Memória Cache

Arquitetura e Organização de Computadores

Prof. Lucas de Oliveira Teixeira

UEM

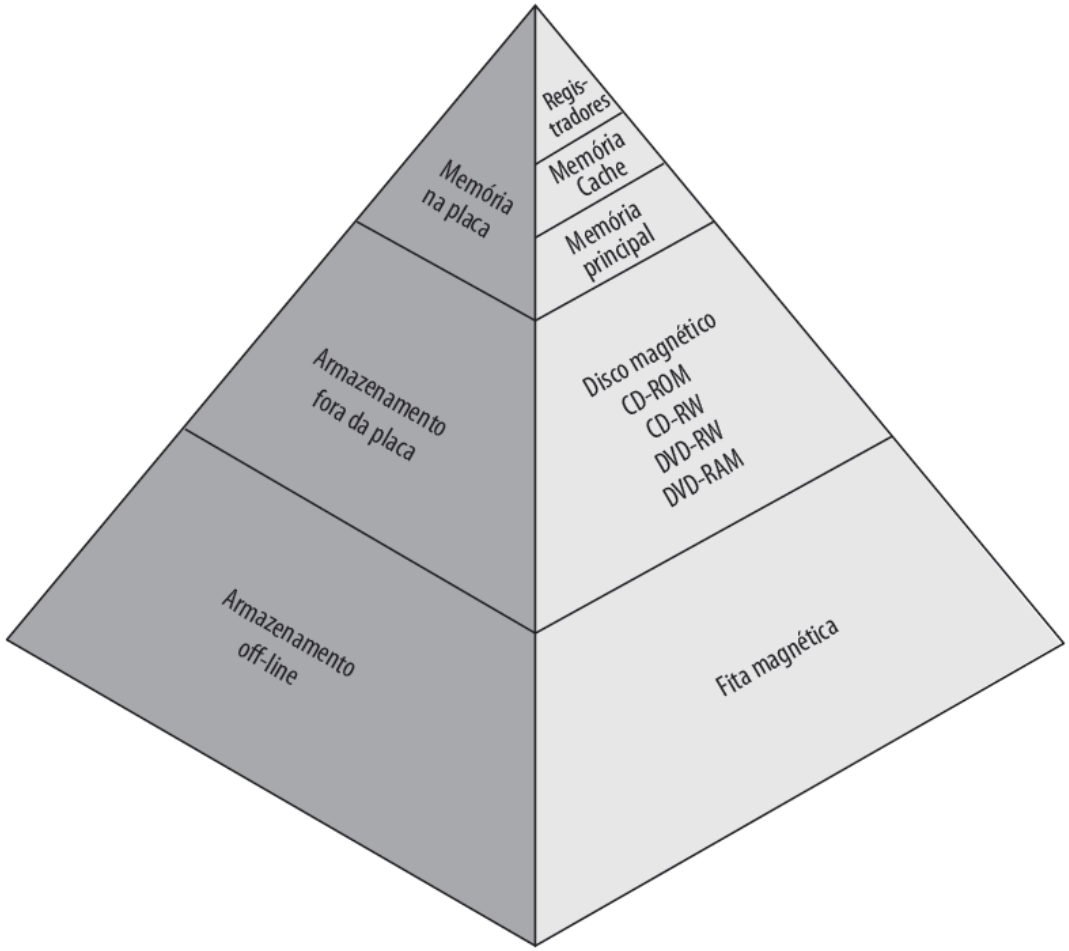
Introdução

Introdução

- A memória do computador é organizada segundo uma hierarquia.
- O desafio do projeto é organizar os dados e os programas na memória de modo que as palavras de memória mais acessadas normalmente estejam na memória mais rápida.

Hierarquia de memória:

Figura 4.1 A hierarquia de memória



Introdução

Hierarquia de memória:

- A medida que se desce na hierarquia temos:
  - Diminuição do custo por bit.
  - Aumento da capacidade.
  - Aumento no tempo de acesso.

Memória cache

Memória cache

- A cache retém automaticamente uma cópia de algumas das palavras usadas recentemente.
- Em geral, é provável que a maioria dos acessos futuros à memória principal, feitos pelo processador, seja para locais acessados recentemente.
- Se a cache for projetada corretamente, então, na maior parte do tempo o processador solicitará palavras da memória que já estão na cache.

