

Pontifícia Universidade Católica de Minas Gerais



PUC Minas

Relatório 9 MIPS

Alunos	Cora Silberschneider Ian Rodrigues dos Reis Paixão Luigi Domenico Cecchini Soares Rithie Natan Carvalhaes Prado
Professor	Romanelli Lordron Zuim

Belo Horizonte, 21 de maio de 2017

Conteúdo

1	Perguntas	1
2	Programas	2
2.1	Programa 13	2
2.2	Programa 14	3
2.3	Programa 15	3
2.4	Programa 16	4
2.5	Programa 17	4
2.6	Programa 18	5
2.7	Programa 19	6
2.8	Programa 20	6
2.9	Programa 21	7
2.10	Programa 22	7
2.11	Programa 23	8

1 Perguntas

1. Se tivermos 2 inteiros, cada um com 32 bits, quantos bits podemos esperar para o produto?

R: C (64 bits)

2. Quais os registradores que armazenam os resultados na multiplicação?

R: B (hi e lo)

3. Qual a operação usada para multiplicar inteiros em comp. de dois?

R: A (mult)

4. Qual instrução move os bits menos significativos da multiplicação para o reg. 8?

R: C (mflo \$8)

5. Se tivermos dois inteiros, cada um com 32 bits, quantos bits deveremos estar preparados para receber no quociente?

R: B (32 bits)

6. Após a instrução div, qual registrador possui o quociente?

R: A (lo)

7. Qual a inst. Usada para dividir dois inteiros em comp. de dois?

R: D (div)

8. Faça um arithmetic shift right de dois no seguinte padrão de bits: 1001 1011

R: A (1110 0110)

9. Qual o efeito de um arithmetic shift right de uma posição?

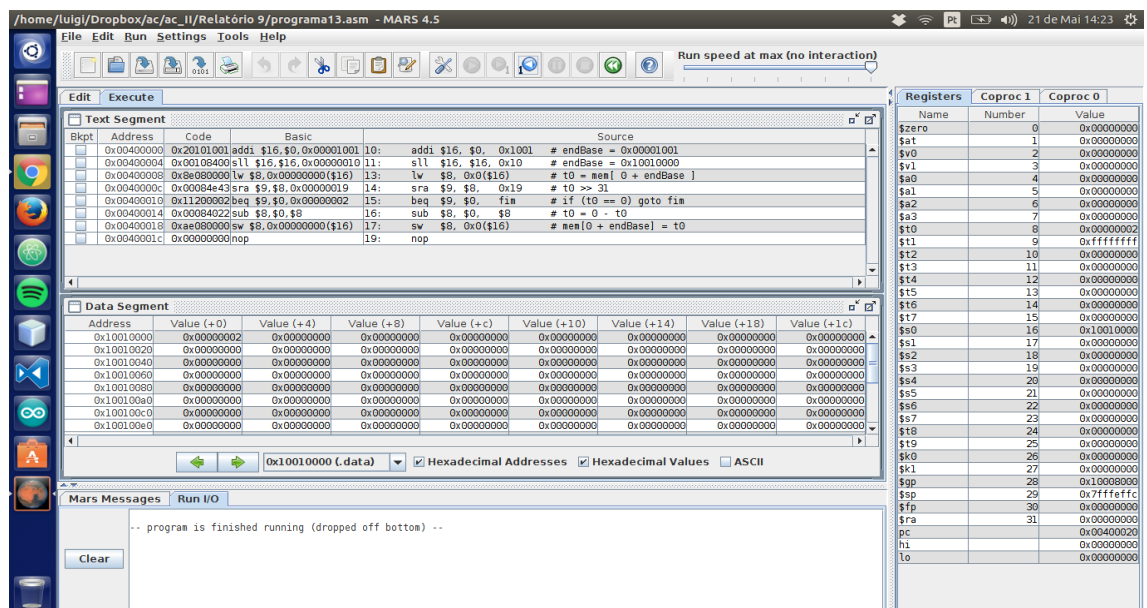
R: A (Se o inteiro for unsigned, o shift o divide por 2. Se o inteiro for signed, o shift o divide por 2)

