CENTRO TECNOLÓGICO
Universidade Federal de Santa Catarina

Relatório Laboratório 01

Atividades via Console e via Digital Lab

Aluna(s): Jessica Regina dos Santos e Myllena da Conceição Correa
Disciplina: INE5411 - Organização de Computadores I
Professor: Marcelo Daniel Berejuck

- **Objetivos**

  Atividades via Console:

  1. Desenvolver um programa em Assembly do MIPS para ser executado no simulador MARS e que realize as operações de alto nível:

  $$a = b + 35$$
  $$c = d^3 - (a + e)$$

  2. Utilizar chamadas de sistema (*syscall*) para realizar a entrada (teclado) e saída (tela) dos dados.
  3. Contar e comparar quantidade de linhas de código entre programas.

  Atividade via Digital Lab:

  1. Implementar um programa em Assembly do MARS que escreva sequencialmente os números 0, 1, 2, 3, 4, 5, 6, 7, 8 e 9 em um dos displays de sete segmentos disponíveis na ferramenta Digital Lab Sim.
  2. Implementar um programa em Assembly para o MARS que leia o teclado alfanumérico e mostre em um dos displays de sete segmentos o valor da tecla pressionada (de 0 até f).

Atividade 1

1)

1. Inicialização e carregamento de variáveis
   Carregamos os endereços das variáveis b, e e d (Load Address).
   Lemos os valores de d, b e e da memória e armazenamento nos registradores $t0, $t8 e $t9 (Load Word).

```
 7  .text
 8  main:
 9      # Carrega valores da memoria
10      la $s0, b
11      la $s1, e
12      la $s2, d
13      lw $t0, 0($s2)
14      lw $t8, 0($s0)
15      lw $t9, 0($s1)
```

2. Cálculo de d^2
   Realizamos a multiplicação de d por ele mesmo (d^2) manualmente, bit a bit.
   A lógica é a seguinte:
   - Para cada bit de d (de 0 a 3), verificamos se o bit está ligado.
   - Se estiver ligado, acumula-se d deslocado para a esquerda por i posições (equivalente a multiplicar por 2^i).
   - O valor resultante da multiplicação bit a bit é acumulado em $t1.

   Exemplo:
   Se d = 4 (0100), então apenas o bit 2 está ligado.
   Resultado:
   $d^2 = d << 2 = 4 \times 4 = 16$.

```
20      # Bit 0
21      li $t3, 1          # Máscara para bit 0
22      and $t4, $t0, $t3  # Isola bit 0
23      sub $t7, $zero, $t4 # Cria máscara de seleção (0 ou 0xFFFFFFFF)
24      sll $t5, $t0, 0    # d * 2? = d * 1
25      and $t6, $t5, $t7  # Seleciona valor se bit=1
26      add $t1, $t1, $t6  # Acumula
27
28      # Bit 1
29      li $t3, 2          # Máscara para bit 1
30      and $t4, $t0, $t3  # Isola bit 1
31      srl $t4, $t4, 1    # Desloca para bit 0
32      sub $t7, $zero, $t4 # Cria máscara de seleção
33      sll $t5, $t0, 1    # d * 2¹ = d * 2
34      and $t6, $t5, $t7  # Seleciona valor se bit=1
35      add $t1, $t1, $t6  # Acumula
```

```
    # Bit 2
    li $t3, 4           # Máscara para bit 2
    and $t4, $t0, $t3   # Isola bit 2
    srl $t4, $t4, 2     # Desloca para bit 0
    sub $t7, $zero, $t4 # Cria máscara de seleção
    sll $t5, $t0, 2     # d * 2² = d * 4
    and $t6, $t5, $t7   # Seleciona valor se bit=1
    add $t1, $t1, $t6   # Acumula

    # Bit 3
    li $t3, 8           # Máscara para bit 3
    and $t4, $t0, $t3   # Isola bit 3
    srl $t4, $t4, 3     # Desloca para bit 0
    sub $t7, $zero, $t4 # Cria máscara de seleção
    sll $t5, $t0, 3     # d * 2³ = d * 8
    and $t6, $t5, $t7   # Seleciona valor se bit=1
    add $t1, $t1, $t6   # Acumula
```

