

**Nome: Tális Breda - 22102202**

### **Exercício 1**

Implemente o algoritmo abaixo no MARS que realiza a soma de uma matriz com outra transposta. O tamanho das matrizes (MAX) deve ser parametrizável.

Observação: para a construção da matriz, foi utilizado o primeiro algoritmo do laboratório 3 da disciplina. Para os exemplos desse relatório, será utilizado o tamanho padrão 8x8 para a matriz

### **Resultado esperado:**

Matriz A:

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	57	58	59	60	61	62	63

Matriz B:

64	65	66	67	68	69	70	71
72	73	74	75	76	77	78	79
80	81	82	83	84	85	86	87
88	89	90	91	92	93	94	95
96	97	98	99	100	101	102	103
104	105	106	107	108	109	110	111
112	113	114	115	116	117	118	119
120	121	122	123	124	125	126	127

Matriz resultado:

64	73	82	91	100	109	118	127
73	82	91	100	109	118	127	136
82	91	100	109	118	127	136	145
91	100	109	118	127	136	145	154
100	109	118	127	136	145	154	163
109	118	127	136	145	154	163	172
118	127	136	145	154	163	172	181
127	136	145	154	163	172	181	190

**Resultado obtido:**

Matriz A (imagem pré soma):

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

Matriz B:

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

Matriz resultado (imagem pós soma):

Address	Value (+0)	Value (+4)	Value (+8)	Value (+12)	Value (+16)	Value (+20)	Value (+24)	Value (+28)
268500992	64	73	82	91	100	109	118	127
268501024	73	82	91	100	109	118	127	136
268501056	82	91	100	109	118	127	136	145
268501088	91	100	109	118	127	136	145	154
268501120	100	109	118	127	136	145	154	163
268501152	109	118	127	136	145	154	163	172
268501184	118	127	136	145	154	163	172	181
268501216	127	136	145	154	163	172	181	190

## Exercício 2

Implemente o algoritmo abaixo no MARS que realiza a soma de uma matriz com outra transposta, porém utilizando a técnica de cache blocking. O tamanho das matrizes (MAX) e dos blocos (block\_size) devem ser parametrizáveis:

Nessa questão, apesar de o algoritmo utilizado ser diferente, o resultado esperado é o mesmo, e o obtido está a seguir:

### Resultado obtido:

Matriz A (imagem pré soma):

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

Matriz B:

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

Matriz resultado (imagem pós soma):

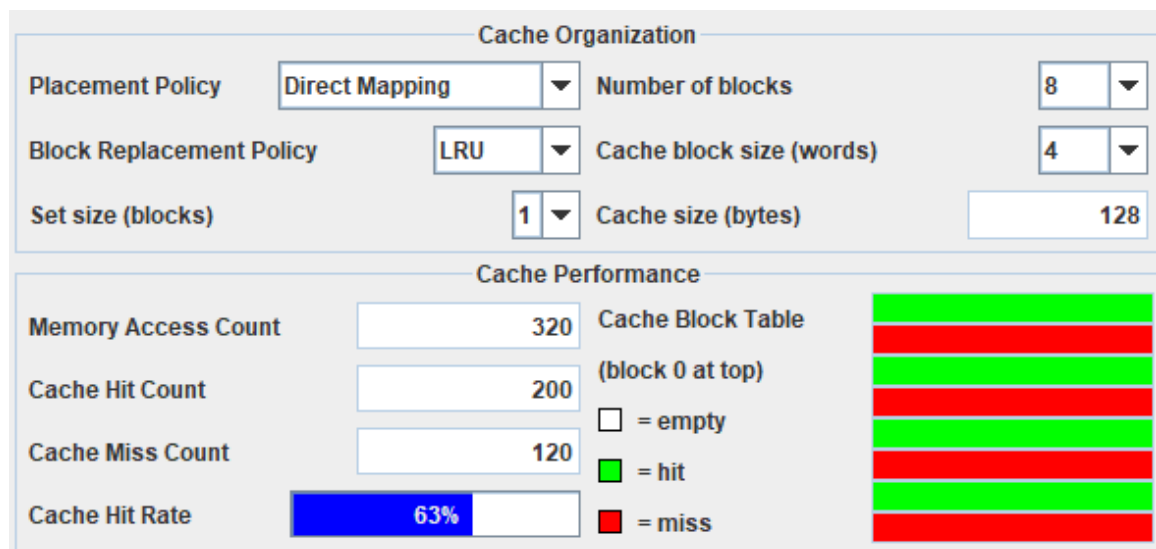
Address	Value (+0)	Value (+4)	Value (+8)	Value (+12)	Value (+16)	Value (+20)	Value (+24)	Value (+28)
268500992	64	73	82	91	100	109	118	127
268501024	73	82	91	100	109	118	127	136
268501056	82	91	100	109	118	127	136	145
268501088	91	100	109	118	127	136	145	154
268501120	100	109	118	127	136	145	154	163
268501152	109	118	127	136	145	154	163	172
268501184	118	127	136	145	154	163	172	181
268501216	127	136	145	154	163	172	181	190

