Aluna: **Camila Xavier de Medeiros**

Lista – Capítulo 5 - Hierarquia de Memória

1. [1.0] Caches são importantes para fornecer uma hierarquia de memória de alto desempenho para processadores. Abaixo está uma lista de referências a memória acessada por palavras de 64-bits:

Endereços: 0x03, 0xb4, 0x2b, 0x02, 0xbf, 0x58, 0xbe, 0x0e, 0xb5, 0x2c, 0xba, 0xfd

a. Para cada uma dessas referências, identifique no endereço os campos de tag e de índice para uma cache de mapeamento direto com 16 slots de uma palavra. Também liste se cada referência é um hit ou um miss, assumindo que a cache está inicialmente vazia.

Resposta:

Endereço	**Tag**	**Index**	**Hit/Miss**
0x03 - 0000 0011 (3)	0000 (0)	0011 (3)	Miss
0xb4 - 1011 0100 (180)	1011 (b)	0100 (4)	Miss
0x2b - 0010 1011 (43)	0010 (2)	1011 (b)	Miss
0x02 - 0000 0010 (2)	0000 (2)	0010 (2)	Miss
0xbf - 1011 1111 (191)	1011 (b)	1111 (f)	Miss
0x58 - 0101 1000 (88)	0101 (5)	1000 (8)	Miss
0xbe - 1011 1110 (190)	1011 (b)	1110 (e)	Miss
0x0e - 0000 1110 (14)	0000 (0)	1110 (e)	Miss
0xb5 - 1011 0101 (181)	1011	0101	Miss
0x2c - 0010 1100 (44)	0010	1100	Miss
0xba - 1011 1010 (186)	1011	1010	Miss
0xfd - 1111 1101 (253)	1111	1101	Miss

b. Repita a questão anterior considerando blocos de duas palavras e um tamanho total de oito slots. Também liste se cada referência é um hit ou um miss, assumindo que a cache está inicialmente vazia.

Resposta:

Endereço	Tag	Index	Hit/Miss	offset
0x03 - 0000 0011 (3)	0000 (0)	001 (1)	Miss	1
0xb4 - 1011 0100 (180)	1011 (b)	010 (2)	Miss	0
0x2b - 0010 1011 (43)	0010 (2)	101 (5)	Miss	1
0x02 - 0000 0010 (2)	0000 (0)	001 (1)	hit	0
0xbf - 1011 1111 (191)	1011 (b)	111 (7)	Miss	1
0x58 - 0101 1000 (88)	0101 (5)	100 (4)	Miss	0
0xbe - 1011 1110 (190)	1011 (b)	111 (7)	hit	0
0x0e - 0000 1110 (14)	0000 (0)	111 (7)	Miss	0
0xb5 - 1011 0101 (181)	1011 (b)	010 (2)	hit	1
0x2c - 0010 1100 (44)	0010 (2)	110 (6)	Miss	0
0xba - 1011 1010 (186)	1011 (b)	101 (5)	Miss	0
0xfd - 1111 1101 (253)	1111 (f)	110 (6)	Miss	1

c. Você é solicitado a otimizar um projeto de uma cache para as referências dadas. Existem três projetos de cache de mapeamento direto possíveis. Considere que a cache pode armazenar 8 palavras. Qual a melhor configuração?

- C1 tem blocos de 1 palavra,
- C2 tem blocos de 2 palavras,
- C3 tem blocos de 4 palavras

Resposta:

	Cache 1		Cache 2		Cache 3	
Endereço	**index**	**hit/miss**	**index**	**hit/miss**	**index**	**hit/miss**
0x03 - 0000 0011 (3)	3	Miss	1	Miss	0	Miss
0xb4 - 1011 0100 (180)	4	Miss	2	Miss	1	Miss
0x2b - 0010 1011 (43)	3	Miss	1	Miss	0	Miss
0x02 - 0000 0010 (2)	2	Miss	1	Miss	0	Miss
0xbf - 1011 1111 (191)	7	Miss	3	Miss	1	Miss
0x58 - 0101 1000 (88)	0	Miss	0	Miss	0	Miss
0xbe - 1011 1110 (190)	6	Miss	3	Hit	1	Hit
0x0e - 0000 1110 (14)	6	Miss	3	Miss	1	Miss
0xb5 - 1011 0101 (181)	5	Miss	2	Hit	1	Miss
0x2c - 0010 1100 (44)	4	Miss	2	Miss	1	Miss
0xba - 1011 1010 (186)	2	Miss	1	Miss	0	Miss
0xfd - 1111 1101 (253)	5	Miss	2	Miss	1	Miss

Analisando a tabela podemos concluir:

1) Na **cache 1** tem uma taxa de falta de 100% e 324 ciclos (12* 25 +12*2);
2) Na **cache 2** tem uma taxa de falta de 83% e 286 ciclos (10* 25 +12*3);
3) Na **cache 3** tem uma taxa de falta de 92% e 335 ciclos (11* 25 +12*5);

   Logo, a **cache 2** tem uma melhor configuração.

