Relatório 8

Bernardo Temponi, Diego Arruda, Diogo Araujo, Guilherme Fróes

Programa 09

Text Segment

Bkpt	Address	Code	Basic	Source
	0x00400000	0x21081001	addi $8,$8,0x00001001	7: addi $t0, $t0, 0x1001
	0x00400004	0x00084400	sll $8,$8,0x00000010	8: sll $t0,$t0, 16 #endereco base
	0x00400008	0x8d100000	lw $16,0x00000000($8)	10: lw $s0, 0 ($t0) #s0 = x1
	0x0040000c	0x02308820	add $17,$17,$16	12: add $s1, $s1, $s0 #s1 = s1 + x1
	0x00400010	0x8d100004	lw $16,0x00000004($8)	14: lw $s0, 0x4 ($t0) #s0 = x2
	0x00400014	0x02308820	add $17,$17,$16	16: add $s1, $s1, $s0 #s1 = s1 + x2
	0x00400018	0x8d100008	lw $16,0x00000008($8)	18: lw $s0, 0x8 ($t0) #s0 = x3
	0x0040001c	0x02308820	add $17,$17,$16	20: add $s1, $s1, $s0 #s1 = s1 + x3
	0x00400020	0x8d10000c	lw $16,0x0000000c($8)	22: lw $s0, 0xc ($t0) #s0 = x4
	0x00400024	0x02308820	add $17,$17,$16	24: add $s1, $s1, $s0 #s1 = s1 + x4
	0x00400028	0xad110010	sw $17,0x00000010($8)	26: sw $s1, 0x10 ($t0) #armazenando 0x46 no endereco 0x10010010

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)
0x10010000	0x0000000f	0x00000019	0x0000000d	0x00000011	0x00000046

Resposta Exercícios :

1) **C (64 bits);**
2) **B (hi, lo), parte mais significativa é armazenada no registrador hi e a parte**
   menos significativa é armazenada no lo;
3) **B MULT**
4) **B**
5) **A**
6) **A**
7) **D**
8) **B**
9) **B**
10) **A**

Programas

13)

```
.text
.globl main
main:
    addi $t0, $t0, 0x1001
    sll $t0,$t0, 16      #endereco base

    lw $t1, 0x0 ($t0)            #t1 = A

    srl $t2, $t1, 31             #t2 = ultimo Bit de A

    bne $t2 ,$0 , Negativo       # if ($t2 != 0)
    j Fim                        # (false) go to Fim

Negativo:
    sub $t3, $0, $t2             #t3 = -1 se ultimo Bit
== 1
    mult $t3, $t1                # lo <-- -1A
    mflo $t1                     # $t1 = -A

Fim:
    sw $t1, 0x0 ($t0) #no endereco 0x10010000

.data
A: .word 0x7fffffff
```

Text Segment

Bkpt	Address	Code	Basic		Source
	0x00400000	0x21081001	addi $8,$8,4097	15:	addi $t0, $t0, 0x1001
	0x00400004	0x00084400	sll $8,$8,16	16:	sll $t0,$t0, 16 #endereco base
	0x00400008	0x8d090000	lw $9,0($8)	18:	lw $t1, 0x0 ($t0) #t1 = A
	0x0040000c	0x000957c2	srl $10,$9,31	20:	srl $t2, $t1, 31 #t2 = ultimo Bit de A
	0x00400010	0x15400001	bne $10,$0,1	22:	bne $t2 ,$0 , Negativo # if ($t2 != 0)
	0x00400014	0x08100009	j 0x00400024	23:	j Fim # (false) go to Fim
	0x00400018	0x000a5822	sub $11,$0,$10	26:	sub $t3, $0, $t2 #t3 = -1 se ultimo Bit == 1
	0x0040001c	0x01690018	mult $11,$9	27:	mult $t3, $t1 # lo <-- -1A
	0x00400020	0x00004812	mflo $9	28:	mflo $t1 # $t1 = -A
	0x00400024	0xad090000	sw $9,0($8)	31:	sw $t1, 0x0 ($t0) #no endereco 0x10010000

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	2147483647
$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
$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)
0x10010000	2147483647	0
0x10010020	0	0

14)

```
.text

addi $t9, $0, 0x1001
sll $t9, $t9, 16
```

```
lw $s0, 0($t9)

addi $t0, $0, 0x1E
slt $s1, $t0, $s0     # $s1 -> $t0 < $s0
bne $s1, $0, true1    # if s1 != 0 (eh verdadeiro) :
true1
beq $s0, $t0, true1   # if $s0 == $t0 : true1
j flag0               # else : flag0

true1:
     addi $t1, $0, 0x32    # t1 -> 0x32 (50)
     slt $s1, $s0, $t1    # s1 -> $s0 < $t1
     beq $s1, $0, false   # if $s1 == 0 (eh maior)
     j flag1

false:
     bne $s0, $t1, flag0    # if $s0 != $t1 : FIM
     j flag1

flag1:
     addi $s3, $0, 0x1
     sw $s3, 4($t9)
     j FIM

flag0:
     addi $s3, $0, 0x0
     sw $s3, 4($t9)
     j FIM

FIM:

.data
TEMP: .word 50 # 0x1001000
```

```
FLAG: .word -1   # 0x1001004
```

