Exercício 02 - Arquitetura de Computadores 2

Henrique Oliveira da Cunha Franco

1 Programa 1

```
.text
.globl _start

_start:
    # Inicializacao de s1, s2, s3, e s4 com valores especificos
    addi s1, zero, 2    # s1 = 2
    addi s2, zero, 3    # s2 = 3
    addi s3, zero, 4    # s3 = 4
    addi s4, zero, 5    # s4 = 5

    # Calcula t0 = s1 + s2 e t1 = s3 + s4
    add t0, s1, s2      # t0 = s1 + s2
    add t1, s3, s4      # t1 = s3 + s4

    # Calcula diferenca entre t0 e t1, guardando em s5
    sub s5, t0, t1      # s5 = t0 - t1

    # Calcula t0 = s1 - s2
    sub t0, s1, s2      # t0 = s1 - s2

    # Calcula s6 = t0 + s5
    add s6, t0, s5      # s6 = t0 + s5

    # Calcula diferenca entre s5 e s6, guardando em s2
    sub s2, s5, s6      # s2 = s5 - s6
```

2 Programa 2

```
.text
.globl _start

_start:
    # Inicializa s1 com o valor 1
    addi s1, zero, 1    # s1 = 1

    # Calcula o negativo de s1 e armazena em t0
    sub t0, zero, s1    # t0 = -s1

    # Adiciona 5 ao valor armazenado em t0 e armazena em t1
    addi t1, t0, 5      # t1 = t0 + 5

    # Adiciona 15 ao valor armazenado em t1 e armazena em s2
    addi s2, t1, 15     # s2 = t1 + 15
```

3 Programa 3

```
.text
.globl _start

_start:
    # Inicializa s1 com o valor 3
    addi s1, zero, 3     # s1 = 3

    # Inicializa s2 com o valor 4
    addi s2, zero, 4     # s2 = 4

    # Calcula o negativo de s1 e armazena em t0
    sub t0, zero, s1     # t0 = -s1

    # Adiciona 15 ao valor armazenado em t0 e armazena em t1
    addi t1, t0, 15      # t1 = t0 + 15

    # Calcula o negativo de s2 e armazena em t2
    sub t2, zero, s2     # t2 = -s2

    # Adiciona 67 ao valor armazenado em t2 e armazena em t3
    addi t3, t2, 67      # t3 = t2 + 67

    # Soma os valores armazenados em t1 e t3 e armazena em t4
    add t4, t1, t3       # t4 = t1 + t3

    # Adiciona 4 ao valor armazenado em t4 e armazena em s3
    addi s3, t4, 4       # s3 = t4 + 4
```

4 Programa 4

```
.text
.globl _start

_start:
    # Inicializa s1 com o valor 1
    addi s1, zero, 1     # s1 = 1

    # Inicializa t0 com o valor 5
    addi t0, zero, 5     # t0 = 5

    # Multiplica s1 por t0 e armazena em t1
    mul t1, s1, t0       # t1 = s1 * t0

    # Adiciona 15 ao valor armazenado em t1 e armazena em s2
    addi s2, t1, 15      # s2 = t1 + 15
```

5 Programa 5

```
.text
.globl _start

_start:
    # Inicializa s1 com o valor 3
    addi s1, zero, 3        # s1 = 3

    # Inicializa s2 com o valor 4
    addi s2, zero, 4        # s2 = 4

    # Inicializa t0 com o valor 15
    addi t0, zero, 15       # t0 = 15

    # Multiplica t0 por s1 e armazena em t1
    mul t1, t0, s1          # t1 = t0 * s1

    # Inicializa t2 com o valor 67
    addi t2, zero, 67       # t2 = 67

    # Multiplica t2 por s2 e armazena em t3
    mul t3, t2, s2          # t3 = t2 * s2

    # Soma os valores armazenados em t1 e t3 e armazena em t4
    add t4, t1, t3          # t4 = t1 + t3

    # Inicializa t5 com o valor 4
    addi t5, zero, 4        # t5 = 4

    # Multiplica o valor armazenado em t4 por t5 e armazena em s3
    mul s3, t4, t5          # s3 = t4 * t5
```

6 Programa 6

```
.text
.globl _start

_start:
    # Inicializa t0 com o valor 1
    addi t0, zero, 1        # t0 = 1

    # Desloca t0 para a esquerda 20 bits e armazena em s1
    slli s1, t0, 20         # s1 = t0 << 20

    # Reinicializa t0 com o valor 1
    addi t0, zero, 1        # t0 = 1

    # Desloca t0 para a esquerda 12 bits e armazena em s2
    slli s2, t0, 12         # s2 = t0 << 12

    # Soma os valores armazenados em s1 e s2 e armazena em s3
    add s3, s1, s2          # s3 = s1 + s2
```

7 Programa 7

```
.text
.globl _start

_start:
    # Carrega o valor imediato 0xFFFFFFFF em s1
    li s1, 0xFFFFFFFF       # s1 = 0xFFFFFFFF (hexadecimal)

    # Carrega o valor imediato 8192 em s2
    li s2, 8192             # s2 = 8192

    # Desloca s2 para a esquerda 2 bits e armazena em t0
    slli t0, s2, 2          # t0 = s2 << 2

    # Subtrai o valor armazenado em t0 de s1 e armazena em s3
    sub s3, s1, t0          # s3 = s1 - t0
```

8 Programa 8

```
.text
.globl _start

_start:
    ori x8, x0, 0x01      # x8 = 0x00000001 inicialmente

    slli x8, x8, 1        # Desloca o bit 1 para a esquerda, x8 =
       0x00000002
    ori x8, x8, 0x01      # Define o bit 0, x8 = 0x00000003

    slli x8, x8, 2        # Desloca os bits 1 e 0 para a esquerda, x8 =
       0x0000000C
    ori x8, x8, 0x03      # Define os bits 1 e 0, x8 = 0x0000000F

    slli x8, x8, 4        # Desloca os bits 3, 2, 1 e 0 para a esquerda,
       x8 = 0x000000F0
    ori x8, x8, 0x0F      # Define os bits 3, 2, 1 e 0, x8 = 0x000000FF

    slli x8, x8, 8        # Desloca os bits 7 ate 0 para a esquerda, x8 =
       0x0000FF00
    ori x8, x8, 0xFF      # Define os bits 7 ate 0, x8 = 0x0000FFFF

    slli x8, x8, 16       # Desloca todos os bits para a esquerda, x8 = 0x
       FFFF0000
    ori x8, x8, 0xFFFF    # Define todos os bits, x8 = 0xFFFFFFFF
```

9 Programa 9

```
.text
.globl _start

_start:
    # Inicializacao do valor em x8
    ori x8, x0, 0x12345678

    # Extrair os valores individuais
    andi x9, x8, 0xFF          # x9 = 0x78 (extrai os 8 bits menos
        significativos)
    srl x8, x8, 8              # Desloca x8 para a direita 8 bits
    andi x10, x8, 0xFF         # x10 = 0x56 (extrai os proximos 8 bits)
    srl x8, x8, 8              # Desloca x8 para a direita 8 bits
    andi x11, x8, 0xFF         # x11 = 0x34 (extrai os proximos 8 bits)
    srl x8, x8, 8              # Desloca x8 para a direita 8 bits
    andi x12, x8, 0xFF         # x12 = 0x12 (extrai os 8 bits mais
        significativos)
```