Infraestrutura de Hardware

Explorando Desempenho com a Hierarquia de Memória

Perguntas que Devem ser Respondidas ao Final do Curso

- Como um programa escrito em uma linguagem de alto nível é entendido e executado pelo HW?
- Qual é a interface entre SW e HW e como o SW instrui o HW a executar o que foi planejado?
- O que determina o desempenho de um programa e como ele pode ser melhorado?
- **Que técnicas um projetista de HW pode utilizar para melhorar o desempenho?**

Tempo de CPU Incluindo Cache

Tempo de CPU = Ciclos de Clock x Período de Clock

- Ciclos de clock devem incluir:
  - Ciclos utilizados pela CPU para executar instruções
  - Ciclos gastos na espera da memória
- Espera da memória → Falta na cache
- Rescrevendo a expressão:

Tempo de CPU = (CPU $_{ciclos}$ + Memoria $_{ciclos}$) x Período

- Ciclos de espera comumente é o componente mais significativo do tempo de CPU
  - Necessidade de técnicas de redução de espera

Tempo de Espera de Memória

$\text{Memória}_{\text{ciclos}} = \text{Leitura}_{\text{ciclos}} + \text{Escrita}_{\text{ciclos}}$

- Ciclos gastos em leitura depende da quantidade de leituras, taxa de faltas e a penalidade

$\text{Leitura}_{\text{ciclos}} = \#\text{Leituras} \times \text{Miss rate} \times \text{Miss penalty}$

- Na escrita, por simplicidade, podemos negligenciar tempo de espera de write buffers (se houver)

$\text{Escrita}_{\text{ciclos}} = \#\text{Escritas} \times \text{Miss rate} \times \text{Miss penalty}$

Exemplo: Calculando Desempenho de CPU com Cache

Problema:

CPU com CPI_{base} = 2, caso não haja nenhuma falta. Calcule a nova CPI caso:

Miss rate $_{instrução}$ = 2%

Miss rate $_{dado}$ = 4%

Miss penalty = 100 ciclos

Frequencia load/store = 36%

Solução:

Seja i = #instrucoes

Miss_instrucao $_{ciclos}$ = i x 0,02 x 100 = 2 x i

Miss_dado $_{ciclos}$ = i x 0,36 x 0,04 x 100 = 1,44 x i

Memoria $_{ciclos}$ = 2 x i + 1,44 x i = 3,44 x i

CPI $_{memoria}$ = CPI_{base} + Memoria $_{ciclos}$ / i

CPI $_{memoria}$ = 2 + 3,44 = 5,44

Exemplo: Impacto da Cache no Desempenho

- E se reduzíssemos CPI sem faltas do exemplo para 1 (através de um pipeline por exemplo)?
- Qual seria o impacto de faltas da cache no desempenho?

$CPI_{antigo} = 2$
$CPI_{memoria} = 2 + 3,44 = 5,44$
% de tempo de execução = 3,44/5,44 = 63%

$CPI_{novo} = 1$
$CPI_{memoria} = 1 + 3,44 = 4,44$
% de tempo de execução = 3,44/4,44 = 77%

Hit Time e Desempenho

- **Hit time também pode influenciar o desempenho**

  Hit time = tempo de acesso ao dado + tempo de verificação se dado está ou não na cache

  Diferentes fatores podem alterar hit time

  - Exs: tamanho de cache, associatividade, etc

- **Tempo de acesso a memória deve levar em conta tempo de misses e hits**

$Tempo\_acesso_{memória} = Hit\ time + Miss\ rate \times Miss\ penalty$

Exemplo: Calculando Tempo Médio de Acesso à Memória

Problema:

CPU com clock de 1ns. Calcule o tempo médio de acesso à memória caso:

Miss rate $_{\text{instrução}}$ = 5%

Miss penalty = 20 ciclos

Hit time = 1 ciclo

Solução:

Tempo_acesso $_{\text{medio}}$ = Hit time + Miss rate x Miss penalty

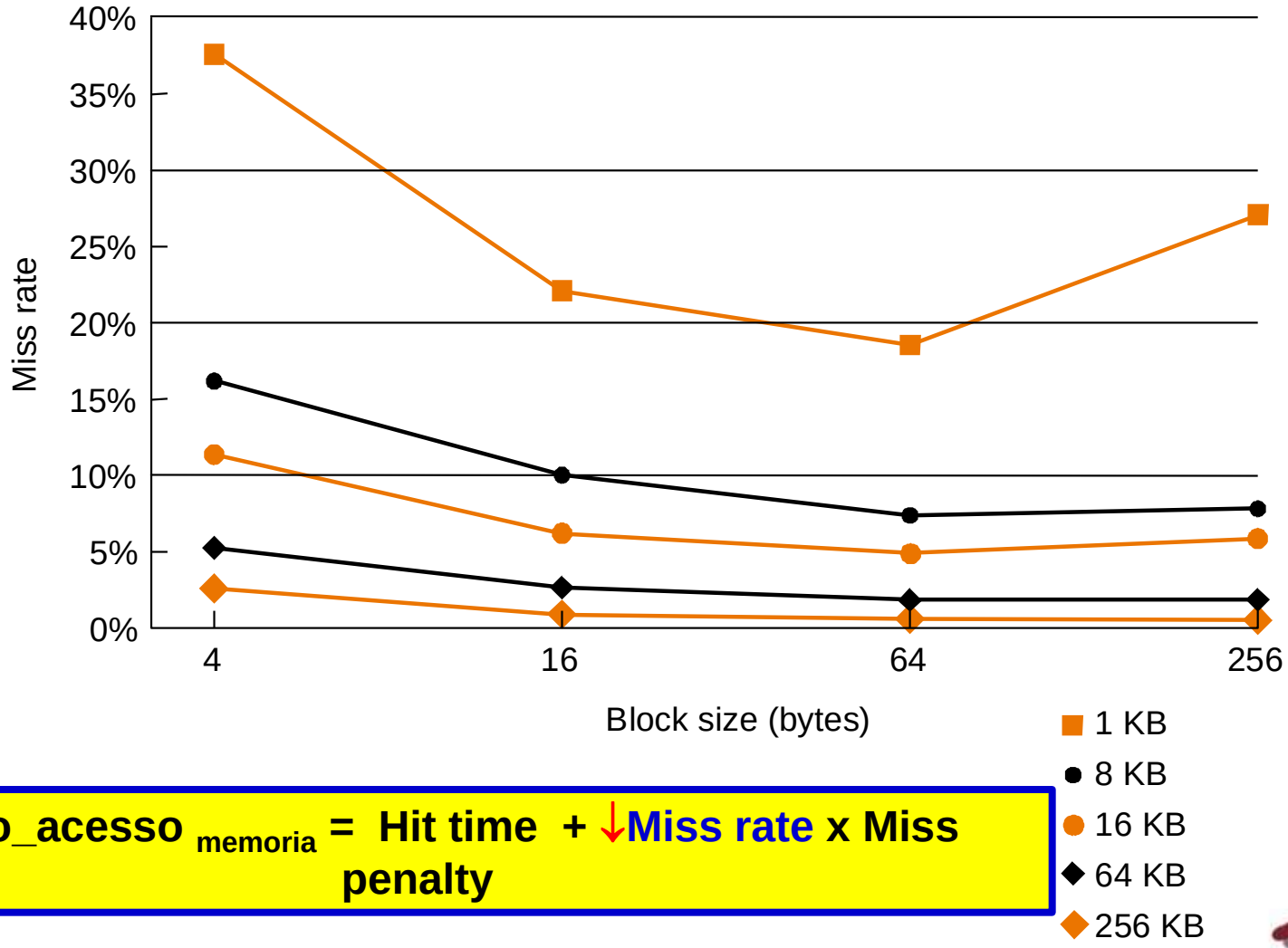
Tempo_acesso $_{\text{medio}}$ = 1 + 0,05 x 20 = 2 ciclos ou 2ns

