Exercício 3 – Guilherme Meyer

Programa 10

```
.data
x1: .word 15
x2: .word 25
x3: .word 13
x4: .word 17

.text
.globl _start
_start:

   addi s1, zero, 1



   la t0, x1
   la t1, x2
   la t2, x3
   la t3, x4

   lw s2, 0(t0)
   lw s3, 0(t1)
   lw s4, 0(t2)
   lw s5, 0(t3)
```

```
  li a7, 10
  ecall
```

Programa 11:

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

.text
.globl _start

_start:

  lw t0, x1
  lw t1, x2
  lw t2, x3
  lw t3, x4

  # soma dos valores
  add t4, t0, t1
  add t4, t4, t2
  add t4, t4, t3 # (53 + 17 = 70)
```

```
  sw t4, soma, t5



  j end

end:
  j end
```

Programa 12:

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

.text
.globl main

main:

  lw t0, x
  lw t1, z



  slli t2, t0, 7    # x << 7 (128x)
  sub  t2, t2, t0   # 128x - x = 127x



  slli t3, t1, 6    # z << 6 (64z)
```

```
  add  t3, t3, t1   # 64z + z = 65z



  sub t4, t2, t3
  addi t4, t4, 1

  sw t4, y



  j end

end:
  j end
```

Programa 13:

```
.data
A: .word 1, 3, 5, 7, 9
B: .word 2, 4, 6, 8, 10

.text
.globl main

main:

  la t0, A
  la t1, B
```

```
# A[0] = B[0] * 1 + A[0]
lw t2, 0(t1)
lw t3, 0(t0)
add t4, t2, t3
sw t4, 0(t0)

# A[1] = B[1] * 2 + A[1]
lw t2, 4(t1)
lw t3, 4(t0)
slli t5, t2, 1
add t4, t5, t3
sw t4, 4(t0)

# A[2] = B[2] * 3 + A[2]
lw t2, 8(t1)
lw t3, 8(t0)
slli t5, t2, 1
add t5, t5, t2
add t4, t5, t3
sw t4, 8(t0)

# A[3] = B[3] * 4 + A[3]
lw t2, 12(t1)
lw t3, 12(t0)
slli t5, t2, 2
add t4, t5, t3
sw t4, 12(t0)
```

```
    # A[4] = B[4] * 5 + A[4]
    lw t2, 16(t1)
    lw t3, 16(t0)
    slli t5, t2, 2
    add t5, t5, t2
    add t4, t5, t3
    sw t4, 16(t0)



    j end

end:
    j end
```

Programa 14:

```
.text
.globl main

main:

    li s0, 10
    li s1, 1
    li s2, 0

    bgt s0, s1, xmaior

    mv s2, s1
    j end_if
```

```
xmaior:

   mv s2, s0

end_if:

   j end_if
```

Programa 15:

```
   .text
   .globl main

main:
   li s0, 20      # x = 20
   li s1, 35      # y = 35
   li s2, 0       # m = 0

   bgt s0, s1, x_maior

   # else: m = y
   mv s2, s1
   j fim

x_maior:
   # if: m = x
   mv s2, s0
```

```
fim:
   # loop infinito só pra não encerrar direto
   j fim
```

Programa 16:

```
/*int exemploC(int x) {
   switch(x) {
      case 0: return 10;
      case 1: return 20;
      case 2: return 30;
      case 3: return 40;
      default: return -1;
   }
}*/
.text
.globl exemploAssembly

exemploAssembly:


   li t0, 0
   beq a0, t0, case0

   li t0, 1
   beq a0, t0, case1


   li t0, 2
```

```
    beq a0, t0, case2

    li t0, 3
    beq a0, t0, case3

    j default

case0:
    li a0, 10
    ret

case1:
    li a0, 20
    ret

case2:
    li a0, 30
    ret

case3:
    li a0, 40
    ret

default:
    li a0, -1
    ret
```

Programa 17:

```
.text
.globl main

main:

  li s0, 25
  li s1, 0



  li t0, 10
  beq s0, t0, case10

  li t0, 25
  beq s0, t0, case25



  j default

case10:
  li s1, 10
  j fim_switch

case25:
  li s1, 25
  j fim_switch

default:
```

```
    li s1, 0

fim_switch:

    j fim_switch
```

Programa 18:

```
    .text
    .globl main

main:
    li s0, 8
    li s1, 0

loop:
    li t0, 8
    bne s0, t0, fim

    mv s1, s0
    addi s0, s0, 1

    j loop

fim:
    j fim
```

Programa 19:

```
    .data
```

```
A:  .word 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

    .text
    .globl main

main:
    li s0, 0
    la s1, A

loop:
    li t3, 10
    bge s0, t3, fim

    slli t0, s0, 2

    add t4, s1, t0
    lw t1, 0(t4)
    addi t1, t1, 1
    sw t1, 0(t4)

    addi s0, s0, 1

    j loop

fim:
    j fim
```

Programa 20:

```
  .data
A:  .word 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

  .text
  .globl main

main:
  li s0, 0
  la s1, A

loop:
  li t5, 10
  bge s0, t5, fim

  andi t0, s0, 1     # t0 = i % 2
  bnez t0, else  # se i % 2 != 0, vai pro else

  # --- IF (i % 2 == 0) ---
  slli t1, s0, 2     # t1 = i * 4
  add t2, s1, t1     # t2 = &A[i]
  lw t3, 0(t2)       # t3 = A[i]

  addi t4, s0, 1     # t4 = i + 1
  slli t4, t4, 2     # t4 = (i + 1) * 4
  add t4, s1, t4     # t4 = &A[i+1]
  lw t4, 0(t4)       # t4 = A[i+1]
```

```
    add t3, t3, t4      # t3 = A[i] + A[i+1]
    sw t3, 0(t2)        # A[i] = t3

    j incremento

else:
    slli t1, s0, 2      # t1 = i * 4
    add t2, s1, t1      # t2 = &A[i]
    lw t3, 0(t2)        # t3 = A[i]
    slli t3, t3, 1      # t3 = A[i] * 2
    sw t3, 0(t2)        # A[i] = t3

incremento:
    addi s0, s0, 1      # i++
    j loop

fim:
    j fim
```

Programa 21:

```
    .data
A:  .word 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

    .text
    .globl main

main:
    li s0, 0
```

```
    la s1, A

loop:
    li t5, 10
    bge s0, t5, fim

    andi t0, s0, 1      # t0 = i % 2
    bnez t0, else  # se i % 2 != 0, vai pro else

    # --- IF (i % 2 == 0) ---
    slli t1, s0, 2      # t1 = i * 4
    add t2, s1, t1      # t2 = &A[i]
    lw t3, 0(t2)        # t3 = A[i]

    addi t4, s0, 1      # t4 = i + 1
    slli t4, t4, 2      # t4 = (i + 1) * 4
    add t4, s1, t4      # t4 = &A[i+1]
    lw t4, 0(t4)        # t4 = A[i+1]

    add t3, t3, t4      # t3 = A[i] + A[i+1]
    sw t3, 0(t2)        # A[i] = t3

    j incremento

else:
    slli t1, s0, 2      # t1 = i * 4
    add t2, s1, t1      # t2 = &A[i]
    lw t3, 0(t2)        # t3 = A[i]
```

```
    slli t3, t3, 1      # t3 = A[i] * 2
    sw t3, 0(t2)        # A[i] = t3

incremento:
    addi s0, s0, 1      # i++
    j loop

fim:
    j fim
```