Introdução à Informática

Alexandre Meslin
(meslin@nce.ufrj.br)

cederj

Organização da Memória

- Conceito de hierarquia de memória
- Memória principal e memórias secundárias
- Projeto lógico da memória principal
- Memórias cache
- Memória virtual

Conceito de Hierarquia de Memória

- Memória no computador
  - Registradores
  - Memória principal
  - **Memória cache**
  - Memória ROM
  - Discos magnéticos
  - Discos ópticos
  - Fitas magnéticas

Memória Cache

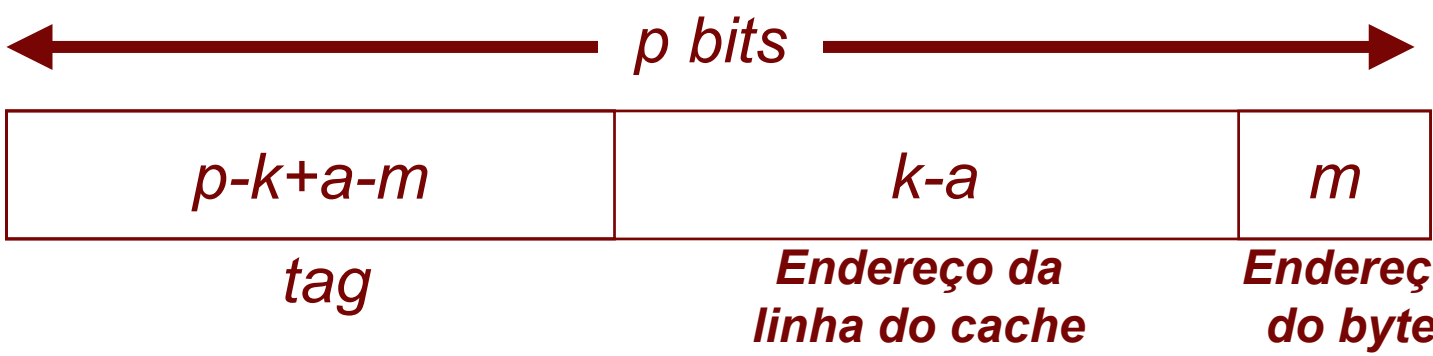
- Na aula anterior, foi visto:
  - Memória cache com mapeamento direto
  - Conflito de endereços na cache

Cache com Conjunto Associativo

- Semelhante ao mapeamento direto
- Bits mais baixos utilizados para selecionar linhas com conjuntos de 2, 4, 8 ou mais blocos
- Parte alta do endereço utilizada para garantir que realmente houve um acerto

Cache com Conjunto Associativo

- Assumindo que:
  - Memória cache tem 2^k linhas
  - Bloco com 2^m bytes
  - p bits de barramento de endereço
  - 2^a conjuntos associativos
- Bits mais baixos utilizados para selecionar a linha da cache
- Bits mais altos usados para comparar o endereço da linha



Cache Associativo

- Cache com *n* conjuntos associativos
  - Cada linha de cache pode ter *n* blocos
  - *n* pode ser pequeno (2, 4, 8)
  - Melhor desempenho
  - Custo moderado

Cache Associativo

- Como cada linha possui o mesmo endereço básico →
- Para selecionar a linha, basta calcular o seu endereço, com no exemplo de mapeamento direto
- Dentro de uma linha, vários blocos
- Para selecionar o bloco, deve se eleita uma política de substituição

