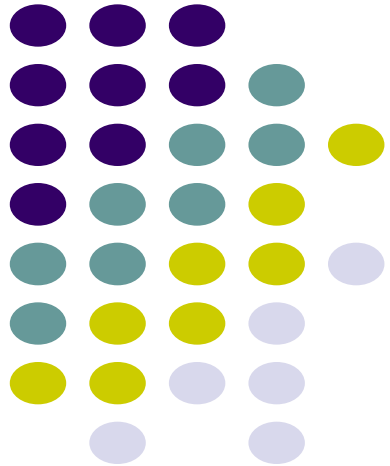
UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO
DEPARTAMENTO DE CIÊNCIAS EXATAS E NATURAIS
CURSO DE CIÊNCIA DA COMPUTAÇÃO

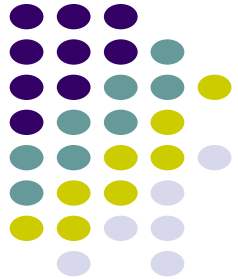
Arquitetura e Organização de Computadores

Hierarquia de Memória - Parte III



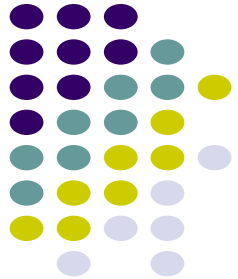
Prof. Sílvio Fernandes

Tratando falhas de cache



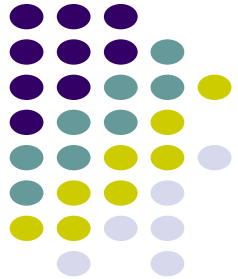
- **Falha de cache:** uma requisição de dados da cache que não pode ser atendida porque os dados não estão presentes na cache
- A unidade de controle precisa detectar uma falha e processá-la buscando os dados requisitados da memória (ou cache de nível inferior)
  - Se a cache reportar um acerto, o computador continua usando os dados como se nada tivesse acontecido

Tratando falhas de cache



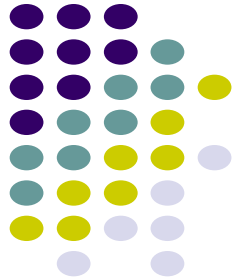
- Modificar o controle de um processador para tratar um acerto é fácil; as falhas, no entanto, exigem um trabalho maior
- O tratamento da falha de cache é feito com a unidade de controle do processador e com um controlador separado que inicia o acesso à memória e preenche novamente a cache
- O processamento da falha cria um stall, basicamente congelando o conteúdo dos registradores temporários e visíveis ao programador, enquanto esperamos a memória