Obs: Temos que levar em conta que quando agrupamos demais não é vantajoso interessante para o tempo (quando não há localidade espacial), pois pode aumentar a taxa de miss trazendo palavras que talvez nem sejam usadas além também de ter o tempo de carregamento de bloco, quanto maior o bloco, mais demorada a operação.

2. [1.5] Por convenção, uma cache é nomeada de acordo com a quantidade de dados que contém (ou seja, uma cache de 4 KiB pode conter 4 KiB de dados); no entanto, as caches também armazenam informações como tags e bits de validade. Para este exercício, você vai examinar como a configuração da cache pode afetar a quantidade total de bits necessária para implementação, bem como o desempenho da cache. Para todas as partes, suponha que as caches são endereçadas por byte, e que os endereços e palavras são de 64 bits.

a. Calcule o número total de bits necessários para implementar uma cache de 64 KiB com blocos de duas palavras.

Resposta:

1) A cache tem 64 KIB, logo temos 2^16 bytes na cache;
2) Como cada bloco tem 2 palavras e cada palavra tem 8 bytes: 2^4 bytes (16 bytes);
3) Com esses valores podemos calcular o número de slots: (2^16 / 2^4) = 2^12;
4) Considerando que temos 12 bits de índice, 1 bit de bloco offset e 3 bits de word, encontramos uma tag com 28 bits;
5) Tendo tudo isso como base, nossa cache terá: (1+48+128) * 2^12 = **177 * 2^12 bits.**

b. Calcule o número total de bits necessários para implementar uma cache de 128 KiB com blocos de 32 palavras. Quanto maior é esta cache do que a cache de 64 KiB descrita na letra a.? (Observe que, mudando o tamanho do bloco, nós dobramos a quantidade de dados por slot sem dobrar o tamanho total da cache)

.

Resposta:

1) A cache tem 128 KIB, logo temos 2^17 bytes na cache;
2) Como cada bloco tem 32 palavras e cada palavra tem 8 bytes: 2^8 (258) bytes;
3) Com esses valores podemos calcular o número de slots: (2^17/2^8) = 2^9;
4) Considerando que temos 9 bits de índice, 5 bits de bloco offset e 3 bits de word, encontramos uma tag que possui 47 bits;
5) Tendo tudo isso como base, nossa cache terá: (1+47+(64*32)) * 2^9 = **2096 * 2^9 bits. Que nos dará um aumento de aproximadamente 48%.**

c. Explique por que a cache de 128 KiB, apesar da maior quantidade de dados, pode ter um desempenho mais lento do que a primeira cache.

Resposta: Apesar da maior quantidade de dados, a cache de 128 KIB por ser grande acaba levando para a memória endereços que não foram solicitados, aumentando, assim, a taxa de falta para os endereços que realmente foram solicitados. Além disso, quando ocorre uma MISS (falta), o bloco todo tem que ser copiado, e como esse bloco é maior, o seu tempo de cópia também se torna maior.

3. [2.0] Para o projeto de cache de mapeamento direto com um endereço de 64 bits, os seguintes bits de endereço são usados para acessar a cache.

- TAG: bits 63 – 11
- Index: Bits 10-5
- Offset: bits 4-0

a. Qual é o tamanho do bloco da cache (em palavras)?

Resposta: Tomando como referência o livro base da disciplina e considerando que cada palavra tem 8 bytes, temos 3 bits de offset de palavra (que faz referência a cada byte) e 2 bits para offset de bloco (que faz referencia para cada palavra do bloco), logo temos 2^2 = 4 palavras no total.

b. Quantos blocos a cache tem?

Resposta: Tomando como base que existem 6 bits de índice. Isso nos mostra que há 2^6 = **64 linhas na cache.**

c. Qual é a razão entre bits totais necessários para a implementação da cache sobre os bits de armazenamento de dados?

