UNIVALI

Laboratório de Sistemas
Embarcados e Distribuídos

Hierarquia de Memória

Parte II

Histórico de revisões

Revisão	Data	Responsável	Descrição
0.1	-x-	Prof. Cesar Zeferino	Primeira versão
0.2	03/2017	Prof. Cesar Zeferino	Revisão do modelo
0.3	06/2020	Prof. Cesar Zeferino	Atualização da bibliografia

1 Introdução

- **Objetivo**
  - Conhecer os princípios do funcionamento e do projeto de uma cache
- **Conteúdo**
  - Caches diretamente mapeadas e associativas
  - Desempenho de sistemas com caches
  - Projeto e dimensionamento de caches

Introdução

- **Bibliografia**
  - PATTERSON, David A.; HENNESSY, John L. Grande e rápida: explorando a hierarquia de memória. *In*: _____. **Organização e projeto de computadores**: a interface hardware/software. 4. ed. Rio de Janeiro: Campus, 2014.
    cap. 5.

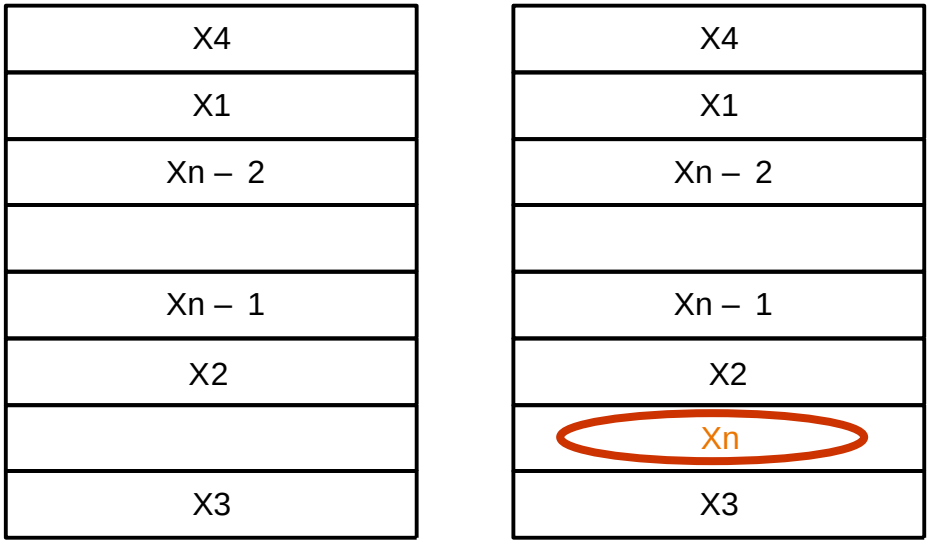
  - Edições anteriores
    - Patterson e Hennessy (2005, cap. 7)
    - Patterson e Hennessy (2000, cap. 7)

Introdução

- **A memória cache**
  - Introduzida em computadores no início dos anos 60
  - Curiosidades
    - O termo “cache” vem do verbo francês *cacher* (esconder)
    - Segundo o dicionário Webster, “cache” significa “um lugar seguro para esconder ou guardar coisas”
    - Em computação, o termo cache é usado para referenciar qualquer armazenamento usado para tirar proveito da localidade de acesso

Introdução

- **Cache muito simples**
  - Um bloco é igual a uma word (palavra)
  - Exemplo: A cache, imediatamente antes e após uma referência a uma word X_n que não estava na cache



Antes da referência a Xn

Após a referência a Xn

  - Problemas
    - Como saber se um item está ou não na cache?
    - Como encontrar esse item na cache?

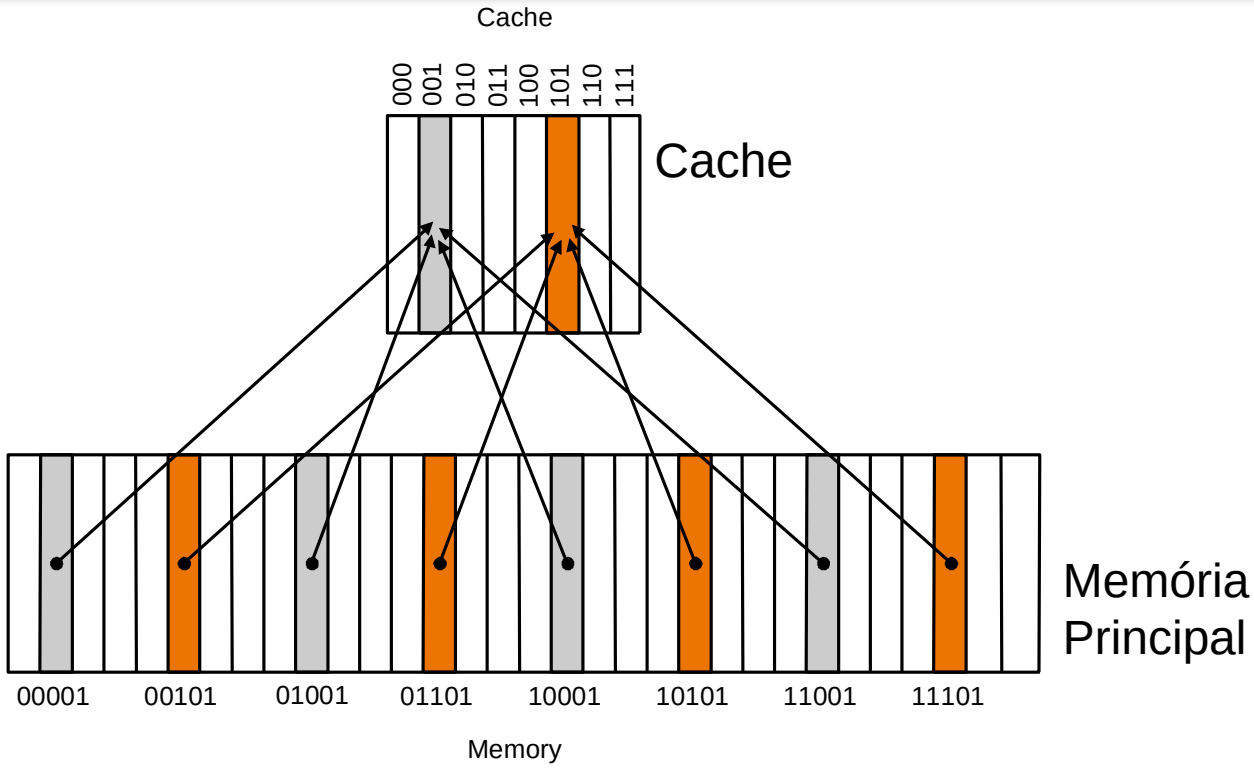
Mapeamento Direto

Como funciona o mapeamento direto

- Cada posição da memória é mapeada em uma posição da cache baseada no endereço do item na memória
- A posição do item na cache pode ser obtida pela equação

Posição na cache = (posição na memória) módulo (número de posições na cache)

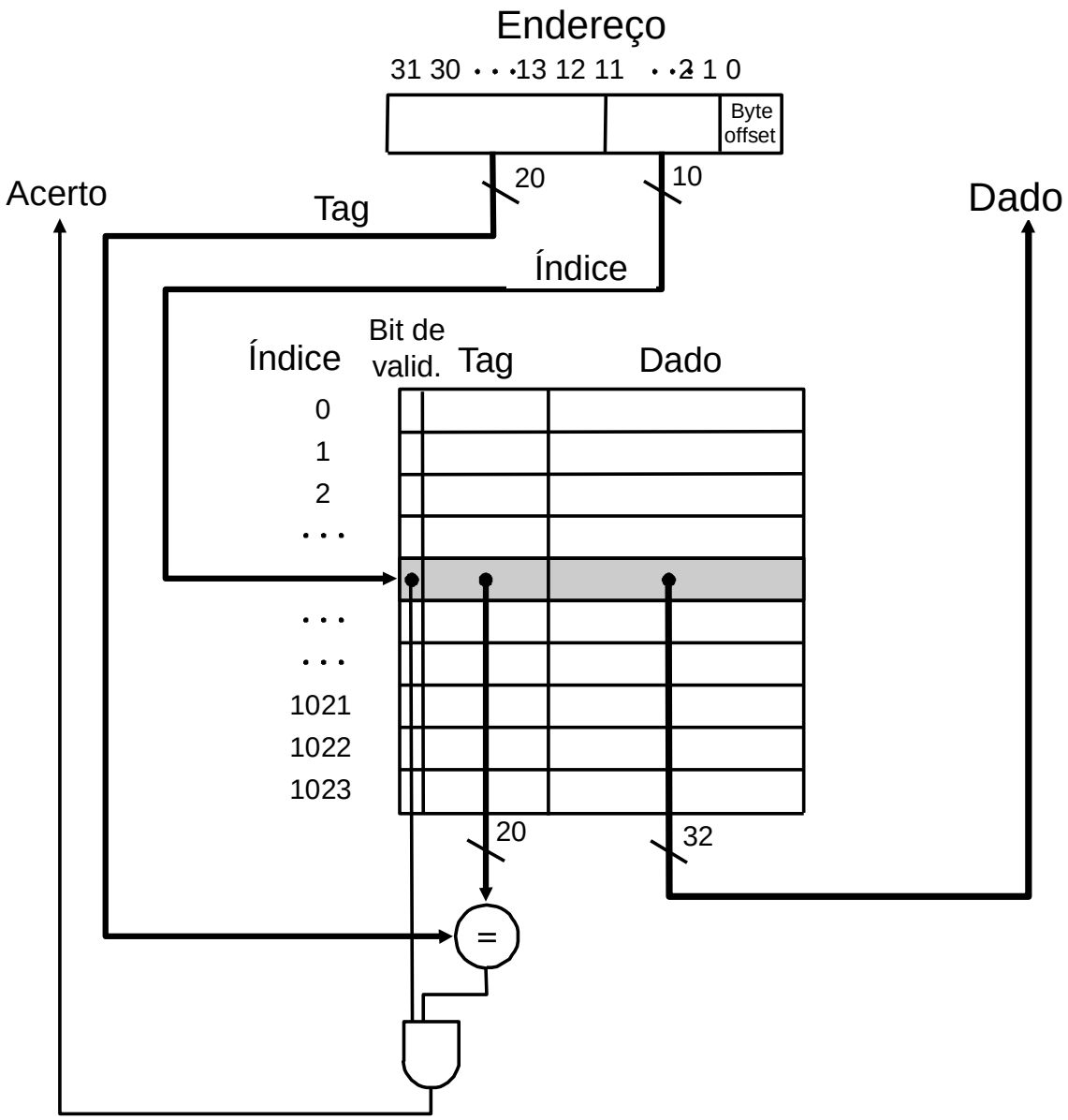
Se o número de posições (**entradas**) na cache for uma potência de 2, o módulo pode ser calculado usando os $\log_2$ bits menos significativos do endereço da memória (**índice**). No exemplo, os 3 bits mais à esquerda são suficientes.