Melhorando Desempenho da Cache

Tempo_acesso $_{memoria}$ = Hit time + Miss rate x Miss penalty

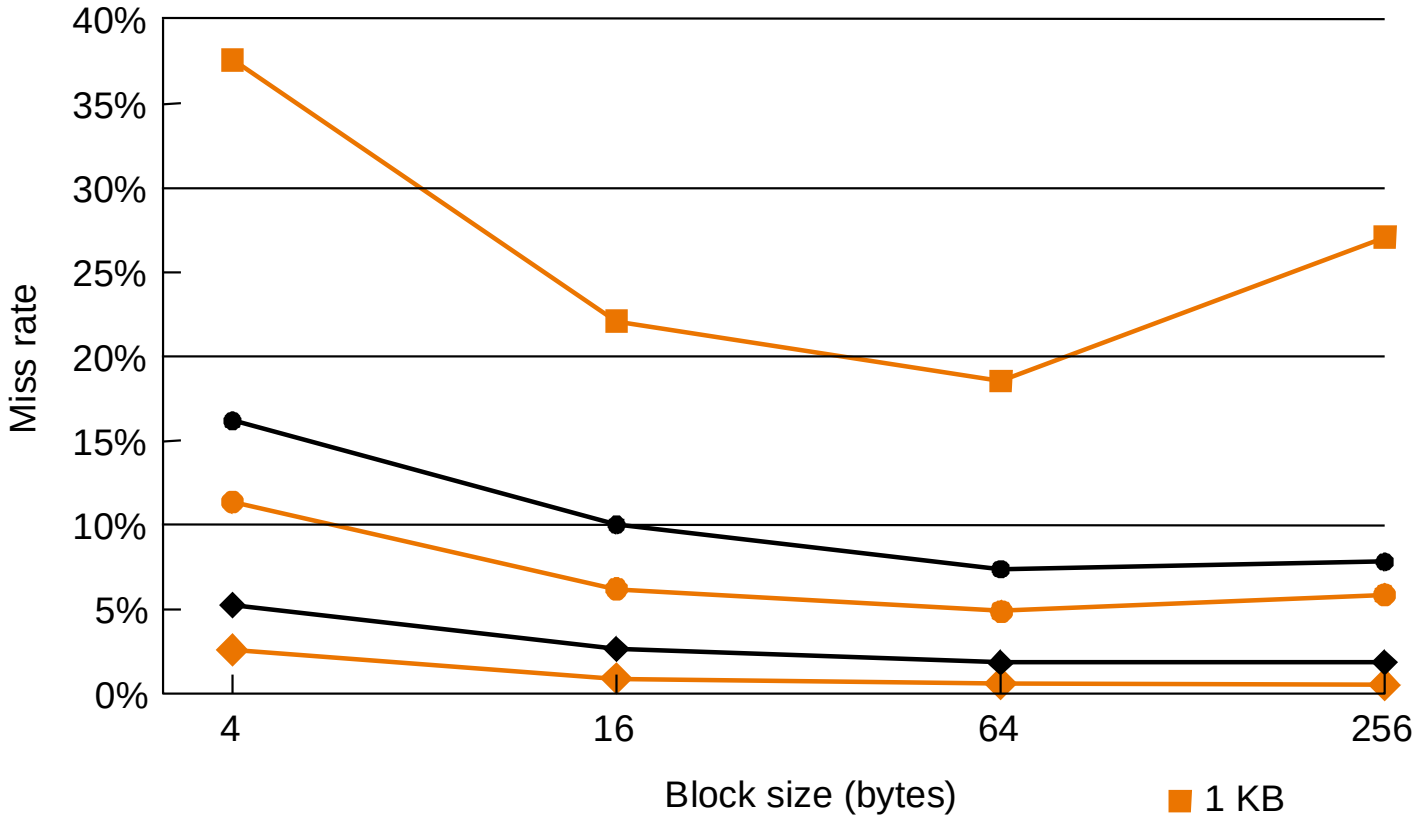
- Melhorar desempenho: Diferentes estratégias para atacar os diferentes fatores que influenciam tempo médio de acesso
  - Redução do Miss rate
  - Redução do Miss penalty
  - Redução do Hit time

Reduzindo Miss Rate: Aumentar Tamanho do Bloco



Tempo_acesso $_{memoria}$ = Hit time + ↓Miss rate x Miss penalty

Tamanho do Bloco x Miss Rate



Aumento indiscriminado do tamanho do bloco pode aumentar miss rate se tamanho de cache permanece constante

Considerações sobre o Aumento do Tamanho do Bloco

- **Blocos maiores podem reduzir miss rate**
  - Por causa da localidade espacial
- **Contudo, em cache de tamanho fixo**
  - Blocos maiores $\Rightarrow$ menos blocos
    - Mais competição $\Rightarrow$ aumenta miss rate
- **Aumenta o miss penalty**
  - Tempo de transmissão de bloco é maior

Tempo_acesso $_{memoria}$ = Hit time + $\downarrow$Miss rate x $\uparrow$ Miss penalty