Resposta: Levando em conta que a cache armazena um total de 64 linhas * 4 palavras por bloco * 8 bytes por palavra = 2048 bytes = **16384 bits**.
Porém, além disso, cada linha contém 54 bits de tag e 1 bit de validade. Logo, o total de bits necessários são: 8192 + 54*64 + 1*64 = **19904 bits**.
Com isso a razão é : 19904/16384 = **1,21.**

Na utilização da cache descrita acima os seguintes endereços foram acessados (decimal): 0,4,16,132,232,160,1024,30,140,3100,180,2180

d. Para cada endereço acessado, liste (1) sua TAG, índice e deslocamento, (2) se foi um hit ou um miss, e (3) que slots foram substituídos (se houver).

Resposta:

Endereço	tag	index	offset	hit/miss	slots trocados
0x00 - 000 000 000 000 (0)	0	0	0	Miss	x
0x04 - 000 000 000 100 (4)	0	0	4	Hit	x
0x10 - 000 000 010 000 (16)	0	0	16	Hit	x
0x84 - 000 010 000 100 (132)	0	4	4	Miss	x
0xe8 - 000 011 101 000 (232)	0	7	8	Miss	x
0xa0 - 000 010 100 000 (160)	0	5	0	Miss	x
0x400 - 010 000 000 000 (1024)	1	0	0	Miss	0x00-0x1F
0x1e - 000 000 011 110 (30)	0	0	30	Miss	0x400-0x41F
0x8c - 000 010 001 100 (140)	0	4	12	Hit	x
0xc1c - 110 000 011 100 (3100)	3	0	28	Miss	0x00-0x1F
0xb4 - 000 010 110 100 (180)	0	5	20	Hit	x
0x884 - 100 010 000 100 (2180)	2	4	4	Miss	0x80-0x9f

e. Qual é a razão de hits?

Resposta: 4/12 = **33%.**

f. Mostre o estado final do cache, a cada entrada válida representada como <índice, tag, dados>. Por exemplo: <0,3, Mem(0xC00) – Mem[0xC1F].

Resposta:

< 0, 3, Mem[0xc00] → Mem[0xc1f] >
< 4, 2, Mem[0x880] → Mem[0x89f] >
< 5, 0, Mem[0x0a0] → Mem[0x0bf] >
< 7, 0, Mem[0x0e0] → Mem[0x0ff] >

4. [1.5] O tamanho do bloco da cache (B) pode afetar tanto a taxa de faltas quanto a latência de uma falta. Assumindo uma máquina com um CPI base de 1, e uma média de 1,38 referências (ambas instruções e dados) por instrução, encontre o tamanho do bloco que minimiza a latência total de faltas dadas as seguintes taxas de falta para vários tamanhos de bloco.

a. Bloco de 8 bytes: 4%
b. Bloco de 16 bytes: 3%
c. Bloco de 32 bytes: 2%
d. Bloco de 64 bytes: 1,5%
e. Bloco de 128 bytes: 1%

a. Qual é o tamanho ideal do bloco para uma penalidade de 20 × B Ciclos?

Resposta: Como estamos usando o acesso médio, que é igual para todas as instruções, vamos apenas considerar o produto **Miss rate x Penalty**:

1) Bloco de 8 bytes: 0,04 * (20 * 8) = **6,4**;
2) Bloco de 16 bytes: 0,03 * (20 * 16) = **9,6**;
3) Bloco de 32 bytes: 0,02 * (20 * 32) = **12,8**;
4) Bloco de 64 bytes: 0,015 * (20 * 64) = **19,2**;
5) Bloco de 128 bytes: 0,01 * (20 * 128) = **25,6**;
   Logo, o tamanho ideal é o de **8 Bytes**.

b. Qual é o tamanho ideal do bloco para uma penalidade de 24 + B Ciclos?

Resposta:

1) Bloco de 8 bytes: 0,04 * (24 + 8) = **1,28**;
2) Bloco de 16 bytes: 0,03 * (24 + 16) = **1,20**;
3) Bloco de 32 bytes: 0,02 * (24 + 32) = **1,12**;
4) Bloco de 64 bytes: 0,015 * (24 + 64) = **1,32**;
5) Bloco de 128 bytes: 0,01 * (24 +128) = **1,52**;
   Logo, o tamanho ideal é o de **32 bytes**.

c. Para uma penalidade constante, qual é o tamanho ideal do bloco?

Resposta: O tamanho ideal do bloco para uma penalidade constante seria de **128 bytes**, pois quando minimizamos a taxa de falta também conseguimos minimizar a penalidade (latência).

