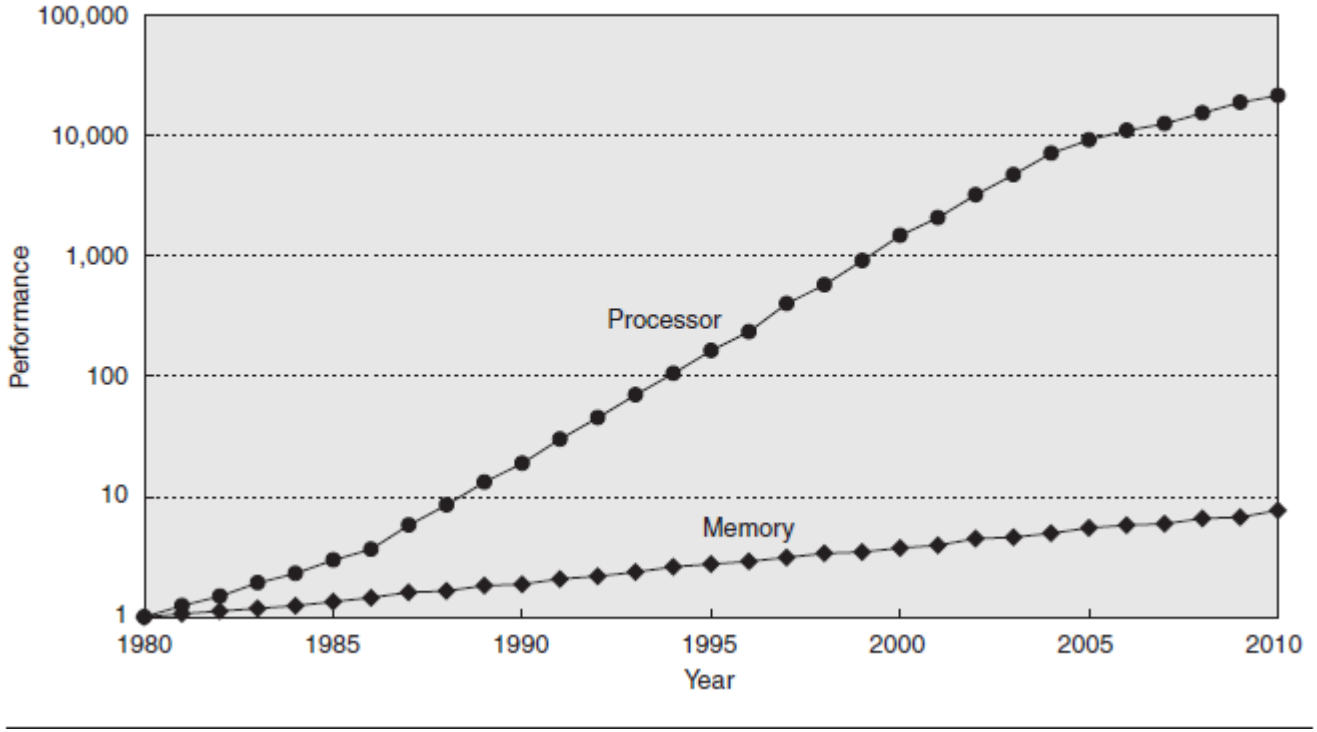
Organização de Computadores II
DCC007

Aula 13 – Revisão de Memória

Prof. Omar Paranaiba Vilela Neto

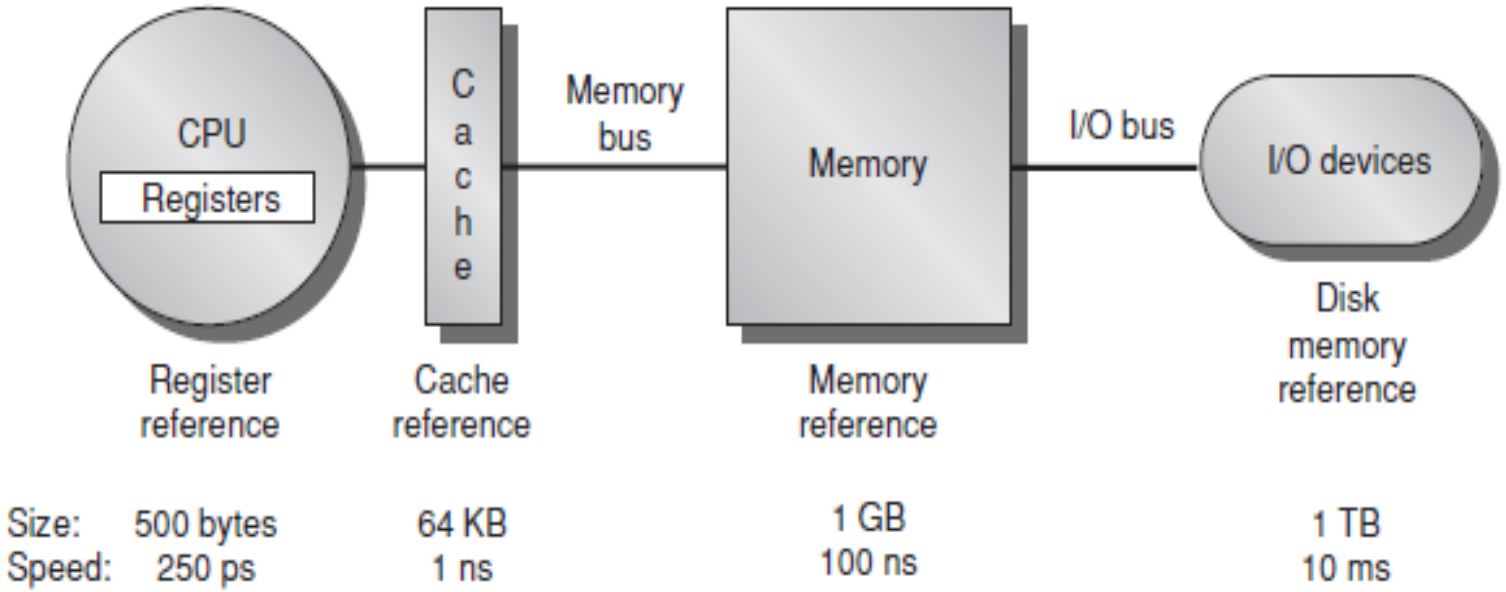
Por que Hierarquia de Memória?



