Fundamentos de Arquitetura de Computadores

Tiago Alves

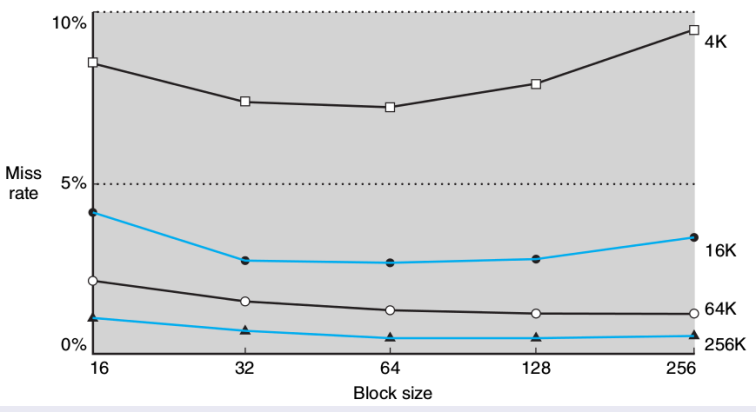
Faculdade UnB Gama
Universidade de Brasília

Visão Geral

- Introdução
- Princípios Básicos da Cache
- **Desempenho da Cache**
- **Memória Virtual**
- **Hierarquia de Memória**

Acertos x Falhas

Aumentar os tamanhos de bloco tende a reduzir a taxa de falhas:



Desempenho

- Modelo simplificado:

  Tempo de execução = (ciclos de execução + ciclos de stall) × tempo de ciclo

  Ciclos de stall = número de instruções × taxa de falhas × penalidade de falha;
- Alternativas para melhorar o desempenho:
  - reduzir a taxa de falhas
  - reduzir a penalidade de falhas.
- Quais são os efeitos de se aumentar o tamanho do bloco básico da cache?

Desempenho

- Ciclos de stall na memória:

  ```
  ciclos_de_stall_memoria = ciclos_stall_leitura + ciclos_stall_escrita
  ```

- A análise da leitura é mais simples. Stalls provocados por:
  - acesso de leitura e
  - penalidade de falha.

$$\text{Ciclos\_stall\_leitura} = \frac{\text{Leituras}}{\text{Programa}} \times \text{Taxa\_falhas\_leitura} \times \text{Penalidade\_falha\_leitura}$$

Desempenho

- Pode-se combinar taxas de falhas de leituras e escritas:

$$\text{Ciclos\_stall\_memoria} = \frac{\text{Acesso\_memoria}}{\text{Programa}} \times \text{Taxa\_falhas} \times \text{Penalidade\_falhas}$$

- Que pode ser fatorado como:

$$\text{Ciclos\_stall\_memoria} = \frac{\text{Instrucoes}}{\text{Programa}} \times \frac{\text{Falhas}}{\text{Instrucao}} \times \text{Penalidade\_falhas}$$

Desempenho

- Taxa de falhas de cache de instruções de um programa em 2%. A taxa de falha de dados de cache 4%. Um processador possui um CPI 2 sem qualquer stall de memória e a penalidade de falha é de 100 ciclos para todas as falhas.
- Determine o quanto mais rápido um processador executaria com uma cache perfeita (sem falhas).
- Workload (SPECint2000) de 36% para acessos à memória (dados).

Desempenho

- Número de ciclos de falha de instrução para I instruções:

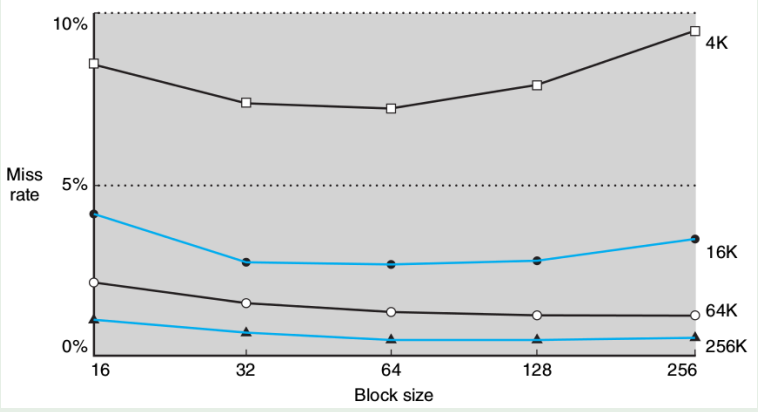
  Ciclos_falha_instrução = I x 2% x 100 = 2xI
- Supondo a freqüência de todos os loads e stores de 36% (Ex.: SPECint2000)