5. [1.5] O uso de vários níveis de caches é uma técnica importante para superar o tamanho limitado da cache de primeiro nível em detrimento de sua velocidade. Considere um processador com os seguintes parâmetros:

a. CPI base (sem stalls de memória): 1.5;
b. Velocidade processador: 2GHz;
c. Tempo de acesso a memória principal: 100ns;
d. Taxa falta L1 por instrução: 8% **;
e. Tempo de acesso da Cache L2 – Mapeamento direto: 12 ciclos;
f. Taxa de Falta com cache L2 – mapeamento direto: 3,2%;
g. Tempo de acesso da Cache L2 – 8-way associativa: 28 ciclos;
h. Taxa de falta com cache L2- 8-way associativa: 1.6%.

**Taxa de falta de cache de primeiro nível é por instrução. Suponha que o número total de faltas na cache L1 (instruções e dados combinados) equivale a 8% do número de instruções. Calcular o CPI para o processador na tabela usando:

obs: Levar em consideração que para um processador de **2 GHz** temos um período de **0,5ps**. Logo, precisamos de **100/0,5 = 200 ciclos** para termos acesso à memória principal.

a. Apenas uma cache de primeiro nível:

Resposta:

CPI = 1,5 + (0,08 * 200) = **17,5**.

b. Uma cache de segundo nível de mapeamento direto:

Resposta:

CPI = 1,5 + (0,08 * (12 + (0,032 * 200))) = 1,5 + (0,08 *18,4) = **2,972**.

c. Um segundo nível cache 8 way set-associativa:

Resposta:

CPI = 1,5 + (0,08 * (28 + (0,016 * 200))) = 1,5 + (0,08 * 31,2) = **3,996**.

d. Como esses números mudam se o acesso principal à memória tempo dobra? (Dar cada mudança como um CPI absoluto e uma mudança percentual.) Observe até que ponto um cache L2 pode esconder os efeitos de uma memória lenta.

Resposta:

Considerando que o tempo de acesso à memória principal dobra, temos agora 200ns em um processador de 2 GHz. Logo, precisamos de **200/0,5 = 400 ciclos** para termos acesso à memória principal.

(Cache L1):

CPI = 1,5 + (0,08 * 400) = **33,5**.

Mudança percentual = 33,5/17,5 = **91%** de aumento.

(Cache L2 - Mapeamento Direto):

CPI = 1,5 + (0,08 * (12 + (0,032 * 400))) = 1,5 + (0,08 * 24,8) = **3,484**.

Mudança percentual = 3,484/2,972 = **17%** de aumento.

(Cache L2 - 8-Way Set):

CPI = 1,5 + (0,08 * (28 + (0,016 * 400))) = 1,5 + (0,08 * 34,4) = **4,252**.

Mudança percentual = 4,252/3,996 = **6%** de aumento.

6. [1.5] A memória virtual usa uma tabela de páginas para rastrear o mapeamento de endereços virtuais para endereços físicos. Este exercício mostra como isso tabela deve ser atualizada à medida que os endereços são acessados. Os seguintes dados constituem um fluxo de endereços virtuais. Assuma páginas de 4 KiB, uma TLB totalmente associativa de quatro entradas e a política de substituição LRU. Se as páginas devem ser trazidas a partir do disco, incrementar o próximo maior número de páginas.

Decimal	4669	2227	13916	34587	48870	12608	49225
Decimal	4653	2195	13900	34619	48838	12624	49193

TLB

Valid	Tag	Número página física	Tempo último acesso
1	0xb	12	4
1	0x7	4	1
1	0x3	6	3
0	0x4	9	7

TABELA DE PÁGINAS

Índice	Valid	Página Física ou Disco
0	1	5
1	0	Disco

2	0	Disco
3	1	6
4	1	9
5	1	11
6	0	Disco
7	1	4
8	0	Disco
9	0	Disco
A	1	3
B	1	12
C	0	Disco

a. Para cada acesso mostrado acima, liste se o acesso é um hit ou erro no TLB; se o acesso é um hit ou erro na tabela de página, se o acesso é uma falta de página e o estado atualizado do TLB.

Resposta:

Endereço	Tag	Miss/hit na TLB
0x123d - 0001 0010 0011 1101	0x1	Miss na TLB; Hit na tabela de página; Falta de página.
0x08b3 - 0000 1000 1011 0011	0x0	Miss na TLB; Hit na tabela de página.
0x365c - 0011 0110 0101 1100	0x3	Hit na TLB.
0x871b - 1000 0111 0001 1011	0x8	Miss na TLB; Hit na tabela de página; Falta de página.
0xbee6 - 1011 1110 1110 0110	0xb	Miss na TLB; Hit na tabela de página.
0x3140 - 0011 0001 0100 0000	0x3	Hit na tabela de página.

