

Nome: Tális Breda - 22102202

## Exercício 1

(8 blocos e 4 words por bloco)

Qual foi a taxa final de acertos do cache?

R:

**Simulate and illustrate data cache performance**

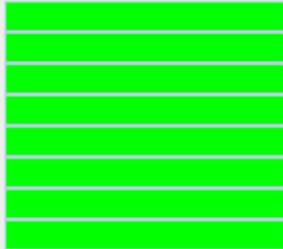
**Cache Organization**

Placement Policy:  Number of blocks:

Block Replacement Policy:  Cache block size (words):

Set size (blocks):  Cache size (bytes):

**Cache Performance**

Memory Access Count:  Cache Block Table: 

Cache Hit Count:  (block 0 at top)

Cache Miss Count:  ☐ = empty

Cache Hit Rate:  ☒ = hit

☐ = miss

**Runtime Log**

☐ Enabled

**Tool Control**

75%, pois a cada 4 acessos ocorrem 3 acertos, por conta do tamanho da word.

Quando é feita a primeira tentativa de leitura, ocorre um *miss* pois o que está sendo procurado não está presente na cache. Com o miss, um bloco com 4 words é escrito na cache. Essas 4 words incluem a que foi buscada e as próximas 3. Como o acesso, nesse caso, é sequencial, as próximas words buscadas são as que acabaram de ser escritas na cache, resultando em 3 *hits*.

Qual será a taxa de acertos se o tamanho do bloco for aumentado de 4 para 8 words?

R:

**Simulate and illustrate data cache performance**

**Cache Organization**

Placement Policy: Direct Mapping    Number of blocks: 8

Block Replacement Policy: LRU    Cache block size (words): 8

Set size (blocks): 1    Cache size (bytes): 256

**Cache Performance**

Memory Access Count: 257    Cache Block Table (block 0 at top)

Cache Hit Count: 224    ☐ = empty

Cache Miss Count: 33    ☒ = hit

Cache Hit Rate: 87%    ☐ = miss

**Runtime Log**

☐ Enabled

**Tool Control**

Disconnect from MIPS    Reset    Close

Agora, a taxa de acertos passa a ser de 87%, equivalente a 7 em 8. A lógica segue a mesma: no primeiro acesso ocorre o miss, e um bloco com a word atual e as próximas 7 é escrito na cache, resultando em *hit* nos próximos 7 acessos.

E se for diminuído de 4 words para 2 words?

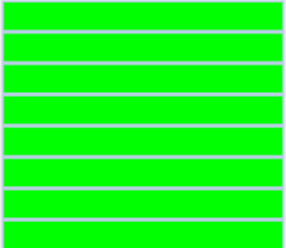
R:

### Simulate and illustrate data cache performance

#### Cache Organization

Placement Policy	Direct Mapping	Number of blocks	8
Block Replacement Policy	LRU	Cache block size (words)	2
Set size (blocks)	1	Cache size (bytes)	64

#### Cache Performance

Memory Access Count	257	Cache Block Table (block 0 at top) <input type="checkbox"/> = empty <input checked="" type="checkbox"/> = hit <input type="checkbox"/> = miss	
Cache Hit Count	128		
Cache Miss Count	129		
Cache Hit Rate	50%		

#### Runtime Log

☐ Enabled

#### Tool Control

Disconnect from MIPSResetClose

Seguindo a mesma lógica, a taxa de acertos agora passa a ser 50%, equivalente a 1 em 2. Nesse caso, quando a primeira tentativa de leitura é feita, ocorre um *miss* e um bloco de 2 words é escrito na memória, contendo apenas o próximo elemento a ser lido. Por isso, para cada *miss* ocorre apenas 1 *hit*.

## Exercício 2

(8 blocos e 4 words por bloco)

Qual foi o desempenho do cache para este programa?