3. Cálculo de $d^3 = d^2 \times d$
   Usamos $t1 (que contém d^2) e multiplicamos novamente por d, também bit a bit, com a mesma lógica anterior. O resultado da multiplicação é acumulado em $t2, que conterá d^3.

```
    # Bit 0
    li $t3, 1           # Máscara para bit 0
    and $t4, $t0, $t3   # Isola bit 0
    sub $t7, $zero, $t4 # Cria máscara de seleção
    sll $t5, $t1, 0     # d² * 2? = d² * 1
    and $t6, $t5, $t7   # Seleciona valor se bit=1
    add $t2, $t2, $t6   # Acumula

    # Bit 1
    li $t3, 2           # Máscara para bit 1
    and $t4, $t0, $t3   # Isola bit 1
    srl $t4, $t4, 1     # Desloca para bit 0
    sub $t7, $zero, $t4 # Cria máscara de seleção
    sll $t5, $t1, 1     # d² * 2¹ = d² * 2
    and $t6, $t5, $t7   # Seleciona valor se bit=1
    add $t2, $t2, $t6   # Acumula
```

```
85      # Bit 2
86      li $t3, 4           # Máscara para bit 2
87      and $t4, $t0, $t3   # Isola bit 2
88      srl $t4, $t4, 2     # Desloca para bit 0
89      sub $t7, $zero, $t4 # Cria máscara de seleção
90      sll $t5, $t1, 2     # d² x 2² = d² x 4
91      and $t6, $t5, $t7   # Seleciona valor se bit=1
92      add $t2, $t2, $t6   # Acumula
93
94      # Bit 3
95      li $t3, 8           # Máscara para bit 3
96      and $t4, $t0, $t3   # Isola bit 3
97      srl $t4, $t4, 3     # Desloca para bit 0
98      sub $t7, $zero, $t4 # Cria máscara de seleção
99      sll $t5, $t1, 3     # d² x 2³ = d² x 8
.00     and $t6, $t5, $t7   # Seleciona valor se bit=1
.01     add $t2, $t2, $t6   # Acumula
```

4. Cálculo de $a = b + 35 + e$
   A variável a não está na memória, mas é construída como uma expressão intermediária: a = b + 35 + e.
   b já está em $t8 e e em $t9, então essa parte apenas soma.

```
103     ## a = b+35
104     addi $t8, $t8, 35
105     #a+e
106     add $t8, $t8, $t9
```

5. Cálculo final: c = d³ - a
   Subtraimos o valor acumulado de a (em $t8) do valor d³ (em $t2).
   O resultado c é armazenado em $s3.

```
107     ## c = d^3 - (a+e)
108     lw $s3, c
109     sub $s3, $t2, $t8
110
111 # O resultado final está em $s3
```

2) Adaptamos o programa da primeira questão.
Agora, o valor para as variáveis b, d e e devem ser fornecido pelo usuário (via teclado). A ordem de inserção é D , B, depois E (apenas valores *integer*).
Intervalo limitante: O valor de d deve estar no intervalo **0 ≤ d ≤ 15** para que a multiplicação bit a bit funcione corretamente.

```
2  .text
3  main:
4      li $v0, 5          # syscall: read_int
5      syscall
6      move $t0, $v0      # $t0 = d
7      move $t9, $t0      # salva d original em $t9 para uso depois (d³ = d² × d)
```

```
86      ####### Recebe B e E #######
87      li $v0, 5
88      syscall
89      move $t8, $v0       # $t8 = B
90
91      li $v0, 5
92      syscall
93      move $t9, $v0       # $t9 = E
```

O resultado final é apresentado no terminal.

```
102     ####### print resultado #######
103     li $v0, 1
104     move $a0, $s3
105     syscall
106
107     li $v0, 10
108     syscall
```

Resultados Encontrados

Questão 1) Para b = 2, d = 4, e = 1.

