# EP_06

## Parte 1 - Questões fechadas

1. A
2. B
3. A
4. C
5. D
6. C
7. D
8. D
9. B
10. A
11. B
12. A
13. B
14. C
15. A
16. D
17. A
18. A

## Parte 2 - Implementação de programas

### 1

```
# a = $s1
# b = $s2
# c = $s3
# d = $s4
# x = $s5
# y = $s6

.text
.globl main
main:

# Atribuir valores das variaveis iniciais
addi $s1, $0, 2
```

```
addi $s2, $0, 3
addi $s3, $0, 4
addi $s4, $0, 5

# Fazer calculos

# x = a + b - c - d
add $s5, $s1 $s2
sub $s5, $s5, $s3
sub $s5, $s5, $s4

# y = a - b + x  --> y = a + x - b
add $s6, $s1, $s5
sub $s6, $s6, $s2

# b = x - y
sub $s2, $s5, $s6
```

## 2

```
# x = $s1
# y = $s2

.text
.globl main
main:

# x = 1
ori $s1, $0, 0x1

# y = 5*x
add $s2, $s1, $s1
add $s2, $s2, $s2
add $s2, $s2, $s1

# y += 15
addi $s2, $s2, 15
```

## 3

```
# x = $s1
# y = $s2
```

```
# z = $s3

.text
.globl main
main:

# x = 3
ori $s1, $0, 3

# y = 4
ori $s2, $0, 4

# tmp1 = 15*x
add $t1, $s1, $s1 # t1 = 2x
add $t1, $t1, $t1 # t1 = 4x
add $t1, $t1, $t1 # t1 = 8x
add $t1, $t1, $t1 # t1 = 16x
sub $t1, $t1, $s1 # t1 = 15x

# tmp2 = 67*y
add $t2, $s2, $s2 # t2 = 2y
add $t2, $t2, $t2 # t2 = 4y
add $t2, $t2, $t2 # t2 = 8y
add $t2, $t2, $t2 # t2 = 16y
add $t2, $t2, $t2 # t2 = 32y
add $t2, $t2, $t2 # t2 = 64y
add $t2, $t2, $s2 # t2 = 65y
add $t2, $t2, $s2 # t2 = 66y
add $t2, $t2, $s2 # t2 = 67y

# tmp1 = tmp1 + tmp2
add $t1, $t1, $t2

# z = tmp1 * 4
add $t1, $t1, $t1
add $s3, $t1, $t1
```

## 4

```
# x = $s1
# y = $s2
```

```
# z = $s3

.text
.globl main
main:

# x = 3
ori $s1, $0, 3

# y = 4
ori $s2, $0, 4

# z = 15x
sll $s3, $s1, 4
sub $s3, $s3, $s1

# tmp1 = 67y
sll $t1, $s2, 6 # tmp1 = 64y
add $t1, $t1, $s2 # tmp1 = 65y
add $t1, $t1, $s2 # tmp1 = 66y
add $t1, $t1, $s2 # tmp1 = 67y

# z = 4*(z + 67y)
add $s3, $s3, $t1
sll $s3, $s3, 2
```

## 5

```


# x = $s1
# y = $s2
# z = $s3

.text
.globl main
main:

# x = 100000
ori $s1, $0, 0x1000
sll $s1, $s1, 4
ori $s1, $s1, 0x86A0

# y = 200000
```

```
ori $s2, $0, 0x3000
sll $s2, $s2, 4
ori $s2, $s2, 0x0D40

# z = x + y
add $s3, $s1, $s2
```

## 6

```


# x = $s1
# y = $s2
# z = $s3

.text
.globl main
main:

# x = maior inteiro possivel
ori $s1, $0, 0x7FFF
sll $s1, 16
ori $s1, $s1, 0xFFFF

# y = 300000
ori $s2, $0, 0x0004
sll $s2, 16
ori $s2, $s2, 0x93E0

# z = x - 4y
sll $t1, $s2, 2 # t1 = 4y
sub $s3, $s1, $t1 # z = x - t1
```

