

# Memória Cache

Profa. Débora Christina Muchaluat Saade  
debora@midiaacom.uff.br

<http://www.midiaacom.uff.br/debora>

1

# Memória Cache

- ✓ Capítulo 5 – Livro do Mário Monteiro
- ✓ Conceituação
  - *Princípio da localidade*
  - *Funcionamento da memória cache*
- ✓ Elementos de projeto de memória cache
  - *Mapeamento de dados MP/cache*
  - *Algoritmos de substituição de dados na cache*
  - *Políticas de escrita pela memória cache*

2

## Diferença de velocidade entre Processador/MP

- ✓ Processador executa operação rapidamente e fica em estado de espera (*wait state*) para receber dados da memória

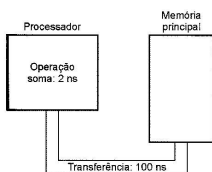


Figura 5.1 Exemplo de diferença de velocidade P/MP. Enquanto o processador gasta 2 ns adicionando dois dados a MP gasta 100 ns transferindo os dados para o processador.

3

## Diferença de velocidade entre Processador/MP

- ✓ Na década de 1960, diversos pesquisadores (principalmente da IBM) se reuniram para analisar o comportamento dos processos (programas)
  - *Princípio da localidade (locality of reference ou principle of locality)*

4

## Conceito de Localidade

- ✓ Programa executável tem instruções ordenadas sequencialmente
  - *Quando programa executa, processador busca instruções sequencialmente em memória, exceto quando ocorre um loop ou comando de desvio, em que a sequência de acesso é alterada abruptamente.*
    - Princípio de localidade
      - *Localidade espacial*
      - *Localidade temporal*

5

## Localidade Espacial

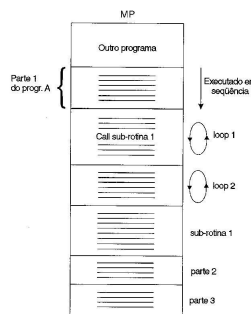


Figura 5.2 Um programa em execução com várias partes (exemplo do princípio de localidade espacial).

## Localidade Temporal

Fundamentos de Arquiteturas de Computadores

- Programas tendem a usar o mesmo endereço em um curto espaço de tempo
- Loops repetem um mesmo conjunto de instruções

```

Cálculo da média de 2 turmas, A e B
void main ()
{
    printf("Número de alunos da turma A: ");
    scanf("%d", &numA); // início de execução em seqüência
    soma_num_A = 0; // início execução em seqüência
    for (i=0; i<numA; i++)
    {
        printf("Informe a matrícula do aluno: ");
        scanf("%d", &matrA[i]); // loop - fixo
        printf("Informe a nota: ");
        scanf("%d", &notaA[i]); // loop - fixo
        if (matrA[i] < matrA[numA-1])
            soma_num_A = soma_num_A + matrA[i]; // loop - memória
    }
    media_num_A = soma_num_A / numA; // sub-rotina (tempo fixo)
    return 0;
}

Cálculo da média de 2 turmas, B e C
void main ()
{
    printf("Número de alunos da turma B: ");
    scanf("%d", &numB); // início de exec. em seq
    soma_num_B = 0; // início exec. seq.
    for (j=0; j<numB; j++) // loop - índice
    {
        printf("Informe a matrícula do aluno: ");
        scanf("%d", &matrB[j]); // loop - índice
        printf("Informe a nota: ");
        scanf("%d", &notaB[j]); // loop - índice
        if (matrB[j] < matrB[numB-1])
            soma_num_B = soma_num_B + matrB[j]; // loop - índice
    }
    media_num_B = soma_num_B / numB; // memória - loop
}

Cálculo da média turma B
printf("A média dos alunos da turma B foi: %.2f", media_num_B);
printf("A nota mais alta da turma B foi: %.2f", max_num_B);
printf("O(s) aluno(s) da turma B obtiver(m) nota superior à maior nota da turma A: %s",

```

Figura 5.3 Exemplo de programa para demonstração de localidades na sua execução.

## Cache

Fundamentos de Arquiteturas de Computadores

- Memória cache (cache de RAM – entre CPU/MP)
- Cache de disco (entre MP/Disco)
- Cache do browser web (armazenamento local)
- Etc.