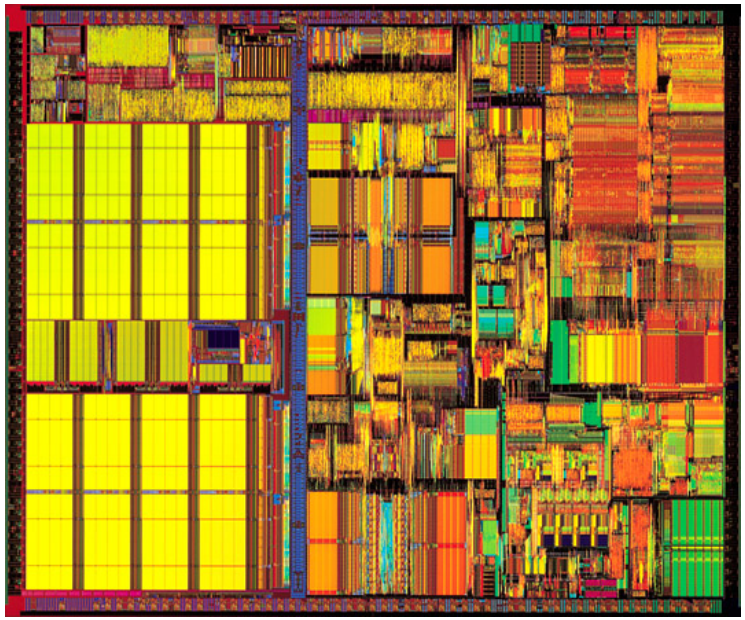
Por que se Preocupar com a Hierarquia de Memória?

- 1985: não haviam caches em microprocessadores (i386)
- 1990: caches de dois níveis
  (> 10x em diferença de desempenho)

