**Relatório do Laboratório 07 - Organização de Computadores**

Grupo:

- Henrique Mateus Teodoro - 23100472
- Rodrigo Schwartz - 23100471

## Exercício 1

No exercício 1 pede-se a implementação do seguinte algoritmo, que percorre uma matriz de 16 x 16 elementos, linha por linha, atribuindo aos elementos os valores de 0 a 255.

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

Memória após a execução do algoritmo do exercício 1:

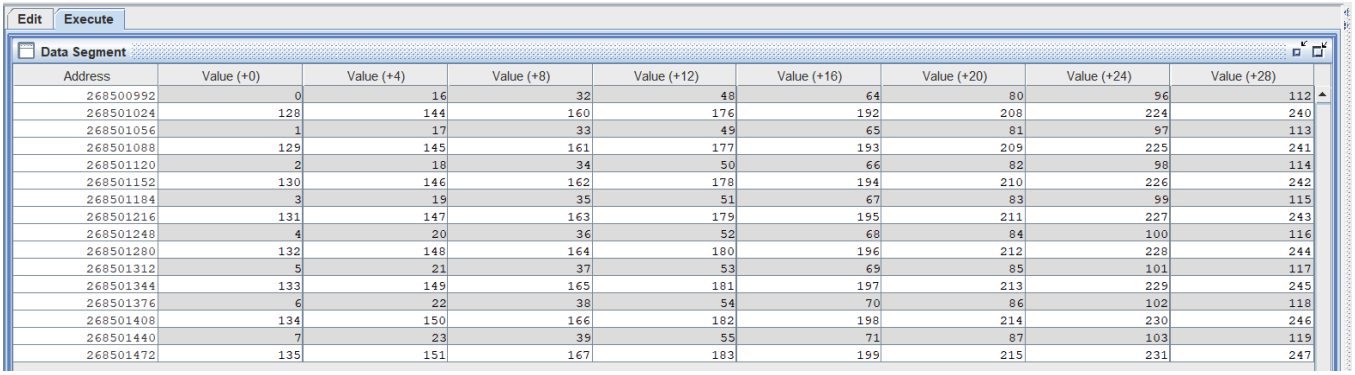
Data Segment

| Address | Value (+0) | Value (+4) | Value (+8) | Value (+12) | Value (+16) | Value (+20) | Value (+24) | Value (+28) |
|---|---|---|---|---|---|---|---|---|
| 268500992 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 268501024 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 268501056 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
| 268501088 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| 268501120 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 |
| 268501152 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |
| 268501184 | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 |
| 268501216 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 |
| 268501248 | 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 |
| 268501280 | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 |
| 268501312 | 80 | 81 | 82 | 83 | 84 | 85 | 86 | 87 |
| 268501344 | 88 | 89 | 90 | 91 | 92 | 93 | 94 | 95 |
| 268501376 | 96 | 97 | 98 | 99 | 100 | 101 | 102 | 103 |
| 268501408 | 104 | 105 | 106 | 107 | 108 | 109 | 110 | 111 |
| 268501440 | 112 | 113 | 114 | 115 | 116 | 117 | 118 | 119 |
| 268501472 | 120 | 121 | 122 | 123 | 124 | 125 | 126 | 127 |

## Exercício 2

No exercício 2 pede-se a implementação do seguinte algoritmo, que percorre uma matriz de 16 x 16 elementos, coluna por coluna, atribuindo aos elementos os valores de 0 a 255.

