

Nome: Gabriel Henrique Vieira de Oliveira

Data: 09/06/2024

Matéria: Arquitetura de Computadores II - Exercício prático 06

Parte 1 - Resposta

- 1) A
- 2) B
- 3) A
- 4) C
- 5) D (É o indicador de endereço de memória externa de programa. Ele serve para endereçar a próxima instrução a ser lida pelo microprocessador a fim de que ele "saiba" onde estava e para onde deve ir (ler nova instrução no próximo endereço).
- 6) C
- 7) D
- 8) D
- 9) B
- 10) A
- 11) B (OPERAÇÃO BITWISE)
- 12) D
- 13) A
- 14) C
- 15) B
- 16) D
- 17) A
- 18) A

Parte 2 - Implementar em MIPS/MARS os seguintes programas

PROGRAMA 1:

The screenshot displays the MARS MIPS simulator interface. The main window shows the assembly code for 'Programa1.asm'. The code includes comments in Portuguese and MIPS assembly instructions. The register window on the right shows the state of the MIPS registers, with the 'Zero' register set to 0.

```
1  # //programa 1 (add, addi, sub, lógicas) {
2  #   a = 2;
3  #   b = 3;
4  #   c = 4;
5  #   d = 5;
6  #   x = (a+b) - (c+d);
7  #   y = a - b + x;
8  #   b = x - y;
9  # }
10
11 #Associações
12 # a = $s0
13 # b = $s1
14 # c = $s2
15 # d = $s3
16
17 # x = $s4
18 # y = $s5
19
20 .text
21 .globl main
22 main:
23 addi $s0, $zero, 2 # a = 2
24 addi $s1, $zero, 3 # b = 3
25 addi $s2, $zero, 4 # c = 4
26 addi $s3, $zero, 5 # d = 5
27
28 add $s4, $s0, $s1 # x = a + b
29 sub $s4, $s4, $s2 # x = x - c
30 sub $s4, $s4, $s3 # x = x - d
31
32 sub $s5, $s0, $s1 # y = (a+b) - a - b
33 add $s5, $s5, $s4 # y = y + x
34
35 sub $s1, $s4, $s5 # b = x - y
```

Name	Number	Value
\$zero	0	0
\$at	1	0
\$v0	2	0
\$v1	3	0
\$a0	4	0
\$a1	5	0
\$a2	6	0
\$a3	7	0
\$t0	8	0
\$t1	9	0
\$t2	10	0
\$t3	11	0
\$t4	12	0
\$t5	13	0
\$t6	14	0
\$t7	15	0
\$s0	16	0
\$s1	17	0
\$s2	18	0
\$s3	19	0
\$s4	20	0
\$s5	21	0
\$s6	22	0
\$s7	23	0
\$t8	24	0
\$t9	25	0
\$t0	26	0
\$t1	27	0
\$gp	28	268468224
\$sp	29	2147479548
\$fp	30	0
\$ra	31	0
\$PC		4194304
\$hi		0
\$lo		0

Run speed at max (no interaction)

Edit Execute

Text Segment

Offset	Address	Code	Basic	Source
0x00400000	0x20100002	addi \$t6, \$t0, 2	23: addi \$t6, \$zero, 2 # a = 2	
0x00400004	0x20100003	addi \$t7, \$t0, 3	24: addi \$t7, \$zero, 3 # b = 3	
0x00400008	0x20100004	addi \$t8, \$t0, 4	25: addi \$t8, \$zero, 4 # c = 4	
0x0040000c	0x20100005	addi \$t9, \$t0, 5	26: addi \$t9, \$zero, 5 # d = 5	
0x00400010	0x2011a020	add \$t0, \$t6, \$t7	28: add \$t0, \$t6, \$t7 # x = a + b	
0x00400014	0x0292a022	sub \$t0, \$t0, \$t8	29: sub \$t0, \$t0, \$t8 # x = x - c	
0x00400018	0x0293a022	sub \$t0, \$t0, \$t9	30: sub \$t0, \$t0, \$t9 # x = x - d	
0x0040001c	0x0211e022	sub \$t1, \$t6, \$t7	32: sub \$t5, \$t0, \$t1 # y (t5) = a - b	
0x00400020	0x01b4a020	add \$t1, \$t5, \$t0	33: add \$t5, \$t5, \$t0 # y = y + x	
0x00400024	0x0295e022	sub \$t1, \$t0, \$t5	35: sub \$t1, \$t0, \$t5 # b = x - y	

Labels

Label	Address
(global)	
main	0x00400000

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0	0	0	0	0	0	0	0
0x10010020	0	0	0	0	0	0	0	0
0x10010040	0	0	0	0	0	0	0	0
0x10010060	0	0	0	0	0	0	0	0
0x10010080	0	0	0	0	0	0	0	0
0x100100a0	0	0	0	0	0	0	0	0
0x100100c0	0	0	0	0	0	0	0	0
0x100100e0	0	0	0	0	0	0	0	0
0x10010100	0	0	0	0	0	0	0	0
0x10010120	0	0	0	0	0	0	0	0
0x10010140	0	0	0	0	0	0	0	0
0x10010160	0	0	0	0	0	0	0	0
0x10010180	0	0	0	0	0	0	0	0
0x100101a0	0	0	0	0	0	0	0	0

Mars Messages Run IO

program is finished running (dropped off bottom) --

PROGRAMA 2:

Edit Execute

Programa1.asm Programa2.asm

```
1 # //programa 2 (add, addi, sub, logicas) (  
2 # x = 1;  
3 # y = 5*x + 15;  
4 # )  
5  
6 #Associateades  
7 # x = $t0  
8 # y = $t1  
9  
10 .text  
11 .globl main  
12  
13 main:  
14 ori $t0, $zero, 0x0001 # x = 1  
15 add $t0, $t0, $t0 # t0 = 2x (x + x)  
16 add $t0, $t0, $t0 # t0 = 4x (2x + 2x)  
17 add $t0, $t0, $t0 # t0 = 8x (4x + 4x)  
18  
19 addi $t1, $t0, 15 # y = 5x + 15  
20 #fin
```

Registers Coproc 1 Coproc 0

Name	Number	Value
\$zero	0	0
\$at	1	0
\$v0	2	0
\$v1	3	0
\$a0	4	0
\$a1	5	0
\$a2	6	0
\$a3	7	0
\$t0	8	5
\$t1	9	0
\$t2	10	0
\$t3	11	0
\$t4	12	0
\$t5	13	0
\$t6	14	0
\$t7	15	0
\$t8	16	5
\$t9	17	20
\$s0	18	0
\$s1	19	0
\$s2	20	0
\$s3	21	0
\$s4	22	0
\$s5	23	0
\$s6	24	0
\$s7	25	0
\$s8	26	0
\$s9	27	0
\$gp	28	268469224
\$sp	29	2147479545
\$fp	30	0
\$ra	31	0
pc		4194324
hi		0
lo		0

Text Segment

Offset	Address	Code	Basic	Source
0x00400000	0x34100001	ori \$t0, \$zero, 0x0001 # x = 1	24: ori \$t0, \$zero, 0x0001 # x = 1	
0x00400004	0x02104020	add \$t0, \$t0, \$t0	15: add \$t0, \$t0, \$t0 # t0 = 2x (x + x)	
0x00400008	0x01084020	add \$t0, \$t0, \$t0	16: add \$t0, \$t0, \$t0 # t0 = 4x (2x + 2x)	
0x0040000c	0x01104020	add \$t0, \$t0, \$t0	17: add \$t0, \$t0, \$t0 # t0 = 8x (4x + 4x)	
0x00400010	0x2111000f	addi \$t1, \$t0, 15 # y = 5x + 15	19: addi \$t1, \$t0, 15 # y = 5x + 15	

Labels

Label	Address
(global)	
main	0x00400000

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0	0	0	0	0	0	0	0
0x10010020	0	0	0	0	0	0	0	0
0x10010040	0	0	0	0	0	0	0	0
0x10010060	0	0	0	0	0	0	0	0
0x10010080	0	0	0	0	0	0	0	0
0x100100a0	0	0	0	0	0	0	0	0
0x100100c0	0	0	0	0	0	0	0	0
0x100100e0	0	0	0	0	0	0	0	0
0x10010100	0	0	0	0	0	0	0	0
0x10010120	0	0	0	0	0	0	0	0
0x10010140	0	0	0	0	0	0	0	0
0x10010160	0	0	0	0	0	0	0	0
0x10010180	0	0	0	0	0	0	0	0
0x100101a0	0	0	0	0	0	0	0	0

Registers Coproc 1 Coproc 0

Name	Number	Value
\$zero	0	0
\$at	1	0
\$v0	2	0
\$v1	3	0
\$a0	4	0
\$a1	5	0
\$a2	6	0
\$a3	7	0
\$t0	8	5
\$t1	9	0
\$t2	10	0
\$t3	11	0
\$t4	12	0
\$t5	13	0
\$t6	14	0
\$t7	15	0
\$t8	16	5
\$t9	17	20
\$s0	18	0
\$s1	19	0
\$s2	20	0
\$s3	21	0
\$s4	22	0
\$s5	23	0
\$s6	24	0
\$s7	25	0
\$s8	26	0
\$s9	27	0
\$gp	28	268469224
\$sp	29	2147479545
\$fp	30	0
\$ra	31	0
pc		4194324
hi		0
lo		0

PROGRAMA 3:

EditExecute

Programa1.asmPrograma2.asmPrograma3.asm

```
1 # // programa 3 (add, addi, sub, lógicas) {
2 #   x = 3;
3 #   y = 4;
4 #   z = ( 15*x + 67*y)*4
5 # }
6
7 #Associações
8 # x = $s0
9 # y = $s1
10 # z = $s2
11
12 .text
13 .globl main
14
15 main:
16     ori $s0, $zero, 3 # x = 3
17     ori $s1, $zero, 0x0004 # y = 4
18     add $t0, $s0, $s0 # t0 = 2x
19     add $t0, $t0, $t0 # t0 = 4x
20     add $t0, $t0, $t0 # t0 = 8x
21     add $t0, $t0, $t0 # t0 = 16x
22     sub $t0, $t0, $s0 # t0 = 16x - x = 15x (como usamos uma variavel temporaria, o x se torna o elemento da conta fazendo o numero diminuir para 15x)
23     add $t1, $s1, $s1 # t1 = 2y
24     add $t1, $t1, $t1 # t1 = 4y
25     add $t1, $t1, $t1 # t1 = 8y
26     add $t1, $t1, $t1 # t1 = 16y
27     add $t1, $t1, $t1 # t1 = 32y
28     add $t1, $t1, $t1 # t1 = 64y
29     add $t1, $t1, $s1 # t1 = 64y + y = 65y
30     add $t1, $t1, $s1 # t1 = 64y + y = 66y
31     add $t1, $t1, $s1 # t1 = 64y + y = 67y
32     add $t3, $t0, $t1 # t3 = 15x + 67y
33     add $t3, $t3, $t3 # t3 = 2*(15x + 67y)
34     add $s2, $t3, $t3 # z = 4*(15x + 67y)
35
36
```

Line: 34 Column: 39 Show Line Numbers

RegistersCoproccoproc 0

Name	Number	Value
\$zero	0	0
\$at	1	0
\$v0	2	0
\$v1	3	0
\$a0	4	0
\$a1	5	0
\$a2	6	0
\$a3	7	0
\$t0	8	45
\$t1	9	268
\$t2	10	0
\$t3	11	626
\$t4	12	0
\$t5	13	0
\$t6	14	0
\$t7	15	0
\$s0	16	3
\$s1	17	4
\$s2	18	1252
\$s3	19	0
\$s4	20	0
\$s5	21	0
\$s6	22	0
\$s7	23	0
\$s8	24	0
\$s9	25	0
\$k0	26	0
\$k1	27	0
\$gp	28	268468224
\$fp	29	2147479546
\$tp	30	0
\$ra	31	0
pc		4194380
hi		0
lo		0

