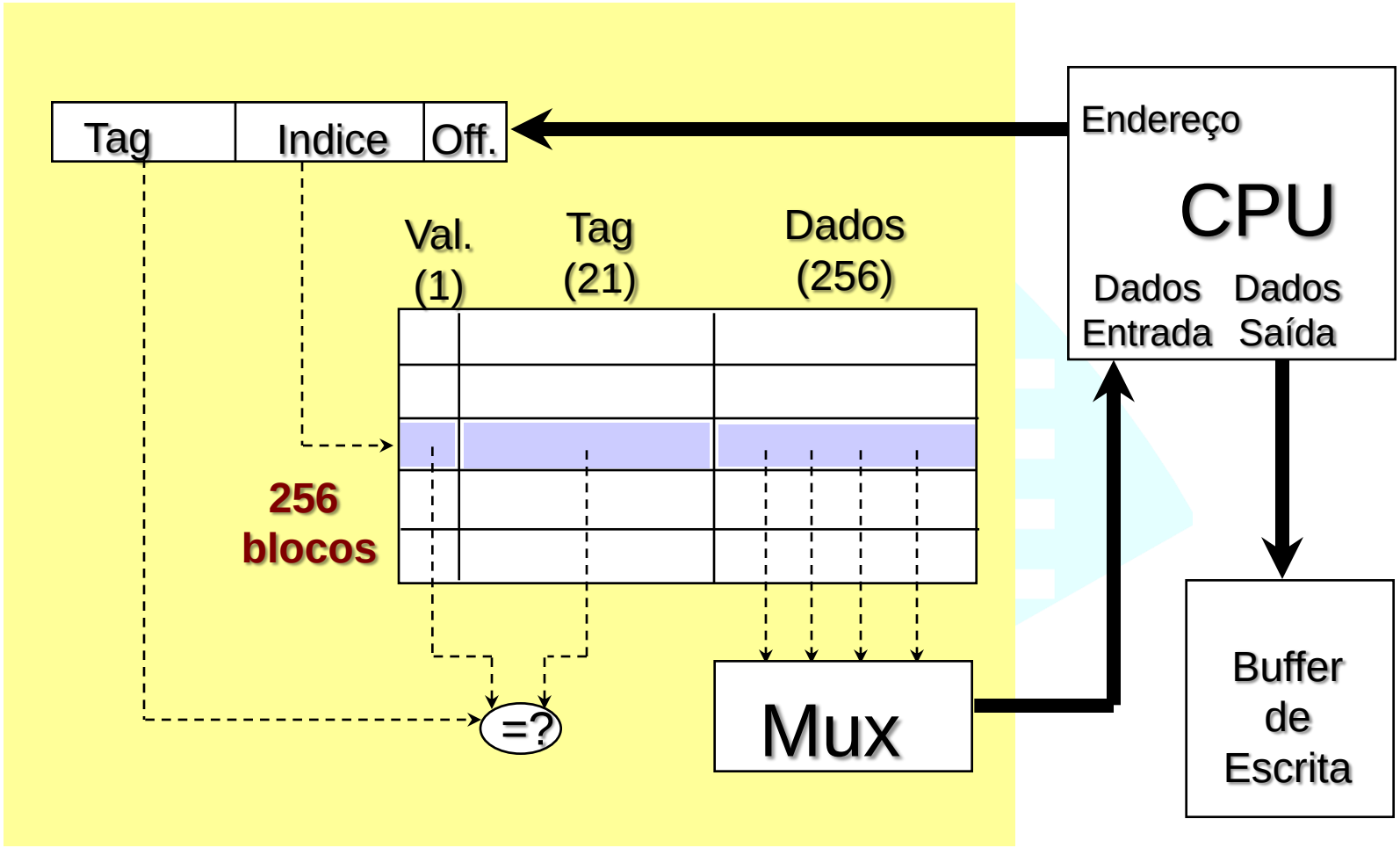
- ✓Cache de dados e cache de instruções
  - Vantagens:
    - Melhor capacidade de otimizações
    - Evita hazard estrutural
  - Desvantagens:
    - maior taxa de falta

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

Desempenho de uma CPU

- ✓ CPU_{time}= (CPU_Ciclos para execução + Memória_ciclos-stall) x ClK_período
- ✓ CPU_ciclos execução= #instruções x CPI
- ✓ $Memória_{ciclos}$= $Leitura_{ciclos}$+ $Escrita_{ciclos}$
- ✓ $Leitura_{ciclos}$= Leituras x $Miss_rate_{leitura}$x $Penalty_{leitura}$
- ✓ $Escrita_{ciclos}$= Escritas x $Miss_rate_{escrita}$x $Penalty_{escrita}$

CPI = nr. de clocks para cada instrução

Desempenho de uma CPU

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

Desempenho de uma CPU

- ✓ $Miss\_instr_{ciclos} = I \times 2\% \times 100 = 2{,}00\ I$
- ✓ $Miss\_dados_{ciclos} = I \times 36\% \times 4\% \times 100 = 1{,}44\ I$
- ✓ $CPU_{time} = (CPU_{ciclos\text{-}execução} + Memória_{ciclos\text{-}stall}) \times Clk$
- ✓ $Memória_{ciclos} = 2\ I + 1{,}44\ I = 3{,}44\ I$
- ✓ $CPI_{mem} = 2{,}0 + 3{,}44 = 5{,}44$
- ✓ $CPI_{stall} \ / \ CPI_{perf} = 5{,}44/2 = 2{,}72$

Processador mais rápido...

