UNIVERSIDADE FEDERAL DE SANTA CATARINA
CAMPUS FLORIANÓPOLIS
INE-DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

INE5411 - ORGANIZAÇÃO DE COMPUTADORES I

PEDRO TAGLIALENHA (22203674)
VITOR PRAXEDES CALEGARI (22200379)

RELATÓRIO LABORATÓRIO 5

FLORIANÓPOLIS
2023

1 INTRODUÇÃO

Este relatório tem como objetivo apresentar a análise e implementação de dois programas desenvolvidos como parte do laboratório 5 da disciplina de Organização de Computadores. Os programas em questão têm relevância na avaliação do desempenho em memória cache e consistem em tarefas que percorrem uma matriz de 16 por 16 elementos, atribuindo valores de 0 a 255 de maneiras distintas.

O primeiro programa aborda a tarefa de preencher a matriz linha a linha, enquanto o segundo programa foca na atribuição dos valores coluna a coluna. Ambos os programas foram desenvolvidos no ambiente MARS.

O relatório apresentará os códigos-fonte dos programas, suas análises e conclusões.

2 CÓDIGOS DESENVOLVIDOS

Figura 1 - Código da questão 1

```
.data
    row: .word 0
    col: .word 0
    value: .word 0
    mat: .word

.text

    la     $s0, row
    la     $s1, col
    la    $s2, mat

    lw    $s3, 0($s0)
    lw    $s4, 0($s1)

    addi    $s2, $s2,  20

    la    $s6, value
    lw    $s7, 0($s6)


linha:
    beq    $s3, 16, exit

    add    $s4, $zero, $zero
    sw    $s4, 0($s1)

    jal     coluna

    addi    $s3, $s3, 1
    sw    $s3, 0($s0)

    j     linha
exit:
    li    $v0, 10
    syscall


coluna:
    bne    $s4, 16, continua
    jr    $ra

continua:
    mul    $t0, $s3, 64
    mul    $t1, $s4, 4
    add    $t0, $t0, $t1
    add    $t0, $t0, $s2

    sw    $s7, 0($s6)

    sw    $s7, 0($t0)
    addi    $s7, $s7, 1

    addi    $s4, $s4, 1
    sw    $s4, 0($s1)

    j     coluna
```

Fonte: Elaborado por autores (2023)

Figura 2 - Código da questão 2

```
.data
    row: .word 0
    col: .word 0
    value: .word 0
    mat: .word

.text

    la     $s0, row
    la     $s1, col
    la    $s2, mat

    lw    $s3, 0($s0)
    lw    $s4, 0($s1)

    addi    $s2, $s2,  20

    la    $s6, value
    lw    $s7, 0($s6)


linha:
    beq    $s3, 16, exit

    add    $s4, $zero, $zero
    sw    $s4, 0($s1)

    jal     coluna

    addi    $s3, $s3, 1
    sw    $s3, 0($s0)

    j     linha
exit:
    li    $v0, 10
    syscall


coluna:
    bne    $s4, 16, continua
    jr    $ra

continua:
    mul    $t0, $s3, 4
    mul    $t1, $s4, 64
    add    $t0, $t0, $t1
    add    $t0, $t0, $s2

    sw    $s7, 0($s6)

    sw    $s7, 0($t0)
    addi    $s7, $s7, 1

    addi    $s4, $s4, 1
    sw    $s4, 0($s1)

    j     coluna
```

Fonte: Elaborado por autores (2023)

2.1 ANÁLISE DO CÓDIGO 1

O código desenvolvido pode ser quebrado nos seguintes principais pontos:

1. **Alternância entre Linha e Coluna:**
    - O programa possui dois loops principais, "linha" e "coluna," que alternam entre o preenchimento linha a linha e coluna a coluna da matriz.
2. **Multiplicações para Cálculo de Deslocamento:**
    - Dentro das funções "coluna" e "continua," são realizadas operações de multiplicação para calcular o deslocamento na matriz. Isso é necessário porque os elementos da matriz não estão armazenados em uma estrutura bidimensional, mas sim em uma área de memória contígua.
3. **Função "linha":**
    - A função "linha" é responsável por preencher os elementos da matriz em uma linha. Ela começa com a coordenada da coluna ($s4) definida como zero e, em seguida, chama a função "coluna" para preencher a linha atual. Após isso, incrementa a coordenada da linha e repete o processo até preencher todas as linhas.
4. **Função "coluna":**
    - A função "coluna" preenche os elementos da matriz em uma coluna. Ela começa com a coordenada da coluna ($s4) definida como zero e, em seguida, entra em um loop. Dentro desse loop, o código calcula o deslocamento no endereço da matriz com base nas coordenadas da linha e coluna, realiza a atribuição do valor (0 a 255) e, em seguida, incrementa a coordenada da coluna e o valor a ser atribuído ($s7). O loop continua até que a coordenada da coluna seja igual a 16.
5. **Função "continua":**
    - A função "continua" é chamada a partir de "coluna" e lida com a tarefa de calcular o deslocamento no endereço da matriz. Isso envolve multiplicar as coordenadas da linha e coluna pelos valores apropriados, adicionar o deslocamento ao endereço base da matriz e realizar as operações de armazenamento.

Esses loops aninhados garantem que a matriz seja preenchida linha por linha, pois o loop externo controla a iteração pelas linhas, enquanto o loop interno preenche os elementos da coluna dentro de cada linha. No final do loop externo, todas as linhas da matriz terão sido preenchidas com os valores de 0 a 255.

2.2 ANÁLISE DO CÓDIGO 2

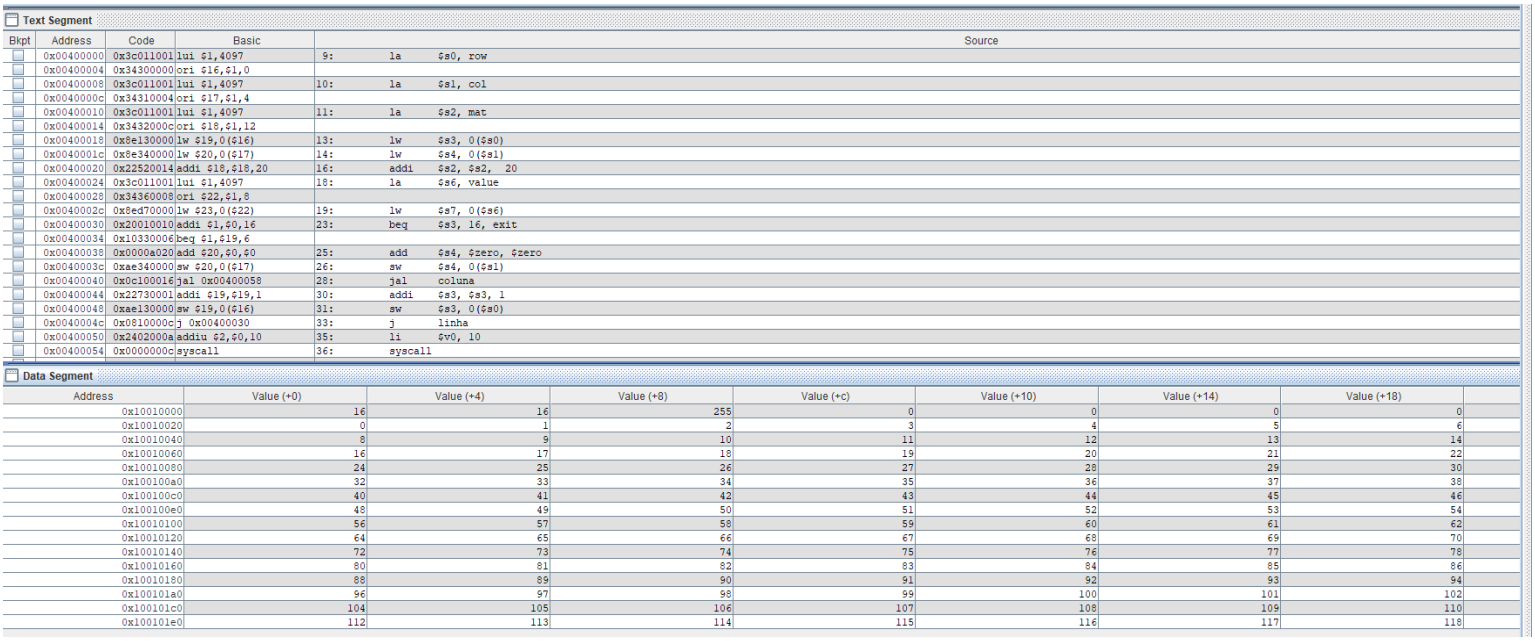
O Código 2 é semelhante ao Código 1, pois ambos visam preencher uma matriz 16x16 com valores de 0 a 255 alternando entre preenchimento por linha e coluna. No entanto, existe algumas diferenças:

1. **Função "coluna" e "continua":**
    - No Código 2, as funções "coluna" e "continua" foram modificadas em relação ao Código 1.
    - A função "continua" calcula o deslocamento de maneira diferente: multiplica $s3 (linha) por 4 e $s4 (coluna) por 64 para obter o deslocamento apropriado no endereço da matriz.
    - A ordem das multiplicações e a estratégia de cálculo são invertidas em relação ao Código 1.
2. **Diferenças na Ordem de Preenchimento:**
    - Enquanto o Código 1 preenche a matriz linha a linha, o Código 2 preenche coluna a coluna. Isso significa que, no Código 2, a coordenada da coluna é reiniciada a cada iteração do loop "linha," enquanto a coordenada da linha é reiniciada no Código 1 para cada iteração do loop "coluna."
3. **Multiplicação para Cálculo do Deslocamento:**
    - O uso da multiplicação no Código 2 para calcular o deslocamento é diferente da estratégia do Código 1. O Código 2 multiplica a coordenada da linha por 4 e a coordenada da coluna por 64 para calcular o deslocamento, enquanto o Código 1 utiliza uma abordagem inversa.

O Código 2 é semelhante ao Código 1 em termos de objetivo, mas há diferenças nas estratégias de cálculo de deslocamento e na ordem de preenchimento da matriz.