10. Qual sequencia de instruções avalia $3x+7$, onde x é iniciado no reg. \$8 e o resultado armazenado em \$9?

R: A (ori \$3,\$0,3
mult \$8,\$3
mflo \$9
addi \$9,\$9,7)

2 Programas

2.1 Programa 13



2.2 Programa 14

The screenshot shows the MARS MIPS assembler simulator. The title bar indicates the file path: `/home/luigi/Dropbox/ac_jac_II/Relatório 9/programa14.asm - MARS 4.5`. The status bar shows the date and time: 21 de Mai 14:24.

The **Text Segment** pane displays the assembly code for Program 14. The code is as follows:

```

0x00400000: 0x21081001 addi $0,$0,0x00001001 8: addi $0, $0, 0x1001 # t0 = 0x00001001
0x00400004: 0x00084000 sll $0,$0,0x00000010 9: sll $0, $0, 0x10 # t0 = 0x10010000
0x00400008: 0x0d090000 lw $9,0x00000000($0) 10: lw $9, 0x0($0) # t1 = temp
0x0040000c: 0x212affe2 addi $10,$9,0xffffffe2 11: addi $10, $9, -30 # t2 = temp - 30
0x00400010: 0x212bffe2 addi $11,$9,0xffffffe2 12: addi $11, $9, -50 # t3 = temp - 50
0x00400014: 0x000a5fc2 srl $10,$10,0x00000013: srl $10, $10, 31 # t2 = t2 >> 31
0x00400018: 0x000b5fc2 srl $11,$11,0x0000001f 14: srl $11, $11, 31 # t3 = t3 >> 31
0x0040001c: 0x014b6026 xor $12,$10,$11 15: xor $12, $10, $11 # t4 = t2 xor t3
0x00400020: 0x18000002 beq $12,$0,0x00000002 16: beq $12, $0, foraDaFaixa # if (t4 == 0) goto foraDaFaixa
0x00400024: 0xad0c0044 sw $12,0x00000004($0) 17: sw $12, 0x4($0) # mem[4 + t0] = 1
0x00400028: 0x00100000 j 0x00400030 18: j foraDaFaixa # goto foraDaFaixa
0x0040002c: 0xad0c0044 sw $12,0x00000004($0) 20: sw $12, 0x4($0) # mem[4 + t0] = 0
0x00400030: 0x00000000 nop 22: nop

```

The **Data Segment** pane shows a memory layout with addresses from 0x10010000 to 0x1001000f and values from 0x00000000 to 0x0000000f.

The **Registers** pane shows the state of the MIPS registers. The registers are numbered 0 to 31, with names \$zero, \$at, \$v0, \$v1, \$a0, \$a1, \$a2, \$a3, \$t0, \$t1, \$t2, \$t3, \$t4, \$t5, \$t6, \$t7, \$s0, \$s1, \$s2, \$s3, \$s4, \$s5, \$s6, \$s7, \$t8, \$t9, \$k0, \$k1, \$gp, \$sp, \$fp, \$ra, \$pc, \$hi, and \$lo. The values are shown in hexadecimal.

The **Mars Messages** pane shows the output of the program, indicating that the program is finished running (dropped off bottom).

2.3 Programa 15

The screenshot shows the MARS MIPS assembler simulator. The title bar indicates the file path: `/home/luigi/Dropbox/ac_jac_II/Relatório 9/programa15.asm - MARS 4.5`. The status bar shows the date and time: 21 de Mai 14:24.