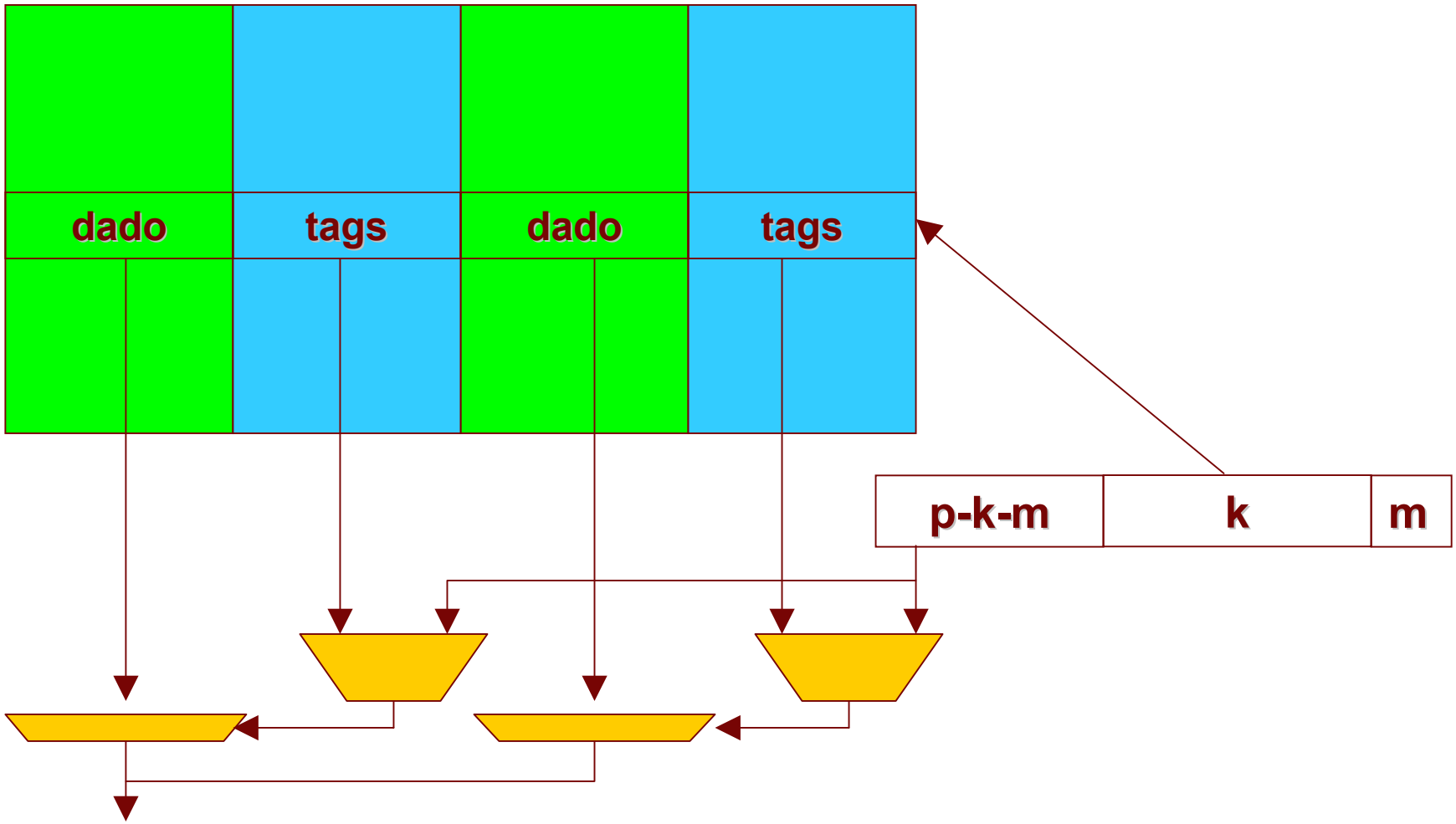
Política de Substituição de Bloco

- FIFO (First-in, First-out): primeiro a chegar, primeiro a sair
  - Se houver blocos vazios, qualquer um pode ser escolhido
  - Se todos estiverem ocupados, o mais antigo deve sair

Política de Substituição de Bloco

- LRU (least recently used): menos recentemente utilizado
  - Se houver blocos vazios, qualquer um pode ser escolhido
  - Se todos estiverem ocupados, o bloco que está a mais tempo sem ser utilizado será substituido

Cache com Conjunto Associativo



Exemplo

- Supondo memória cache com:
  - 256 bytes
  - 32 bytes / bloco
  - 8 linhas
  - Mapeamento direto
  - Barramento de endereços de 32 bits
  - Palavra de dados de 32 bits

Exemplo

- Comportamento de um programa:
  - Supor um programa que inicie no endereço 0
  - Muito provavelmente a seqüência de endereços de instruções será 0, 4, 8, 12, 16, 20, …
  - Inicialmente a cache está toda vazia (bit de validade em zero)

Exemplo

- Estado inicial da cache

Dado	Endereço	Validade
?	?	0
?	?	0
?	?	0
?	?	0
?	?	0
?	?	0
?	?	0
?	?	0

Exemplo

- Estado da cache depois do acesso 0 (**falha**)

Dado	Endereço	Validade
Inst 0 → 28	0	1
?	?	0
?	?	0
?	?	0
?	?	0
?	?	0
?	?	0
?	?	0

Exemplo

- Estado da cache depois do acesso 4 (acerto)

