Aula 5: Hierarquia de Memória - Parte I

Professor(a): Virgínia Fernandes Mota
http://www.dcc.ufmg.br/∼virginiaferm

OCS (TEORIA) - SETOR DE INFORMÁTICA

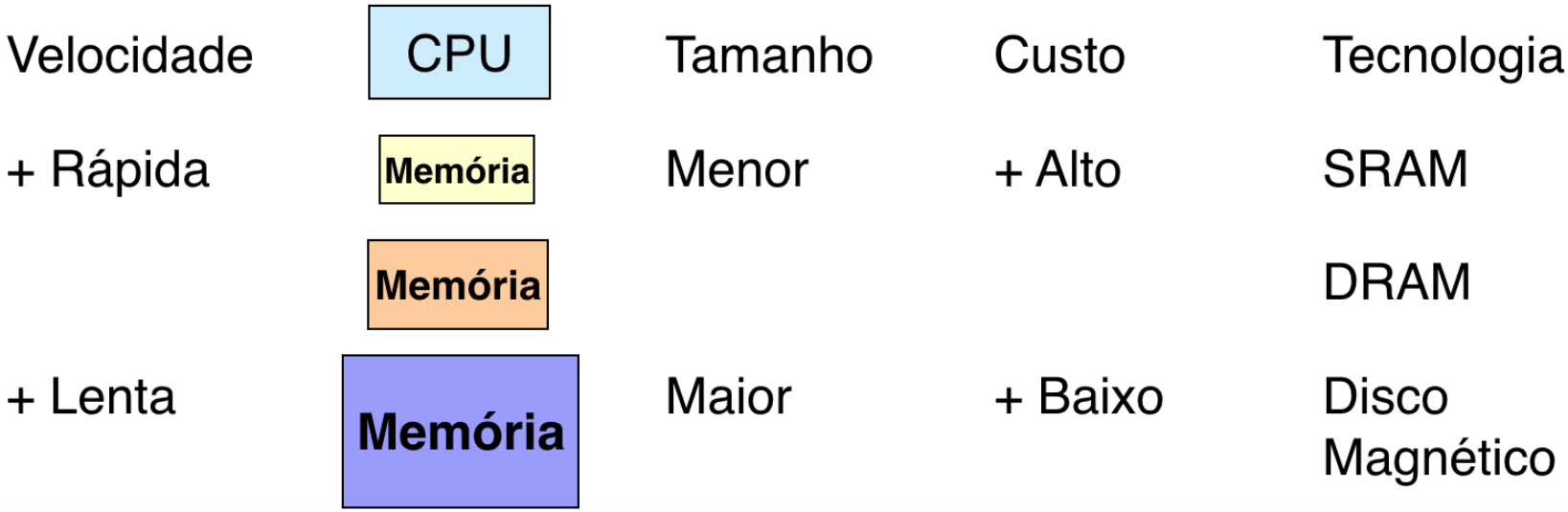
Introdução

- Programadores desejam quantidades ilimitadas de memória rápida
- Analogia: biblioteca
- **Princípio da localidade**: programas acessam parte relativamente pequena do espaço de endereçamento em qualquer instante do tempo
  - Localidade temporal: se um item é referenciado, ele tenderá a referenciado novamente em breve (loops)
  - Localidade espacial: se um item é referenciado, os itens cujos endereços estão próximos tenderão a ser referenciados em breve (instruções acessadas sequencialmente, dados de vetor, registros...)

Introdução

- **Hierarquia de memória**
  - Múltiplos níveis de memória com diferentes velocidades e tamanhos
  - Conforme distância da CPU aumenta, tamanho das memórias e tempo de acesso também aumentam
- Três tecnologias para construção de hierarquias de memória: DRAM, SRAM e disco magnético.
- Objetivo: oferecer o máximo de memória disponível na tecnologia mais barata, enquanto se fornece acesso na velocidade oferecida pela memória mais rápida

Introdução


Velocidade
CPU
Tamanho
Custo
Tecnologia
+ Rápida
Memória
Menor
+ Alto
SRAM
Memória
DRAM
+ Lenta
Memória
Maior
+ Baixo
Disco
Magnético