0xc049 - 1100 0000 0100 1001	0xc	Miss na TLB; Miss na tabela de página.
0x122d - 0001 0010 0010 1101	0x1	Miss na TLB; Hit na tabela de página.
0x893 - 0000 0010 0010 1101	0x0	Miss na TLB; Hit na tabela de página.
0x364c - 0011 0110 0100 1100	0x3	Hit na tabela de página.
0x873b - 1000 0111 0011 1011	0x8	Miss na TLB; Hit na tabela de página; Falta de página.
0xbec6 - 1011 1110 1100 0110	0xb	Miss na TLB; Hit na tabela de página.
0x3150 - 0011 0001 0101 0000	0x3	Hit na tabela de página;
0xc029 - 1100 0000 0010 1001	0xc	Miss na TLB; Miss na tabela de página.

obs: a outra tabela está em anexo no documento de excel.

b. Repita a questão da letra a. anterior, mas desta vez use páginas de 16 KiB. Quais seriam algumas das vantagens de ter um tamanho de página maior? Quais são algumas das desvantagens?

Resposta:

Endereço	Tag	Hit/Miss na TLB
0x123d - 0001 0010 0011 1101	0x1	Hit na tabela de página; Falta de página.
0x08b3 - 0000 1000 1011 0011	0x0	Hit na TLB
0x365c - 0011 0110 0101 1100	0x3	Hit na TLB
0x871b - 1000 0111 0001 1011	0x8	Hit na tabela de página; Falta de página.
0xbee6 - 1011 1110 1110 0110	0xb	Hit na TLB
0xc049 - 1100 0000 0100 1001	0x3	Hit na TLB

0xc049 - 1100 0000 0100 1001	0xc	Hit na TLB
0x122d - 0001 0010 0010 1101	0x1	Hit na TLB
0x893 - 0000 0010 0010 1101	0x0	Hit na TLB
0x364c - 0011 0110 0100 1100	0x3	Hit na TLB
0x873b - 1000 0111 0011 1011	0x8	Hit na TLB
0xbec6 - 1011 1110 1100 0110	0xb	Hit na tabela de página
0x3150 - 0011 0001 0101 0000	0x3	Hit na TLB
0xc029 - 1100 0000 0010 1001	0xc	Hit na TLB

Conclusão: Logo, a vantagem é que uma página maior pode diminuir o número de TLB MISS, mas a desvantagem é que pode aumentar a segmentação.

obs: a outra tabela está em anexo no documento de excel.

c. Repita a questão da letra a., mas desta vez use uma TLB 2-way associativa:

Resposta:

Endereço	Tag	Hit/Miss na TLB
0x123d - 0001 0010 0011 1101	0x1	Miss na TLB; Hit na tabela de página; Falta de página.
0x08b3 - 0000 1000 1011 0011	0x0	Miss na TLB; Hit na tabela de página.
0x365c - 0011 0110 0101 1100	0x3	Hit na TLB
0x871b - 1000 0111 0001 1011	0x8	Miss na TLB; Hit na tabela de página; Falta de página.
0xbee6 - 1011 1110 1110 0110	0xb	Miss na TLB; Hit na tabela de página.
0xc049 - 1100 0000 0100 1001	0x3	Hit na TLB
0xc049 - 1100 0000 0100 1001	0xc	Miss na TLB; Hit na tabela de página; Falta de página.

0x122d - 0001 0010 0010 1101	0x1	Miss na TLB; Hit na tabela de página; Falta de página.
0x893 - 0000 0010 0010 1101	0x0	Hit na TLB
0x364c - 0011 0110 0100 1100	0x3	Hit na TLB
0x873b - 1000 0111 0011 1011	0x8	Miss na TLB; Hit na TLB
0xbec6 - 1011 1110 1100 0110	0xb	Hit na TLB
0x3150 - 0011 0001 0101 0000	0x3	Hit na TLB
0xc029 - 1100 0000 0010 1001	0xc	Hit na TLB

obs: a outra tabela está em anexo no documento de excel.

7. [1.0] Qual o propósito de se usar código corretor de erros em um sistema hierárquico de memória e como funciona o código de Hamming com distância 3?