## 7

```


.text
.globl main
main:

ori $8, $0, 0x01
srl $8, 1
not $8, $8
```

## 8

```
.text
.globl main
main:

# $8 = 0x12345678
ori $8, $0, 0x1234
sll $8, $8, 16
ori $8, $8, 0x5678

# registrador 9
srl $9, $8, 24

# registrador 10
srl $10, $8, 16
andi $10, $10, 0x00FF

# registrador 11
srl $11, $8, 8
andi $11, $11, 0xFF

# registrador 12
andi $12, $8, 0xFF
```

## 9

```
.text
.globl main

.data
x1: .word 15
x2: .word 25
x3: .word 13
x4: .word 17
soma: .word -1

.text

main:
```

```
lui $t0, 0x1001

lw $t1, 0($t0)
lw $t2, 4($t0)

# adiciomar primeiros 2 valores ao resultado
add $t3, $t1, $t2

lw $t1, 8($t0)
lw $t2, 12($t0)

# adicionar outros 2 valores
add $t3, $t3, $t1
add $t3, $t3, $t2

# gravar resultado
sw $t3, 16($t0)
```

## 10

```
.text
.globl main

.data
x: .word 5
z: .word 7
y: .word 0

# y = 127x – 65z + 1
.text

main:
lui $t0, 0x1001

lw $t1, 0($t0)
lw $t2, 4($t0)

# t1 = 127 * t1
sll $t3, $t1, 7
sub $t1, $t3, $t1

# t2 = 65 * t2
```

```
sll $t3, $t2, 6
add $t2, $t3, $t2

# t1 = t1 - t2 + 1
sub $t1, $t1, $t2
addi $t1, $t1, 1

# gravar resultado
sw $t1, 8($t0)
```

## 11

```
.text
.globl main

.data
x: .word 100000
z: .word 200000
y: .word 0

# y = x – z + 300000
.text

main:
lui $t0, 0x1001

lw $t1, 0($t0)
lw $t2, 4($t0)

# carregar 300000 em t3
lui $t3, 0x0004
ori $t3, $t3, 0x93E0

sub $t1, $t1, $t2
add $t1, $t1, $t3

sw $t1, 8($t0)
```

## 12

```
.text
.globl main

.data
x: .word 30

.text
main:

# preparar para endereçamento na memoria
lui $t0, 0x1001
ori $t1, $t0, 0x0020

# int* x
sw $t0, 0($t1)

# int** x
sw $t1, 4($t1)

# int*** x
addi $t2, $t1, 4
sw $t2, 4($t0)

# ler dados, a partir do ponteiro x ( 4($t0) )

# primeiro endereco
lw $t1, 4($t0)

# primeira dereferencia
lw $t1, 0($t1)

# segunda dereferencia
lw $t1, 0($t1)

# terceira dereferencia (guardando o valor de x em t2)
lw $t2, 0($t1)

# agora t2 contem ***x e t1 contem o seu endereco
# logo, basta multiplicar t2 por 2 e gravar no endereço armazenado em t1
sll $t2 $t2, 1
sw $t2, 0($t1)
```

## 13

```
.text
.globl main

.data
x: .word 30

.text
main:
lui $t0, 0x1001
lw $s0, 0($t0)

lui $t2, 0x8000
and $t1, $s0, $t2

bne $t2, $t1, par

sub $s0, $0, $s0
sw $s0, 0($t0)

par:
```

## 14

```
.text
.globl main

.data
x: .word 57

.text
main:
lui $t0, 0x1001
lw $s0, 0($t0)

andi $t1, $s0, 0x0001

sw $t1, 4($t0)
```