Text Segment

Bkpt	Address	Code	Basic	Source
	0x00400000	0x20191001	addi $25,$0,4097	3: addi $t9, $0, 0x1001
	0x00400004	0x0019cc00	sll $25,$25,16	4: sll $t9, $t9, 16
	0x00400008	0x8f300000	lw $16,0($25)	6: lw $s0, 0($t9)
	0x0040000c	0x2008001e	addi $8,$0,30	8: addi $t0, $0, 0x1E
	0x00400010	0x0110882a	slt $17,$8,$16	9: slt $s1, $t0, $s0 # $s1 -> $t0 < $s0
	0x00400014	0x16200002	bne $17,$0,2	10: bne $s1, $0, true1 # if s1 != 0 (eh verdadeiro) : true1
	0x00400018	0x12080001	beq $16,$8,1	11: beq $s0, $t0, true1 # if $s0 == $t0 : true1
	0x0040001c	0x08100011	j 0x00400044	12: j flag0 # else : flag0
	0x00400020	0x20090032	addi $9,$0,50	15: addi $t1, $0, 0x32 # t1 -> 0x32 (50)
	0x00400024	0x0209882a	slt $17,$16,$9	16: slt $s1, $s0, $t1 # s1 -> $s0 < $t1
	0x00400028	0x12200001	beq $17,$0,1	17: beq $s1, $0, false # if $s1 == 0 (eh maior)
	0x0040002c	0x0810000e	j 0x00400038	18: j flag1
	0x00400030	0x16090004	bne $16,$9,4	21: bne $s0, $t1, flag0 # if $s0 != $t1 : FIM
	0x00400034	0x0810000e	j 0x00400038	22: j flag1
	0x00400038	0x20130001	addi $19,$0,1	25: addi $s3, $0, 0x1
	0x0040003c	0xaf330004	sw $19,4($25)	26: sw $s3, 4($t9)
	0x00400040	0x08100014	j 0x00400050	27: j FIM
	0x00400044	0x20130000	addi $19,$0,0	30: addi $s3, $0, 0x0
	0x00400048	0xaf330004	sw $19,4($25)	31: sw $s3, 4($t9)
	0x0040004c	0x08100014	j 0x00400050	32: j FIM

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	30
$t1	9	50
$t2	10	0
$t3	11	0
$t4	12	0
$t5	13	0
$t6	14	0
$t7	15	0
$s0	16	50
$s1	17	0
$s2	18	0
$s3	19	1
$s4	20	0
$s5	21	0
$s6	22	0
$s7	23	0
$t8	24	0
$t9	25	268500992
$k0	26	0
$k1	27	0
$gp	28	268468224
$sp	29	2147479548
$fp	30	0
$ra	31	0
pc		4194384
hi		0
lo		0

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)
0x10010000	50	1	0
0x10010020	0	0	0

15)

```
.text
addi $t0, $0, 0x1001
sll $t0, $t0, 16

lw $s0, 0($t0)
```

```
addi $t1, $0, 0x0  # contador($t1) -> 0
addi $t2, $0, 0x64 # tam -> 0x64 (100)
j LOOP

LOOP:
    sll $s1, $t1, 1     # 2*i
    addi $s1, $s1, 0x1  # 2*i + 1

    sw $s1, 0($t0)
    addi $t0, $t0, 0x4  # $t0 -> $t0 + 4

    addi $t1, $t1, 0x1  # $t1 -> $t1 + 1 (contador)

    bne $t1, $t2, LOOP  # if $t1 != $t2 : LOOP
    j SOMA

SOMA:
   addi $t0, $0, 0x1001
   sll $t0, $t0, 16

   addi $t1, $0, 0x0  # contador($t1) -> 0
   addi $t2, $0, 0x64 # tam -> 0x64 (100)

   addi $s0, $0, 0x0  # valor inicial da soma

   j LOOP2

LOOP2:

   lw $t3, 0($t0)
   add $s0, $s0, $t3
```

```
    addi $t0, $t0, 0x4 # $t0 -> $t0 + 4 (endereço)
    addi $t1, $t1, 0x1  # $t1 -> $t1 + 1 (contador)

    bne $t1, $t2, LOOP2  # if $t1 != $t2 : LOOP
    j FIM

FIM:
    sw $s0, 0($t0)


.data
vetor: .word 0 # 0x1001000
```

Text Segment

Bkpt	Address	Code	Basic	Source
	0x00400000	0x20081001	addi $8,$0,4097	2: addi $t0, $0, 0x1001
	0x00400004	0x00084400	sll $8,$8,16	3: sll $t0, $t0, 16
	0x00400008	0x8d100000	lw $16,0($8)	5: lw $s0, 0($t0)
	0x0040000c	0x20090000	addi $9,$0,0	7: addi $t1, $0, 0x0 # contador($t1) -> 0
	0x00400010	0x200a0064	addi $10,$0,100	8: addi $t2, $0, 0x64 # tam -> 0x64 (100)
	0x00400014	0x08100006	j 0x00400018	9: j LOOP
	0x00400018	0x00098840	sll $17,$9,1	12: sll $s1, $t1, 1 # 2*i
	0x0040001c	0x22310001	addi $17,$17,1	13: addi $s1, $s1, 0x1 # 2*i + 1
	0x00400020	0xad110000	sw $17,0($8)	15: sw $s1, 0($t0)
	0x00400024	0x21080004	addi $8,$8,4	16: addi $t0, $t0, 0x4 # $t0 -> $t0 + 4
	0x00400028	0x21290001	addi $9,$9,1	18: addi $t1, $t1, 0x1 # $t1 -> $t1 + 1 (contador)
	0x0040002c	0x152afffa	bne $9,$10,-6	20: bne $t1, $t2, LOOP # if $t1 != $t2 : LOOP
	0x00400030	0x0810000d	j 0x00400034	21: j SOMA
	0x00400034	0x20081001	addi $8,$0,4097	24: addi $t0, $0, 0x1001
	0x00400038	0x00084400	sll $8,$8,16	25: sll $t0, $t0, 16
	0x0040003c	0x20090000	addi $9,$0,0	27: addi $t1, $0, 0x0 # contador($t1) -> 0
	0x00400040	0x200a0064	addi $10,$0,100	28: addi $t2, $0, 0x64 # tam -> 0x64 (100)
	0x00400044	0x20100000	addi $16,$0,0	30: addi $s0, $0, 0x0 # valor inicial da soma
	0x00400048	0x08100013	j 0x0040004c	32: j LOOP2
	0x0040004c	0x8d0b0000	lw $11,0($8)	36: lw $t3, 0($t0)
	0x00400050	0x020b8020	add $16,$16,$11	37: add $s0, $s0, $t3
	0x00400054	0x21080004	addi $8,$8,4	39: addi $t0, $t0, 0x4 # $t0 -> $t0 + 4 (endereÃ§o)
	0x00400058	0x21290001	addi $9,$9,1	40: addi $t1, $t1, 0x1 # $t1 -> $t1 + 1 (contador)
	0x0040005c	0x152afffb	bne $9,$10,-5	42: bne $t1, $t2, LOOP2 # if $t1 != $t2 : LOOP
	0x00400060	0x08100019	j 0x00400064	43: j FIM
	0x00400064	0xad100000	sw $16,0($8)	46: sw $s0, 0($t0)