Resposta:

O propósito é que não podemos garantir que, quando a informação vai do disco para a memória e da memória para a cache, os componentes físicos envolvidos nessa passagem (como as placas, os fios, os barramentos…) não vão induzir erros nessas informações (troca de 0 e 1 entre os bits). Logo, esses erros podem ser muito ruins ao funcionamento geral do computador e com isso criaram vários métodos de solucionar esses erros.

Entre esses métodos está o código de Hamming. O código de Hamming com distância mínima de 3 fornece correção do erro no caso de 1 bit e detecção do erro no caso de 2 bits. Utilizando, bits de paridade para conseguir chegar nesses resultados.

O algoritmo começa numerando os bits à esquerda. Assim, todas as posições dos bits que são uma potência de 2 são considerados bits de paridade. Nesse caso, então, possuímos 4 bits de paridade e cada um vai verificar a paridade de um conjunto de bits de dados. Vale lembrar que os valores dos bits de paridade indicam quais bits estão com erro.

Se a quantidade de bits 1 for ímpar, a paridade é ímpar e dessa forma esse bit assume o valor de 1 para fazer com que o total de bits 1 seja par. Porém, caso a quantidade de bits 1 seja par, o bit 1 assume o valor de 0. No final do algoritmo, se juntarmos os bits de paridade e formarmos uma palavra, se a palavra que se formou for 1010, isso indicará que o bit 10 foi invertido. Logo, é assim que funciona o código de Hamming para a detecção de erros em um sistema hierárquico de memória.

8. (extra – 1,5) Considere o programa abaixo que acessa os elementos de um vetor que tem os seguintes valores

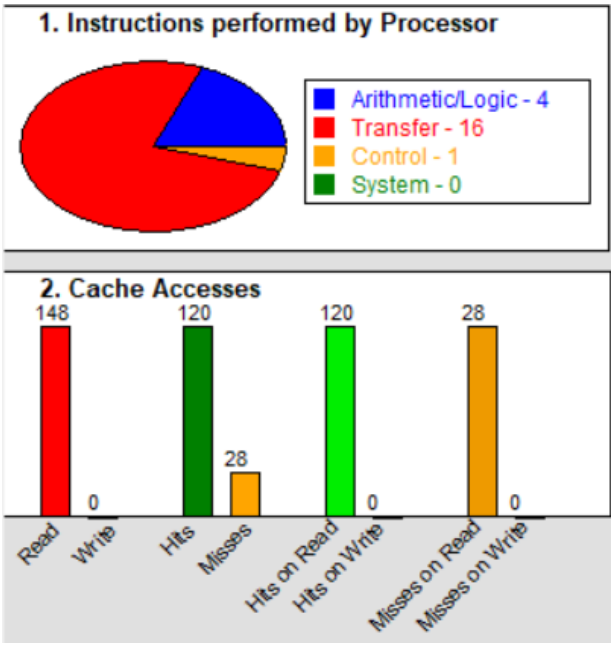
```
        start:
         addi x6, x0, 64
         lw x7, 104(x0)
         lw x7, 116(x0)
         lw x7, 132(x0)
         lw x7, 120(x0)
         lw x7, 180(x0)
         lw x7, 168(x0)
         lw x7, 176(x0)
         lw x7, 324(x0)
         lw x7, 136(x0)
loop:    lw x7, 144(x0)
         lw x7, 116(x0)
         lw x7, 272(x0)
         lw x7, 120(x0)
         lw x7, 124(x0)
         lw x7, 136(x0)
         lw x7, 168(x0)
         addi x6, x6, -1
         addi x0, x0, 0
         addi x0, x0, 0
         bne x6, x0, loop
         addi x0, x0, 0
         addi x0, x0, 0
         addi x0, x0, 0
         halt
        vetor: .word 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18,
        19, 20,  21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35 , 36, 37,
        38, 39, 40, 41,  42, 43, 44, 45, 46, 47 ,48, 49, 50, 51, 52, 53, 54, 55, 56
```

Simule o programa no CompSim faça um relato comparativo nos casos enumerados abaixo, considerando as seguintes características da memória cache: Número de faltas e se existe localidade espacial e temporal.

- **Caso 1:** Compare os resultados obtidos com uma cache de 16 palavras versus uma cache de 32 palavras com os mesmos tamanhos de bloco e graus de associatividade (a seu critério). Qual a justificativa para os resultados obtidos na taxa de faltas para as duas configurações analisadas?