EditExecute

Text Segment

Byte	Address	Code	Basic	Source
0x00400000	0x01080020	add \$s,t0,\$t0	19:	add \$t0, \$s0, \$s0 # t0 = 2x
0x00400000	0x01080020	add \$s,t0,\$t0	19:	add \$t0, \$t0, \$t0 # t0 = 4x
0x00400010	0x01080020	add \$s,t0,\$t0	20:	add \$t0, \$t0, \$t0 # t0 = 8x
0x00400014	0x01080020	add \$s,t0,\$t0	21:	add \$t0, \$t0, \$t0 # t0 = 16x
0x00400018	0x01080020	sub \$s,t0,\$s0	22:	sub \$t0, \$t0, \$s0 # t0 = 16x - x = 15x (como usamos uma variavel temporaria, o ...
0x0040001c	0x02314820	add \$s,t1,\$t1	23:	add \$t1, \$s1, \$s1 # t1 = 2y
0x00400020	0x02314820	add \$s,t1,\$t1	24:	add \$t1, \$t1, \$t1 # t1 = 4y
0x00400024	0x02314820	add \$s,t1,\$t1	25:	add \$t1, \$t1, \$t1 # t1 = 8y
0x00400028	0x02314820	add \$s,t1,\$t1	26:	add \$t1, \$t1, \$t1 # t1 = 16y
0x0040002c	0x02314820	add \$s,t1,\$t1	27:	add \$t1, \$t1, \$t1 # t1 = 32y
0x00400030	0x02314820	add \$s,t1,\$t1	28:	add \$t1, \$t1, \$t1 # t1 = 64y
0x00400034	0x03114820	add \$s,t1,\$t1	29:	add \$t1, \$t1, \$s1 # t1 = 64y + y = 65y
0x00400038	0x03114820	add \$s,t1,\$t1	30:	add \$t1, \$t1, \$s1 # t1 = 64y + y = 66y
0x0040003c	0x03114820	add \$s,t1,\$t1	31:	add \$t1, \$t1, \$s1 # t1 = 64y + y = 67y
0x00400040	0x01085820	add \$s,t3,\$t3	32:	add \$t3, \$t0, \$t1 # t3 = 15x + 67y
0x00400044	0x01085820	add \$s,t3,\$t3	33:	add \$t3, \$t3, \$t3 # t3 = 2*(15x + 67y)
0x00400048	0x01085820	add \$s,t3,\$t3	34:	add \$s2, \$t3, \$t3 # z = 4*(15x + 67y)

Labels

Label	Address
(global)	
main	0x00400000

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0	0	0	0	0	0	0	0
0x10010020	0	0	0	0	0	0	0	0
0x10010040	0	0	0	0	0	0	0	0
0x10010060	0	0	0	0	0	0	0	0
0x10010080	0	0	0	0	0	0	0	0
0x100100a0	0	0	0	0	0	0	0	0
0x100100c0	0	0	0	0	0	0	0	0
0x100100e0	0	0	0	0	0	0	0	0
0x10010100	0	0	0	0	0	0	0	0
0x10010120	0	0	0	0	0	0	0	0
0x10010140	0	0	0	0	0	0	0	0
0x10010160	0	0	0	0	0	0	0	0
0x10010180	0	0	0	0	0	0	0	0
0x100101a0	0	0	0	0	0	0	0	0
0x100101c0	0	0	0	0	0	0	0	0

Mars Messages

Run IO

PROGRAMA 4:

EditExecute

Programa1.asmPrograma2.asmPrograma3.asmPrograma4.asm

```
1 # // programa 4 {
2 #   x = 3;
3 #   y = 4;
4 #   z = ( 15*x + 67*y)*4
5 # }
6
7 #Associações
8 # x = $s0
9 # y = $s1
10 # z = $s2
11
12 .text
13 .globl main
14
15 main:
16     ori $s0, $zero, 3 # x = 3
17     ori $s1, $zero, 0x0004 # y = 4
18     ori $t0, $s0, 4 # 2*x = 16, ou seja, t0 = 16x
19     sub $t0, $t0, $s0 # t0 = 16x - x = 15x
20     sll $t3, $s1, 6 # 2^6 = 64, ou seja, t1 = 64y
21     add $t1, $t1, $s1 # t1 = 64y + y = 65y
22     add $t1, $t1, $s1 # t1 = 65y + y = 66y
23     add $t1, $t1, $s1 # t1 = 65y + y = 67y
24     add $t3, $t0, $t1 # t3 = 15x + 67y
25     sll $s2, $t3, 0x0002 # 2^2 = 4, ou seja, z = 4*(15x + 67y)
26
```

Line: 25 Column: 60 Show Line Numbers

RegistersCoproccoproc 0

Name	Number	Value
\$zero	0	0
\$at	1	0
\$v0	2	0
\$v1	3	0
\$a0	4	0
\$a1	5	0
\$a2	6	0
\$a3	7	0
\$t0	8	45
\$t1	9	268
\$t2	10	0
\$t3	11	313
\$t4	12	0
\$t5	13	0
\$t6	14	0
\$t7	15	0
\$s0	16	3
\$s1	17	4
\$s2	18	1252
\$s3	19	0
\$s4	20	0
\$s5	21	0
\$s6	22	0
\$s7	23	0
\$s8	24	0
\$s9	25	0
\$k0	26	0
\$k1	27	0
\$gp	28	268468224
\$fp	29	2147479546
\$tp	30	0
\$ra	31	0
pc		4194344
hi		0
lo		0

Text Segment

Byte	Address	Code	Basic	Source
	0x00400000	0x34100003	ori t16,t0,3	16: ori t\$0, \$zero, 3 # x = 3
	0x00400004	0x34110004	ori t17,t0,4	17: ori t\$1, \$zero, 0x0004 # y = 4
	0x00400008	0x00104100	slil \$t0,\$t0,16	18: slil t\$0, \$t0, 4 # 2^4 = 16, ou seja, t0 = 16x
	0x0040000c	0x01104012	sub t\$0,t\$0,t\$0	19: sub t\$0, t\$0, t\$0 # t0 = 16x - x = 15x
	0x00400010	0x00114980	slil t\$1,t\$1,6	20: slil t\$1, t\$1, 6 # 2^4 = 64, ou seja, t1 = 64y
	0x00400014	0x01314820	add t\$9,t\$9,t17	21: add t\$1, t\$1, t\$1 # t1 = 64y + y = 65y
	0x00400018	0x01314820	add t\$9,t\$9,t17	22: add t\$1, t\$1, t\$1 # t1 = 65y + y = 66y
	0x0040001c	0x01114820	add t\$9,t\$9,t17	23: add t\$1, t\$1, t\$1 # t1 = 65y + y = 67y
	0x00400020	0x01095820	add t11,t\$0,t\$9	24: add t\$3, t\$0, t\$1 # t3 = 15x + 67y
	0x00400024	0x000b0900	slil t10,t11,2	25: slil t\$2, t\$3, 0x0002 # 2^2 = 4, ou seja, z = 4*(15x + 67y)

Labels

Label	Address
(global)	
main	0x00400000

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0	0	0	0	0	0	0	0
0x10010020	0	0	0	0	0	0	0	0
0x10010040	0	0	0	0	0	0	0	0
0x10010060	0	0	0	0	0	0	0	0
0x10010080	0	0	0	0	0	0	0	0
0x100100a0	0	0	0	0	0	0	0	0
0x100100c0	0	0	0	0	0	0	0	0
0x100100e0	0	0	0	0	0	0	0	0
0x10010100	0	0	0	0	0	0	0	0
0x10010120	0	0	0	0	0	0	0	0
0x10010140	0	0	0	0	0	0	0	0
0x10010160	0	0	0	0	0	0	0	0
0x10010180	0	0	0	0	0	0	0	0
0x100101a0	0	0	0	0	0	0	0	0

Registers

Coproc 1

Coproc 0

Name	Number	Value
\$zero	0	0
\$at	1	0
\$v0	2	0
\$v1	3	0
\$a0	4	0
\$a1	5	0
\$a2	6	0
\$a3	7	0
\$t0	8	45
\$t1	9	265
\$t2	10	0
\$t3	11	313
\$t4	12	0
\$t5	13	0
\$t6	14	0
\$t7	15	0
\$a0	16	3
\$a1	17	4
\$a2	18	1252
\$a3	19	0
\$a4	20	0
\$a5	21	0
\$a6	22	0
\$a7	23	0
\$t8	24	0
\$t9	25	0
\$k0	26	0
\$k1	27	0
\$gp	28	265465224
\$sp	29	2147479545
\$fp	30	0
\$ra	31	0
\$pc		4194344
\$t1		0
\$t0		0

PROGRAMA 5:

Programa5.asm

Programa5.asm

```
1 // programa 5 (
2 # x = 100000; HEXA = 0x0001 $6A0
3 # y = 200000; HEXA = 0x0003 $D40
4 # z = x + y;
5 # }
6
7 #Associações
8 # x = $t0
9 # y = $t1
10 # z = $t3
11
12 # O NUMERO DE CASAS PARA ANDAR, NESSE CASO, TEM QUE ESTAR EM BINARIO
13 # 0x0010 equivale a 16 bits
14 .text
15 .globl main
16
17 main:
18     ori t$0, $zero, 0x0001 # t0 = 0x0000 0001
19     slil t$0, t$0, 16 # t0 = 0x0001 0000 (andar quatro casas para a esquerda, ou seja, 2^4 = 16 bits)
20     ori t$0, t$0, 0x6A0 # x = 0x0001 $6A0
21
22     ori t$1, $zero, 0x0003 # t1 = 0x0000 0003
23     slil t$1, t$1, 0x0010 # t1 = 0x0002 0000
24     ori t$1, t$1, 0x0D40 # y = 0x0003 $D40
25
26     add t$3, t$0, t$1 # z = x + y
27
28
```

Line: 19 Column: 99 | Show Line Numbers

Text Segment

Byte	Address	Code	Basic	Source
	0x00400000	0x34080001	ori t\$0,t0,1	18: ori t\$0, \$zero, 0x0001 # t0 = 0x0000 0001
	0x00400004	0x00084400	slil t\$0,t\$0,16	19: slil t\$0, t\$0, 16 # t0 = 0x0001 0000 (andar quatro casas para a esquerda, ou se...
	0x00400008	0x351086A0	ori t\$0,t\$0,34464	20: ori t\$0, t\$0, 0x6A0 # x = 0x0001 \$6A0
	0x0040000c	0x34090003	ori t\$1,t\$0,3	22: ori t\$1, \$zero, 0x0003 # t1 = 0x0000 0003
	0x00400010	0x00094600	slil t\$1,t\$1,9	23: slil t\$1, t\$1, 0x0010 # t1 = 0x0002 0000
	0x00400014	0x35110440	ori t\$1,t\$1,3392	24: ori t\$1, t\$1, 0x0D40 # y = 0x0003 \$D40
	0x00400018	0x02119820	add t19,t\$0,t\$1	26: add t\$3, t\$0, t\$1 # z = x + y

Labels

Label	Address
(global)	
main	0x00400000

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0	0	0	0	0	0	0	0
0x10010020	0	0	0	0	0	0	0	0
0x10010040	0	0	0	0	0	0	0	0
0x10010060	0	0	0	0	0	0	0	0
0x10010080	0	0	0	0	0	0	0	0
0x100100a0	0	0	0	0	0	0	0	0
0x100100c0	0	0	0	0	0	0	0	0
0x100100e0	0	0	0	0	0	0	0	0
0x10010100	0	0	0	0	0	0	0	0
0x10010120	0	0	0	0	0	0	0	0
0x10010140	0	0	0	0	0	0	0	0
0x10010160	0	0	0	0	0	0	0	0
0x10010180	0	0	0	0	0	0	0	0
0x100101a0	0	0	0	0	0	0	0	0