Mapeamento Direto

- **Como funciona o mapeamento direto**
  - Um conjunto de bits menos significativos forma o **índice (index)** que define o bloco (entrada) da cache para o qual um bloco da memória será copiado
  - Diferentes blocos da memória concorrem pelo mesmo bloco da cache
  - Para saber qual posição da memória está sendo mantida na cache inclui-se um conjunto de **tags** na cache que contém essa informação
  - A tag contém a parte superior do endereço que não é usada para endereçar a entrada na cache (no exemplo anterior, os dois bits mais à esquerda). O índice não é armazenado.
  - Um **bit de validade** é usado para indicar se uma entrada da cache possui um item válido ou não.

Lendo um dado de uma Cache



Exemplo: Mapeamento direto

- **Considere uma cache diretamente mapeada, com 16 blocos e com uma palavra por bloco. Considere que a cache esteja inicialmente vazia e identifique as referências aos endereços abaixo como acerto ou falta e mostre o conteúdo final da cache.**
  - 1, 4,8, 5, 20, 17, 19, 56, 9, 11, 4, 43, 5, 6, 9, 17

Exemplo: Mapeamento direto

- **Bloco: 1 mod 16 = 1**

Endereço	Acerto ou falta
1	Falta
4	
8	
5	
20	
17	
19	
56	
9	
11	
4	
43	
5	
6	
9	
17	

Estado da Cache após o acesso

Bloco	Endereço
0	
1	1
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	

Exemplo: Mapeamento direto

- **Bloco: 4 mod 16 = 4**

Endereço	Acerto ou falta
1	Falta
4	Falta
8	
5	
20	
17	
19	
56	
9	
11	
4	
43	
5	
6	
9	
17	

Estado da Cache após o acesso

Bloco	Endereço
0	
1	1
2	
3	
4	4
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	

Exemplo: Mapeamento direto

- **Bloco: 8 mod 16 = 8**

Endereço	Acerto ou falta
1	Falta
4	Falta
8	Falta
5	
20	
17	
19	
56	
9	
11	
4	
43	
5	
6	
9	
17	

Estado da Cache após o acesso

Bloco	Endereço
0	
1	1
2	
3	
4	4
5	
6	
7	
8	8
9	
10	
11	
12	
13	
14	
15	

Exemplo: Mapeamento direto

- **Bloco: 5 mod 16 = 5**

Endereço	Acerto ou falta
1	Falta
4	Falta
8	Falta
5	Falta
20	
17	
19	
56	
9	
11	
4	
43	
5	
6	
9	
17	

Estado da Cache após o acesso

Bloco	Endereço
0	
1	1
2	
3	
4	4
5	5
6	
7	
8	8
9	
10	
11	
12	
13	
14	
15	

Exemplo: Mapeamento direto

- **Bloco: 20 mod 16 = 4**

Endereço	Acerto ou falta
1	Falta
4	Falta
8	Falta
5	Falta
20	Falta
17	
19	
56	
9	
11	
4	
43	
5	
6	
9	
17	

Estado da Cache após o acesso

Bloco	Endereço
0	
1	1
2	
3	
4	20
5	5
6	
7	
8	8
9	
10	
11	
12	
13	
14	
15	

Exemplo: Mapeamento direto

- **Bloco: 17 mod 16 = 1**

Endereço	Acerto ou falta
1	Falta
4	Falta
8	Falta
5	Falta
20	Falta
17	Falta
19	
56	
9	
11	
4	
43	
5	
6	
9	
17	

Estado da Cache após o acesso

Bloco	Endereço
0	
1	17
2	
3	
4	20
5	5
6	
7	
8	8
9	
10	
11	
12	
13	
14	
15	

Exemplo: Mapeamento direto

- **Bloco: 19 mod 16 = 3**

Endereço	Acerto ou falta
1	Falta
4	Falta
8	Falta
5	Falta
20	Falta
17	Falta
19	Falta
56	
9	
11	
4	
43	
5	
6	
9	
17	

Estado da Cache após o acesso

Bloco	Endereço
0	
1	17
2	
3	19
4	20
5	5
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	

Exemplo: Mapeamento direto

- **Bloco: 56 mod 16 = 8**

Endereço	Acerto ou falta
1	Falta
4	Falta
8	Falta
5	Falta
20	Falta
17	Falta
19	Falta
56	Falta
9	
11	
4	
43	
5	
6	
9	
17	

Estado da Cache após o acesso

Bloco	Endereço
0	
1	17
2	
3	19
4	20
5	5
6	
7	
8	56
9	
10	
11	
12	
13	
14	
15	

Exemplo: Mapeamento direto

- **Bloco: 9 mod 16 = 9**

Endereço	Acerto ou falta
1	Falta
4	Falta
8	Falta
5	Falta
20	Falta
17	Falta
19	Falta
56	Falta
9	Falta
11	
4	
43	
5	
6	
9	
17	

Estado da Cache após o acesso

Bloco	Endereço
0	
1	17
2	
3	19
4	20
5	5
6	
7	
8	56
9	9
10	
11	
12	
13	
14	
15	

Exemplo: Mapeamento direto

- **Bloco: 11 mod 16 = 11**

Endereço	Acerto ou falta
1	Falta
4	Falta
8	Falta
5	Falta
20	Falta
17	Falta
19	Falta
56	Falta
9	Falta
11	Falta
4	
43	
5	
6	
9	
17	

Estado da Cache após o acesso

Bloco	Endereço
0	
1	17
2	
3	19
4	20
5	5
6	
7	
8	56
9	9
10	
11	11
12	
13	
14	
15	

Exemplo: Mapeamento direto

- **Bloco: 4 mod 16 = 4**

Endereço	Acerto ou falta
1	Falta
4	Falta
8	Falta
5	Falta
20	Falta
17	Falta
19	Falta
56	Falta
9	Falta
11	Falta
4	Falta
43	
5	
6	
9	
17	

Estado da Cache após o acesso

Bloco	Endereço
0	
1	17
2	
3	19
4	4
5	5
6	
7	
8	56
9	9
10	
11	11
12	
13	
14	
15	

Exemplo: Mapeamento direto

- **Bloco: 43 mod 16 = 11**

Endereço	Acerto ou falta
1	Falta
4	Falta
8	Falta
5	Falta
20	Falta
17	Falta
19	Falta
56	Falta
9	Falta
11	Falta
4	Falta
43	Falta
5	
6	
9	
17	

Estado da Cache após o acesso

Bloco	Endereço
0	
1	17
2	
3	19
4	4
5	5
6	
7	
8	56
9	9
10	
11	43
12	
13	
14	
15	

Exemplo: Mapeamento direto

- **Bloco: 5 mod 16 = 5**

Endereço	Acerto ou falta
1	Falta
4	Falta
8	Falta
5	Falta
20	Falta
17	Falta
19	Falta
56	Falta
9	Falta
11	Falta
4	Falta
43	Falta
5	Acerto
6	
9	
17	

Estado da Cache após o acesso

Bloco	Endereço
0	
1	17
2	
3	19
4	4
5	5
6	
7	
8	56
9	9
10	
11	43
12	
13	
14	
15	

Exemplo: Mapeamento direto

- **Bloco: 6 mod 16 = 6**

Endereço	Acerto ou falta
1	Falta
4	Falta
8	Falta
5	Falta
20	Falta
17	Falta
19	Falta
56	Falta
9	Falta
11	Falta
4	Falta
43	Falta
5	Acerto
6	Falta
9	
17	

Estado da Cache após o acesso

Bloco	Endereço
0	
1	17
2	
3	19
4	4
5	5
6	6
7	
8	56
9	9
10	
11	43
12	
13	
14	
15	

Exemplo: Mapeamento direto

- **Bloco: 9 mod 16 = 9**

Endereço	Acerto ou falta
1	Falta
4	Falta
8	Falta
5	Falta
20	Falta
17	Falta
19	Falta
56	Falta
9	Falta
11	Falta
4	Falta
43	Falta
5	Acerto
6	Falta
9	Acerto
17	

