

**Arthur de Sá Braz de Matos**

**Ana Fernanda Sousa Cancado**

1. O que é um arquivo fonte?

A. um arquivo de texto que contém instruções de linguagem de programação.

2. O que é um registrador?

B. uma parte do processador que possui um padrão de bits.

3. Qual o caracter que, na linguagem assembly do SPIM, inicia um comentário?

A. #

4. Quantos bits há em cada instrução de máquina MIPS?

C. 32

5. O que é o contador de programa?

D. parte do processador que contém o endereço da próxima instrução de máquina para ser obtida.

6. Ao executarmos uma instrução, quanto será adicionado ao contador de programa?

A. 1

7. O que é uma diretiva, tal como a diretiva .text?

D. uma declaração que diz o montador algo sobre o que o programador quer, mas não corresponde diretamente a uma instrução de máquina.

8. O que é um endereço simbólico?

D. um nome usado no código-fonte em linguagem assembly para um local na memória.

9. Em qual endereço o simulador SPIM coloca a primeira instrução de máquina quando ele está sendo executado?

B. 0x00400000

10. Algumas instruções de máquina possuem uma constante como um dos operandos. Como é chamado tal operando?

A. operando imediato

11. Como é chamada uma operação lógica executada entre bits de cada coluna dos operandos para produzir um bit de resultado para cada coluna?

A. operação lógica

12. Quando uma operação é de fato executada, como estão os operandos na ALU?

D. Cada um dos registradores deve possuir 32 bit.

13. Dezesseis bits de dados de uma instrução de ori são usados como um operando imediato. Durante execução, o que deve ser feito primeiro?

B. Os dados são estendidos em zero à esquerda por 16 bits.

14. Qual das instruções seguintes armazenam no registrador \$5 um padrão de bits que representa positivo 48?

C. ori \$5,\$0,48

15. A instrução de ori pode armazenar o complemento de dois de um número em um registrador?

A. Não.

16. Qual das instruções seguintes limpa todos os bits no registrador \$8 com exceção do byte de baixa ordem que fica inalterado?

D. andi \$8,\$8,0xFF

17. Qual é o resultado de um ou exclusivo de padrão sobre ele mesmo?

A. Todos os bits em zero.

18. Todas as instruções de máquina têm os mesmos campos?

A. Não. Diferentes de instruções de máquina possuem campos diferentes.

# #programa 1

ep6\_1.asm

```
1  #a = $s0
2  #b = $s1
3  #c = $s2
4  #d = $s3
5  #x = $s4
6  #y = $s5
7
8  .text
9  ori $s0, $zero, 2 #a =2;
10 ori $s1, $zero, 3 #b =3;
11 ori $s2, $zero, 4 #c =4;
12 ori $s3, $zero, 5 #d =5;
13
14 add $t0, $s0, $s1 #(a+b)
15 add $t1, $s2, $s3 #(c+d)
16 sub $s4, $t0, $t1 #x = (a+b) - (c+d)
17
18 sub $t0, $s0, $s1 #a-b
19 add $s5, $t0, $s4 #y = a - b + x;
20
21 sub $s1, $s4, $s5 #b = x - y;
```

Text Segment

Bkpt	Address	Code	Basic	Source
<input type="checkbox"/>	0x00400000	0x34100002	ori \$16,\$0,2	9: ori \$s0, \$zero, 2 #a =2;
<input type="checkbox"/>	0x00400004	0x34110003	ori \$17,\$0,3	10: ori \$s1, \$zero, 3 #b =3;
<input type="checkbox"/>	0x00400008	0x34120004	ori \$18,\$0,4	11: ori \$s2, \$zero, 4 #c =4;
<input type="checkbox"/>	0x0040000c	0x34130005	ori \$19,\$0,5	12: ori \$s3, \$zero, 5 #d =5;
<input type="checkbox"/>	0x00400010	0x02114020	add \$8,\$16,\$17	14: add \$t0, \$s0, \$s1 #(a+b)
<input type="checkbox"/>	0x00400014	0x02534820	add \$9,\$18,\$19	15: add \$t1, \$s2, \$s3 #(c+d)
<input type="checkbox"/>	0x00400018	0x0109a022	sub \$20,\$8,\$9	16: sub \$s4, \$t0, \$t1 #x = (a+b) - (c+d)
<input type="checkbox"/>	0x0040001c	0x02114022	sub \$8,\$16,\$17	18: sub \$t0, \$s0, \$s1 #a-b
<input type="checkbox"/>	0x00400020	0x0114a820	add \$21,\$8,\$20	19: add \$s5, \$t0, \$s4 #y = a - b + x;
<input type="checkbox"/>	0x00400024	0x02958822	sub \$17,\$20,\$21	21: sub \$s1, \$s4, \$s5 #b = x - y;