R:

**Simulate and illustrate data cache performance**


**Cache Organization**

Placement Policy:  Number of blocks:

Block Replacement Policy:  Cache block size (words):

Set size (blocks):  Cache size (bytes):

**Cache Performance**

Memory Access Count:  Cache Block Table:   
(block 0 at top)

Cache Hit Count:  ☐ = empty  
Cache Miss Count:  ☐ = hit  
Cache Hit Rate:  ☐ = miss

**Runtime Log**

☐ Enabled

**Tool Control**

Para este caso, a matriz não está sendo acessada linha por linha (sequencialmente), mas está sendo acessada coluna por coluna. Dessa forma, quando é feita a primeira tentativa de leitura, ocorre um *miss* e é escrito na cache um bloco com 4 words, consistindo do endereço atual e dos próximos 3. Porém, os próximos 3 endereços consistem nos elementos da linha atual da matriz, e o acesso é feito na coluna. Então, na próxima leitura, também ocorrerá *miss*, pois o endereço buscado continuará não existindo na cache.

Dessa forma, não ocorre nenhum *hit* durante a execução do programa, e a taxa de acertos fica em 0%.

(16 blocos e 4 words por bloco)

Qual foi o desempenho do cache para este programa?

R:

**Simulate and illustrate data cache performance**

**Cache Organization**

Placement Policy:  Number of blocks:

Block Replacement Policy:  Cache block size (words):

Set size (blocks):  Cache size (bytes):

**Cache Performance**

Memory Access Count:  Cache Block Table (block 0 at top)

Cache Hit Count:  ☐ = empty

Cache Miss Count:  ☐ = hit

Cache Hit Rate:  ☐ = miss

**Runtime Log**

☐ Enabled

**Tool Control**

Nesse caso, a quantidade de blocos de memória na cache aumentou de 8 para 16. Porém, assim como no caso anterior, a cada *miss*, é escrito um bloco com 4 words na cache, consistindo dos próximos 3 endereços da mesma linha. Como o acesso é feito em colunas, esses endereços não estarão mais presentes na cache no momento em que forem acessados, pois serão substituídos antes que isso aconteça. Por isso, a taxa de acerto continua em 0%.

Obs: É possível aumentar a taxa de acertos aumentando apenas a quantidade de blocos de memória na cache, porém seriam necessários muito mais blocos. No caso da matriz 16x16, uma cache com 64 blocos de 4 words tem uma taxa de acerto de 75%, porém serão necessários ainda mais blocos para matrizes maiores.

Isso ocorre pois, mesmo que as primeiras leituras sejam *miss*, os próximos endereços são escritos na cache, e ela é grande o suficiente para que alguns deles não sejam substituídos, e permaneçam armazenados até que sejam lidos.

(16 blocos e 16 words por bloco)

Qual foi o desempenho do cache para este programa?

R:

**Simulate and illustrate data cache performance**


**Cache Organization**

Placement Policy:  Number of blocks:

Block Replacement Policy:  Cache block size (words):

Set size (blocks):  Cache size (bytes):

**Cache Performance**

Memory Access Count:  Cache Block Table: 

Cache Hit Count:  (block 0 at top)

Cache Miss Count:  ☐ = empty

Cache Hit Rate:  ☐ = hit

☐ = miss

**Runtime Log**

☐ Enabled

**Tool Control**

A lógica para esse caso é a que segue:

- Primeira leitura: *miss*, pois ainda não tem nada armazenado na cache. Um bloco com 16 words é escrito nela, consistindo em todos os elementos da linha atual da matriz.
- Próxima leitura: *miss*, pois como o acesso ainda está sendo feito coluna após coluna, e na cache está armazenada apenas a linha anterior, o endereço buscado não existirá na cache. Novamente, é escrito na cache um bloco com 16 words, equivalente a todos os elementos da linha atual.
- O passo anterior se repete até que o programa chegue ao fim da primeira coluna. A partir daí, todas as linhas da matriz foram escritas na cache. Isso significa que a matriz inteira agora está armazenada na cache. Logo, todos os próximos acessos resultarão em *hit*, pois a cache tem tamanho suficiente para que não ocorra substituição em nenhum momento do programa.