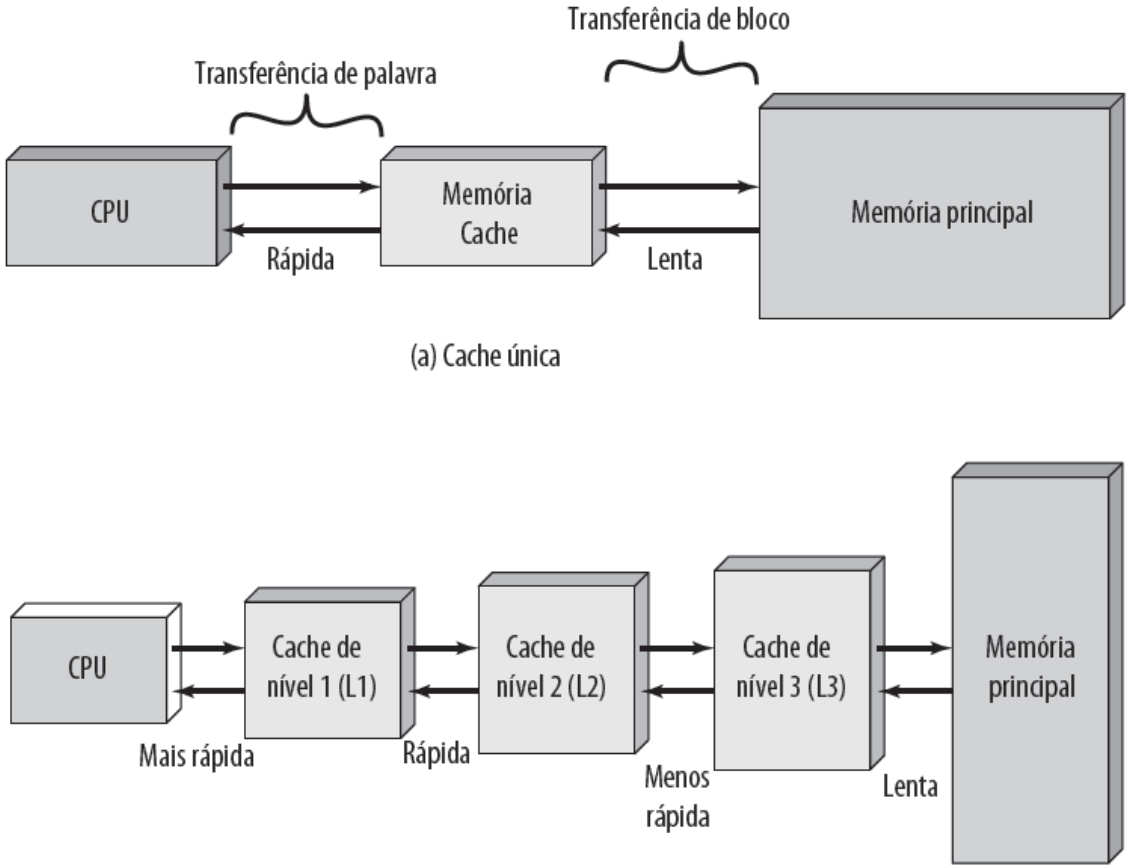
Memória cache

Princípio:

- Pequena quantidade de memória rápida.
- Fica entre a memória principal normal e a CPU.
- Atualmente, está localizada no chip da CPU.
- Cada linha da cache possui múltiplas palavras da memória principal (princípio da localidade).