## Funcionamento da Memória Cache

Fundamentos de Arquiteturas de Computadores

- Memória cache possui velocidade de acesso alta e capacidade para armazenar partes de um programa de forma a aproveitar o princípio da localidade

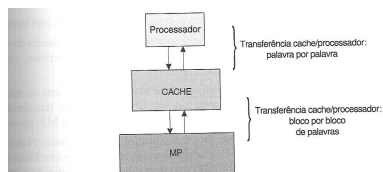


Figura 5.4 Organização para transferência de informações entre Processador/Cache/MP.

## Funcionamento do Acesso a Memória

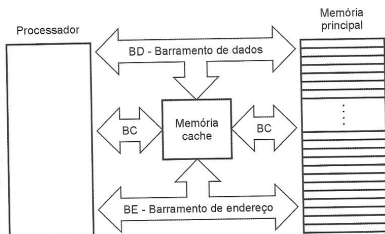
Fundamentos de Arquiteturas de Computadores

- Processador inicia operação de leitura e coloca endereço da MP no BE
- Sistema de controle de cache (*chipset*) intercepta o endereço, interpreta o conteúdo e verifica se o dado está ou não em cache
  - Se estiver, *acerto (hit)* → cópia do dado é transferida da cache para o processador pelo BD
  - Se não estiver, *falta (miss)* → controle da MP é acionado para recuperar o bloco da MP e transferi-lo para cache, para em seguida, transferir ao processador pelo BD
    - Tempo de acesso é bem maior

## Funcionamento do Acesso a Memória

Fundamentos de Arquiteturas de Computadores

- Conexão entre processador, memória cache e MP



## Funcionamento do Acesso a Memória

Fundamentos de Arquiteturas de Computadores

- Princípio da localidade espacial, é muito provável que o acesso seguinte seja sequencial
  - Sistema busca da MP o dado solicitado e mais alguns, que supõe que serão usados em seguida
  - MP dividida em blocos de X bytes e cache dividida em linhas de X bytes de largura
- Deseja-se máximo de acertos (*hits*) e mínimo de faltas (*misses*) para um bom desempenho
  - Eficiência da cache
    - $E_c = (\text{acertos (hits)} / \text{total de acessos}) * 100$
    - Normalmente é da ordem de 95% a 98%

## Organização da Memória Cache

Fundamentos de Arquiteturas de Computadores

- ✓ Memória cache é organizada em linhas de X bytes

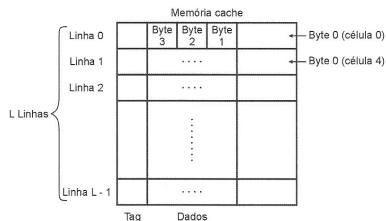


Figura 5.6 Organização básica de uma memória cache.

13

## Organização da Memória Cache e MP

Fundamentos de Arquiteturas de Computadores

- ✓ MP organizada em blocos de X células (1 byte)
- ✓ Cache organizada em linhas de X bytes
  - *Tag ou rótulo indica o endereço do bloco da MP*

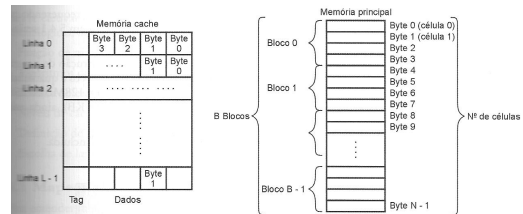


Figura 5.7 Organização memória cache/memória principal.

## Organização da Memória Cache e MP

Fundamentos de Arquiteturas de Computadores

- ✓ Quantidade de blocos da MP é muito maior que quantidade de linhas da cache
  - *Métodos para mapear blocos da MP em linhas da cache*
    - Direto
    - Associativo
    - Associativo por conjunto



Figura 5.8 Exemplo de um sistema de computação (microcomputador) com utilização de memória cache em um barramento único.

## Projeto de Memória Cache

Fundamentos de Arquiteturas de Computadores

- ✓ Função de mapeamento de dados MP/cache
- ✓ Algoritmo para substituição de dados na cache
- ✓ Política de escrita pela cache
- ✓ Níveis de cache
- ✓ Definição do tamanho das memórias cache L1 e L2
- ✓ Escolha da largura de linha de cache

16

## Mapeamento de Dados MP/cache

Fundamentos de Arquiteturas de Computadores

- ✓ MP tem N células (numeradas de 0 a N-1)
- ✓ MP organizada como conjunto de B blocos (numerados de 0 a B-1)
  - *Cada bloco da MP é constituído de X células (bytes)*
  - *tamanho da MP = B x X*
- ✓ Quantidade de blocos  $B = N / X$
- ✓ Memória cache organizada como conjunto de L linhas, cada uma com X bytes
  - *Tamanho da cache = L x X*
- ✓ Tamanho da cache  $\ll$  tamanho da MP