The **Text Segment** pane displays the assembly code for Program 15. The code is as follows:

```

0x00400000: 0x20081001 addi $0,$0,0x00001001 9: addi $0, $0, 0x1001 # t0 = 0x00001001
0x00400004: 0x00084000 sll $16,$0,0x00000010 10: sll $16, $0, 0x10 # endBase = 0x10010000
0x00400008: 0x200a0064 addi $10,$0,0x00000064 11: addi $10, $0, 0x64 # i = 100
0x0040000c: 0x00114080 sll $8,$17,0x00000002 14: sll $8, $17, 0x2 # i = i * 4
0x00400010: 0x00110402 add $8,$8,$16 15: add $8, $8, $16 # t0 = i * 4 + endBase
0x00400014: 0x00114840 sll $9,$17,0x00000001 16: sll $9, $17, 0x1 # t1 = i * 2
0x00400018: 0x21290001 addi $9,$9,0x00000001 17: addi $9, $9, 0x1 # t1 = i * 2 + 1
0x0040001c: 0xad090000 sw $9,0x00000000($0) 18: sw $9, 0($0) # vet[i] = t1
0x00400020: 0x02499020 add $18,$18,$9 19: add $18, $18, $9 # soma = soma + t1
0x00400024: 0x22310001 addi $17,$17,0x0000... 20: addi $17, $17, 0x1 # i = i + 1
0x00400028: 0x162afffb bne $17,$10,0xfffffff8 21: bne $17, $10, do # if (i != 0) goto do
0x0040002c: 0xae120190 sw $18,0x0000190($16) 23: sw $18, 400($16) # vet[100] = soma

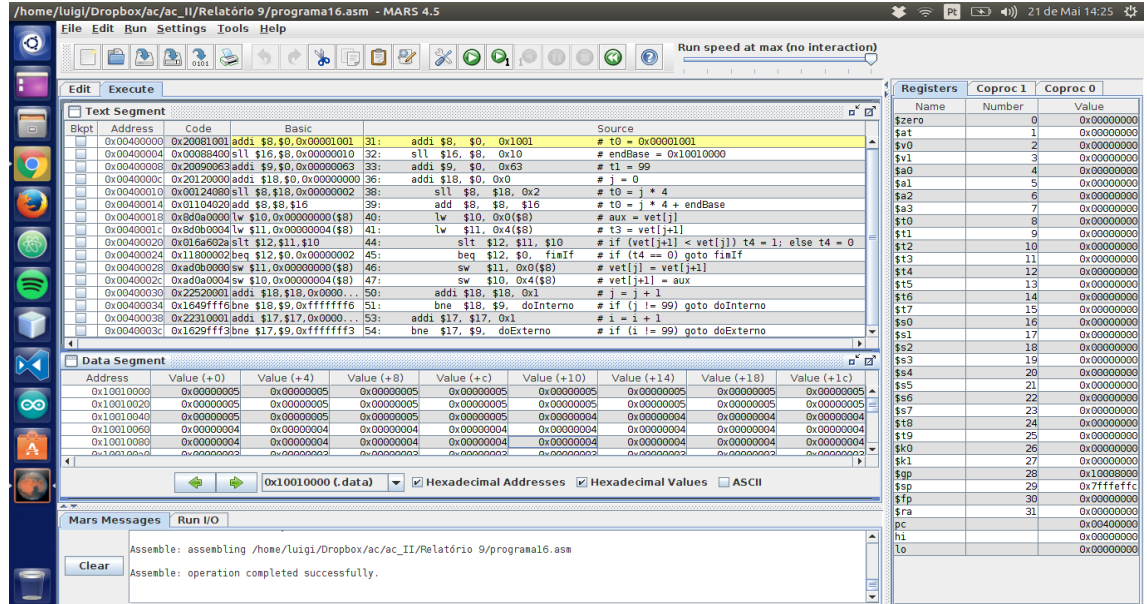
```

The **Data Segment** pane shows a memory layout with addresses from 0x10010000 to 0x1001000f and values from 0x00000000 to 0x0000000f.