Reduzindo Miss Rate – Associatividade

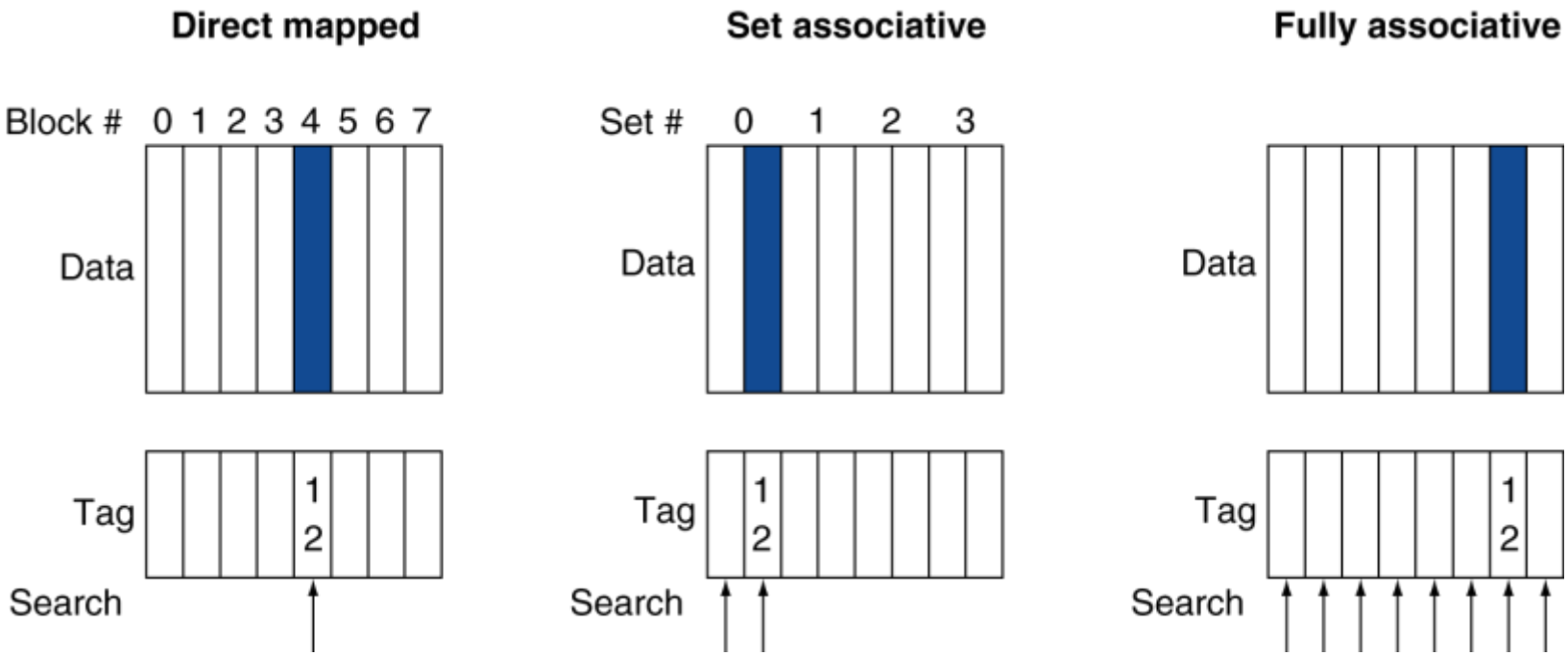
- **Estratégias para posicionamento dos blocos:**

  Mapeamento direto: cada bloco de memória possui posição única na cache

  Associativa por conjunto: cada bloco de memória pode ser colocado em algumas posições na cache

  Completamente Associativa: cada bloco de memória pode ser colocado em qualquer posição da cache

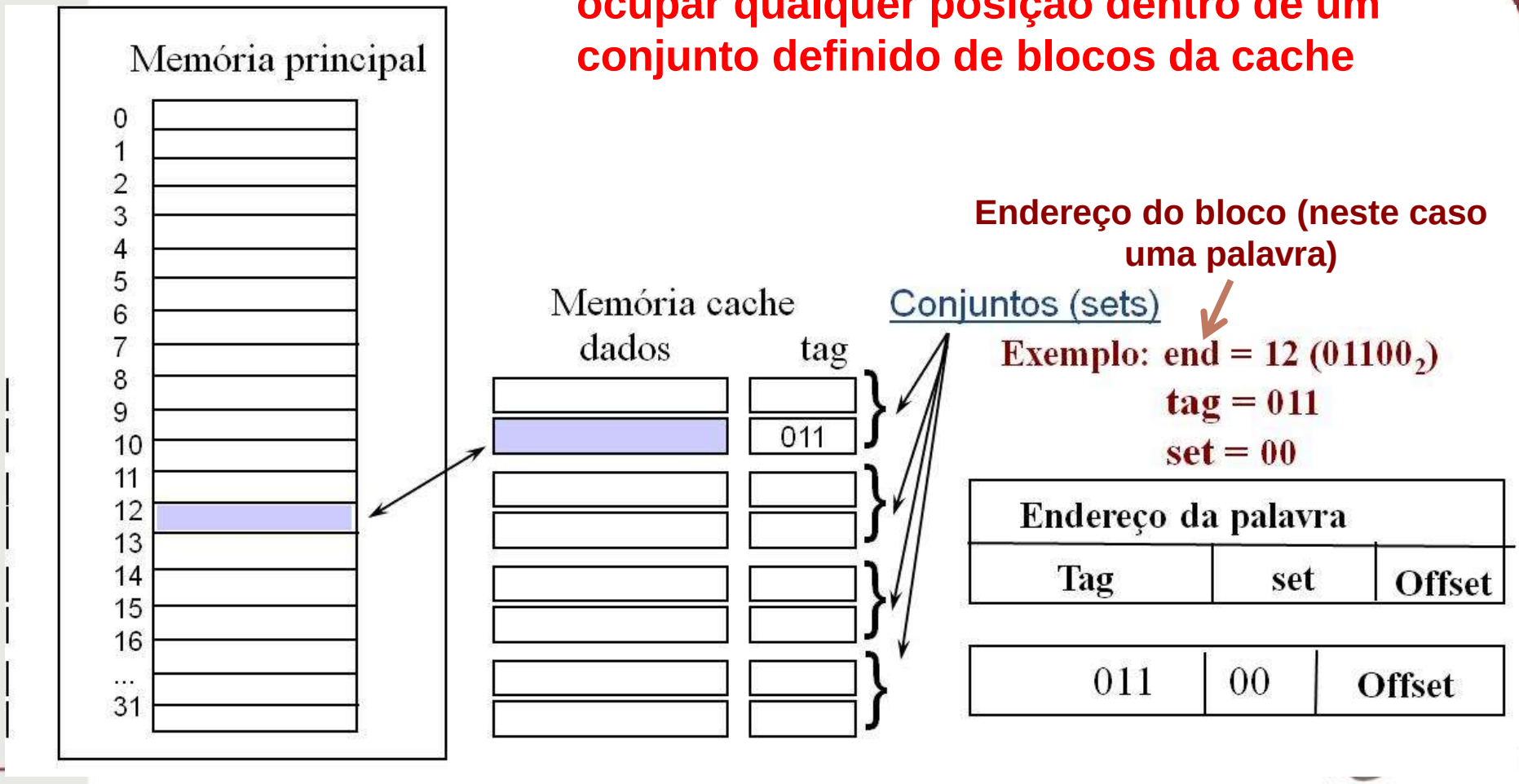
Reduzindo Miss Rate: Aumentar Associatividade