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	268501392
$t1	9	100
$t2	10	100
$t3	11	199
$t4	12	0
$t5	13	0
$t6	14	0
$t7	15	0
$s0	16	10000
$s1	17	199
$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		4194408
hi		0
lo		0

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	1	3	5	7	9	11	13	15
0x10010020	17	19	21	23	25	27	29	31
0x10010040	33	35	37	39	41	43	45	47
0x10010060	49	51	53	55	57	59	61	63
0x10010080	65	67	69	71	73	75	77	79
0x100100a0	81	83	85	87	89	91	93	95
0x100100c0	97	99	101	103	105	107	109	111
0x100100e0	113	115	117	119	121	123	125	127
0x10010100	129	131	133	135	137	139	141	143
0x10010120	145	147	149	151	153	155	157	159
0x10010140	161	163	165	167	169	171	173	175
0x10010160	177	179	181	183	185	187	189	191
0x10010180	193	195	197	199	10000	0	0	0

16)

```
.text
addi $t0, $0, 0x1001
sll $t0, $t0, 16

lw $s0, 0($t0)

addi $t1, $0, 0x0  # contador($t1) -> 0
addi $t2, $0, 0x64 # tam -> 0x64 (100)
addi $t3, $0, 0x64 # aux -> 0x64 (100)
j LOOPCriarVetor

LOOPCriarVetor: # colocar elementos no vetor (decrescente_
    sw $t3, 0($t0)

    addi $t0, $t0, 0x4  # $t0 -> $t0 + 4
    addi $t1, $t1, 0x1  # $t1 -> $t1 + 1 (contador)
    addi $t3, $t3, -0x1 # $t3--

    bne $t1, $t2, LOOPCriarVetor  # if $t1 != $t2 : LOOP
    j BubbleSort

BubbleSort:
        addi $t1, $0, 0x0 # contador (i)
    j LOOP1 # mais interno

LOOP1:
    addi $t4, $0, 0x0 # j = 0
    sub $t3, $t2, $t1 # parada -> tam-i
    j LOOP2 # mais externo

CONTINUE:
```

```
    addi $t1, $t1, 0x1 # i++
    bne $t1, $t2 LOOP1 #
    j FIM

LOOP2:
   addi $t8, $0, 0x1001 # controle de acesso a mem
   sll $t8, $t8, 16  # MEM [ J ]
   addi $t9, $t8, 0x4   # MEM [ J + 1 ]

   FOR:
      lw $s0, 0($t8) # $s0 -> MEM[J]
      lw $s1, 0($t9) # $s0 -> MEM[J+1]

      # if MEM [ J + 1 ] < MEM [ J  ] :
      slt $t5, $s1, $s0
      bne $t5, $0, SWAP # if true

      CONTINUE_FOR:
      addi $t8, $t8, 0x4 # MEM[J] = MEM[J+1]
      addi $t9, $t9, 0x4 # MEM[J+1] = MEM[J+1+1]
      addi $t4, $t4, 0x1 # j++
      bne $t4, $t3, FOR # if -> j != tam-i(parada)
   j CONTINUE

SWAP:
   sw $s1, 0($t8)
   sw $s0, 0($t9)
   j CONTINUE_FOR

FIM:
.data
vetor: .word 0 # 0x1001000
```