Introdução

- Dados são copiados apenas entre dois níveis adjacentes
- Bloco ou linha: unidade de informação mínima que pode estar presente ou ausente na hierarquia de dois níveis. Consiste de múltiplas palavras contíguas na memória.
- Acerto: dados encontrados em algum bloco no nível superior
- Falha: dados não encontrados $\rightarrow$ nível inferior acessado
- Taxa de acerto: fração dos acessos à memória encontrados no nível superior
- Taxa de falhas: proporção dos acessos não encontrados no nível superior. 1 - taxa de acertos

Introdução

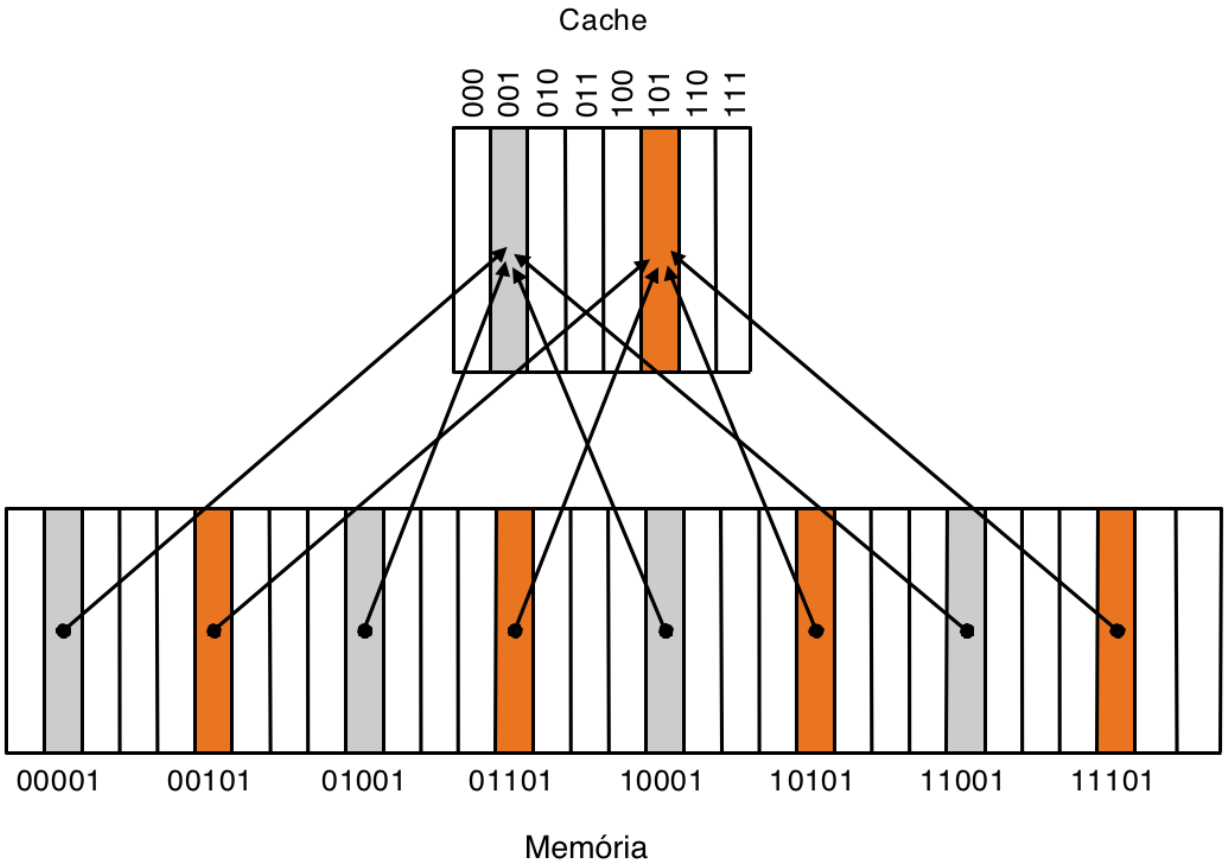
- Tempo de acerto: tempo para acessar o nível superior da hierarquia de memória, incluindo o tempo necessário para determinar se o acesso é um acerto ou uma falha
- Penalidade de falha: tempo para substituir um bloco no nível superior pelo bloco correspondente do nível inferior, mais o tempo para transferir esse bloco para o processador

Princípios básicos de cache

- Memória cache: tipo de memória ultra rápida que armazena os dados e instruções mais utilizadas pelo processador.
- Cache simples: blocos consistem de uma única palavra.
- Como sabemos se um dado está na cache?
- Se estiver na cache, como encontrá-lo?