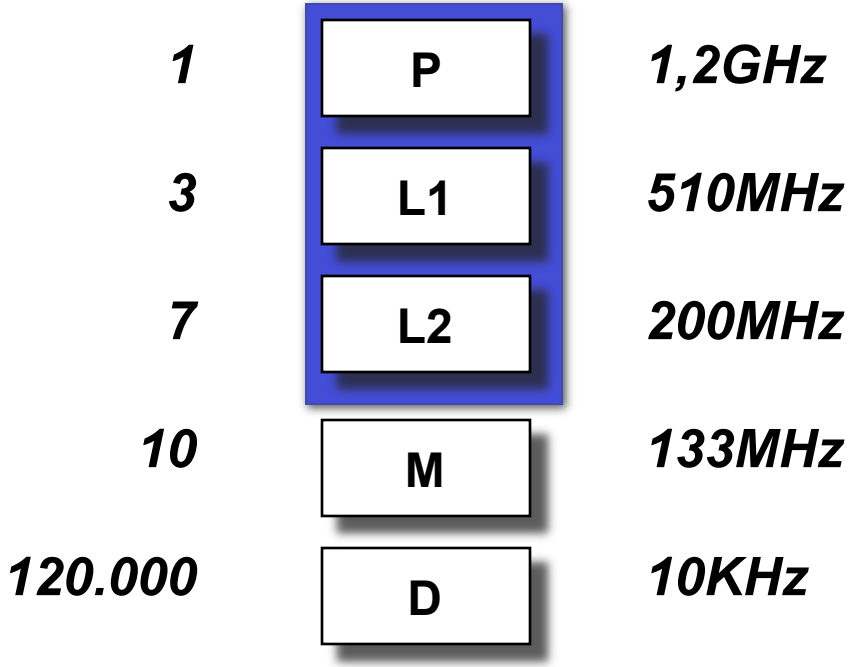
Hierarquia de Memória



Princípios Gerais de Hierarquia de Memória



PentiumIII



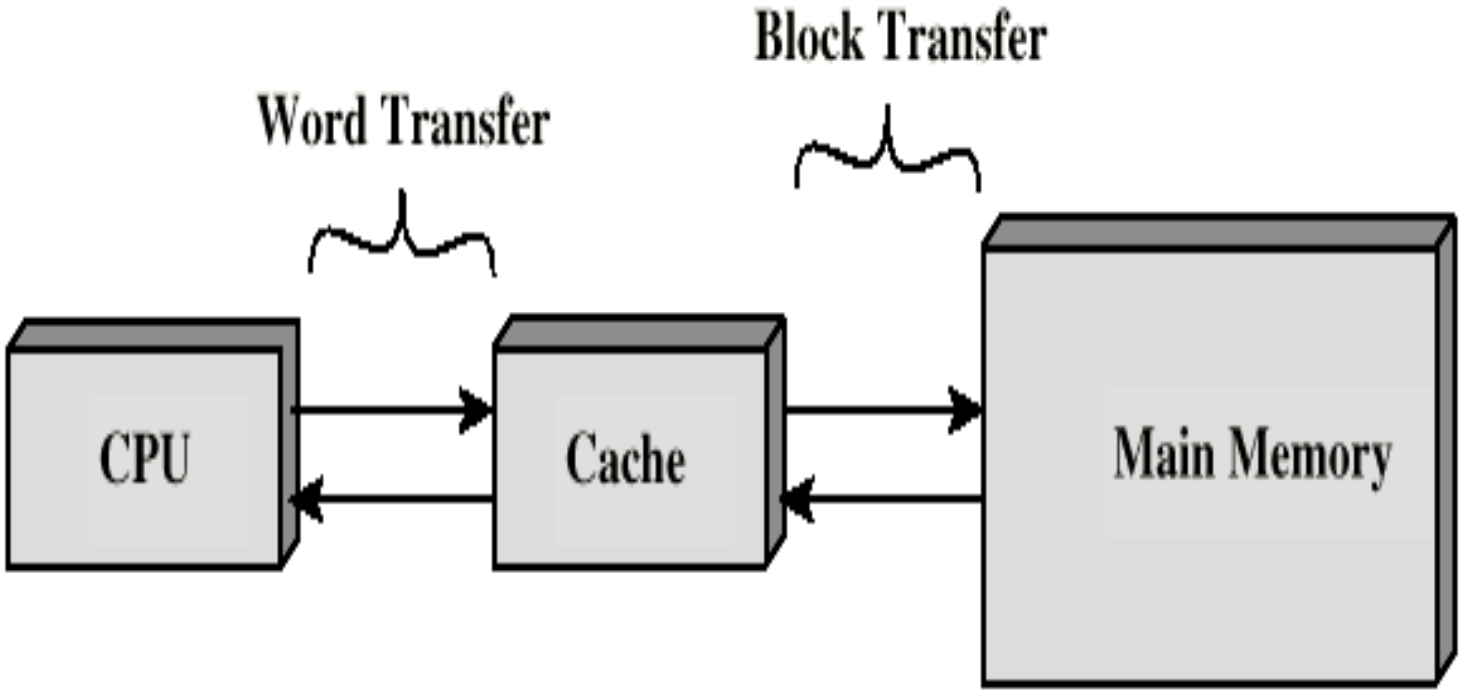
Caches

...a safe place for hiding or storing things...

Princípios Gerais de Caches

- Localidade
  - Localidade temporal: dados serão referenciados novamente
  - Localidade espacial: itens serão referenciados na vizinhança
- Localidade + HW menor é mais rápido == hierarquia de memória
  - Níveis: cada menor e mais rápida é mais cara que a de nível mais baixo
  - Inclusive: dado encontrado no topo também é encontrado no nível mais baixo
- Definições
  - Nível mais alto é mais próximo ao processador
  - Bloco: unidade mínima de dados (também conhecido como linha)
  - Endereço = endereço do bloco + offset dentro do bloco
  - Hit time: Tempo para acessar dado no nível mais alto