Text Segment

Bkpt	Address	Code	Basic	Source
	0x00400000	0x20081001	addi $8,$0,4097	2: addi $t0, $0, 0x1001
	0x00400004	0x00084400	sll $8,$8,16	3: sll $t0, $t0, 16
	0x00400008	0x8d100000	lw $16,0($8)	5: lw $s0, 0($t0)
	0x0040000c	0x20090000	addi $9,$0,0	7: addi $t1, $0, 0x0 # contador($t1) -> 0
	0x00400010	0x200a0064	addi $10,$0,100	8: addi $t2, $0, 0x64 # tam -> 0x64 (100)
	0x00400014	0x200b0064	addi $11,$0,100	9: addi $t3, $0, 0x64 # aux -> 0x64 (100)
	0x00400018	0x08100007	j 0x0040001c	10: j LOOPCriarVetor
	0x0040001c	0xad0b0000	sw $11,0($8)	13: sw $t3, 0($t0)
	0x00400020	0x21080004	addi $8,$8,4	15: addi $t0, $t0, 0x4 # $t0 -> $t0 + 4
	0x00400024	0x21290001	addi $9,$9,1	16: addi $t1, $t1, 0x1 # $t1 -> $t1 + 1 (contador)
	0x00400028	0x216bffff	addi $11,$11,-1	17: addi $t3, $t3, -0x1 # $t3--
	0x0040002c	0x152afffb	bne $9,$10,-5	19: bne $t1, $t2, LOOPCriarVetor # if $t1 != $t2 : LOOP
	0x00400030	0x0810000d	j 0x00400034	20: j BubbleSort
	0x00400034	0x20090000	addi $9,$0,0	23: addi $t1, $0, 0x0 # contador (i)
	0x00400038	0x0810000f	j 0x0040003c	24: j LOOP1 # mais interno
	0x0040003c	0x200c0000	addi $12,$0,0	27: addi $t4, $0, 0x0 # j = 0
	0x00400040	0x01495822	sub $11,$10,$9	28: sub $t3, $t2, $t1 # parada -> tam-i
	0x00400044	0x08100015	j 0x00400054	29: j LOOP2 # mais externo
	0x00400048	0x21290001	addi $9,$9,1	32: addi $t1, $t1, 0x1 # i++
	0x0040004c	0x152afffb	bne $9,$10,-5	33: bne $t1, $t2 LOOP1 #
	0x00400050	0x08100024	j 0x00400090	34: j FIM
	0x00400054	0x20181001	addi $24,$0,4097	37: addi $t8, $0, 0x1001 # controle de acesso a mem
	0x00400058	0x0018c400	sll $24,$24,16	38: sll $t8, $t8, 16 # MEM [J]
	0x0040005c	0x23190004	addi $25,$24,4	39: addi $t9, $t8, 0x4 # MEM [J + 1]
	0x00400060	0x8f100000	lw $16,0($24)	42: lw $s0, 0($t8) # $s0 -> MEM[J]
	0x00400064	0x8f310000	lw $17,0($25)	43: lw $s1, 0($t9) # $s0 -> MEM[J+1]
	0x00400068	0x0230682a	slt $13,$17,$16	46: slt $t5, $s1, $s0
	0x0040006c	0x15a00005	bne $13,$0,5	47: bne $t5, $0, SWAP # if true
	0x00400070	0x23180004	addi $24,$24,4	50: addi $t8, $t8, 0x4 # MEM[J] = MEM[J+1]
	0x00400074	0x23390004	addi $25,$25,4	51: addi $t9, $t9, 0x4 # MEM[J+1] = MEM[J+1+1]
	0x00400078	0x218c0001	addi $12,$12,1	52: addi $t4, $t4, 0x1 # j++
	0x0040007c	0x158bfff8	bne $12,$11,-8	53: bne $t4, $t3, FOR # if -> j != tam-i(parada)
	0x00400080	0x08100012	j 0x00400048	54: j CONTINUE
	0x00400084	0xaf110000	sw $17,0($24)	57: sw $s1, 0($t8)
	0x00400088	0xaf300000	sw $16,0($25)	58: sw $s0, 0($t9)
	0x0040008c	0x0810001c	j 0x00400070	59: j CONTINUE_FOR

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	268501392
$t1	9	100
$t2	10	100
$t3	11	1
$t4	12	1
$t5	13	1
$t6	14	0
$t7	15	0
$s0	16	1
$s1	17	0
$s2	18	0
$s3	19	0
$s4	20	0
$s5	21	0
$s6	22	0
$s7	23	0
$t8	24	268500996
$t9	25	268501000
$k0	26	0
$k1	27	0
$gp	28	268468224
$sp	29	2147479548
$fp	30	0
$ra	31	0
pc		4194448
hi		0
lo		0

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010040	16	17	18	19	20	21	22	23
0x10010060	24	25	26	27	28	29	30	31
0x10010080	32	33	34	35	36	37	38	39
0x100100a0	40	41	42	43	44	45	46	47
0x100100c0	48	49	50	51	52	53	54	55
0x100100e0	56	57	58	59	60	61	62	63
0x10010100	64	65	66	67	68	69	70	71
0x10010120	72	73	74	75	76	77	78	79
0x10010140	80	81	82	83	84	85	86	87
0x10010160	88	89	90	91	92	93	94	95
0x10010180	96	97	98	99	100	0	0	0

17)

```
.text
addi $t0, $0, 0x1001
sll $t0, $t0, 16

lw $s0, 0($t0) # $s0 -> x
div $t0, $s0, 0x2
mfhi $t0 # resto da divisao

beq $t0, $0, PAR
j IMPAR

PAR:
   # x^4
   addi $t8, $0, 0x3 # contador = expoente-1
   addi $t1, $s0, 0x0
   LOOP_ELEVAR1: # return -> $t1
       mult $s0, $t1
       mflo $t1
       addi $t8, $t8, -0x1 # contador--
       bne $t8, $t0, LOOP_ELEVAR1
   addi $s2, $t1, 0x0

   # x^3
   addi $t8, $0, 0x2 # contador = expoente-1
   addi $t1, $s0, 0x0
   LOOP_ELEVAR2: # return -> $t1
       mult $s0, $t1
       mflo $t1
       addi $t8, $t8, -0x1 # contador--
       bne $t8, $t0, LOOP_ELEVAR2
   add $s2, $s2, $t1
```

```
    # 2*x^2
    addi $t8, $0, 0x1 # contador = expoente-1
    addi $t1, $s0, 0x0
    LOOP_ELEVAR3: # return -> $t1
        mult $s0, $t1
        mflo $t1
        addi $t8, $t8, -0x1 # contador--
        bne $t8, $t0, LOOP_ELEVAR3

    sll $t1, $t1, 1 # x^2 * 2

    sub $s2, $s2, $t1 # x^4 + x^3 - 2*x^2
    j FIM

IMPAR:
    # x^5
    addi $t8, $0, 0x5 # contador = expoente
    addi $t1, $s0, 0x0
    LOOP_ELEVAR4: # return -> $t1
        mult $s0, $t1
        mflo $t1
        addi $t8, $t8, -0x1 # contador--
        bne $t8, $t0, LOOP_ELEVAR4

    addi $s2, $t1, 0x0

    # x^3
    addi $t8, $0, 0x3 # contador = expoente
    addi $t1, $s0, 0x0
    LOOP_ELEVAR5: # return -> $t1
        mult $s0, $t1
        mflo $t1
```

```
        addi $t8, $t8, -0x1 # contador--
        bne $t8, $t0, LOOP_ELEVAR5

    sub $s2, $s2, $t1
    addi $s2, $s2, 0x1
    j FIM


FIM:
    addi $t0, $0, 0x1001
    sll $t0, $t0, 16
    sw $s2, 4($t0)

.data
x: .word 3 # 0x1001000
y: .word -1 # 0x1001004
```