Tratando falhas de cache



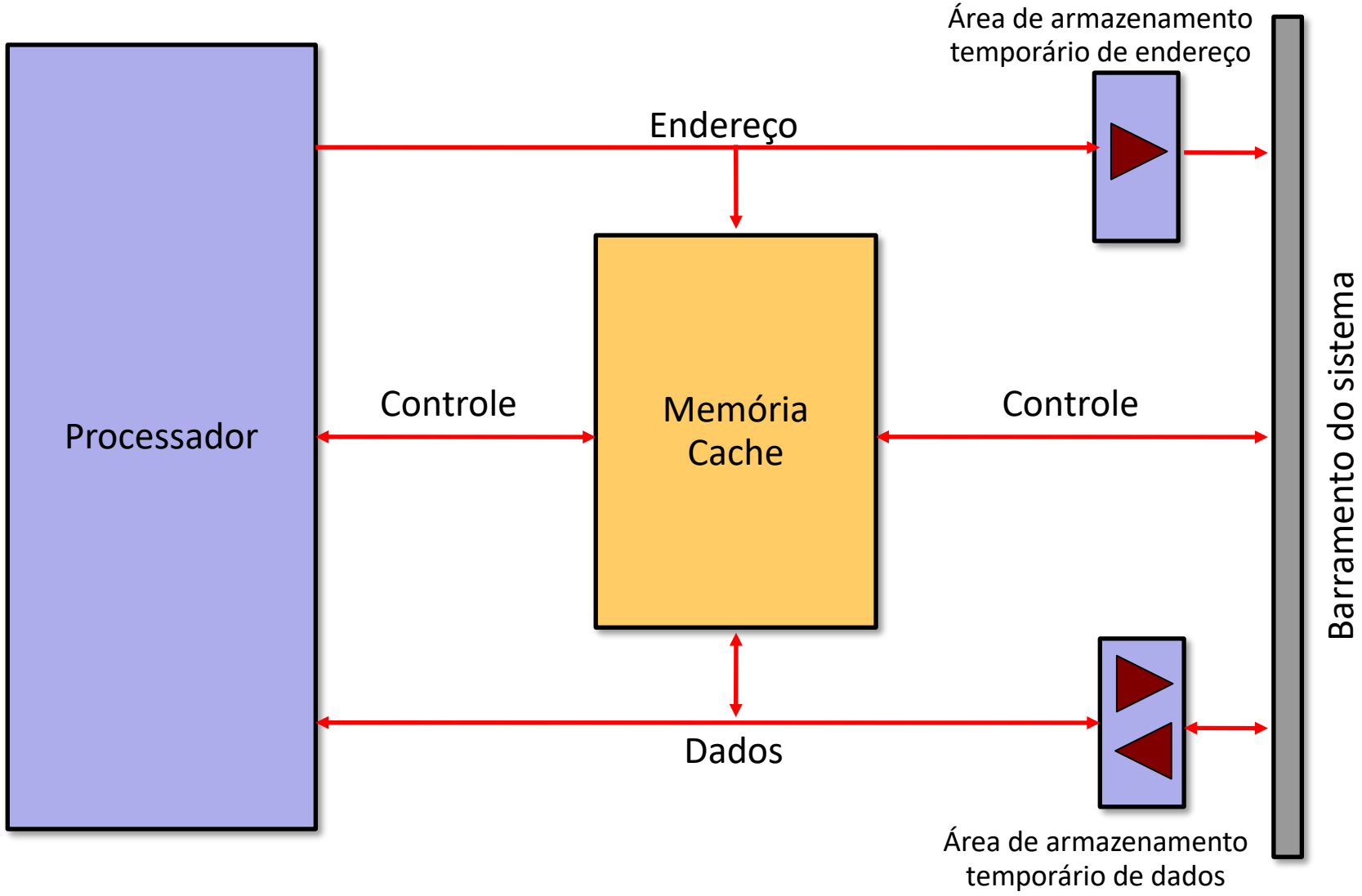
- Se um acesso à instrução resultar em um falha, o conteúdo do registrador de instrução será inválido
  - Para colocar a instrução correta na cache, precisamos ser capazes de instruir o nível inferior na hierarquia de memória a realizar uma leitura
  - O endereço da instrução que gera uma falha de cache de instruções é igual ao valor do contador de programa menos 4