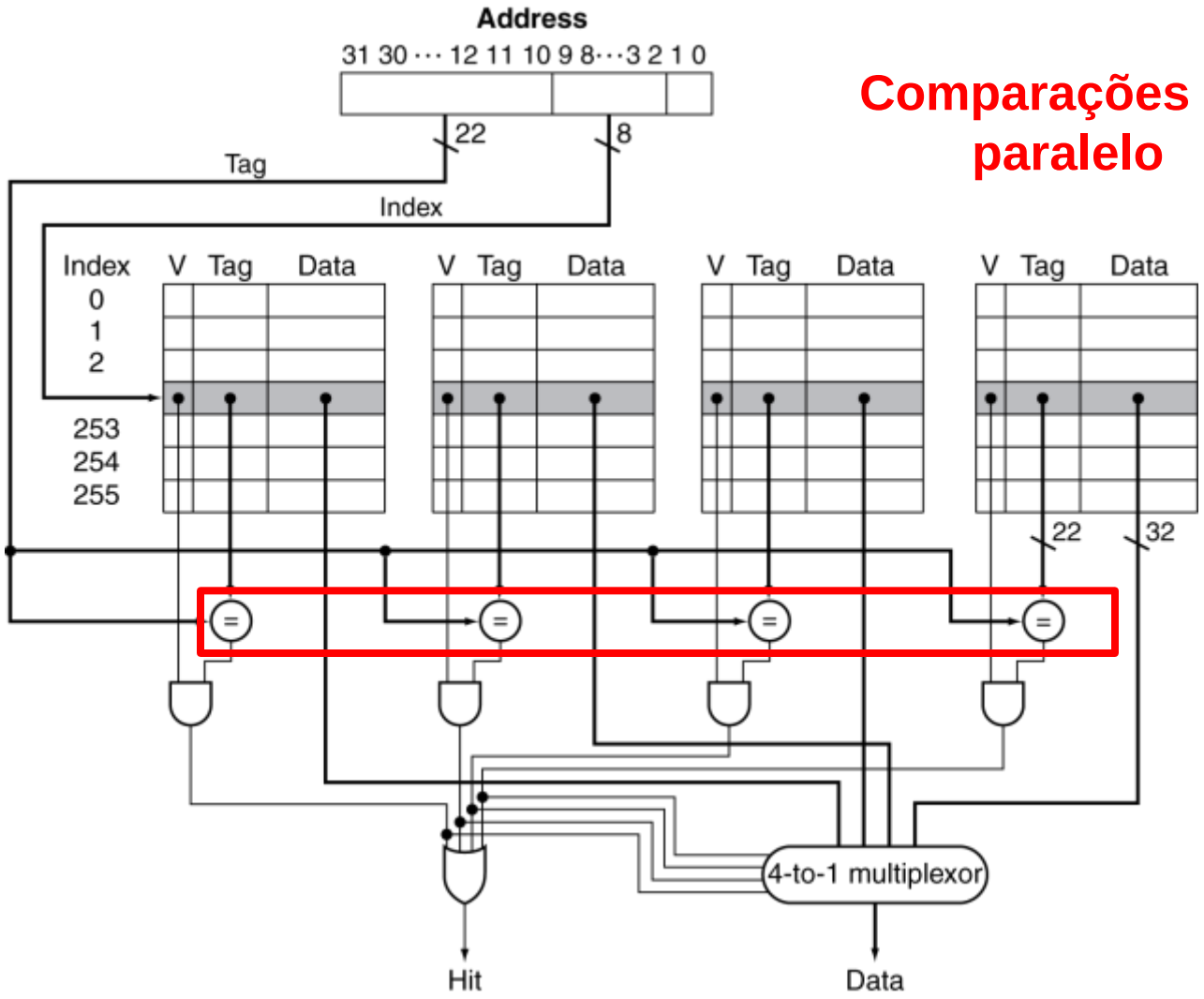
Tempo_acesso $_{memoria}$ = Hit time + ↓Miss rate x Miss penalty

Mapeamento Associativo por Conjunto

Um bloco na memória principal pode ocupar qualquer posição dentro de um conjunto definido de blocos da cache