Registers

Coproc 1

Coproc 0

Name	Number	Value
\$zero	0	0
\$at	1	0
\$v0	2	0
\$v1	3	0
\$a0	4	0
\$a1	5	0
\$a2	6	0
\$a3	7	0
\$t0	8	65536
\$t1	9	196608
\$t2	10	0
\$t3	11	0
\$t4	12	0
\$t5	13	0
\$t6	14	0
\$t7	15	0
\$a0	16	100000
\$a1	17	200000
\$a2	18	300000
\$a3	19	300000
\$a4	20	0
\$a5	21	0
\$a6	22	0
\$a7	23	0
\$t8	24	0
\$t9	25	0
\$k0	26	0
\$k1	27	0
\$gp	28	265465224
\$sp	29	2147479545
\$fp	30	0
\$ra	31	0
\$pc		4194332
\$t1		0
\$t0		0

PROGRAMA 6:

1 // programa 6 {
2 # x = o maior inteiro possivel; # o maior inteiro possivel é 0x7FFF FFFF
3 # y = 300000; Hexa = 0x0004 93E0
4 # z = x - 4y
5 # }
6
7 #associações
8 # x = t0
9 # y = t1
10 # z = t3
11
12 # O NUMERO DE CASAS PARA ANDAR, NESSE CASO, TEM QUE ESTAR EM BINARIO
13 .text
14 .globl main
15
16 main:
17 ori t0, t0, 0x7FFF # t0 = 0x0000 7FFF
18 sll t0, t0, 16 # estou andando 4 casas de bit para a esquerda, ou seja 2^4 = 16 --> 0x7FFF 0000 (2^4 = x)
19 # sll t0, t0, 0x0010 TANTO FAZ USAR ESSA OPÇÃO OU A DE CIMA, AS DUAS, NESSE CASO, ANDAM 4 BITS PARA A ESQUERDA
20 ori t0, t0, 0x7FFF
21
22 ori t1, t1, 0x0004 # t1 = 0x0000 0004
23 sll t1, t1, 0x0010 # t1 = 0x0004 0000 (16 bits ou também quatro casas para a esquerda)
24 ori t1, t1, 0x53E0 # t1 = 0x0004 93E0
25
26 sll t3, t3, 2 # 2 ^2 = 4, ou seja, 4*y
27 sub t3, t3, t0 # z = x - 4y
28
29
30

Line: 23 Column: 89 Show Line Numbers

Mars Messages Run IO

Registers Coproc 1 Coproc 0

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	0x7ffff000
\$t1	9	0x00040000
\$t2	10	0x00000000
\$t3	11	0x00124ff0
\$t4	12	0x00000000
\$t5	13	0x00000000
\$t6	14	0x00000000
\$t7	15	0x00000000
\$s0	16	0x7ffff000
\$s1	17	0x000453e0
\$s2	18	0x00000000
\$s3	19	0x7fedb07e
\$s4	20	0x00000000
\$s5	21	0x00000000
\$s6	22	0x00000000
\$s7	23	0x00000000
\$s8	24	0x00000000
\$s9	25	0x00000000
\$k0	26	0x00000000
\$k1	27	0x00000000
\$gp	28	0x10000000
\$fp	29	0x7ffff000
\$ra	30	0x00000000
\$ra	31	0x00000000
\$pc		0x00400028
\$hi		0x00000000
\$lo		0x00000000

Text Segment

Byte	Address	Code	Basic	Source
<input type="checkbox"/>	0x00400000	0x35087fff	ori t0,t0,0x00007fff	17: ori t0, t0, 0x7FFF # t0 = 0x0000 7FFF
<input type="checkbox"/>	0x00400004	0x00084400	sll t0,t0,16	18: sll t0, t0, 16 # estou andando 4 casas de bit para a esquerda, ou seja 2^4 = ...
<input type="checkbox"/>	0x00400008	0x03510fff	ori t1,t1,0x00000fff	20: ori t0, t0, 0x7FFF
<input type="checkbox"/>	0x0040000c	0x35290004	ori t1,t1,0x00000004	22: ori t1, t1, 0x0004 # t1 = 0x0000 0004
<input type="checkbox"/>	0x00400010	0x00094c00	sll t1,t1,0x00000010	23: sll t1, t1, 0x0010 # t1 = 0x0004 0000 (16 bits ou também quatro casas para a ...
<input type="checkbox"/>	0x00400014	0x353193e0	ori t1,t1,0x000093e0	24: ori t1, t1, 0x53E0 # t1 = 0x0004 93E0
<input type="checkbox"/>	0x00400018	0x00115800	sll t3,t3,2	26: sll t3, t3, 2 # 2 ^2 = 4, ou seja, 4*y
<input type="checkbox"/>	0x0040001c	0x020b9322	sub t3,t3,t0	27: sub t3, t0, t3 # z = x - 4y

Labels

Label	Address
(global)	
main	0x00400000

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x00000000	0x00000000	0x00000000	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
0x100100e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010100	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010120	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010140	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010160	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010180	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100101a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100101c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100101e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010200	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010220	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010240	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010260	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010280	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100102a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100102c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100102e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010300	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010320	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010340	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010360	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010380	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100103a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100103c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100103e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010400	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010420	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010440	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010460	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010480	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100104a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100104c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100104e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010500	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010520	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010540	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010560	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010580	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100105a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100105c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100105e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010600	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010620	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010640	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010660	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010680	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100106a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100106c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100106e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010700	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010720	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010740	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010760	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010780	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100107a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100107c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100107e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010800	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010820	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010840	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010860	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010880	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100108a0	0x00000000	0						

Text Segment

Btpt	Address	Code	Basic	Source
<input type="checkbox"/>	0x00400000	0x34080001	ori s8,s0,0x00000001	15: ori s8, s0, 0x0001 # s8 = 0x0000 0001
<input type="checkbox"/>	0x00400004	0x000847c0	all s8,s8,0x0000001f	18: all s8, s8, 0x001f # s8 = 0x1000 0000
<input type="checkbox"/>	0x00400008	0x000847c3	aza s8,s8,0x0000001f	19: aza s8, s8, 0x001f # s8 = 0xffff ffff

Labels

Label	Address
(global)	
main	0x00400000

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x00000000	0x00000000	0x00000000	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
0x100100e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010100	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010120	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010140	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010160	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010180	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100101a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100101c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100101e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100101f0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

0x10010000 (data)

Hexadecimal Addresses

Hexadecimal Values

ASCII

Mars Messages

Run IO

Registers	Coproc 1	Coproc 0
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	0xffffffff
\$t1	9	0x00000000
\$t2	10	0x00000000
\$t3	11	0x00000000
\$t4	12	0x00000000
\$t5	13	0x00000000
\$t6	14	0x00000000
\$t7	15	0x00000000
\$s0	16	0x00000000
\$s1	17	0x00000000
\$s2	18	0x00000000
\$s3	19	0x00000000
\$s4	20	0x00000000
\$s5	21	0x00000000
\$s6	22	0x00000000
\$s7	23	0x00000000
\$s8	24	0x00000000
\$s9	25	0x00000000
\$t0	26	0x00000000
\$k1	27	0x00000000
\$fp	28	0x10000000
\$sp	29	0xffffffff
\$fp	30	0x00000000
\$ra	31	0x00000000
pc		0x00400024
hi		0x00000000
lo		0x00000000

PROGRAMA 8:

Programa1.asm

Programa2.asm

Programa3.asm

Programa4.asm

Programa5.asm

Programa6.asm

Programa7.asm

Programa8.asm

```
1 // programa 8
2 # Inicialmente escreva um programa que faça:
3 # s8 = 0x12345678.
4 # A partir do registrador $t0 acima, usando apenas instruções lógicas (or, ori, and, andi,
5 # xor, xori) e instruções de deslocamento (sll, srl e sra), você deverá obter os seguintes
6 # valores nos respectivos registradores:
7 # s9 = 0x12
8 # t0 = 0x34
9 # t1 = 0x56
10 # t2 = 0x78
11
12 .text
13 .globl main
14
15 #main:
16 ori s8, s0, 0x1234 #s8 = 0x0000 1234
17 sll s8, s8, 16 # s7 = 4*4 casas para o lado = 16 bits, ou seja, 0x1234 0000
18 ori s8, s8, 0x5678 # s8 = 0x1234 5678
19
20 sll s9, s8, 24 # s9 = 0x0000 0012 --> 0x12
21
22 sll t0, s8, 8 # t0 = 0x3456 7800 (4*2 = 8)
23 sll t0, t0, 24 # t0 = 0x0000 0034 (4*6 = 24)
24
25 andi t1, s8, 0xFF00 # t1 = and entre 0x1234 5678 e 0x0000 FF00 = 0x0000 5600
26 sll t1, t1, 8 # t1 = 4*2 = 8 0x0000 0056
27
28 andi t2, s8, 0x00FF # t2 = and entre 0x1234 5678 e 0x0000 00FF = 0x0000 0078
```

Line: 28 Column: 80 Show Line Numbers

Mars Messages

Run IO

Registers	Coproc 1	Coproc 0
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	0x12345678
\$t1	9	0x00000012
\$t2	10	0x00000034
\$t3	11	0x00000056
\$t4	12	0x00000078
\$t5	13	0x00000000
\$t6	14	0x00000000
\$t7	15	0x00000000
\$s0	16	0x00000000
\$s1	17	0x00000000
\$s2	18	0x00000000
\$s3	19	0x00000000
\$s4	20	0x00000000
\$s5	21	0x00000000
\$s6	22	0x00000000
\$s7	23	0x00000000
\$s8	24	0x00000000
\$s9	25	0x00000000
\$k0	26	0x00000000
\$k1	27	0x00000000
\$fp	28	0x10000000
\$sp	29	0xffffffff
\$fp	30	0x00000000
\$ra	31	0x00000000
pc		0x00400024
hi		0x00000000
lo		0x00000000

Text Segment

Btpt	Address	Code	Basic	Source
<input type="checkbox"/>	0x00400000	0x34081234	ori s8,s0,0x00001234	16: ori s8, s0, 0x1234 #s8 = 0x0000 1234
<input type="checkbox"/>	0x00400004	0x00084400	sll s8,s8,16	17: sll s8, s8, 16 # s7 = 4*4 casas para o lado = 16 bits, ou seja, 0x1234 0000
<input type="checkbox"/>	0x00400008	0x56085678	ori s8,s8,0x00005678	18: ori s8, s8, 0x5678 # s8 = 0x1234 5678
<input type="checkbox"/>	0x0040000c	0x00084402	srl s9,s8,24	20: srl s9, s8, 24 # s9 = 0x0000 0012 --> 0x12
<input type="checkbox"/>	0x00400010	0x00085200	sll t0,s8,8	22: sll t0, s8, 8 # t0 = 0x3456 7800 (4*2 = 8)
<input type="checkbox"/>	0x00400014	0x00085602	sll t0,t0,24	23: sll t0, t0, 24 # t0 = 0x0000 0034 (4*6 = 24)
<input type="checkbox"/>	0x00400018	0x10bfff00	andi t1,s8,0x0000ff00	25: andi t1, s8, 0xFF00 # t1 = and entre 0x1234 5678 e 0x0000 FF00 = 0x0000 5600
<input type="checkbox"/>	0x0040001c	0x00084402	sll t1,t1,8	26: sll t1, t1, 8 # t1 = 4*2 = 8 0x0000 0056
<input type="checkbox"/>	0x00400020	0x31c000ff	andi t2,s8,0x000000ff	28: andi t2, s8, 0x00FF # t2 = and entre 0x1234 5678 e 0x0000 00FF = 0x0000 0078