### Exercício 3

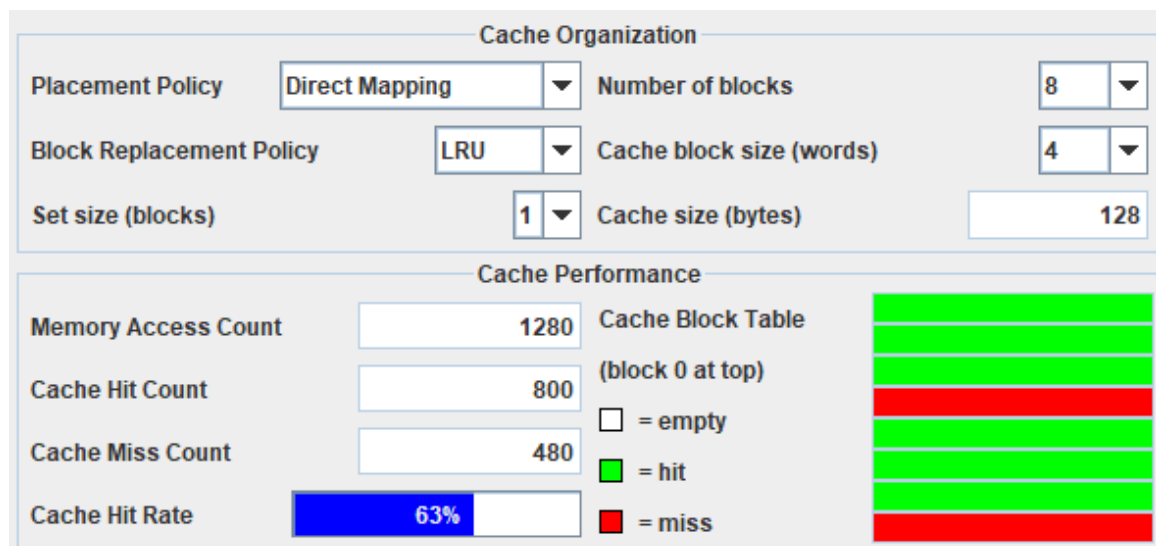
Execute os dois programas anteriores utilizando a ferramenta Data Cache Simulator, fazendo combinações de tamanhos de matrizes e tamanhos de blocos de cache blocking. Apresente e explique os resultados obtidos.

