Hierarquia de Memória - Parte 2 - Memória Cache

Arquitetura de Computadores

Emilio Francesquini
e.francesquini@ufabc.edu.br

2020.Q1

Centro de Matemática, Computação e Cognição
Universidade Federal do ABC

UFABC

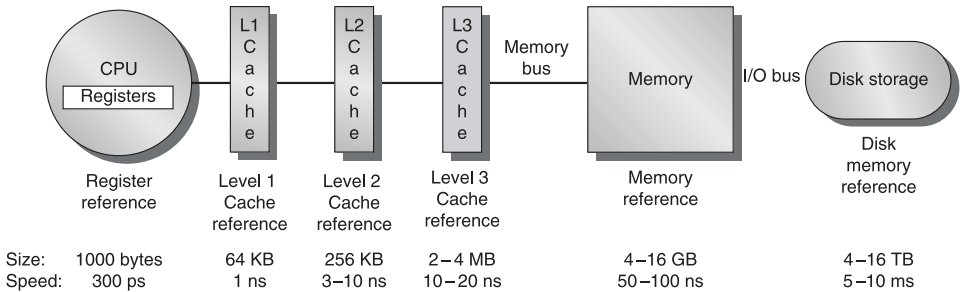
Disclaimer

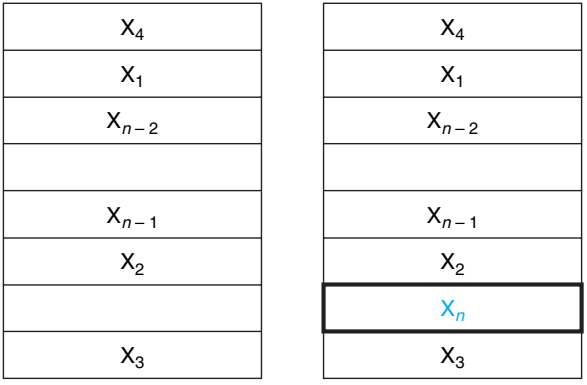
UFABC

- Estes slides foram preparados para o curso de Arquitetura de Computadores na UFABC.
- Este material pode ser usado livremente desde que sejam mantidos, além deste aviso, os créditos aos autores e instituições.
- O conteúdo destes slides foi baseado no conteúdo do livro *Computer Organization And Design: The Hardware/Software Interface*, 5th Edition.

Introdução

- Memória cache (*cache memory*)
  - Nível de memória mais próximo à CPU





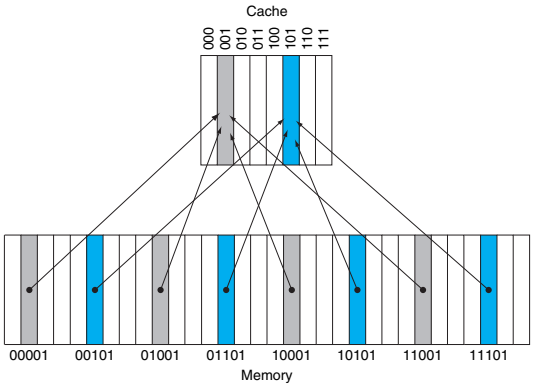
a. Before the reference to X_n b. After the reference to X_n

Dados acessos $X_1, X_2, \ldots X_{n-1}, X_n$

- Como saber se o dado está na cache?
- Como saber onde procurar?

Caches com mapeamento direto

- A localicação dentro da cache é determinada pelo endereço
- Em caches com mapeamento direto (*direct mapped*) há apenas uma escolha de localização possível.
  - (Endereço da linha) módulo (Nº de blocos da cache)
  - Nº de blocos da cache é uma potência de 2
  - Utiliza-se o bits menos significativos do endereço



Tags e *Valid* Bits

UFABC

- Como saber se um bloco/linha está localizada em uma determinada posição?
  - Precisamos guardar a qual posição da memória cada bloco se refere, ou seja, armazenamos tanto o endereço quanto o dado
  - Sendo justo, precisamos apenas os bits mais significativos
    - Esses bits mais significativos são chamados tag
- E se não houver nenhum dado guardado naquela posição?
  - Incluímos um bit que indica se a linha é válida
    - Se 1, linha presente. Se 0, linha não presente.
    - Inicalmente todos valem 0.