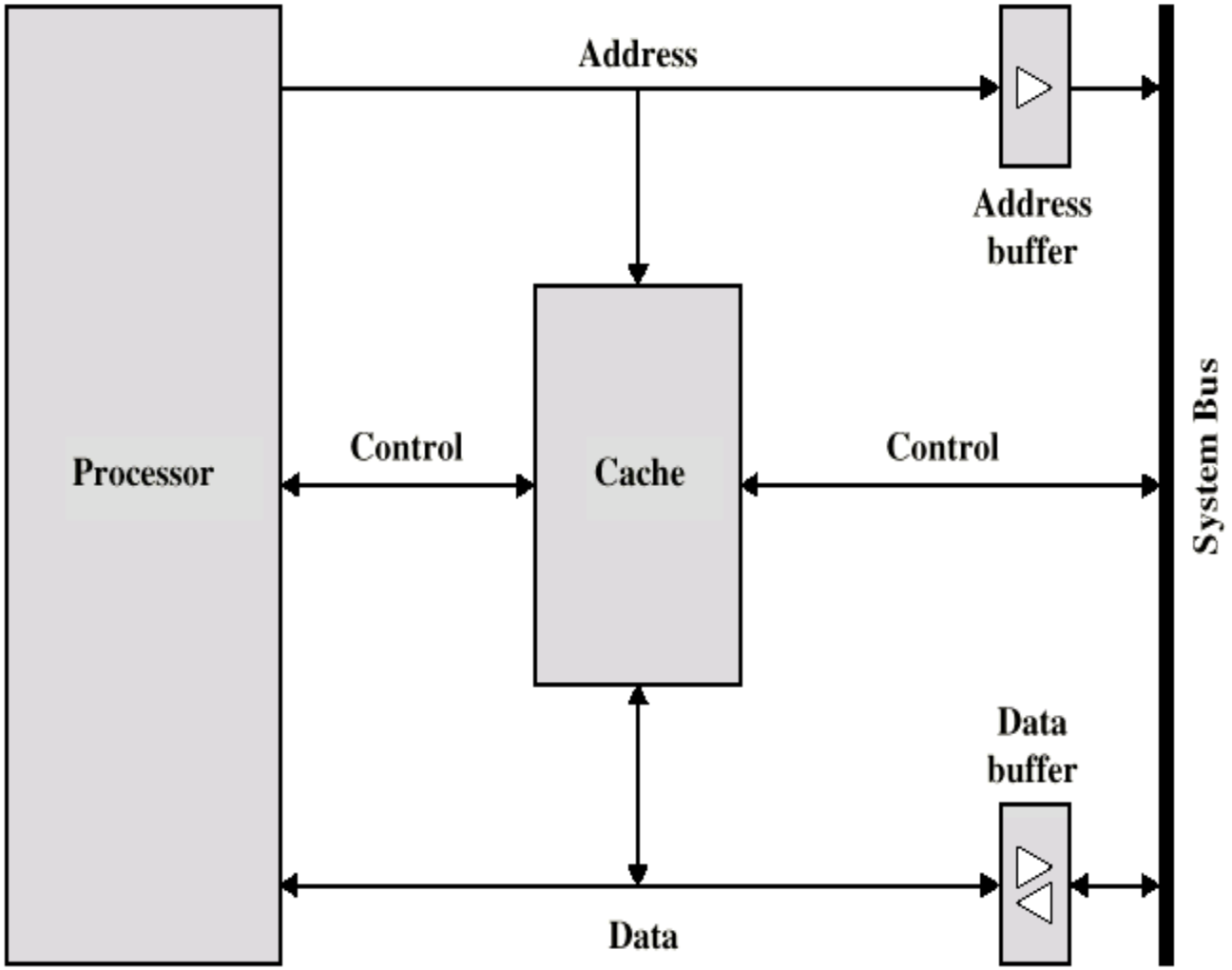
Transferências de Dados em Caches



Operação de Caches

- Processador envia requisição com endereço de memória
- Cache checa se endereço está presente na cache
- Se estiver, cache recupera dado rapidamente
- Se não estiver presente, cache inicia leitura de bloco do nível mais baixo
- Cache entrega dado ao processador
- Cache faz atualização de informação para *book keeping*

Organização de Caches



Medições em Caches

- *Hit rate*: Taxa de acerto no nível
  - Tão alta que as vezes usamos *Miss rate*
- *Miss penalty*: tempo requerido para trocar um bloco vindo de um nível mais baixo, incluindo tempo até carregar dado na CPU
  - *tempo de acesso*: tempo para nível mais baixo = f(latência do nível mais baixo)
  - *tempo de transferência*: tempo para transferir bloco = f(BW nível + alto e + baixo, tamanho do bloco)
- Tempo médio de acesso à memória (AMAT) = Hit time + Miss rate x Miss penalty (ns ou clocks)

Organização de Caches

Mapeamento Direto

- Cada bloco da memória mapeia para somente um bloco da cache

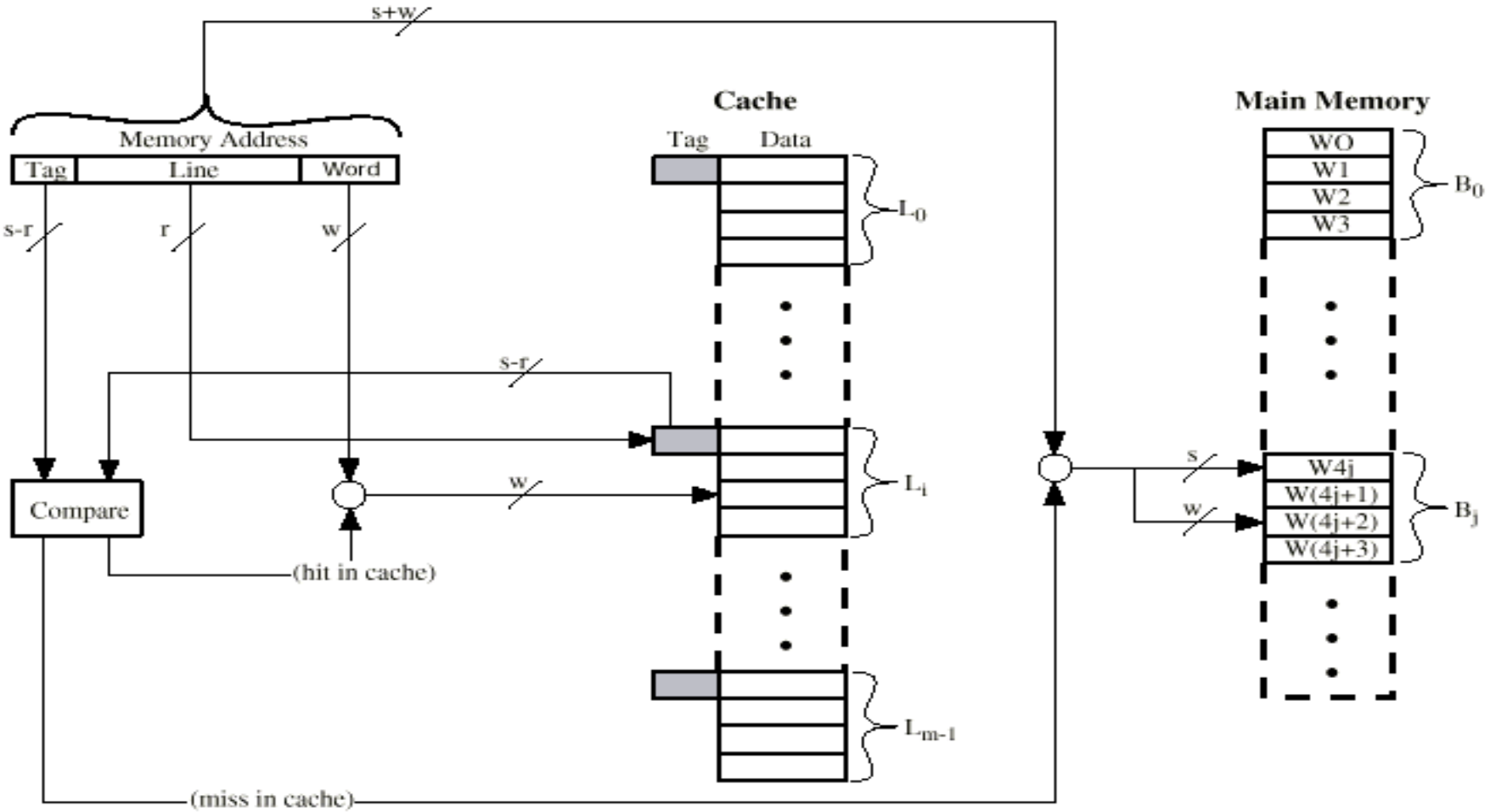
- Endereço é dividido em duas partes
  - w identifica qual palavra no bloco é acessada
  - s identifica qual bloco está sendo procurado
    - r especifica qual linha
    - s-r especifica tag