$a = b + 35$

$c = d^3 - (a + e)$

Logo,

$c = 26$

Name	Number	Value
$zero	0	0
$at	1	268500992
$v0	2	0
$v1	3	0
$a0	4	0
$a1	5	0
$a2	6	0
$a3	7	0
$t0	8	4
$t1	9	16
$t2	10	64
$t3	11	8
$t4	12	0
$t5	13	128
$t6	14	0
$t7	15	0
$s0	16	268500992
$s1	17	268501004
$s2	18	268501000
$s3	19	26
$s4	20	0
$s5	21	0
$s6	22	0
$s7	23	0
$t8	24	38
$t9	25	1
$k0	26	0
$k1	27	0
$gp	28	268468224
$sp	29	2147479548
$fp	30	0
$ra	31	0

Questão 2) Para b = 10, d = 6, e = 14.

$a = b + 35$

$c = d^3 - (a + e)$

Logo,

$c = 157$

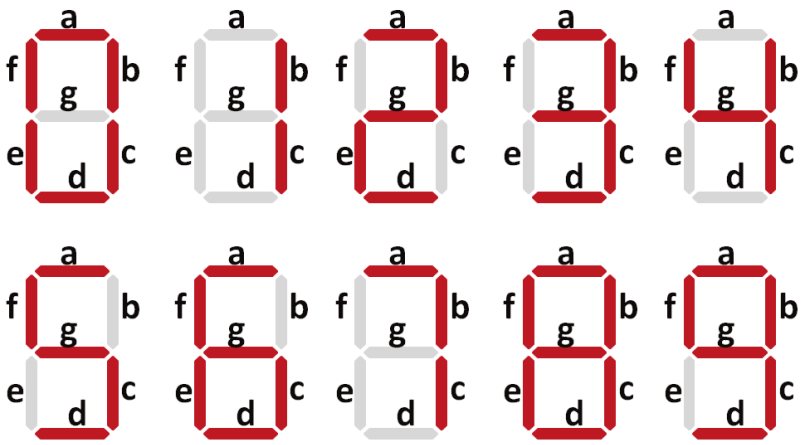
```
6
10
14
157
-- program is finished running --
```

Registers | Coproc 1 | Coproc 0

Name	Number	Value
$zero	0	0
$at	1	0
$v0	2	10
$v1	3	0
$a0	4	157
$a1	5	0
$a2	6	0
$a3	7	0
$t0	8	6
$t1	9	36
$t2	10	216
$t3	11	8
$t4	12	0
$t5	13	288
$t6	14	0
$t7	15	0
$s0	16	0
$s1	17	0
$s2	18	0
$s3	19	157
$s4	20	0
$s5	21	0
$s6	22	0
$s7	23	0
$t8	24	59
$t9	25	14
$k0	26	0
$k1	27	0
$gp	28	268468224
$sp	29	2147479548
$fp	30	0
$ra	31	0

3) Comparação quantidade de linhas: o programa da questão 1 possui 100 linhas. Já o programa da questão 2 possui 108 linhas.
No segundo programa, foram adicionadas 13 novas linhas para realizar a chamada de sistema corretamente e removidas 4 linhas de .data (valores de b, d, e agora são inseridos via teclado).

Atividade 2



1)

```
.data
```

Seção usada para declarar e inicializar variáveis (fica omitida para esse exercício).

```
.text
```

Seção que contém o código executável.

```
li $s0, 0xFFFF0010
```

Carregar o endereço de memória 0xFFFF0010 no registrador $s0 usando a instrução Load Immediate. O endereço corresponde ao endereço do display da direita no Digital Lab Sim.

```
Loop:
```

Marcador que define o início do loop.

```
li $t0, 63
```

Carregar o valor 63 no registrador $t0. Esse valor representa os bits necessários para exibir o número 0 no display de 7 segmentos (00111111b).

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

Armazenar o valor contido em $t0 no endereço de memória apontado por $s0. Isso escreve 63 = 00111111b = 0 no display.

```
li $t0, 6
```

Carregar o valor 6 no registrador $t0. Esse valor representa os bits necessários para exibir o número 1 no display de 7 segmentos (00000110).