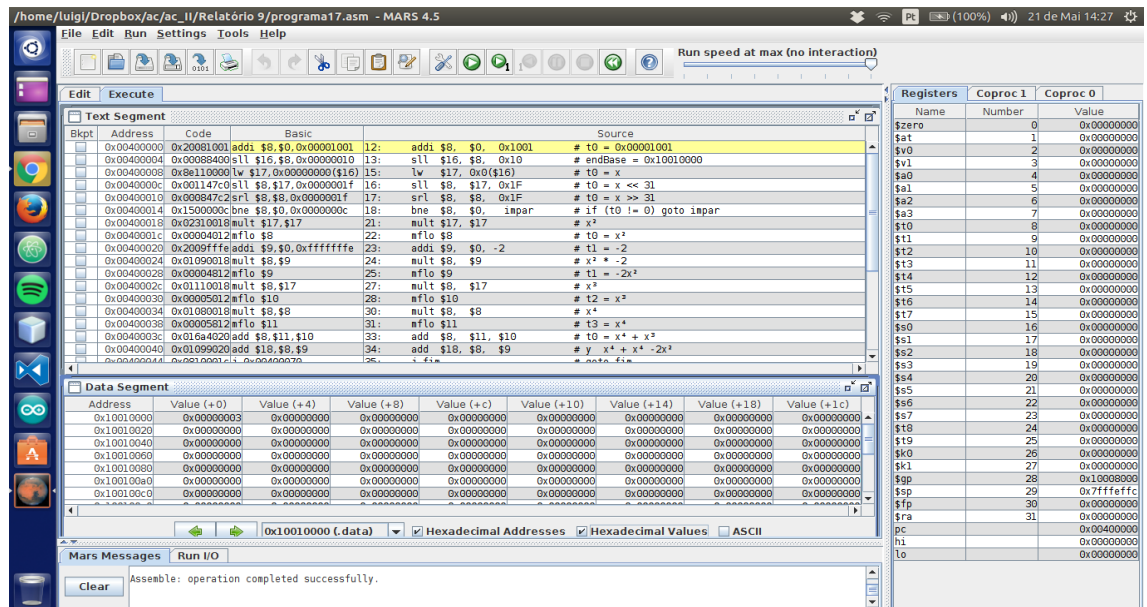
The **Registers** pane shows the state of the MIPS registers. The registers are numbered 0 to 31, with names \$zero, \$at, \$v0, \$v1, \$a0, \$a1, \$a2, \$a3, \$t0, \$t1, \$t2, \$t3, \$t4, \$t5, \$t6, \$t7, \$s0, \$s1, \$s2, \$s3, \$s4, \$s5, \$s6, \$s7, \$t8, \$t9, \$k0, \$k1, \$gp, \$sp, \$fp, \$ra, \$pc, \$hi, and \$lo. The values are shown in hexadecimal.

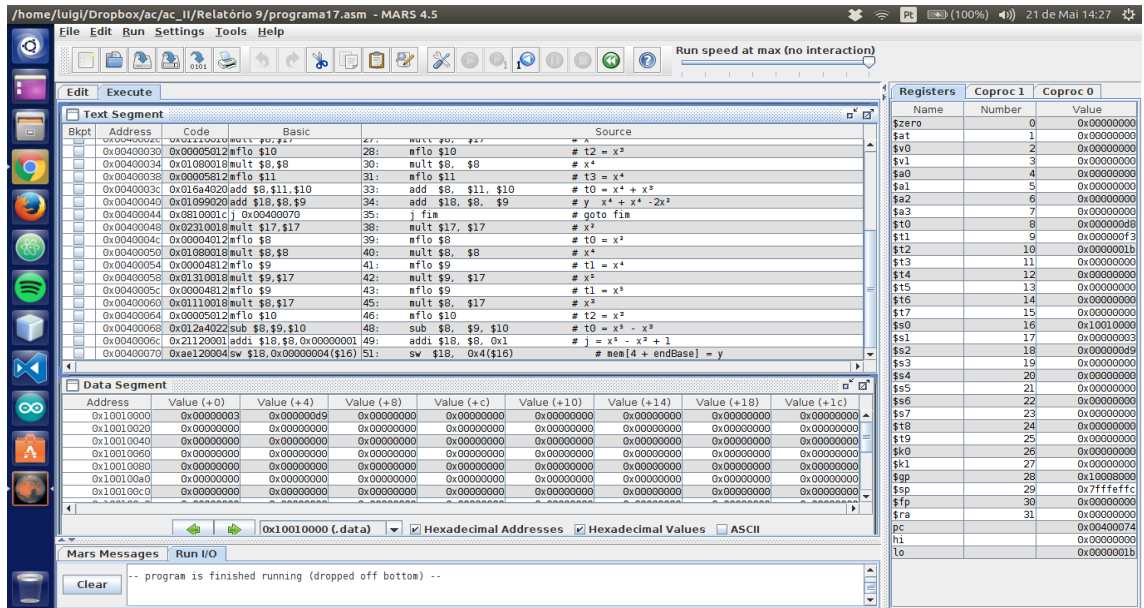
The **Mars Messages** pane shows the output of the program, indicating that the program is finished running (dropped off bottom).