## 15

```
.text
.globl main

.data
arr: .word 0

.text
main:

# t0 = enderecamento
# t3 = soma
# t1 = iterador
# t2 = valor atual
# t4 = end

lui $t0, 0x1001
addi $t4, $t0, 400

loop:
add $t2, $t1, $t1
addi $t2, $t2, 1

add $t3, $t3, $t2

sw $t2, 0($t0)
addi $t0, $t0, 4
addi $t1, $t1, 1

bne $t4, $t0, loop

sw $t3, 0($t0)
```

## 16

```
.text
.globl main

.data
```

```
x: .word 0x186A001
y: .word 0x13880
z: .word 0x61A80


.text
# (0x186A00 * 0x13880) / 0x61A80
main:

lui $s1, 0x0018
ori $s1, $s1, 0x6A00

lui $s2, 0x0001
ori $s2, $s2, 0x3880

lui $s3, 0x0006
ori $s3, $s3, 0x1A80

or $s6, $0, $s1
or $s7, $0, $s3
jal div_s5_s6_s7

or $s6, $0, $s5
or $s7, $0, $s2
jal mult_s5_s6_s7

or $s4, $0, $s5

j fim

# funcao para s5 = s6 * s7
mult_s5_s6_s7:
  or $t1, $0, $s6
  or $s5, $0, $0
  loop1:
  add $s5, $s5, $s7
  addi $t1, $t1, -1
  bne $t1, $0, loop1
  jr $ra

# funcao para s5 = s6 / s7
div_s5_s6_s7:
  or $t1, $0, $s6
```

```
  or $s5, $0, $0
  loop2:
  addi $s5, $s5, 1
  sub $t1, $t1, $s7
  srl $t2, $t1, 31
  beq $t2, $0, loop2
  addi $s5, $s5, -1
  jr $ra

fim:
```

## 17

```
.text
.globl main

.data
x: .word 13
y: .word 12


.text
# (0x186A00 * 0x13880) / 0x61A80
main:
lui $s0, 0x1001
lw $s6, 0($s0)
lw $s7, 4($s0)
jal mult_s5_s6_s7
sw $s5, 8($s0) # Gravar resultado na memoria

j fim

# funcao para s5 = s6 * s7
mult_s5_s6_s7:
  or $t1, $0, $s6
  or $s5, $0, $0
  loop1:
  add $s5, $s5, $s7
  addi $t1, $t1, -1
  bne $t1, $0, loop1
  jr $ra
```

```
fim:
```

## 18

```
.text
.globl main

.data
x: .word 3
y: .word 5

.text
# (0x186A00 * 0x13880) / 0x61A80
main:
lui $s0, 0x1001
lw $s3, 0($s0)
lw $s4, 4($s0)
jal pot_s2_s3_s4
sw $s2, 8($s0) # Gravar resultado na memoria

j fim

# funcao para s5 = s6 * s7
mult_s5_s6_s7:
  or $t1, $0, $s6
  or $s5, $0, $0
  loop1:
  add $s5, $s5, $s7
  addi $t1, $t1, -1
  bne $t1, $0, loop1
  jr $ra

# funcao para s2 = s3 ^ s4
pot_s2_s3_s4:
  or $t9, $0, $ra

  addi $t8, $s4, -1
  or $s7, $s3, $0   # s7 = s3
  or $s2, $s3, $0   # s2 = s3
  loop2:
  or $s6, $s2, $0
```

```
  jal mult_s5_s6_s7
  or $s2, $0, $s5

  addi $t8, $t8, -1
  srl $t2, $t8, 31
  bne $t8, $0, loop2




  or $ra, $0, $t9
  jr $ra
fim:
```