Estado da Cache após o acesso

Bloco	Endereço
0	
1	17
2	
3	19
4	4
5	5
6	6
7	
8	56
9	9
10	
11	43
12	
13	
14	
15	

Exemplo: Mapeamento direto

- **Bloco: 17 mod 16 = 1**

Endereço	Acerto ou falta
1	Falta
4	Falta
8	Falta
5	Falta
20	Falta
17	Falta
19	Falta
56	Falta
9	Falta
11	Falta
4	Falta
43	Falta
5	Acerto
6	Falta
9	Acerto
17	Acerto

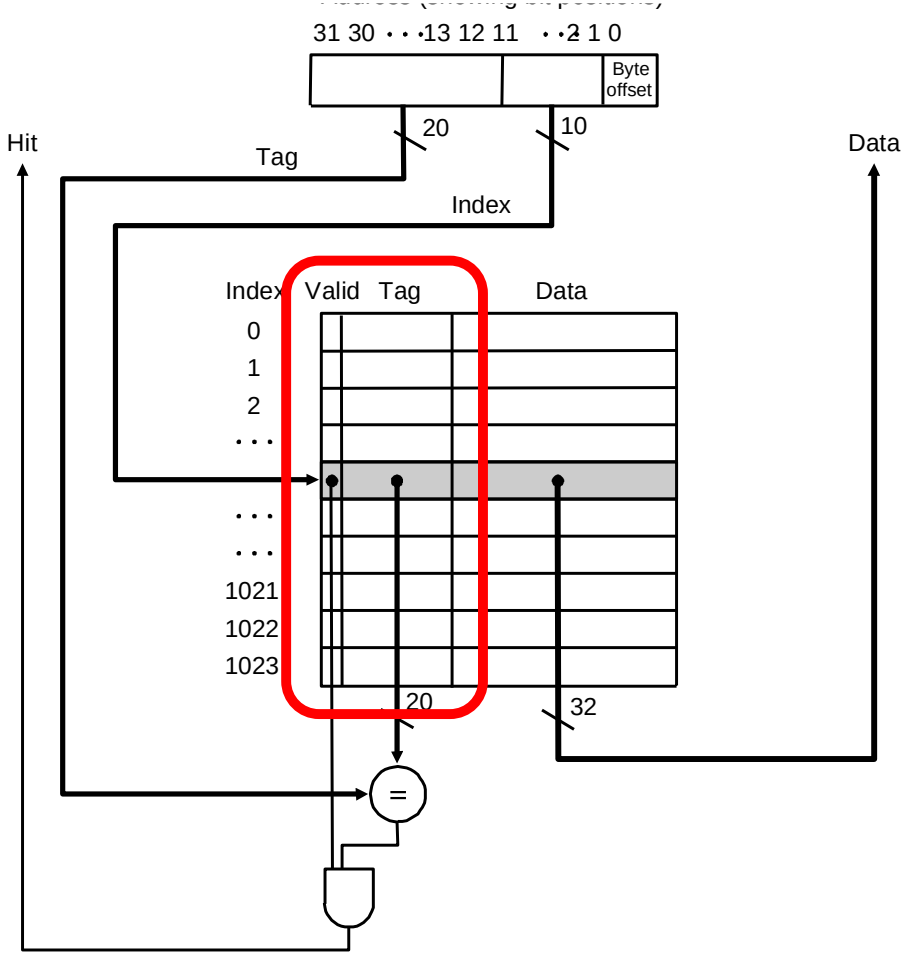
Estado da Cache após o acesso

Bloco	Endereço
0	
1	17
2	
3	19
4	4
5	5
6	6
7	
8	56
9	9
10	
11	43
12	
13	
14	
15	

Dimensionando a Cache

- **Os bits de tag e de validade resultam em um sobrecusto à cache**

- **Exemplo**
  - 1Kblocos
  - 1 palavra/bloco
  - 32 Kbits de dados
  - 21 Kbits de tags e valid
  - Sobrecusto = 21/(32+21)
    = 39,6%



Dimensionando a Cache

Cache diretamente mapeada para o MIPS

- Tamanho da cache = 2^n x {2^m x 32 + [32 – (n + m + 2)] + 1}
- onde
  - n = número de bits no índice
  - 2^n = número de blocos (entradas) na cache
  - 2^m = número de words em um bloco (entrada) da cache
  - 32 = número de bits em uma word
  - 2^m x 32 = número de bits em um bloco da cache
  - 32 – (n + m + 2) = tamanho da tag em bits
  - 1 = tamanho do bit de validade

- Logo

Tamanho da cache = 2^n x (2^m x 32 + 31 – n – m)

Para a cache do slide anterior

- 2^n = 1024 blocos
- n = 10
- 2^m = 1 word/bloco
- m = 0

Tamanho da cache = 2^{10} x (2^0 x 32 + 31 – 10 – 0)
= 1024 x (1 x 32 + 31 – 10 – 0)
= 1024 x (32 + 21)
= 1024 x 53
= 53 Kbits

Dimensionando a Cache

- **Tamanho da cache (em Kbits) para diferentes quantidades de linhas (blocos) e de palavras por bloco**

	Palavras/Bloco (2^m)				
Linhas (n)	**1**	**2**	**3**	**4**	**5**
10	53	84	115	147	179
11	104	166	229	292	355
12	204	328	454	580	707
13	400	648	899	1152	1405
14	784	1280	1783	2288	2795
15	1536	2528	3533	4544	5558
16	3008	4992	7003	9024	11051
17	5888	9856	13877	17920	21975
18	11520	19456	27498	35584	43694
19	22528	38400	54484	70656	86875
20	44032	75776	107945	140288	172726

Dimensionando a Cache (Overhead)

- **Sobrecusto (em Kbits)**

	Palavras/Bloco (2^m)				
Linhas (n)	**1**	**2**	**3**	**4**	**5**
10	39,6%	23,8%	16,8%	12,9%	10,5%
11	38,5%	22,9%	16,1%	12,3%	9,9%
12	37,3%	22,0%	15,4%	11,7%	9,4%
13	36,0%	21,0%	14,6%	11,1%	8,9%
14	34,7%	20,0%	13,8%	10,5%	8,4%
15	33,3%	19,0%	13,1%	9,9%	7,9%
16	31,9%	17,9%	12,3%	9,2%	7,3%
17	30,4%	16,9%	11,5%	8,6%	6,8%
18	28,9%	15,8%	10,6%	7,9%	6,3%
19	27,3%	14,7%	9,8%	7,2%	5,7%
20	25,6%	13,5%	8,9%	6,6%	5,1%

Taxa de falta x tamanho do bloco

- **Aumentar o tamanho do bloco implica em aumentar a capacidade de explorar a localidade espacial**

- **Ex.: Varredura de uma matriz 8x8 (linha-coluna)**

```
for(i=0; i<8; i++)
    for(j=0; j<8; j++)
        A[i,j] = i*8*4 + j*4;
```

Tamanho do bloco (Words)	Taxa de acerto (tamanho da cache de dados)			
	64 B	128 B	256 B	512 B
1	0	0	0	0
2	0,5	0,5	0,5	0,5
4	0,75	0,75	0,75	0,75
8	0,88	0,88	0,88	0,88
16	0,94	0,94	0,94	0,94
32		0,97	0,97	0,97
64			0,98	0,98
128				0,98

Taxa de falta x tamanho do bloco

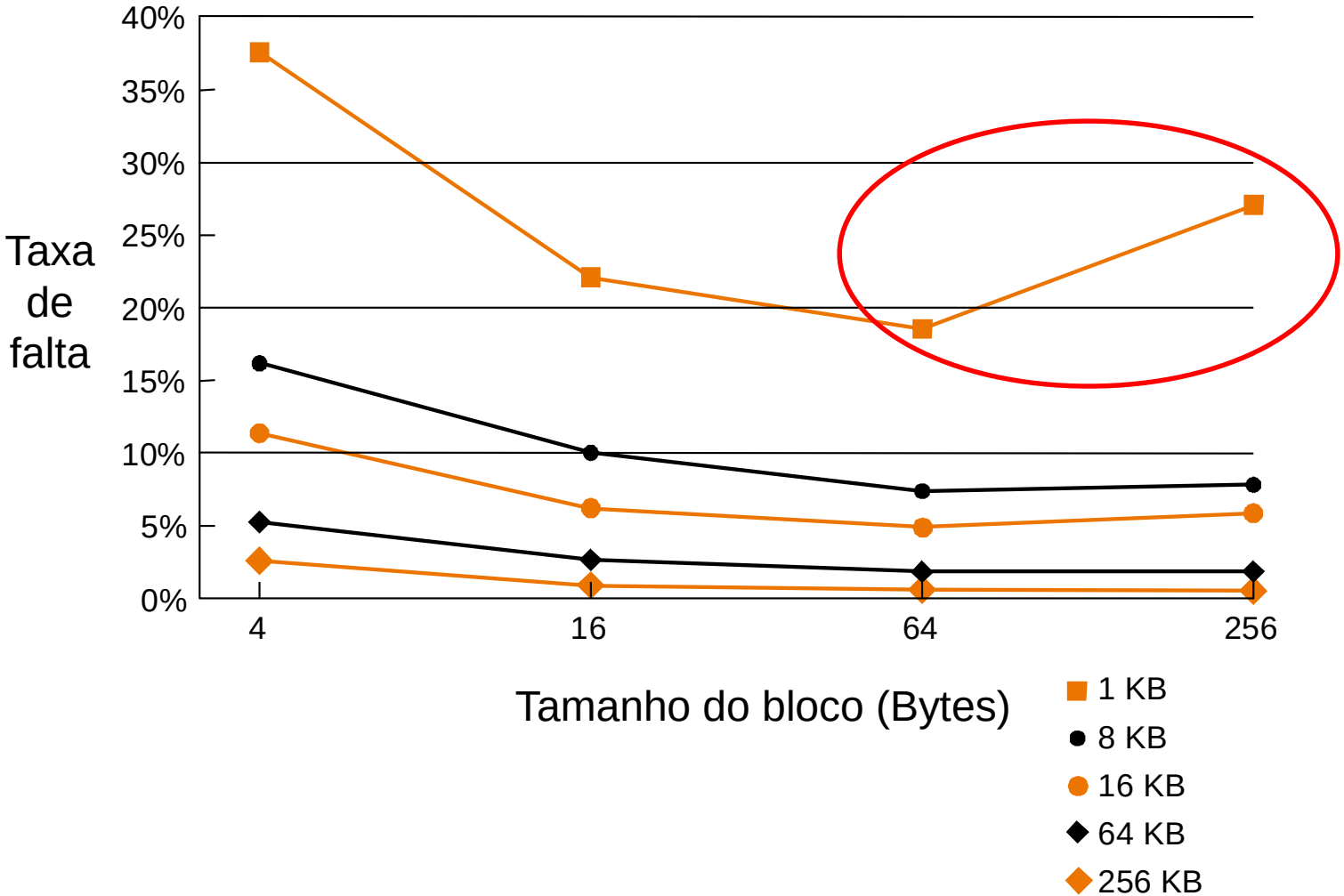
- Aumentar o tamanho do bloco também implica em reduzir a quantidade de blocos na cache