#### Algoritmo 1:

Matriz 8x8, 8 blocos de 4 words



Matriz 16x16, 8 blocos de 4 words



Matriz 8x8, 8 blocos de 8 words

### 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: 320 Cache Block Table (block 0 at top)  
Cache Hit Count: 273 ☐ = empty  
Cache Miss Count: 47 ☒ = hit  
Cache Hit Rate: 85% ☒ = miss

**Runtime Log**  
☐ Enabled

**Tool Control**  
Disconnect from MIPS Reset Close

Matriz 16x16, 8 blocos de 8 words

### 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: 1280 Cache Block Table (block 0 at top)  
Cache Hit Count: 896 ☐ = empty  
Cache Miss Count: 384 ☒ = hit  
Cache Hit Rate: 70% ☒ = miss

Matriz 32x32, 8 blocos de 8 words

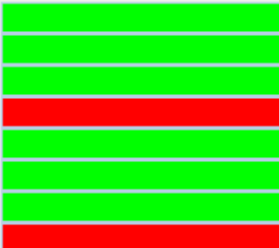
Cache Organization			
Placement Policy	Direct Mapping	Number of blocks	8
Block Replacement Policy	LRU	Cache block size (words)	8
Set size (blocks)	1	Cache size (bytes)	256

Cache Performance	
Memory Access Count	5120
Cache Hit Count	3584
Cache Miss Count	1536
Cache Hit Rate	70%

Cache Block Table
(block 0 at top)
<input type="checkbox"/> = empty
<input checked="" type="checkbox"/> = hit
<input type="checkbox"/> = miss



Matriz 8x8, 16 blocos de 8 words


Simulate and illustrate data cache performance			
Cache Organization			
Placement Policy	Direct Mapping	Number of blocks	16
Block Replacement Policy	LRU	Cache block size (words)	8
Set size (blocks)	1	Cache size (bytes)	512

Cache Performance	
Memory Access Count	320
Cache Hit Count	304
Cache Miss Count	16
Cache Hit Rate	95%

Cache Block Table
(block 0 at top)
<input type="checkbox"/> = empty
<input checked="" type="checkbox"/> = hit
<input type="checkbox"/> = miss

Runtime Log	
<input type="checkbox"/> Enabled	

Tool Control		
Disconnect from MIPS	Reset	Close

Matriz 8x8, 8 blocos de 16 words

**Simulate and illustrate data cache performance**

**Cache Organization**

Placement Policy: **Direct Mapping** Number of blocks: **8**

Block Replacement Policy: **LRU** Cache block size (words): **16**

Set size (blocks): **1** Cache size (bytes): **512**

**Cache Performance**

Memory Access Count: **320** Cache Block Table (block 0 at top)

Cache Hit Count: **312**

Cache Miss Count: **8**

Cache Hit Rate: **98%**

☐ = empty ☒ = hit ☐ = miss

**Runtime Log**

☐ Enabled

**Tool Control**

**Disconnect from MIPS** **Reset** **Close**

Matriz 4x4, 8 blocos de 4 words

**Cache Organization**

Placement Policy: **Direct Mapping** Number of blocks: **8**

Block Replacement Policy: **LRU** Cache block size (words): **4**

Set size (blocks): **1** Cache size (bytes): **128**

**Cache Performance**

Memory Access Count: **80** Cache Block Table (block 0 at top)

Cache Hit Count: **72**

Cache Miss Count: **8**

Cache Hit Rate: **90%**

☐ = empty ☒ = hit ☐ = miss

Nota-se, com essas e algumas outras combinações, que a taxa de hit do cache fica mais alta quando o tamanho da matriz é um pouco menor ou igual à quantidade de words, desde que o número de blocos seja semelhantemente alto. Se o tamanho da matriz aumenta muito e o tamanho da cache permanece o mesmo, a taxa de hit tende a se manter igual. E, é claro, se o tamanho da cache aumenta e a matriz fica suficientemente menor, a taxa de hit fica altíssima.