Memória cache

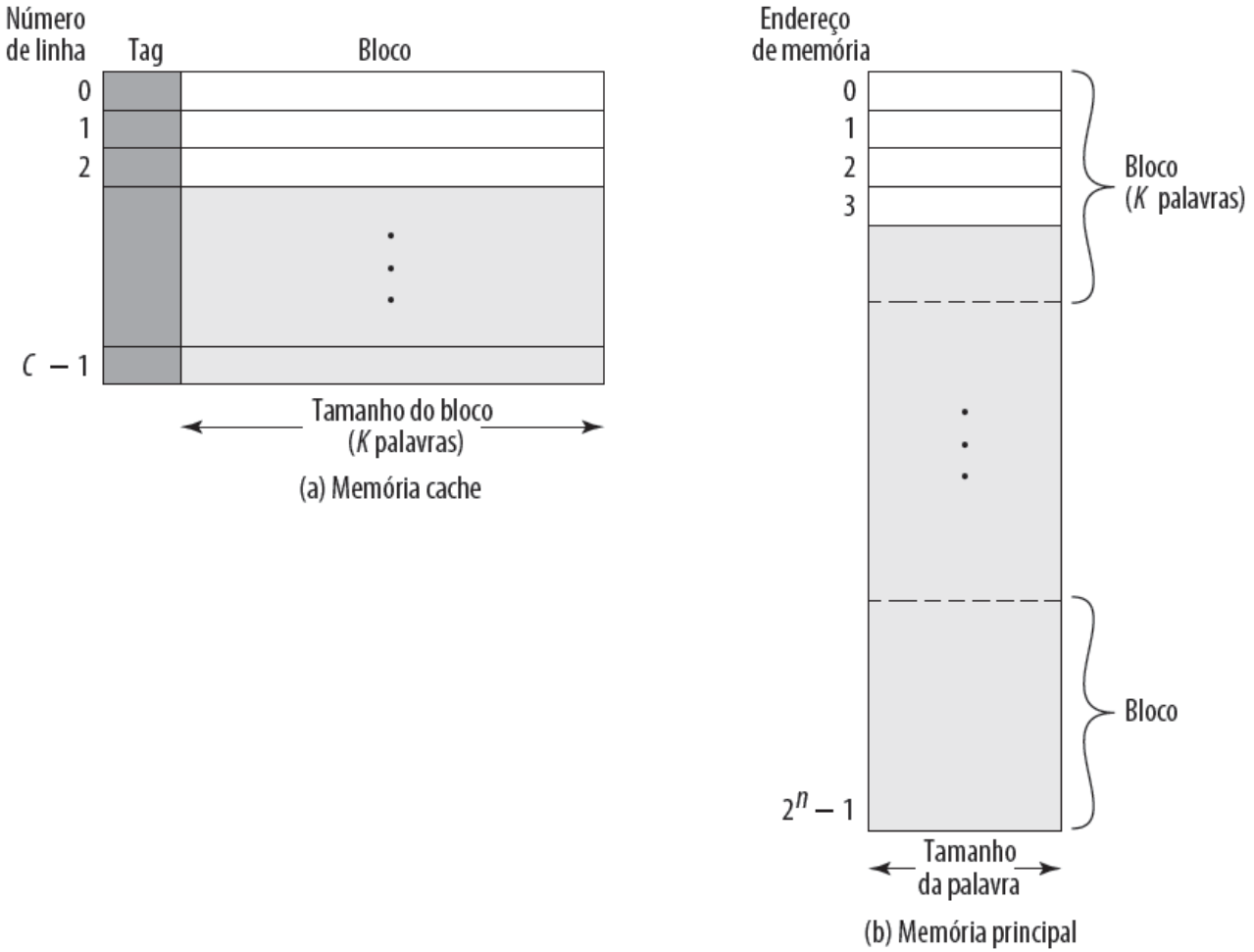
Figura 4.3 Cache e memória principal



Memória cache

Estrutura:

Figura 4.4 Estrutura de cache/memória principal



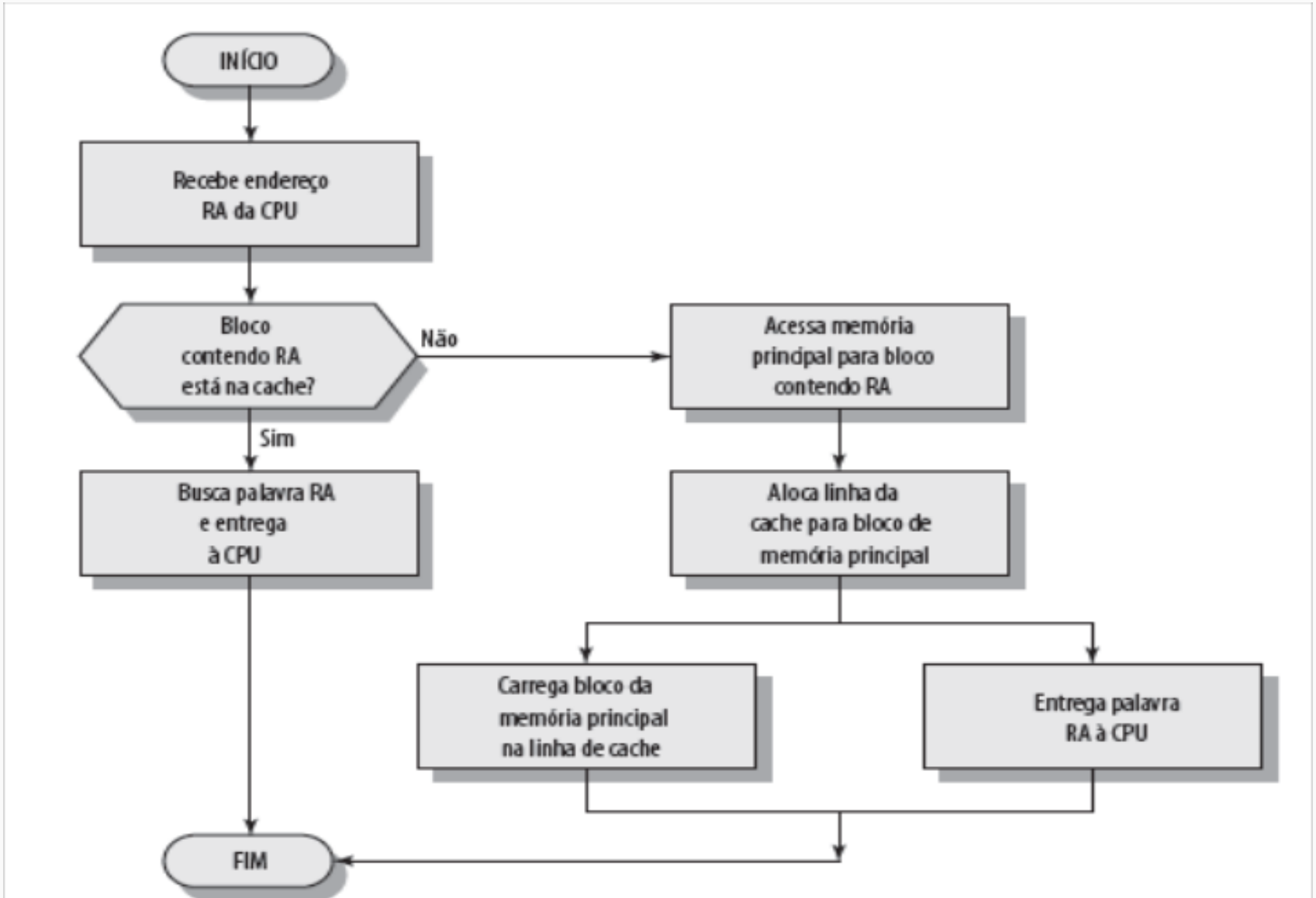
Memória cache

Operação:

- CPU requisita conteúdo do local de memória.
- Verifica se os dados estão em cache.
- Se estiverem, pega da cache (rápido).
- Se não, lê bloco solicitado da memória principal para a cache.
- Depois, entrega da cache à CPU.

Memória cache

Operação:



Memória cache

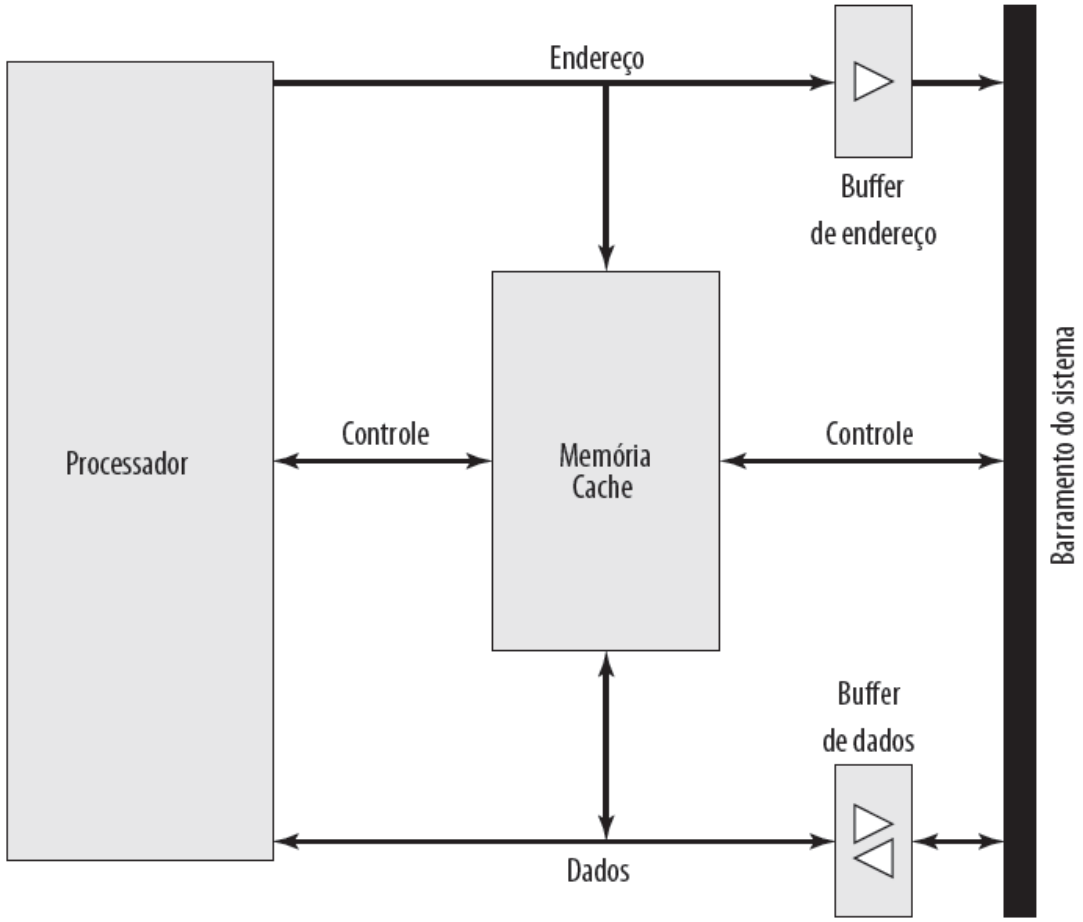
Operação:

- Cache hit: Os dados estão na cache, ou seja, ocorreu um acerto do endereço na cache.
- Cache miss: Os dados não estão na cache, ou seja, ocorreu um erro do endereço na cache.

Memória cache

Organização:

Figura 4.6 Organização típica da memória cache



Memória cache

Operação:

- A operação da cache ocorre em paralelo com a memória principal devido a organização do barramento interno.
- Em um cache hit, os barramentos de dados e endereço são desativados e a comunicação é direta com a cache.
- Em um cache miss, o endereço desejado é carregado no barramento do sistema e os dados são transferidos através do buffer de dados para a cache e para o processador.

Projeto de Memória Cache

Projeto de Memória Cache

Detalhes de projeto:

- Endereçamento.
- Tamanho.
- Algoritmo de substituição.
- Política de escrita.
- Tamanho da linha.
- Número de caches.
- Função de mapeamento.

Projeto de Memória Cache

Endereçamento:

- A cache pode usar endereços reais ou virtuais para os endereços das palavras.
- A Unidade de Gerenciamento de Memória Virtual (MMU) é responsável por fazer a tradução entre endereço real e virtual.
- A memória virtual é uma abstração da memória real para facilitar a programação.

Projeto de Memória Cache

Endereçamento:

- Cache Lógica armazena dados usando endereço virtual.
  - Acesso à cache mais rápido, antes da tradução de endereço da MMU.
  - Endereços virtuais usam o mesmo espaço de endereços para diferentes aplicações.
  - Deve-se esvaziar cache a cada troca de contexto.
- Cache Física armazena dados usando endereços físicos da memória principal.

Projeto de Memória Cache

Tamanho:

- Em teoria, quanto mais melhor.
- No entanto, a cache é cara e ocupa espaço no chip do processador.
- Além disso, quanto mais linhas na cache mais tempo é necessário para fazer a verificação.

Projeto de Memória Cache

Algoritmo de substituição:

- Com o passar do tempo a cache fica cheia.
- Com isso, é necessário substituir linhas antigas por novas.

Projeto de Memória Cache

Algoritmo de substituição:

- LRU (Least Recently Used): Determina para substituição os blocos que não foram acessados recentemente.
- FIFO (First-In-First-Out): Seleciona como candidato para substituição o bloco que foi armazenado primeiro na cache.
- LFU (Least Frequently Used): Seleciona o bloco que tem tido menos acessos por parte do processador.
- Escolha Aleatória: O sistema de controle da memória Cache escolhe aleatoriamente o bloco que será removido.

Projeto de Memória Cache

Política de escrita:

- As alterações realizadas nos dados são feitas diretamente na cache.
- Com isso, elas devem ser repassadas para a memória principal antes da substituição da cache.
- Além disso, múltiplas CPUs/núcleos podem ter caches individuais e E/S pode endereçar a memória principal diretamente.

Projeto de Memória Cache

Política de escrita:

- As alterações realizadas nos dados são feitas diretamente na cache.
- Com isso, elas devem ser repassadas para a memória principal antes da substituição da cache.
- Além disso, múltiplas CPUs/núcleos podem ter caches individuais e E/S pode endereçar a memória principal diretamente.
- As técnicas usadas são: write-through, write-back e write-once.

Projeto de Memória Cache

Write-through:

- Todas as escritas vão para a memória principal e também para a cache.
- Múltiplas CPUs podem monitorar o tráfego da memória principal para manter a sua cache local (à CPU) atualizada.
- Mesmo conteúdo na memória cache e na memória principal!
- O grande problema é que isso gera muito tráfego no barramento.