- **Mapeamento direto**
  - Cada local da memória é mapeado exatamente para um local da cache
    (Endereço de bloco) módulo (Número de blocos de cache)
  - Se número de entradas na cache for potência de 2, basta olhar os log_2 bits menos significativos

Princípios básicos de cache



Princípios básicos de cache

- Cada local de cache pode armazenar o conteúdo de diversos locais de memória diferentes
  - Como saber se dados correspondem a uma palavra requisitada?
  - Tags (rótulos) incluídos
    - Campo que contém informações necessárias para identificar bloco
    - Contém parte superior do endereço
  - Precisamos reconhecer se bloco possui informações válidas
    - Bit de validade: Indica se bloco contém dados válidos

Princípios básicos de cache

Índice	V	Tag	Dados
000	N		
001	N		
010	N		
011	N		
100	N		
101	N		
110	N		
111	N		

Princípios básicos de cache

Endereço Decimal	Endereço Binário	Acerto ou falha	Bloco de cache atribuído
22	10110	Falha	110

Princípios básicos de cache

Índice	V	Tag	Dados
000	N		
001	N		
010	N		
011	N		
100	N		
101	N		
110	S	10	Memória (10110)
111	N		

Princípios básicos de cache

Endereço Decimal	Endereço Binário	Acerto ou falha	Bloco de cache atribuído
22	10110	Falha	110
26	11010	Falha	010

Princípios básicos de cache

Índice	V	Tag	Dados
000	N		
001	N		
010	S	11	Memória (11010)
011	N		
100	N		
101	N		
110	S	10	Memória (10110)
111	N		

Princípios básicos de cache

Endereço Decimal	Endereço Binário	Acerto ou falha	Bloco de cache atribuído
22	10110	Falha	110
26	11010	Falha	010
22	10110	Acerto	110

Princípios básicos de cache

Endereço Decimal	Endereço Binário	Acerto ou falha	Bloco de cache atribuído
22	10110	Falha	110
26	11010	Falha	010
22	10110	Acerto	110
26	11010	Acerto	010

Princípios básicos de cache

Endereço Decimal	Endereço Binário	Acerto ou falha	Bloco de cache atribuído
22	10110	Falha	110
26	11010	Falha	010
22	10110	Acerto	110
26	11010	Acerto	010
16	10000	Falha	000

Princípios básicos de cache

Índice	V	Tag	Dados
000	S	10	Memória (10000)
001	N		
010	S	11	Memória (11010)
011	N		
100	N		
101	N		
110	S	10	Memória (10110)
111	N		

Princípios básicos de cache

Endereço Decimal	Endereço Binário	Acerto ou falha	Bloco de cache atribuído
22	10110	Falha	110
26	11010	Falha	010
22	10110	Acerto	110
26	11010	Acerto	010
16	10000	Falha	000
3	00011	Falha	011

Princípios básicos de cache

Índice	V	Tag	Dados
000	S	10	Memória (10000)
001	N		
010	S	11	Memória (11010)
011	S	00	Memória (00011)
100	N		
101	N		
110	S	10	Memória (10110)
111	N		

Princípios básicos de cache

Endereço Decimal	Endereço Binário	Acerto ou falha	Bloco de cache atribuído
22	10110	Falha	110
26	11010	Falha	010
22	10110	Acerto	110
26	11010	Acerto	010
16	10000	Falha	000
3	00011	Falha	011
16	10000	Acerto	000