Labels

☒ Data
 ☒ Text

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	-1
\$t1	9	9
\$t2	10	0
\$t3	11	0
\$t4	12	0
\$t5	13	0
\$t6	14	0
\$t7	15	0
\$s0	16	2
\$s1	17	1
\$s2	18	4
\$s3	19	5
\$s4	20	-4
\$s5	21	-5
\$s6	22	0
\$s7	23	0
\$s8	24	0
\$s9	25	0
\$k0	26	0
\$k1	27	0
\$gp	28	268468224
\$sp	29	2147479548
\$fp	30	0
\$ra	31	0
pc		4194344
hi		0
lo		0

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

0x10010000 (.data)

☒ Hexadecimal Addresses
 ☐ Hexadecimal Values
 ☐ ASCII

#programa 2

ep6\_2.asm\*

1

#x = \$s0

2

#y = \$s1

3

4

.text

5

ori \$s0, \$zero, 1 #x = 1

6

add \$t0, \$s0, \$s0 #t0 = 2x

7

add \$t0, \$t0, \$t0 #t0 = 4x

8

add \$t0, \$t0, \$s0 #t0 = 5x

9

addi \$s1, \$t0, 15 #y = 5x + 15

10

Text Segment

Bkpt	Address	Code	Basic	Source
<input type="checkbox"/>	0x00400000	0x34100001	ori \$16,\$0,1	5: ori \$s0, \$zero, 1 #x = 1
<input type="checkbox"/>	0x00400004	0x02104020	add \$8,\$16,\$16	6: add \$t0, \$s0, \$s0 #t0 = 2x
<input type="checkbox"/>	0x00400008	0x01084020	add \$8,\$8,\$8	7: add \$t0, \$t0, \$t0 #t0 = 4x
<input type="checkbox"/>	0x0040000c	0x01104020	add \$8,\$8,\$16	8: add \$t0, \$t0, \$s0 #t0 = 5x
<input type="checkbox"/>	0x00400010	0x2111000f	addi \$17,\$8,15	9: addi \$s1, \$t0, 15 #y = 5x + 15

Labels

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
\$s0	16	1
\$s1	17	20
\$s2	18	0
\$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	268468224
\$sp	29	2147479548
\$fp	30	0
\$ra	31	0
pc		4194324
hi		0
lo		0

☒ Data
 ☒ Text

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

0x10010000 (.data)

☒ Hexadecimal Addresses
 ☐ Hexadecimal Values
 ☐ ASCII

#programa 3

```
1  #x = $s0; y = $s1; z = %s2
2
3  .text
4  ori $s0, $zero, 3
5  ori $s1, $zero, 4
6
7  add $t0, $s0, $s0 #t0 = 2x
8  add $t0, $t0, $t0 #t0 = 4x
9  add $t0, $t0, $t0 #t0 = 8x
10 add $t0, $t0, $t0 #t0 = 16x
11 sub $t0, $t0, $s0 #t0 = 15x
12
13 add $t1, $s1, $s1 #t1 = 2y
14 add $t2, $t1, $s1 #t2 = 3y
15 add $t1, $t1, $t1 #t1 = 4y
16 add $t1, $t1, $t1 #t1 = 8y
17 add $t1, $t1, $t1 #t1 = 16y
18 add $t1, $t1, $t1 #t1 = 32y
19 add $t1, $t1, $t1 #t1 = 64y
20 add $t2, $t1, $t2 #t2 = 67y
21
22 add $t2, $t1, $t0 #t2 = 67y + 15x
```



```
1  #x = $s0
2  #y = $s1
3  #z = $s2
4
5  .text
6  ori $s0, $zero, 3 #x = 3
7  ori $s1, $zero, 4 #y = 4
8
9  sll $t0, $s0, 4 #t0 = 16x
10 sub $t0, $t0, $s0 #t0 = 15x
11
12 sll $t1, $s1, 6 #t1 = 64y
13 sll $t2, $s1, 1 #t2 = 2y
14 add $t2, $t1, $s1 #t2 = 3y
15 add $t2, $t1, $t2 #y = 64y + 3y
16
17 add $t1, $t0, $t2 #t1 = 15x + 67y
18 sll $s2, $t1, 2 #z = ( 15*x + 67*y) *4
```





ep6\_5.asm

```

1  #x = $s0
2  #y = $s1
3
4  .text
5  ori $t0, $zero, 0x186A
6  sll $s0, $t0, 4  #x = 100.000
7
8  ori $t0, $zero, 0x30D4
9  sll $s1, $t0, 4  #y = 200.000
10
11 add $s2, $s1, $s0 #z = x + y