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

Armazenar o valor contido em $t0 no endereço de memória apontado por $s0. Isso escreve 6 = 00000110b = 1 no display.

```
li $t0, 91
```
Carregar o valor 91 no registrador $t0. Esse valor representa os bits necessários para exibir o número 2 no display de 7 segmentos (1011011).

```
sw $t0, 0($s0)
```
Armazenar o valor contido em $t0 no endereço de memória apontado por $s0. Isso escreve o valor 91 = 1011011b = 2 no display.

```
li $t0, 79
```
Carregar o valor 79 no registrador $t0. Esse valor representa os bits necessários para exibir o número 3 no display de 7 segmentos (1001111b).

```
sw $t0, 0($s0)
```
Armazenar o valor contido em $t0 no endereço de memória apontado por $s0. Isso escreve o valor 79 = 1001111b = 3 no display.

```
li $t0, 102
```
Carregar o valor 102 no registrador $t0. Esse valor representa os bits necessários para exibir o número 4 no display de 7 segmentos (1100110b).

```
sw $t0, 0($s0)
```
Armazenar o valor contido em $t0 no endereço de memória apontado por $s0. Isso escreve o valor 102 = 1100110b = 4 no display.

```
li $t0, 109
```
Carregar o valor 109 no registrador $t0. Esse valor representa os bits necessários para exibir o número 5 no display de 7 segmentos (1101101b).

```
sw $t0, 0($s0)
```
Armazenar o valor contido em $t0 no endereço de memória apontado por $s0. Isso escreve o valor 109 = 1101101b = 5 no display.

```
li $t0, 125
```
Carregar o valor 125 no registrador $t0. Esse valor representa os bits necessários para exibir o número 6 no display de 7 segmentos (1111101b).

```
sw $t0, 0($s0)
```
Armazenar o valor contido em $t0 no endereço de memória apontado por $s0. Isso escreve o valor 125 = 1111101b = 6 no display.

```
li $t0, 7
```

Carregar o valor 7 no registrador $t0. Esse valor representa os bits necessários para exibir o número 7 no display de 7 segmentos (0000111b).

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

Armazenar o valor contido em $t0 no endereço de memória apontado por $s0. Isso escreve o valor 7 = 0000111b = 7 no display.

```
li $t0, 127
```

Carregar o valor 127 no registrador $t0. Esse valor representa os bits necessários para exibir o número 8 no display de 7 segmentos (1111111b).

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

Armazenar o valor contido em $t0 no endereço de memória apontado por $s0. Isso escreve o valor 127 = 1111111b = 8 no display.

```
li $t0, 111
```

Carregar o valor 111 no registrador $t0. Esse valor representa os bits necessários para exibir o número 9 no display de 7 segmentos (1101111b).