## Parte 3 - Mais Questões Fechadas

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

## Parte 4 - Mais programas (mult, div, mflo e mfhi para frente)

### 19

```
# $s0 -> x
# $s1 -> y
# $s2 -> MEM[lo]
# $s3 -> MEM[hi]
# $t0 -> TAM_X
# $t1 -> TAM_Y
# $t2 -> endereco base

.text
.globl main
main:
```

```
lui $t2, 0x1001 # t2 = MEM[0]
lw $s0, 0($t2) # x = MEM[t2]
lw $s1, 4($t2) # y = MEM[t2 + 1]
 pre_loop_x:
  ori $t3, $zero, 0 # TAM = 0
  beq $s0, $zero, pre_loop_y
  ori $t3, $zero, 32 # TAM = 32
  lui $t4, 0x8000 # t4 = 0x80000000
  and $t5, $s0, $t4 # testa bit significativo
  bne $t5, $zero, pre_loop_y
loop_x:
  addi $t3, $t3, -1 # TAM_X = TAM_X - 1
  srl $t4, $t4, 1 # t4 >> 1
  and $t5, $s0, $t4 # testa bit significativo
  beq $t5, $zero, loop_x
pre_loop_y:
  or $t0, $zero, $t3 # t0 = TAM_X
  ori $t3, $zero, 0 # TAM = 0
  beq $s1, $zero, pre_multi
  ori $t3, $zero, 32 # TAM = 32
  lui $t4, 0x8000 # t4 = 0x80000000
  and $t5, $s1, $t4 # testa bit significativo
  bne $t5, $zero, pre_multi
loop_y:
  addi $t3, $t3, -1 # TAM_Y = TAM_Y - 1
  srl $t4, $t4, 1 # t4 >> 1
  and $t5, $s1, $t4 # testa bit significativo
  beq $t5, $zero, loop_y
pre_multi:
  or $t1, $zero, $t3 # t1 = TAM_Y
  add $t4, $t0, $t1 # t4 = TAM_X + TAM_Y
  slti $t4, $t4, 33 # Se t4 for menor que 32 bits, dar 1
  beq $t4, $zero, maior32bits
menor32bits:
  mult $s0, $s1
  mflo $s2
  addi $v0, $zero, 10
  syscall
maior32bits:
  mult $s0, $s1
  mflo $s2
  mfhi $s3
  addi $v0, $zero, 10
```

```
    syscall
.data
x: .word 1073741823
y: .word 2
```

## 20

```
.text
.globl main
main:
lui $t0, 0x1001
lw $s0, 0($t0)
andi $t1, $s0, 0x1
beq $t1, $zero, par # Se o resultado for 0, eh par. Se 1, eh impar
impar:
mult $s0, $s0 # x^2
mflo $t3 # t3 = x^2
mult $t3, $s0 # x^3
mflo $t3 # t3 = x^3
mult $t3, $s0 # x^4
mflo $t2 # t2 = x^4
mult $t2, $s0 # x^5
mflo $t2 # t2 = x^5

sub $t2, $t2, $t3 # t2 = x^5 - x^3
addi $s1, $t2, 1 # y = x^5 - x^3 + 1
j resultado

par:
mult $s0, $s0 # x^2
mflo $t4 # t2 = x^2
mult $t4, $s0 # x^3
mflo $t3 # t3 = x^3
mult $t3, $s0 # x^4
mflo $t2 # t2 = x^4

sll $t4, $t4, 1 # t4 = 2x^2

add $t2, $t2, $t3 # t2 = x^4 + x^3
sub $s1, $t2, $t4 # y = x^4 + x^3 - 2x^2

resultado:
```

```
sw $s1, 4($t0) # MEM[t0 + 4] = y

.data
x: 3
y: -1 # sera sobrescrito no final
```

## 21

```


.text
.globl main
main:
lui $t0, 0x1001
lw $s0, 0($t0)
slti $t1, $s0, 1
beq $t1, $zero, maiorZero # Se for 0, eh maior que zero

menorigualZero:
mult $s0, $s0 # x^2
mflo $t2 # t2 = x^2
mult $t2, $t2 # x^4
mflo $t2 # t2 = x^4

addi $s1, $t2, -1
j resultado

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

addi $s1, $t2, 1 # y = x^3 + 1

resultado:
sw $s1, 4($t0)
.data
x: .word -4
y: -1 # sera sobrescrito no final
```