```

EditExecute

Text Segment

Bkpt	Address	Code	Basic	Source
<input type="checkbox"/>	0x00400000	0x3408186a	ori \$8,\$0,6250	5: ori \$t0, \$zero, 0x186A
<input type="checkbox"/>	0x00400004	0x00088100	sll \$16,\$8,4	6: sll \$s0, \$t0, 4 #x = 100.000
<input type="checkbox"/>	0x00400008	0x340830d4	ori \$8,\$0,12500	8: ori \$t0, \$zero, 0x30D4
<input type="checkbox"/>	0x0040000c	0x00088900	sll \$17,\$8,4	9: sll \$s1, \$t0, 4 #y = 200.000
<input type="checkbox"/>	0x00400010	0x02309020	add \$18,\$17,\$16	11: add \$s2, \$s1, \$s0 #z = x + y

Labels

☒ Data
☒ Text

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

0x10010000 (.data)

☒ Hexadecimal Addresses
☐ Hexadecimal Values
☐ ASCII

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	12500
\$t1	9	0
\$t2	10	0
\$t3	11	0
\$t4	12	0
\$t5	13	0
\$t6	14	0
\$t7	15	0
\$s0	16	100000
\$s1	17	200000
\$s2	18	300000
\$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	268468224
\$sp	29	2147479548
\$fp	30	0
\$ra	31	0
pc		4194324
hi		0
lo		0

#programa 6

```

1  #x = $s0
2  #y = $s1
3  #z = $s2
4
5  .text
6  ori $t0, $zero, 0xFFFF
7  sll $t0, $t0, 16  #deslocando para a esquerda, ficando 0xFFFF0000
8  ori $s0, $t0, 0xFFFF #x = FFFFFFFF
9
10 ori $t0, $zero, 0x493E
11 sll $s1, $t0, 4  #y = 300.000
12
13 sll $t0, $s1, 2  #t0 = 4y
14
15 sub $s2, $s0, $t0 #z = x - 4y

```

Text Segment				
Bkpt	Address	Code	Basic	Source
	0x00400000	0x3408ffff	ori \$8,\$0,65535	6: ori \$t0, \$zero, 0xFFFF
	0x00400004	0x00084400	sll \$8,\$8,16	7: sll \$t0, \$t0, 16 #deslocando para a esquerda, ficando 0xFFFF0000
	0x00400008	0x3510ff00	ori \$16,\$8,65535	8: ori \$s0, \$t0, 0xFFFF #x = FFFFFFFF
	0x0040000c	0x3408493e	ori \$8,\$0,18750	10: ori \$t0, \$zero, 0x493E
	0x00400010	0x00088900	sll \$17,\$8,4	11: sll \$s1, \$t0, 4 #y = 300.000
	0x00400014	0x00114080	sll \$8,\$17,2	13: sll \$t0, \$s1, 2 #t0 = 4y
	0x00400018	0x02089022	sub \$18,\$16,\$8	15: sub \$s2, \$s0, \$t0 #z = x - 4y

The screenshot shows a window titled "Labels". The window contains a large empty rectangular area. At the bottom of the window, there are two checkboxes, both of which are checked. The first checkbox is labeled "Data" and the second is labeled "Text".

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

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	1200000
\$t1	9	0
\$t2	10	0
\$t3	11	0
\$t4	12	0
\$t5	13	0
\$t6	14	0
\$t7	15	0
\$s0	16	-1
\$s1	17	300000
\$s2	18	-1200001
\$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	268468224
\$sp	29	2147479548
\$fp	30	0
\$ra	31	0
pc		4194332
hi		0
lo		0

## #programa 7

ep6\_7.asm

```
1 ori $8, $0, 0x01    #estado inicial
2 ori $8, $8, 0xFFFF  #fazendo ORI, ficando 0000FFFF
3 sll $8, $8, 16       #deslocando para esquerda, ficando FFFF0000
4 ori $8, $8, 0xFFFF  #fazendo ORI, ficando FFFFFFFF
```