17

## Mapeamento de Dados MP/cache

Fundamentos de Arquiteturas de Computadores

- ✓ Uma linha da cache pode armazenar diferentes blocos da MP, por isso precisa identificar que bloco armazena em cada momento
  - *Tag ou rótulo da linha*
- ✓ Como determinar em que linha da cache cada bloco de memória será armazenado?
  - *Métodos para mapear blocos da MP em linhas da cache*
    - Mapeamento direto
    - Mapeamento associativo
    - Mapeamento associativo por conjunto

18

## Mapeamento Direto

Fundamentos de Arquiteturas de Computadores

- ✓ Cada bloco da MP tem uma linha previamente definida onde será armazenado
  - **Cache tem L linhas**
    - Linha = bloco MOD L
  - **Bloco 0 na linha 0**
  - **Bloco 1 na linha 1**
  - ...
  - **Bloco L-1 na linha L-1**
  - **Blocos 0, L, 2L, 3L, etc. são mapeados na linha 0**
  - **Blocos 1, L+1, 2L+1, 3L+1, etc. são mapeados na linha 1**
  - ...

19

## Mapeamento Direto

Fundamentos de Arquiteturas de Computadores

- ✓ MP tem 64B
  - **Bloco de 4B**
- ✓ Cache tem 16B
  - **Linha de 4B**

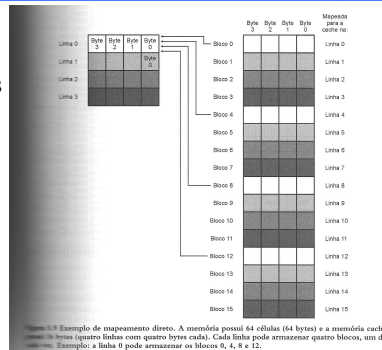
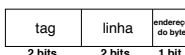


Figura 5.9 Exemplo de mapeamento direto. A memória possui 64 células (64 bytes) e a memória cache possui 16 bytes (quatro linhas com quatro bytes cada). Cada linha pode armazenar quatro blocos, um de cada tipo. Exemplo: a linha 0 pode armazenar os blocos 0, 4, 8 e 12.

## Mapeamento Direto

Fundamentos de Arquiteturas de Computadores

- ✓ Capacidade da MP =  $32B = 2^5$ 
  - **cada endereço tem 5 bits**
- **Blocos de 2B**
- ✓ Total de blocos =  $32B / 2B = 16 = 2^4$ 
  - **Cada bloco tem endereço de 4 bits**
- ✓ Capacidade da cache =  $8B = 2^3$
- ✓ Total de linhas =  $8B / 2B = 4 = 2^2$ 
  - **cada linha tem 2 bits**
- ✓ Tag do bloco tem 2 bits



21

## Mapeamento Direto

Fundamentos de Arquiteturas de Computadores

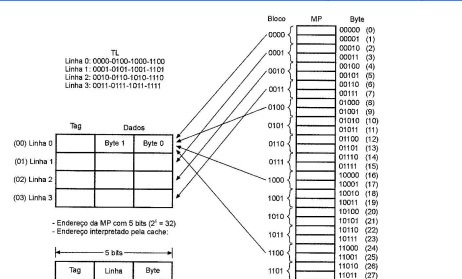
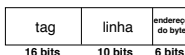


Figura 5.10 Exemplo de organização com mapeamento direto em uma MP com 32 células (bytes) e cache com quatro linhas de 2 bytes cada.

## Mapeamento Direto

Fundamentos de Arquiteturas de Computadores

- ✓ Capacidade da MP =  $4GB = 2^{32}$ 
  - **cada endereço tem 32 bits**
- **Blocos de 64B**
- ✓ Total de blocos =  $4GB / 64B = 2^{26}$ 
  - **Cada bloco tem endereço de 26 bits**
- ✓ Capacidade da cache = 64KB
- ✓ Total de linhas =  $64KB / 64B = 1K = 2^{10}$ 
  - **cada linha tem 10 bits**
- ✓ Tag do bloco tem 16 bits



23

## Mapeamento Direto

Fundamentos de Arquiteturas de Computadores

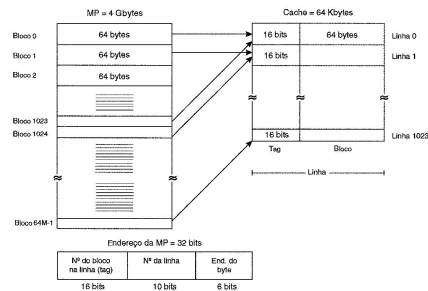


Figura 5.11 Memória cache com mapeamento direto.