- ✓ Diminuindo CPI
  - CPI_{novo}= 1,0
  - CPI_{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
  - $Miss_{ciclos}$(por instr.)= 2% x 200 + 36% (4% x 200) = 6,88
  - CPI_{mem}= 2,0 + 6,88 = 8,88
  - CI x $CPI_{ck\text{-}lento}$ x ciclo / CI x $CPI_{ck\text{-}rap.}$ x ciclo = 5,44 / 8,88x0,5 = 1,23

Desempenho de uma cache

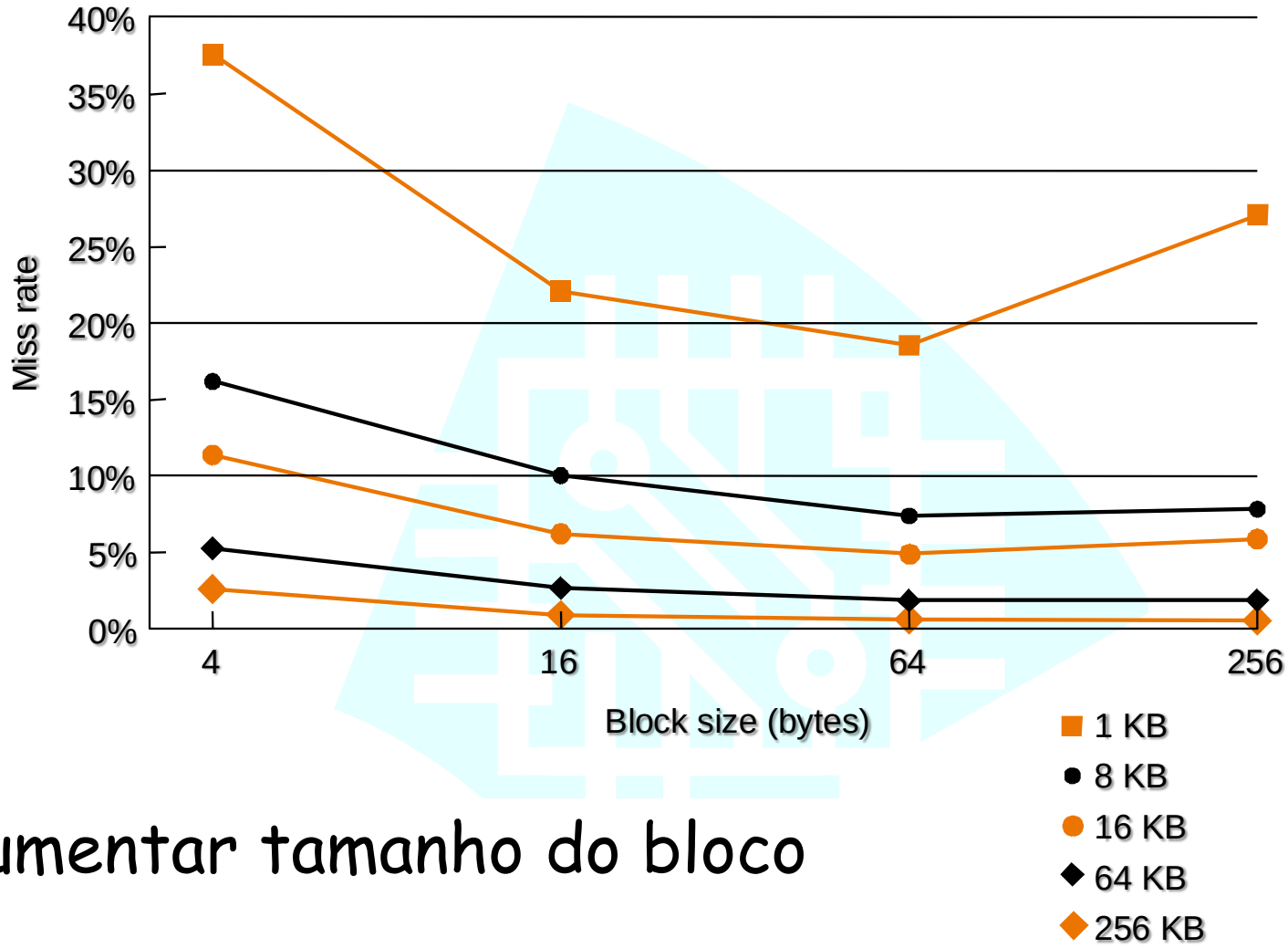
- ✓ CPI(#clocks por instrução)
  - Cache Perfeita => 2,0
  - Cache (2% $miss_{instr}$, 4% $miss_{dado}$) => 5,44
  - Sem cache => 68.5
- ✓ Melhorando o processador
  - Diminuindo CPI
    - CPI => 4,44 (em vez de 1,0)
  - Duplicando clock
    - CPI => 8,88

Melhorando desempenho da cache

$$Tempo\_acesso_{médio} = Hit\ time + Miss\ rate \times Miss\ penalty$$

- ✓ Estratégias:
  - Redução de faltas
  - Redução da penalidade
  - Redução do tempo de acesso