Tamanho do bloco (Words)	Quantidade de blocos (tamanho da cache)			
	64 B	128 B	256 B	512 B
1	16	32	64	128
2	8	16	32	64
4	4	8	16	32
8	2	4	8	16
16	1	2	4	8
32		1	2	4
64			1	2
128				1

- O que pode acontecer com isso?

Taxa de falta x tamanho do bloco

- **Ao reduzir o número de blocos, diminui-se a capacidade para explorar a localidade temporal e pode-se aumentar a taxa de faltas**



Tratando faltas na cache de instruções

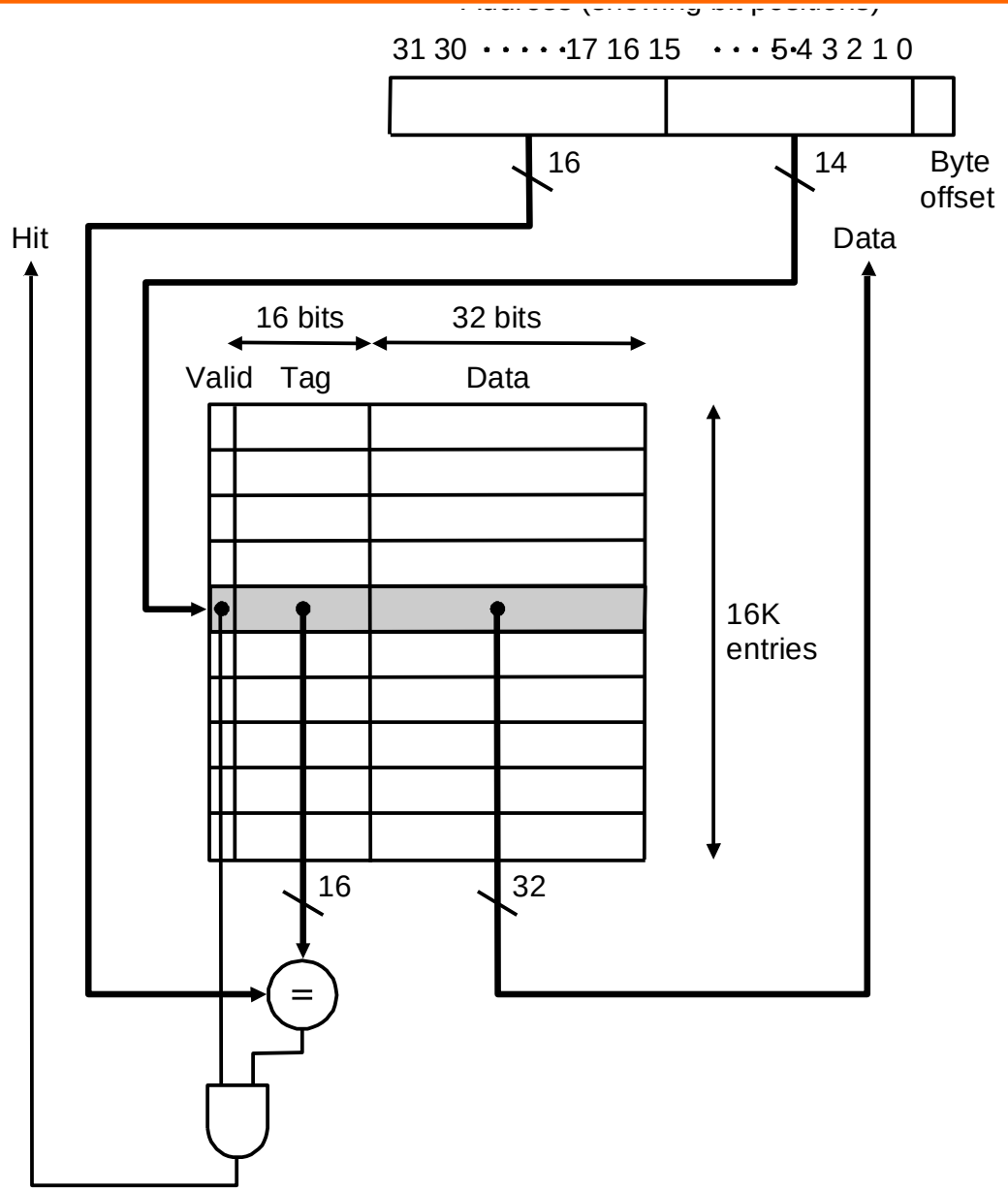
1. Enviar o valor do PC original (PC atual – 4) para a memória

2. Instruir a memória principal a realizar uma leitura e esperar que a memória complete seu acesso

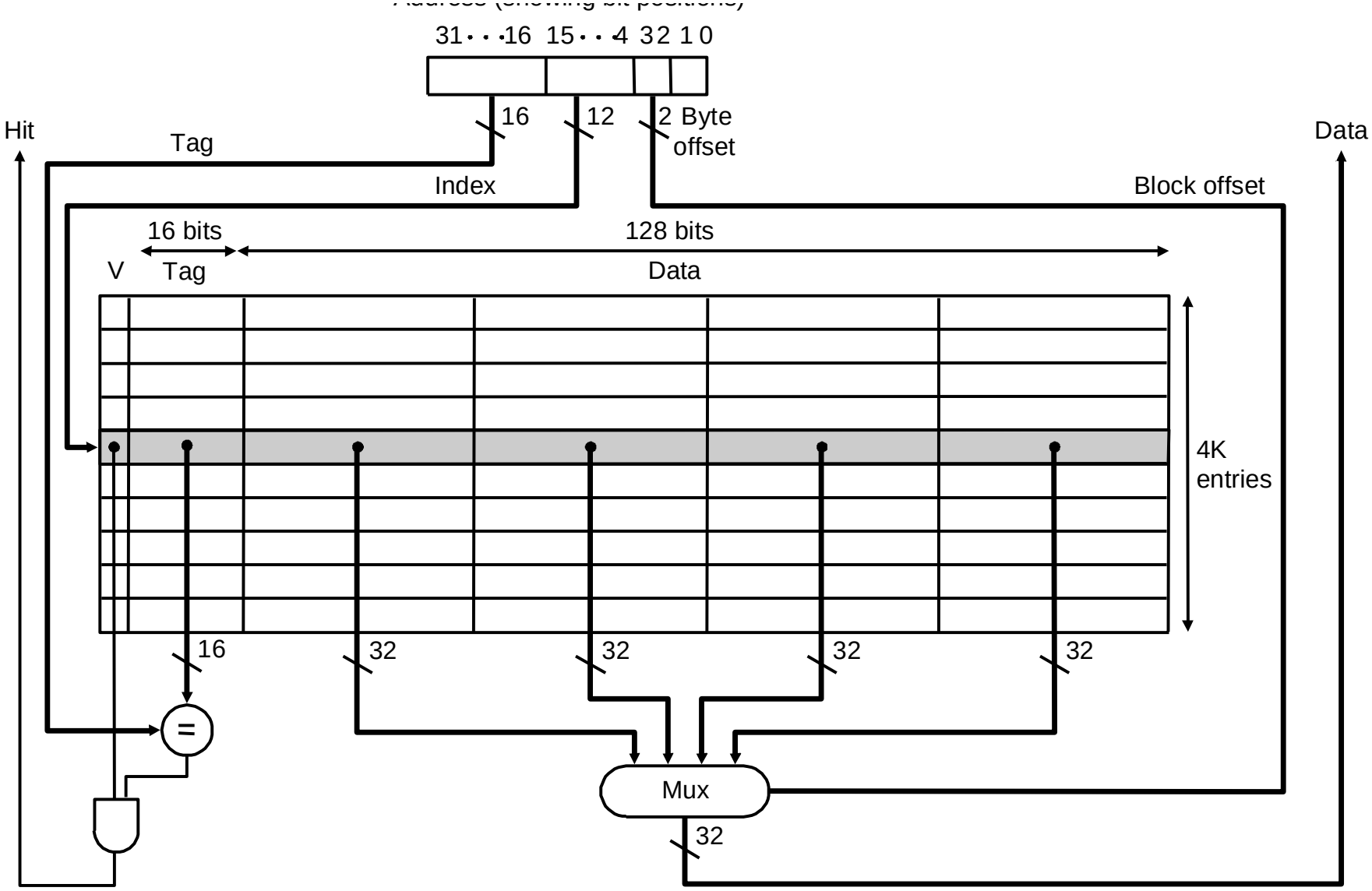
3. Escrever o bloco lido na entrada da cache, escrevendo também nessa entrada os bits mais significativos do endereço no campo tag e ligando o bit de validade