## Algoritmo 2:

Matriz 8x8, block\_size 4, 8 blocos de 4 words

Cache Organization			
Placement Policy	Direct Mapping	Number of blocks	8
Block Replacement Policy	LRU	Cache block size (words)	4
Set size (blocks)	1	Cache size (bytes)	128

Cache Performance			
Memory Access Count	320	Cache Block Table	
Cache Hit Count	236	(block 0 at top)	
Cache Miss Count	84	<input type="checkbox"/> = empty	
Cache Hit Rate	74%	<input checked="" type="checkbox"/> = hit	
		<input type="checkbox"/> = miss	

Matriz 8x8, block\_size 2, 8 blocos de 4 words

Cache Organization			
Placement Policy	Direct Mapping	Number of blocks	8
Block Replacement Policy	LRU	Cache block size (words)	4
Set size (blocks)	1	Cache size (bytes)	128

Cache Performance			
Memory Access Count	320	Cache Block Table	
Cache Hit Count	220	(block 0 at top)	
Cache Miss Count	100	<input type="checkbox"/> = empty	
Cache Hit Rate	69%	<input checked="" type="checkbox"/> = hit	
		<input type="checkbox"/> = miss	

Matriz 16x16, block\_size 4, 8 blocos de 4 words

Cache Organization			
Placement Policy	Direct Mapping	Number of blocks	8
Block Replacement Policy	LRU	Cache block size (words)	4
Set size (blocks)	1	Cache size (bytes)	128

Cache Performance			
Memory Access Count	1280	Cache Block Table	
Cache Hit Count	800	(block 0 at top)	
Cache Miss Count	480	<input type="checkbox"/> = empty	
Cache Hit Rate	63%	<input checked="" type="checkbox"/> = hit	
		<input type="checkbox"/> = miss	



Matriz 16x16, block\_size 8, 8 blocos de 4 words

Cache Organization			
Placement Policy	Direct Mapping	Number of blocks	8
Block Replacement Policy	LRU	Cache block size (words)	4
Set size (blocks)	1	Cache size (bytes)	128

Cache Performance		
Memory Access Count	1280	Cache Block Table (block 0 at top) <input type="checkbox"/> = empty <input checked="" type="checkbox"/> = hit <input checked="" type="checkbox"/> = miss
Cache Hit Count	800	
Cache Miss Count	480	
Cache Hit Rate	63%	

Matriz 16x16, block\_size 2, 8 blocos de 4 words

Cache Organization			
Placement Policy	Direct Mapping	Number of blocks	8
Block Replacement Policy	LRU	Cache block size (words)	4
Set size (blocks)	1	Cache size (bytes)	128

Cache Performance		
Memory Access Count	1280	Cache Block Table (block 0 at top) <input type="checkbox"/> = empty <input checked="" type="checkbox"/> = hit <input checked="" type="checkbox"/> = miss
Cache Hit Count	880	
Cache Miss Count	400	
Cache Hit Rate	69%	

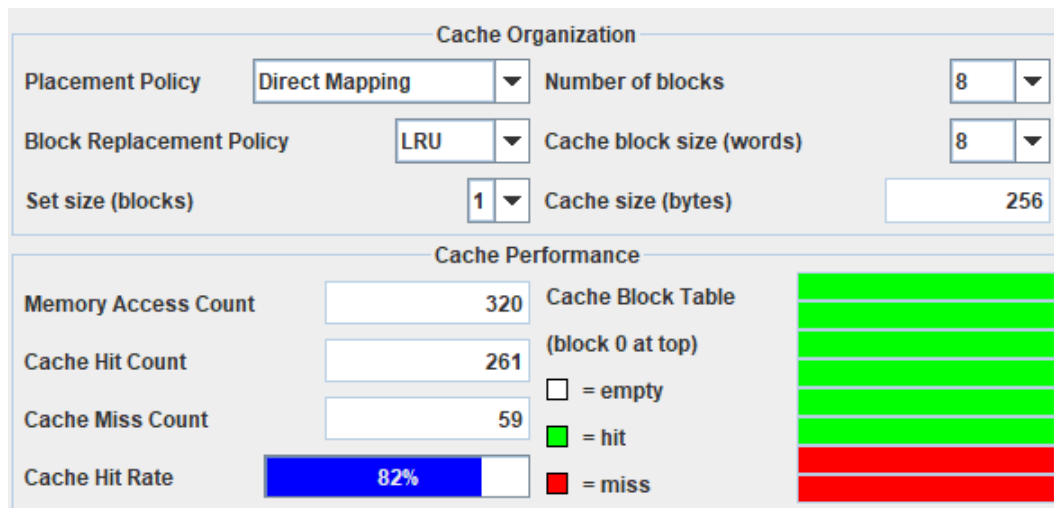
Matriz 8x8, block\_size 4, 8 blocos de 8 words

