Memória Cache

Conceitos Básicos

- Bloco
    - Unidade de transferência de dados.
    - Ao acessar um dado na memória, todo o bloco onde este dado se encontra será transferido para a memória cache.
    - Na memória cache temos blocos de bytes que podem ter sido lidos de qualquer lugar da memória principal.
    - Blocos da memória cache devem ser rotulados com seus endereços na memória principal (tag).
- Cache Hit
    - Quando um dado solicitado pelo processador se encontra na cache, dizemos que houve um "cache hit".
- Cache Miss
    - Quando um dado solicitado pelo processado não se encontra na cache, é necessário acessar a memória principal, trazer o bloco onde o dado se encontra para a cache (podendo eventualmente ser necessário remover algum bloco da cache ("eviction")) e, então, retornar o dado ao processador. Quando isso ocorre, dizemos que houve um "cache miss".
- Princípio da Localidade
    - Espacial: se um dado foi acessado recentemente, é muito provável que outros dados próximos a este serão utilizados num futuro próximo.
    - Temporal: se um dado foi acessado recentemente, é muito provável que este seja acessado novamente num futuro próximo.
    - Hot Spots: "pontos" da memória utilizados com frequência por um processo. Programas típicos tendem a executar muitas vezes as mesmas instruções e utilizar repetidamente os mesmos dados.

Operações de Escrita:

- Write-hit: cache contém bloco com endereço onde o processador deseja escrever. Temos duas formas de realizar essa operação:
    - Write-through: dado é escrito imediatamente na memória principal.
        - Este processo é lento, sendo pouco utilizado atualmente.
    - Write-back: dado é inicialmente escrito apenas na memória cache e a escrita na memória principal é realizada somente quando o bloco contendo o dado tiver que ser retirado da cache.
        - Mais rápido que o write-through, porém requer um bit adicional para indicar que o dado foi modificado ("dirty bit").
- Write-miss: cache não contém bloco com endereço onde o processador deseja escrever. Temos duas formas de realizar essa operação:

- Write-allocate: bloco da memória é trazido para a cache e a escrita é feita somente na cache. A escrita na memória principal é feita somente quando o bloco contendo o dado tiver que ser removido da cache (similar ao "write-back").
  - Este é o método mais utilizado atualmente.
- Write-no-allocate: escreve dado na memória principal, mas não o traz para a cache.

Memória Cache com Mapeamento Direto

- Bloco da RAM pode ser colocado em um único bloco da cache.

Para saber o número da blocos da cache:

$$< \text{número de blocos da cache} > = \frac{< \text{tamanho da cache} >}{< \text{tamanho do bloco} >}$$

Para saber em qual bloco da cache um bloco da RAM será colocado:

$$< \text{bloco da cache} > = < \text{bloco da RAM} > \% < \text{número de blocos da cache} >$$

- Endereçamento dos dados em memória cache com mapeamento direto:

Calculamos o número de bits utilizados para o índice dos blocos da seguinte forma:

$$< \text{bits índice} > = \log_2 < \text{número de blocos da cache} >$$

Ou seja,

$$< \text{número de blocos da cache} > = 2^{<\text{bits índice}>}$$

O offset para acessar um dado dentro de um bloco da cache é igual ao offset para acessar este dado dentro do respectivo bloco na RAM. Para calcularmos o número de bits necessários para o offset, fazemos o seguinte:

$$< \text{bits offset} > = \log_2 < \text{tamanho do bloco} >$$

Usamos também os bits de tag, que servem para identificar qual bloco da RAM foi mapeado para o bloco da cache (identificado pelo índice). Sendo <bits endereço> o número total de bits de um endereço de memória, calculamos o número de bits de tag como