4. Reiniciar a execução da instrução, gerando uma nova busca da instrução na cache, mas dessa vez com a certeza de que ela será encontrada

A cache da DECStation 3100



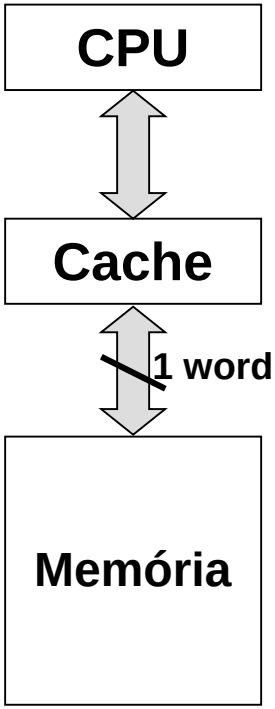
Uma cache de 64KB com blocos de 4 words



Tratando escritas

- **A cache deve ser consistente com a memória principal**
- **Opções de atualização da memória principal**
  - Write-through
    - Uma escrita sempre atualiza a cache e a memória principal, garantindo a consistência de dados nos dois níveis
    - Compromete o desempenho do processador pois uma escrita consome muitos ciclos, embora esse problema possa ser minimizado com o uso de um buffer de escrita
  - Write-back
    - Inicialmente, apenas a cache é atualizada. Após, quando o bloco é substituído, então ele é copiado para a memória principal
    - Oferece um desempenho melhor, mas sua implementação é mais complexa que a do write through

Projetando o sistema de memória



- **Considere o custo de busca de um bloco da memória na qual gasta-se**
  - 1 ciclo de clock do barramento de memória para enviar o endereço
  - 15 ciclos de clock do barramento de memória para cada acesso iniciado na DRAM
  - 1 ciclo de clock do barramento de memória para enviar uma word de dados

- **Considere um bloco igual a 4 words e uma memória com largura de barramento igual a 1 word**

Penalidade por falta = 1 + 4x15 + 4x1 = 65 ciclos de clock*

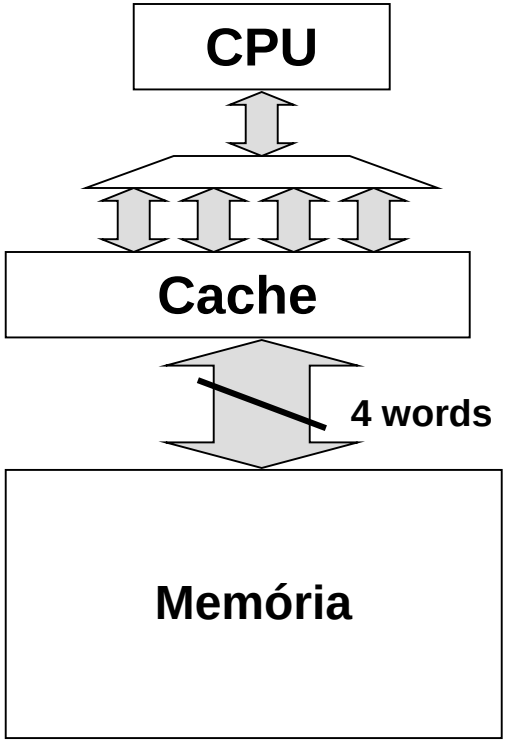
Largura de banda = (4x4)/65 = 0,25 byte/ciclo de clock*

*** do barramento de memória**

Projetando o sistema de memória

- Alternativas para reduzir a penalidade por falta

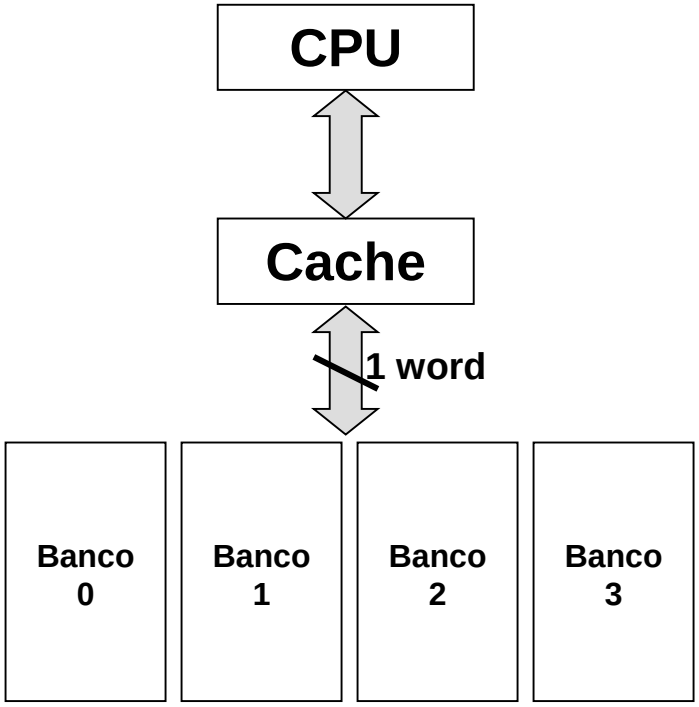
Memória larga



Penalidade por falta =
1 + 1x15 + 1x1 = 17 ciclos

Largura de banda =
(4x4)/17 = 0,94 byte/ciclo

Memória intercalada



Penalidade por falta =
1 + 1x15 + 4x1 = 20 ciclos

Largura de banda =
(4x4)/20 = 0,80 byte/ciclo

Estimando o desempenho da cache

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

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

$$\text{Ciclos de stall de leitura} = \frac{\text{Leituras}}{\text{Programa}} \times \text{Taxa de faltas de leitura} \times \text{Penalidade de falta de leitura}$$

$$\text{Ciclos de stall de escrita} = \left(\frac{\text{Escritas}}{\text{Programa}} \times \text{Taxa de faltas de escrita} \times \text{Penalidade de falta de escrita} \right) + \text{Stalls do buffer de escrita}$$

$$\text{Ciclos de stall de memória} = \frac{\text{Acessos à memória}}{\text{Programa}} \times \text{Taxa de faltas} \times \text{Penalidade de falta}$$

$$\text{Ciclos de stall de memória} = \frac{\text{Instruções}}{\text{Programa}} \times \frac{\text{Faltas}}{\text{Instrução}} \times \text{Penalidade de falta}$$

Ver exemplos no livro

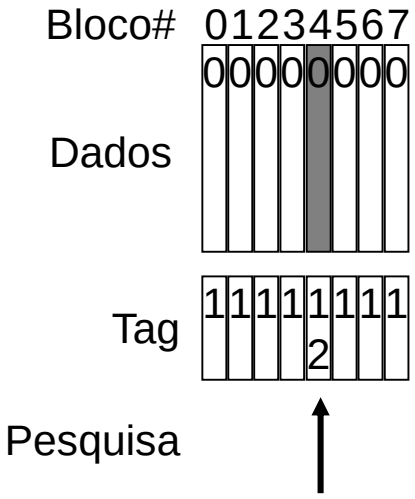
Associatividade em caches

- **Cache totalmente associativa**
  - Estrutura de cache em que um bloco pode ser posicionado em qualquer entrada da cache
- **Cache associativa por conjunto**
  - Estrutura de cache que possui um número fixo de entradas (no mínimo duas) nas quais um bloco pode ser posicionado
- **Cache diretamente mapeada (não associativa)**
  - Estrutura de cache em cada bloco da memória é mapeado exatamente para uma entrada na cache

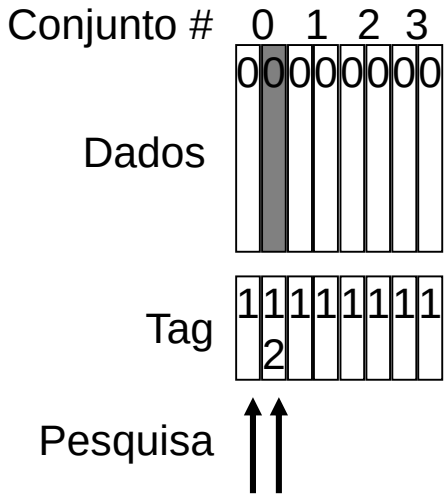
Associatividade em caches

- **A entrada (local) de um bloco de memória cujo endereço é 12 em uma cache com 8 blocos varia conforme o tipo de mapeamento**

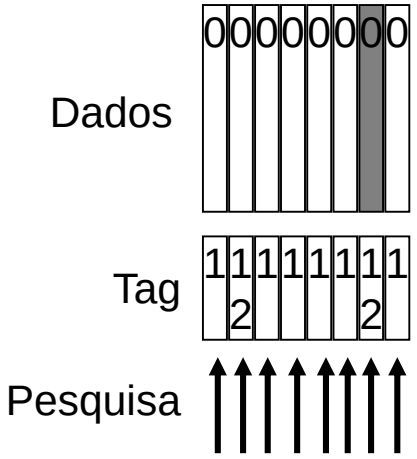
Diretamente Mapeada



Associativa por Conjunto



Totalmente Associativa



Notas:

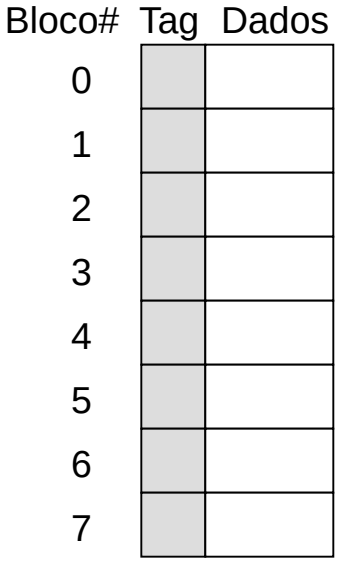
Na diretamente mapeada, cada bloco corresponde a um conjunto com um único bloco

Na totalmente associativa, há um único conjunto com 8 blocos

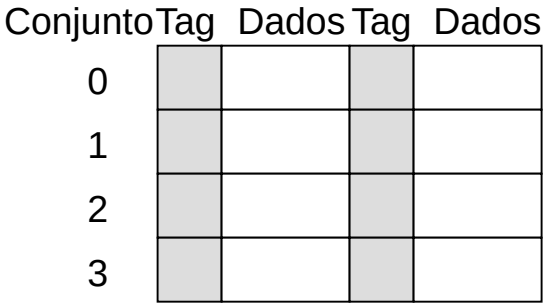
Associatividade em caches

Configurações possíveis para uma cache com oito blocos

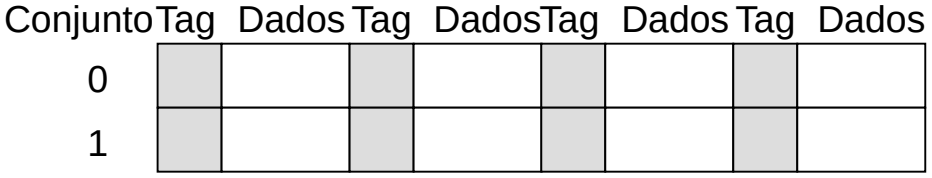
Associativa por conjunto de 1 via (Diretamente mapeada)



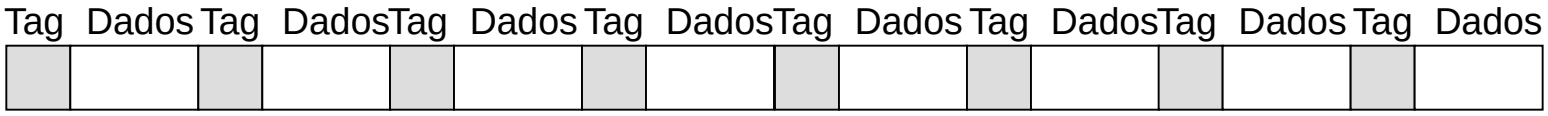
Associativa por conjunto de 2 vias



Associativa por conjunto de 4 vias



Associativa por conjunto de 8 vias (Totalmente associativa)



Mapeando blocos em caches associativas

- **Cache diretamente mapeada**

Posição na cache = (posição na memória) módulo (número de posições na cache)

- **Cache associativa por conjunto**
  - O bloco pode ser colocado em qualquer entrada do conjunto que é determinado por

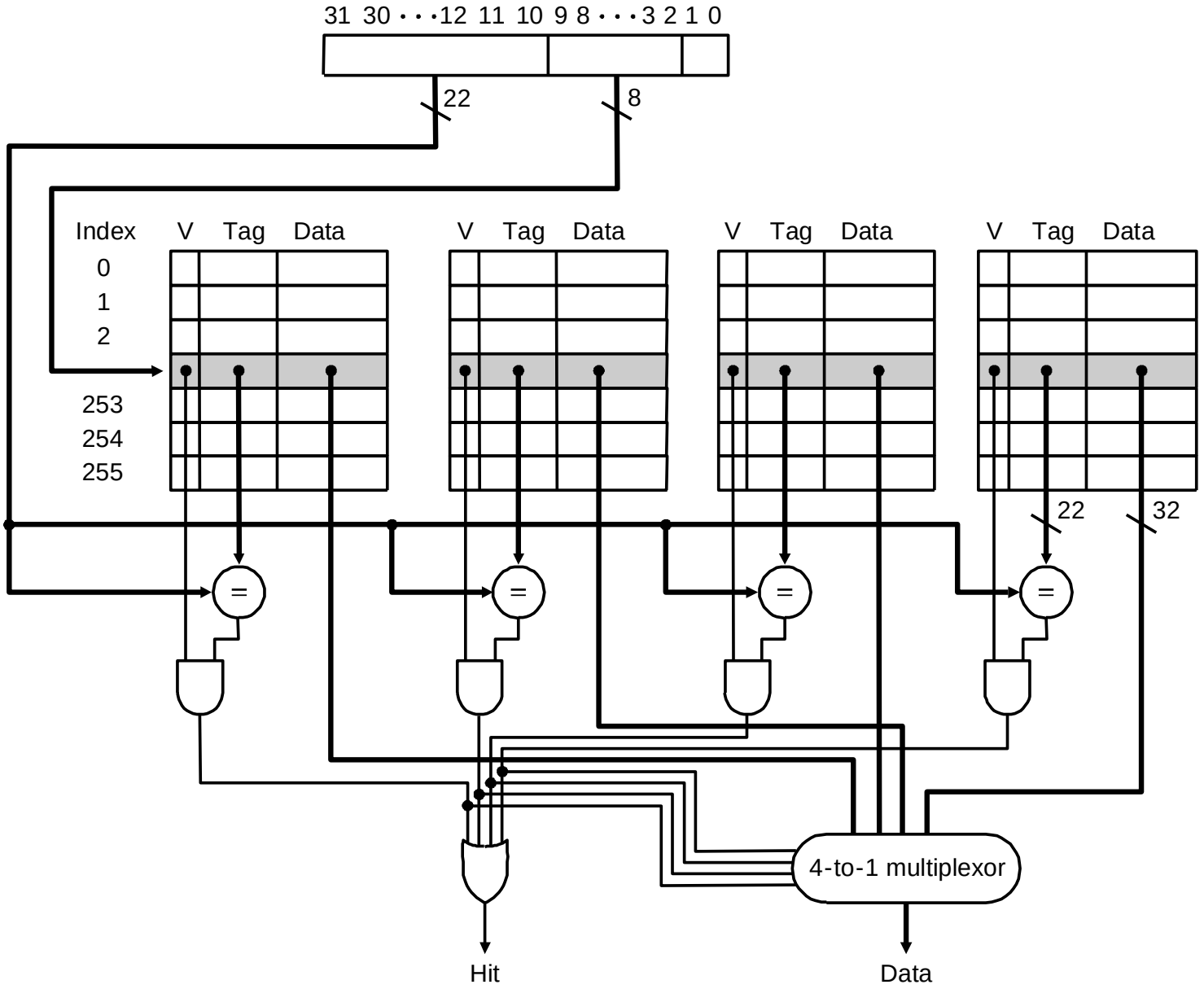
Conjunto na cache = (posição na memória) módulo (número de conjuntos na cache)

Ver exemplos no livro texto

Uma cache associativa por conjunto com 4 vias

NOTA:

Quanto maior for o grau de associativi-dade, menor será o índice e maior será o tag, e portanto a cache.



Tamanho das tags x Associatividade do conjunto

- **Quanto maior a associatividade, maior é sobrecusto (overhead) em tags na cache**

- **O custo em tags de uma cache associativa pode ser calculado por**

$$\text{Custo em tags} = w \times 2^n \times (32 - n - m - k)$$

- **Onde**
  - w : número de blocos por conjunto (número de vias)
  - 2^n : número de conjuntos (de blocos no mapeamento direto)
  - n : tamanho do índice para selecionar um conjunto
  - m : tamanho do índice para selecionar uma word no bloco
  - k : tamanho do índice para selecionar um byte na word

Tamanho das tags x Associatividade do conjunto

- **Custo em tags de caches de 4 Kblocos, onde**
  - 2^m = 4 words/bloco, logo m = 2
  - 2^k = 4 bytes/word, logo k = 2
  - Palavra de endereço de 32 bits e
  - w e n dependem do grau de associatividade

  - Diretamente mapeada (w = 1, n = 12)
    - Custo em tags = 1 x 4K x (32 – 12 – 2 – 2) = 4K x 16 = **64 Kbits**

  - Associativa por conjunto de duas vias (w = 2, n = 11)
    - Custo em tags = 2 x 2K x (32 – 11 – 2 – 2) = 4K x 17 = **68 Kbits**

  - Associativa por conjunto de quatro vias (w = 4, n = 10)
    - Custo em tags = 4 x 1K x (32 – 10 – 2 – 2) = 4K x 18 = **72 Kbits**

  - Totalmente associativa (w = 4K, n = 0)
    - Custo em tags = 4K x 1 x (32 – 0 – 2 – 2) = 4K x 28 = **112 Kbits**

Custo em dados = 4Kblocos x 4 words/bloco x 32 bits/word = 512 Kbits

Substituindo blocos em cache associativas

- **Para escolher um bloco a ser substituído em um conjunto, usa-se o esquema LRU (Least Recently Used – Usado Menos Recentemente)**
  - O bloco a ser substituído será aquele que não foi usado há mais tempo

Caches multinível

- Sistemas de memória modernos usam mais de um nível de cache (L1, L2, ...) dentro e/ou fora do processador
- Uso de caches L2 reduz a penalidade sofrida pelo processador quando ocorre uma falta na cache primária (L1) e a informação necessária já está na cache secundária (L2)
- Assim com a cache L1, a cache L2 pode ser implementada no mesmo chip do processador (ou fora dele)
- Alguns sistema suportam níveis adicionais de cache, além do L1 e do L2