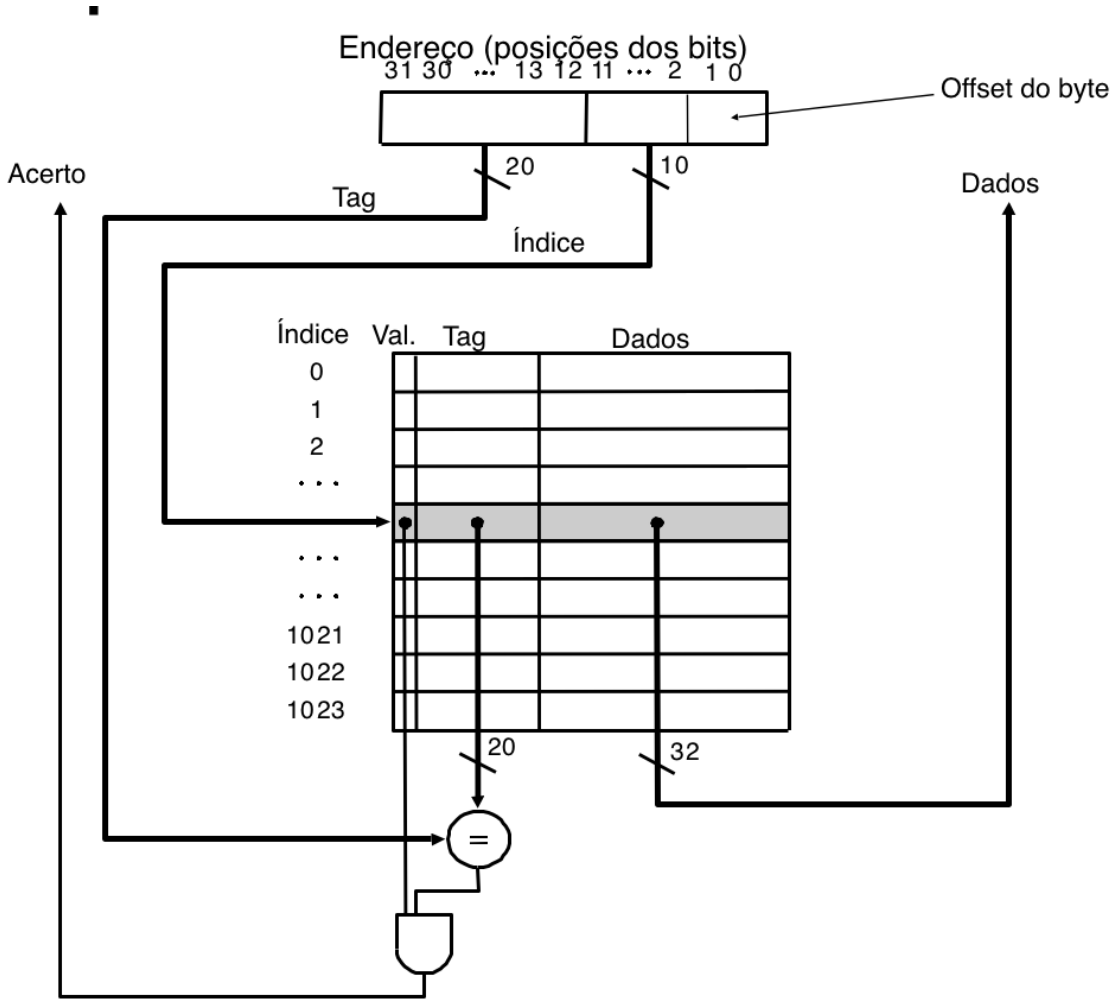
Princípios básicos de cache

Endereço Decimal	Endereço Binário	Acerto ou falha	Bloco de cache atribuído
22	10110	Falha	110
26	11010	Falha	010
22	10110	Acerto	110
26	11010	Acerto	010
16	10000	Falha	000
3	00011	Falha	011
16	10000	Acerto	000
18	10010	Falha	010

Princípios básicos de cache

Índice	V	Tag	Dados
000	S	10	Memória (10000)
001	N		
010	S	10	Memória (10010)
011	S	00	Memória (00011)
100	N		
101	N		
110	S	10	Memória (10110)
111	N		