Mapeamento Direto Exemplo

Tag s-r	Line or Slot r	Word w
8	14	2

- 24 bits de endereço / 16 K blocos
- Bloco com 4 bytes, endereçável a nível de bytes
- Identificador de bloco: 22 bits
  - 14 identificador de bloco
  - 8 bit tag (=22-14)
- Dois blocos idênticos não possuem o mesmo tag

Mapeamento Direto Acesso



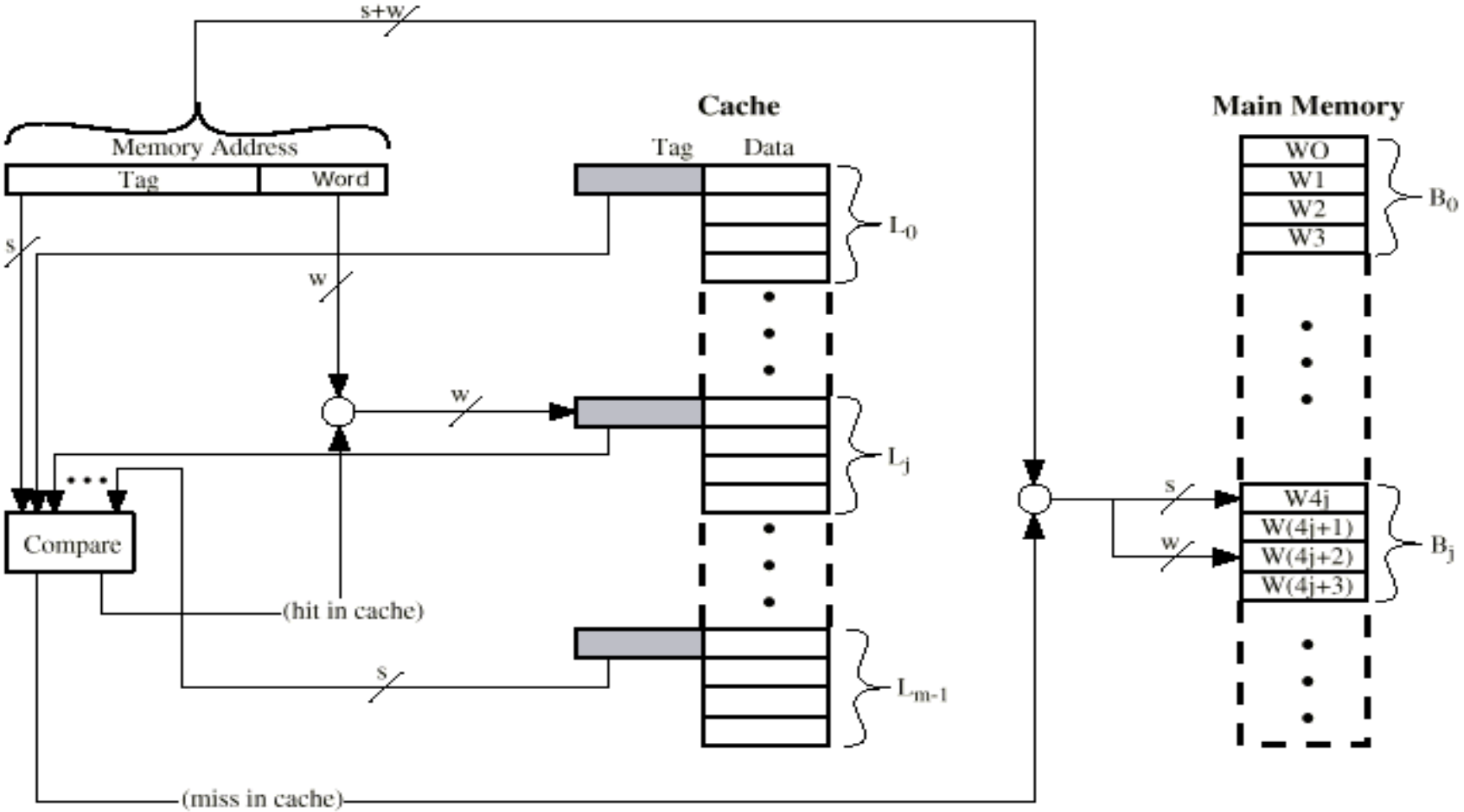
Mapeamento Direto pros & cons

- Fácil de implementar
  - Barata
  - Rápida
- Pode gerar muitos *misses*

Cache com Associatividade Completa

- Bloco da memória principal pode ir para qualquer bloco de memória da cache
- Endereço da memória inteiro é considerado tag
- Tag identifica univocamente qual bloco é desejado
- Todos os blocos são comparados para obter um *match*
- Pesquisa é **cara**