3 SAÍDAS DOS CÓDIGOS NO MARS

Figura 3 - Saída do código 1



Text Segment

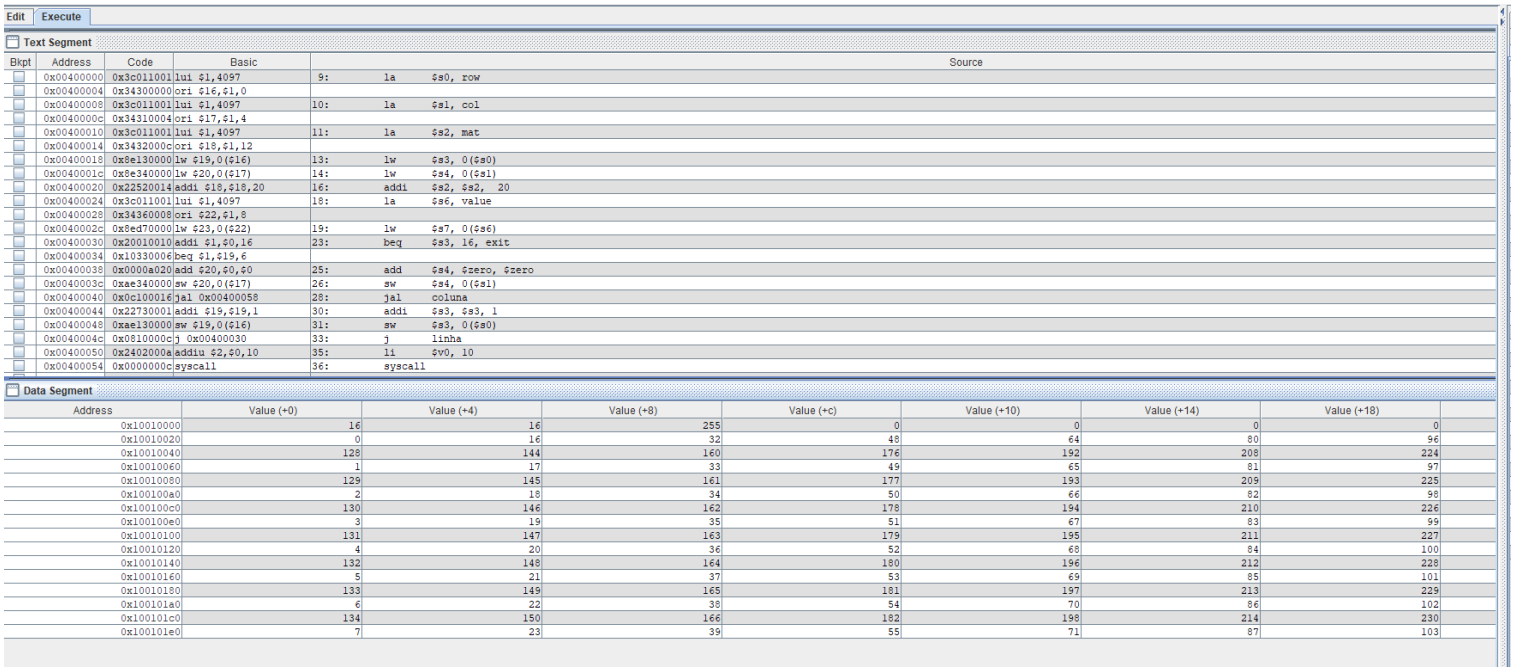
Bkpt	Address	Code	Basic			Source
	0x00400000	0x3c011001	lui $1,4097	9:	la	$s0, row
	0x00400004	0x34300000	ori $16,$1,0			
	0x00400008	0x3c011001	lui $1,4097	10:	la	$s1, col
	0x0040000c	0x34310004	ori $17,$1,4			
	0x00400010	0x3c011001	lui $1,4097	11:	la	$s2, mat
	0x00400014	0x3432000c	ori $18,$1,12			
	0x00400018	0x8e130000	lw $19,0($16)	13:	lw	$s3, 0($s0)
	0x0040001c	0x8e340000	lw $20,0($17)	14:	lw	$s4, 0($s1)
	0x00400020	0x22520014	addi $18,$18,20	16:	addi	$s2, $s2, 20
	0x00400024	0x3c011001	lui $1,4097	18:	la	$s6, value
	0x00400028	0x34360008	ori $22,$1,8			
	0x0040002c	0x8ed70000	lw $23,0($22)	19:	lw	$s7, 0($s6)
	0x00400030	0x20010010	addi $1,$0,16	23:	beq	$s3, 16, exit
	0x00400034	0x10330006	beq $1,$19,6			
	0x00400038	0x0000a020	add $20,$0,$0	25:	add	$s4, $zero, $zero
	0x0040003c	0xae340000	sw $20,0($17)	26:	sw	$s4, 0($s1)
	0x00400040	0x0c100016	jal 0x00400058	28:	jal	coluna
	0x00400044	0x22730001	addi $19,$19,1	30:	addi	$s3, $s3, 1
	0x00400048	0xae130000	sw $19,0($16)	31:	sw	$s3, 0($s0)
	0x0040004c	0x08100009	j 0x00400030	33:	j	linha
	0x00400050	0x2402000a	addiu $2,$0,10	35:	li	$v0, 10
	0x00400054	0x0000000c	syscall	36:	syscall	