Exemplo de acesso à cache

UFABC

- Vamos assumir uma cache com 8 blocos/linhas, com mapeamento direto
- O estado inicial é:

Index	V	Tag	Data
000	N		
001	N		
010	N		
011	N		
100	N		
101	N		
110	N		
111	N		

Exemplo de acesso à cache - Acesso 1

Endereço em Decimal	Endereço em Binário	Linha de Cache End. *mod* 8_{10}
22	10110	110

- Miss - Carrega linha

Index	V	Tag	Data
000	N		
001	N		
010	N		
011	N		
100	N		
101	N		
110	Y	10_{two}	Memory (10110_{two})
111	N		

Exemplo de acesso à cache - Acesso 2

Endereço em Decimal	Endereço em Binário	Linha de Cache End. *mod* 8_{10}
26	11010	010

- Miss - Carrega linha

Index	V	Tag	Data
000	N		
001	N		
010	Y	11_{two}	Memory (11010_{two})
011	N		
100	N		
101	N		
110	Y	10_{two}	Memory (10110_{two})
111	N		

Exemplo de acesso à cache - Acesso 3

Endereço em Decimal	Endereço em Binário	Linha de Cache End. *mod* 8_{10}
22	10110	110

- Hit - não há mudança de estado

Index	V	Tag	Data
000	N		
001	N		
010	Y	11_{two}	Memory (11010_{two})
011	N		
100	N		
101	N		
110	Y	10_{two}	Memory (10110_{two})
111	N		

Exemplo de acesso à cache - Acesso 4

Endereço em Decimal	Endereço em Binário	Linha de Cache End. *mod* 8_{10}
26	11010	010

- Hit - não há mudança de estado

Index	V	Tag	Data
000	N		
001	N		
010	Y	11_{two}	Memory (11010_{two})
011	N		
100	N		
101	N		
110	Y	10_{two}	Memory (10110_{two})
111	N		

Exemplo de acesso à cache - Acesso 5

Endereço em Decimal	Endereço em Binário	Linha de Cache End. *mod* 8_{10}
16	10000	000

- Miss - Carrega linha

Index	V	Tag	Data
000	Y	10_{two}	Memory (10000_{two})
001	N		
010	Y	11_{two}	Memory (11010_{two})
011	N		
100	N		
101	N		
110	Y	10_{two}	Memory (10110_{two})
111	N		

Exemplo de acesso à cache - Acesso 6

Endereço em Decimal	Endereço em Binário	Linha de Cache End. *mod* 8_{10}
3	00011	011

- Miss - Carrega linha

Index	V	Tag	Data
000	Y	10_{two}	Memory (10000_{two})
001	N		
010	Y	11_{two}	Memory (11010_{two})
011	Y	00_{two}	Memory (00011_{two})
100	N		
101	N		
110	Y	10_{two}	Memory (10110_{two})
111	N		

Exemplo de acesso à cache - Acesso 7

Endereço em Decimal	Endereço em Binário	Linha de Cache End. *mod* 8_{10}
16	10000	000

- Hit - não há mudança de estado

Index	V	Tag	Data
000	Y	10_{two}	Memory (10000_{two})
001	N		
010	Y	11_{two}	Memory (11010_{two})
011	Y	00_{two}	Memory (00011_{two})
100	N		
101	N		
110	Y	10_{two}	Memory (10110_{two})
111	N		

Exemplo de acesso à cache - Acesso 8

Endereço em Decimal	Endereço em Binário	Linha de Cache End. *mod* 8_{10}
18	10010	010