Projeto de Memória Cache

Write-back:

- Atualizações feitas inicialmente apenas na cache.
- Um bit de atualização para cada linha de cache é definido quando ocorre a atualização.
- Se o bloco deve ser substituído, ele será atualizado na memória principal apenas se o bit de atualização estiver marcado.
- Caches de diferentes núcleos saem de sincronismo, ou seja, não deve ser utilizada em sistemas multi-core.
- E/S deve acessar a memória principal através da cache.

Projeto de Memória Cache

Write-once:

- É uma combinação do write-through com o write-back.
- Ideal para sistemas multi-core que compartilhem o mesmo barramento.
- A primeira escrita é realizada na cache e na memória principal (write-through) e a palavra fica reservada impedindo outros processadores de utilizarem.
- As escritas subsequentes são realizadas somente na cache (write-back) e a palavra da memória só será atualizado quando o bloco for substituído na cache.

Projeto de Memória Cache

Tamanho da linha:

- Relembrando, a cache recupera não apenas a palavra desejada, mas também uma série de palavras adjacentes.
- Em geral, se tamanho de bloco é aumentado, aumentará razão de acerto: O princípio da localidade.
- No entanto, existe um limite para esse aumento:
  - Se tamanho da linha é grande demais em relação ao tamanho da cache, a taxa de miss vai aumentar, pois existem poucas linhas.
  - Maior penalidade de miss, demora mais tempo para preencher a linha da cache.

Projeto de Memória Cache

Tamanho da linha:

- O aumento do tamanho da linha:
  - Reduz número de linhas da cache.
  - As linhas podem ser sobrescritos pouco depois de serem buscados.
  - Cada palavra adicional no bloco é 'menos local', de modo que é menos provável de ser necessária.
- Nenhum valor ideal definitivo foi descoberto.
- Normalmente, 8 a 64 bytes são usados.

Projeto de Memória Cache

Números de cache:

- Atualmente, a memória cache é projetada em multinível.
- Cache L1: Uma pequena porção de memória presente dentro do processador.
- Cache L2: Redução da penalidade de miss: mais um local pra buscar antes da memória principal.
- Cache L3: Similarmente a nível 2, outro local para buscar antes de chegar a memória.

Projeto de Memória Cache

Cache unificada vs separada:

- A cache unificada significa que dados e instruções ficam juntos.
- A cache separada significa que dados e instruções ficam em cache separadas.

Projeto de Memória Cache

Cache unificada vs separada:

- A cache unificada aumenta a taxa de acerto e facilita o projeto.
- A cache separada elimina disputa pela cache entre a unidade de busca de instrução e a unidade de execução em um pipeline, evitando bolhas.

Projeto de Memória Cache

Mapeamento:

- O mapeamento é responsável por detemrinar onde cada informação da memória principal estará na cache.
- O próprio endereço da informação na memória principal é usado no mapeamento para a cache.
- O objetivo é aumentar a quantidade de cache hit.

Projeto de Memória Cache

Mapeamento direto:

- Cada bloco de memória principal é mapeado apenas para uma linha de cache.
- Assim, se um bloco está na cache, ele deve estar em um local específico.
- Endereço é dividido em:
  - Tag (rótulo): identifica a linha (ou bloco) da memória principal (usado para validar se o bloco procurado é o mesmo que está na cache).
  - Linha: identifica a linha (ou bloco) da cache.
  - Palavra: identifica a palavra dentro do bloco.

Projeto de Memória Cache

Exemplo do mapeamento direto:

- Memória principal com 16 linhas.
- Memória cache com 4 linhas.
- Bloco da cache com 2 palavras.
- Com isso, os endereços serão:
  - Tag (rótulo): 1 bit
  - Linha: 2 bits
  - Palavra: 1 bit

Projeto de Memória Cache

Exemplo do mapeamento direto:

Memória cache:

Linha	Tag	Palavra	Palavra
00			
01			
10			
11			

Memória principal:

Endereço	Valor
0000	A
0001	B
0010	C
0011	D
0100	E
0101	F
0110	G
0111	H
1000	I
1001	J
1010	K
1011	L
1100	M
1101	N
1110	O
1111	P

Projeto de Memória Cache

Exemplo do mapeamento direto:

- Vamos supor que o processador faça a requisição dos seguintes endereços de memória: 0100, 1000, 0011 e 0111.