Tratando falhas de cache

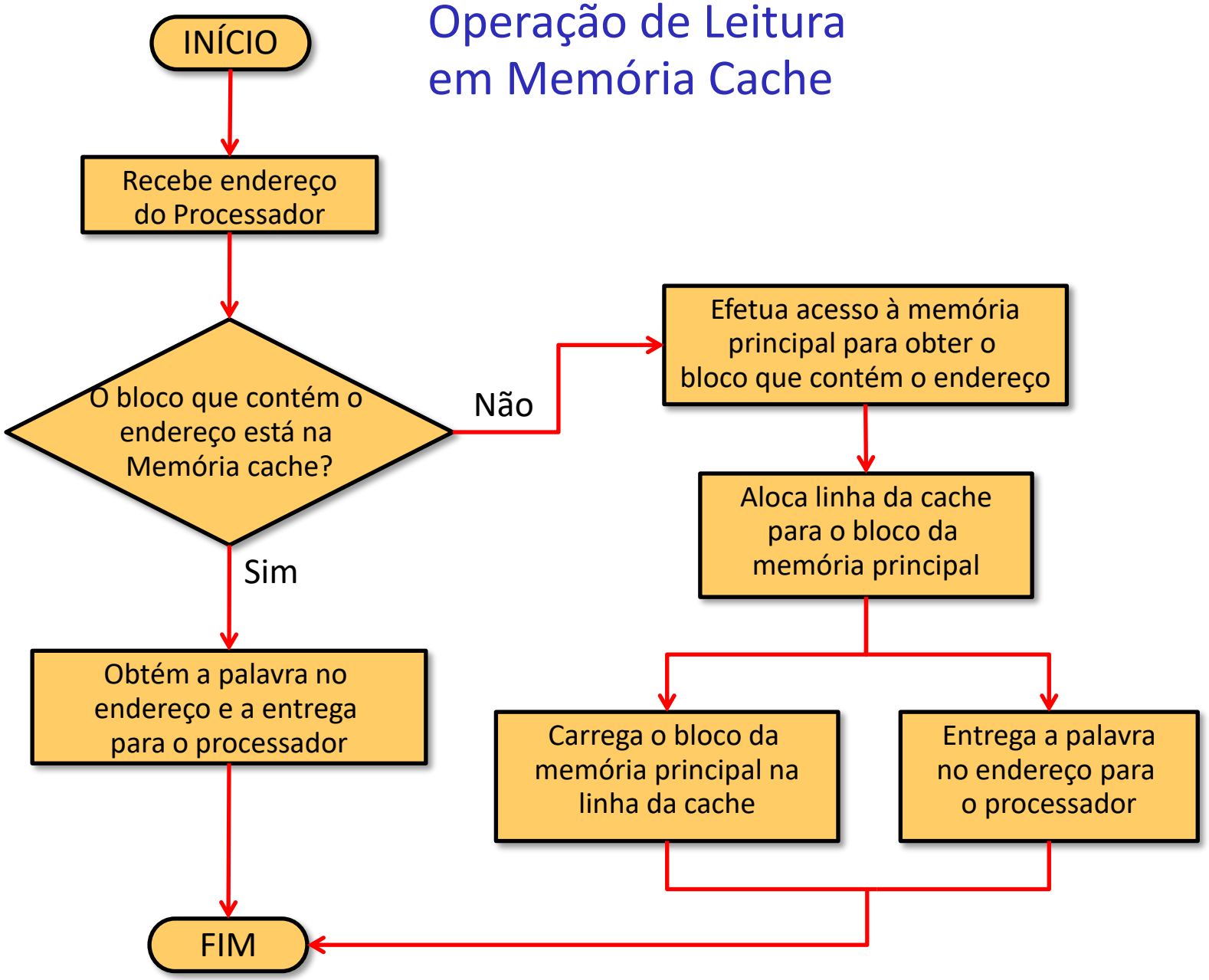


- Etapas quando acontece uma falha de cache
  1. Enviar o valor de PC original (PC atual – 4) para memória
  2. Instruir a memória principal a realizar uma leitura e esperá-la completar seu acesso
  3. Escrever na entrada da cache, colocando os dados na parte dos dados da entrada, escrevendo os bits mais significativos no endereço (vindo da ALU) no campo tag e ligando o bit de validade
  4. Reiniciar a execução da instrução na 1ª etapa, o que buscará novamente a instrução, desta vez encontrando-a na cache