Labels

Label	Address
(global)	
main	0x00400000

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x00000000	0x00000000	0x00000000	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
0x100100e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010100	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010120	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010140	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010160	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010180	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100101a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100101c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100101e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

0x10010000 (data)

Hexadecimal Addresses

Hexadecimal Values

ASCII

Mars Messages

Run IO

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	0x12345678
\$t1	9	0x00000012
\$t2	10	0x00000034
\$t3	11	0x00000056
\$t4	12	0x00000078
\$t5	13	0x00000000
\$t6	14	0x00000000
\$t7	15	0x00000000
\$s0	16	0x00000000
\$s1	17	0x00000000
\$s2	18	0x00000000
\$s3	19	0x00000000
\$s4	20	0x00000000
\$s5	21	0x00000000
\$s6	22	0x00000000
\$s7	23	0x00000000
\$s8	24	0x00000000
\$s9	25	0x00000000
\$k0	26	0x00000000
\$k1	27	0x00000000
\$fp	28	0x10000000
\$sp	29	0xffffffff
\$fp	30	0x00000000
\$ra	31	0x00000000
pc		0x00400024
hi		0x00000000
lo		0x00000000

PROGRAMA 9:

Programa1.asmPrograma2.asmPrograma3.asmPrograma4.asmPrograma5.asmPrograma6.asmPrograma7.asmPrograma8.asmPrograma9.asm

```
12 # x1 = MEM[0]
13 # x2 = MEM[1]
14 # x3 = MEM[2]
15 # x4 = MEM[3]
16 # soma -> MEM[4] = x1 + x2 + x3 + x4
17
18 #Associações
19 # x1 = $t0
20 # x2 = $t1
21 # x3 = $t2
22 # x4 = $t3
23 # soma = $t10
24
25 .text
26 .globl main
27
28 main:
29     lui $t0, 0x1001 # endereço 0x1001 é onde começa a memória no Mars 0x1001 0000
30     lw $t0, 0x0000 ($t0) # MEM[0]
31     lw $t1, 0x0004 ($t0) # MEM[1]
32     lw $t2, 0x0008 ($t0) # MEM[2]
33     lw $t3, 0x000C ($t0) # MEM[3]
34     add $t1, $t0, $t1 # t1 = x1 + x2
35     add $t1, $t1, $t2 # t1 = t1 + x3
36     add $t5, $t1, $t3 # soma = t1 + x4
37     sw $t5, 0x0010 ($t0) # MEM[4] = soma
38
39 .data
40 x1: .word 15
41 x2: .word 25
42 x3: .word 13
43 x4: .word 17
44 soma: .word -1
45
```

Line: 37 Column: 36 Show Line Numbers

Registers

	Name	Number	Value
\$zero		0	0
\$at		1	0
\$v0		2	0
\$v1		3	0
\$a0		4	0
\$a1		5	0
\$a2		6	0
\$a3		7	0
\$t0		8	268500992
\$t1		9	53
\$t2		10	0
\$t3		11	0
\$t4		12	0
\$t5		13	0
\$t6		14	0
\$t7		15	0
\$s0		16	15
\$s1		17	25
\$s2		18	13
\$s3		19	17
\$s4		20	0
\$s5		21	70
\$s6		22	0
\$s7		23	0
\$t8		24	0
\$t9		25	0
\$k0		26	0
\$k1		27	0
\$gp		28	268468224
\$fp		29	2147479546
\$fp		30	0
\$ra		31	0
pc			4194344
hi			0
lo			0

Text Segment

Addr	Address	Code	Source
0x00400000	0x00400000	lui \$t0, 0x1001	30: lui \$t0, 0x1001 # endereço 0x1001 é onde começa a memória no Mars 0x1001 0000
0x00400004	0x00400004	lw \$t0, 0x0000 (\$t0)	31: lw \$t0, 0x0000 (\$t0) # MEM[0]
0x00400008	0x00400008	lw \$t1, 0x0004 (\$t0)	32: lw \$t1, 0x0004 (\$t0) # MEM[1]
0x0040000c	0x0040000c	lw \$t2, 0x0008 (\$t0)	33: lw \$t2, 0x0008 (\$t0) # MEM[2]
0x00400010	0x00400010	lw \$t3, 0x000C (\$t0)	34: lw \$t3, 0x000C (\$t0) # MEM[3]
0x00400014	0x002114520	add \$t1, \$t0, \$t1	35: add \$t1, \$t0, \$t1 # t1 = x1 + x2
0x00400018	0x01324520	add \$t1, \$t1, \$t2	36: add \$t1, \$t1, \$t2 # t1 = t1 + x3
0x0040001c	0x013a520	add \$t5, \$t1, \$t3	37: add \$t5, \$t1, \$t3 # soma = t1 + x4
0x00400020	0x0ad150010	sw \$t5, 0x0010 (\$t0)	38: sw \$t5, 0x0010 (\$t0) # MEM[4] = soma

Labels

Label	Address
(global)	
main	0x00400000
Programa9.asm	
x1	0x10010000
x2	0x10010004
x3	0x10010008
x4	0x1001000c
soma	0x10010010

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	15	25	13	17	70	0	0	0
0x10010020	0	0	0	0	0	0	0	0
0x10010040	0	0	0	0	0	0	0	0
0x10010060	0	0	0	0	0	0	0	0
0x10010080	0	0	0	0	0	0	0	0
0x100100a0	0	0	0	0	0	0	0	0
0x100100c0	0	0	0	0	0	0	0	0
0x100100e0	0	0	0	0	0	0	0	0
0x10010100	0	0	0	0	0	0	0	0
0x10010120	0	0	0	0	0	0	0	0
0x10010140	0	0	0	0	0	0	0	0
0x10010160	0	0	0	0	0	0	0	0
0x10010180	0	0	0	0	0	0	0	0
0x100101a0	0	0	0	0	0	0	0	0
0x100101c0	0	0	0	0	0	0	0	0

0x10010000 (.data) Hexadecimal Addresses Hexadecimal Values ASCII

Mars MessagesRun IO

PROGRAMA 10:

Programa1.asmPrograma2.asmPrograma3.asmPrograma4.asmPrograma5.asmPrograma6.asmPrograma7.asmPrograma8.asmPrograma9.asmPrograma10.asm

```
5 # escrito, ou seja:
6 .data
7 # x: .word 5
8 # z: .word 7
9 # y: .word 0 # esse valor deverá ser sobrescrito após a execução do programa.
10
11 # Associações
12 # x = $t1
13 # z = $t2
14 # y = $t3
15
16 .text
17 .globl main
18
19 main:
20     lui $t0, 0x1001 # endereço 0x1001 é onde começa a memória no Mars 0x1001 0000
21     lw $t1, 0x0000 ($t0) # MEM[0]
22     lw $t2, 0x0004 ($t0) # MEM[1]
23
24     sll $t1, $t1, 7 # t1 = 128*x
25     sub $t1, $t1, $t1 # t1 = 128x - x = 127x
26
27     sll $t2, $t2, 6 # t2 = 64z
28     add $t2, $t2, $t2 # t2 = 64z + z
29
30     sub $t3, $t1, $t2 # t3 = 127x - 65z
31     addi $t3, $t3, 1 # t3 = 127x - 65z + 1
32
33     sw $t3, 0x0008 ($t0) # y -> MEM[2] = 127x - 65z + 1
34
35 .data
36 x: .word 5
37 z: .word 7
38 y: .word 0 # esse valor deverá ser sobrescrito após a execução do programa.
39
```

Line: 24 Column: 30 Show Line Numbers

Registers

	Name	Number	Value
\$zero		0	0
\$at		1	0
\$v0		2	0
\$v1		3	0
\$a0		4	0
\$a1		5	0
\$a2		6	0
\$a3		7	0
\$t0		8	268500992
\$t1		9	635
\$t2		10	655
\$t3		11	180
\$t4		12	0
\$t5		13	0
\$t6		14	0
\$t7		15	0
\$s0		16	0
\$s1		17	5
\$s2		18	7
\$s3		19	183
\$s4		20	0
\$s5		21	0
\$s6		22	0
\$s7		23	0
\$t8		24	0
\$t9		25	0
\$k0		26	0
\$k1		27	0
\$gp		28	268468224
\$fp		29	2147479546
\$fp		30	0
\$ra		31	0
pc			4194344
hi			0
lo			0

Mars MessagesRun IO

Interface showing assembly code and registers. The assembly code includes instructions like `lui $t0, 0x1001`, `lw $a0, 0x0000($t0)`, and `sw $a0, 0x0008($t0)`. The registers panel shows values for `$zero`, `$at`, `$v0`, `$a0`, `$a1`, `$a2`, `$a3`, `$t0`, `$t1`, `$t2`, `$t3`, `$t4`, `$t5`, `$t6`, `$t7`, `$s0`, `$s1`, `$s2`, `$s3`, `$s4`, `$s5`, `$s6`, `$s7`, `$s8`, `$s9`, `$ra`, `$sp`, `$fp`, `$ra`, `$pc`, `$hi`, and `$lo`.

PROGRAMA 11:

Interface showing assembly code and registers. The assembly code includes instructions like `lui $t0, 0x1001`, `lw $a0, 0x0000($t0)`, and `sw $a0, 0x0008($t0)`. The registers panel shows values for `$zero`, `$at`, `$v0`, `$a0`, `$a1`, `$a2`, `$a3`, `$t0`, `$t1`, `$t2`, `$t3`, `$t4`, `$t5`, `$t6`, `$t7`, `$s0`, `$s1`, `$s2`, `$s3`, `$s4`, `$s5`, `$s6`, `$s7`, `$s8`, `$s9`, `$ra`, `$sp`, `$fp`, `$ra`, `$pc`, `$hi`, and `$lo`.

Interface showing assembly code and registers. The assembly code includes instructions like `lui $t0, 0x1001`, `lw $a0, 0x0000($t0)`, and `sw $a0, 0x0008($t0)`. The registers panel shows values for `$zero`, `$at`, `$v0`, `$a0`, `$a1`, `$a2`, `$a3`, `$t0`, `$t1`, `$t2`, `$t3`, `$t4`, `$t5`, `$t6`, `$t7`, `$s0`, `$s1`, `$s2`, `$s3`, `$s4`, `$s5`, `$s6`, `$s7`, `$s8`, `$s9`, `$ra`, `$sp`, `$fp`, `$ra`, `$pc`, `$hi`, and `$lo`.