Princípios básicos de cache



Princípios básicos de cache

- Total de bits necessário para uma cache
- Função do tamanho da cache e do tamanho do endereço
- Tamanho do bloco pode ser de várias palavras
  - Endereço de 32 bits
  - 2^n blocos
  - 2^m palavras por bloco
  - Palavras alinhadas como múltiplos de 4 bytes
  - Tag: 32 - (n+m+2)?
  - Total de bits: $2^n \times$ (tamanho do bloco + tamanho da tag + tamanho do campo de validade) => $2^n \times \{(m \times 32) + [32 - (n+m+2)] + 1\} = 2^n \times [(m \times 32) + (31 - n - m)]$
  - Por convenção deve-se excluir o tamanho da tag e do campo de validade

Princípios básicos de cache

- Quantos bits são necessários para uma cache diretamente mapeada com 16KB de dados e blocos de 4 palavras, considerando um endereço de 32 bits e um alinhamento de bytes?

Princípios básicos de cache

- 16 KB = 4 K palavras = 2^{12} palavras
- Tamanho de bloco de 4 palavras = 2^{10} bloco
- Cada bloco possui 4 x 32 ou 128 bits de dados
- Tag: 32 - 10 - 2 - 2 = 18 bits
- Tamanho da cache = $2^{10} \times (128 + 18 + 1) = 147$ Kbits, ou 18,4 KB para uma cache de 16 KB
- Aproximadamente 1,15 vezes o necessário para o armazenamento dos dados

Princípios básicos de cache

- Blocos maiores exploram a localidade espacial para diminuir as taxas de falhas
- PORÉM: A taxa de falhas pode subir posteriormente se o tamanho de bloco se tornar uma fração significativa do tamanho de cache

Princípios básicos de cache

- O número de blocos que pode ser armazenado na cache se tornará pequeno e haverá competição entre esses blocos
  - Bloco será retirado da cache antes que muitas de suas palavras sejam acessadas
- Custo da falha aumenta
  - Penalidade de falha determinada pelo tempo para buscar bloco e carregá-lo na cache
    - Latência até primeira palavra e tempo de transferência
    - Tempo de transferência aumenta conforme o tamanho de bloco aumenta
  - Aumento na taxa de falhas começa a crescer conforme os blocos se tornam maiores

Tratando falhas de cache

- Memórias do caminho de dados dos tópicos anteriores podem ser substituídas por caches
- Unidade de controle precisa detectar uma falha e processá-la, buscando os dados do nível inferior
- Falha de cache cria stall semelhante ao do pipeline
  - Stall do processador inteiro, congelando o conteúdo dos registradores enquanto esperamos a memória

Tratando falhas de cache

- Falha na cache de instruções (semelhante a falha na cache de dados):
  1. Enviar valor de PC original (PC - 4) para a memória
  2. Instruir a memória a realizar leitura e esperar resultado
  3. Escrever entrada na cache, atualizando tag e bit de validade
  4. Reiniciar execução da instrução na primeira etapa, o que buscará novamente a instrução, desta vez encontrando-a na cache

Tratando escritas

- Se dados escritos apenas na cache de dados, memória teria valor diferente da cache $\rightarrow$ Inconsistente
- *Write-through*: dados escritos na cache e na memória principal
- O que fazer quando ocorre falha de dados na escrita?
  - Buscamos dados na memória, e em seguida sobrescrevemos seu valor, inclusive na memória principal

Tratando escritas

- Escritas tratadas de forma simples, mas não oferece bom desempenho
- Toda escrita acessa memória principal
  100 ciclos para acessar memória
  10% de stores
  CPI sem falhas igual a 1
  CPI = 1 + 100 x 10% = 11