Reduzindo falta de cache



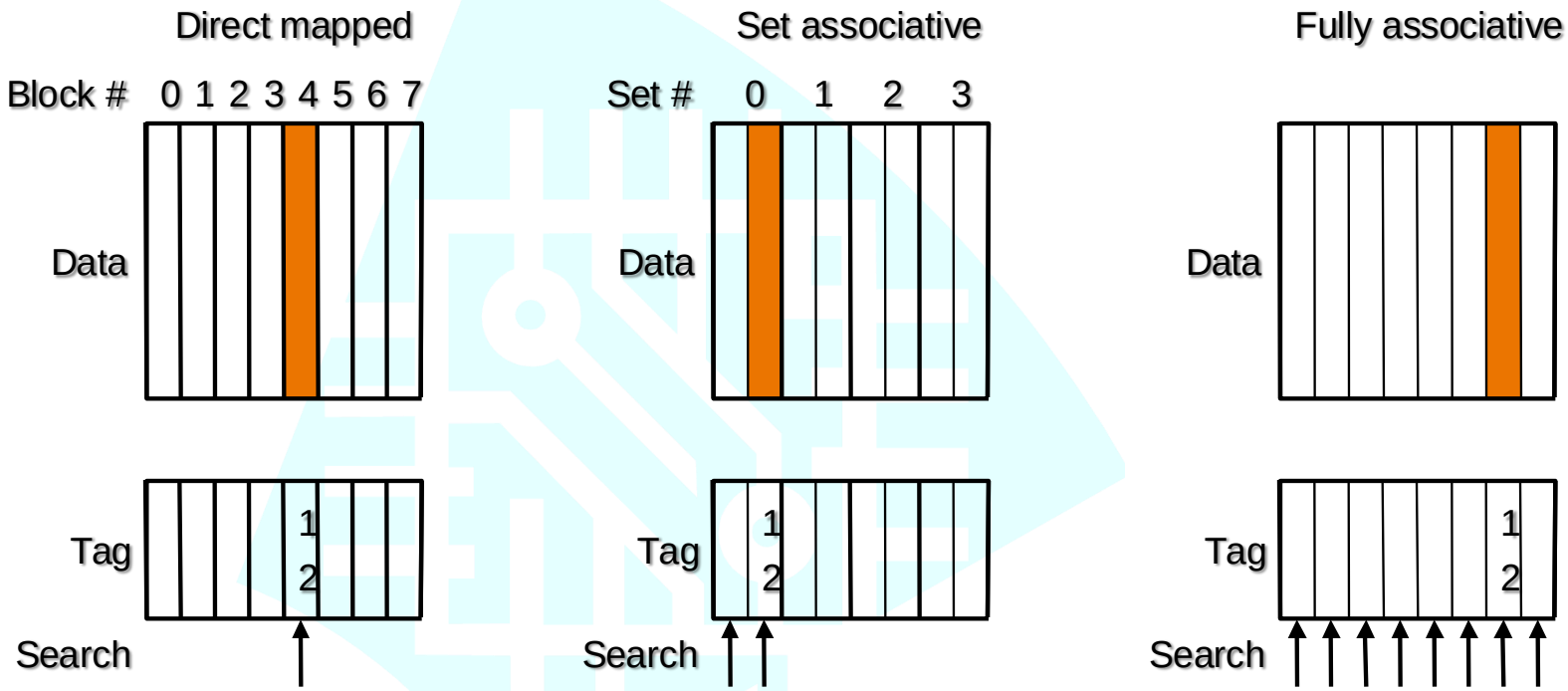
- ✓ Aumentar tamanho do bloco

Reduzindo faltas 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

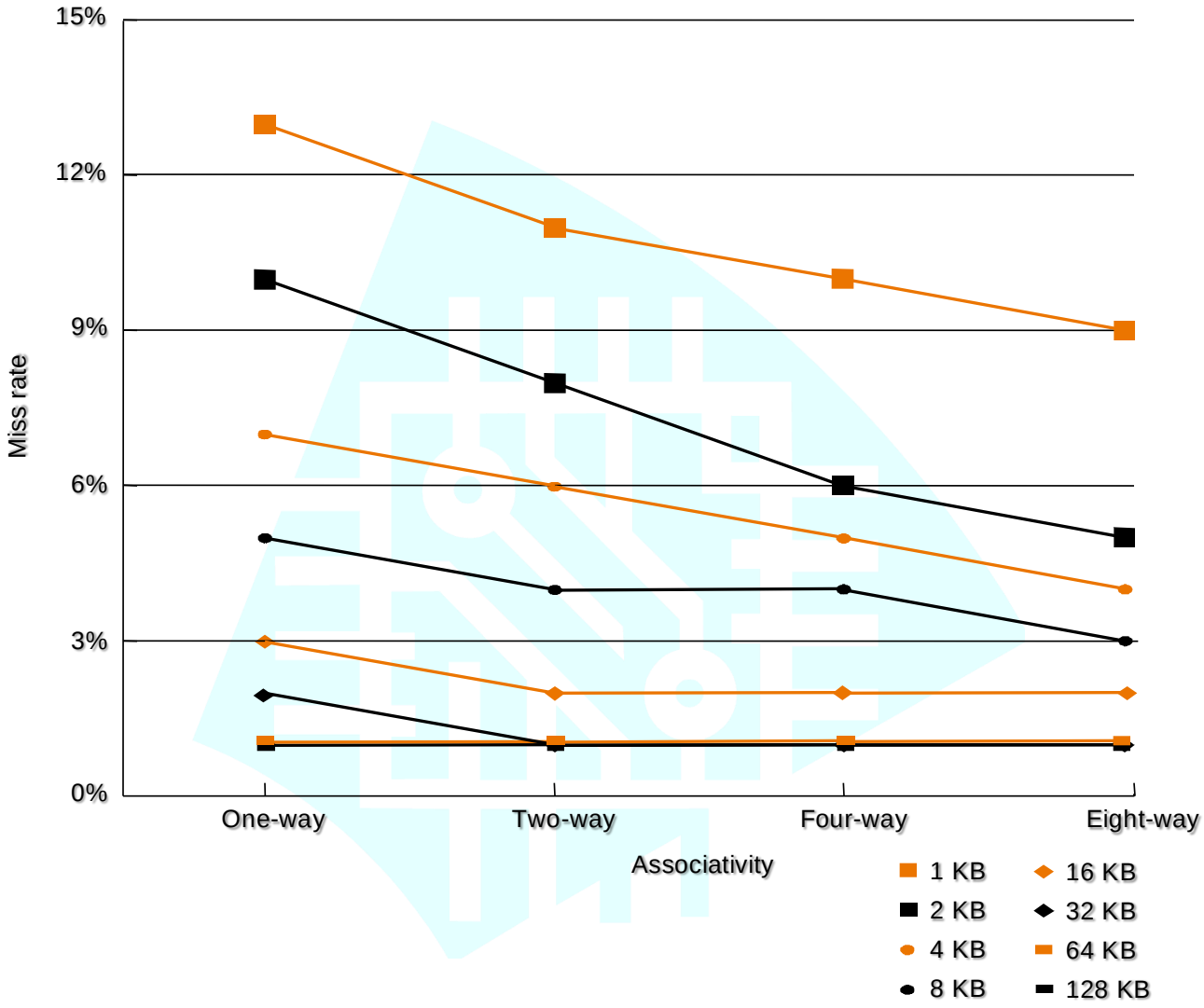
Aumento da Associatividade



Aumento da Associatividade

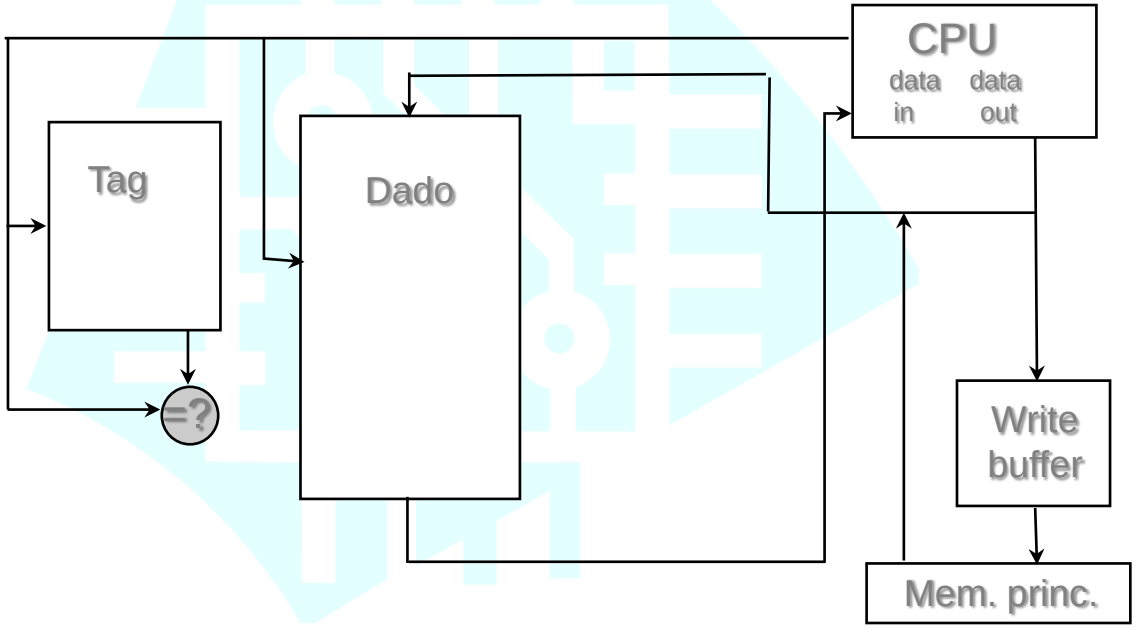
Programa	Associatidade	Miss instr.	Miss dado	Miss Total