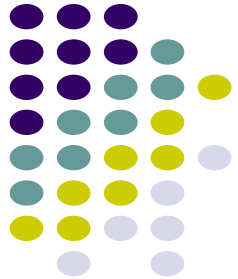
Tratando falhas de cache



Operação de Leitura em Memória Cache

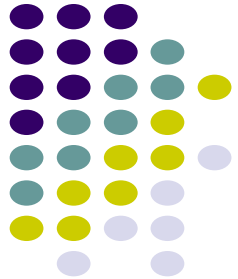


Tratando escritas



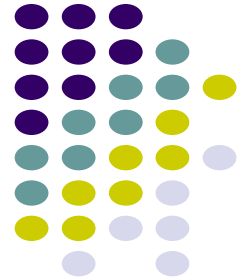
- As escritas funcionam de maneira um pouco diferente
- Suponha que, em uma instrução store, escrevemos os dados apenas na cache de dados (sem alterar a memória principal)
  - Então, após a escrita na cache, a memória teria um valor diferente do valor na cache
  - Nesse caso, dizemos que a cache e a memória estão *inconsistentes*

Tratando escritas



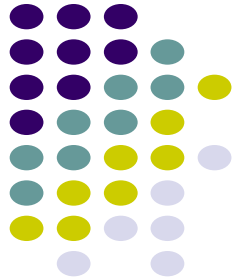
- Políticas de escrita
  - **write-through:** as escritas sempre atualizam a cache e a memória, garantindo que os dados sejam sempre consistentes entre os dois
  - **buffer de escrita:** uma fila que contém os dados enquanto estão esperando para serem escritos na memória
  - **write-back:** manipula escritas atualizando valores apenas no bloco da cache e, depois, escrevendo o bloco modificado no nível inferior da hierarquia quando o bloco é substituído

Projetando o sistema de memória para suportar caches



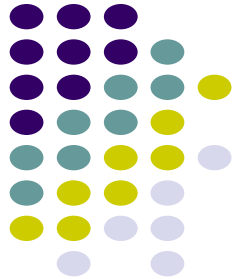
- O processador geralmente é conectado à memória por meio de um barramento
- A velocidade de clock do barramento geralmente é muito mais lenta que a do processador
- A velocidade desse barramento afeta a penalidade de falha