Dado	Endereço	Validade
Inst 0 → 28	0	1
?	?	0
?	?	0
?	?	0
?	?	0
?	?	0
?	?	0
?	?	0

Exemplo

- Estado da cache depois do acesso 8 (acerto)

Dado	Endereço	Validade
Inst 0 → 28	0	1
?	?	0
?	?	0
?	?	0
?	?	0
?	?	0
?	?	0
?	?	0

Exemplo

- Estado da cache depois do acesso 12 (acerto)

Dado	Endereço	Validade
Inst 0 → 28	0	1
?	?	0
?	?	0
?	?	0
?	?	0
?	?	0
?	?	0
?	?	0

Exemplo

- Estado da cache depois do acesso 16 (acerto)

Dado	Endereço	Validade
Inst 0 → 28	0	1
?	?	0
?	?	0
?	?	0
?	?	0
?	?	0
?	?	0
?	?	0

Exemplo

- Estado da cache depois do acesso 20 (acerto)

Dado	Endereço	Validade
Inst 0 → 28	0	1
?	?	0
?	?	0
?	?	0
?	?	0
?	?	0
?	?	0
?	?	0

Exemplo

- Estado da cache depois do acesso 24 (acerto)

Dado	Endereço	Validade
Inst 0 → 28	0	1
?	?	0
?	?	0
?	?	0
?	?	0
?	?	0
?	?	0
?	?	0

Exemplo

- Estado da cache depois do acesso 28 (acerto)

Dado	Endereço	Validade
Inst 0 → 28	0	1
?	?	0
?	?	0
?	?	0
?	?	0
?	?	0
?	?	0
?	?	0

Exemplo

- Estado da cache depois do acesso 32 (**falha**)

Dado	Endereço	Validade
Inst 0 → 32	0	1
Inst 32 → 60	32	1
?	?	0
?	?	0
?	?	0
?	?	0
?	?	0
?	?	0

Exemplo

- Estado da cache depois do acesso 36 (acerto).

Dado	Endereço	Validade
Inst 0 → 32	0	1
Inst 32 → 60	32	1
?	?	0
?	?	0
?	?	0
?	?	0
?	?	0
?	?	0

Outro Exemplo

- Este mesmo sistema, agora somando duas matrizes
- Cada matriz ocupa 256 bytes de memória
- Matriz 8x8 inteiros de 4 bytes
- Observe que a distância entre as matrizes pode ser múltipla do tamanho do cache
- Alto risco de haver conflito

Trecho do Programa-Exemplo

```
Para i ← 0 até 7 faça
    Para j ← 1 até 7 faça
        a(i,j) ← b(i,j)+c(i,j)
    fim para
fim para
```

- Este programa irá gerar (além de outros) acessos a endereços com 256 bytes de diferença para acessar as matrizes b e c

Estado da Cache

- No início

Dado	Endereço	Validade
?	?	0
?	?	0
?	?	0
?	?	0
?	?	0
?	?	0
?	?	0
?	?	0

Estado da Cache

- Leitura do dado b(0, 0) (**falha**)

Dado	Endereço	Validade
b(0,0)	End b(0,0)	1
?	?	0
?	?	0
?	?	0
?	?	0
?	?	0
?	?	0
?	?	0

Estado da Cache

- Leitura do dado c(0, 0) (**falha**)

Dado	Endereço	Validade
c(0,0)	End c(0,0)	1
?	?	0
?	?	0
?	?	0
?	?	0
?	?	0
?	?	0
?	?	0

Estado da Cache

- Leitura do dado b(0, 1) (**falha**)

Dado	Endereço	Validade
b(0,1)	End b(0,1)	1
?	?	0
?	?	0
?	?	0
?	?	0
?	?	0
?	?	0
?	?	0

Estado da Cache

- Leitura do dado c(0, 1) (**falha**)

Dado	Endereço	Validade
c(0,1)	End c(0,1)	1
?	?	0
?	?	0
?	?	0
?	?	0
?	?	0
?	?	0
?	?	0

Cache com Conjunto Associativo

- Mesmo sistema, mesmo programa, agora com cache com conjunto associativo
- Supondo memória cache com:
  - 256 bytes
  - 32 bytes / bloco
  - 4 linhas
  - 2 ou mais conjuntos associativos
  - Barramento de endereços de 32 bits
  - Palavra de dados de 32 bits