Gcc	1	2.0%	1.7%	1.9%
Gcc	2	1.6%	1.4%	1.5%
Gcc	4	1.6%	1.4%	1.5%
Spice	1	0.3%	0.6%	0.4%
Spice	2	0.3%	0.6%	0.4%
Spice	4	0.3%	0.6%	0.4%

Aumento da Associatividade



Reduzindo penalidade de cache

- ✓ Write Buffers



Reduzindo penalidade

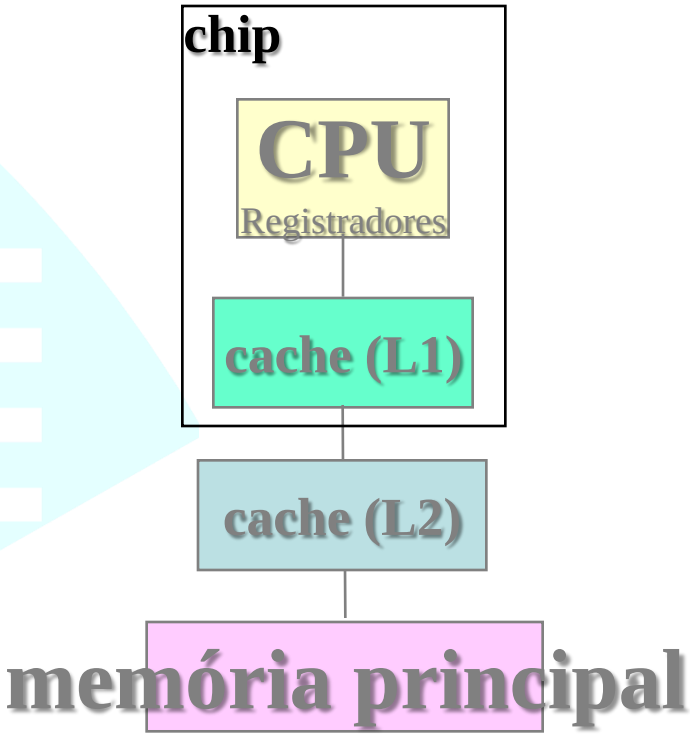
- ✓ Early Restart
  - Assim que palavra procurada foi carregada na cache esta é enviada para a CPU.