2.4 Programa 16

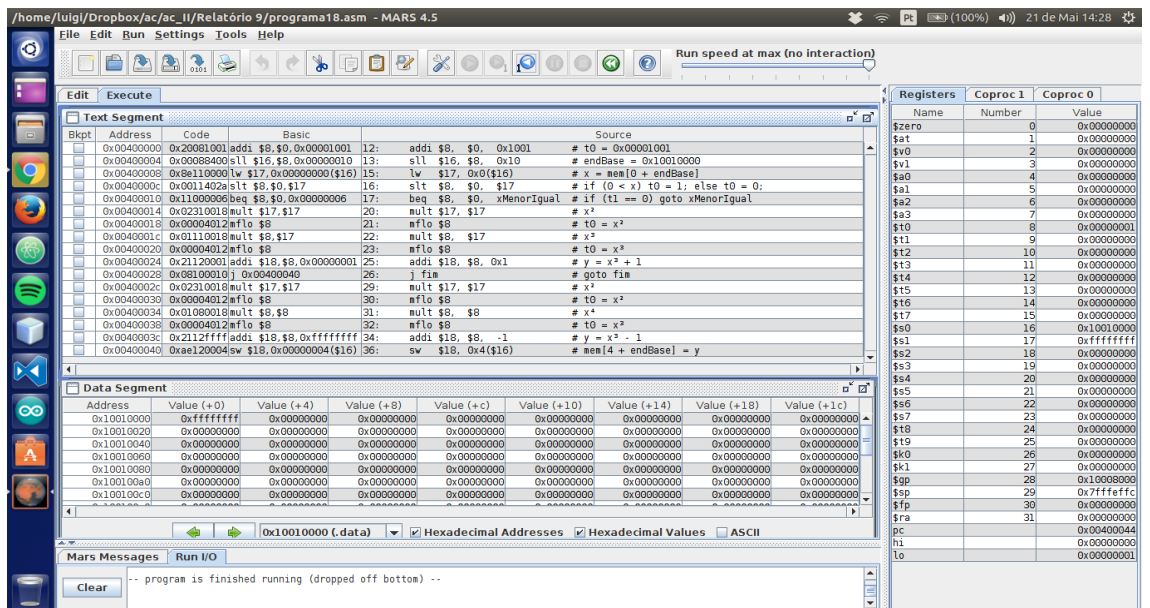


2.5 Programa 17

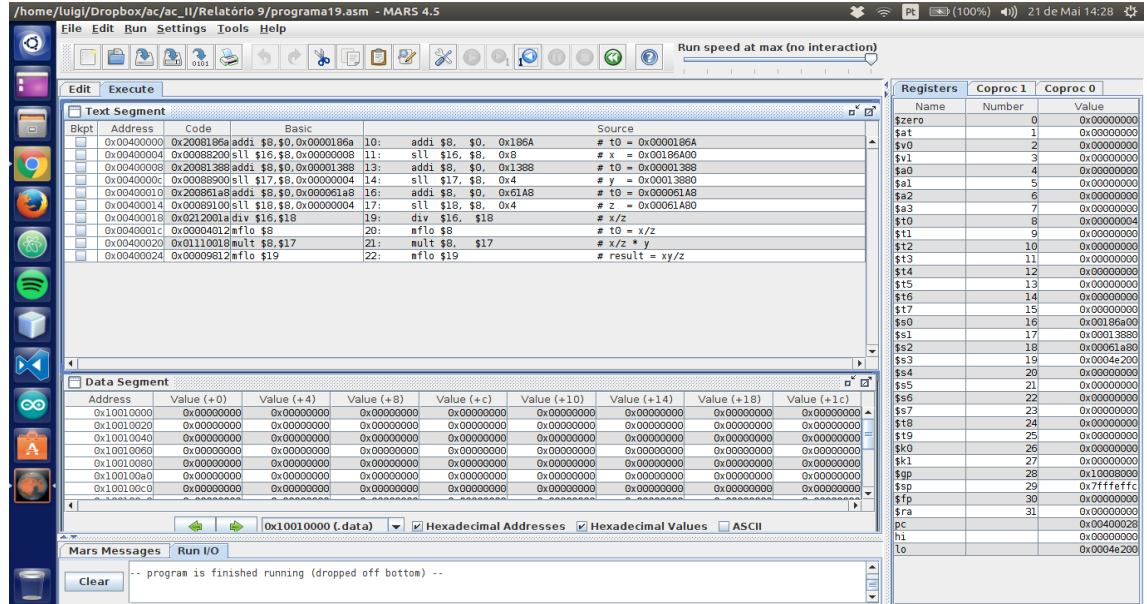




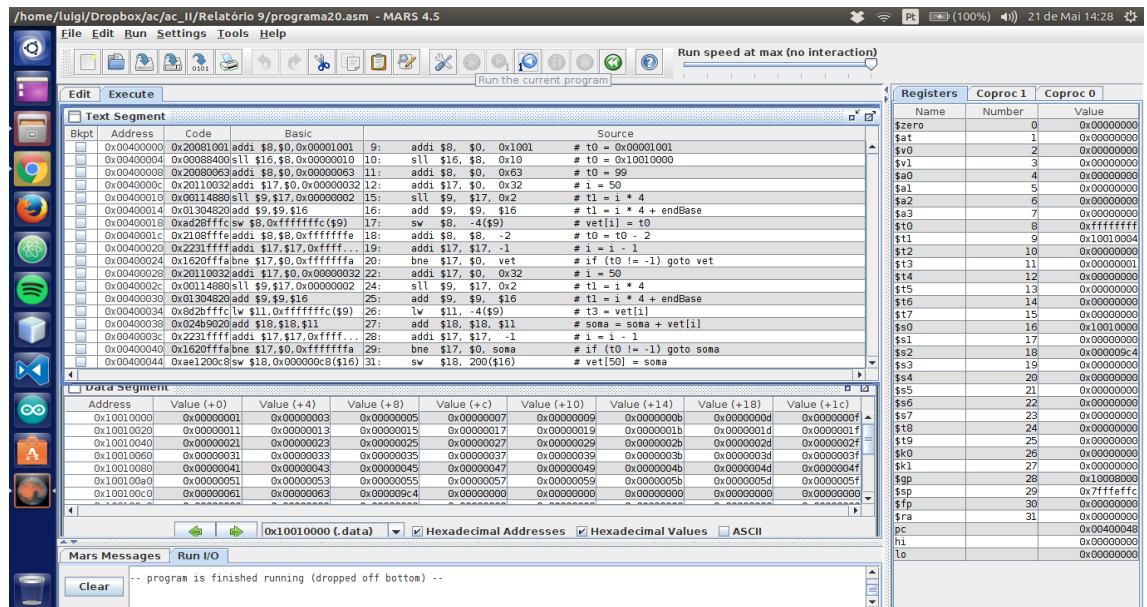