Text Segment

Bkpt	Address	Code	Basic	Source
	0x00400000	0x34080001	ori \$8,\$0,1	1: ori \$8, \$0, 0x01 #estado inicial
	0x00400004	0x3508ffff	ori \$8,\$8,65535	2: ori \$8, \$8, 0xFFFF #fazendo ORI, ficando 0000FFFF
	0x00400008	0x00084400	sll \$8,\$8,16	3: sll \$8, \$8, 16 #deslocando para esquerda, ficando FFFF0000
	0x0040000c	0x3508ffff	ori \$8,\$8,65535	4: ori \$8, \$8, 0xFFFF #fazendo ORI, ficando FFFFFFFF

Labels

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

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	-1
\$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
\$s8	24	0
\$t9	25	0
\$k0	26	0
\$k1	27	0
\$gp	28	268468224
\$sp	29	2147479548
\$fp	30	0
\$ra	31	0
pc		4194320
hi		0
lo		0

0x10010000 (data) Hexadecimal Addresses Hexadecimal Values ASCII

## #programa 8

Edit

Execute

ep6\_8.asm

```

1          #colocando 0x12345678 no registrador $8
2  ori $8, $zero, 0x1234
3  sll $8, $8, 16
4  ori $8, $8, 0x5678
5
6  srl $9, $8, 24      #colocando 0x12 no registrador $9
7
8  sll $10, $8, 8      #colocando 0x34 no registrador $10
9  srl $10, $10, 24
10
11 andi $11, $8, 0xFF00 #colocando 0x56 no registrador $10
12 srl $11, $11, 8
13
14 andi $12, $8, 0x00FF #colocando 0x78 no registrador $10

```

Edit

Execute

Text Segment

Bkpt	Address	Code	Basic	Source
<input type="checkbox"/>	0x00400000	0x34081234	ori \$8,\$0,4660	2: ori \$8, \$zero, 0x1234
<input type="checkbox"/>	0x00400004	0x00084400	sll \$8,\$8,16	3: sll \$8, \$8, 16
<input type="checkbox"/>	0x00400008	0x35085678	ori \$8,\$8,22136	4: ori \$8, \$8, 0x5678
<input type="checkbox"/>	0x0040000c	0x00084e02	srl \$9,\$8,24	6: srl \$9, \$8, 24 #colocando 0x12 no registrador \$9
<input type="checkbox"/>	0x00400010	0x00085200	sll \$10,\$8,8	8: sll \$10, \$8, 8 #colocando 0x34 no registrador \$10
<input type="checkbox"/>	0x00400014	0x000a5602	srl \$10,\$10,24	9: srl \$10, \$10, 24
<input type="checkbox"/>	0x00400018	0x310bff00	andi \$11,\$8,65280	11: andi \$11, \$8, 0xFF00 #colocando 0x56 no registrador \$10
<input type="checkbox"/>	0x0040001c	0x000b5a02	srl \$11,\$11,8	12: srl \$11, \$11, 8
<input type="checkbox"/>	0x00400020	0x310c00ff	andi \$12,\$8,255	14: andi \$12, \$8, 0x00FF #colocando 0x78 no registrador \$10

Labels

☒ Data
☒ Text

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	305419896
\$t1	9	18
\$t2	10	52
\$t3	11	86
\$t4	12	120
\$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
\$k0	26	0
\$k1	27	0
\$gp	28	268468224
\$sp	29	2147479548
\$fp	30	0
\$ra	31	0
pc		4194340
hi		0
lo		0

0x10010000 (.data)