Reduzindo penalidade

- ✓ Critical Word First:
  - 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:
    - Tempo médio de acesso=hit_{L1}+$miss_{L1}$x pen_{L1}
    - Pen_{L1}=hit_{L2}+$miss_{L2}$ x 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, $Miss\text{-}rate_{mem}$=5%.
- ✓ Segundo nível: $Time_{L2}$=20ns, $Miss\text{-}rate_{mem}$=2%
- ✓ Qual o desempenho da máquina com 2. nível de cache?

Reduzindo a penalidade

- ✓ Qual o desempenho da máquina com 2. nível de cache?
  - $Penalty_{mem}$=200ns/2ns/clk = 100 ciclos
  - CPI_{total}= CPI_{base}+Mem_{ciclos}/I = 1.0 + 5%x100=6.0
  - $Penalty_{L2}$=20/2=10 ciclos
  - CPI_{total}=1+L1-stalls+L2stalls = 1+((5%-2%)x10)+(2%x(10+100))= 1+0.3+2.2=3.5
  - 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

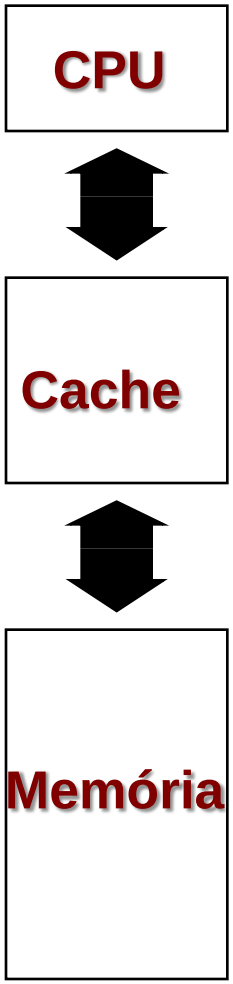
Reduzindo penalidade

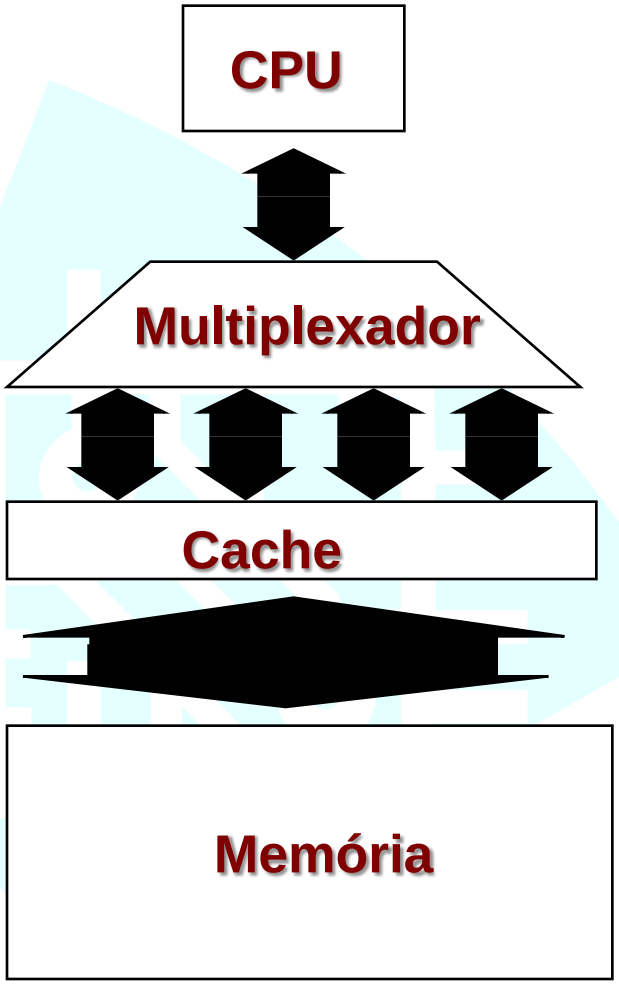
- ✓ Segundo nível de cache:
  - Redução da taxa de falta
    - cache do segundo nível maior que a do primeiro nível
  - E quanto a duplicação de dados nos dois níveis?
    - Os dados devem ser duplicados (consistência)