Projeto de Memória Cache

Exemplo do mapeamento direto:

Memória cache:

Linha	Tag	Palavra	Palavra
00	1	I	J
01	0	C	D
10	0	E	F
11	0	G	H

Memória principal:

Endereço	Valor
0000	A
0001	B
0010	C
0011	D
0100	E
0101	F
0110	G
0111	H
1000	I
1001	J
1010	K
1011	L
1100	M
1101	N
1110	O
1111	P

Projeto de Memória Cache

Exemplo do mapeamento direto:

- Em seguida, o processador faz a requisição dos seguintes endereços de memória: 1101 e 1110.

Projeto de Memória Cache

Exemplo do mapeamento direto:

Memória cache:

Linha	Tag	Palavra	Palavra
00	1	I	J
01	0	C	D
10	1	M	N
11	1	O	P

Memória principal:

Endereço	Valor
0000	A
0001	B
0010	C
0011	D
0100	E
0101	F
0110	G
0111	H
1000	I
1001	J
1010	K
1011	L
1100	M
1101	N
1110	O
1111	P

Projeto de Memória Cache

Outro exemplo do mapeamento direto:

- A palavra possui 1 byte.
- Cache única de 64 KB.
- Bloco de cache de 4 bytes (4 palavras).
- Memória principal de 16 MB.
- Quantos bits são necessários para os campos de tag, linha e palavra?

Projeto de Memória Cache

Outro exemplo do mapeamento direto:

- Primeiro, o endereço da memória principal possui 24 bits ($2^{24} = 16$ MB).
- Cada bloco da cache possui 4 palavras, então são necessários 2 bits para identificar cada palavra.
- A cache possui 16 K linhas com 4 bytes (4 palavras), então são necessários 14 bits para identificar cada bloco.
- Os 8 bits restantes fazem parte da tag.

Projeto de Memória Cache

Vantagens do mapeamento direto:

- Simples.

Desvantagens do mapeamento direto:

- Inflexível em relação ao estabelecimento da associação entre memória principal e cache, a associação é sempre a mesma.
- Quando dois ou mais blocos que são mapeados para o mesmo lugar sempre são requisitados em sequência, a cache fica sempre trocando (thrashing).

Projeto de Memória Cache

Mapeamento associativo:

- Um bloco de memória principal pode ser carregado em qualquer linha de cache.
- Isso resolve o problema do mapeamento inflexível e o thrashing do mapeamento direto.
- No entanto, a pesquisa da cache é dispendiosa.
- Endereço é dividido em:
  - Tag (rótulo): identifica a linha (ou bloco) da memória principal (usado para validar se o bloco procurado é o mesmo que está na cache).
  - Palavra: identifica a palavra dentro do bloco.

Projeto de Memória Cache

Exemplo do mapeamento associativo:

- Memória principal com 16 linhas.
- Memória cache com 4 linhas.
- Bloco da cache com 2 palavras.
- Com isso, os endereços serão:
  - Tag (rótulo): 3 bits
  - Palavra: 1 bit

Projeto de Memória Cache

Exemplo do mapeamento associativo:

Memória cache:

Tag	Palavra	Palavra

Memória principal:

Endereço	Valor
0000	A
0001	B
0010	C
0011	D
0100	E
0101	F
0110	G
0111	H
1000	I
1001	J
1010	K
1011	L
1100	M
1101	N
1110	O
1111	P

Projeto de Memória Cache

Exemplo do mapeamento associativo:

- Vamos supor que o processador faça a requisição dos seguintes endereços de memória: 0100, 1000, 0011 e 0111.

Projeto de Memória Cache

Exemplo do mapeamento associativo:

Memória cache:

Tag	Palavra	Palavra
010	E	F
100	I	J
001	C	D
011	G	H

Memória principal:

Endereço	Valor
0000	A
0001	B
0010	C
0011	D
0100	E
0101	F
0110	G
0111	H
1000	I
1001	J
1010	K
1011	L
1100	M
1101	N
1110	O
1111	P

Projeto de Memória Cache

Exemplo do mapeamento associativo:

- Em seguida, o processador faz a requisição dos seguintes endereços de memória: 1101 e 1110.
- Com isso, será necessário utilizar algum algoritmo de substituição.
- Por exemplo, vamos usar o FIFO (first in first out).

Projeto de Memória Cache

Exemplo do mapeamento associativo:

Memória cache:

Tag	Palavra	Palavra
110	M	N
111	O	P
001	C	D
011	G	H

Memória principal:

Endereço	Valor
0000	A
0001	B
0010	C
0011	D
0100	E
0101	F
0110	G
0111	H
1000	I
1001	J
1010	K
1011	L
1100	M
1101	N
1110	O
1111	P

Projeto de Memória Cache

Outro exemplo do mapeamento associativo:

- A palavra possui 1 byte.
- Cache única de 64 KB.
- Bloco de cache de 4 bytes (4 palavras).
- Memória principal de 16 MB.
- Quantos bits são necessários para os campos de tag e palavra?