Projetando o sistema de memória para suportar caches



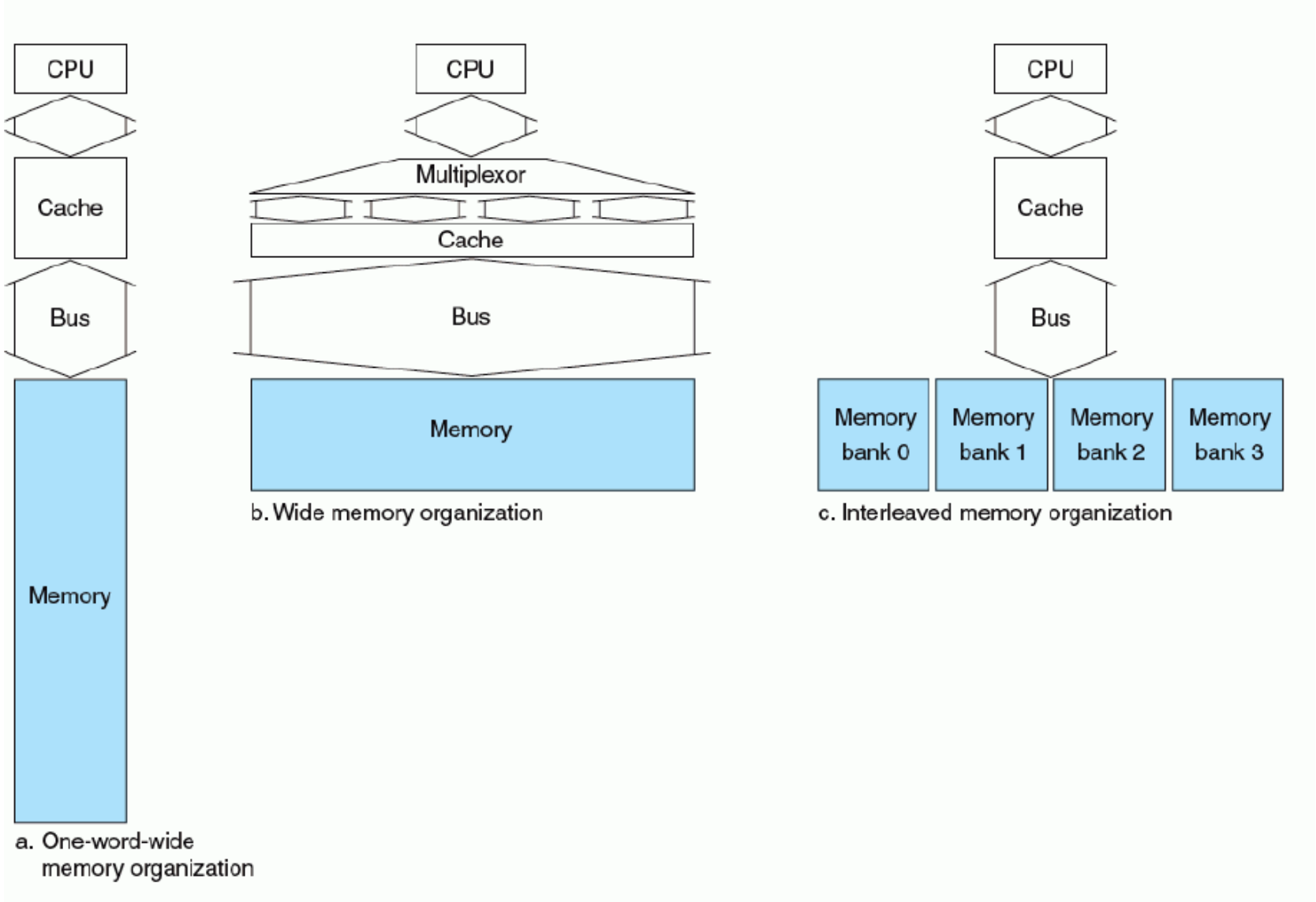
- Para entender o impacto, vamos definir um conjunto hipotético de tempos de acesso à memória
  - 1 ciclo de clock de barramento de memória para enviar o endereço
  - 15 ciclos de clock de barramento de memória para cada acesso a DRAM iniciado
  - 1 ciclo de clock de barramento de memória para enviar uma word de dados

Projetando o sistema de memória para suportar caches

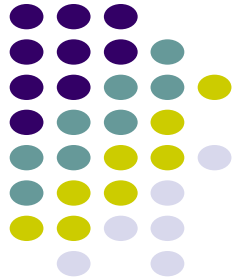


- Se tivermos um bloco de cache de 4 words e um banco de DRAMs com a largura de uma word, a penalidade de falha seria
  - 1 + 4 x 15 + 4 x 1 = 65 ciclos de clock de bar. de mem.
- O número de bytes transferidos por ciclo de clock de barramento para uma única falha
  - (4 x 4) / 65 = 0,25
- A seguir temos 3 opções para projetar o sistema de memória

Projetando o sistema de memória para suportar caches

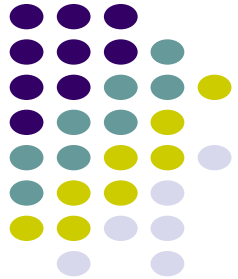


Projetando o sistema de memória para suportar caches



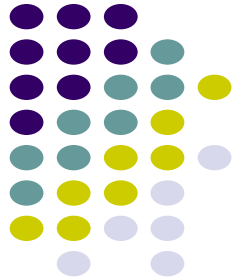
1. A memória possui uma word de largura e todos os acessos são feitos sequencialmente
2. Aumenta a largura de banda para a memória alargando a memória e os barramentos entre o processador e a memória; isso permite acessos paralelos a todas as words do bloco
3. Aumenta a largura de banda alargando a memória mas não o barramento de interconexão