- Miss - Carrega linha

Index	V	Tag	Data
000	Y	10_{two}	Memory (10000_{two})
001	N		
010	Y	10_{two}	Memory (10010_{two})
011	Y	00_{two}	Memory (00011_{two})
100	N		
101	N		
110	Y	10_{two}	Memory (10110_{two})
111	N		

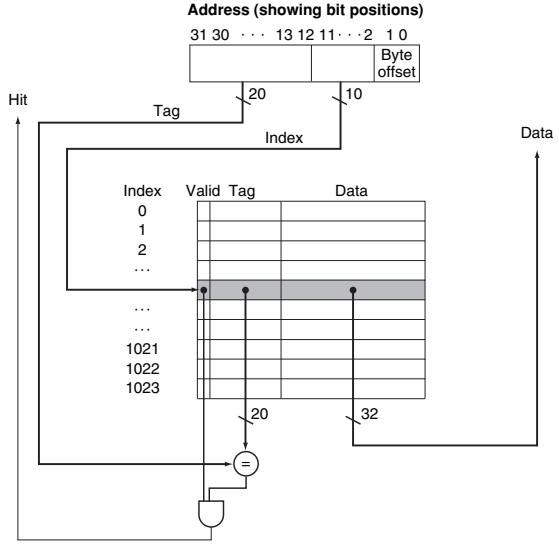
Exemplo de acesso à cache - Acesso 9

Endereço em Decimal	Endereço em Binário	Linha de Cache End. *mod* 8_{10}
16	10000	000

- Hit - Não há mudança de estado.

Index	V	Tag	Data
000	Y	10_{two}	Memory (10000_{two})
001	N		
010	Y	10_{two}	Memory (10010_{two})
011	Y	00_{two}	Memory (00011_{two})
100	N		
101	N		
110	Y	10_{two}	Memory (10110_{two})
111	N		

Divisão dos endereços

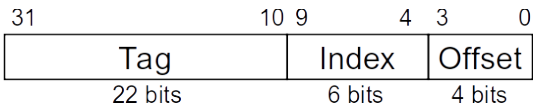


Exemplo com um bloco maior

- Considere uma cache com mapeamento direto com 64 blocos
- Cada bloco com 16 bytes (4 palavras)

Para qual bloco da cache o endereço 1200 deve ser mapeado?

- Bloco = $\left\lfloor \frac{1200}{16} \right\rfloor = 75$
- Nº do Bloco na Cache = $75 \mod 64 = 11$