Organização de uma Cache Associativa

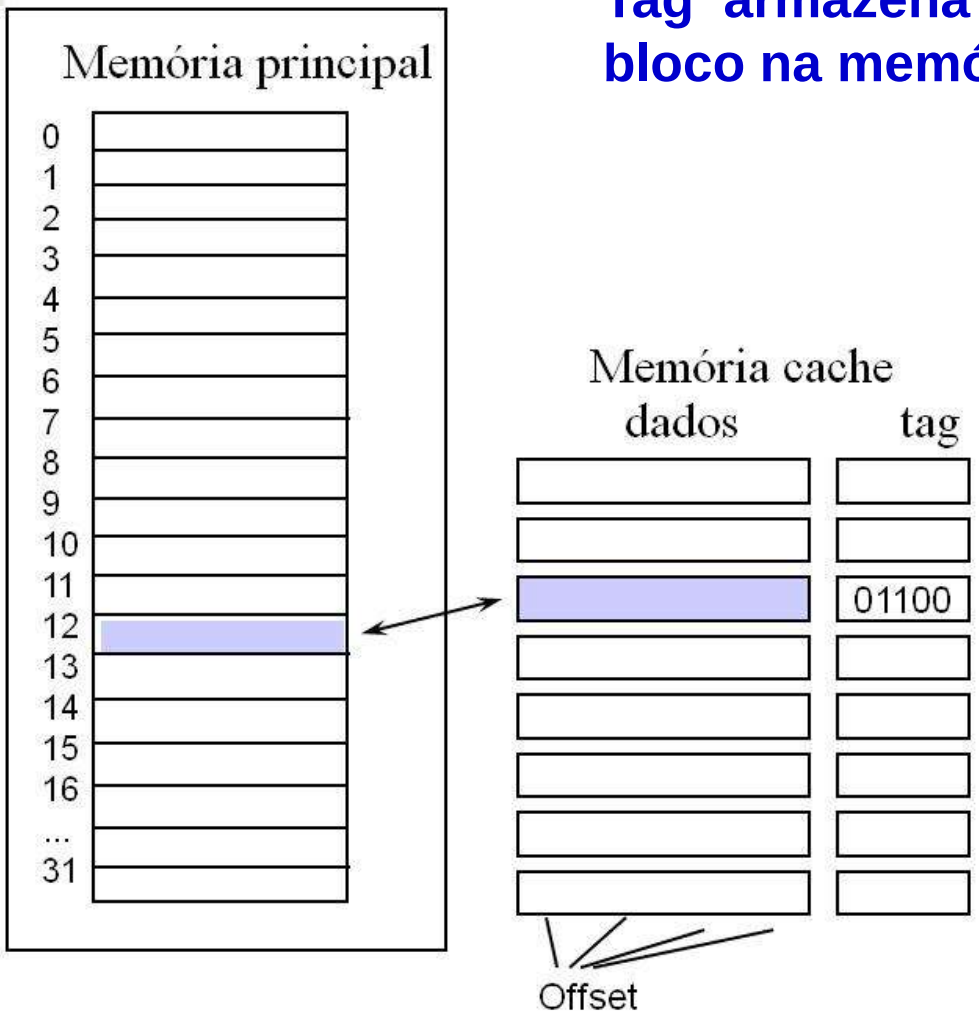


Comparações em paralelo

Mapeamento Completamente Associativo

Um bloco na memória principal pode ocupar qualquer posição na cache

Tag armazena na cache o endereço do bloco na memória



Exemplo: tag = 12 (01100_2)

Endereço da palavra	
Tag	Offset

000000...01100	Offset

Grau de Associatividade

- **Para cache com 8 entradas**

One-way set associative
(direct mapped)

Block	Tag	Data
0		
1		
2		
3		
4		
5		
6		
7		

Two-way set associative

Set	Tag	Data	Tag	Data
0				
1				
2				
3				

Four-way set associative

Set	Tag	Data	Tag	Data	Tag	Data	Tag	Data
0								
1								

Eight-way set associative (fully associative)

Tag	Data	Tag	Data	Tag	Data	Tag	Data	Tag	Data	Tag	Data	Tag	Data	Tag	Data

Exemplo de Associatividade

- Comparativo entre caches de 4 blocos

  Mapeamento Direto, associativa grau 2, completamente associativa

  Sequencia de acesso aos blocos: 0, 8, 0, 6, 8

- **Mapeamento Direto**

Block address	Cache index	Hit/miss	Cache content after access			
			0	1	2	3
0	0	miss	**Mem[0]**			
8	0	miss	**Mem[8]**			
0	0	miss	**Mem[0]**			
6	2	miss	Mem[0]		**Mem[6]**	
8	0	miss	**Mem[8]**		Mem[6]	

Exemplo de Associatividade

- **Associativa de grau 2**

Block address	Cache index	Hit/miss	Cache content after access			
			Set 0		Set 1	
0	0	miss	**Mem[0]**			
8	0	miss	Mem[0]	**Mem[8]**		
0	0	hit	**Mem[0]**	Mem[8]		
6	0	Miss	Mem[0]	**Mem[6]**		
8	0	miss	**Mem[8]**	Mem[6]		

- **Completamente associativa**

Block address		Hit/miss	Cache content after access			
0		miss	**Mem[0]**			
8		miss	Mem[0]	**Mem[8]**		
0		hit	**Mem[0]**	Mem[8]		
6		miss	Mem[0]	Mem[8]	**Mem[6]**	
8		hit	Mem[0]	**Mem[8]**	Mem[6]	

Comparação de Métodos de Mapeamento

- Mapeamento direto
  - Simples e Barata
  - Mais faltas
- Associativa
  - Menos Faltas
    - Nem sempre significativa
  - Cara (comparação do endereço em paralelo)

- Exemplo: Simulação de um sistema com 64KB D-cache, 16-palavras/bloco, SPEC2000 (Miss Rate)
  - 1-way (Mapeamento direto): 10.3%
  - 2-way: 8.6%
  - 4-way: 8.3%
  - 8-way: 8.1%

Considerações sobre Aumento da Associatividade

- **Blocos de memória podem ser mapeados para posições diferentes na cache**

  Diminui quantidade de conflitos entre blocos de memória mapeados para um mesmo bloco da cache → redução de miss rate

- **Delay gerado por comparações**

  Aumento do hit time

  Vários comparadores em paralelo

  - HW fica mais caro

Tempo_acesso $_{memoria}$ = ↑ Hit time + ↓Miss rate x Miss penalty

Reduzindo Miss Rate: Políticas de Substituição

- Mapeamento direto: sem escolha
- Associativo por conjunto

  Entrada inválida, se existir uma
- Least-recently used (LRU)

  Escolhe a menos usada recentemente
  - Simples para associatividade de grau 2, gerenciável para grau 4, difícil para grau além destes
- Randômico

  Aproximadamente mesmo desempenho de LRU para alta associatividade

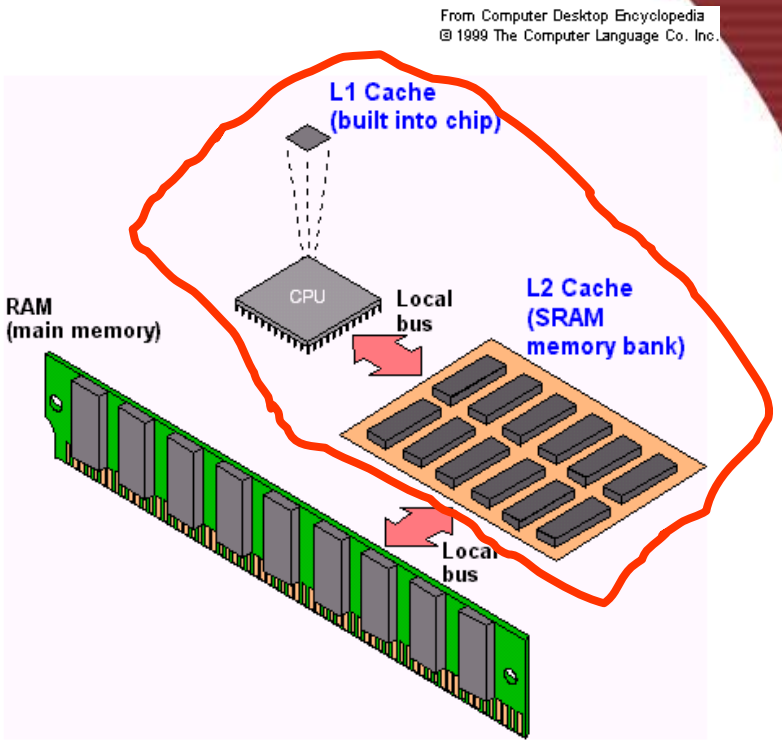
Reduzindo Miss Penalty: Diminuição de Espera da CPU