Projetando o sistema de memória para suportar caches



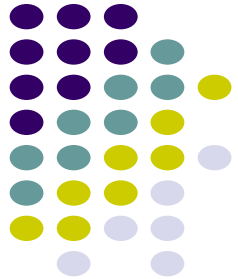
- Aumentar a largura da memória e do barramento aumentará a largura de banda da memória proporcionalmente, diminuindo as partes do tempo de acesso e do tempo de transferência da penalidade de falha
- Com uma largura de memória de 2 words:
  - 1 + 2 x 15 + 2 x 1 = 33 ciclos de clock de barramento de penalidade
  - A largura de banda para uma única falha é 0,48 bytes por ciclo de clock de barramento
- Se a largura de memória for de 4 words
  - A largura de banda é de 0,94 bytes por ciclo

Projetando o sistema de memória para suportar caches



- Em vez de tornar todo o caminho entre a memória e a cache mais largo, os chips de memória podem ser organizados em bancos para ler ou escrever múltiplas words em um único tempo de acesso
- Com 4 bancos, o tempo para obter um bloco de 4 words consistiria em
  - 1 + 1 x 15 + 4 x 1 = 20 ciclos de clock de barramento
  - Largura de banda de 0,80 bytes por ciclo

A programação pode influenciar o desempenho



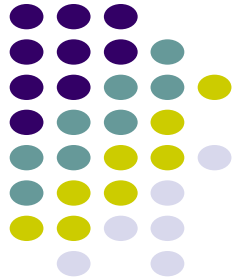
- Quais as semelhanças e diferenças dos programas abaixo?

```
for (row = 0; row < 16; row++)
   for (col = 0; col < 16; col++)
      data[row][col] = value++;
```

```
for (col = 0; col < 16; col++)
   for (row = 0; row < 16; row++)
      data[row][col] = value++;
```

- Comparar a taxa de miss no ferramenta "Data Cache Simulator" do MARS
  - Programas "row-major.asm" e "column-major.asm"

Medindo e melhorando o desempenho da cache



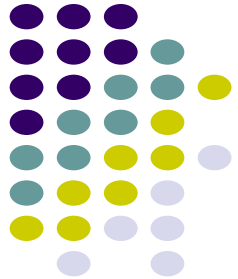
- O tempo de CPU pode ser dividido nos ciclos de clock que a CPU gasta executando o programa e os ciclos de clock que gasta esperando o sistema de memória

  Tempo de CPU = (ciclos de clock de execução da CPU + Ciclos de clock de stall de memória) x Tempo de ciclo de clock

- Os ciclos de clock de stall de memória vêm principalmente das falhas de cache e é isso que iremos considerar

  Ciclos de clock de stall de memória = ciclos de stall de leitura + ciclos de stall de escrita

Medindo e melhorando o desempenho da cache



- Os ciclos de stall de leitura

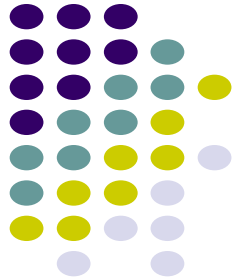
$$Ciclos\ de\ stall\ de\ leitura = \frac{Leituras}{Programa}\ x\ Taxa\ de\ falhas\ de\ leituras\ x\ Penalidade\ de\ falha\ de\ leitura$$

- Escritas são mais complicadas

$$Ciclos\ de\ stall\ de\ escrita = \left(\frac{Escrtias}{Programa}\ x\ Taxa\ de\ falhas\ de\ escrita\ x\ Penalidade\ de\ falha\ de\ escrita\right) + Stalls\ do\ buffer\ de\ escrita$$