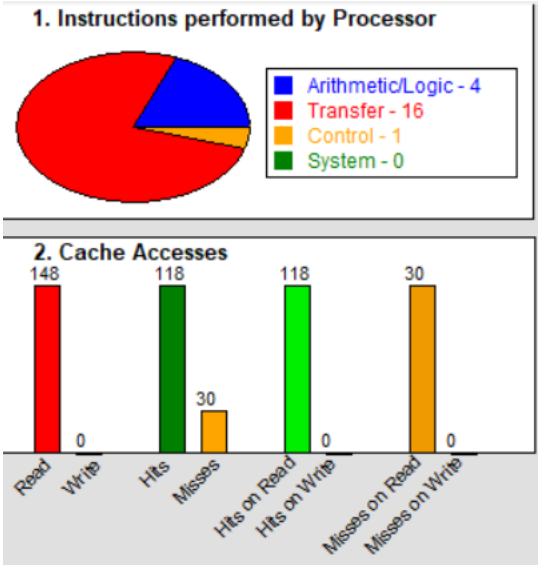
Cache de 16 palavras, 4 linhas;

148 no total, 120 de acertos e 28 erros = 18% de erro.



Cache de 32 palavras, 4 linhas;

148 no total, 118 de acertos e 30 erros = 20.3% de erro.



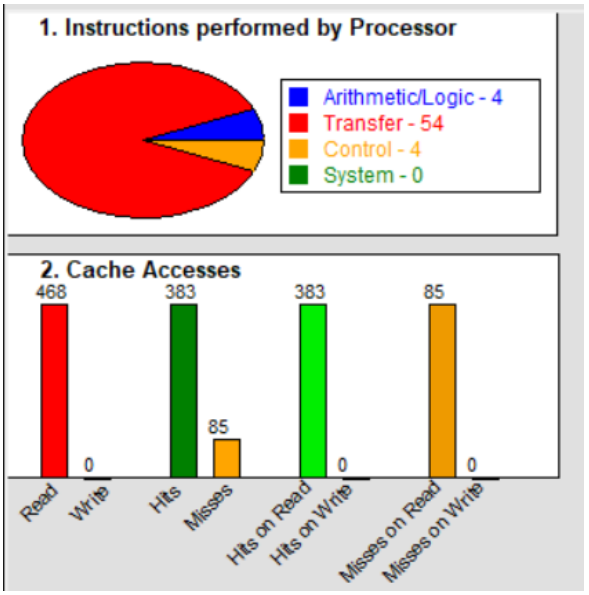
Conclusão: Mesmo com o aumento da cache a porcentagem de erros aumentou, isso pode ocorrer pois quando uma cache aumenta, vem com ela palavras que não foram solicitadas, podendo aumentar o número de faltas.

- **Caso 2:** Compare os resultados obtidos com uma cache de 16 palavras com três

tipos de associatividade: Mapeamento direto, Associativa por conjunto 2-way e Completamente associativo. Qual a justificativa para os resultados obtidos na taxa de faltas para as configurações analisadas?

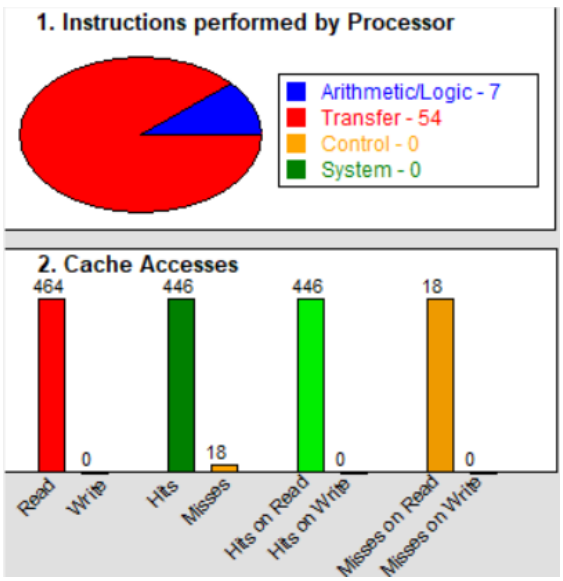
Mapeamento direto:

468 no total, 383 de acertos e 85 erros = 18% de erro.