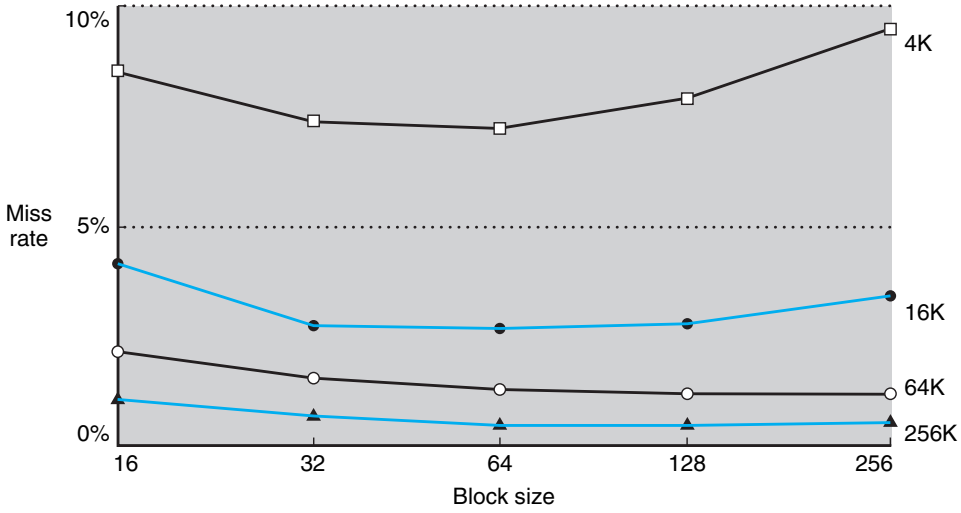


Algumas considerações sobre o tamanho do bloco

UFABC

- Blocos maiores tendem a diminuir a taxa de cache misses
  - Devido a localidade espacial
- Mas, dado um tamanho fixo de cache:
  - Quanto maior o bloco $\rightarrow$ Menos blocos
    - Mais competição por blocos $\rightarrow$ aumento na taxa de cache misses
  - Maior custo de acesso à memória
    - Pode desperdiçar os ganhos de uma taxa de misses mais baixa
    - *Early restart* e *critical-word-first* podem ajudar

Algumas considerações sobre o tamanho do bloco



Resultados do SPEC92

Falhas de cache

- Quando há um cache hit
  - CPU prossegue normalmente
- Quando há um cache miss
  - Stall no pipeline
  - Busca o bloco requerido do próximo nível da hierarquia
  - Miss de instrução
    - Reinicia o fetch da instrução
  - Miss de dados
    - Completa o acesso aos dados

Políticas de escrita na cache - *Write-through*

UFABC

- Quando houver um hit, durante uma escrita, poderia simplesmente atualizar o bloco na cache
  - Mas neste caso a cache e a memória ficariam inconsistentes
- Na política de ***write-through*** quando houver uma escrita também atualizamos a memória
- Isso, contudo, atrasa as escritas
  - Por exemplo: se a o CPI base for 1, 10% das instruções forem stores, e escrever na memória levar 100 ciclos
    - CPI = $1 + 0.1 * 100 = 11$
- Solução: write buffer
  - Mantém os dados esperando para ser escritos para a memória
  - CPU continua a execução sem pausas
    - Stalls ocorrem apenas quando o write buffer estiver cheio

Políticas de escrita na cache - *Write-back*

UFABC

- Alternativa à política de write-through: *write-back*
  - Apenas atualiza a linha na cache
  - Mantém um controle se cada um dos blocos está sujo (*dirty*) ou limpo (*clean*).
- Quando um bloco sujo precisar sair da cache
  - Escreve o bloco de volta à memória
  - Pode ser usada com um write buffer para permitir que a substituição do bloco sujo por um limpo ocorra antes da escrita na memória

Alocacão de escritas

UFABC

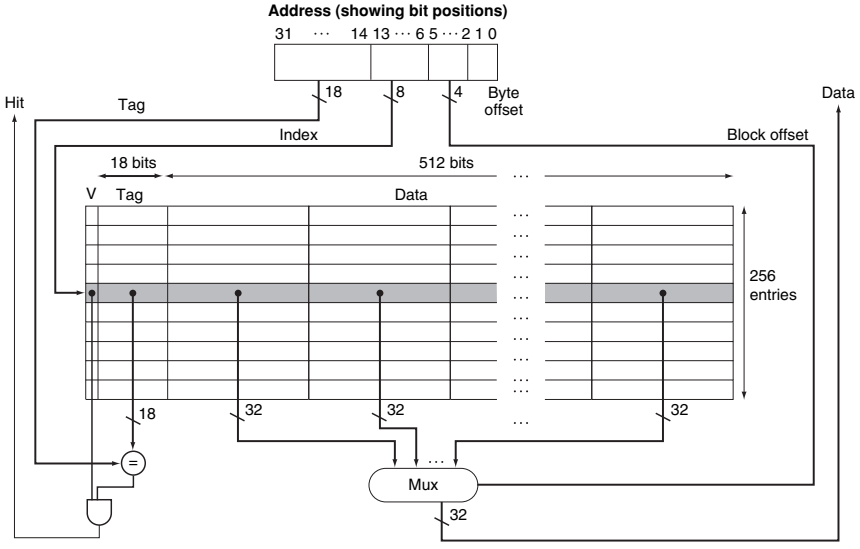
- O que devemos fazer quando queremos escrever em um bloco que não está na cache?
- Write-through
  - *Allocate on miss*: busca o bloco e escreve
  - *Write around*: escreve diretamente na memória
    - Útil pois programas frequentemente escrevem na memória e não leem logo em seguida (ex. inicialização com zeros).
- Write-back
  - Normalmente busca o bloco antes