Projeto de Memória Cache

Outro exemplo do mapeamento associativo:

- Primeiro, o endereço da memória principal possui 24 bits ($2^{24} = 16$ MB).
- Cada bloco da cache possui 4 palavras, então são necessários 2 bits para identificar cada palavra.
- Os 22 bits restantes fazem parte da tag.

Projeto de Memória Cache

Vantagens do mapeamento associativo:

- Flexibilidade em relação a substituição de blocos na cache.
- Possibilidade de emprego de algoritmos de substituição.

Desvantagens do mapeamento associativo:

- Complexidade do circuito necessário para comparar as tags, todas as tags devem ser checadas em cada busca.

Projeto de Memória Cache

Mapeamento associativo por conjunto:

- A cache é dividida em uma série de conjuntos.
- Cada conjunto contém uma série de linhas.
- Um bloco é mapeado para qualquer linha em determinado conjunto.
- É uma mistura entre o mapeamento direto e o associativo.
- Endereço é dividido em:
  - Tag (rótulo): identifica a linha (ou bloco) da memória principal (usado para validar se o bloco procurado é o mesmo que está na cache).
  - Conjunto: identifica o conjunto da cache.
  - Palavra: identifica a palavra dentro do bloco.

Projeto de Memória Cache

Exemplo do mapeamento associativo por conjunto:

- Memória principal com 16 linhas.
- Memória cache com 4 linhas.
- Bloco da cache com 2 palavras.
- Conjunto da cache com 2 linhas (2-way).
- Com isso, os endereços serão:
  - Tag (rótulo): 2 bits
  - Conjunto: 1 bit
  - Palavra: 1 bit

Projeto de Memória Cache

Exemplo do mapeamento associativo por conjunto:

Memória cache:

Conjunto	Tag	Palavra	Palavra
0			
0			
1			
1			

Memória principal:

Endereço	Valor
0000	A
0001	B
0010	C
0011	D
0100	E
0101	F
0110	G
0111	H
1000	I
1001	J
1010	K
1011	L
1100	M
1101	N
1110	O
1111	P

Projeto de Memória Cache

Exemplo do mapeamento associativo por conjunto:

- Vamos supor que o processador faça a requisição dos seguintes endereços de memória: 0100, 1000, 0011 e 0111.

Projeto de Memória Cache

Exemplo do mapeamento associativo por conjunto:

Memória cache:

Conjunto	Tag	Palavra	Palavra
0	01	E	F
0	10	I	J
1	00	C	D
1	01	G	H

Memória principal:

Endereço	Valor
0000	A
0001	B
0010	C
0011	D
0100	E
0101	F
0110	G
0111	H
1000	I
1001	J
1010	K
1011	L
1100	M
1101	N
1110	O
1111	P

Projeto de Memória Cache

Exemplo do mapeamento associativo por conjunto:

- Em seguida, o processador faz a requisição dos seguintes endereços de memória: 1101 e 1110.
- Com isso, será necessário utilizar algum algoritmo de substituição dentro de cada grupo.
- Por exemplo, vamos usar o FIFO (first in first out).

Projeto de Memória Cache

Exemplo do mapeamento associativo por conjunto:

Memória cache:

Conjunto	Tag	Palavra	Palavra
0	11	M	N
0	10	I	J
1	11	O	P
1	01	G	H

Memória principal:

Endereço	Valor
0000	A
0001	B
0010	C
0011	D
0100	E
0101	F
0110	G
0111	H
1000	I
1001	J
1010	K
1011	L
1100	M
1101	N
1110	O
1111	P

Projeto de Memória Cache

Exemplo do mapeamento associativo por conjunto:

- A palavra possui 1 byte.
- Cache única de 64 KB.
- Bloco de cache de 4 bytes (4 palavras).
- Conjunto da cache com 8 linhas (8-way).
- Memória principal de 16 MB.
- Quantos bits são necessários para os campos de tag, conjunto e palavra?

Projeto de Memória Cache

Exemplo do mapeamento associativo por conjunto:

- Primeiro, o endereço da memória principal possui 24 bits ($2^{24} = 16$ MB).
- Cada bloco da cache possui 4 palavras, então são necessários 2 bits para identificar cada palavra.
- A cache possui 16 K linhas com 4 bytes (4 palavras), agrupados em conjuntos de 8 linhas, no total existem 2 K grupos, então são necessários 11 bits para identificar cada grupo.
- Os 11 bits restantes fazem parte da tag.