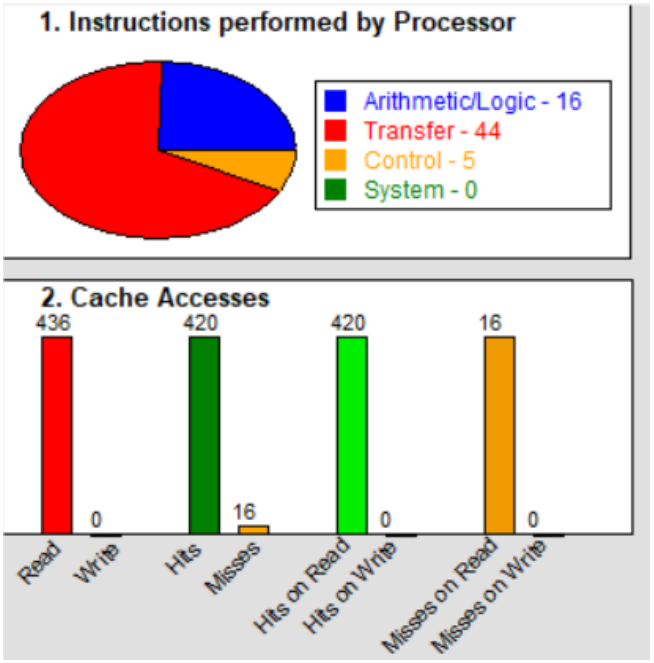
Associativa por conjunto 2-way:

464 no total, 446 de acertos e 18 erros = 3.8% de erro.



Completamente associativa:

436 no total, 420 de acertos e 16 erros = 3.6% de erro.

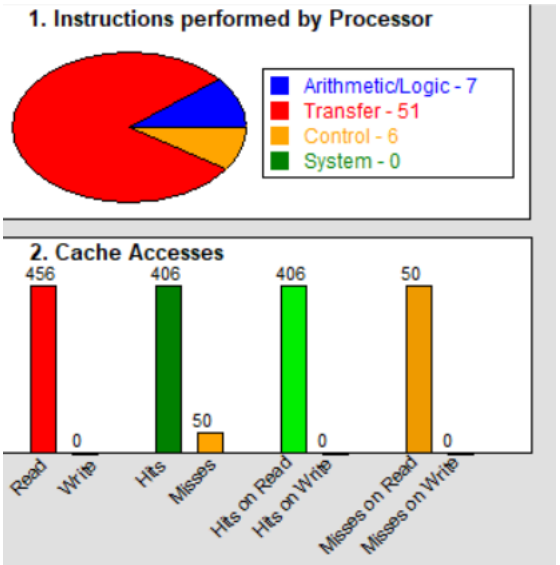


Conclusão: A mais eficiente é a completamente associativa, pois esse tipo de cache consegue acessar todos os seus dados de uma só vez e só da miss (falta) quando está completamente cheia. Porém, também é a mais cara.

- **Caso 3**: Compare uma cache de 32 palavras com blocos de 2 palavras versus uma cache de 32 palavras com blocos de 4 palavras, ambas de mapeamento direto. Qual a justificativa para os resultados obtidos na taxa de faltas para as duas configurações analisadas?

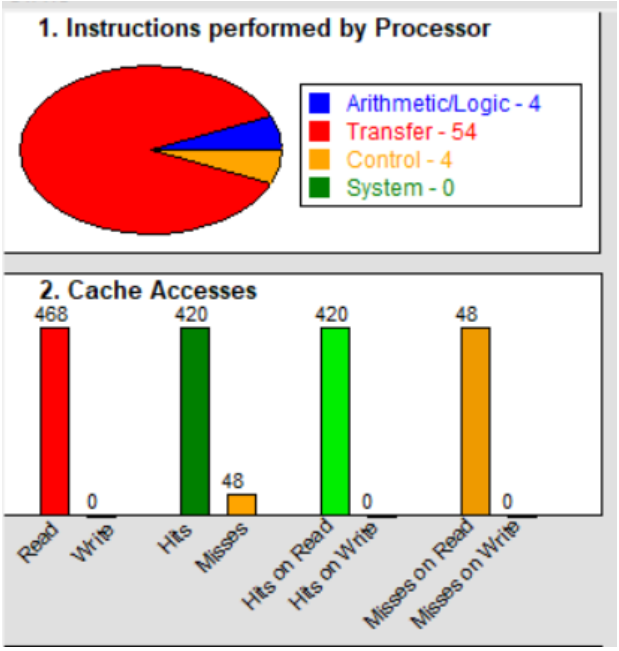
32 palavras com blocos de 2 palavras;

456 no total, 406 de acertos e 50 erros = 11% de erro.



32 palavras com blocos de 4 palavras;

468 no total, 420 de acertos e 48 erros = 10.2% de erro.



Conclusão: A mais eficiente é a cache de 32 palavras com blocos de 4 palavras, pois quando se aumenta o número de palavras por bloco a tendência é que o número de faltas diminua.