Basic	Source
addi $8,$0,4097	2: addi $t0, $0, 0x1001
sll $8,$8,16	3: sll $t0, $t0, 16
lw $16,0($8)	5: lw $s0, 0($t0) # $s0 -> x
addi $1,$0,2	6: div $t0, $s0, 0x2
div $16,$1	
mflo $8	
mfhi $8	7: mfhi $t0 # resto da divisao
beq $8,$0,1	9: beq $t0, $0, PAR
j 0x00400080	10: j IMPAR
addi $24,$0,3	14: addi $t8, $0, 0x3 # contador = expoente-1
addi $9,$16,0	15: addi $t1, $s0, 0x0
mult $16,$9	17: mult $s0, $t1
mflo $9	18: mflo $t1
addi $24,$24,-1	19: addi $t8, $t8, -0x1 # contador--
bne $24,$8,-4	20: bne $t8, $t0, LOOP_ELEVAR1
addi $18,$9,0	21: addi $s2, $t1, 0x0
addi $24,$0,2	24: addi $t8, $0, 0x2 # contador = expoente-1
addi $9,$16,0	25: addi $t1, $s0, 0x0
mult $16,$9	27: mult $s0, $t1
mflo $9	28: mflo $t1
addi $24,$24,-1	29: addi $t8, $t8, -0x1 # contador--
bne $24,$8,-4	30: bne $t8, $t0, LOOP_ELEVAR2
add $18,$18,$9	31: add $s2, $s2, $t1
addi $24,$0,1	34: addi $t8, $0, 0x1 # contador = expoente-1
addi $9,$16,0	35: addi $t1, $s0, 0x0
mult $16,$9	37: mult $s0, $t1
mflo $9	38: mflo $t1
addi $24,$24,-1	39: addi $t8, $t8, -0x1 # contador--
bne $24,$8,-4	40: bne $t8, $t0, LOOP_ELEVAR3
sll $9,$9,1	42: sll $t1, $t1, 1 # x^2 * 2
sub $18,$18,$9	44: sub $s2, $s2, $t1 # x^4 + x^3 - 2*x^2
j 0x004000c0	45: j FIM
addi $24,$0,5	49: addi $t8, $0, 0x5 # contador = expoente
addi $9,$16,0	50: addi $t1, $s0, 0x0
mult $16,$9	52: mult $s0, $t1
mflo $9	53: mflo $t1
addi $24,$24,-1	54: addi $t8, $t8, -0x1 # contador--
bne $24,$8,-4	55: bne $t8, $t0, LOOP_ELEVAR4
addi $18,$9,0	57: addi $s2, $t1, 0x0
addi $24,$0,3	60: addi $t8, $0, 0x3 # contador = expoente
addi $9,$16,0	61: addi $t1, $s0, 0x0
mult $16,$9	63: mult $s0, $t1
mflo $9	64: mflo $t1
addi $24,$24,-1	65: addi $t8, $t8, -0x1 # contador--
bne $24,$8,-4	66: bne $t8, $t0, LOOP_ELEVAR5
sub $18,$18,$9	68: sub $s2, $s2, $t1
addi $18,$18,1	69: addi $s2, $s2, 0x1
j 0x004000c0	70: j FIM
addi $8,$0,4097	74: addi $t0, $0, 0x1001
sll $8,$8,16	75: sll $t0, $t0, 16
sw $18,4($8)	76: sw $s2, 4($t0)

Registers | Coproc 1 | Coproc 0

Name	Number	Value
$zero	0	0
$at	1	2
$v0	2	0
$v1	3	0
$a0	4	0
$a1	5	0
$a2	6	0
$a3	7	0
$t0	8	268500992
$t1	9	27
$t2	10	0
$t3	11	0
$t4	12	0
$t5	13	0
$t6	14	0
$t7	15	0
$s0	16	3
$s1	17	0
$s2	18	217
$s3	19	0
$s4	20	0
$s5	21	0
$s6	22	0
$s7	23	0
$t8	24	1
$t9	25	0
$k0	26	0
$k1	27	0
$gp	28	268468224
$sp	29	2147479548
$fp	30	0
$ra	31	0
pc		4194508
hi		0
lo		27