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

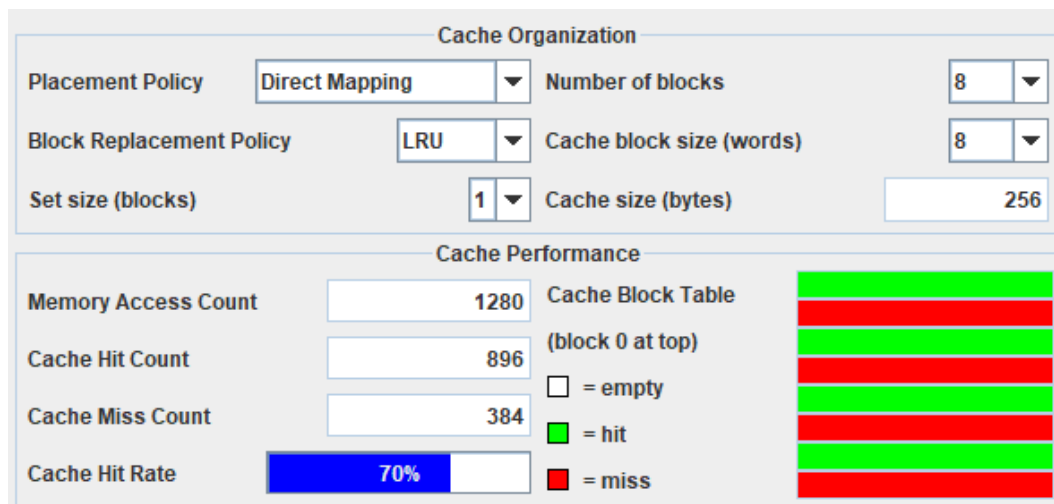
  

Cache Performance		
Memory Access Count	320	Cache Block Table (block 0 at top) <input type="checkbox"/> = empty <input checked="" type="checkbox"/> = hit <input checked="" type="checkbox"/> = miss
Cache Hit Count	261	
Cache Miss Count	59	
Cache Hit Rate	82%	

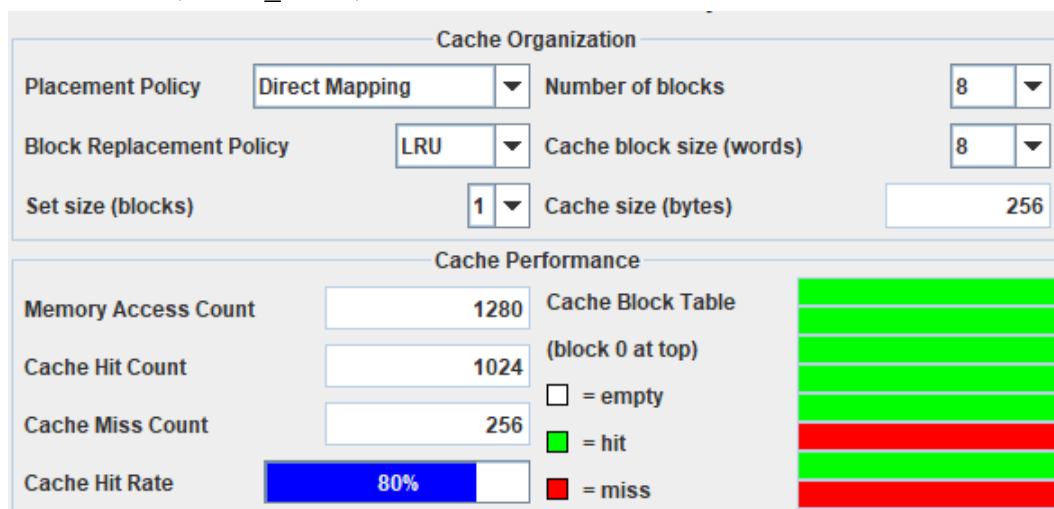
Matriz 8x8, block\_size 2, 8 blocos de 8 words