☒ Hexadecimal Addresses
☐ Hexadecimal Values
☐ ASCII

## #programa 9

```
Edit Execute
ep6_9.asm
1  #x1 = $s0; x2 = $s1; x3 = $s2; x4 = $s3; soma = $s4
2  #endereço inicial = A[0] = $t0 = 0x10010000
3
4  .text
5  lui $t0, 0x1001 #salvando o endereço do primeiro endereço da memória
6
7  lw $s0, 0, ($t0) #pegando valor de x1
8  lw $s1, 4 ($t0)  #pegando valor de x2
9  lw $s2, 8 ($t0)  #pegando valor de x3
10 lw $s3, 12 ($t0) #pegando valor de x4
11
12 add $t1, $s0, $s1
13 add $t1, $t1, $s2
14 add $s4, $t1, $s3 #salvando no registrador $s4 o valor da soma
15
16 sw $s4, 16 ($t0) #salvando na memória o valor de 'soma', que esta em $s4
17
18 .data
19 x1: .word 15
20 x2: .word 25
21 x3: .word 13
22 x4: .word 17
23 soma: .word -1
```



```
1  #$s0 = x; $s1 = z; $s2 = y; $t0 = endereco da primeira posicao de memoria
2  .text
3  lui $t0, 0x1001
4
5  lw $s0, 0 ($t0)
6  sll $t1, $s0, 7    #t1 = 128x
7  sub $s0, $t1, $s0 #x = 127x
8
9  lw $s1, 4 ($t0)
10 sll $t1, $s1, 6    #t1 = 64z
11 add $s1, $t1, $s1 #z = 65z
12
13 sub $t1, $s0, $s1    #t1 = 127x - 65z
14 ori $t2, $zero, 0x0001 #t2 = 1
15 add $s2, $t1, $t2    #z = 127x - 65z + 1
16
17 sw $s2, 8 ($t0)    #salvando na memoria o valor de Y atualizado
18
19 .data
20 x: .word 5
21 z: .word 7
22 y: .word 0
```





```
1  #$s0 = x; $s1 = z; $s2 = y; $t0 = endereco da primeira posicao de memoria
2  .text
3  lui $t0, 0x1001
4
5  lw $s0, 0 ($t0) #pegando x
6  lw $s1, 4 ($t0) #pegando z
7
8  ori $t1, $zero, 0x493E
9  sll $t1, $t1, 4    #salvando 300.000 em t1
10
11 sub $t2, $s0, $s1 #t2 = x - z
12 add $s2, $t2, $t1 #y = x - z + 300000
13
14 sw $s2, 8 ($t0) #salvando na memoria o valor de Y atualizado
15
16 .data
17 x: .word 100000
18 z: .word 200000
19 y: .word 0
```



```

1  .data
2  x: .word 10
3
4
5  .text
6  ori $t0, $zero, 0x1001
7  sll $t0, $t0, 16
8
9  lw $s0, 0($t0)
10
11
12  add $t1, $t0, $zero
13  sw $t1, 4($t0)      # MEM[x] na posição 4 de memória
14
15  # endereço de MEM[MEM[x]] na terceira posição de memória
16  addi $t2, $t1, 4
17  sw $t2, 8($t0)
18
19  # MEM[MEM[MEM[x]]] na quarta posição de memória
20  addi $t3, $t2, 4
21  sw $t3, 12($t0)
22
23  lw $t3, 12($t0)
24  sll $t3, $t3, 1     # MEM[MEM[MEM[x]]] * 2
25
26
27  # resultado na posição 16)
28  sw $t3, 16($t0)
29
30