24

## Tamanho da Cache

Fundamentos de Arquiteturas de Computadores

- ✓ **Número de bits necessários para armazenar dados + Número de bits necessários para armazenar tags**

- **No exemplo:**

- 64K x 8 bits – dados
- 1K x 16 bits – tags
- Total de 66KB

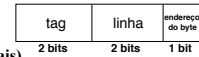
25

## Acesso a Cache com Mapeamento Direto

Fundamentos de Arquiteturas de Computadores

- ✓ Endereço da MP é interpretado pelo sistema de controle da cache

- 2 bits tag
- 2 bits linha
- 1 bit endereço do byte



- ✓ Identifica a linha selecionada (2 bits centrais)
- ✓ Verifica se bloco está na cache
  - *Compara tag linha com tag endereço*
- ✓ Se for igual → hit
  - *Valor do byte é passado para processador pelo BD*
- ✓ senão → miss
  - *Sistema inicia a localização do bloco na MP para transferir cópia para a linha específica*
    - endereço do bloco corresponde aos 4 bits mais significativos

26

## Acesso a Cache com Mapeamento Direto

Fundamentos de Arquiteturas de Computadores

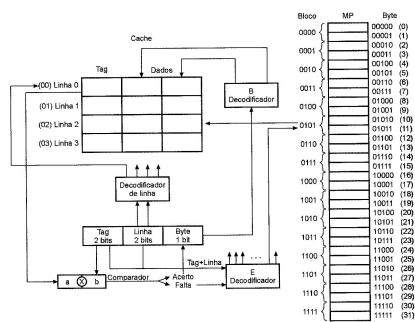


Figura 5.12 Exemplo de acesso à memória cache por meio de mapeamento direto.

## Acesso a Cache com Mapeamento Direto

Fundamentos de Arquiteturas de Computadores

- ✓ Se processador modificar o valor do bloco em cache (operação de escrita), este deve ser copiado para memória principal
- ✓ Para saber se houve alteração no conteúdo do bloco em cache, utiliza-se um bit adicional em cada linha que será setado (1) por uma operação de escrita, ou mantido em zero (0), caso contrário.

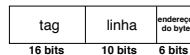
28

## Acesso a Cache com Mapeamento Direto

Fundamentos de Arquiteturas de Computadores

- ✓ **Outro exemplo:**

- *Endereço da MP é interpretado pelo sistema de controle da cache*
  - 16 bits tag
  - 10 bits linha
  - 6 bits endereço do byte



- *Identifica a linha selecionada (10 bits centrais)*
- *Verifica se bloco está na cache*
  - Compara tag linha com tag endereço
- *Se for igual → hit*
  - Valor do byte é passado para para processador pelo BD
- *senão → miss*
  - Sistema inicia a localização do bloco na MP para transferir cópia para a linha específica
    - *endereço do bloco corresponde aos 26 bits mais significativos*

29

## Acesso a Cache com Mapeamento Direto

Fundamentos de Arquiteturas de Computadores

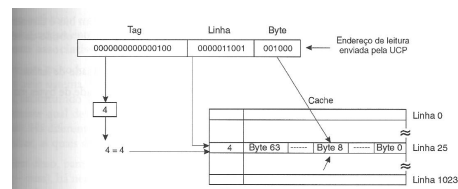


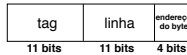
Figura 5.13 Exemplo de operação de leitura em memória cache com mapeamento direto.

30

## Exemplo

Fundamentos de Arquiteturas de Computadores

- ✓ Considere uma MP com 64MB de capacidade associada a uma cache de 2K linhas, cada uma com largura de 16 bytes. Determine o formato do endereço a ser interpretado pelo sistema de controle de cache que utiliza mapeamento direto.
  - *cada endereço tem 26 bits*
  - *Blocos de 16B*
- ✓ Capacidade da MP =  $64\text{MB} = 2^{26}$ 
  - *cada endereço tem 26 bits*
  - *Blocos de 16B*
- ✓ Total de blocos =  $64\text{MB} / 16\text{B} = 4\text{M} = 2^{22}$ 
  - *Cada bloco tem endereço de 22 bits*
- ✓ Capacidade da cache =  $2\text{K} \times 16\text{B} = 32\text{KB} = 2^{15}$
- ✓ Total de linhas =  $2\text{K} = 2^{11}$ 
  - *cada linha tem 11 bits*
- ✓ Tag do bloco tem 11 bits