  Ciclos_falha_dados = I x 36% x 4% x 100 = 1,44 x I

  Ciclos_total_memoria = 2 x I + 1,44 x I = 3,44 x I
- Assim a CPI com stalls de memória é 2 + 3,44 = 5,44

Desempenho

Como não há mudança no número de instruções ou na velocidade do clock, temos:



Desempenho

Pode-se melhorar o desempenho a partir da velocidade de clock e da CPI Consequências:

- Quanto menor a CPI, maior será o impacto
- Se as memórias principais tiverem o mesmo tempo de acesso absolutos, uma velocidade maior produzirá uma penalidade de falha maior.

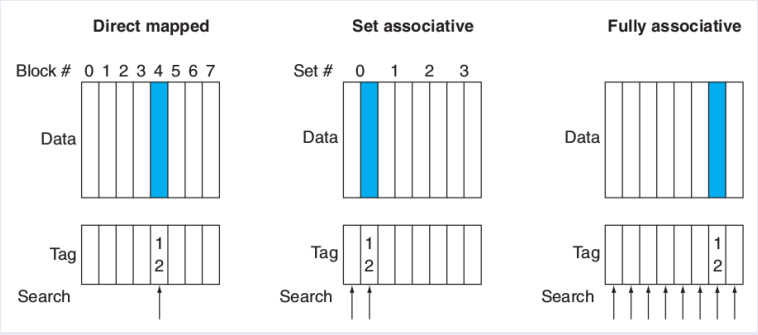
Reduzindo Falhas

Três estratégias.

- Em uma extremidade: utilizamos o mapeamento direto

  O bloco só pode ser colocado exatamente em um local;
- Em outra extremidade: Totalmente Associativo.
  - Bloco posicionado em qualquer local na cache;
  - Faz-se buscas pelo bloco (funcional para pequenas caches!);
- Na faixa intermediária: Associativa por conjunto
  - Existem locais fixos (mapeamento direto);
  - n locais para uma cache de n vias;
  - Cada bloco é mapeado para um conjunto único da cache, determinado pelo índice, e um bloco pode ser colocado em qualquer elemento desse conjunto;

Reduzindo a Taxas de Falhas com Associatividade



Mapeamento Direto x Associativo por Conjunto

- Mapeamento Direto:

  `Numero_bloco mod Capacidade_blocos_cache`

- Associativa por Conjunto:

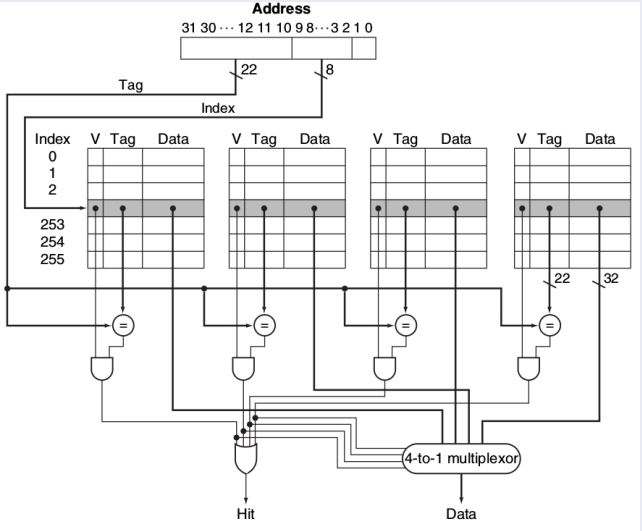
  `Numero_bloco mod Numero_Conjuntos_cache`

Localizando um Bloco na Cache

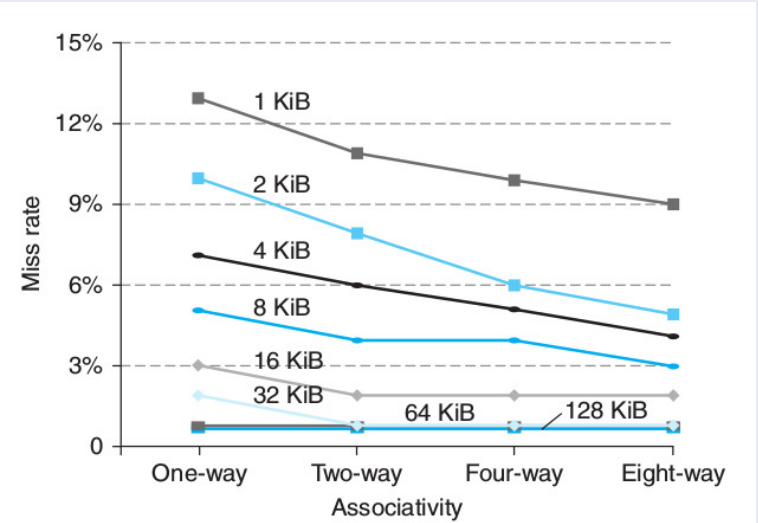
Na Associativa por Conjunto:

- Como na diretamente mapeada, cada bloco inclui uma tag de endereço para o endereço do bloco;
- Possui índice para selecionar o conjunto de interesse;
- Offset é o endereço dos dados desejados dentro do bloco;

Implementação Viável para 4 conjuntos



Desempenho x Associatividade



Reduzindo as penalidades de Falhas com Caches Multiníveis

Acrescente um segundo nível de cache:

- Normalmente, o cache primário está no mesmo chip od processador.
- Use SRAMs para acrescentar outro cache acima (antes) da memória principal.
- A penalidade de falhas diminui se os dados estiverem no cache de segundo nível.

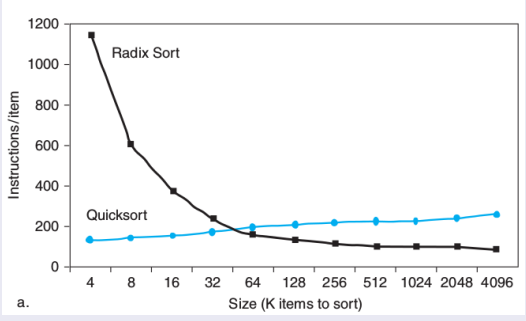
Exemplo:

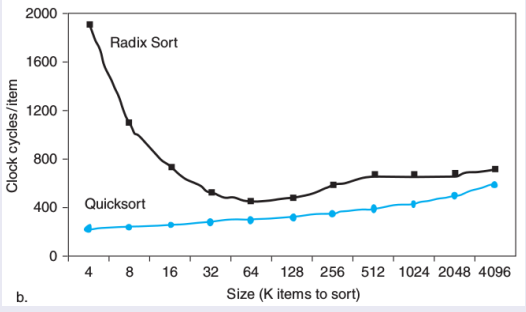
- CPI de 1,0 em uma máquina de 5 GHz com taxa de falhas de 5%, acesso a DRAM de 100 ns.
- Acrescentar cache de segundo nível com tempo de acesso de 5 ns diminui a taxa de falhas para 0,5 %.

Usando cache multinível:

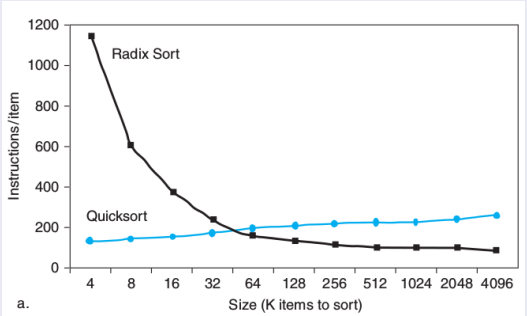
- Tente otimizar o tempo de acerto no cache de primeiro nível.
- Tente otimizar a taxa de falhas no cache de segundo nível.

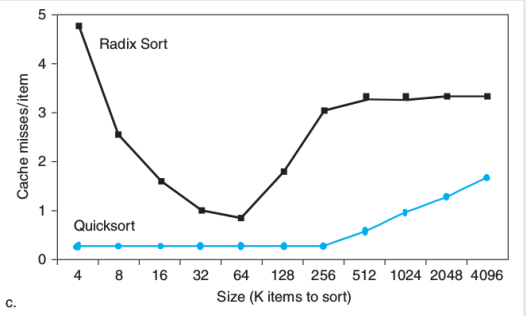
Complexidades da Cache





Complexidades da Cache





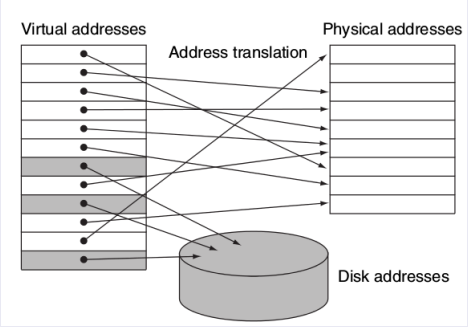
O desempenho do sistema de memória é, normalmente, um fator crítico:

- Caches multinível e processadores em pipeline dificultam a previsão de resultados (baseados exclusivamente pela análise de complexidade algorítmica.)
- Otimizações de compilador aumentam a localidade e prejudicam, algumas vezes, o ILP (Camada de Integração de Processamento).