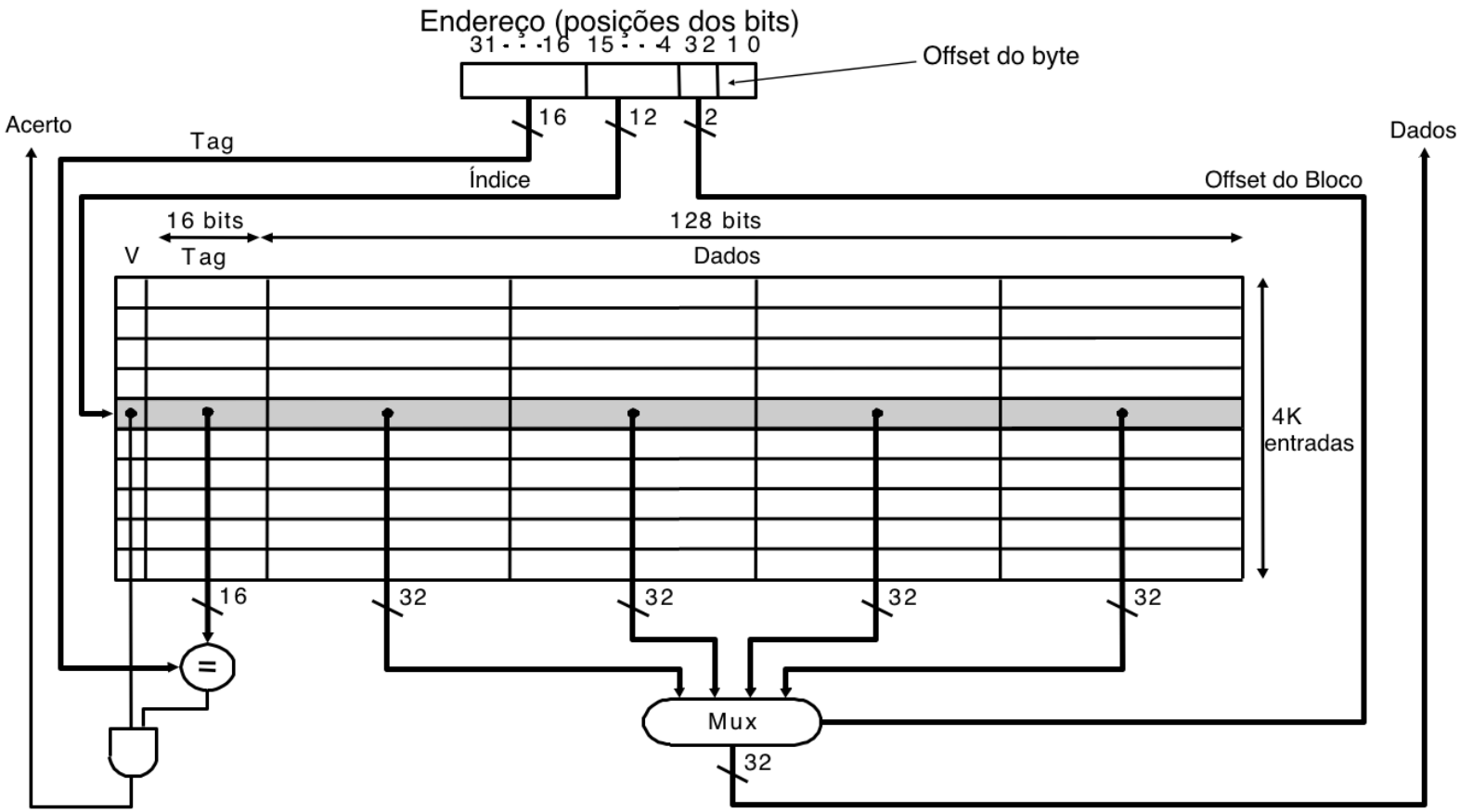
Tratando escritas

- Solução: buffer de escrita
- Armazena dados enquanto estão esperando para serem escritos na memória
- Entrada do buffer liberada quando escrita em memória concluída
- Processador sofre stall se não há espaços para mais escritas

Tratando escritas

- Alternativa: *copy-back* (ou *write-back*)
- Valor escrito apenas na cache
- Nível inferior atualizado quando dado retirado da cache