Data Segment

Address	Value (+0)	Value (+4)
0x10010000	3	217
0x10010020	0	0

18)

```
.text

lui $t0, 0x1001
```

```
lw $a0, 0 ($t0)                 # a0 => x

addi $sp,$sp -8                 # adicionando na pilha
sw $t0, 8($sp)                  # guardando $t0 na pilha
sw $a0, 4($sp)                  # guardando $a0 na pilha


slt $t1, $0,$a0                 # Se 0 < x
beq $t1, $0 , Else              # nao => va para Else
jal MaiorZero                   # chamando funcao Se x > 0

j Continua

Else:
jal MenorIgualZero              # chamando funcao Se x <= 0

Continua:

lw $t0, 8($sp)                  # reatribuindo o valor de t0
lw $a0, 4($sp)                  # reatribuindo o valor de a0
addi $sp,$sp, 8                 # voltar na pilha

sw $v0, 4($t0)                  # guardando o resultado na
segunda posicao

j Fim

# a0 = valor
# a1 = expoente
Elevar:
    addi $sp,$sp,-8             # adicionando na pilha
    sw $s0, 8($sp)              # guardando $s0 na pilha
    sw $s1, 4($sp)              # guardando $s1 na pilha
```

```
    add $s0, $a0,$0          # s0 = resultado + valor

    addi $s1, $a1, -1        # a1 = expoente -1
# a1 = numero de repeticoes
Loop:
    slt $t0, $0, $s1         # se 0 < a1
    beq $t0, $0, FimLoop     # nao => va pra FimLoop
    addi $s1, $s1, -1        # repeticoes -1
    mult $s0, $a0
    mflo $s0
    j Loop                   # volta ao loop
FimLoop:

    add $v0,$s0,$0           # atribui o retorno

    lw $s0, 8($sp)           # reatribuindo o valor de s0
    lw $s1, 4($sp)           # reatribuindo o valor de s1
    addi $sp,$sp, 8          # voltar na pilha

    jr $ra                   # retorna

# a0 => x
MaiorZero:
    addi $sp,$sp,-8          # adicionando na pilha
    sw $ra, 8($sp)           # guardando $ra na pilha
    sw $a1, 4($sp)           # guardando $a1 na pilha =>
expoente

    addi $a1, $0,3           # a1 => 3

    #TODO CORPO
    jal Elevar               # v0 => x^3
```

```
    addi $v0,$v0,1          # v0 => x^3 + 1

    lw $ra, 8($sp)          # reatribuindo o valor de $ra
    lw $a1, 4($sp)          # reatribuindo o valor de $a1
    addi $sp,$sp, 8         # voltar na pilha

    jr $ra                  # retorna

# a0 => x
MenorIgualZero:
    addi $sp,$sp,-8         # adicionando na pilha
    sw $ra, 8($sp)          # guardando $ra na pilha
    sw $a1, 4($sp)          # guardando $a1 na pilha =>
expoente

    addi $a1, $0,4          # a1 => 4

    #TODO CORPO
    jal Elevar              # v0 => x^4

    addi $v0,$v0,-1         # v0 => x^4 - 1

    lw $ra, 8($sp)          # reatribuindo o valor de $ra
    lw $a1, 4($sp)          # reatribuindo o valor de $a1
    addi $sp,$sp, 8         # voltar na pilha

    jr $ra                  # retorna

Fim:

.data
x: .word 2
```

```
y: .word -1
```

Basic	Source
lui $8,4097	3: lui $t0, 0x1001
lw $4,0($8)	5: lw $a0, 0 ($t0) # a0 => x
addi $29,$29,-8	7: addi $sp,$sp -8 # adicionando na pilha
sw $8,8($29)	8: sw $t0, 8($sp) # guardando $t0 na pilha
sw $4,4($29)	9: sw $a0, 4($sp) # guardando $a0 na pilha
slt $9,$0,$4	12: slt $t1, $0,$a0 # Se 0 < x
beq $9,$0,2	13: beq $t1, $0 , Else # nao => va para Else
jal 0x0040007c	14: jal MaiorZero # chamando funcao Se x > 0
j 0x00400028	16: j Continua
jal 0x004000a4	19: jal MenorIgualZero # chamando funcao Se x <= 0
lw $8,8($29)	23: lw $t0, 8($sp) # reatribuindo o valor de t0
lw $4,4($29)	24: lw $a0, 4($sp) # reatribuindo o valor de a0
addi $29,$29,8	25: addi $sp,$sp, 8 # voltar na pilha
sw $2,4($8)	27: sw $v0, 4($t0) # guardando o resultado na segunda posicao
j 0x004000cc	29: j Fim
addi $29,$29,-8	34: addi $sp,$sp,-8 # adicionando na pilha
sw $16,8($29)	35: sw $s0, 8($sp) # guardando $s0 na pilha
sw $17,4($29)	36: sw $s1, 4($sp) # guardando $s1 na pilha
add $16,$4,$0	38: add $s0, $a0,$0 # s0 = resultado + valor
addi $17,$5,-1	40: addi $s1, $a1, -1 # a1 = expoente -1
slt $8,$0,$17	43: slt $t0, $0, $s1 # se 0 < a1
beq $8,$0,4	44: beq $t0, $0, FimLoop # nao => va pra FimLoop
addi $17,$17,-1	45: addi $s1, $s1, -1 # repeticoes -1
mult $16,$4	46: mult $s0, $a0
mflo $16	47: mflo $s0
j 0x00400050	48: j Loop # volta ao loop
add $2,$16,$0	51: add $v0,$s0,$0 # atribui o retorno
lw $16,8($29)	53: lw $s0, 8($sp) # reatribuindo o valor de s0
lw $17,4($29)	54: lw $s1, 4($sp) # reatribuindo o valor de s1
addi $29,$29,8	55: addi $sp,$sp, 8 # voltar na pilha
jr $31	57: jr $ra # retorna
addi $29,$29,-8	61: addi $sp,$sp,-8 # adicionando na pilha
sw $31,8($29)	62: sw $ra, 8($sp) # guardando $ra na pilha
sw $5,4($29)	63: sw $a1, 4($sp) # guardando $a1 na pilha => expoente
addi $5,$0,3	65: addi $a1, $0,3 # a1 => 3
jal 0x0040003c	68: jal Elevar # v0 => x^3
addi $2,$2,1	70: addi $v0,$v0,1 # v0 => x^3 + 1
lw $31,8($29)	72: lw $ra, 8($sp) # reatribuindo o valor de $ra
lw $5,4($29)	73: lw $a1, 4($sp) # reatribuindo o valor de $a1
addi $29,$29,8	74: addi $sp,$sp, 8 # voltar na pilha
jr $31	76: jr $ra # retorna
addi $29,$29,-8	80: addi $sp,$sp,-8 # adicionando na pilha
sw $31,8($29)	81: sw $ra, 8($sp) # guardando $ra na pilha
sw $5,4($29)	82: sw $a1, 4($sp) # guardando $a1 na pilha => expoente
addi $5,$0,4	84: addi $a1, $0,4 # a1 => 4
jal 0x0040003c	87: jal Elevar # v0 => x^4
addi $2,$2,-1	89: addi $v0,$v0,-1 # v0 => x^4 - 1
lw $31,8($29)	91: lw $ra, 8($sp) # reatribuindo o valor de $ra
lw $5,4($29)	92: lw $a1, 4($sp) # reatribuindo o valor de $a1
addi $29,$29,8	93: addi $sp,$sp, 8 # voltar na pilha
jr $31	95: jr $ra # retorna