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

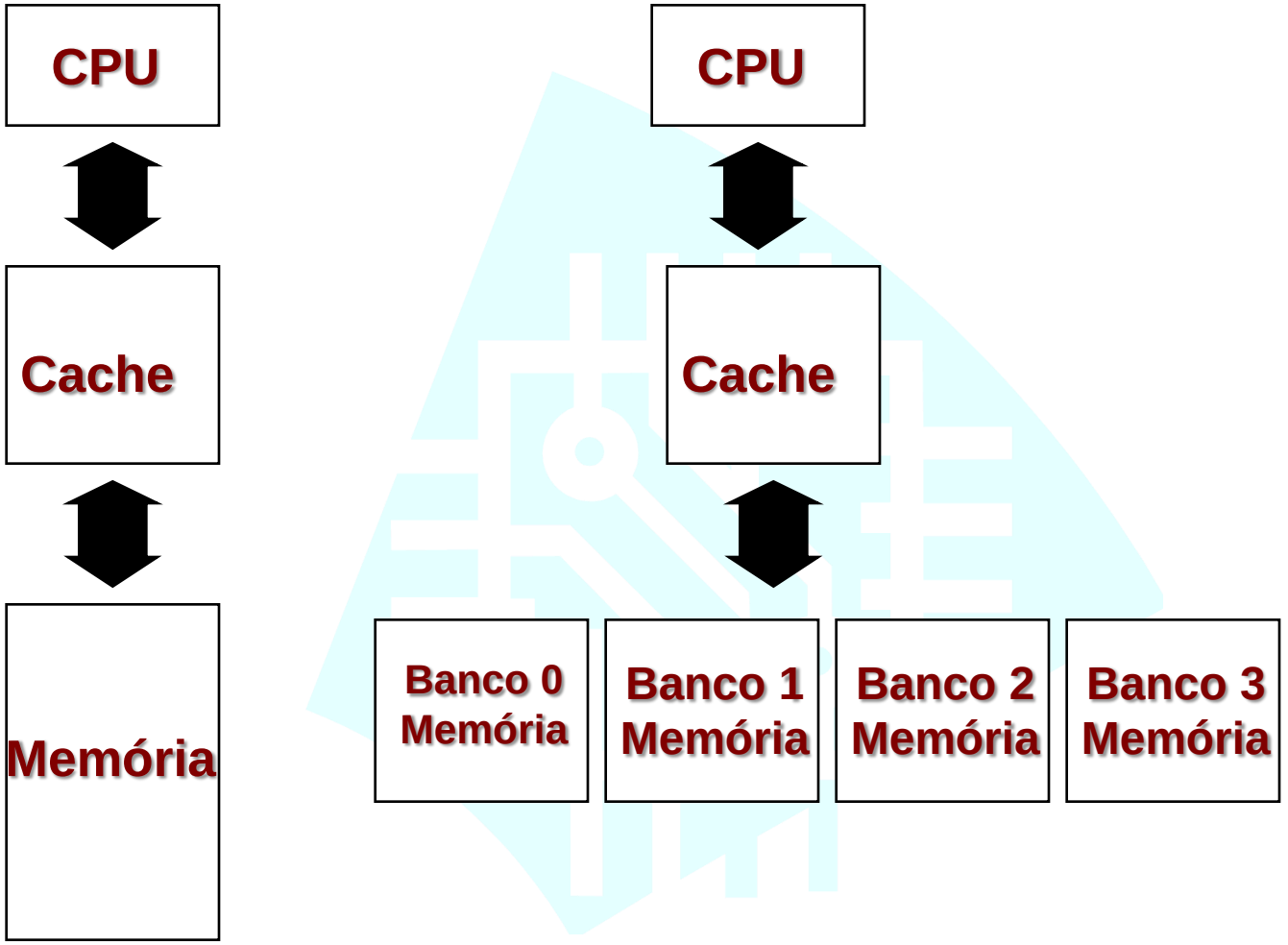




Memórias mais largas

- ✓ 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

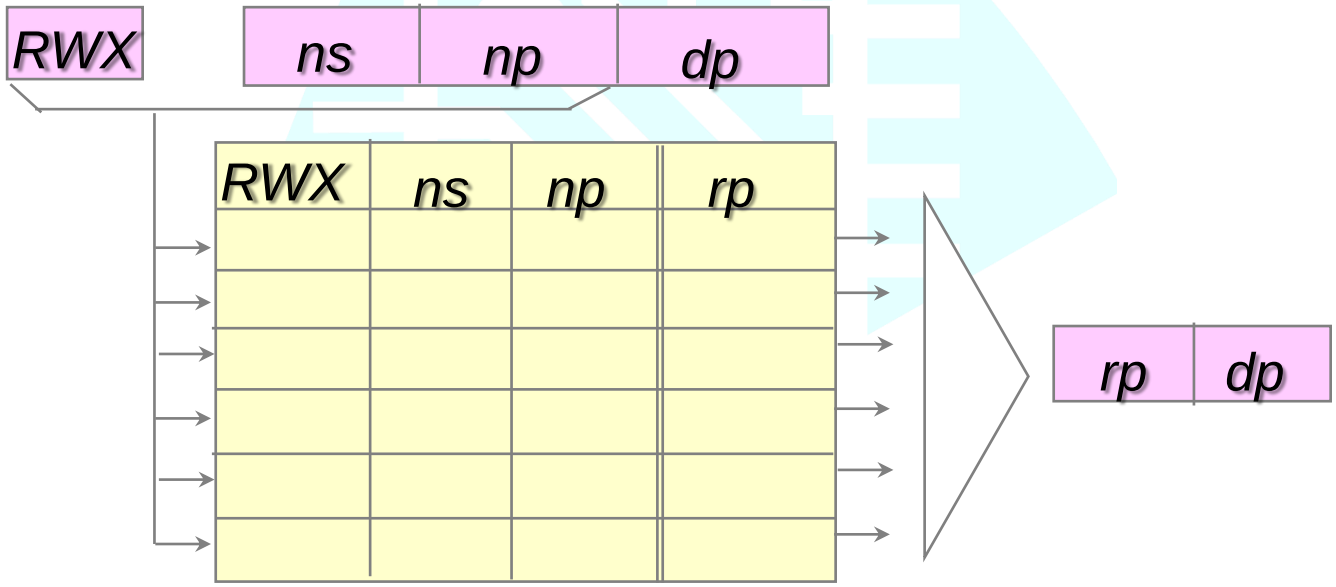
- CPI (# clocks por instrução)
  - 32 bits, sem interleaving=3.54
  - 32 bits, interleaving=2.86

TLB

- ✓ Tamanho
  - 32 a 4.096 entradas
- ✓ Tamanho entrada:
  - 1-2 entrada da tabela (4-8 bytes)
- ✓ Tempo de acesso:
  - 0.5 a 1 ciclo
- ✓ Taxa de faltas:
  - 0.01 - 1%

TLB (Translation lookaside buffers)