- **Blocos de memória buscados durante um miss causa a interrupção da execução da CPU**
  - CPU recomeça quando bloco é colocado na cache
- **Early restart**
  - Assim que o bloco que contiver a palavra procurada for carregada na cache esta é enviada para a CPU
- **Critical Word First**
  - Requisita palavra procurada primeiro e a envia para a CPU assim que a mesma foi carregada.
- **Técnicas aplicáveis para grandes blocos**

$Tempo\_acesso_{memoria}$ = Hit time + Miss rate x ↓ Miss penalty

Reduzindo Miss Penalty: Multi-níveis de Cache

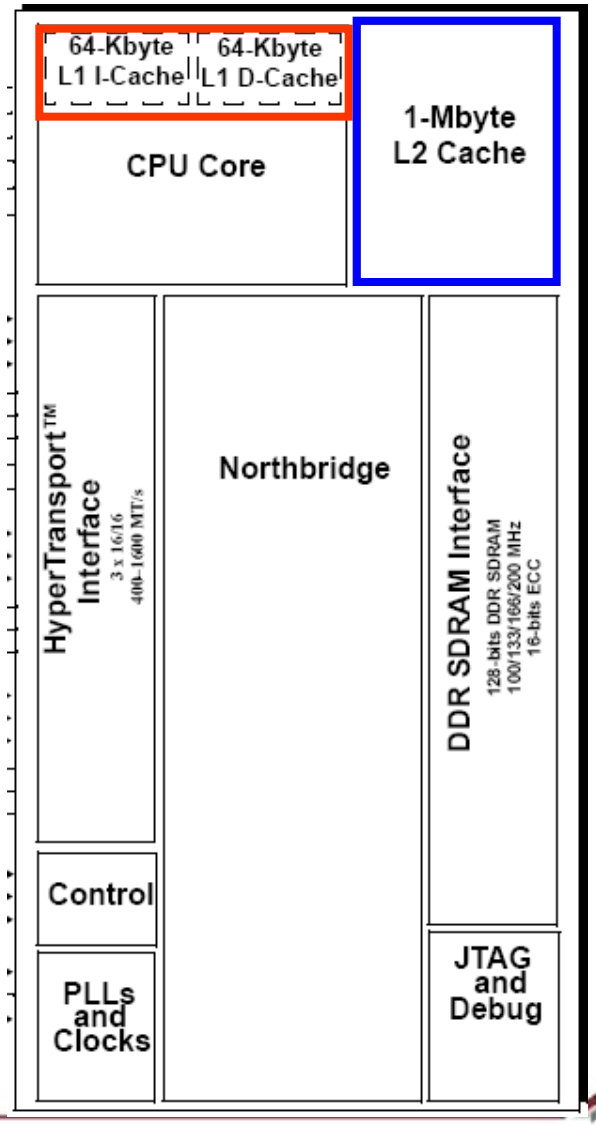
- Muitos processadores vem com pelo menos dois níveis de cache
  - 1º nível:
    - Menor tempo de acesso (hit time)
    - Menor capacidade → blocos menores
    - Objetivo: Reduzir miss penalty e hit time
  - 2º nível:
    - Maior tempo de acesso
    - Maior capacidade → geralmente blocos maiores
    - Objetivo: Reduzir miss rate



Integrado no microprocessador

Exemplo de Cache de 2 Níveis: AMD Athlon 64

Cache de 1° nível



Cache de 2° nível

Centro de Informática

Desempenho de Cache de 2 Níveis

- **Primeiro nível de cache:**

  Foco em minimizar hit time

  $$\text{Tempo\_acesso}_{L1} = \downarrow\text{Hit time}_{L1} + \text{Miss rate}_{L1} \times \text{Miss penalty}_{L1}$$

- **Segundo nível de cache**

  Foco em minimizar miss rate

  $$\text{Tempo\_acesso}_{L2} = \text{Hit time}_{L2} + \downarrow \text{Miss rate}_{L2} \times \text{Miss penalty}_{L2}$$

Exemplo: Melhorando Desempenho com 2 Níveis de Cache

Problema:

CPU com 1 nível de cache, CPI_{base} = 1 e Frequência do clock = 4GHz, Miss rate $_{L1}$ = 2%,
Tempo de acesso memória = 100ns

Calcule o ganho de desempenho se adicionássemos:
2º nível de cache com hit ou miss time = 5ns e que seja grande o suficiente para reduzir o miss rate global em relação à memória para 0,5%.

Exemplo: Melhorando Desempenho com 2 Níveis de Cache

Problema:

CPI_{base} = 1, Frequência do clock = 4GHz, Miss rate $_{L1}$ = 2%, Tempo de acesso memória = 100ns

Calculando CPI com 1 nível de cache

Miss Penalty $_{cycles}$ = Tempo de Acesso a Memoria/Período
Miss Penalty $_{cycles}$ = 100ns/0,25ns/ciclo = 400 ciclos
Memory stall $_{cycles}$ = Miss Rate x Miss Penalty $_{cycles}$
= 0,02 x 400ciclos = 8 ciclos
CPI_{L1} = CPI_{base} + Memory stall $_{cycles}$ = 1 + 8 = 9

Exemplo: Melhorando Desempenho com 2 Níveis de Cache

Problema:

CPI_{base} = 1, Frequência do clock = 4GHz, Miss rate $_{L1}$ = 2%, Tempo de acesso memória = 100ns, Hit $time_{L2}$ = 5ns e Miss $rate_{Global}$ = 0,5%

Calculando CPI com 2 níveis de cache

Hit time $_{cycles}$ = Hit $time_{L2}$ / Período = 5ns/0,25ns/ciclo = 20 ciclos
Primary stalls $_{cycles}$ = 2% x Hit $time_{cycles}$ = 0,02 x 20 = 0,4 ciclos
Secondary stalls $_{cycles}$ = 0,005 x 100/0,25/ciclo = 2 ciclos
CPI_{L1L2} = CPI_{base} + Primary stalls $_{cycles}$ + Secondary stalls $_{cycles}$ = 1 + 0,4 + 2 = 3,4
$Ganho_{Desempenho}$ = CPI_{L1}/CPI_{L1L2} = 9,0/3,4 = 2,6

Considerações sobre Multi-níveis de Cache

- Cache L1
  - Foca em minimizar hit time → cache menor e com menos associatividade
- Cache L-2
  - Foca em minimizar miss rate → cache maior e blocos grandes, mais associatividade
- Geralmente blocos de cache L1 são menores que blocos de L2
- E quanto a duplicação de dados nos dois níveis?
  - Os dados devem ser duplicados (consistência)

Reduzindo Hit Time: Caches Separadas

- **Cache de dados e cache de instruções**
  - Vantagens:
    - Maior largura de banda
    - Evita hazard estrutural
  - Desvantagens:
    - maior taxa de falta

Programa	Miss rate (instr.)	Miss rate (dado)	Miss rate (sep.)	Miss rate (única)
Gcc	6.1%	2.1%	5.4%	4.8%
Spice	1.2%	1.3%	1.2%	