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



Edit | Execute

Data Segment

| Address | Value (+0) | Value (+4) | Value (+8) | Value (+12) | Value (+16) | Value (+20) | Value (+24) | Value (+28) |
|---|---|---|---|---|---|---|---|---|
| 268500992 | 0 | 16 | 32 | 48 | 64 | 80 | 96 | 112 |
| 268501024 | 128 | 144 | 160 | 176 | 192 | 208 | 224 | 240 |
| 268501056 | 1 | 17 | 33 | 49 | 65 | 81 | 97 | 113 |
| 268501088 | 129 | 145 | 161 | 177 | 193 | 209 | 225 | 241 |
| 268501120 | 2 | 18 | 34 | 50 | 66 | 82 | 98 | 114 |
| 268501152 | 130 | 146 | 162 | 178 | 194 | 210 | 226 | 242 |
| 268501184 | 3 | 19 | 35 | 51 | 67 | 83 | 99 | 115 |
| 268501216 | 131 | 147 | 163 | 179 | 195 | 211 | 227 | 243 |
| 268501248 | 4 | 20 | 36 | 52 | 68 | 84 | 100 | 116 |
| 268501280 | 132 | 148 | 164 | 180 | 196 | 212 | 228 | 244 |
| 268501312 | 5 | 21 | 37 | 53 | 69 | 85 | 101 | 117 |
| 268501344 | 133 | 149 | 165 | 181 | 197 | 213 | 229 | 245 |
| 268501376 | 6 | 22 | 38 | 54 | 70 | 86 | 102 | 118 |
| 268501408 | 134 | 150 | 166 | 182 | 198 | 214 | 230 | 246 |
| 268501440 | 7 | 23 | 39 | 55 | 71 | 87 | 103 | 119 |
| 268501472 | 135 | 151 | 167 | 183 | 199 | 215 | 231 | 247 |

## Exercício 3

Assim como pedia no enunciado da questão, organizamos os resultados da Data Cache Simulator em formato de tabela:

**a)**

| Exercício | Número de blocos | Words por bloco | Cache Hit Rate | Cache Miss Rate |
|---|---|---|---|---|
| 1 | 8 | 4 | 75% | 25% |
| 2 | 8 | 4 | 0% | 100% |

Para o caso a, utiliza-se uma cache de 8 blocos, com 4 words por bloco. Como estão sendo utilizadas matrizes 16 x 16, significa que a cache irá suportar apenas 4 valores inteiros por bloco.

Para o exercício 1, a cache irá buscar o valor 0, que não está na cache, e irá portanto contabilizar um miss, e assim, irá buscar o bloco que contenha o valor 0. Como o algoritmo 1 organiza os valores na memória de modo que os valores se deem de forma contínua, um do lado do outro na linha, o bloco que contém o valor 0 buscado trará também os valores 1, 2, 3 (bloco buscado: [0, 1, 2, 3]). Com isso, serão buscados então os valores seguintes, ou seja, 1, 2 e 3. Como estes valores já foram colocados na cache devido a busca anterior, haverão 3 acertos (hits) consecutivos. Depois disso, será então buscado o valor 4, que não está na cache, acarretando novamente em um miss e portanto, será necessário trazer todo o bloco que contém o valor 4 ([4, 5, 6, 7]). Percebe-se assim que haverá então mais 3 acertos consecutivos, na busca dos valores 5, 6 e 7. Com isso, nota-se a proporção de 1 erro (miss) para cada 3 acertos (hit), que será repetida até o término do

programa. Portanto, chega-se então no resultado obtido, com 75% de Cache Hit Rate, e 25% de Cache Miss Rate.

No entanto, para o exercício 2, como a forma de se organizar os valores não se dão de forma sequencial em linha, e sim de forma sequencial em coluna, o não aproveitamento da localidade espacial prejudicará totalmente a Cache Hit Rate, favorecendo completamente a Cache Miss Rate.

Ao buscar o valor 0 no exercício 2, será contabilizado um miss, visto que o valor 0 não está na cache. Será então buscado o bloco que contenha o valor 0, que contém juntamente os valores 16, 32, 48 (bloco buscado: [0, 16, 32, 48]). Após isso, será buscado então o valor 1, que também não estará na cache, contabilizando mais um miss, e com isso, será buscado o bloco que contém o valor 1 (bloco buscado: [1, 17, 33, 49]). Com isso, como as buscas são realizadas de forma sequencial, mas a organização na memória se dá de forma não sequencial (os números nos blocos não estão ordenados de forma crescente), isso implica que a cada busca por um valor na cache resultará em um miss. Assim, tem-se um Cache Hit Rate de 0%, e consequentemente, um Cache Miss Rate de 100%. O que causará um grande impacto na velocidade de execução do programa, visto que será necessário acessar a memória principal a cada valor.