Estado da Cache

- No início

Dado	End.	Val.	Dado	End.	Val.
?	?	0	?	?	0
?	?	0	?	?	0
?	?	0	?	?	0
?	?	0	?	?	0

Estado da Cache

- Leitura do dado b(0,0) (**falha**)

Dado	End.	Val.	Dado	End.	Val.
b(0,0) b(0,7)	End. b(0,0)	1	?	?	0
?	?	0	?	?	0
?	?	0	?	?	0
?	?	0	?	?	0

Estado da Cache

- Leitura do dado c(0,0) (**falha**)

Dado	End.	Val.	Dado	End.	Val.
b(0,0) b(0,7)	End. b(0,0)	1	c(0,0) c(0,7)	End. c(0,0)	1
?	?	0	?	?	0
?	?	0	?	?	0
?	?	0	?	?	0

Estado da Cache

- Leitura do dado b(0,1) (acerto)

Dado	End.	Val.	Dado	End.	Val.
b(0,0) b(0,7)	End. b(0,0)	1	c(0,0) c(0,7)	End. c(0,0)	1
?	?	0	?	?	0
?	?	0	?	?	0
?	?	0	?	?	0

Estado da Cache

- Leitura do dado c(0,1) (acerto)

Dado	End.	Val.	Dado	End.	Val.
b(0,0) b(0,7)	End. b(0,0)	1	c(0,0) c(0,7)	End. c(0,0)	1
?	?	0	?	?	0
?	?	0	?	?	0
?	?	0	?	?	0

Estado da Cache

- Mais tarde…

Dado	End.	Val.	Dado	End.	Val.
b(0,0) b(0,7)	End. b(0,0)	1	c(0,0) c(0,7)	End. c(0,0)	1
?	?	0	?	?	0
?	?	0	?	?	0
?	?	0	?	?	0

Estado da Cache

- Leitura do dado b(1,0) (acerto)

Dado	End.	Val.	Dado	End.	Val.
b(0,0) b(0,7)	End. b(0,0)	1	c(0,0) c(0,7)	End. c(0,0)	1
b(1,0) b(1,7)	End. B(1,0)	0	?	?	0
?	?	0	?	?	0
?	?	0	?	?	0

Estado da Cache

- Leitura do dado c(1,0) (acerto).

Dado	End.	Val.	Dado	End.	Val.
b(0,0) b(0,7)	End. b(0,0)	1	c(0,0) c(0,7)	End. c(0,0)	1
b(1,0) b(1,7)	End. B(1,0)	0	c(1,0) c(1,7)	End. c(1,0)	0
?	?	0	?	?	0
?	?	0	?	?	0