Desempenho em caches multinível

- **Exemplo**
  - Sistema com cache L1 ideal
    - f_{clk} = 5 GHz, T_{clk} = 0,2 ns
    - $CPI_{cache\ ideal}$ = 1,0 pois a taxa de acertos = 1,0
  - Sistema com apenas uma cache L1 não ideal
    - Taxa de faltas por instrução$_{cache\ L1}$ = 2%
    - Tempo de acesso à memória principal = 100 ns
    - Penalidade de falta$_{cache\ L1}$ = 100 ns / 0,2 ns = 500 ciclos de clock
    - $CPI_{cache\ L1\ não\text{-}ideal}$ = 1,0 + (2% x 500) = 11,0
  - Sistema com caches L1 e L2 não ideais
    - L1: Taxa de faltas por instrução$_{cache\ L1}$ = 2%
    - L1: Tempo de acesso à cache L2 = 5 ns
    - L1: Penalidade de falta$_{cache\ L1}$ = 5 ns / 0,2 ns = 25 ciclos de clock
    - L2: Taxa de faltas por instrução$_{cache\ L2}$ = 0,5%
    - L2: Tempo de acesso à memória principal = 100 ns
    - L2: Penalidade de falta$_{cache\ L2}$ = 100 ns / 0,2 ns = 500 ciclos de clock
    - $CPI_{cache\ L1\ e\ L2\ não\ ideais}$ = 1,0 + (2% x 25) + (0,5% X 500) = 4,0

Conclusão
O sistema com cache L2 é 11/4 = **2,8 vezes mais rápido** que o sistema que possui apenas a cache L1.

Projeto de caches multinível

- **Caches de dois níveis permitem que**
  - A cache L1 seja projetada visando minimizar o tempo de acerto para produzir um ciclo de clock menor
  - A cache L2 seja projetada visando minimizar a taxa de faltas para reduzir a penalidade dos longos tempos de acesso à memória
  - A penalidade por falta da L1 seja reduzida pela presença da L2, o que permite que a L1 seja menor e tenha uma taxa de falta maior

- **Em comparação a sistemas com um único nível de cache, nos sistemas com dois níveis**
  - A L1 tem menos capacidade e utiliza blocos menores
  - A L2 tem mais capacidade e utiliza blocos maiores

Caches do Intel Pentium 4 e do AMD Opteron

Cache	Característica	Pentium 4	Opteron
L1	Organização	$D e $I divididas	$D e $I divididas
	Tamanho	8 KB ($D) + 96 KB ($I)*	64 KB ($D) + 64 KB ($I)
	Associatividade	Por conjunto de 4 vias	Por conjunto de 2 vias
	Substituição	LRU aproximada	LRU
	Tamanho do bloco	64 Bytes	64 Bytes
	Política de escrita	Write-through	Write-back
L2	Organização	Unificada	Unificada
	Tamanho	512 KB	1024 KB (1 MB)
	Associatividade	Por conjunto de 8 vias	Por conjunto de 16 vias
	Substituição	LRU aproximada	LRU aproximada
	Tamanho do bloco	128 Bytes	64 Bytes
	Política de escrita	Write-back	Write-back

- **Onde**
  - $D : cache de dados
  - $I : cache de instruções

* A L1 $I do Pentium 4 é uma cache de trace para instruções RISC

Caches no die do Intel Pentium 4

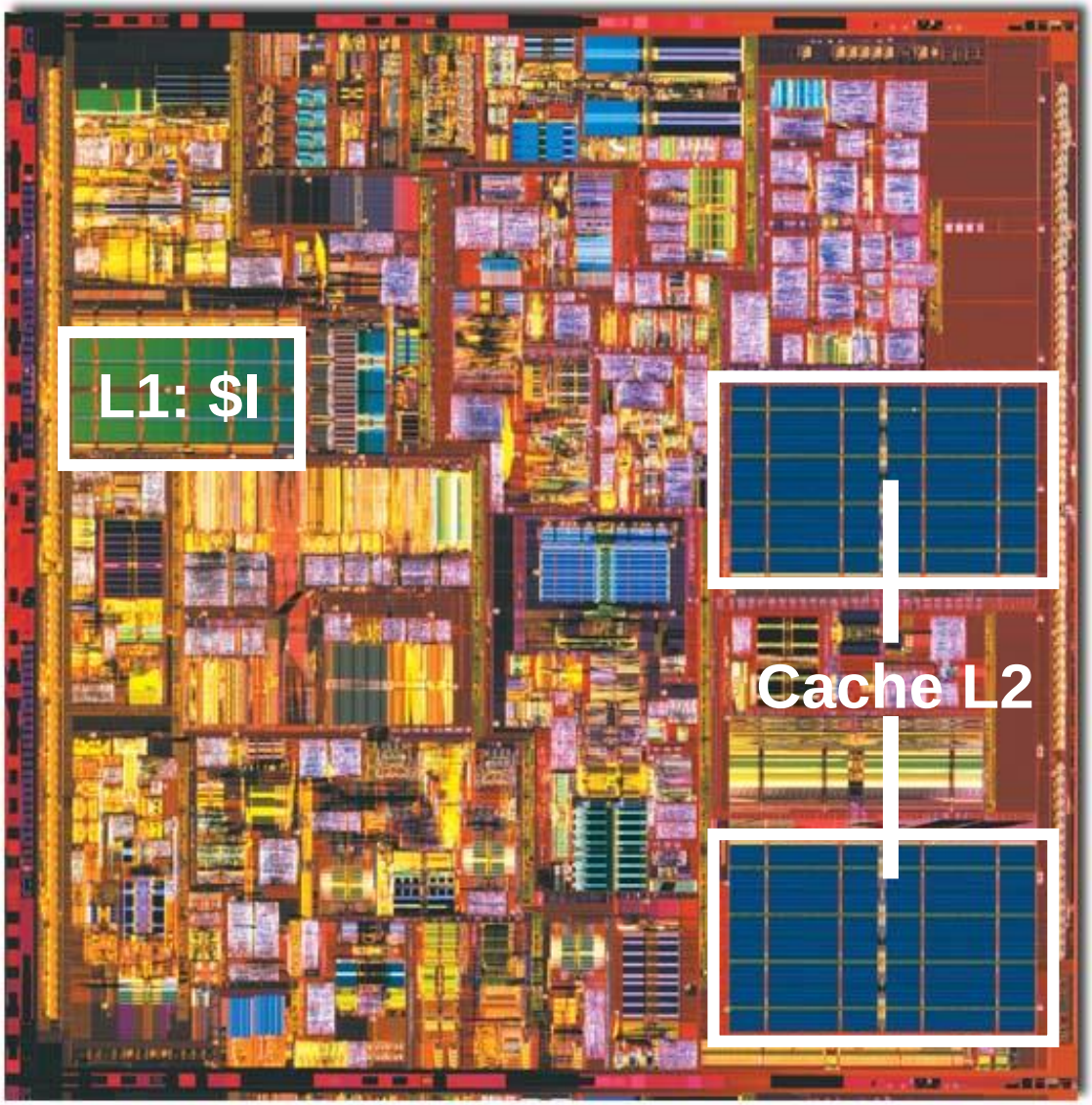
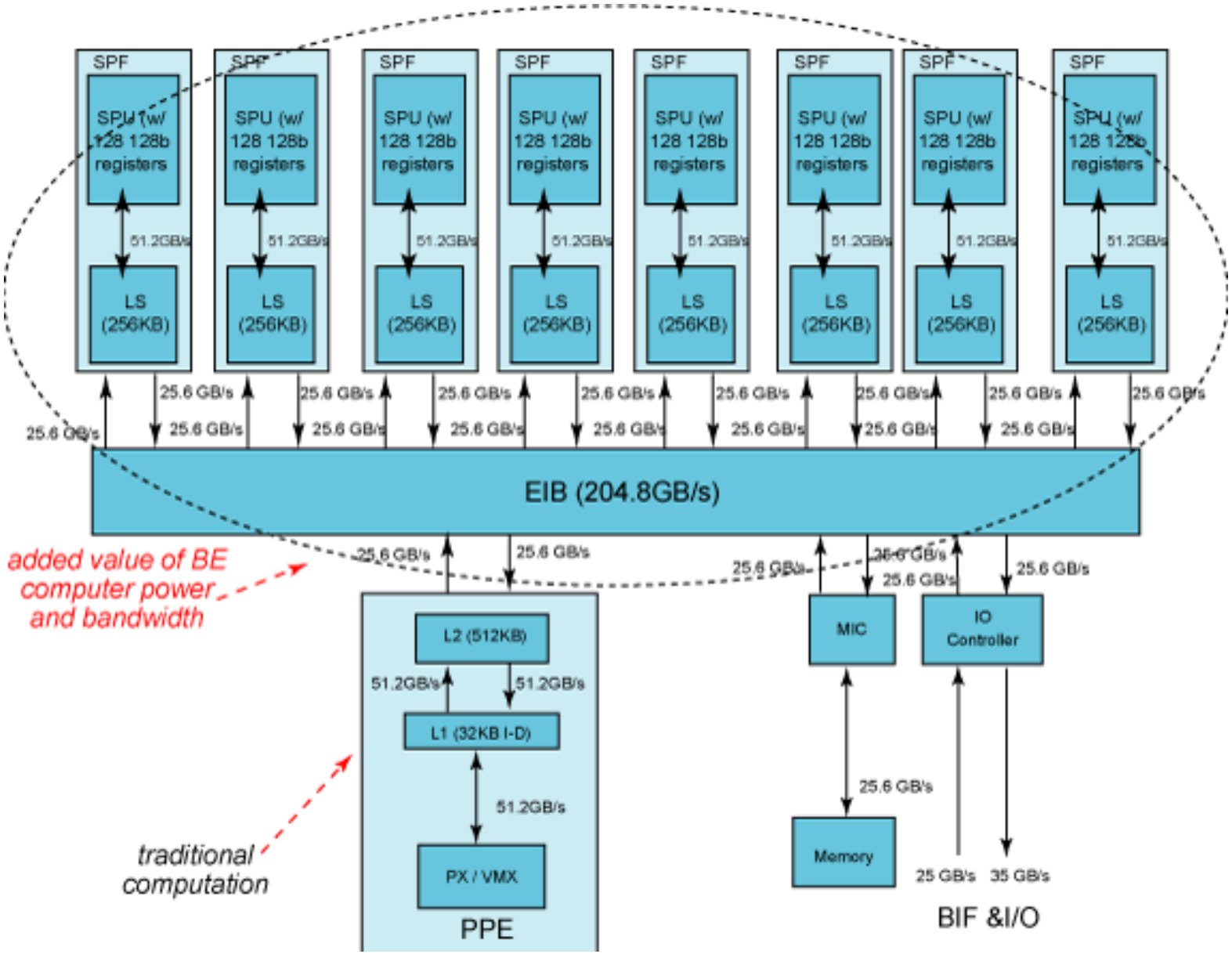
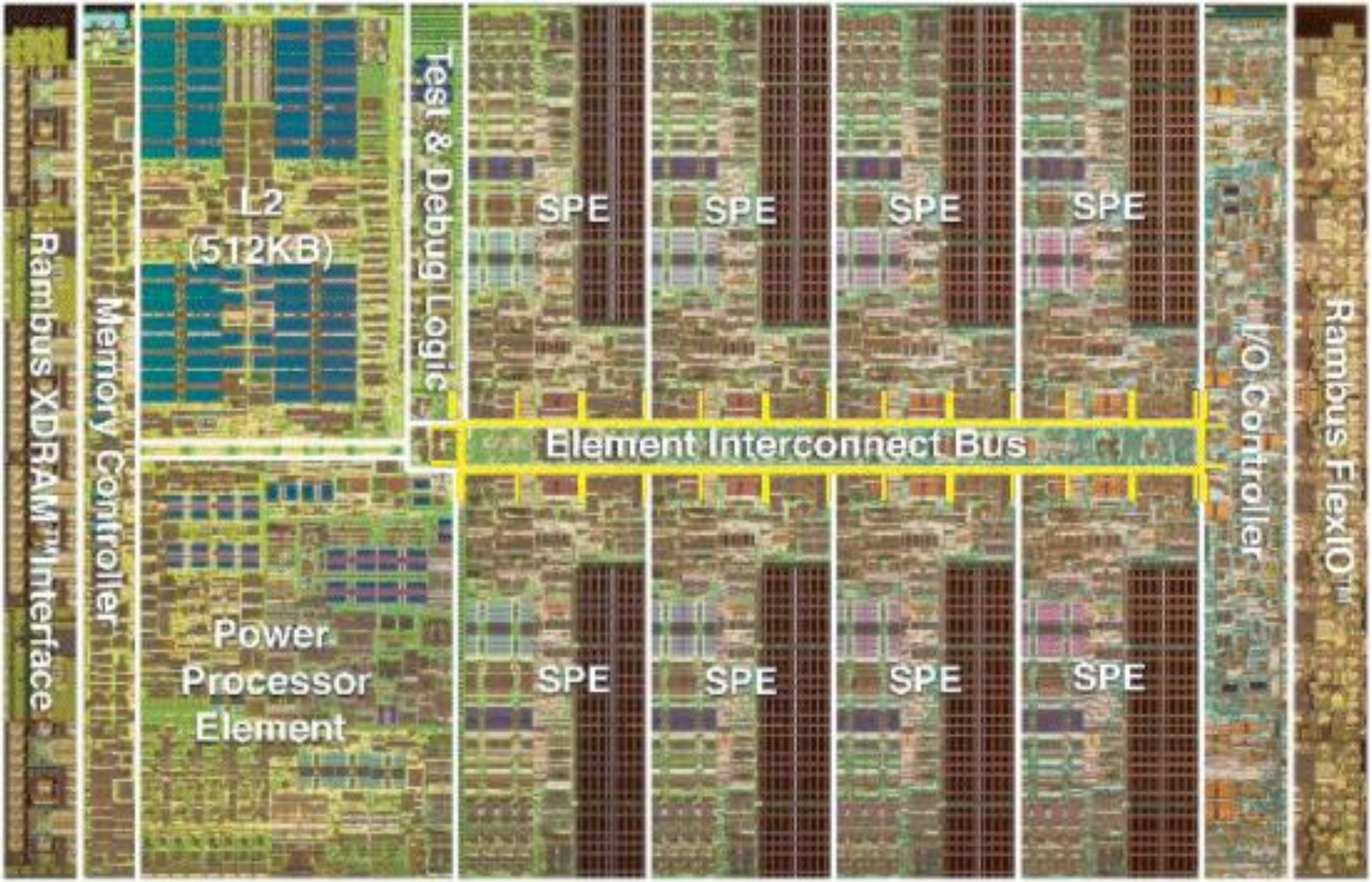