  - Os stalls do buffer de escrita serão pequenos e podemos ignorá-los

Medindo e melhorando o desempenho da cache



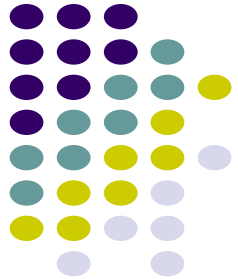
- Os esquemas de write-back também possuem stalls potenciais extras surgindo da necessidade de escrever um bloco de cache novamente na memória quando o bloco é substituído
  - As penalidades de falha de leitura e escritas são iguais

$$Ciclos\ de\ clock\ de\ stall\ de\ memória = \frac{Acessos\ à\ memória}{Programa}\ x\ taxa\ de\ falhas\ x\ penalidade\ de\ falha$$

  - Também podemos fatorar isso como

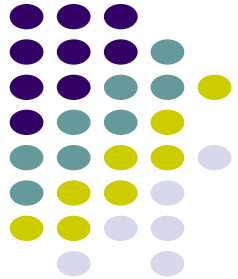
$$Ciclos\ de\ clock\ de\ stall\ de\ memória = \frac{Instruções}{Programa}\ x\ \frac{Falhas}{Instrução}\ x\ Penalidade\ de\ falha$$

Medindo e melhorando o desempenho da cache



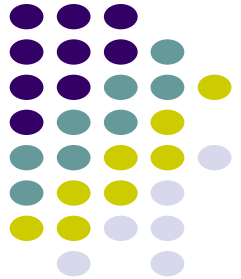
- **Exemplo:** Suponha que uma taxa de falhas de cache de instruções para um programa seja de 2% e que uma taxa de falhas de cache de dados seja de 4%. Se um processador possui um CPI de 2 em qualquer stall de memória e a penalidade de falha é de 100 ciclos para todas as falhas, determine o quanto mais rápido um processador executaria com uma cache perfeita que nunca falhasse. Use as frequências de instruções do SPECint2000.

Medindo e melhorando o desempenho da cache



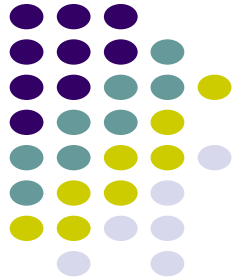
- **Resposta:**
  - O número de ciclos de falha da memória para instruções em termos de contagem de instruções (I) é
    - Ciclos de falha de instruções = I x 2% x 100 = 2,00 I
  - A frequência de todos os loads e store no SPECint2000 é de 36%
    - Ciclos de falha de dados = I x 36% x 4% x 100 = 1,44 I
  - O número total de ciclos de stall da memória é
    - 2,00 I + 1,44 I = 3,44 I
      - Isso é mais do que 3 ciclos de stall da memória por instrução

Medindo e melhorando o desempenho da cache



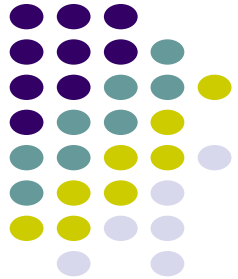
- **Resposta:**
  - O CPI com stalls da memória é
    - 2 + 3,44 = 5,44
  - Como não há mudança alguma na contagem de instruções, ou velocidade de clock, a taxa dos tempos de execução da CPU é
    - $$\frac{Tempo\ de\ CPU\ com\ stalls}{Tempo\ de\ CPU\ com\ cache\ perfeita} = \frac{I\ x\ CPIstall\ x\ Ciclo\ de\ clock}{I\ x\ CPIperfeito\ x\ Ciclo\ de\ clock}$$
    - $$\frac{CPI\ stall}{CPI\ perfeito} = \frac{5{,}44}{2} = 2{,}72$$
    - O desempenho com a cache perfeita é melhor 2,72

Medindo e melhorando o desempenho da cache



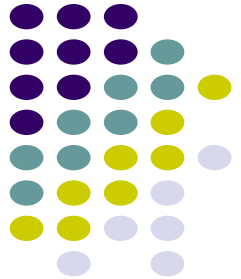
- **Exemplo:** Suponha que aumentamos o desempenho do computador do exemplo anterior dobrando sua velocidade de clock. Como a velocidade da memória principal é improvável de ser alterada, considere que o tempo absoluto para manipular uma falha de cache não mude. O quanto mais rápido será o computador com o clock mais rápido, considerando a mesma taxa de falhas do exemplo anterior?