31

## Mapeamento Direto

Fundamentos de Arquiteturas de Computadores

- ✓ Simples e de baixo custo de implementação
- ✓ Processamento rápido do endereço
- ✓ Problema da localização fixa dos blocos em cache
  - *Se o programa fizer sucessivos acessos (loop) a palavras situadas em blocos alocados na mesma linha, haverá necessidade de sucessivos acessos a MP (misses)*
  - *Relação acerto/faltas será baixa → baixo desempenho*

32

## Mapeamento Associativo

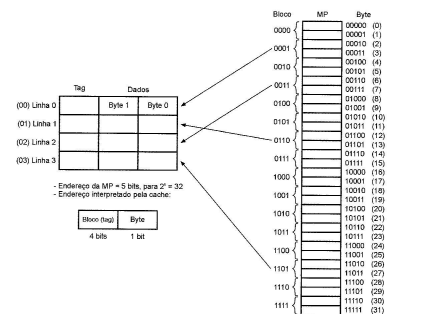
Fundamentos de Arquiteturas de Computadores

- ✓ Oposto do mapeamento direto, não existe posição fixa para cada bloco de memória em cache
- ✓ Um bloco de memória pode ser armazenado em qualquer linha da cache
  - *Necessidade de escolha de qual bloco será substituído*
    - políticas de substituição de linhas
- ✓ Sistema de controle compara endereço do bloco completo com a tag de cada linha
- ✓ Para verificação rápida
  - *Deve realizar comparação simultânea em todas as linhas*
    - Em memórias cache maiores, custo do hardware é elevado

33

## Mapeamento Associativo

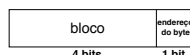
Fundamentos de Arquiteturas de Computadores



## Mapeamento Associativo

Fundamentos de Arquiteturas de Computadores

- ✓ Capacidade da MP =  $32\text{B} = 2^5$ 
  - *cada endereço tem 5 bits*
  - *Blocos de 2B*
- ✓ Total de blocos =  $32\text{B} / 2\text{B} = 16 = 2^4$ 
  - *Cada bloco tem endereço de 4 bits (tag)*



35

## Acesso a Cache com Mapeamento Associativo

Fundamentos de Arquiteturas de Computadores

- ✓ Endereço da MP é interpretado pelo sistema de controle da cache
  - *4 bits bloco (tag)*
  - *1 bit endereço do byte*
- ✓ Verifica se bloco está na cache
  - *Valor do campo bloco é replicado em todos os elementos de comparação (1 por linha)*
  - *Compara tag de cada linha com bloco do endereço*
  - *Todas as comparações são feitas simultaneamente*
- ✓ Se for igual → hit
  - *Valor do byte é passado para processador pelo BD*
- ✓ senão → miss
  - *Sistema inicia a localização do bloco na MP para transferir cópia para a linha escolhida de acordo com a política de substituição de linhas*

36

## Acesso a Cache com Mapeamento Associativo

Fundamentos de Arquiteturas de Computadores

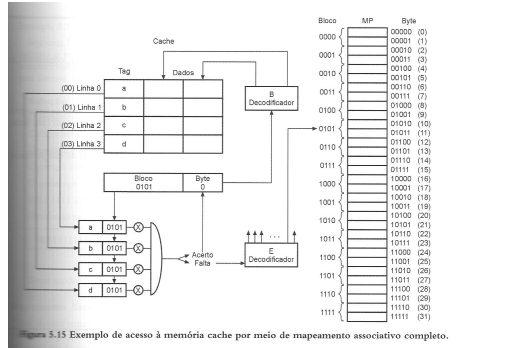


Figura 5.15 Exemplo de acesso à memória cache por meio de mapeamento associativo completo.