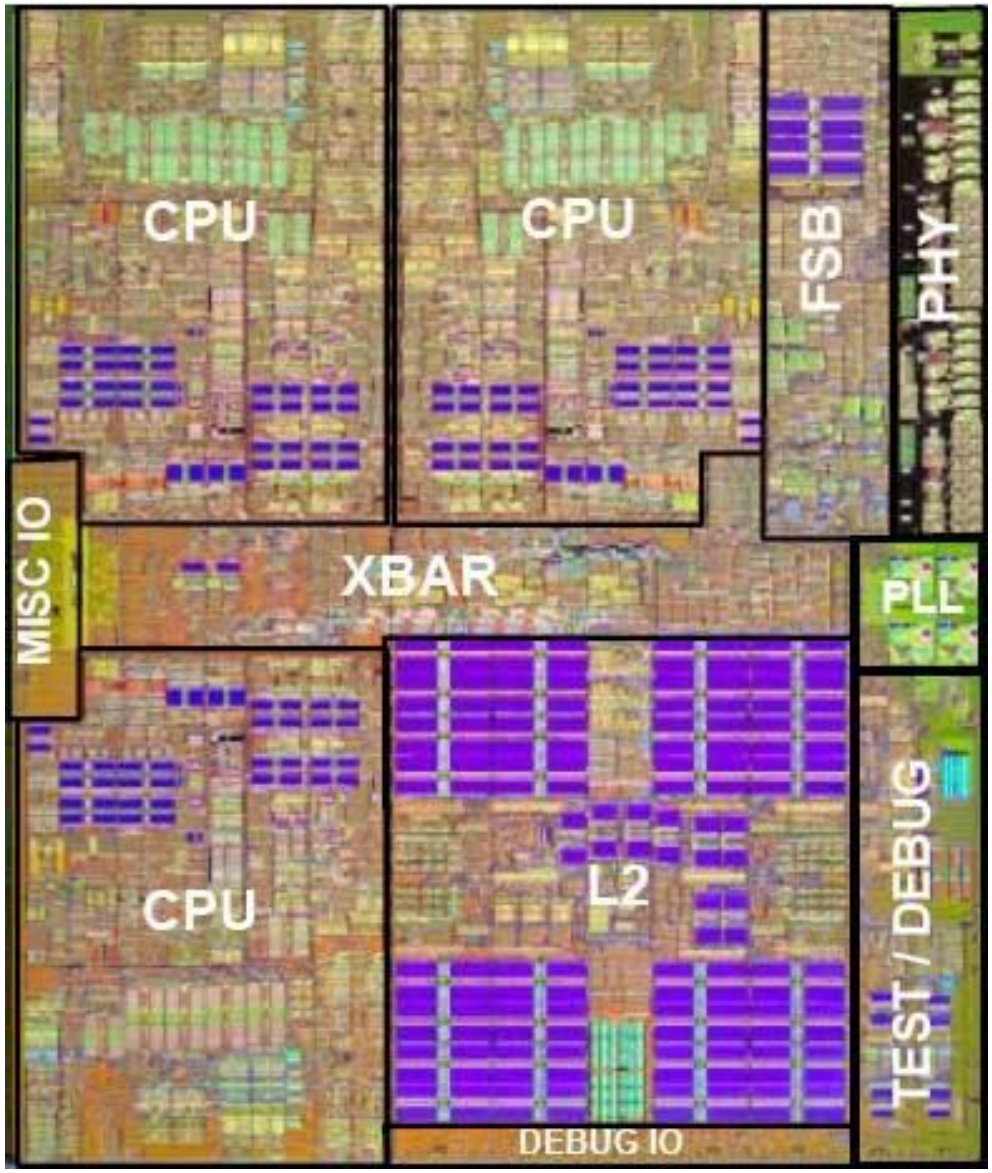
Diagrama do Cell BE



Die do Cell BE



Die do Xenon



Caches do Intel Core i7 Mobile

- **Características gerais do Core i7**
  - 774 milhões de transistores
  - Área do die = 296 mm2
  - 4 núcleos de execução
  - Suporte a Hyper-threading (2 threads simultâneas/núcleo)
  - Controladores de memória e PCIe integrados
  - 3 níveis de cache
    - L1: 64 K divididos
    - L2: 256 K unificados
    - L3: 8M unificados

Caches do Intel Core i7 Mobile