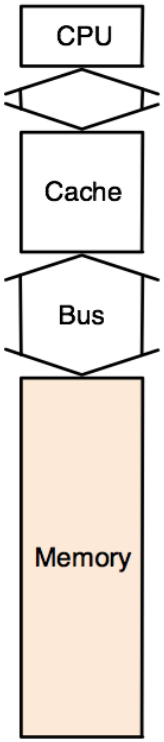
Princípios básicos de cache



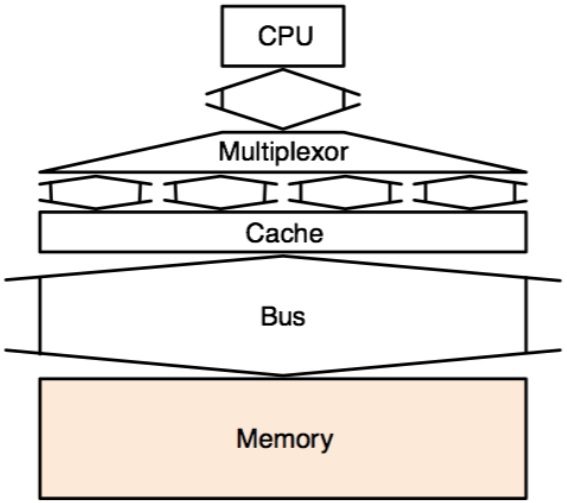
Projetando o sistema de memória para suportar caches

- Memória principal: DRAM
- CPU conectada a memória por meio de barramento: 10 vezes mais lento que processador
- Para entender o impacto das diferentes organizações de memória, considere os seguintes tempos hipotéticos:
  - 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 palavra de dados
  - Bloco de cache de quatro palavras
  - Banco de DRAM com largura de uma palavra (1a Opção)
  - Penalidade: 1 + 4x15 + 4x1 = 65 ciclos de barramento de memória
  - Bytes por ciclo: 4 x 4 / 65 = 0,25

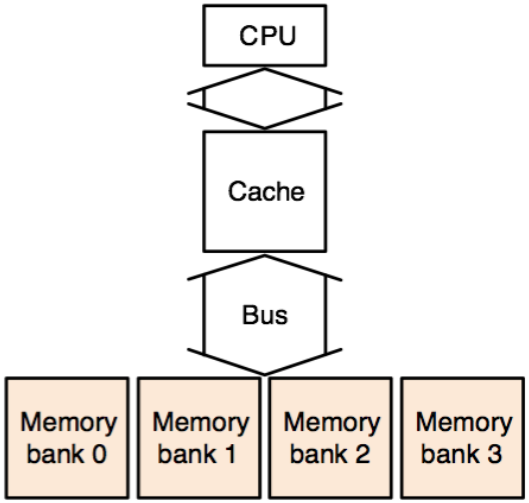
Projetando o sistema de memória para suportar caches



a. One-word-wide memory organization



b. Wide memory organization



c. Interleaved memory organization

Projetando o sistema de memória para suportar caches

- Segunda opção: aumentar a largura da memória e do barramento
- Supondo 2 palavras
  - 1 + 2 x 15 + 2 x 1 = 33 ciclos de clock
  - Largura de banda = 16/33 = 0,48 bytes por ciclo
  - Supondo 4 palavras
  - 17 ciclos
  - Largura de banda = 0,94 bytes por ciclo

Projetando o sistema de memória para suportar caches

- Terceira opção: chips de memória organizados em bancos, para ler ou escrever múltiplas palavras em único tempo de acesso
- Endereço enviado para vários bancos (intercalação)
- Leitura simultânea
  - 1 + 15 + 4 x 1 = 20 ciclos de clock
  - Largura de banda = 16/20 = 0,8 bytes por ciclo

Calculando o desempenho da cache

- Tempo de CPU pode ser dividido:
- Ciclos gastos na execução do programa
- Ciclos gastos esperando o sistema de memória
  - Custos de acesso à cache que são acertos considerados parte dos ciclos de execução normais da CPU
- Tempo de CPU = (ciclos de clock de execução da CPU + ciclos de clock de stall de memória) x Tempo de ciclo de clock
- Stall principalmente oriundos das falhas de cache
- Modelo simplificado

Calculando o desempenho da cache

- Ciclos de clock de stall = ciclos de stall de leitura + ciclos de stall de escrita
- Ciclos de stall de leitura = (leituras por programa) x (taxa de falhas de leitura) x (penalidade de falha de leitura)
- Escritas mais complicadas $\rightarrow$ Depende do esquema utilizado