## Exemplo

Fundamentos de Arquiteturas de Computadores

- ✓ Considere um sistema de computação com memória cache de 32KB de capacidade, constituída de linhas com 8 bytes de largura. A MP possui capacidade de 16MB. Calcule a quantidade de bits necessárias para implementação da cache com mapeamento associativo.
- ✓ Total bits cache = total bits dados + total bits tags
- ✓ Total bits dados =  $32K \times 8 \text{ bits} = 262.144 \text{ bits}$
- ✓ Total bits tags = quantidade linhas cache  $\times$  largura do bloco (tag)
- ✓ Quantidade linhas cache =  $32KB / 8 \text{ bytes} = 4K = 2^{12}$
- ✓ Quantidade de blocos =  $16MB / 8B = 2M = 2^{21}$
- ✓ Endereço do bloco tem 21 bits (largura do bloco)
- ✓ Total bits tags =  $4K \times 21 = 84K\text{bits} = 86.016$
- ✓ Total bits cache = total bits dados + total bits tags
  - Total =  $262.144 + 86.016 = 348.160 \text{ bits} = 340K\text{bits}$

bloco	endereço do byte
21 bits	3 bits

38

## Mapeamento Associativo

Fundamentos de Arquiteturas de Computadores

- ✓ Flexibilidade na alocação e armazenamento de blocos em cache
- ✓ Hardware mais complexo com aumento do custo e complexidade do sistema de controle de cache
  - Para cache com milhares de linhas, teríamos milhares de circuitos comparadores
- ✓ Maior quantidade de bits na cache (tag armazena endereço completo do bloco)

39

## Mapeamento Associativo por Conjunto

Fundamentos de Arquiteturas de Computadores

- ✓ Tenta resolver o problema de conflito de blocos na mesma linha (mapeamento direto) e problema de custo da comparação do campo tag (mapeamento associativo)
  - Solução de compromisso entre as técnicas anteriores
- ✓ Divide-se o espaço de linhas da cache em conjuntos de N linhas
- ✓ Cada conjunto é tratado pelo sistema como método direto
- ✓ Dentro de cada conjunto, o método é o associativo

40

## Mapeamento Associativo por Conjunto

Fundamentos de Arquiteturas de Computadores

- ✓ Exemplo:
  - MP de  $32B (2^5)$ 
    - Blocos de 2B
    - 16 blocos
  - Memória cache de 4 linhas de 2B cada
  - Cache é dividida em 2 conjuntos ( $2^1$ ) de 2 linhas cada
    - Conjunto tem 1 bit
  - 8 blocos em cada conjunto ( $2^3$ )
    - Tag tem 3 bits

tag	conjunto	endereço do byte
3 bits	1 bit	1 bit

41

## Mapeamento Associativo por Conjunto

Fundamentos de Arquiteturas de Computadores

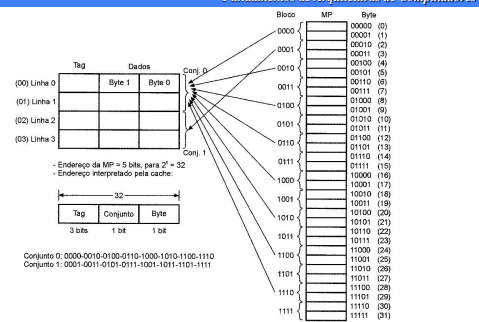
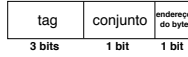


Figura 5.16 Exemplo de organização com mapeamento associativo por conjunto em MP com 32 bytes e uma cache com quatro linhas de 2 conjuntos de duas linhas.

## Acesso a Cache com Mapeamento Associativo por Conjunto

Fundamentos de Arquiteturas de Computadores

- Endereço da MP é interpretado pelo sistema de controle da cache
  - Tag – 3 bits
  - Conjunto – 1 bit
  - 1 bit endereço do byte
- Valor do campo conjunto é identificado
- Verifica se bloco está na cache
  - Valor do campo tag é replicado em todos os elementos de comparação (1 por linha do conjunto)
  - Todas as comparações são feitas simultaneamente
- Se for igual → hit
  - Valor do byte é passado para processador pelo BD
- senão → miss
  - Sistema inicia a localização do bloco na MP para transferir cópia para a o conjunto específico
    - A linha dentro do conjunto é escolhida de acordo com a política de substituição de linhas



43

## Acesso a Cache com Mapeamento Associativo por Conjunto

Fundamentos de Arquiteturas de Computadores

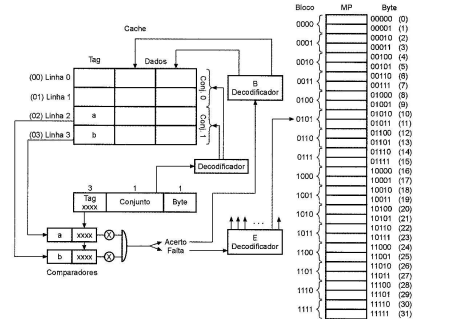


Figura 5.17 Exemplo de acesso à memória cache por meio de mapeamento associativo por conjunto

44

## Exemplo

Fundamentos de Arquiteturas de Computadores

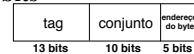
- Seja uma MP constituída de blocos com largura de 32 bytes, associada a uma cache com 128KB, que usa mapeamento associativo por conjunto de 4. Em um dado instante, o processador realiza um acesso, colocando o seguinte endereço 3FC92B6. Determine qual deverá ser o valor binário do conjunto que será localizado pelo sistema de controle de cache.