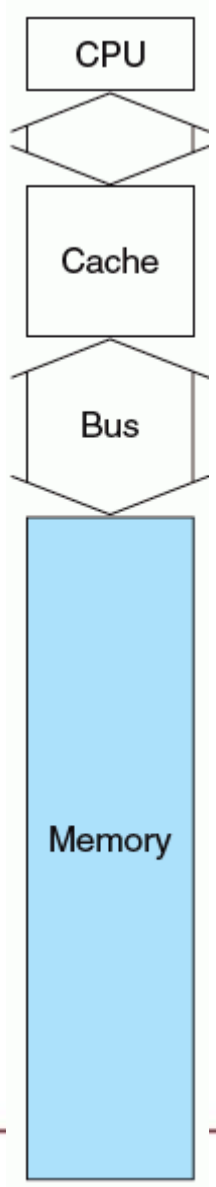
Outras Formas de Reduzir Hit Time

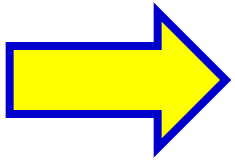
- **Diminuir tamanho da cache**
  - Blocos menores → menor tempo de envio
  - Pode ficar no mesmo chip da CPU
- **Diminuir associatividade**
  - Comparações em caches com associatividade alta provoca delays

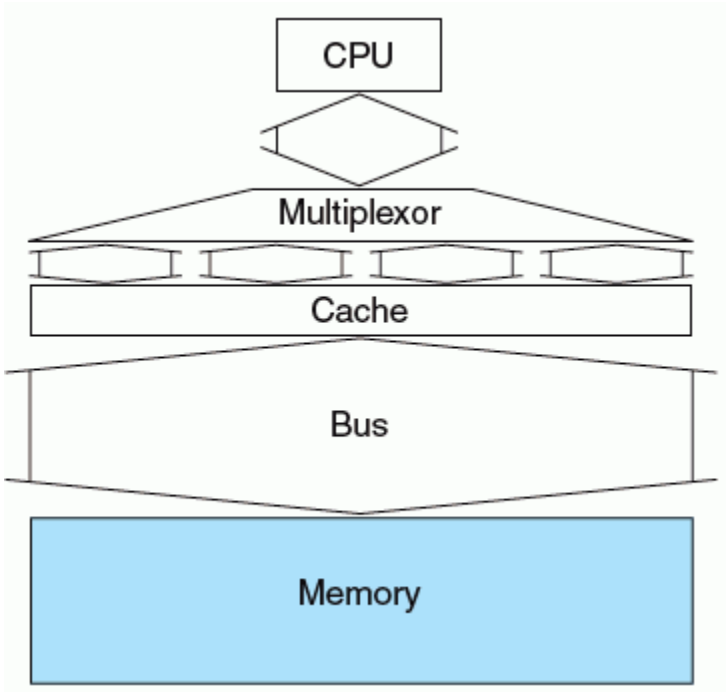
Projetando Memória para Dar Suporte a Caches

- Tempo de acesso(latência) a primeira palavra de bloco na memória é significativo
  - Decodificação do endereço e localização da palavra é demorada
  - Afeta miss penalty de uma cache
- Memória utiliza barramento para transmissão de dados
  - Velocidade do barramento < Velocidade da CPU
  - Afeta miss penalty de uma cache
- Solução: Aumentar largura de banda entre cache e memória

Aumentando Largura de Banda: Memória Mais Larga



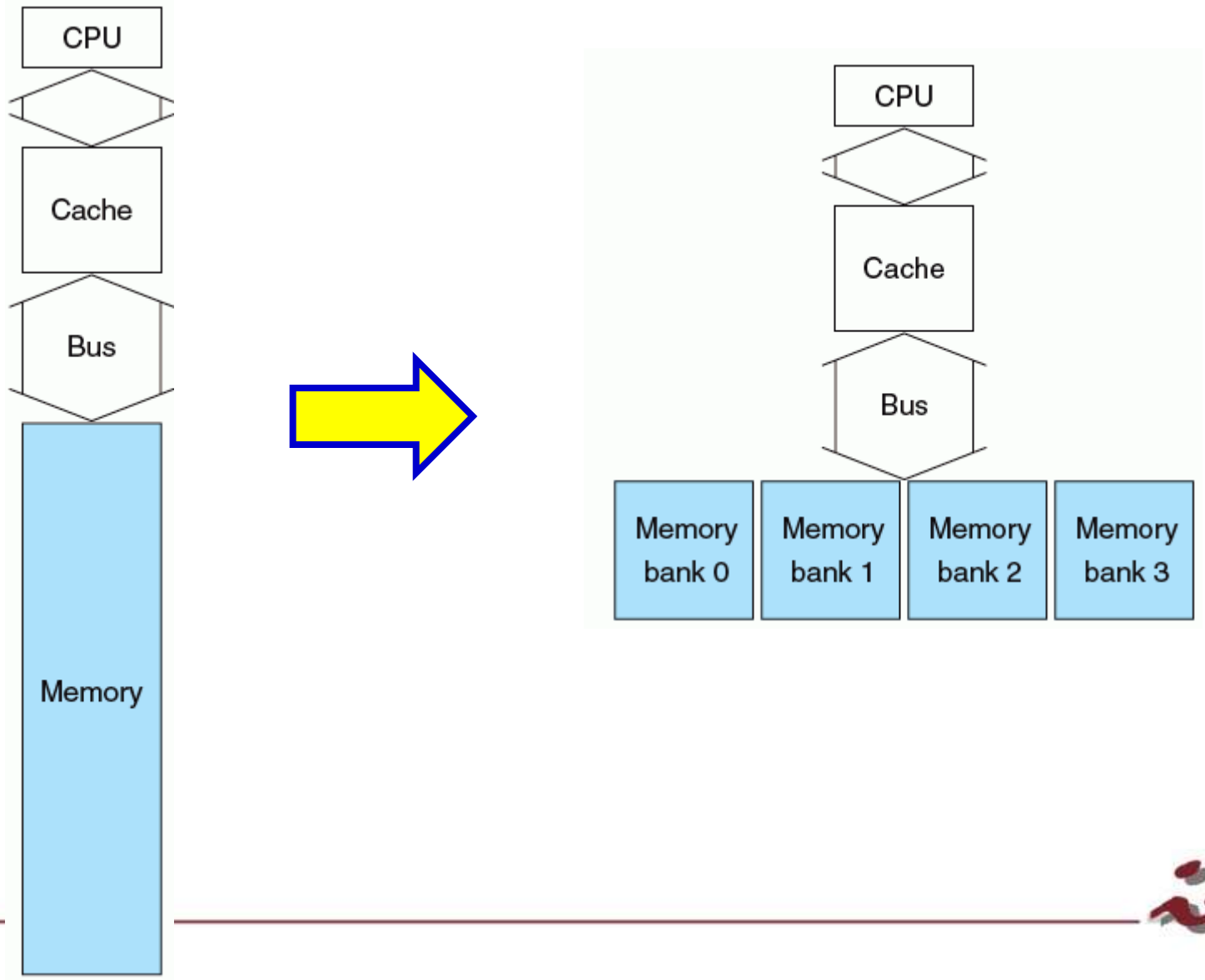




Memórias Mais Largas

- Acesso paralelo a mais de uma palavra por bloco
  - Redução da penalidade de cache
- Necessidade de barramento mais largo
  - Expansão da largura da memória condicionada a barramento
- Necessidade de multiplexadores e lógica de controle