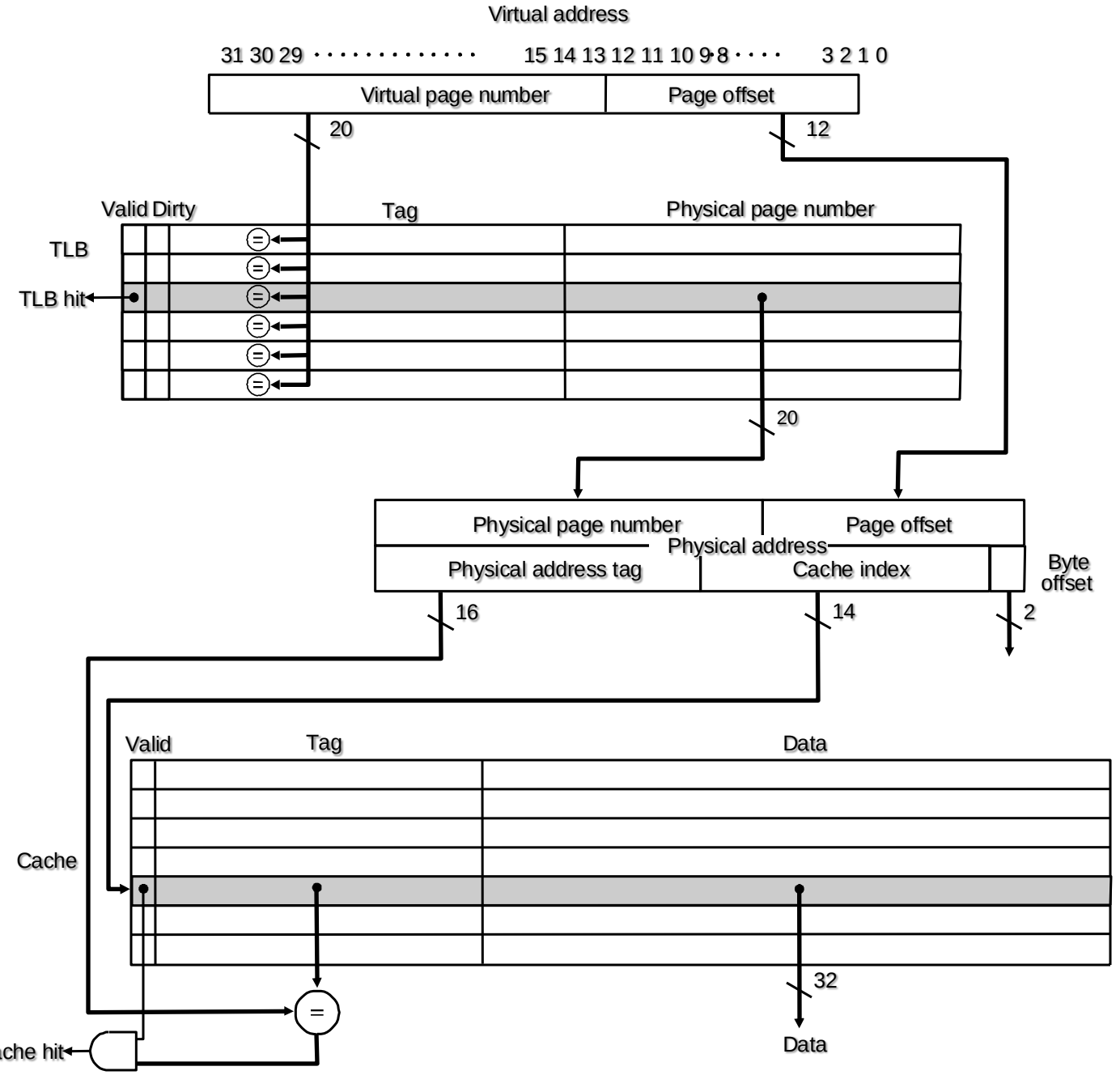
- ✓ Grau de Associatividade
- ✓ TLB pequena
  - cache associativa
  - Substituição randomica