PROGRAMA 12:

4 # onde x contem um ponteiro para um ponteiro para um ponteiro para um inteiro.
5 # Nessa situação, considere que a posição inicial de memória contenha o inteiro em questão.
6 # Coloque todos os outros valores em registradores, use os endereços de memória que quiser dentro
7 # do espaço de endereçamento do Mips.
8 # Resumo do problema:
9 # k = MEM [MEM [MEM [x]]].
10 # Crie um programa que implemente a estrutura de dados acima, leia o valor de K, o multiplique por
11 # 2 e o reescreva no local correto começando-se apenas o valor de x.
12
13 # {
14 # int ***x;
15 # k = MEM [MEM [MEM [x]]];
16 # k = 2*k
17 # MEM [MEM [MEM [x]]] = k
18 # }
19
20 # Associações
21 # x = \$s0
22 # k = \$s1
23
24 .text
25 .globl main
26
27 main:
28 lui \$s0, 0x1001 # x = 0x1001 0000
29 lw \$t0, 0x0000(\$s0) # t0 = MEM[x]
30 lw \$t1, 0x0000(\$t0) # t1 = MEM[MEM[x]]
31 lw \$s1, 0x0000(\$t1) # k = MEM [MEM [MEM [x]]]
32 sll \$s1, \$s1, 1 # k = 2 * k (2^1 = 2)
33 sw \$s1, 0x0000(\$t1) # MEM [MEM [MEM [x]]] = k, sendo k = 2*k
34
35 .data
36 x: .word 0x10010004
37 pont_x: .word 0x10010008
38 pont_pont_x: .word 0
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
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
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166

Text Segment

Byte	Address	Code	Basic	Source
	0x00400000	0x3c081001	lui \$8,4097	16: lui \$t0, 0x1001 # endereço 0x1001 é onde começa a memória no m30 0x1001 0000
	0x00400004	0x8d100000	lw \$16,0(\$8)	16: lw \$t0, 0x0000(\$t0) # a = MEM[0]
	0x00400008	0x00104fc2	srl \$9,\$16,31	17: srl \$t1, \$t0, 0x001f # metodo para identificar se o numero é positivo ou nao, v...
	0x0040000c	0x11200002	breq \$9,\$0,2	18: breq \$t1, \$zero, fim # if(a > 0) o programa acaba
	0x00400010	0x00108022	sub \$16,\$0,\$16	19: sub \$t0, \$zero, \$t0 #else(a = 1a1, ou seja, vai inverter o sinal dele
	0x00400014	0xad100000	sw \$16,0(\$8)	20: sw \$t0, 0(\$t0) # MEM[0] =

Labels

Label	Address
(global)	
main	0x00400000
Programa13.asm	
fim	0x00400018
a	0x10010000

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+C)	Value (+10)	Value (+14)	Value (+18)	Value (+1C)
0x10010000	5	0	0	0	0	0	0	0
0x10010004	0	0	0	0	0	0	0	0
0x10010008	0	0	0	0	0	0	0	0
0x1001000c	0	0	0	0	0	0	0	0
0x10010010	0	0	0	0	0	0	0	0
0x10010014	0	0	0	0	0	0	0	0
0x10010018	0	0	0	0	0	0	0	0
0x1001001c	0	0	0	0	0	0	0	0
0x10010020	0	0	0	0	0	0	0	0
0x10010024	0	0	0	0	0	0	0	0
0x10010028	0	0	0	0	0	0	0	0
0x1001002c	0	0	0	0	0	0	0	0
0x10010030	0	0	0	0	0	0	0	0
0x10010034	0	0	0	0	0	0	0	0
0x10010038	0	0	0	0	0	0	0	0
0x1001003c	0	0	0	0	0	0	0	0
0x10010040	0	0	0	0	0	0	0	0
0x10010044	0	0	0	0	0	0	0	0
0x10010048	0	0	0	0	0	0	0	0
0x1001004c	0	0	0	0	0	0	0	0
0x10010050	0	0	0	0	0	0	0	0
0x10010054	0	0	0	0	0	0	0	0
0x10010058	0	0	0	0	0	0	0	0
0x1001005c	0	0	0	0	0	0	0	0
0x10010060	0	0	0	0	0	0	0	0
0x10010064	0	0	0	0	0	0	0	0
0x10010068	0	0	0	0	0	0	0	0
0x1001006c	0	0	0	0	0	0	0	0
0x10010070	0	0	0	0	0	0	0	0
0x10010074	0	0	0	0	0	0	0	0
0x10010078	0	0	0	0	0	0	0	0
0x1001007c	0	0	0	0	0	0	0	0
0x10010080	0	0	0	0	0	0	0	0
0x10010084	0	0	0	0	0	0	0	0
0x10010088	0	0	0	0	0	0	0	0
0x1001008c	0	0	0	0	0	0	0	0
0x10010090	0	0	0	0	0	0	0	0
0x10010094	0	0	0	0	0	0	0	0
0x10010098	0	0	0	0	0	0	0	0
0x1001009c	0	0	0	0	0	0	0	0
0x100100a0	0	0	0	0	0	0	0	0
0x100100a4	0	0	0	0	0	0	0	0
0x100100a8	0	0	0	0	0	0	0	0
0x100100ac	0	0	0	0	0	0	0	0
0x100100b0	0	0	0	0	0	0	0	0
0x100100b4	0	0	0	0	0	0	0	0
0x100100b8	0	0	0	0	0	0	0	0
0x100100bc	0	0	0	0	0	0	0	0
0x100100c0	0	0	0	0	0	0	0	0
0x100100c4	0	0	0	0	0	0	0	0
0x100100c8	0	0	0	0	0	0	0	0
0x100100cc	0	0	0	0	0	0	0	0
0x100100d0	0	0	0	0	0	0	0	0
0x100100d4	0	0	0	0	0	0	0	0
0x100100d8	0	0	0	0	0	0	0	0
0x100100dc	0	0	0	0	0	0	0	0
0x100100e0	0	0	0	0	0	0	0	0
0x100100e4	0	0	0	0	0	0	0	0
0x100100e8	0	0	0	0	0	0	0	0
0x100100ec	0	0	0	0	0	0	0	0
0x100100f0	0	0	0	0	0	0	0	0
0x100100f4	0	0	0	0	0	0	0	0
0x100100f8	0	0	0	0	0	0	0	0
0x100100fc	0	0	0	0	0	0	0	0
0x10010100	0	0	0	0	0	0	0	0
0x10010104	0	0	0	0	0	0	0	0
0x10010108	0	0	0	0	0	0	0	0
0x1001010c	0	0	0	0	0	0	0	0
0x10010110	0	0	0	0	0	0	0	0
0x10010114	0	0	0	0	0	0	0	0
0x10010118	0	0	0	0	0	0	0	0
0x1001011c	0	0	0	0	0	0	0	0
0x10010120	0	0	0	0	0	0	0	0
0x10010124	0	0	0	0	0	0	0	0
0x10010128	0	0	0	0	0	0	0	0
0x1001012c	0	0	0	0	0	0	0	0
0x10010130	0	0	0	0	0	0	0	0
0x10010134	0	0	0	0	0	0	0	0
0x10010138	0	0	0	0	0	0	0	0
0x1001013c	0	0	0	0	0	0	0	0
0x10010140	0	0	0	0	0	0	0	0
0x10010144	0	0	0	0	0	0	0	0
0x10010148	0	0	0	0	0	0	0	0
0x1001014c	0	0	0	0	0	0	0	0
0x10010150	0	0	0	0	0	0	0	0
0x10010154	0	0	0	0	0	0	0	0
0x10010158	0	0	0	0	0	0	0	0
0x1001015c	0	0	0	0	0	0	0	0
0x10010160	0	0	0	0	0	0	0	0
0x10010164	0	0	0	0	0	0	0	0
0x10010168	0	0	0	0	0	0	0	0
0x1001016c	0	0	0	0	0	0	0	0
0x10010170	0	0	0	0	0	0	0	0
0x10010174	0	0	0	0	0	0	0	0
0x10010178	0	0	0	0	0	0	0	0
0x1001017c	0	0	0	0	0	0	0	0
0x10010180	0	0	0	0	0	0	0	0
0x10010184	0	0	0	0	0	0	0	0
0x10010188	0	0	0	0	0	0	0	0
0x1001018c	0	0	0	0	0	0	0	0
0x10010190	0	0	0	0	0	0	0	0
0x10010194	0	0	0	0	0	0	0	0
0x10010198	0	0	0	0	0	0	0	0
0x1001019c	0	0	0	0	0	0	0	0
0x100101a0	0	0	0	0	0	0	0	0
0x100101a4	0	0	0	0	0	0	0	0
0x100101a8	0	0	0	0	0	0	0	0
0x100101ac	0	0	0	0	0	0	0	0
0x100101b0	0	0	0	0	0	0	0	0
0x100101b4	0	0	0	0	0	0	0	0
0x100101b8	0	0	0	0	0	0	0	0
0x100101bc	0	0	0	0	0	0	0	0
0x100101c0	0	0	0	0	0	0	0	0
0x100101c4	0	0	0	0	0	0	0	0
0x100101c8	0	0	0	0	0	0	0	0
0x100101cc	0	0	0	0	0	0	0	0
0x100101d0	0	0	0	0	0	0	0	0
0x100101d4	0	0	0	0	0	0	0	0
0x100101d8	0	0	0	0	0	0	0	0
0x100101dc	0	0	0	0	0	0	0	0
0x100101e0	0	0	0	0	0	0	0	0
0x100101e4	0	0	0	0	0	0	0	0
0x100101e8	0	0	0	0	0	0	0	0
0x100101ec	0	0	0	0	0	0	0	0
0x100101f0	0	0	0	0	0	0	0	0
0x100101f4	0	0	0	0	0	0	0	0
0x100101f8	0	0	0	0	0	0	0	0
0x100101fc	0	0	0	0	0	0	0	0
0x10010200	0	0	0	0	0	0	0	0
0x10010204	0	0	0	0	0	0	0	0
0x10010208	0	0	0	0	0	0	0	0
0x1001020c	0	0	0	0	0	0	0	0
0x10010210	0	0	0	0	0	0	0	0
0x10010214	0	0	0	0	0	0	0	0
0x10010218	0	0	0	0	0	0	0	0
0x1001021c	0	0	0	0	0	0	0	0
0x10010220	0	0	0	0	0	0	0	0
0x10010224	0	0	0	0	0	0	0	0
0x10010228	0	0	0	0	0	0	0	0
0x1001022c	0	0	0	0	0	0	0	0
0x10010230	0	0	0	0	0	0	0	0
0x10010234	0	0	0	0	0	0	0	0
0x10010238	0	0	0	0	0	0	0	0
0x1001023c	0	0	0	0	0	0	0	0
0x10010240	0	0	0	0	0	0	0	0
0x10010244	0	0	0	0	0	0	0	0
0x10010248	0	0	0	0	0	0	0	0
0x1001024c	0	0	0	0	0	0	0	0
0x10010250	0	0	0	0	0	0	0	0
0x10010254	0	0	0	0	0	0	0	0
0x10010258	0	0	0	0	0	0	0	0
0x1001025c	0	0	0	0	0	0	0	0
0x10010260	0	0	0	0	0	0	0	0
0x10010264	0	0	0	0	0	0	0	0
0x10010268	0	0	0	0	0	0	0	0
0x1001026c	0	0	0	0	0	0	0	0
0x10010270	0	0	0	0	0	0	0	0
0x10010274	0	0	0	0	0	0	0	0
0x10010278	0	0	0	0	0	0	0	0
0x1001027c	0	0	0	0	0	0	0	0
0x10010280	0	0	0	0	0	0	0	0
0x10010284	0	0	0	0	0	0	0	0
0x10010288	0	0	0	0	0	0	0	0
0x1001028c	0	0	0	0	0	0	0	0
0x10010290	0	0	0	0	0	0	0	0
0x10010294	0	0	0	0	0	0	0	0
0x10010298	0	0	0	0	0	0	0	0
0x1001029c	0	0	0	0	0	0	0	0
0x100102a0	0	0	0	0	0	0	0	0
0x100102a4	0	0	0	0	0	0	0	0
0x100102a8	0	0	0	0	0	0	0	0
0x100102ac	0	0	0	0	0	0	0	0
0x100102b0	0	0	0					

PROGRAMA 15:

EditExecute

Programa15.asm

```
# )
# Associações
# V[0] = $a0
# soma = $a1
# i = $a2

# 100 em hexadecimal = 0x64
.text
.globl main

main:
    lui $a0, 0x1001 # endereço 0x1001 é onde começa a memória no mips 0x1001 0000, colocando o vetor na sua primeira posição
    # addi $a1, $0, $0 # soma = 0
    # addi $a2, $zero, $zero # i = 0
    or $a1, $zero, $zero # soma = 0
    or $a2, $zero, $zero # i = 0
    ori $t3, $zero, 0x0064 # t3 = 100

loop:
    sll $t1, $a2, 1 # t1 = i * 2, sendo esse 2 2^1 = 2
    addi $t1, $t1, 1 # t1 = 2^1 + 1
    sll $t2, $a2, 2 # t2 = 4^1
    add $t2, $a0, $t2 # t2 = v[i] // temos que multiplicar por quatro para colocar, por exemplo, dentro do vetor
    sw $t1, 0x0000($t2) # MEM[i] = 2^i + 1, ou seja, t2 que valei 4i vai para memória, MEM[i]
    add $a1, $a1, $t1 # soma = soma + 2^i + 1
    addi $a2, $a2, 1 # i++
    bne $a2, $t3, loop # if (i < 100) goto loop, ou seja, enquanto o i for diferente de 100, ele vai continuar executando o código
    sw $a1, 0x0004($t2) # MEM[100] = soma

#fim do loop
```

Line: 1 Column: 1 Show Line Numbers

Mars MessagesRun IO

RegistersCoproccoproc 0

Name	Number	Value
\$zero	0	0
\$at	1	0
\$v0	2	0
\$v1	3	0
\$a0	4	0
\$a1	5	0
\$a2	6	0
\$a3	7	0
\$t0	8	0
\$t1	9	199
\$t2	10	268501388
\$t3	11	100
\$t4	12	0
\$t5	13	0
\$t6	14	0
\$t7	15	0
\$s0	16	268500992
\$a1	17	10000
\$a2	18	100
\$a3	19	0
\$a4	20	0
\$a5	21	0
\$a6	22	0
\$a7	23	0
\$t8	24	0
\$t9	25	0
\$k0	26	0
\$k1	27	0
\$gp	28	268468224
\$sp	29	2147479540
\$fp	30	0
\$ra	31	0
pc		4194356
hi		0
lo		0

EditExecute

Text Segment

Offset	Address	Code	Basic	Source
0x00400000	0x3c101001	lui \$16,4097	25:	lui \$a0, 0x1001 # endereço 0x1001 é onde começa a memória no mips 0x1001 0000, colocando o vetor na sua primeira posição
0x00400004	0x00008250	or \$17,\$0,\$0	26:	or \$a1, \$zero, \$zero # soma = 0
0x00400008	0x00009025	or \$18,\$0,\$0	29:	or \$a2, \$zero, \$zero # i = 0
0x0040000c	0x340b0064	ori \$11,\$0,100	30:	ori \$t3, \$zero, 0x0064 #t3 = 100
0x00400010	0x00124800	sll \$9,\$16,1	33:	sll \$t1, \$a2, 1 # t1 = i * 2, sendo esse 2 2^1 = 2
0x00400014	0x21290001	addi \$9,\$9,1	34:	addi \$t1, \$t1, 1 # t1 = 2^1 + 1
0x00400018	0x00125080	sll \$10,\$16,2	35:	sll \$t2, \$a2, 2 # t2 = 4^1
0x0040001c	0x020a5020	add \$10,\$16,\$10	36:	add \$t2, \$a0, \$t2 # t2 = v[i] // temos que multiplicar por quatro para colocar, por exemplo, dentro do vetor
0x00400020	0x04d80000	sw \$9,\$0(\$10)	37:	sw \$t1, 0x0000(\$t2) # MEM[i] = 2^i + 1, ou seja, t2 que valei 4i vai para memória, MEM[i]
0x00400024	0x02298201	addi \$17,\$17,\$9	38:	addi \$a1, \$a1, \$t1 # soma = soma + 2^i + 1
0x00400028	0x02520001	addi \$18,\$18,1	39:	addi \$a2, \$a2, 1 # i++
0x0040002c	0x14b0ff0b	bne \$18,\$11,-8	40:	bne \$a2, \$t3, loop # if (i < 100) goto loop, ou seja, enquanto o i for diferente de 100, ele vai continuar executando o código
0x00400030	0x04d10004	sw \$17,\$0(\$10)	41:	sw \$a1, 0x0004(\$t2) # MEM[100] = soma

0x10010000 (.data)

☒ Hexadecimal Addresses ☐ Hexadecimal Values ☐ ASCII

RegistersCoproccoproc 0

Name	Number	Value
\$zero	0	0
\$at	1	0
\$v0	2	0
\$v1	3	0
\$a0	4	0
\$a1	5	0
\$a2	6	0
\$a3	7	0
\$t0	8	0
\$t1	9	199
\$t2	10	268501388
\$t3	11	100
\$t4	12	0
\$t5	13	0
\$t6	14	0
\$t7	15	0
\$s0	16	268500992
\$a1	17	10000
\$a2	18	100
\$a3	19	0
\$a4	20	0
\$a5	21	0
\$a6	22	0
\$a7	23	0
\$t8	24	0
\$t9	25	0
\$k0	26	0
\$k1	27	0
\$gp	28	268468224
\$sp	29	2147479540
\$fp	30	0
\$ra	31	0
pc		4194356
hi		0
lo		0

Instruction Statistics, Version 1.0 (Ingo Kofler)

Total: 805

ALU: 604 75%

Jump: 0 0%

Branch: 100 12%

Memory: 101 12%

Other: 0 0%

Tool Control

Disconnect from MIPSResetClose

PROGRAMA 16:

Programa15.asm

Programa16.asm

```
# Escreva um programa que avalie a expressão: (x*y)/z.
# Use x = 1600000 (=0x018 6A00), y = 80000 (=0x13980), e z = 400000 (=0x61A80). Inicializar os
# registradores com os valores acima.

# x = 1600000 (=0x018 6A00)
# y = 80000 (=0x0001 3980)
# z = 400000 (=0x0006 1A80)
# conta = (x*y)/z

# Associações
# x = $s0
# y = $s1
# z = $s2
# conta = $s3

.text
.global main

main:
    lui $t0, 0x0018 # t0 = 0x0018 0000
    ori $s0, $t0, 0x6A00 # x = 0x0018 6A00 --> 1600000
    lui $t1, 0x0001 # t1 = 0x0001 0000
    ori $s1, $t1, 0x3980 # y = 0x0001 3980 --> 80000
    lui $t2, 0x0006 # t2 = 0x0006 0000
    ori $s2, $t2, 0x1A80 # z = 0x0006 1A80
    div $s0, $s2 # x/z
    mflo $t0 # t0 = x/z
    mult $s1, $t0 # y * x/z
    mflo $s3 # conta = (x*y)/z
```

Registers

Coproc 1

Coproc 0

Name	Number	Value
\$zero	0	0
\$at	1	0
\$v0	2	0
\$v1	3	0
\$a0	4	0
\$a1	5	0
\$a2	6	0
\$a3	7	0
\$t0	8	4
\$t1	9	65536
\$t2	10	393216
\$t3	11	0
\$t4	12	0
\$t5	13	0
\$t6	14	0
\$t7	15	0
\$s0	16	1600000
\$s1	17	80000
\$s2	18	400000
\$s3	19	320000
\$s4	20	0
\$s5	21	0
\$s6	22	0
\$s7	23	0
\$t8	24	0
\$t9	25	0
\$k0	26	0
\$k1	27	0
\$gp	28	268469224
\$fp	29	2147479545
\$tp	30	0
\$ra	31	0
\$pc		4194344
\$hi		0
\$lo		320000

Line: 29 Column: 28

Show Line Numbers

Mars Messages

Run IO

Text Segment

Byte	Address	Code	Basic	Source
<input type="checkbox"/>	0x00400000	0x3c080018	lui \$t0, 20	lui \$t0, 0x0018 # t0 = 0x0018 0000
<input type="checkbox"/>	0x00400004	0x35106a00	ori \$s0, 21	ori \$s0, \$t0, 0x6A00 # x = 0x0018 6A00 --> 1600000
<input type="checkbox"/>	0x00400008	0x3c090001	lui \$t1, 22	lui \$t1, 0x0001 # t1 = 0x0001 0000
<input type="checkbox"/>	0x0040000c	0x35113980	ori \$s1, 23	ori \$s1, \$t1, 0x3980 # y = 0x0001 3980 --> 80000
<input type="checkbox"/>	0x00400010	0x3c0a0006	lui \$t2, 24	lui \$t2, 0x0006 # t2 = 0x0006 0000
<input type="checkbox"/>	0x00400014	0x35521a80	ori \$s2, 25	ori \$s2, \$t2, 0x1A80 # z = 0x0006 1A80
<input type="checkbox"/>	0x00400018	0x0212001a	div \$s0, 26	div \$s0, \$s2 # x/z
<input type="checkbox"/>	0x0040001c	0x00004012	mflo \$t0, 27	mflo \$t0 # t0 = x/z
<input type="checkbox"/>	0x00400020	0x02280018	mult \$s1, 28	mult \$s1, \$t0 # y * x/z
<input type="checkbox"/>	0x00400024	0x00009812	mflo \$s3, 29	mflo \$s3 # conta = (x*y)/z

Labels

Label	Address
(global)	
main	0x00400000

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0	0	0	0	0	0	0	0
0x10010020	0	0	0	0	0	0	0	0
0x10010040	0	0	0	0	0	0	0	0
0x10010060	0	0	0	0	0	0	0	0
0x10010080	0	0	0	0	0	0	0	0
0x100100a0	0	0	0	0	0	0	0	0
0x100100c0	0	0	0	0	0	0	0	0
0x100100e0	0	0	0	0	0	0	0	0
0x10010100	0	0	0	0	0	0	0	0
0x10010120	0	0	0	0	0	0	0	0
0x10010140	0	0	0	0	0	0	0	0
0x10010160	0	0	0	0	0	0	0	0
0x10010180	0	0	0	0	0	0	0	0
0x100101a0	0	0	0	0	0	0	0	0
0x100101c0	0	0	0	0	0	0	0	0
0x100101e0	0	0	0	0	0	0	0	0

0x10010000 (data)

☒ Hexadecimal Addresses

☐ Hexadecimal Values

☐ ASCII

Line: 36 Column: 11

Show Line Numbers

Mars Messages

Run IO

PROGRAMA 17:

Programa15.asm

Programa16.asm

Programa17.asm

```
# k = x * y (Você deverá realizar a multiplicação através de somas!)
# O valor de x deve ser lido da primeira posição livre da memória e o valor de y deverá lido da
# segunda posição livre. O valor de k, após calculado, deverá ainda ser escrito na terceira posição
# livre da memória.

# {
#   x = MEM[0]
#   y = MEM[1]
#   k = x * y
#   MEM[2] = k
# }

# Associações
# x = $s0
# y = $s1
# k = $s2

.text
.global main

main:
    lui $t0, 0x1001 # endereço 0x1001 é onde começa a memória no Mars 0x1001 0000
    lw $s0, 0x0000($t0) # x = MEM[0]
    lw $s1, 0x0004($t0) # y = MEM[1]

multip:
    add $t1, $t1, $s0 # t1 = t1 + x
    addi $t2, $t2, 1 # contador++
    bne $t2, $s1, multip # se (contador != y) goto multip
    add $s2, $s2, $t1 # k = x * y
    sw $s2, 0x0008($t0) # MEM[2] = k

.data
x: .word 5
y: .word 5
```

Registers

Coproc 1

Coproc 0

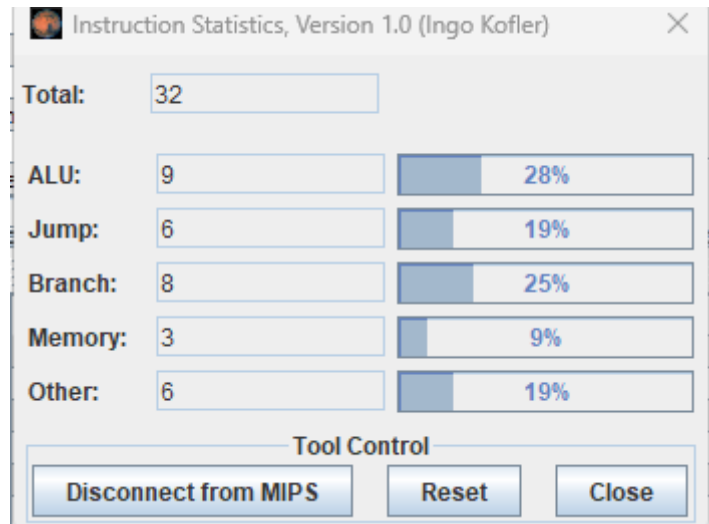
Name	Number	Value
\$zero	0	0
\$at	1	0
\$v0	2	0
\$v1	3	0
\$a0	4	0
\$a1	5	0
\$a2	6	0
\$a3	7	0
\$t0	8	268500992
\$t1	9	25
\$t2	10	5
\$t3	11	0
\$t4	12	0
\$t5	13	0
\$t6	14	0
\$t7	15	0
\$s0	16	5
\$s1	17	5
\$s2	18	25
\$s3	19	0
\$s4	20	0
\$s5	21	0
\$s6	22	0
\$s7	23	0
\$t8	24	0
\$t9	25	0
\$k0	26	0
\$k1	27	0
\$gp	28	268469224
\$fp	29	2147479545
\$tp	30	0
\$ra	31	0
\$pc		4194336
\$hi		0
\$lo		0

Line: 36 Column: 11

Show Line Numbers

Mars Messages

Run IO



Para os programas a seguir use instruções mult, div, mflo e mfhi.