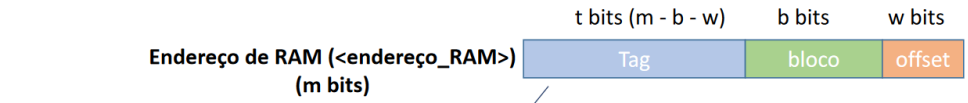
$$< \text{bits tag} > = < \text{bits endereço} > - < \text{bits índice} > - < \text{bits offset} >$$

Aritmética para determinar a **localização dos dados (bloco e offset)**

❑ Blocos da RAM estarão **sempre no mesmo bloco** da cache

A) Separar os bits de endereço de bloco a partir do endereço da RAM

B) Calcular usando operadores DIV e MOD (resultado da divisão inteira e resto da divisão inteira)



bloco_RAM = <endereço_RAM> DIV 2^w
bloco_cache = bloco_RAM MOD 2^b
offset (palavra dentro do bloco) = <endereço_RAM> MOD 2^w

Exemplo:
RAM de 256 bytes (t + b + w = 8)
Cache de 16 bytes @ 4 bytes/bloco – 4 blocos (w = 2; b = 2)
Aritmética:
Endereço de RAM = 99
99 DIV 4 = 24 ; 24 MOD 4 = 0 -> bloco_cache = 0
99 MOD 4 = 2 -> offset 2 dentro da página (bloco)

Memória Cache com Mapeamento Associativo

- Bloco da RAM pode ser colocado em qualquer bloco da cache.
- Para acessar um dado na cache, procuramos por um bloco cuja tag seja igual à do seu endereço e, caso este bloco seja encontrado, buscamos o dado através do seu offset no bloco.
- Implementação é muito cara. Para que o acesso aos dados seja eficiente, é necessário que haja um comparador para cada bloco da cache (caso contrário, seria necessário buscar sequencialmente na cache pelo bloco cuja tag seja igual à do endereço a ser acessado).

- Endereçamento dos dados em memória cache com mapeamento associativo:

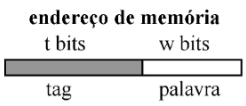
O offset para acessar um dado dentro de um bloco da cache é igual ao offset para acessar este dado dentro do respectivo bloco na RAM. Para calcularmos o número de bits necessários para o offset, fazemos o seguinte:

$$< \text{bits offset} > = \log_2 < \text{tamanho do bloco} >$$

Os bits seguintes serão usados para armazenara tag (que neste caso será igual ao número da página física na memória principal):

$$< \text{bits tag} > = < \text{bits endereço} > - < \text{bits offset} >$$

tag: t bits utilizados para **buscar** o bloco através de **endereçamento associativo**



tamanho do bloco = 2^w

palavra: w bits definem o **deslocamento (byte)** dentro do bloco

Memória Cache com Mapeamento Conjunto Associativo

- Bloco da RAM pode ser colocado em qualquer bloco de um único conjunto da cache.
- Os conjuntos de blocos são indexados (assim como os blocos no mapeamento direto) e cada bloco da RAM pode ser colocado em um único conjunto da cache.
- Dentro do conjunto, o bloco pode ocupar qualquer posição (assim como no mapeamento associativo).

- N-way set associative: cada conjunto de blocos da cache possui N blocos.

Para saber o tamanho dos conjuntos de blocos:

$$< \text{tamanho do conjunto} > = \text{N} * < \text{tamanho do bloco} >$$

Para saber o número de conjuntos da cache:

$$< \text{número de conjuntos} > = \frac{< \text{tamanho da cache} >}{< \text{tamanho do conjunto} >}$$

Para saber o número de bits necessários para indexar os conjuntos (bits de índice):

$$< \text{bits índice} > = \log_2 < \text{número de conjuntos} >$$

O offset para acessar um dado dentro de um bloco da cache é igual ao offset para acessar este dado dentro do respectivo bloco na RAM. Para calcularmos o número de bits necessários para o offset, fazemos o seguinte:

$$< \text{bits offset} > = \log_2 < \text{tamanho do bloco} >$$

Para encontrarmos um determinado bloco dentro do conjunto, precisamos comparar os <bits tag> bits mais significativos do endereço do dado sendo procurado com os <bits tag> bits mais significativos dos endereços dos blocos dentro do conjunto.