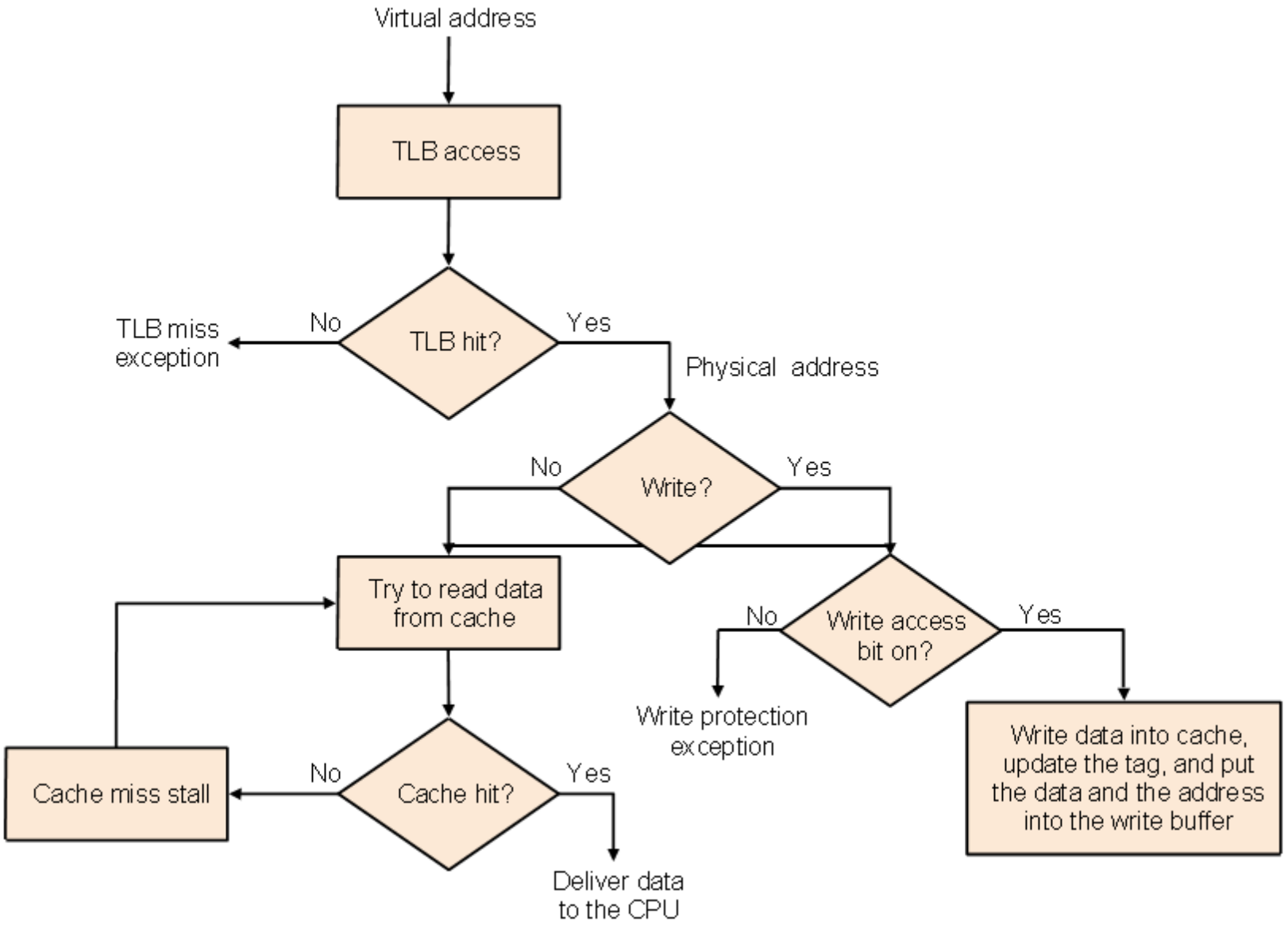


TLB - Exemplo

- ✓ Quantos bits possui a TLB para o sistema abaixo?
  - Endereço virtual: 32 bits
  - Memória física: 16 M bytes
  - Páginas de 4 Kbytes
  - TLB associativa com 64 entradas contendo 1 bit de validade e 1 dirty bit adicionais

MIPS


Virtual address
31 30 29 · · · · · · · · · · · · · 15 14 13 12 11 10 9 8 · · · · 3 2 1 0
Virtual page number
Page offset
20
12
Valid Dirty
Tag
Physical page number
TLB
TLB hit
20
Physical page number
Page offset
Physical address
Physical address tag
Cache index
Byte offset
16
14
2
Valid
Tag
Data
Cache
32
=
Cache hit
Data


Virtual address
TLB access
TLB hit?
No
TLB miss exception
Yes
Physical address
Write?
No
Yes
Try to read data from cache
Write access bit on?
No
Write protection exception
Yes
Write data into cache, update the tag, and put the data and the address into the write buffer
Cache hit?
No
Cache miss stall
Yes
Deliver data to the CPU

Sistema Hierárquico

- ✓ TLB, Memória e Cache

Cache	TLB	Tabela pag.	Possível? Como?
miss	hit	Hit	Sim – falta de cache
hit	miss	Hit	Sim – TLB substituida
miss	miss	Hit	Sim, TLB subst. e falta de cache
miss	miss	Miss	Sim
miss	hit	Miss	impossível
hit	hit	Miss	Impossível
hit	miss	Miss	impossível

Resumindo

- ✓ Onde um bloco/pag. pode ser colocado?
  - Cache:
    - mapeamento direto
    - associativa por conjunto
    - associativa
  - Memória virtual:
    - associativa

Resumindo

- ✓ Como um bloco/pag. pode ser localizado?
  - Cache:
    - mapeamento direto: índice
    - associativa por conjunto: índice + comparação
    - associativa: comparação de toda a cache
  - Memória virtual:
    - tabela de páginas/segmentos

Resumindo

- ✓ Qual o bloco deve ser substituído numa falta?
  - Cache:
    - LRU (grau de associatividade 2-4)
    - randomica
  - Memória virtual:
    - LRU (a penalidade é muito grande)

Resumindo

- ✓ Qual a estratégia de escrita?
  - Cache:
    - Write-through
    - Write-back
  - Memória virtual:
    - Write-back

Exercício

Considere um sistema de memória virtual com as seguintes características:

- Endereço virtual de 32 bits
- Páginas de 8 Kbytes
- 64 M Bytes de memória principal

(a)- Qual o layout do endereço virtual?

(b)- Qual o layout e o tamanho da tabela de páginas em bytes?

Assuma que cada página virtual possui um bit de validade, 2 bits de proteção e um "dirty bit".

Considere ainda que os endereços na memória secundária não estão armazenados nesta tabela.

Bibliografia

- ✓ *Sistemas Operacionais*, Oliveira R., Carissimi A. e Toscani S., Ed. Sagra-Luzzatto, 2004
- ✓ *Organização e Arquitetura de computadores; Henessy, J.L.; Patterson, D. A.;*
- ✓ *Organização de Computadores, Ricarte, I. L. M;*
- ✓ *Ambientes Operacionais; Toscani, S.;*
- ✓ *Sistemas Operacionais*, Oliveira R., Carissimi A. e Toscani S., Ed. Sagra-Luzzatto, 2004
- ✓ *Arquitetura não convencional de computadores; Slides; Barros, E. N. S;*