Logo, fica difícil de prever, baseado exclusivamente em complexidade algorítmica, qual é o melhor algoritmo, pois a decisão poderá ser fortemente afetada pela implementação em software e pela arquitetura de hardware do sistema computacional!

Memória Virtual

A memória principal pode atual como um cache para o armazenamento secundário.



Vantagens:

- ilusão de se ter mais memória física do que o volume instalado de memória RAM;
- remanejamento do programa;
- proteção.

Memória Virtual

Objetivos básicos:

- **Recolocação**: auxiliar diretamente para que tenha seu próprio espaço de endereçamento;
- **Proteção**: impedir o uso indevido de endereços;
- **Swapping**: utilizar mais memória física do que existe.

Paginação

Uma página é o bloco básico de memória virtual.

Falhas de página: semelhante a falhas de cache. Os dados não estão na memória (armazenamento de alta velocidade); recupere-os do disco (armazenamento de baixa velocidade).

- penalidade alta em decorrência da falha. Boa prática é ajustar tamanhos de páginas grandes o suficiente para compensar latências decorrentes de acesso a disco. Tamanhos típicos em torno de 4 KB.
- Faz-se necessário reduzir as falhas de página. Há heurísticas de reposicionamento de quadros em memória que tratam desse problema. Algoritmo LRU.
- Pode manipular as falhas no software em vez de no hardware.
- Quando-se se vai tratar problemas de consistência devido a eventos de escritas em páginas, adota-se a técnica de escrita adiada para se otimizar a manipulação de memória virtual. (Escrita direta é muito onerosa.)