2.6 Programa 18



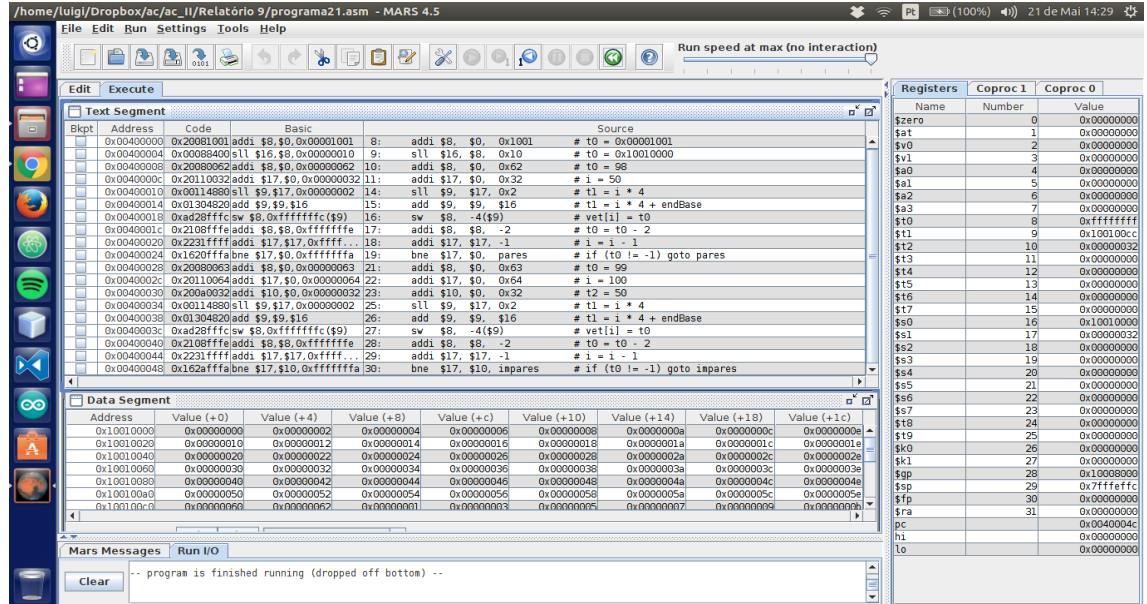
2.7 Programa 19



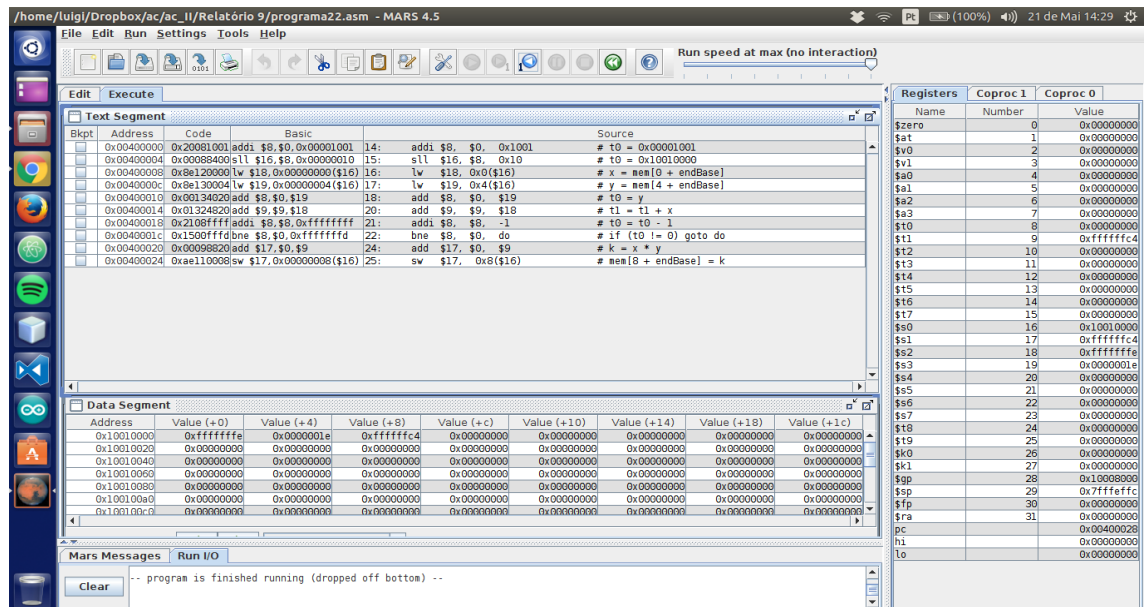
2.8 Programa 20



2.9 Programa 21



2.10 Programa 22



2.11 Programa 23

/home/luigi/Dropbox/ac/ac_II/Relatório 9/programa23.asm - MARS 4.5

File Edit Run Settings Tools Help

Run speed at max (no interaction)