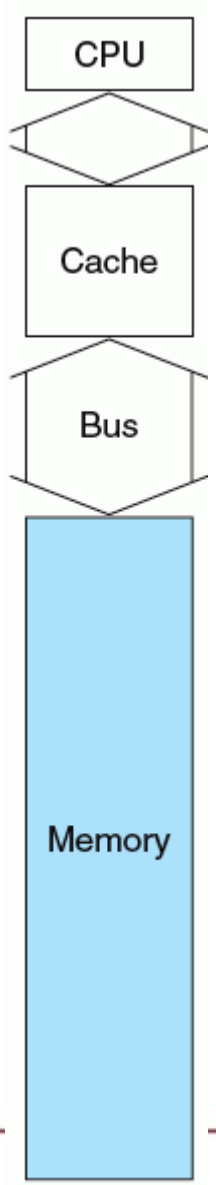
Aumentando Largura de Banda: Memórias "Interleaved"



Memória Interleaved

- Bancos de memória para escrita/leitura de múltiplas palavras
  - Reduz penalidade
- Endereço é enviado simultaneamente para os diferentes bancos de memória
- Largura do barramento não precisa ser aumentado
- Necessita pouco hardware adicional

Exemplo: Tempo de Acesso de Memória Simples



Problema:

Dada uma cache com blocos de 4 palavras e considerando que a largura do banco de memória é de 1 palavra. Calcule tempo de acesso de memória dados o número de ciclos do barramento de cada operação:

- Envio de endereço = 1 ciclo

-Acesso inicial a uma palavra = 15 ciclos

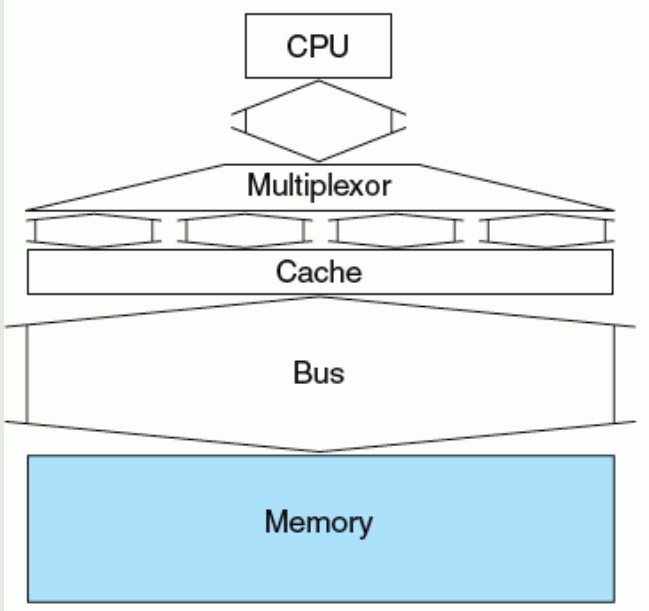
-Envio de uma palavra = 1 ciclo

Solução:

Tempo total = Envio de endereço + acesso inicial + envio de bloco

Tempo total = 1 + 4 x 15 + 4 x 1 = 65 ciclos

Exemplo: Tempo de Acesso de Memória Mais Larga



Problema:

Dada uma cache com blocos de 4 palavras e considerando que a largura do banco de memória é de 2 palavras. Calcule tempo de acesso de memória dados o número de ciclos do barramento de cada operação:

- Envio de endereço = 1 ciclo

-Acesso inicial a uma palavra = 15 ciclos

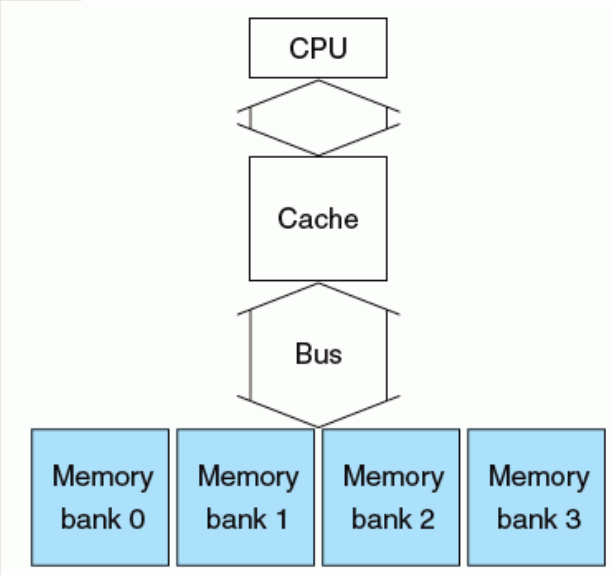
-Envio de uma palavra = 1 ciclo

Solução:

Tempo total = Envio de endereço + acesso inicial + envio de bloco

Tempo total = 1 + 2 x 15 + 2 x 1 = 33 ciclos

Exemplo: Tempo de Acesso de Memória Interleaved



Problema:

Dada uma cache com blocos de 4 palavras e considerando que existem 4 bancos de memória com largura de 1 palavra. Calcule tempo de acesso de memória dados o número de ciclos do barramento de cada operação:

- Envio de endereço = 1 ciclo

-Acesso inicial a uma palavra = 15 ciclos

-Envio de uma palavra = 1 ciclo

Solução:

Tempo total = Envio de endereço + acesso inicial + envio de bloco

Tempo total = 1 + 1 x 15 + 4 x 1 = 20 ciclos

Considerações Finais sobre Desempenho de Caches

- Aumento do desempenho da CPU
  - Miss penalty se torna mais significativo
- Redução da CPI base
  - Maior proporção do tempo é gasto em ciclos de espera pela memória (memory stalls)
- Aumento da frequencia do clock
  - Memory stalls gastam mais ciclos
- Cache não pode ser negligenciado no cálculo do desempenho
  - Muitos parâmetros para conseguir configuração certa de cache
- Organização da memória ajuda a melhorar desempenho de cache