Registers | Coproc 1 | Coproc 0

Name	Number	Value
$zero	0	0
$at	1	0
$v0	2	9
$v1	3	0
$a0	4	2
$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	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	4194336
pc		4194508
hi		0
lo		8

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)
0x10010000	2	9	0
0x10010020	0	0	0

19)

```
.text

lui $t0, 0x1001
```

```
lw $a0, 0 ($t0)                 # a0 => x
lw $a1, 4 ($t0)                 # a1 => y
lw $a2, 8 ($t0)                 # a2 => z

mult $a0, $a1                   # lo <-- x * y
mfhi $t1                        # t1 => x * y > 32bits
mflo $t2                        # t2 => x * y < 32bits

sll $t1, $t1, 24                # t1 => 0x 1d00 0000
srl $t2, $t2, 8                 # t2 => 0x 00cd 6500

add $t1,$t1,$t2                 # t1 => 0x 1dcd 6500

div $t1, $a2

mfhi $t1                        # t1 => 0x 0000 0000
mflo $t2                        # t2 => 0x 0000 04e2

sll $t2, $t2, 8                 # t2 => 0x 0004 e200

sw $t2, 12 ($t0)                # resultado => t2

.data
x: .word 0x186A00
y: .word 0x13880
z: .word 0x61A80
resultado: .word -1
```

Basic	Source	
lui $8,4097	3: lui $t0, 0x1001	
lw $4,0($8)	5: lw $a0, 0 ($t0)	# a0 => x
lw $5,4($8)	6: lw $a1, 4 ($t0)	# a1 => y
lw $6,8($8)	7: lw $a2, 8 ($t0)	# a2 => z
mult $4,$5	9: mult $a0, $a1	# lo <-- x * y
mfhi $9	10: mfhi $t1	# t1 => x * y > 32bits
mflo $10	11: mflo $t2	# t2 => x * y < 32bits
sll $9,$9,24	13: sll $t1, $t1, 24	# t1 => 0x 1d00 0000
srl $10,$10,8	14: srl $t2, $t2, 8	# t2 => 0x 00cd 6500
add $9,$9,$10	16: add $t1,$t1,$t2	# t1 => 0x 1dcd 6500
div $9,$6	18: div $t1, $a2	
mfhi $9	20: mfhi $t1	# t1 => 0x 0000 0000
mflo $10	21: mflo $t2	# t2 => 0x 0000 04e2
sll $10,$10,8	23: sll $t2, $t2, 8	# t2 => 0x 0004 e200
sw $10,12($8)	25: sw $t2, 12 ($t0)	# resultado => t2

Registers | Coproc 1 | Coproc 0

Name	Number	Value
$zero	0	0
$at	1	0
$v0	2	0
$v1	3	0
$a0	4	1600000
$a1	5	80000
$a2	6	400000
$a3	7	0
$t0	8	268500992
$t1	9	0
$t2	10	320000
$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
$k0	26	0
$k1	27	0
$gp	28	268468224
$sp	29	2147479548
$fp	30	0
$ra	31	0
pc		4194364
hi		0
lo		1250

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)
0x10010000	1600000	80000	400000	320000	0
0x10010020	0	0	0	0	0

20)

```
.text

lui $t0, 0x1001

lw $a0, 0 ($t0)                 # a0 => x
lw $a1, 4 ($t0)                 # a1 => y

mult $a0, $a1                   # lo <-- x * y
mflo $t1                        # t1 => x * y < 32bits

sw $t1, 8 ($t0)                # resultado => t2

.data
x: .word 10
y: .word 2
k: .word 0
```

Basic	Source
lui $8,4097	3: lui $t0, 0x1001
lw $4,0($8)	5: lw $a0, 0 ($t0) # a0 => x
lw $5,4($8)	6: lw $a1, 4 ($t0) # a1 => y
mult $4,$5	8: mult $a0, $a1 # lo <-- x * y
mflo $9	9: mflo $t1 # t1 => x * y < 32bits
sw $9,8($8)	11: sw $t1, 8 ($t0) # resultado => t2