Cache com Associatividade Completa



Associatividade Completa Endereçamento

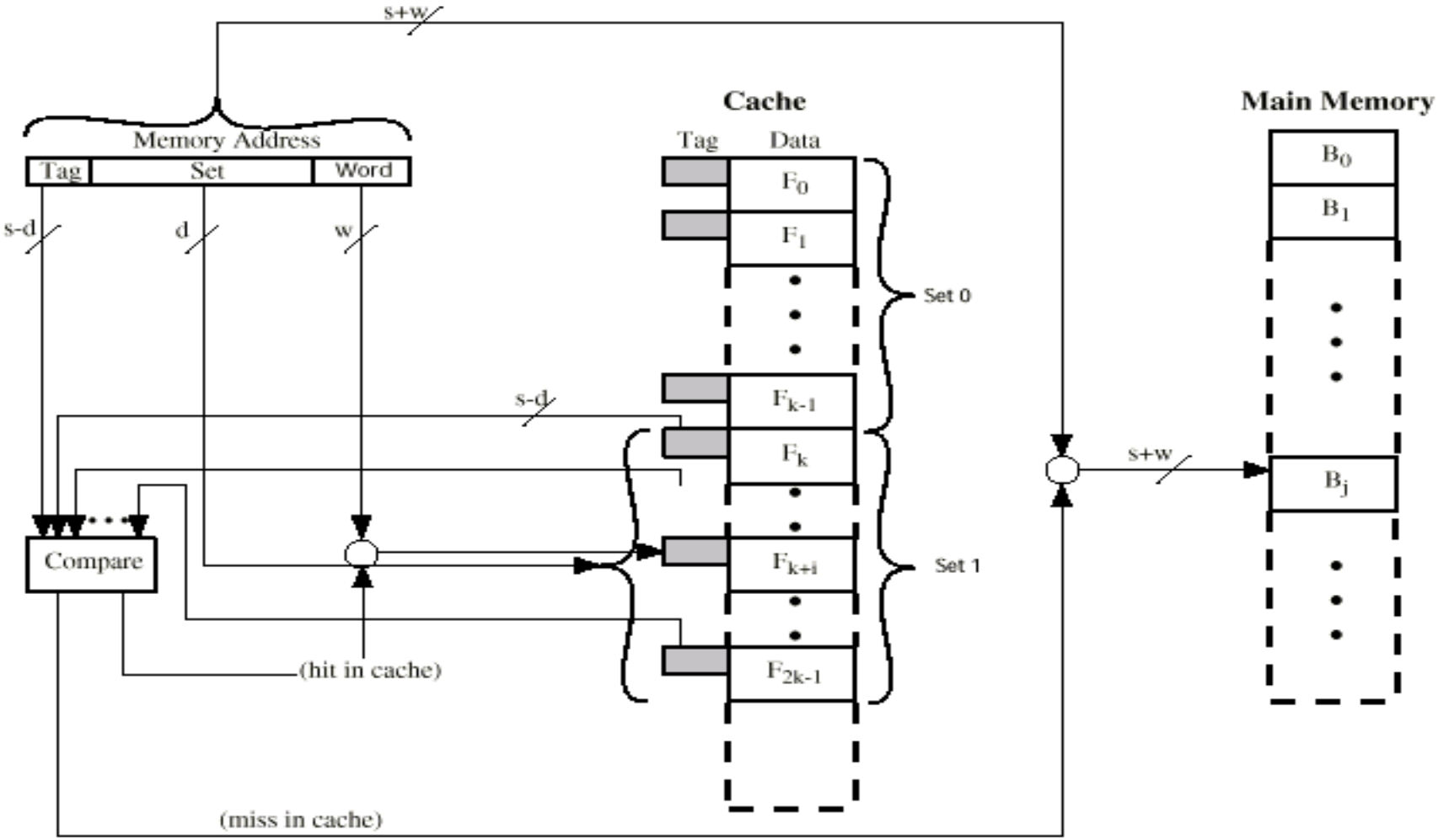
Tag 22 bit	Word 2 bit

- 24 bits de endereço / 16 K blocos
- Bloco com 4 bytes, endereçável a nível de bytes
- 22 bits de tag armazenados com 32 bits de dados

Associatividade por Conjunto

- Cache é dividida em um número de conjuntos
- Cada conjunto contém um número de blocos
- Um bloco da memória principal mapeia em qualquer bloco do conjunto
  - Bloco B pode estar em qualquer linha do conjunto i
- e.g. 2 linhas por conjunto
  - *2 way associative mapping*

K-way Set Associative Cache



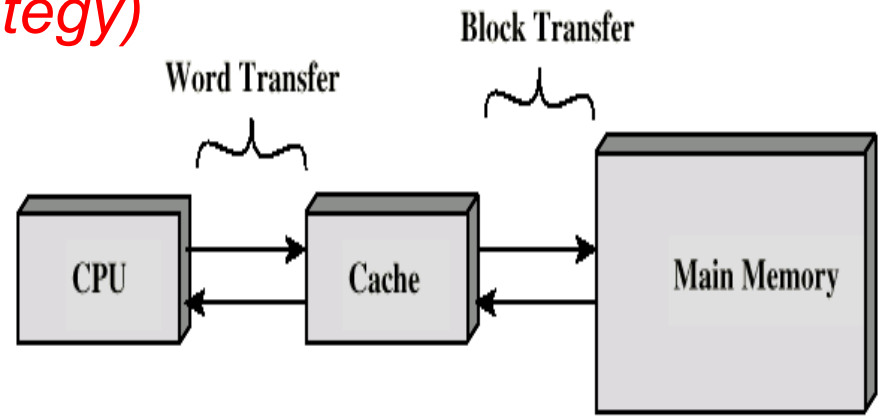
Associatividade por Conjunto Endereçamento