Exemplo real: O processador Intrinsity FastMATH

UFABC

- É um processador MIPS embarcado
  - Pipeline com 12 estágios
  - Acesso às instruções e dados a cada ciclo
- *Split-cache*: Caches de dados e instruções separadas
  - Normalmente chamadas de I-cache e D-cache
  - No FastMath cada uma tem 16KB: 256 blocos $\times$ 16 palavras/bloco
  - D-cache pode ser write-through ou write-back
- Taxa de cache misses no SPEC2000
  - I-cache: 0.4%
  - D-cache: 11.4%
  - Média ponderada: 3.2%

Exemplo real: O processador Intrinsity FastMATH



Caches são realmente necessárias?

UFABC

- Usamos DRAM para memória principal
  - Normalmente tem largura fixa (ex. 1 palavra)
  - Conectadas por um barramento com clock de largura fixa
    - O clock do bus é tipicamente mais lento do que o clock da CPU
  - Exemplo: leitura de um bloco
    - 1 clock do bus para a transferência do endereço
    - 15 clocks do bus por acesso à DRAM
    - 1 clock do bus por palavra transferida
- Para um bloco de 4 palavras, em uma RAM como acima teríamos
  - Miss penalty = $1 + 4 \times 15 + 4 \times 1 = 65$ ciclos do bus
  - Largura de banda = $16\text{bytes}/65\text{ciclos} = 0.25\text{B/ciclo}$

Vamos olhar com mais cuidado como medir o desempenho da cache

UFABC

- Componentes do tempo de CPU
  - Número de ciclos do programa
    - Já incluem os tempos de hits nas caches
  - Ciclos em stall
    - Em sua maioria por cache misses
- Então, de maneira simplificada

$$\text{Ciclos em stall} = \frac{\text{Acessos à memória}}{\text{Programa}} \times \text{Taxa de misses} \times \text{Miss penalty}$$

que é equivalente a

$$\text{Ciclos em stall} = \frac{\text{Instruções}}{\text{Programa}} \times \frac{\text{Misses}}{\text{Instrução}} \times \text{Miss penalty}$$

Exemplos de desempenho de uma cache

UFABC

- Dados
  - I-cache, taxa de misses = 2%
  - D-cache, taxa de misses = 4%
  - Miss penalty = 100 ciclos
  - CPI base (cache ideal) = 2
  - Loads e Stores compõem 36% de todas as instruções
- Ciclos gastos em misses por instrução
  - I-cache: $0.02 \times 100 = 2$
  - D-cache: $0.36 \times 0.04 \times 100 = 1.44$
- CPI real $= 2 + 2 + 1.44 = 5.44$
  - CPU ideal é $\frac{5.44}{2} = 2.72$ vezes mais rápida

UFABC

- O tempo do hit é essencial a um bom desempenho
- O tempo médio de acesso à memória (AMAT na sigla em inglês)
  - AMAT = Tempo Hit + Taxa de Miss × Miss penalty
- Exemplo:
  - CPU com clock de 1ns, tempo hit = 1 ciclo, miss penalty = 20 ciclos, taxa de miss da I-cache 5%
  - AMAT $= 1 + 0.05 \times 20 = 2ns$
  - Ou, 2 ciclos por instrução

Resumo da análise de desempenho

UFABC

- Quanto mais rápida a CPU $\rightarrow$ maior o impacto de cada cache miss
- Aumento do CPI indica uma proporção maior do tempo gasta em stalls causados por acesso à memória
- Conforme a frequência do processador aumenta $\rightarrow$ maior o número de ciclos gastos em stalls
- É essencial que prestemos atenção ao comportamento da cache para avaliar o desempenho do sistema.