**b)**

| Exercício | Número de blocos | Words por bloco | Cache Hit Rate | Cache Miss Rate |
|---|---|---|---|---|
| 1 | 8 | 8 | 88% | 12% |
| 2 | 8 | 8 | 0% | 100% |

Para o caso b, utiliza-se uma cache de 8 blocos, com 8 words por bloco. Como estão sendo utilizadas matrizes 16 x 16, significa que a cache irá suportar 8 valores inteiros por bloco.

Para o exercício 1, será buscado o valor 0 na cache, que não está lá, contabilizando assim um miss, e com isso, será buscado o bloco que contenha o valor 0. O bloco buscado será [0, 1, 2, 3, 4, 5, 6, 7]. Após isso, serão buscados os valores 1, 2, 3, 4, 5, 6 e 7, que já estão na cache, visto que foram carregados para ela através da busca do valor 0, contabilizando assim 7 acertos seguidos. O mesmo irá ocorrer na busca do valor 8, que não está na cache (ocorrerá um miss), mas irá colocar na cache os próximos 7 valores a serem buscados (9, 10, 11, 12, 13, 14, 15), contabilizando assim 7 acertos seguidos. Com isso, percebe-se um padrão de 1 miss a cada 7 hits.

Assim, como a cada 8 buscas ocorrem 7 acertos (hits), tem-se um Cache Hit Rate de 87,5% (⅞) e um Cache Miss Rate de 12,5% (⅛). É importante salientar que na ferramenta Data Cache Simulator do MARS, os valores em porcentagem se dão de modo discreto, e portanto, o Cache Hit Rate mostrado será de 88%.

Ao buscar o valor 0 no exercício 2, será contabilizado um miss, visto que o valor 0 não está na cache. Será então buscado o bloco [0, 16, 32, 48, 64, 80, 96, 112]. Após isso, será buscado então o valor 1, que também não estará na cache, contabilizando mais um miss, e com isso, será buscado o bloco que contém o valor 1 (bloco buscado: [1, 17, 33, 49, 65, 81, 97, 113]). Esse padrão será repetido várias vezes, implicando que a cada busca por um valor na cache resultará em um miss. Assim, tem-se um Cache Hit Rate de 0%, e consequentemente, um Cache Miss Rate de 100%.

**c)**

| Exercício | Número de blocos | Words por bloco | Cache Hit Rate | Cache Miss Rate |
|---|---|---|---|---|
| 1 | 16 | 8 | 88% | 12% |
| 2 | 16 | 8 | 0% | 100% |

Para o caso c, utiliza-se uma cache de 16 blocos, com 8 words por bloco. Como estão sendo utilizadas matrizes 16 x 16, significa que a cache irá suportar 8 valores inteiros por bloco. Com isso, percebe-se uma semelhança com o caso b, que utiliza uma cache de 8 blocos, também com 8 words por bloco. Essa semelhança se dá pelo fato de as taxas de acerto se manterem constantes, visto que o número de words por bloco segue constante. A cache suportará 8 novos blocos, mas isso não irá afetar a proporção de hit rate e miss rate ao se buscar um valor na cache.

**d)**

| Exercício | Número de blocos | Words por bloco | Cache Hit Rate | Cache Miss Rate |
|---|---|---|---|---|
| 1 | 16 | 16 | 94% | 6% |
| 2 | 16 | 16 | 94% | 6% |

Para o caso d, utiliza-se uma cache de 16 blocos, onde cada bloco armazena 16 words, ou seja, 16 valores inteiros.

Para o exercício 1, ao se buscar o valor 0, ele não estará na cache. Isso contabilizará um miss, e fará com que o bloco que contenha o valor 0 seja buscado (bloco buscado: [0, 1, 2, …, 15]). Após isso, serão buscados os valores 1, 2, …, 15. Dessa forma, serão contabilizados 15 hits seguidos, visto que agora esses valores já estão presentes na cache. Quando o valor 16 for buscado, não será encontrado na cache, e portanto, o bloco que contenha o valor 16 será buscado (bloco buscado:

[16, 17, …, 31]). Após isso, serão buscados os valores 17, 18, …, 31. Assim, serão contabilizados 15 hits seguidos, novamente. Com isso, é possível perceber um padrão, onde serão contabilizados 1 miss a cada 15 hits. Portanto, conclui-se que como são feitas 16 buscas por bloco, haverá um Cache Hit Rate de 93,75% (15 hits / 16 buscas) e um Cache Miss Rate de 6,25% (1 miss / 16 buscas).