Tag 9 bit	Set 13 bit	Word 2 bit

- 24 bits de endereço / 16 K blocos
- Bloco com 4 bytes, endereçável a nível de bytes
- 2-way set associative

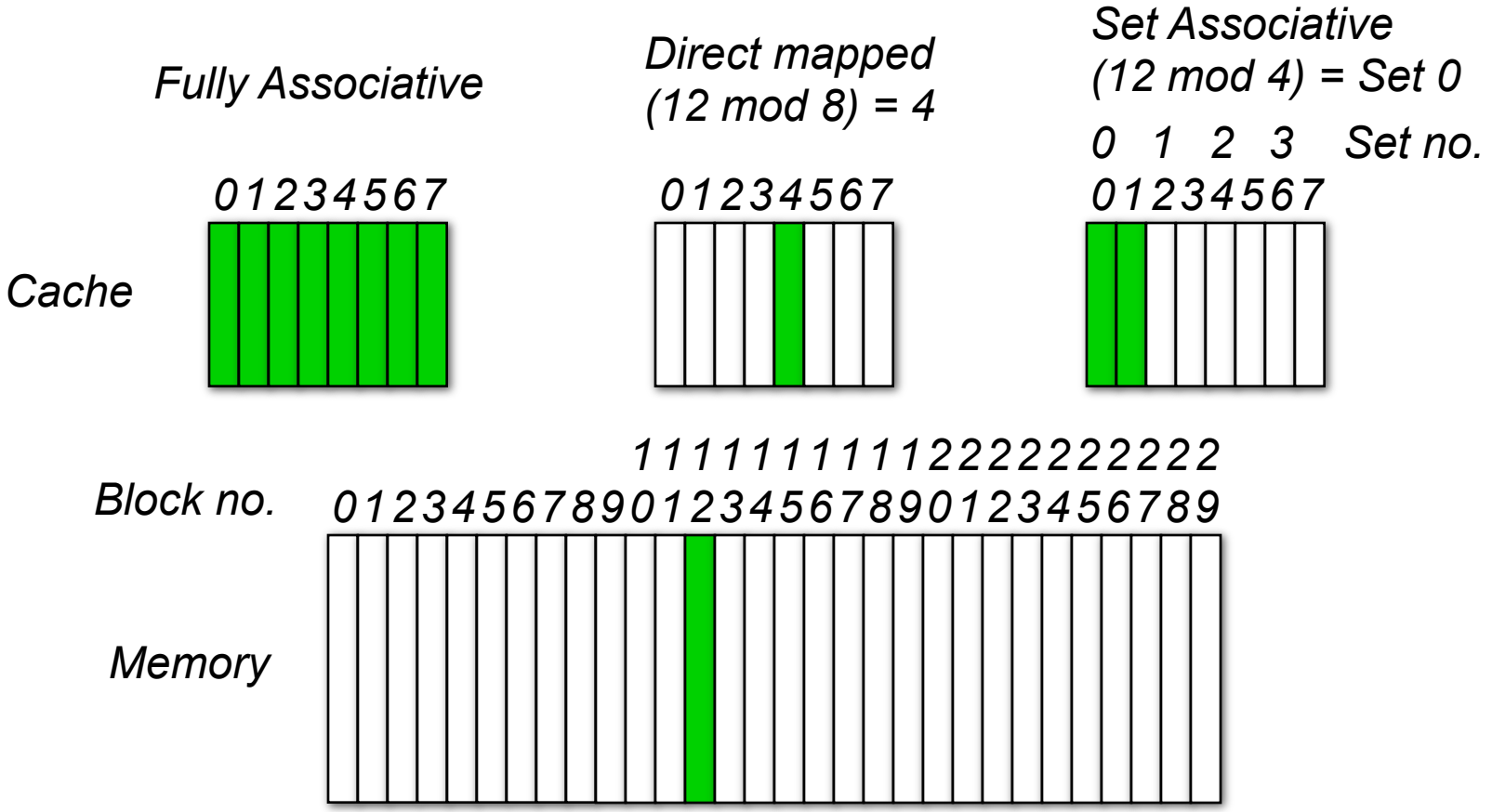
Quatro Perguntas Básicas sobre Hierarquia de Memória

- Q1: Onde o bloco vai ser colocado na memória de nível mais alto? *(Block placement)*
- Q2: Como bloco é encontrado na memória de nível mais alto?
  (Block identification)
- Q3: Quais blocos serão trocados em um miss?
  (Block replacement)
- Q4: O que acontece em uma escrita?
  (Write strategy)



Q1: Onde Bloco Deve Ser Colocado no Nível Mais Alto?

Onde bloco 12 deve ser colocado?



Q2: Como o Bloco é Encontrado no Nível Mais Alto?

- Tag para cada bloco
  - Não é necessário checar índice ou offset

- Aumento de associatividade reduz índice, aumenta tag

Q3: Qual Bloco Será Trocado Durante um Miss?

- Fácil de decidir para caches de mapeamento direto
- Quando empregar cada técnica?
  - Aleatório (associatividade alta)
  - LRU (associatividade baixa)
  - FIFO (para caches menores)

Q3: Qual Bloco Será Trocado Durante um Miss?