Calculando o desempenho da cache

- Duas origens de stall em esquema *write-through*
- Falhas de escrita: Bloco deve ser buscado antes da escrita
- Stall do buffer de escrita: Buffer de escrita cheio
- Ciclos de stall de escrita = (escritas por programa x taxa de falhas de escrita x penalidade de falhas de escrita) + stall do buffer de escrita
- Não é simples calcular stall do buffer
- Depende da sincronização das escritas, e não apenas da freqüência
- Com buffer razoável e memória que aceita escritas em uma velocidade que exceda a freqüência de escrita média em programas, podemos ignorá-lo

Calculando o desempenho da cache

- *Copy-back* tem stall potencial quando bloco é substituído e deve ser atualizado na memória
- Penalidades de falhas de leitura e de escrita são iguais
- Considerando stalls do buffer insignificantes:
  - Ciclos de clock de stall de memória = (acessos à memória por programa) x taxa de falhas x penalidade de falha
  - Ciclos de clock de stall de memória = (instruções por programa) x (falhas por instrução) x penalidade de falha

Exemplo

Suponha que 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 CPI de 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 que nunca falhasse. Considere que a freqüência de todos os loads e stores seja de 36%.

Exemplo

- Ciclos de falha de instrução = I x 2% x 100 = 2I
- Ciclos de falha de dados = I x 36% x 4% x 100 = 1,44I
- Total de ciclos de stall de memória = 2I + 1,44I = 3,44I
- Mais de três ciclos de stall da memória por instrução
- CPI = 2 + 3,44 = 5,44
- Como não há mudança na contagem de instruções e na velocidade do clock

$$\frac{\text{Tempo de CPU}_{\text{com stall}}}{\text{Tempo de CPU}_{\text{com cache perfeita}}} = \frac{I \times CPI_{stall} \times \text{Ciclo de clock}}{I \times CPI_{perfeito} \times \text{Ciclo de clock}} \therefore$$

$$\frac{CPI_{stall}}{CPI_{perfeito}} = \frac{5,44}{2} = 2,72$$

Calculando o desempenho da cache

- O que aconteceria se o processador fosse mais rápido, mas a memória continuassem com a mesma velocidade?
- Quantidade de tempo gasto nos stalls de memória se tornará uma fração cada vez maior do tempo de execução

Calculando o desempenho da cache

- Suponha que pipeline melhorou CPI da máquina do exemplo anterior, sem alterar seu clock
- CPI passou de 2 para 1
- Sistema com falha de cache: CPI = 1 + 3,44 = 4,44
- Sistema com cache perfeita seria (4,44 / 1) = 4,44 vezes mais rápido
- Quantidade de tempo gasto em stalls subiria de (3,44 / 5,44) = 63% para (3,44/4,44) = 77%
- Perda de desempenho também aumenta devido às falhas de cache

Exemplo

Suponha que aumentemos 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?

Calculando o desempenho da cache

- Nova penalidade de falha será o dobro: 200 ciclos de clock
- Total de ciclos de falha por instrução = (2% x 200) + 36% x (4% x 200) = 6,88
- Computador mais rápido terá CPI de 2 + 6,88 = 8,88

$$\frac{\text{Tempo de CPU}_{\text{com stall}}}{\text{Tempo de CPU}_{\text{com cache perfeita}}} = \frac{\text{I x CPI}_{\text{stall}} \text{ x Ciclo de clock}}{\text{I x CPI}_{\text{perfeito}} \text{ x Ciclo de clock}} \therefore$$

$$\frac{\text{CPI}_{\text{stall}}}{\text{CPI}_{\text{perfeito}}} = \frac{5,44}{2} = 2,72$$

- Computador cerca de 1,2 vezes mais rápido, e não 2, caso ignorássemos as falhas de cache

Calculando o desempenho da cache

- Penalidades de cache relativas aumentam à medida que um processador se torna mais rápido
- Se processador melhora velocidade de clock e CPI
  - Quanto menor o CPI, maior o impacto dos ciclos de stall
  - Improvável que memória principal melhore tão rápido quanto o tempo de ciclo de processador

Próxima aula

Hierarquia de Memória - Parte II