Registers | Coproc 1 | Coproc 0

Name	Number	Value
$zero	0	0
$at	1	0
$v0	2	0
$v1	3	0
$a0	4	10
$a1	5	2
$a2	6	0
$a3	7	0
$t0	8	268500992
$t1	9	20
$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
$k0	26	0
$k1	27	0
$gp	28	268468224
$sp	29	2147479548
$fp	30	0
$ra	31	0
pc		4194328
hi		0
lo		20

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)
0x10010000	10	2	20	0
0x10010020	0	0	0	0

21)

```
.text

lui $t0, 0x1001

lw $a0, 0 ($t0)                 # a0 => x
```

```
lw $a1, 4 ($t0)                 # a1 => y

addi $sp,$sp -12                # adicionando na pilha
sw $t0, 12($sp)                 # guardando $t0 na pilha
sw $a0, 8($sp)                  # guardando $a0 na pilha
sw $a1, 4($sp)                  # guardando $a1 na pilha

jal Elevar                      # chamando funcao Loop

#TODO carregar o antigo $t0
lw $t0, 12($sp)                 # reatribuindo o valor de t0
lw $a0, 8($sp)                  # reatribuindo o valor de a0
lw $a1, 4($sp)                  # reatribuindo o valor de a1
addi $sp,$sp, 12                # voltar na pilha


sw $v0, 8($t0)

j FimElevar

# a0 = valor
# a1 = expoente
Elevar:
    addi $sp,$sp,-8             # adicionando na pilha
    sw $s0, 8($sp)              # guardando $s0 na pilha
    sw $s1, 4($sp)              # guardando $s1 na pilha

    add $s0, $a0,$0             # s0 = resultado + valor

    addi $s1, $a1, -1           # a1 = expoente -1
# a1 = numero de repeticoes
Loop:
    slt $t0, $0, $s1            # se 0 < a1
```

```
    beq $t0, $0, FimLoop     # nao => va pra FimLoop
    addi $s1, $s1, -1        # repeticoes -1
    mult $s0, $a0
    mflo $s0
    j Loop                   # volta ao loop
FimLoop:
    add $v0,$s0,$0           # atribui o retorno

    lw $s0, 8($sp)           # reatribuindo o valor de s0
    lw $s1, 4($sp)           # reatribuindo o valor de s1
    addi $sp,$sp, 8          # voltar na pilha

    jr $ra                   # retorna

FimElevar:

.data
x: .word 2
y: .word 3
k: .word 0
```

Basic	Source	
lui $8,4097	3: lui $t0, 0x1001	
lw $4,0($8)	5: lw $a0, 0 ($t0)	# a0 => x
lw $5,4($8)	6: lw $a1, 4 ($t0)	# a1 => y
addi $29,$29,-12	8: addi $sp,$sp -12	# adicionando na pilha
sw $8,12($29)	9: sw $t0, 12($sp)	# guardando $t0 na pilha
sw $4,8($29)	10: sw $a0, 8($sp)	# guardando $a0 na pilha
sw $5,4($29)	11: sw $a1, 4($sp)	# guardando $a1 na pilha
jal 0x00400038	13: jal Elevar	# chamando funcao Loop
lw $8,12($29)	16: lw $t0, 12($sp)	# reatribuindo o valor de t0
lw $4,8($29)	17: lw $a0, 8($sp)	# reatribuindo o valor de a0
lw $5,4($29)	18: lw $a1, 4($sp)	# reatribuindo o valor de a1
addi $29,$29,12	19: addi $sp,$sp, 12	# voltar na pilha
sw $2,8($8)	22: sw $v0, 8($t0)	
j 0x00400078	24: j FimElevar	
addi $29,$29,-8	29: addi $sp,$sp,-8	# adicionando na pilha
sw $16,8($29)	30: sw $s0, 8($sp)	# guardando $s0 na pilha
sw $17,4($29)	31: sw $s1, 4($sp)	# guardando $s1 na pilha
add $16,$4,$0	33: add $s0, $a0,$0	# s0 = resultado + valor
addi $17,$5,-1	35: addi $s1, $a1, -1	# a1 = expoente -1
slt $8,$0,$17	38: slt $t0, $0, $s1	# se 0 < a1
beq $8,$0,4	39: beq $t0, $0, FimLoop	# nao => va pra FimLoop
addi $17,$17,-1	40: addi $s1, $s1, -1	# repeticoes -1
mult $16,$4	41: mult $s0, $a0	
mflo $16	42: mflo $s0	
j 0x0040004c	43: j Loop	# volta ao loop
add $2,$16,$0	45: add $v0,$s0,$0	# atribui o retorno
lw $16,8($29)	47: lw $s0, 8($sp)	# reatribuindo o valor de s0
lw $17,4($29)	48: lw $s1, 4($sp)	# reatribuindo o valor de s1
addi $29,$29,8	49: addi $sp,$sp, 8	# voltar na pilha
jr $31	51: jr $ra	# retorna

Registers | Coproc 1 | Coproc 0

Name	Number	Value
$zero	0	0
$at	1	0
$v0	2	8
$v1	3	0
$a0	4	2
$a1	5	3
$a2	6	0
$a3	7	0
$t0	8	268500992
$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
$k0	26	0
$k1	27	0
$gp	28	268468224
$sp	29	2147479548
$fp	30	0
$ra	31	4194336
pc		4194424
hi		0
lo		8

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)
0x10010000	2	3	8	0
0x10010020	0	0	0	0