```



```
1  #t0 = primeira posicao da memoria
2  #s0 = numero
3  #t1 = bit de sinal (1 = negativo, 0 = positivo)
4
5  .text
6  lui $t0, 0x1001 #endereco da primeira posicao da memoria
7  lw $s0, 0 ($t0) #carregando numero
8
9  srl $t1, $s0, 31 #deslocando os bits, deixando o de sinal na primeira posicao a direita
10 beq $t1, $zero, positivo #se o numero for positivo, ã fazer nada
11                               #caso negativo, pegar seu modulo e substituir na memória
12 sub $s0, $zero, $s0
13 sw $s0, 0 ($t0)
14
15 positivo:
16     #se ja for positivo ã faz nada
17
18
19 .data
20 A: .word -15
```



```
1  #t0 = primeira posicao da memoria
2  #s0 = numero
3  #t1 = bit de verificacao de par ou impar (0 = par, 1 = impar)
4
5  .text
6  lui $t0, 0x1001 #endereco da primeira posicao da memoria
7  lw $s0, 0 ($t0) #carregando numero
8
9  andi $t1, $s0, 1 #faz um ANDi com o valor e o numero 1, se der 0 é par, se der 1, é impar
10 bne $t1, $zero, impar #se for par, vai atribuir 0 a t1, se for impar, vai atribuir 1 a t1
11 sw $t1, 4 ($t0)
12 j fim
13
14 impar:
15     sw $t1, 4 ($t0)
16
17 fim:
18
19 .data
20 A: .word 2023
```





```
3  #t3 = contador da posicao atual (i)
4  #t4 = valor a ser inserido na memoria
5
6  .text
7  lui $t0, 0x1001 #endereco da primeira posicao da memoria
8  lw $s0, 0 ($t0) #carregando numero
9
10 ori $t1, $zero, 0 #inicio da soma
11 ori $t2, $zero, 100 #condicao de parada do loop
12 ori $t3, $zero, 0 #contador da posicao atual (i)
13
14 loop:
15     beq $t2, $t3, fim #condicao de parada: contador de posicao ser igual a ultima posicao do array + 1
16     add $t4, $t3, $t3 #operacao
17     addi $t4, $t4, 1 #operacao
18     add $t1, $t1, $t4 #soma = soma + t4 (v[i])
19     sw $t4, 0 ($t0) #armazenando o valor na memoria
20     addi $t0, $t0, 4 #atualizando a posicao da memoria
21     addi $t3, $t3, 1 #atualizando o contador de posicoes
22     j loop
23
24 fim:
25     sw $t1, 0 ($t0) #escreve na memoria a soma dos valores do vetor
26
27 .data
28
```



```
1  #$s0 = x; $s1 = y; $s2 = z
2
3  .text
4  ori $s0, $zero, 0x186A
5  sll $s0, $s0, 8           #s0 = 1.600.000
6  ori $s1, $zero, 0x1388
7  sll $s1, $s1, 4           #s1 = 80.000
8  ori $s2, $zero, 0x61A8
9  sll $s2, $s2, 4           #s2 = 400.000
10
11 div $s0, $s2    #x/z
12 mflo $t0        #pega resultado e salva em t0
13 mult $t0, $s1     #(x/z) * y
14 mflo $t0        #pega resultado e salva em t0
15
16 .data
17
```



```
1  #$s0 = x; $s1 = y; $t2 = k
2
3  .text
4  lui $t0, 0x1001 ##endereco da primeira posicao da memoria
5  lw $s0, 0 ($t0) #s0 = x
6  lw $s1, 4 ($t0) #s1 = y
7  ori $t1, $zero, 0 #contador de somas realizadas
8  ori $t2, $zero, 0 #resultado temporario da soma (no final, K)
9
10 loop:
11     beq $t1, $s1, fim #so termina quando o contador for igual a qtd de multiplicacoes necessarias
12     add $t2, $t2, $s0
13     addi $t1, $t1, 1 #incrementa contador
14     j loop
15
16 fim:
17     sw $t2, 8 ($t0) #guardando K na memoria
18
19 .data
20 x: .word 20
21 y: .word 10
```



```
1  #$s0 = x; $s1 = y
2
3  .text
4  lui $t0, 0x1001 #endereço da primeira posição da memória
5  lw $s0, 0 ($t0) #s0 = x
6  lw $s1, 4 ($t0) #s1 = y
7  ori $t1, $zero, 1 #resultado temporário da exponenciação (no final, K)
8  ori $t3, $zero, 0 #contador do expoente
9
10 loop1:
11     beq $s1, $t3, fim
12     ori $t2, $zero, 0 #produto parcial
13     ori $t4, $zero, 0 #contador de somas (multiplicações)
14
15     loop2:
16         beq $t4, $t1, incExpo #so termina quando o contador for igual a qtd de somas necessárias
17         add $t2, $t2, $s0 #produto parcial
18         addi $t4, $t4, 1 #incrementa contador
19         j loop2
20
21 incExpo:
22     ori $t1, $zero, 0 #volta o resultado temporário para 0
23     add $t1, $t1, $t2 #produto passa a ser o resultado da multiplicação do loop2
24     addi $t3, $t3, 1 #incrementa contador do expoente
25     j loop1
26
27
28 fim:
29     sw $t1, 8 ($t0) #guardando K na memória
30
31 .data
32 x: .word 2
33 y: .word 3
```





```
1  .text
2  lui $t0, 0x1001  #pegando primeiro endereco de memoria e os 2 primeiros valores armazenados
3  lw $s0, 0 ($t0)
4  lw $s1, 4 ($t0)
5
6  mult $s0, $s1     #fazendo a multiplicacao
7  mflo $t1          #pegando o valor que esta no registrador low
8  sw $t1, 8 ($t0)   #guardando tal valor na 3a posicao de memoria
9  mfhi $t1          #pegando o valor que esta no registrador high
10 sw $t1, 12 ($t0)  #guardando tal valor na 4a posicao de memoria
11
12 .data
13 x: .word -10
14 y: .word 10
```

Bkpt	Address	Code	Basic	Source
<input type="checkbox"/>	0x00400000	0x3c081001	lui \$t0,\$0x3c081001	2: lui \$t0, 0x1001 #pegando primeiro endereco de memoria e os 2 primeiros valores armaze...
<input type="checkbox"/>	0x00400004	0x8d100000	lw \$s0,0(\$t0)	3: lw \$s0, 0 (\$t0)
<input type="checkbox"/>	0x00400008	0x8d110004	lw \$s1,4(\$t0)	4: lw \$s1, 4 (\$t0)
<input type="checkbox"/>	0x0040000C	0x02110018	mult \$l6,\$l7	6: mult \$s0, \$s1 #fazendo a multiplicacao
<input type="checkbox"/>	0x00400010	0x00004812	mflo \$9	7: mflo \$t1 #pegando o valor que esta no registrador low
<input type="checkbox"/>	0x00400014	0xad090008	sw \$9,8(\$t0)	8: sw \$t1, 8 (\$t0) #guardando tal valor na 3a posicao de memoria
<input type="checkbox"/>	0x00400018	0x00004810	mfhi \$9	9: mfhi \$t1 #pegando o valor que esta no registrador high
<input type="checkbox"/>	0x0040001c	0xad09000c	sw \$9,12(\$t0)	10: sw \$t1, 12 (\$t0) #guardando tal valor na 4a posicao de memoria