Falhas na cache por 1000 instruções

Size	Associativity								
	2-way			4-way			8-way		
	LRU	Random	FIFO	LRU	Random	FIFO	LRU	Random	FIFO
16KB	114.1	117.3	115.5	111.7	115.1	113.3	109.0	111.8	110.4
64KB	103.4	104.3	103.9	102.4	102.3	103.1	99.7	100.5	100.3
256KB	92.2	92.1	92.5	92.1	92.1	92.5	92.1	92.1	92.5

Q4: O Que Acontece Durante uma Escrita?

- Write through: A informação é escrita tanto para o bloco da cache quanto para a memória de nível mais baixo.
- Write back: A informação é escrita somente para o bloco da cache. Este bloco é escrito na memória quando ele for trocado.
  - Precisa de dirty bit
- Vantagens e Desvantagens:
  - WT: miss de leitura não resulta em escrita na memória
  - WB: reduz BW para memória de nível mais baixo
- WT está sempre combinada com write buffers de forma a não precisarmos esperar pelo nível mais baixo de memória

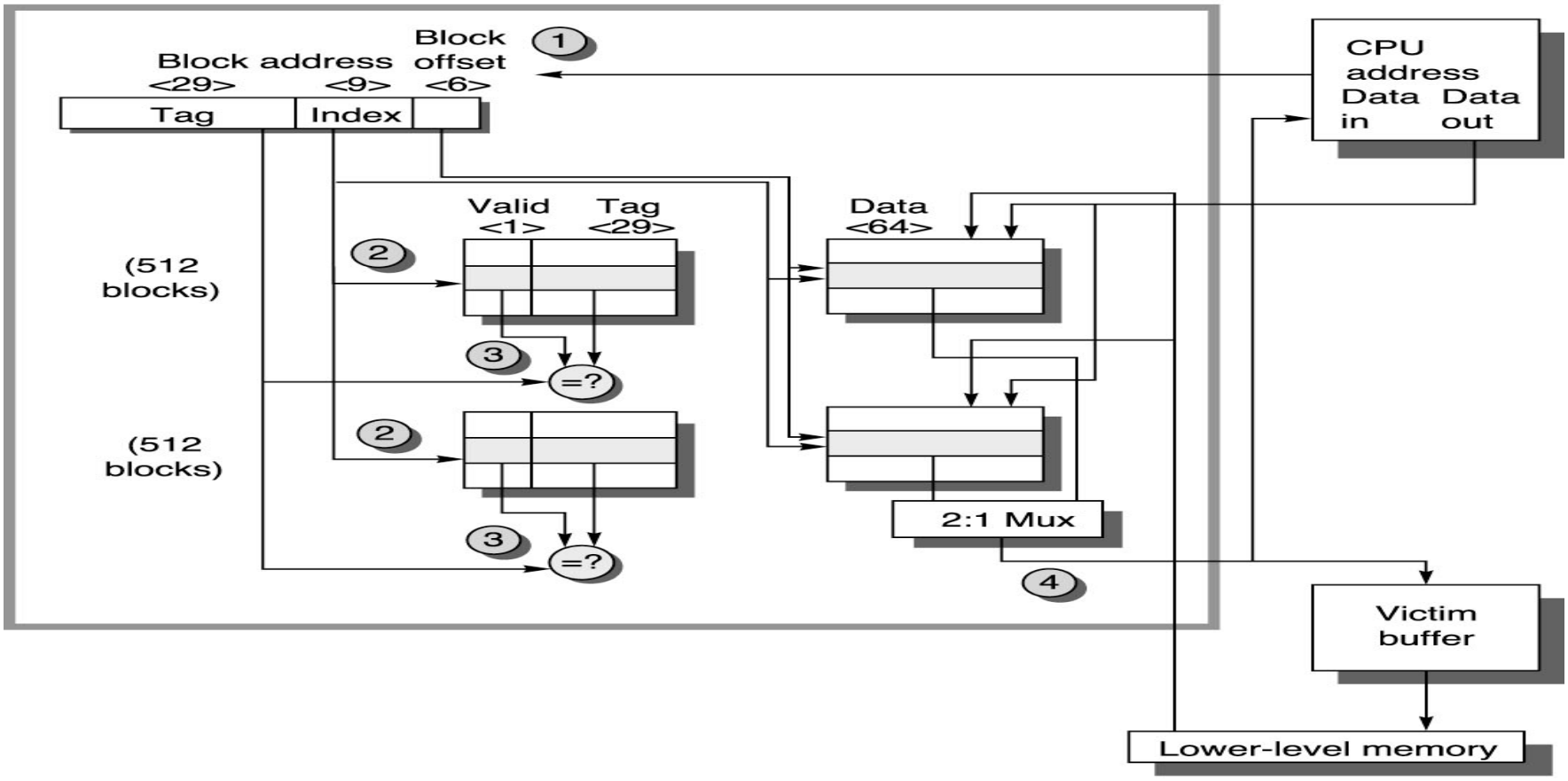
Q4: O Que Acontece Durante uma Escrita?

- Write allocate (fetch on write) : bloco é trazido para cache se ocorrer um miss de escrita, seguido por uma escrita com hit.

- No-write allocate (write around): bloco é modificado no nível mais baixo e só depois carregado na cache.

- WB é geralmente utilizado com write allocate
- WT é geralmente utilizado com no-write allocate

Cache de Dados do Alpha 21264



© 2003 Elsevier Science (USA). All rights reserved.

Resumo

- Gap entre CPU e memória é um dos maiores obstáculos para o desempenho de sistemas de computação
- Para melhorarmos a relação custo/ benefício, utilizamos o princípio da localidade
- Para qualquer nível de memória, queremos saber a resposta para os 4 Qs