45

## Exemplo

Fundamentos de Arquiteturas de Computadores

- Endereço de 7 algarismos hexa =  $7 \times 4 = 28$  bits
- 32 bytes ( $2^5$ ) por bloco = endereço do byte tem 5 bits
- Quantidade de linhas = total cache / largura da linha
  - $128KB / 32B = 4K = 2^{12}$
- Quantidade de conjuntos = quantidade linhas / tamanho do conjunto
  - $4K / 4 = 2^{10} \rightarrow$  campo conjunto tem 10 bits
- Campo tag tem  $28 - 5 - 10 = 13$  bits

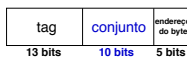


46

## Exemplo

Fundamentos de Arquiteturas de Computadores

- Endereço 3FC92B6
- Em binário: 0011 1111 1100 1001 0010 1011 0110



- Tag (13 bits) = 0011111111001
- Conjunto (10 bits) = 0010010101
- Endereço do byte = 10110

47

## Mapeamento Associativo por Conjunto

Fundamentos de Arquiteturas de Computadores

- A maioria dos sistemas emprega mapeamento associativo por conjunto, variando o valor de N
  - Conjuntos de 4, 8, 16
- À medida que a capacidade da cache aumenta, aumenta-se o número de linhas por conjunto (N), de modo a equilibrar as vantagens dos métodos anteriores
  - facilidade de identificação do conjunto e flexibilidade da posição do bloco no conjunto

48



## Algoritmos de Substituição de Dados na Cache

Fundamentos de Arquiteturas de Computadores

- ✓ Qual dos blocos armazenados deve ser substituído por um novo bloco? ( $L \ll B$ )
- ✓ Decisão necessária quando método de mapeamento é associativo
- ✓ Algoritmos:
  - *LRU – Least Recently Used*
  - *FIFO – First-In, First-Out*
  - *LFU – Least Frequently Used*
  - *Escolha aleatória*

49

## LRU – Least Recently Used

Fundamentos de Arquiteturas de Computadores

- ✓ Escolhe o bloco que não é usado há mais tempo
- ✓ Bit indicando que linha foi usada pelo processador
- ✓ Em caches associativas por conjuntos de 2 → simples de implementar
  - *Quando uma das linhas for acessada, bit é setado (1) e o bit da outra linha do conjunto é zerado*
- ✓ Quando aumenta a associatividade, problema se torna maior → difícil implementar processo de forma exata

50

## Algoritmos de Substituição de Dados na Cache

Fundamentos de Arquiteturas de Computadores

- ✓ FIFO – First-In, First-Out
  - *Esquema de fila*
    - Primeiro a chegar é o primeiro a sair
  - *Escolha independente da frequência de uso do bloco pelo processador*
- ✓ LFU – Least Frequently Used
  - *Bloco que teve menos acessos pelo processador é escolhido*
- ✓ Escolha aleatória
  - *Bloco é escolhido aleatoriamente, independente de seu uso pelo processador*

51

## Algoritmos de Substituição de Dados na Cache

Fundamentos de Arquiteturas de Computadores

- ✓ Estudo sobre memórias cache, baseado em simulações, indica que escolha aleatória reduz muito pouco o desempenho do sistema em comparação com os demais algoritmos
  - *Simple de implementar*
- ✓ Quando associatividade aumenta (conjuntos de 4/8/etc), LRU e aleatório quase se equivalem em desempenho
  - *Aleatório é mais simples e barato em termos de hardware*

52

## Política de Escrita pela Memória Cache

Fundamentos de Arquiteturas de Computadores

- ✓ Operações de escrita do processador são feitas em cache
  - *Necessário atualizar MP para sistema manter correção e integridade*
- ✓ Antes de substituir o bloco em cache, é necessário verificar se foi alterado e se alterações já foram feitas na MP
- ✓ Algumas considerações:
  - *MP pode ser acessada pela cache ou por dispositivos de E/S (DMA – Direct Memory Access)*
    - Cache por ter sido alterada e MP ainda não
    - MP pode ter sido alterada e cache está desatualizada
  - *MP pode ser acessada por vários processadores, cada um com sua cache*
    - MP pode ser alterada e outras caches estarem desatualizadas