Para o exercício 2, ao se buscar o valor 0, ele não estará na cache. Assim, será contabilizado um miss, e o bloco que contenha o valor 0 será buscado (bloco buscado: [0, 16, 32, 48, …, 240]). Os próximos valores a serem buscados serão 1, 2, …, 15. Todos esses valores não estarão na cache, ou seja, serão contabilizados mais 15 misses seguidos. No entanto, cada um desses valores buscados e não encontrados na cache irão fazer com que o seu respectivo bloco seja colocado na cache. Dessa forma, ao término da busca dos 16 valores, toda a matriz estará dentro da cache (todos os blocos foram colocados na cache). Assim, a busca de qualquer outro elemento será contabilizada como um hit, ou seja, ocorrerão 250 hits seguidos. Com isso, conclui-se que a Cache Hit Rate será de 93,75% (250 hits / 256 buscas) e a Cache Miss Rate será de 6,25% (16 misses / 256 buscas).

É importante salientar que na ferramenta Data Cache Simulator do MARS, os valores em porcentagem se dão de modo discreto, e portanto, o Cache Hit Rate de ambos os exercícios mostrados é de 94%.

**e)**
**1 miss para cada 31 h**

| Exercício | Número de blocos | Words por bloco | Cache Hit Rate | Cache Miss Rate |
|---|---|---|---|---|
| 1 | 16 | 32 | 97% | 3% |
| 2 | 16 | 32 | 97% | 3% |

Para o caso e, utiliza-se uma cache de 16 blocos, onde cada bloco armazena 32 words, ou seja, 32 valores inteiros.

Nota-se Para o exercício 1, ao se buscar o valor 0, ele não estará na cache. Isso contabilizará um miss, e fará com que o bloco que contenha o valor 0 seja buscado (bloco buscado: [0, 1, 2, …, 31]). Após isso, serão buscados os valores 1, 2, …, 31. Dessa forma, serão contabilizados 31 hits seguidos, visto que agora esses valores já estão presentes na cache. Quando o valor 32 for buscado, não será encontrado na cache, e portanto, o bloco que contenha o valor 32 será buscado (bloco buscado: [32, 33, …, 63]). Após isso, serão buscados os valores 33, 34, …, 31. Assim, serão contabilizados 31 hits seguidos, novamente. Com isso, é possível perceber um padrão, onde serão contabilizados 1 miss a cada 31 hits. Portanto, a cada 32 buscas, em 31 haverão hits e em 1 haverá miss. Dessa maneira, a Cache Hit Rate será de 96,875% (31/32) e a Cache Miss Rate será de 3,125% (1 / 32),

entretanto como simulador assume somente valores discretos, as porcentagem são arredondadas para 97% e 3%, respectivamente.

Para o exercício 2, ao se buscar o valor 0, ele não estará na cache. Assim, será contabilizado um miss, e o bloco que contenha o valor 0 será buscado (bloco buscado: [0, 16, 32, 48, …, 241]). O próximo valor a ser buscado será 1. Esse valor já foi trazido para a cache, portanto haverá um hit. Após isso, o valor buscado será 2, que não estará na cache, havendo um miss. Seu bloco inteiro será trazido para a cache. O próximo valor a ser buscado é o número 3, que já foi trazido anteriormente. Esse padrão será mantido durante as 16 primeiras buscas (em 8 haverão hits e em 8 haverão misses. Após isso, toda a matriz será trazida da memória para a cache (a cache nesse caso é bem grande, e comporta toda a matriz). Assim, durante as próximas buscas sempre haverão hits. No total haverá, 8 misses e 248 hits. Com isso, conclui-se que a Cache Hit Rate será de 96,875% (248/256) e a Cache Miss Rate será de 3,125% (8/256), entretanto como simulador assume somente valores discretos, as porcentagem são arredondadas para 97% e 3%, respectivamente.