Responda

- 1) C
- 2) B
- 3) A
- 4) C
- 5) B
- 6) A
- 7) D
- 8) A
- 9) A
- 10) A

PROGRAMA 19:

Edit Execute

Programa19.asm Programa20.asm Programa21.asm Programa15.asm Programa17.asm

```

.text
.globl main

main:
    lui $t0, 0x1001 # endereço 0x1001 é onde começa a memória no MARS 0x1001 0000
    lw $a0, 0 ($t0) # x = MEM[0]
    lw $a1, 4 ($t0) # y = MEM[1]

    ori $t0, $zero, 0 # contador X
    ori $t1, $zero, 0 # contador Y
    or $t2, $zero, $a0 # copia de s2
    or $t3, $zero, $a1 # copia de s1

contaBitsX:
    beq $t2, $zero, contaBitsY # verifica se o numero eh igual a 0 (nao ha mais bits significativos)
    addi $t0, $t0, 1 # add 1 no contador de bits
    sli $t2, $t2, 1 # desloca bits para direita
    j contaBitsX

contaBitsY:
    beq $t3, $zero, multiplicacao # verifica se o numero eh igual a 0 (nao ha mais bits significativos)
    addi $t1, $t1, 1 # add 1 no contador de bits
    sli $t3, $t3, 1 # desloca bits para direita
    j contaBitsY

multiplicacao:
    sli $t0, $t0, 32 # verifica se a qtd de bits do primeiro numero eh menor que 32
    beq $t0, $zero, multiplicacaoHL # se for maior (ou igual) ja vai pro tipo de multiplicacao que salva em HI e LO
    sli $t1, $t1, 32 # verifica se a qtd de bits do segundo numero eh menor que 32
    beq $t1, $zero, multiplicacaoHL # se for maior (ou igual) vai pro tipo de multiplicacao que salva em HI e LO

    mult $a0, $a1 # se chegou ate aqui ambos numeros sao menores que 32 bits
    mflo $s2 # faz multiplicacao salvando apenas em LO
    j fim

multiplicacaoHL:
    mult $a0, $a1 # multiplica e depois busca do HI e LO
    mfhi $s2
    mfhi $s3

fim:

.data
x: .word -10
y: .word 10

```

Line: 50 Column: 27 Show Line Numbers

Registers Coproc 1 Coproc 0

Name	Number	Value
\$zero	0	0
\$at	1	0
\$v0	2	0
\$v1	3	0
\$a0	4	0
\$a1	5	0
\$a2	6	0
\$a3	7	0
\$t0	8	0
\$t1	9	4
\$t2	10	0
\$t3	11	0
\$t4	12	0
\$t5	13	0
\$t6	14	0
\$t7	15	0
\$a0	16	-10
\$a1	17	10
\$a2	18	-100
\$a3	19	-5
\$a4	20	0
\$a5	21	0
\$a6	22	0
\$a7	23	0
\$a8	24	0
\$a9	25	0
\$a10	26	0
\$a11	27	0
\$gp	28	268469224
\$sp	29	2147479545
\$fp	30	0
\$ra	31	0
\$pc		4194404
\$hi		-1
\$lo		-100

Text Segment

Btpt	Address	Code	Basic	Source
	0x00400000	0x3c081001	lui \$t0,\$8,4097	18: lui \$t0, 0x1001 # endereco 0x1001 é onde começa a memoria no m32 0x1001 0000
	0x00400004	0x8d100000	lw \$t6,0(\$t0)	19: lw \$t6, 0 (\$t0) # x = MEM[0]
	0x00400008	0x8d100004	lw \$t1,\$4(\$t0)	20: lw \$t1, 4 (\$t0) # y = MEM[4]
	0x0040000c	0x34000000	ori \$t0,\$0,0	22: ori \$t0, \$zero, 0 # contador X
	0x00400010	0x34000000	ori \$t1,\$0,0	23: ori \$t1, \$zero, 0 # contador Y
	0x00400014	0x01005025	or \$t0,\$t0,\$t6	24: or \$t0, \$zero, \$t0 # copia de \$t6
	0x00400018	0x00118285	or \$t3,\$t0,\$t1	25: or \$t3, \$zero, \$t1 # copia de \$t1
	0x0040001c	0x11400003	beq \$t2,\$zero,\$t3	26: beq \$t2, \$zero, contaBitsX # verifica se o numero eh igual a 0 (nao ha mais bi...
	0x00400020	0x21080001	addi \$t0,\$t0,1	29: addi \$t0, \$t0, 1 # add 1 no contador de bits
	0x00400024	0x000a5042	srl \$t2,\$t2,1	30: srl \$t2, \$t2, 1 # desloca bits para direita
	0x00400028	0x00100007	j contaBitsX	31: j contaBitsX
	0x0040002c	0x29290020	ori \$t1,\$t1,\$t6	34: ori \$t1, \$t1, \$t6 # verifica se o numero eh igual a 0 (nao ha mais ...
	0x00400030	0x21290001	addi \$t1,\$t1,1	35: addi \$t1, \$t1, 1 # add 1 no contador de bits
	0x00400034	0x000b5842	srl \$t3,\$t3,1	36: srl \$t3, \$t3, 1 # desloca bits para direita
	0x00400038	0x00100007	j contaBitsY	37: j contaBitsY
	0x0040003c	0x29290020	ori \$t1,\$t1,\$t6	40: ori \$t1, \$t1, \$t6 # verifica se a qtd de bits do primeiro numero...
	0x00400040	0x11000003	beq \$t0,\$zero,\$t3	41: beq \$t0, \$zero, multiplicacaoHL # se for maior (ou igual) ja vai pro tipo de M...
	0x00400044	0x29290020	srl \$t1,\$t1,\$t3	42: srl \$t1, \$t1, \$t3 # verifica se a qtd de bits do segundo numero ...
	0x00400048	0x11000003	beq \$t1,\$zero,\$t3	43: beq \$t1, \$zero, multiplicacaoHL # se for maior (ou igual) vai pro tipo de mult...
	0x0040004c	0x02110018	mult \$t0,\$t1	45: mult \$t0, \$t1 # se chegou ate aqui ambos numeros sao menores que 32 ...
	0x00400050	0x00009012	mflw \$t2	46: mflw \$t2 # fat Multiplicacao salvando apenas em t2
	0x00400054	0x08100019	j fim	47: j fim
	0x00400058	0x02110018	mult \$t0,\$t1	50: mult \$t0, \$t1 # multiplica e depois busca do HI e LO
	0x0040005c	0x00009012	mflw \$t2	51: mflw \$t2
	0x00400060	0x00009810	mflh \$t3	52: mflh \$t3

Labels

Label	Address
(global)	
main	0x00400000
contaBitsX	0x0040001c
contaBitsY	0x0040002c
multiplicacao	0x0040003c
multiplicacaoHL	0x00400058
fin	0x00400064
t	0x10010000
7	0x10010004