Paginação

Endereçamento Lógico (Virtual) vs. Endereçamento Físico

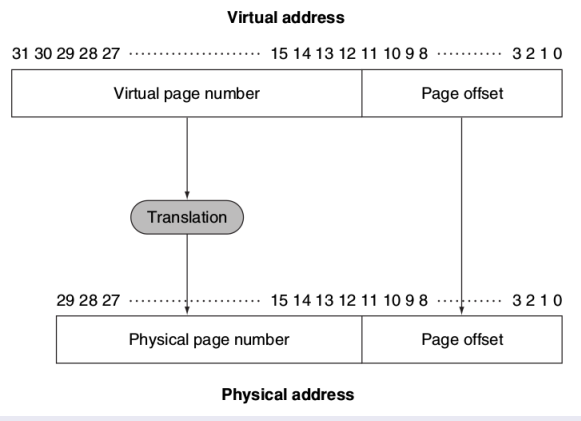


Tabela de Páginas

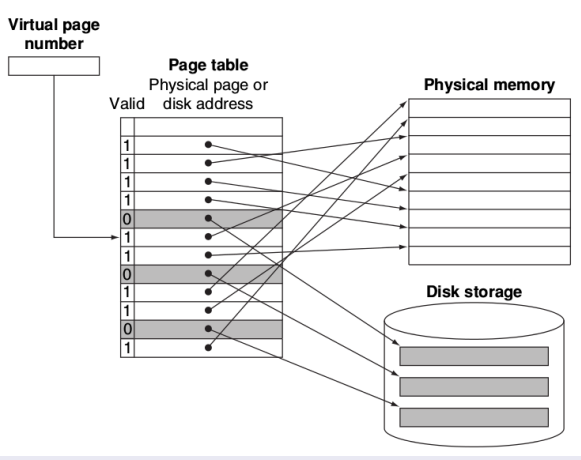


Tabela de Páginas

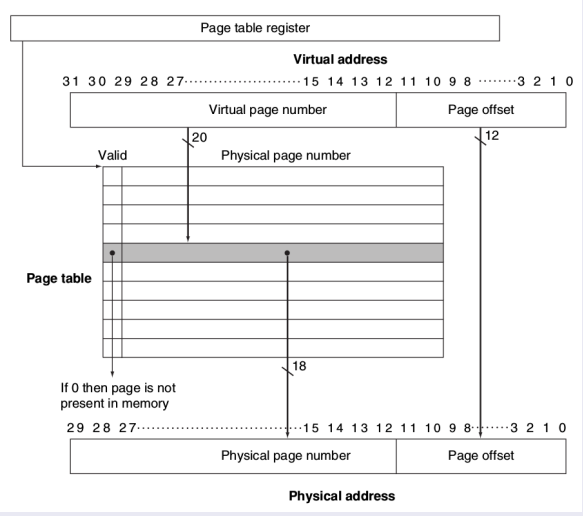
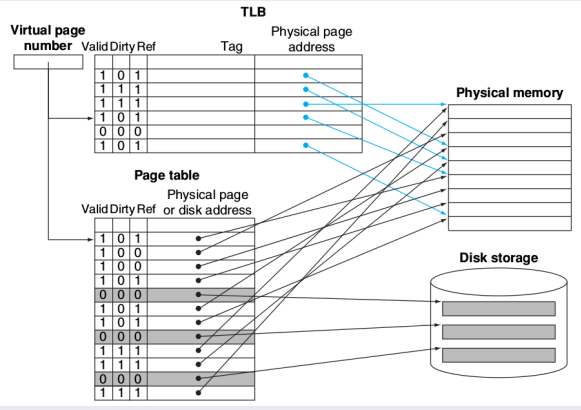
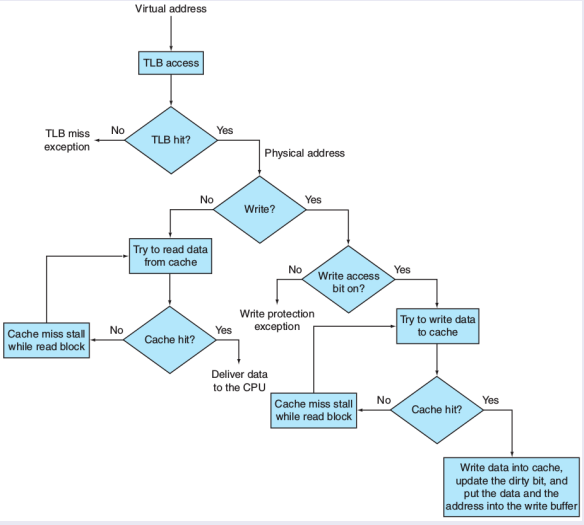


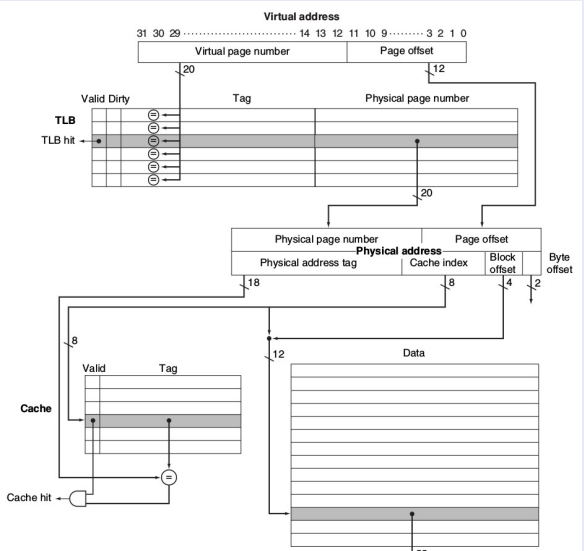
Tabela de Páginas



TLB e Caches


Virtual address
TLB access
TLB hit?
No
TLB miss exception
Yes
Physical address
Write?
No
Yes
Try to read data from cache
Cache hit?
No
Cache miss stall while read block
Yes
Deliver data to the CPU
Write access bit on?
No
Write protection exception
Yes
Try to write data to cache
Cache hit?
No
Cache miss stall while read block
Yes
Write data into cache, update the dirty bit, and put the data and the address into the write buffer

TLB e Caches


Virtual address
31 30 29 ····· 14 13 12 11 10 9 ····· 3 2 1 0
Virtual page number
Page offset
20
12
Valid Dirty
Tag
Physical page number
TLB
TLB hit
20
Physical page number
Page offset
Physical address
Physical address tag
Cache index
Block offset
Byte offset
18
8
4
2
8
12
Data
Valid
Tag
Cache
Cache hit

Alguns Problemas

As velocidades dos processadores continuam a aumentar muito rápido.

- Muito mais rápido do que os tempos de acesso à DRAM ou ao disco.

Desafio de projeto: como lidar com essa crescente disparidade? Prefetching? Caches de terceiro nível (ou mais)? Projeto de memória?