53

## Política de Escrita pela Memória Cache

Fundamentos de Arquiteturas de Computadores

- ✓ Write through – escrita em ambas
  - *Cada escrita em cache acarreta escrita em MP*
  - *Caso haja outros processadores, estes também alteram suas caches*
  - *Mesmo conteúdo sempre nas duas memórias*
- ✓ Write back – escrita somente no retorno
  - *Escrita só é atualizada em MP quando bloco for substituído e se foi atualizado*
  - *Bit adicional é setado (1) se bloco foi atualizado em cache*
  - *Quando for substituído, se bit correspondente estiver setado, escrita é feita na MP*
- ✓ Write once – escrita uma vez

54

## Política de Escrita pela Memória Cache

Fundamentos de Arquiteturas de Computadores

- ✓ **Write through** – escrita em ambas
- ✓ **Write back** – escrita somente no retorno
- ✓ **Write once** – escrita uma vez
  - *Apropriada para sistema multiprocessados (cada um com sua cache) compartilhando mesmo barramento*
  - *Controlador da cache atualiza MP quando bloco foi atualizado pela primeira vez (write through) e alerta outros componentes que compartilham o barramento*
  - *Eles são notificados sobre alteração e impedem o uso da palavra específica*
  - *Próximas alterações são feitas somente em cache local e o bloco só é atualizado em MP quando for substituído (write back)*

55

## Comparação entre as técnicas de escrita

Fundamentos de Arquiteturas de Computadores

- ✓ **Write through**
  - *pode haver grande quantidade de escritas desnecessárias em MP*
  - *reduz desempenho do sistema*
- ✓ **Write back**
  - *Minimiza desvantagem anterior*
  - *MP pode ficar desatualizada para utilização por outros dispositivos (E/S), o que os obriga a acessar o dado através da cache*
- ✓ **Write once**
  - *Conveniente para sistemas multiprocessados*
  - *Não é muito usada ainda*
- ✓ **Estudos sobre cache indicam que a percentagem de escrita é pequena (da ordem de 15%)**
  - *Política simples de write through*

56

## Níveis de Cache

Fundamentos de Arquiteturas de Computadores

- ✓ **Nível 1 (Level 1) ou L1**
  - *Sempre localizada no interior do processador*
  - *Cache primária*
    - Cache L1 de instruções e cache L1 de dados
- ✓ **Nível 2 (Level 2) ou L2**
  - *normalmente localizada no exterior do processador (placa mãe)*
    - Cache secundária
  - *Alguns processadores têm L2 interna a pastilha do processador*
- ✓ **Nível 3 (Level 3) ou L3**
  - *Quando processador possui L1 e L2 interna, é a cache externa ao processador (placa mãe)*

57

## Exemplos de cache L1

Fundamentos de Arquiteturas de Computadores

Tabela 5.1 Exemplos de Caches em Processadores

Processadores	Fabricante	Tamanho da Cache
80486	Intel	8 KB
Athlon K7	AMD	I-64KB e D-64KB
Pentium	Intel	I-8KB e D-8KB
Pentium MMX	Intel	I-8KB e D-24KB
Pentium PRO	Intel	I-8KB e D-8KB
PowerPC 601	Motorola/IBM	32 KB
Pentium III	Intel	I-8KB e D-24KB
Pentium 4	Intel	I-8KB e D-8KB
Athlon 64	AMD	I-64KB e D-64KB
Power 5	IBM	64KB
Itanium	Intel	I-16KB e D-16KB

58

## Tamanho da Memória Cache

Fundamentos de Arquiteturas de Computadores

- ✓ **Definição do tamanho adequado depende de vários fatores:**
  - *Tamanho da memória principal*
  - *Relação acertos/faltas*
  - *Tempo de acesso da MP e das caches L1 e L2*
  - *Custo médio por bit da MP e das caches L1 e L2*
  - *Natureza do programa em execução*
- ✓ **Exemplos**
  - *Faixa entre 32KB e 256KB para cache L1*
  - *Faixa entre 64 KB a 4MB para cache L2*
- ✓ **Largura de linha ótima é uma decisão difícil**
  - *Linha maior atende ao princípio da localidade espacial*
    - Desvios nos programas (if-then-else) aumentam o número de faltas
  - *É comum encontrar caches com linhas entre 8 e 32 bytes*

59