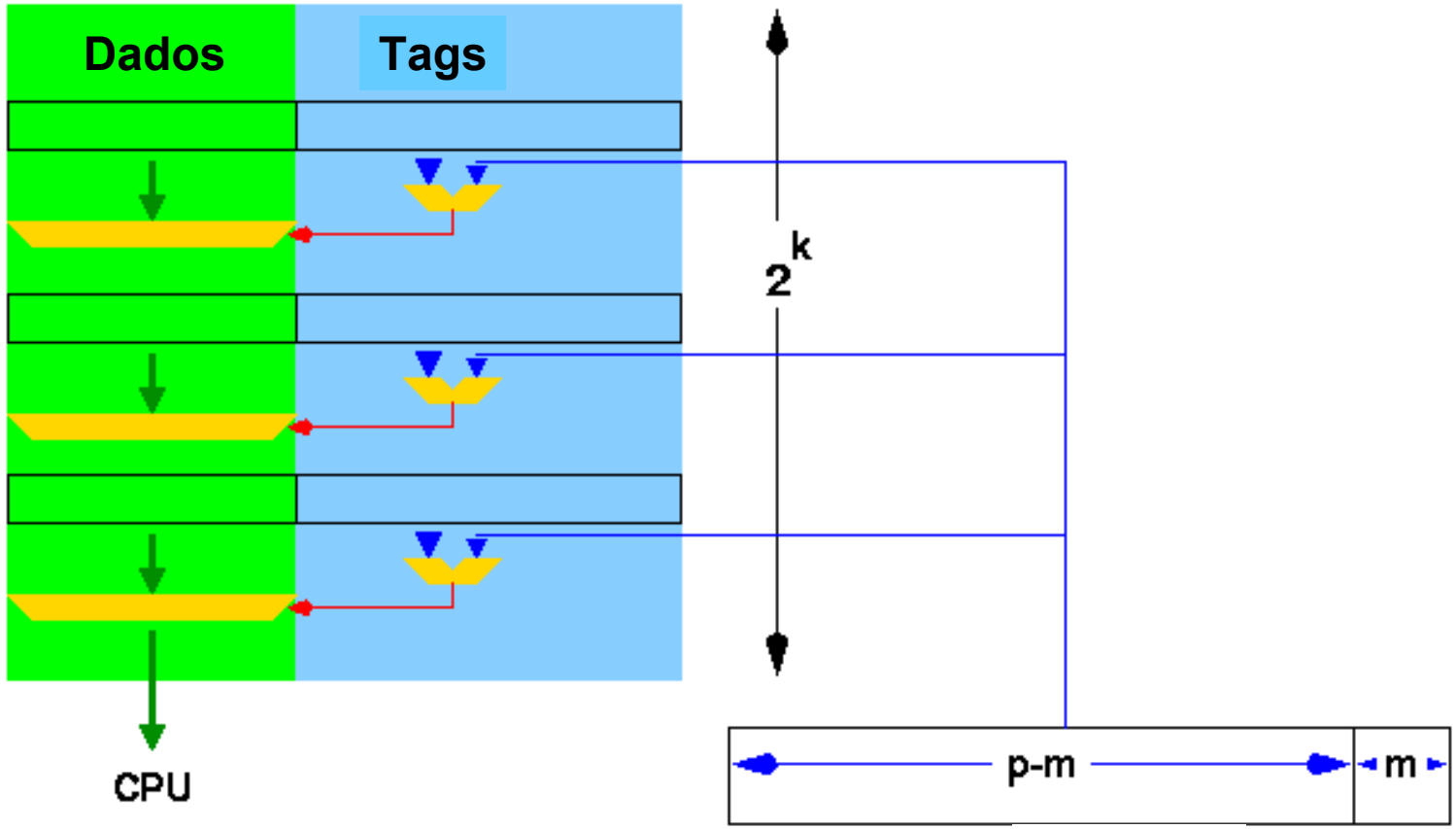
Cache Totalmente Associativo

- Pedido da CPU procurado em todo o cache
  - Se encontrar: acerto na cache (cache hit)
  - Caso contrário: falha na cache (cache miss)
- Todas as tags são comparadas ao mesmo tempo
- Qualquer endereço de memória pode estar sendo representado em qualquer linha da cache
- Evita substituição desnecessária

Cache Totalmente Associativo

- Política de substituição
  - LRU
  - FIFO
- Necessita de muitos bits adicionais de estado (na tag)
- Custo de hardware muito alto
  - Muitos comparadores
  - Tags muito grandes (muitos bits)

Cache Totalmente Associativo



Política de Escrita – Write Through

- Write Through
  - Todo acesso de escrita com acerto é feito simultaneamente na cache e na memória principal
  - Se houver um acesso de escrita com falha, somente a memória principal será modificada
  - Informação na memória principal está sempre atualizada
  - Leitura pode ser feita em um único ciclo
  - Escrita demora o tempo de acesso à memória principal

Política de Escrita – Write Back

- Write Back
  - Todo acesso de escrita com acerto é feito somente na cache
  - O que ocorre em caso de falha dependerá da política de alocação (ver mais tarde)
  - Informação na memória principal pode estar desatualizada
  - Leitura e escrita podem ser feitas em um único ciclo
  - Se o bloco for substituído e a sua informação estiver sido modificada, esta deverá ser enviada para a memória principal

Outros Campos da Tag

- Bit de “dirty” (sujo)
- Utilizado para controlar o estado da memória principal
  - Coerente → memória atualizada
  - Incoerente → memória desatualizada
- Caso a memória esteja incoerente, este bit indicará que o bloco na cache está sujo
- Em caso de substituição deste bloco, os dados deverão ser escritos na memória

Política de Alocação

- O que acontece em caso de falha de escrita em um cache write back?
  1. A informação pode ser lida da memória principal para a cache e então modificada
  2. A escrita pode ser feita diretamente na memória principal

Write Back x Write Through

- Qual o melhor?
- Write back
  - Escritas mais rápidas
- Write through
  - Bom para multiprocessamento (memória sempre coerente)
  - Necessita de menos hardware

Próxima Aula

- Continuação da hierarquia de memória