Text Segment

Bkpt	Address	Code	Basic	Source
	0x00400000	0x20081001	addi \$0,\$0,0x00001001	14: addi \$0, \$0, 0x1001 # t0 = 0x00001001
	0x00400004	0x00088400	sll \$16,\$0,0x0000010	15: sll \$16, \$0, 0x10 # t0 = 0x10010000
	0x00400008	0x8e120000	lw \$18,0x00000000(\$16)	16: lw \$18, 0x0(\$16) # x = mem[0 + endBase]
	0x0040000c	0x8e130004	lw \$19,0x00000004(\$16)	17: lw \$19, 0x4(\$16) # y = mem[4 + endBase]
	0x00400010	0x000a8820	add \$17,\$0,\$10	18: add \$17, \$0, \$10 # k = x*y
	0x00400014	0x00124020	add \$8,\$0,\$19	19: add \$0, \$0, \$18 # t0 = x
	0x00400018	0x00135020	add \$10,\$0,\$19	20: add \$10, \$0, \$19 # t2 = y
	0x0040001c	0x214affff	addi \$10,\$10,0xffff...	21: add \$10, \$10, -1 # t2 = y - 1
	0x00400020	0x00124820	add \$9,\$0,\$18	24: add \$9, \$0, \$18 # t1 = x
	0x00400024	0x00005620	add \$11,\$0,\$0	25: add \$11, \$0, \$0 # t3 = 0
	0x00400028	0x01868620	add \$11,\$11,\$8	27: add \$11, \$11, \$8 # t3 = t3 + t0
	0x0040002c	0x2129ffff	addi \$9,\$9,0xffff...	28: addi \$9, \$9, -1 # t1 = t1 - 1
	0x00400030	0x1520ffff	bne \$9,\$0,0xffff...	29: bne \$9, \$0, doInterno # if (t1 != 0) goto doInterno
	0x00400034	0x000b4020	add \$8,\$0,\$11	31: add \$8, \$0, \$11 # t0 = t3
	0x00400038	0x214affff	addi \$10,\$10,0xffff...	32: addi \$10, \$10, -1 # t2 = t2 - 1
	0x0040003c	0x1540ffff	bne \$10,\$0,0xffff...	33: bne \$10, \$0, doExterno # if (t2 != 0) goto doExterno
	0x00400040	0x000b8820	add \$17,\$0,\$11	35: add \$17, \$0, \$11 # k = x*y
	0x00400044	0xae110008	sw \$17,0x00000008(\$16)	36: sw \$17, 0x8(\$16) # mem[8 + endBase] = k

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x00000000	0x00000004	0x00000051	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

Registers

Name	Number	Value
\$zero	0	0x00000000
\$at	1	0x00000000
\$v0	2	0x00000000
\$v1	3	0x00000000
\$a0	4	0x00000000
\$a1	5	0x00000000
\$a2	6	0x00000000
\$a3	7	0x00000000
\$t0	8	0x00000051
\$t1	9	0x00000000
\$t2	10	0x00000000
\$t3	11	0x00000051
\$t4	12	0x00000000
\$t5	13	0x00000000
\$t6	14	0x00000000
\$t7	15	0x00000000
\$s0	16	0x10010000
\$s1	17	0x00000051
\$s2	18	0x00000003
\$s3	19	0x00000004
\$s4	20	0x00000000
\$s5	21	0x00000000
\$s6	22	0x00000000
\$s7	23	0x00000000
\$t8	24	0x00000000
\$t9	25	0x00000000
\$k0	26	0x00000000
\$k1	27	0x00000000
\$gp	28	0x10008000
\$sp	29	0x7ffffcfc
\$fp	30	0x00000000
\$ra	31	0x00000000
\$pc		0x00400048
\$hi		0x00000000
\$lo		0x00000000

Mars Messages Run I/O

Clear -- program is finished running (dropped off bottom) --