Sendo <bits endereço> o número total de bits de um endereço de memória, calculamos o número de bits de tag como

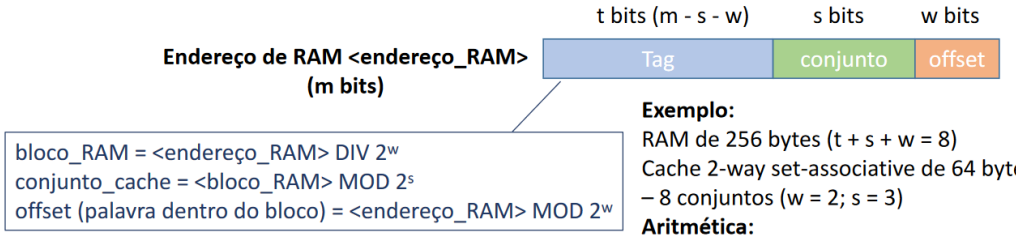
$$< \text{bits tag} > = < \text{bits endereço} > - < \text{bits índice} > - < \text{bits offset} >$$

Aritmética para determinar a **localização do conjunto**

- ❑ Dentro de um conjunto o bloco da RAM pode ocupar qualquer bloco
- ❑ A busca do bloco dentro do conjunto é feita de forma associativa (simultânea)

A) Separar os bits de endereço de conjunto a partir do endereço da RAM

B) Calcular usando operadores DIV e MOD (resultado da divisão inteira e resto da divisão inteira)



bloco_RAM = <endereço_RAM> DIV 2^w
conjunto_cache = <bloco_RAM> MOD 2^s
offset (palavra dentro do bloco) = <endereço_RAM> MOD 2^w

Exemplo:
RAM de 256 bytes (t + s + w = 8)
Cache 2-way set-associative de 64 bytes @ 4 bytes/bloco – 8 conjuntos (w = 2; s = 3)
Aritmética:
Endereço de RAM = 95
95 DIV 4 = 23 ; 23 MOD 8 = 7 -> conjunto_cache = 7
95 MOD 4 = 3 -> offset 3 do bloco com a tag indicada

Cache Misses

Classificação de **cache *misses***

- ❑ Cache *miss* **compulsório**
  - Primeiro acesso a um bloco na memória (não é possível evitar completamente)
- ❑ Cache *miss* de **capacidade**
  - Tamanho da cache insuficiente ou uso ineficiente do espaço disponível (subproduto da organização)
- ❑ Cache *miss* de **conflito**
  - Alocação imperfeita dos blocos no espaço disponível (subproduto da organização)

Efeitos dos parâmetros na **taxa de cache *miss***

Parâmetro da Cache	*Miss* Compulsório	*Miss* de Capacidade	*Miss* de Conflito	*Misses* Totais
Capacidade Menor	Sem efeito	Aumenta	Provavelmente aumenta	Provavelmente aumenta
Capacidade Maior	Sem efeito	Diminui	Provavelmente diminui	Provavelmente diminui
Bloco Menor	Aumenta	Provavelmente diminui	Provavelmente diminui	Variável
Bloco Maior	Diminui	Provavelmente aumenta	Provavelmente aumenta	Variável
Associatividade Menor	Sem efeito	Sem efeito	Provavelmente aumenta	Provavelmente aumenta
Associatividade Maior	Sem efeito	Sem efeito	Provavelmente diminui	Provavelmente diminui
Write-back vs. write-through	Sem efeito	Sem efeito	Sem efeito	Sem efeito
Write-no-allocate	Possivelmente diminui	Possivelmente diminui	Possivelmente diminui	Possivelmente diminui