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	-10	10	0	0	0	0	0	0
0x10010004	0	0	0	0	0	0	0	0
0x10010008	0	0	0	0	0	0	0	0
0x1001000c	0	0	0	0	0	0	0	0
0x10010010	0	0	0	0	0	0	0	0
0x10010014	0	0	0	0	0	0	0	0
0x10010018	0	0	0	0	0	0	0	0
0x1001001c	0	0	0	0	0	0	0	0
0x10010020	0	0	0	0	0	0	0	0
0x10010024	0	0	0	0	0	0	0	0
0x10010028	0	0	0	0	0	0	0	0
0x1001002c	0	0	0	0	0	0	0	0
0x10010030	0	0	0	0	0	0	0	0
0x10010034	0	0	0	0	0	0	0	0
0x10010038	0	0	0	0	0	0	0	0
0x1001003c	0	0	0	0	0	0	0	0
0x10010040	0	0	0	0	0	0	0	0
0x10010044	0	0	0	0	0	0	0	0
0x10010048	0	0	0	0	0	0	0	0
0x1001004c	0	0	0	0	0	0	0	0
0x10010050	0	0	0	0	0	0	0	0
0x10010054	0	0	0	0	0	0	0	0
0x10010058	0	0	0	0	0	0	0	0
0x1001005c	0	0	0	0	0	0	0	0
0x10010060	0	0	0	0	0	0	0	0
0x10010064	0	0	0	0	0	0	0	0
0x10010068	0	0	0	0	0	0	0	0
0x1001006c	0	0	0	0	0	0	0	0
0x10010070	0	0	0	0	0	0	0	0
0x10010074	0	0	0	0	0	0	0	0
0x10010078	0	0	0	0	0	0	0	0
0x1001007c	0	0	0	0	0	0	0	0
0x10010080	0	0	0	0	0	0	0	0
0x10010084	0	0	0	0	0	0	0	0
0x10010088	0	0	0	0	0	0	0	0
0x1001008c	0	0	0	0	0	0	0	0
0x10010090	0	0	0	0	0	0	0	0
0x10010094	0	0	0	0	0	0	0	0
0x10010098	0	0	0	0	0	0	0	0
0x1001009c	0	0	0	0	0	0	0	0
0x100100a0	0	0	0	0	0	0	0	0
0x100100a4	0	0	0	0	0	0	0	0
0x100100a8	0	0	0	0	0	0	0	0
0x100100ac	0	0	0	0	0	0	0	0
0x100100b0	0	0	0	0	0	0	0	0
0x100100b4	0	0	0	0	0	0	0	0
0x100100b8	0	0	0	0	0	0	0	0
0x100100bc	0	0	0	0	0	0	0	0
0x100100c0	0	0	0	0	0	0	0	0
0x100100c4	0	0	0	0	0	0	0	0
0x100100c8	0	0	0	0	0	0	0	0
0x100100cc	0	0	0	0	0	0	0	0
0x100100d0	0	0	0	0	0	0	0	0
0x100100d4	0	0	0	0	0	0	0	0
0x100100d8	0	0	0	0	0	0	0	0
0x100100dc	0	0	0	0	0	0	0	0
0x100100e0	0	0	0	0	0	0	0	0
0x100100e4	0	0	0	0	0	0	0	0
0x100100e8	0	0	0	0	0	0	0	0
0x100100ec	0	0	0	0	0	0	0	0
0x100100f0	0	0	0	0	0	0	0	0
0x100100f4	0	0	0	0	0	0	0	0
0x100100f8	0	0	0	0	0	0	0	0
0x100100fc	0	0	0	0	0	0	0	0
0x10010100	0	0	0	0	0	0	0	0
0x10010104	0	0	0	0	0	0	0	0
0x10010108	0	0	0	0	0	0	0	0
0x1001010c	0	0	0	0	0	0	0	0
0x10010110	0	0	0	0	0	0	0	0
0x10010114	0	0	0	0	0	0	0	0
0x10010118	0	0	0	0	0	0	0	0
0x1001011c	0	0	0	0	0	0	0	0
0x10010120	0	0	0	0	0	0	0	0
0x10010124	0	0	0	0	0	0	0	0
0x10010128	0	0	0	0	0	0	0	0
0x1001012c	0	0	0	0	0	0	0	0
0x10010130	0	0	0	0	0	0	0	0
0x10010134	0	0	0	0	0	0	0	0
0x10010138	0	0	0	0	0	0	0	0
0x1001013c	0	0	0	0	0	0	0	0
0x10010140	0	0	0	0	0	0	0	0
0x10010144	0	0	0	0	0	0	0	0
0x10010148	0	0	0	0	0	0	0	0
0x1001014c	0	0	0	0	0	0	0	0
0x10010150	0	0	0	0	0	0	0	0
0x10010154	0	0	0	0	0	0	0	0
0x10010158	0	0	0	0	0	0	0	0
0x1001015c	0	0	0	0	0	0	0	0
0x10010160	0	0	0	0	0	0	0	0
0x10010164	0	0	0	0	0	0	0	0
0x10010168	0	0	0	0	0	0	0	0
0x1001016c	0	0	0	0	0	0	0	0
0x10010170	0	0	0	0	0	0	0	0
0x10010174	0	0	0	0	0	0	0	0
0x10010178	0	0	0	0	0	0	0	0
0x1001017c	0	0	0	0	0	0	0	0
0x10010180	0	0	0	0	0	0	0	0
0x10010184	0	0	0	0	0	0	0	0
0x10010188	0	0	0	0	0	0	0	0
0x1001018c	0	0	0	0	0	0	0	0
0x10010190	0	0	0	0	0	0	0	0
0x10010194	0	0	0	0	0	0	0	0
0x10010198	0	0	0	0	0	0	0	0
0x1001019c	0	0	0	0	0	0	0	0
0x100101a0	0	0	0	0	0	0	0	0
0x100101a4	0	0	0	0	0	0	0	0
0x100101a8	0	0	0	0	0	0	0	0
0x100101ac	0	0	0	0	0	0	0	0
0x100101b0	0	0	0	0	0	0	0	0
0x100101b4	0	0	0	0	0	0	0	0
0x100101b8	0	0	0	0	0	0	0	0
0x100101bc	0	0	0	0	0	0	0	0
0x100101c0	0	0	0	0	0	0	0	0
0x100101c4	0	0	0	0	0	0	0	0
0x100101c8	0	0	0	0	0	0	0	0
0x100101cc	0	0	0	0	0	0	0	0
0x100101d0	0	0	0	0	0	0	0	0
0x100101d4	0	0	0	0	0	0	0	0
0x100101d8	0	0	0	0	0	0	0	0
0x100101dc	0	0	0	0	0	0	0	0
0x100101e0	0	0	0	0	0	0	0	0
0x100101e4	0	0	0	0	0	0	0	0
0x100101e8	0	0	0	0	0	0	0	0
0x100101ec	0	0	0	0	0	0	0	0
0x100101f0	0	0	0	0	0	0	0	0
0x100101f4	0	0	0	0	0	0	0	0
0x100101f8	0	0	0	0	0	0	0	0
0x100101fc	0	0	0	0	0	0	0	0
0x10010200	0	0	0	0	0	0	0	0
0x10010204	0	0	0	0	0	0	0	0
0x10010208	0	0	0	0	0	0	0	0
0x1001020c	0	0	0	0	0	0	0	0
0x10010210	0	0	0	0	0	0	0	0
0x10010214	0	0	0	0	0	0	0	0
0x10010218	0	0	0	0	0	0	0	0
0x1001021c	0	0	0	0	0	0	0	0
0x10010220	0	0	0	0	0	0	0	0
0x10010224	0	0	0	0	0	0	0	0
0x10010228	0	0	0	0	0	0	0	0
0x1001022c	0	0	0	0	0	0	0	0
0x10010230	0	0	0	0	0	0	0	0
0x10010234	0	0	0	0	0	0	0	0
0x10010238	0	0	0	0	0	0	0	0
0x1001023c	0	0	0	0	0	0	0	0
0x10010240	0	0	0	0	0	0	0	0
0x10010244	0	0	0	0	0	0	0	0
0x10010248	0	0	0	0	0	0	0	0
0x1001024c	0	0	0	0	0	0	0	0
0x10010250	0	0	0	0	0	0	0	0
0x10010254	0	0	0	0	0			

PROGRAMA 21:

EditExecute

Programa19.asmPrograma20.asmPrograma21.asmPrograma15.asmPrograma17.asm

```
# // programa 21
(
  ly = x^3 + 1 se x > 0
  |x^4 - 1 se x <= 0
)

# Os valores de x devem ser lidos da primeira posição livre da memória e o valor de y deverá ser
# escrito na segunda posição livre.

.text
.global main

main:
  lui $t0, 0x1001 # endereço 0x1001 é onde começa a memória no mips 0x1001 0000
  lw $t0, 0($t0) # x = MEM[0]

  slt $t1, $zero, $t0 # verificando se x > 0 (se $t1= 1, sim; se $t1= 0, não)
  bne $t1, $zero, maior # goto maior

menorIgual:
  mult $s0, $s0 # x^2
  mflo $t2
  mult $t2, $s0 # x^3
  mflo $t2
  mult $t2, $s0
  mflo $t2 # x^4

  ori $t3, $zero, 1 # t3 = 1, ou seja coloco 1 em um registrador para poder subtrair com o numero 1
  sub $t3, $t2, $t3 # t3 = x^4 - 1

} fim # jump para o fim

maior:
  mult $s0, $s0 # x^2
  mflo $t2
  mult $t2, $s0
  mflo $t2 # x^3

  ori $t3, $zero, 1 # t3 = 1 para adicionar
  add $t3, $t2, $t3 # t3 = x^3 + 1

fim:
  sw $t3, 4($t0) # Guarda o resultado na memória

.data
x: .word 2
```

RegistersCoproc 1Coproc 0

	Name	Number	Value
\$zero		0	0
\$at		1	0
\$v0		2	0
\$v1		3	0
\$a0		4	0
\$a1		5	0
\$a2		6	0
\$a3		7	0
\$t0		8	268500992
\$t1		9	1
\$t2		10	8
\$t3		11	9
\$t4		12	0
\$t5		13	0
\$t6		14	0
\$t7		15	0
\$s0		16	2
\$s1		17	0
\$s2		18	0
\$s3		19	0
\$s4		20	0
\$s5		21	0
\$s6		22	0
\$s7		23	0
\$s8		24	0
\$s9		25	0
\$k0		26	0
\$k1		27	0
\$gp		28	268469224
\$fp		29	2147479546
\$ra		30	0
\$ra		31	0
pc			4194384
hi			0
lo			8

EditExecute

Text Segment

Offset	Address	Code	Basic	Source
0	0x00400000	0x3c081001	lui \$t0, 0x1001	14: lui \$t0, 0x1001 # endereço 0x1001 é onde começa a memória no mips 0x1001 0000
4	0x00400004	0x8d100000	lw \$t0, 0(\$t0)	15: lw \$t0, 0(\$t0) # x = MEM[0]
8	0x00400008	0x0010482a	slt \$t1, \$zero, \$t0	17: slt \$t1, \$zero, \$t0 # verificando se x > 0 (se \$t1= 1, sim; se \$t1= 0, não)
12	0x0040000c	0x15200009	bne \$t1, \$zero, maior	19: bne \$t1, \$zero, maior # goto maior
16	0x00400010	0x02100018	mult \$s0, \$s0	21: mult \$s0, \$s0 # x^2
20	0x00400014	0x00005012	mflo \$t0	22: mflo \$t0
24	0x00400018	0x01500018	mult \$t2, \$s0	23: mult \$t2, \$s0 # x^3
28	0x0040001c	0x00005012	mflo \$t0	24: mflo \$t0
32	0x00400020	0x00180018	mult \$t2, \$s0	25: mult \$t2, \$s0
36	0x00400024	0x00005012	mflo \$t0	26: mflo \$t0 # x^4
40	0x00400028	0x340b0001	ori \$t3, \$zero, 1	28: ori \$t3, \$zero, 1 # t3 = 1, ou seja coloco 1 em um registrador para poder subtr...
44	0x0040002c	0x014b5822	sub \$t3, \$t2, \$t3	29: sub \$t3, \$t2, \$t3 # t3 = x^4 - 1
48	0x00400030	0x00100018	j fim	31: j fim # jump para o fim
52	0x00400034	0x02100018	mult \$s0, \$s0	34: mult \$s0, \$s0 # x^2
56	0x00400038	0x00005012	mflo \$t0	35: mflo \$t0
60	0x0040003c	0x01500018	mult \$t2, \$s0	36: mult \$t2, \$s0
64	0x00400040	0x00005012	mflo \$t0	37: mflo \$t0 # x^3
68	0x00400044	0x340b0001	ori \$t3, \$zero, 1	39: ori \$t3, \$zero, 1 # t3 = 1 para adicionar
72	0x00400048	0x014b5820	add \$t3, \$t2, \$t3	40: add \$t3, \$t2, \$t3 # t3 = x^3 + 1
76	0x0040004c	0x8d0b0004	sw \$t3, 4(\$t0)	43: sw \$t3, 4(\$t0) # Guarda o resultado na memória

RegistersCoproc 1Coproc 0

	Name	Number	Value
\$zero		0	0
\$at		1	0
\$v0		2	0
\$v1		3	0
\$a0		4	0
\$a1		5	0
\$a2		6	0
\$a3		7	0
\$t0		8	268500992
\$t1		9	1
\$t2		10	8
\$t3		11	9
\$t4		12	0
\$t5		13	0
\$t6		14	0
\$t7		15	0
\$s0		16	2
\$s1		17	0
\$s2		18	0
\$s3		19	0
\$s4		20	0
\$s5		21	0
\$s6		22	0
\$s7		23	0
\$s8		24	0
\$s9		25	0
\$k0		26	0
\$k1		27	0
\$gp		28	268469224
\$fp		29	2147479546
\$ra		30	0
\$ra		31	0
pc			4194384
hi			0
lo			8

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	2	9	0	0	0	0	0	0
0x10010004	0	0	0	0	0	0	0	0
0x10010008	0	0	0	0	0	0	0	0
0x1001000c	0	0	0	0	0	0	0	0
0x10010010	0	0	0	0	0	0	0	0
0x10010014	0	0	0	0	0	0	0	0
0x10010018	0	0	0	0	0	0	0	0
0x1001001c	0	0	0	0	0	0	0	0
0x10010020	0	0	0	0	0	0	0	0
0x10010024	0	0	0	0	0	0	0	0
0x10010028	0	0	0	0	0	0	0	0
0x1001002c	0	0	0	0	0	0	0	0
0x10010030	0	0	0	0	0	0	0	0
0x10010034	0	0	0	0	0	0	0	0
0x10010038	0	0	0	0	0	0	0	0
0x1001003c	0	0	0	0	0	0	0	0
0x10010040	0	0	0	0	0	0	0	0
0x10010044	0	0	0	0	0	0	0	0
0x10010048	0	0	0	0	0	0	0	0
0x1001004c	0	0	0	0	0	0	0	0

0x10010000 (data)Hexadecimal AddressesHexadecimal ValuesASCII