Medindo e melhorando o desempenho da cache



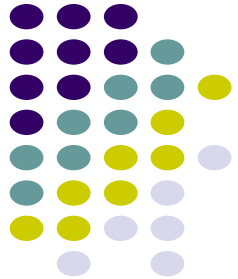
- **Resposta:**
  - Medida com ciclos de clock mais rápidos, a nova penalidade de falha será o dobro dos ciclos de clock, ou 200 ciclos
    - Tempo de ciclos de falha por instrução = (2% x 200) + 36% x (4% x 200) = 6,88
    - Logo, o computador mais rápido com falhas de cache terá um CPI de
      - 2 + 6,88 = 8,88
      - Comparado com um CPI com falhas de cache de 5,44 para o computador mais lento

Medindo e melhorando o desempenho da cache



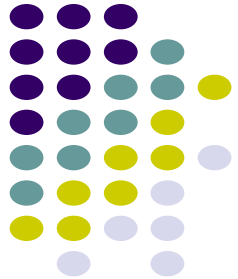
- **Resposta:**
  - Podemos calcular o desempenho relativo:
    - $\frac{Desempenho\ com\ clock\ rápido}{Desempenho\ com\ clock\ lento} = \frac{Tempo\ execução\ com\ clock\ lento}{Tempo\ execução\ com\ clock\ rápido}$
    - $= \frac{CI\ x\ CPI\ clock\ lento\ x\ Ciclo\ de\ clock}{CI\ x\ CPI\ clock\ rápido\ x\ \frac{Ciclo\ de\ clock}{2}}$
    - $= \frac{5{,}44}{8{,}88} = 1{,}23$
  - Portanto, o computador mais rápido é cerca de 1,2 vezes mais rápido, e não 2 vezes mais rápido, que ele seria se ignorássemos as falhas de cache

Medindo e melhorando o desempenho da cache



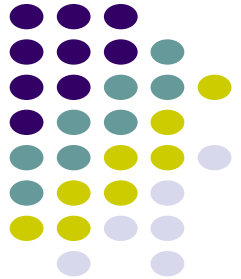
- Se um processador melhorar a velocidade de clock e o CPI, ele experimentará uma consequência dupla:
  1. Quanto menor o CPI, maior será o impacto dos ciclos de stall
  2. É improvável que de memória principal seja melhorado tão rápido quanto o tempo de ciclos do processador, principalmente porque o desempenho da DRAM básica não está se tornando muito mais rápido. Ao calcular o CPI, a penalidade e falha da cache é medida em ciclos de clock do processador para uma falha. Portanto, se as memórias principais de 2 processadores tiverem os mesmos tempos de acesso absolutos, uma velocidade de clock de processador mais alta produzirá uma penalidade de falha maior

Verifique você mesmo



1. A velocidade do sistema de memória afeta a decisão do projetista sobre o tamanho do bloco de cache. Quais dos seguintes princípios de projeto de cache normalmente são válidos?
   a) Quanto mais curta for a latência da memória, menor será o bloco de cache.
   b) Quanto mais curta for a latência da memória, maior será o bloco da cache.
   c) Quanto maior for a latência da memória, menor será o bloco da cache.
   d) Quanto maior for a latência da memória, maior será o bloco de cache.

Respostas



1. Letras a e d: uma penalidade de falha menor pode levar a blocos menores, mas uma largura de banda de memória mais alta normalmente leva a blocos maiores, já que a penalidade de falha é apenas ligeiramente maior

Referências

- PATTERSON, D. A. ; HENNESSY, J.L. Organização e projeto de computadores – a interface hardware software. 3. ed. Editora Campus, 2005.
- STALLINGS, W. Arquitetura e organização de computadores: projeto para o desempenho. 8. ed. Prentice Hall, 2009.