  

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	0
\$t3	11	0
\$t4	12	0
\$t5	13	0
\$t6	14	0
\$t7	15	0
\$s0	16	-10
\$s1	17	10
\$s2	18	0
\$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	268468224
\$sp	29	2147479548
\$fp	30	0
\$ra	31	0
pc		4194336
hi		-1
lo		-100

  

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	-10	10	-100	-1	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

Legend: Hexadecimal Addresses, Hexadecimal Values, ASCII

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	-1
\$t2	10	0
\$t3	11	0
\$t4	12	0
\$t5	13	0
\$t6	14	0
\$t7	15	0
\$s0	16	-10
\$s1	17	10
\$s2	18	0
\$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	268468224
\$ap	29	2147479548
\$fp	30	0
\$ra	31	0
pc		4194336
hi		-1
lo		-100

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

C. 64

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

B. hi e lo

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

A. mult

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

C. mflo \$8

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

B. 32

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

A. lo

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

D. div

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

A. 1110 0110

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

C. Se o inteiro for unsigned, o shift pode ocasionar um valor errado. 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?

A. ori \$3,\$0,3

mult \$8,\$3

mflo \$9

addi \$9,\$9,7

#programa 19

```

1  .text
2  lui $t0, 0x1001
3  lw $s0, 0 ($t0)
4  lw $s1, 4 ($t0)
5
6  ori $t0, $zero, 0  #contador X
7  ori $t1, $zero, 0  #contador Y
8  or $t2, $zero, $s0 #copia de s2
9  or $t3, $zero, $s1 #copia de s1
10
11 contaBitsX:
12     beq $t2, $zero, contaBitsY  #verifica se o numero eh igual a 0 (nao ha mais bits significativos)
13     addi $t0, $t0, 1             #add 1 no contador de bits
14     srl $t2, $t2, 1             #desloca bits para direita
15     j contaBitsX
16
17 contaBitsY:
18     beq $t3, $zero, multiplicacao #verifica se o numero eh igual a 0 (nao ha mais bits significativos)
19     addi $t1, $t1, 1             #add 1 no contador de bits
20     srl $t3, $t3, 1             #desloca bits para direita
21     j contaBitsY
22
23 multiplicacao:
24     slti $t0, $t0, 32            #verifica se a qtd de bits do primeiro numero eh menor que 32
25     beq $t0, $zero, multiplicacaoHL #se for maior (ou igual) ja vai pro tipo de multiplicacao que salva em HI e L0
26     slti $t1, $t1, 32            #verifica se a qtd de bits do segundo numero eh menor que 32
27     beq $t1, $zero, multiplicacaoHL #se for maior (ou igual) vai pro tipo de multiplicacao que salva em HI e L0
28
29     mult $s0, $s1                #se chegou ate aqui ambos numeros sao menores que 32 bits
30     mflo $s2                     #faz multiplicacao salvando apenas em L0
31     j fim
32
33 multiplicacaoHL:
34     mult $s0, $s1                #multiplica e depois busca do HI e L0
35     mflo $s2
36     mfhi $s3
37
38 fim:
39
40 .data
41 x: .word -10
42 y: .word 10