Matriz 16x16, block\_size 8, 8 blocos de 8 words



Matriz 16x16, block\_size 4, 8 blocos de 8 words



Matriz 16x16, block\_size 2, 8 blocos de 8 words

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	1280
Cache Hit Count	968
Cache Miss Count	312
Cache Hit Rate	76%

Cache Block Table	
(block 0 at top)	
<input type="checkbox"/> = empty	
<input checked="" type="checkbox"/> = hit	
<input type="checkbox"/> = miss	

Matriz 32x32, block\_size 16, 8 blocos de 8 words

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	5120
Cache Hit Count	3584
Cache Miss Count	1536
Cache Hit Rate	70%

Cache Block Table	
(block 0 at top)	
<input type="checkbox"/> = empty	
<input checked="" type="checkbox"/> = hit	
<input type="checkbox"/> = miss	

Matriz 32x32, block\_size 8, 8 blocos de 8 words

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	5120
Cache Hit Count	3584
Cache Miss Count	1536
Cache Hit Rate	70%

Cache Block Table	
(block 0 at top)	
<input type="checkbox"/> = empty	
<input checked="" type="checkbox"/> = hit	
<input type="checkbox"/> = miss	

Matriz 32x32, block\_size 4, 8 blocos de 8 words

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	5120	Cache Block Table	
Cache Hit Count	3456	(block 0 at top)	
Cache Miss Count	1664	<input type="checkbox"/> = empty	
Cache Hit Rate	68%	<input checked="" type="checkbox"/> = hit	
		<input type="checkbox"/> = miss	

Matriz 8x8, block\_size 4, 16 blocos de 8 words

Cache Organization			
Placement Policy	Direct Mapping	Number of blocks	16
Block Replacement Policy	LRU	Cache block size (words)	8
Set size (blocks)	1	Cache size (bytes)	512

Cache Performance			
Memory Access Count	1280	Cache Block Table	
Cache Hit Count	1068	(block 0 at top)	
Cache Miss Count	212	<input type="checkbox"/> = empty	
Cache Hit Rate	83%	<input checked="" type="checkbox"/> = hit	
		<input type="checkbox"/> = miss	

Matriz 8x8, block\_size 4, 8 blocos de 16 words

Cache Organization			
Placement Policy	Direct Mapping	Number of blocks	8
Block Replacement Policy	LRU	Cache block size (words)	16
Set size (blocks)	1	Cache size (bytes)	512


Cache Performance			
Memory Access Count	320	Cache Block Table	
Cache Hit Count	312	(block 0 at top)	
Cache Miss Count	8	<input type="checkbox"/> = empty	
Cache Hit Rate	98%	<input checked="" type="checkbox"/> = hit	
		<input type="checkbox"/> = miss	

Matriz 4x4, block\_size 2, 8 blocos de 4 words

Cache Organization			
Placement Policy	Direct Mapping	Number of blocks	8
Block Replacement Policy	LRU	Cache block size (words)	4
Set size (blocks)	1	Cache size (bytes)	128

Cache Performance		
Memory Access Count	80	Cache Block Table (block 0 at top)  <input type="checkbox"/> = empty <input checked="" type="checkbox"/> = hit <input type="checkbox"/> = miss
Cache Hit Count	72	
Cache Miss Count	8	
Cache Hit Rate	90%	



Apesar de que em diversos casos o desempenho do cache é o mesmo ou parecido, ele aparenta ser melhor quando o block\_size tem a metade do tamanho da matriz, por exemplo tamanho 4 em uma matriz 8x8. Nos casos onde a matriz é suficientemente maior que a cache, o desempenho permanece inalterado.