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)
0x10010000	16	16	255	0	0	0	0
0x10010020	0	1	2	3	4	5	6
0x10010040	8	9	10	11	12	13	14
0x10010060	16	17	18	19	20	21	22
0x10010080	24	25	26	27	28	29	30
0x100100a0	32	33	34	35	36	37	38
0x100100c0	40	41	42	43	44	45	46
0x100100e0	48	49	50	51	52	53	54
0x10010100	56	57	58	59	60	61	62
0x10010120	64	65	66	67	68	69	70
0x10010140	72	73	74	75	76	77	78
0x10010160	80	81	82	83	84	85	86
0x10010180	88	89	90	91	92	93	94
0x100101a0	96	97	98	99	100	101	102
0x100101c0	104	105	106	107	108	109	110
0x100101e0	112	113	114	115	116	117	118

Fonte: Elaborado por autores (2023)

Figura 4 - Saída do código 2



Edit Execute

Text Segment

Bkpt	Address	Code	Basic			Source
	0x00400000	0x3c011001	lui $1,4097	9:	la	$s0, row
	0x00400004	0x34300000	ori $16,$1,0			
	0x00400008	0x3c011001	lui $1,4097	10:	la	$s1, col
	0x0040000c	0x34310004	ori $17,$1,4			
	0x00400010	0x3c011001	lui $1,4097	11:	la	$s2, mat
	0x00400014	0x3432000c	ori $18,$1,12			
	0x00400018	0x8e130000	lw $19,0($16)	13:	lw	$s3, 0($s0)
	0x0040001c	0x8e340000	lw $20,0($17)	14:	lw	$s4, 0($s1)
	0x00400020	0x22520014	addi $18,$18,20	16:	addi	$s2, $s2, 20
	0x00400024	0x3c011001	lui $1,4097	18:	la	$s6, value
	0x00400028	0x34360008	ori $22,$1,8			
	0x0040002c	0x8ed70000	lw $23,0($22)	19:	lw	$s7, 0($s6)
	0x00400030	0x20010010	addi $1,$0,16	23:	beq	$s3, 16, exit
	0x00400034	0x10330006	beq $1,$19,6			
	0x00400038	0x0000a020	add $20,$0,$0	25:	add	$s4, $zero, $zero
	0x0040003c	0xae340000	sw $20,0($17)	26:	sw	$s4, 0($s1)
	0x00400040	0x0c100016	jal 0x00400058	28:	jal	coluna
	0x00400044	0x22730001	addi $19,$19,1	30:	addi	$s3, $s3, 1
	0x00400048	0xae130000	sw $19,0($16)	31:	sw	$s3, 0($s0)
	0x0040004c	0x08100009	j 0x00400030	33:	j	linha
	0x00400050	0x2402000a	addiu $2,$0,10	35:	li	$v0, 10
	0x00400054	0x0000000c	syscall	36:	syscall	

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)
0x10010000	16	16	255	0	0	0	0
0x10010020	0	16	32	48	64	80	96
0x10010040	128	144	160	176	192	208	224
0x10010060	1	17	33	49	65	81	97
0x10010080	129	145	161	177	193	209	225
0x100100a0	2	18	34	50	66	82	98
0x100100c0	130	146	162	178	194	210	226
0x100100e0	3	19	35	51	67	83	99
0x10010100	131	147	163	179	195	211	227
0x10010120	4	20	36	52	68	84	100
0x10010140	132	148	164	180	196	212	228
0x10010160	5	21	37	53	69	85	101
0x10010180	133	149	165	181	197	213	229
0x100101a0	6	22	38	54	70	86	102
0x100101c0	134	150	166	182	198	214	230
0x100101e0	7	23	39	55	71	87	103

Fonte : Elaborado por autores (2023)

4 CONCLUSÃO

Em resumo, os códigos desenvolvidos para o laboratório 5 atingiram com êxito os objetivos estabelecidos. Ambos os programas foram implementados de acordo com as especificações das questões, preenchendo uma matriz 16x16 com valores de 0 a 255, alternando entre preenchimento por linha e por coluna.

Agora, estamos prontos para avançar para o laboratório 6, onde analisaremos o desempenho desses códigos em relação à memória cache, buscando compreender como o acesso sequencial à memória afeta o desempenho do sistema de computadores.