```



```

1  .text
2  lui $t0, 0x1001    #primeira posicao da memoria
3  lw $s0, 0 ($t0)    #pegando x
4
5  andi $t1, $s0, 1    #verifica se eh par ou impar (se $t1 = 1, impar; se $t1 = 0, par)
6  beq $t1, $zero, par #se for par vai para o label par, caso impar, seguir as proximas linhas
7
8  impar:
9      mult $s0, $s0 #x^2
10     mflo $t2
11     mult $t2, $s0
12     mflo $t2      #x^3
13     mult $t2, $s0 #x^4
14     mflo $t3
15     mult $t3, $s0
16     mflo $t3      #x^5
17
18     ori $t4, $zero, 1 #colocando 1 em um registrador para poder somar com o numero 1
19     sub $t3, $t3, $t2 #t3 = x^5 - x^3
20     add $t5, $t3, $t4 #t5 = x^5 - x^3 + 1
21
22     j fim
23
24  par:
25     mult $s0, $s0
26     mflo $t2      #x^2
27     mult $t2, $s0
28     mflo $t3      #x^3
29     mult $t3, $s0
30     mflo $t4      #x^4
31
32
33     ori $t5, $zero, 2 #colocando 2 em um registrador para poder multiplicar por pelo numero 2
34     mult $t2, $t5    #t2 = 2x^2
35     mflo $t2
36     add $t4, $t4, $t3 #t4 = x^4 + x^3
37     sub $t5, $t4, $t3 #t5 = x^4 + x^3 - 2x^2
38
39  fim:
40     sw $t5, 4 ($t0)    #guardando o resultado na memoria

```





```
1  .text
2  lui $t0, 0x1001      #primeira posicao da memoria
3  lw $s0, 0 ($t0)      #pegando x
4
5  slt $t1, $zero, $s0 #verificando se x > 0 (se $t1= 1, sim; se $t1= 0, nao)
6  bne $t1, $zero, maior
7
8  menorIgual:
9      mult $s0, $s0 #x^2
10     mflo $t2
11     mult $t2, $s0 #x^3
12     mflo $t2
13     mult $t2, $s0
14     mflo $t2      #x^4
15
16     ori $t3, $zero, 1 #colocando 1 em um registrador para poder subtrair com o numero 1
17     sub $t3, $t2, $t3 #t3 = x^4 - 1
18
19     j fim
20
21 maior:
22     mult $s0, $s0 #x^2
23     mflo $t2
24     mult $t2, $s0
25     mflo $t2      #x^3
26
27     ori $t3, $zero, 1 #colocando 1 em um registrador para poder somar com o numero 1
28     add $t3, $t2, $t3 #t3 = x^3 + 1
29
30 fim:
31     sw $t3, 4 ($t0)
32
33 .data
34 x: .word 2
```

Execute

Text Segment

Bkpt	Address	Code	Basic	Source
	0x0040000c	0x13200009	one \$r0,9	12: one \$t1, \$zero, maior
	0x00400010	0x02100018	mult \$16,\$16	15: mult \$s0, \$s0 #x^2
	0x00400014	0x00005012	mflo \$10	16: mflo \$t2
	0x00400018	0x01500018	mult \$10,\$16	17: mult \$t2, \$s0 #x^3
	0x0040001c	0x00005012	mflo \$10	18: mflo \$t2
	0x00400020	0x01500018	mult \$10,\$16	19: mult \$t2, \$s0
	0x00400024	0x00005012	mflo \$10	20: mflo \$t2 #x^4
	0x00400028	0x340b0001	ori \$11,\$0,1	22: ori \$t3, \$zero, 1 #colocando 1 em um registrador para poder subtrair com o num...
	0x0040002c	0x014b5822	sub \$11,\$10,\$11	23: sub \$t3, \$t2, \$t3 #t3 = x^4 - 1
	0x00400030	0x08100013	j 0x0040004c	25: j fim
	0x00400034	0x02100018	mult \$16,\$16	28: mult \$s0, \$s0 #x^2
	0x00400038	0x00005012	mflo \$10	29: mflo \$t2
	0x0040003c	0x01500018	mult \$10,\$16	30: mult \$t2, \$s0
	0x00400040	0x00005012	mflo \$10	31: mflo \$t2 #x^3
	0x00400044	0x340b0001	ori \$11,\$0,1	33: ori \$t3, \$zero, 1 #colocando 1 em um registrador para poder somar com o numero 1
	0x00400048	0x014b5820	add \$11,\$10,\$11	34: add \$t3, \$t2, \$t3 #t3 = x^3 + 1
	0x0040004c	0xad0b0004	sw \$11,4(\$0)	37: sw \$t3, 4 (\$t0)

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
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

Labels

Label	Address
ep6_21.asm	
menorigual	0x00400010
maior	0x00400034
fim	0x0040004c
x	0x10010000

0x10010000 (.data)

☒ Hexadecimal Addresses ☐ Hexadecimal Values ☐ ASCII

Registers		
Coprocc 1		Coprocc 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
\$t8	24	0
\$t9	25	0
\$k0	26	0
\$k1	27	0
\$gp	28	268468224
\$sp	29	2147479548
\$fp	30	0
\$ra	31	0
pc		4194384
hi		0
lo		8