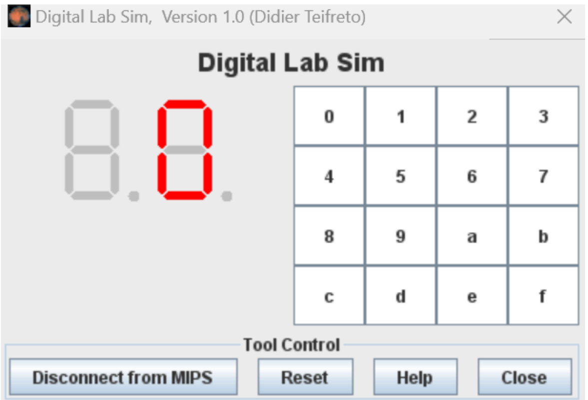
```
sw $t0, 0($s0)
```

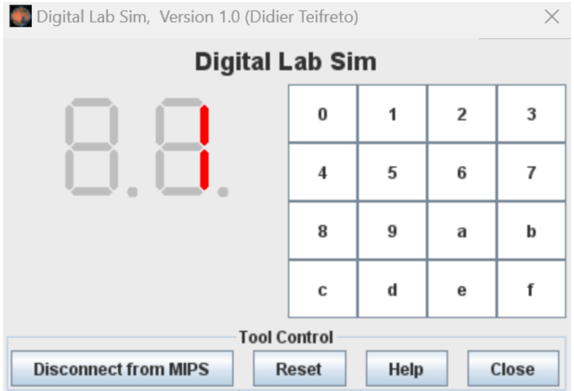
Armazenar o valor contido em $t0 no endereço de memória apontado por $s0. Isso escreve o valor 111 = 1101111bb = 9 no display.

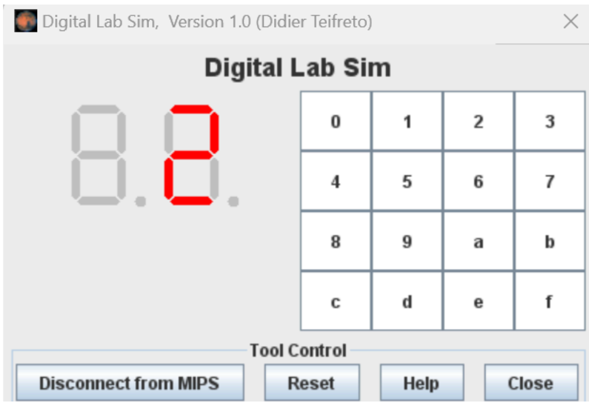
```
j Loop
```

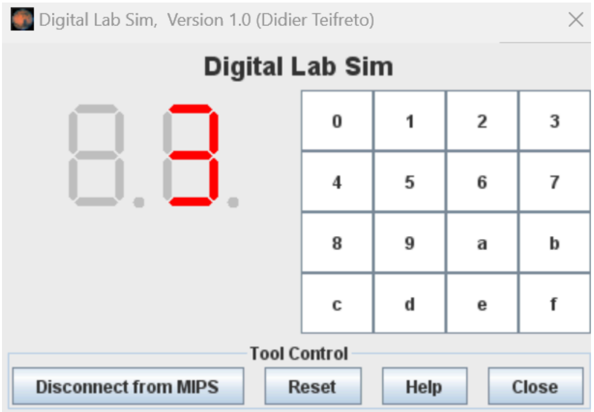
Salto de volta para o marcador Loop reiniciando a contagem.

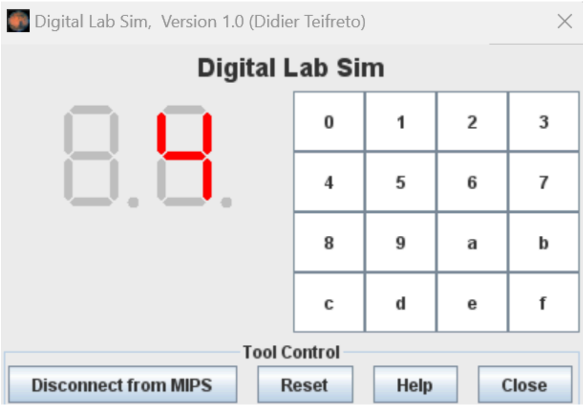
Prints de tela:

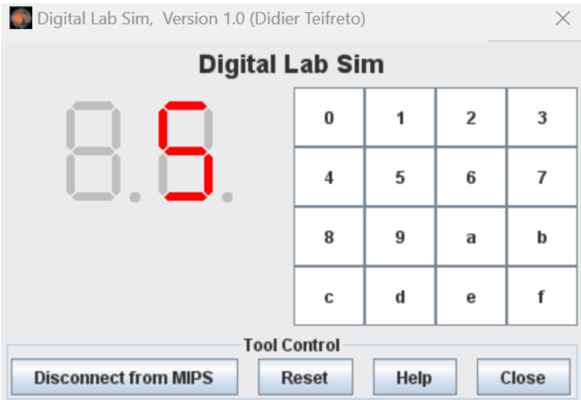


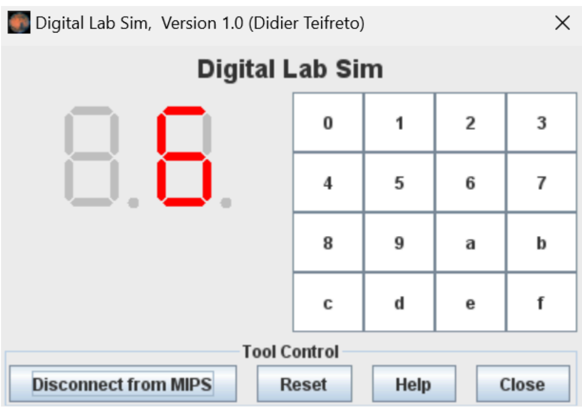


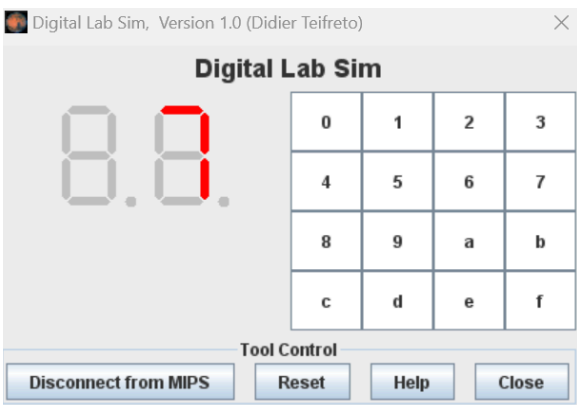


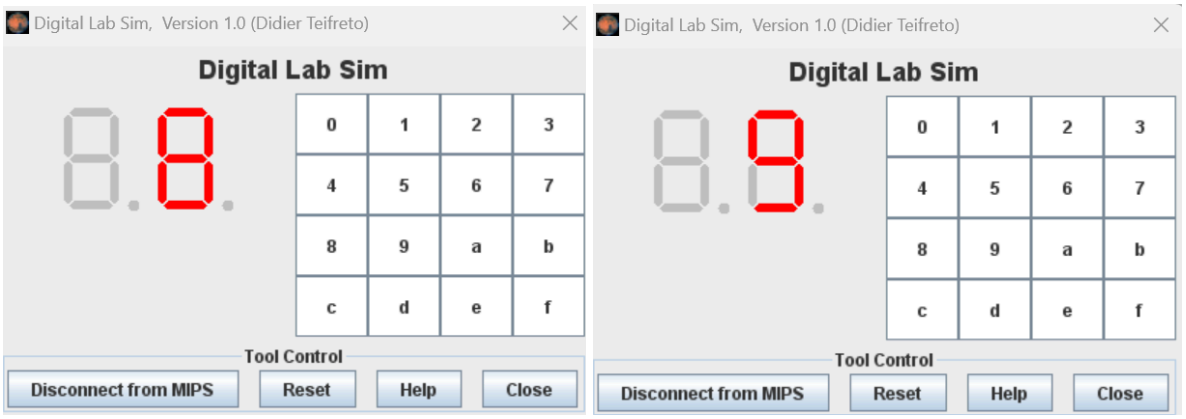












2)

- O programa está estruturado em torno de um loop principal “Loop”.
- Dentro desse loop, são verificadas as teclas pressionadas em cada linha do teclado para que os valores correspondentes sejam exibidos no display.
- De LINHA1 até LINHA4, lemos cada linha do teclado para verificar se alguma tecla é pressionada. Quando um botão é pressionado, o programa direciona para Display0 até DisplayF (de acordo com o botão escolhido). O valor para acender os LEDs do display é carregado e escrito na porta de controle do display. Se nenhum botão é pressionado, ocorre um tratamento (o programa desliga todos os LEDs do display).
- Registradores de salvamento são utilizados para armazenar os endereços de comunicação com o display e o teclado.

Explicação por blocos:

`ler_linha1, ler_linha2, ler_linha3, ler_linha4` são os valores usados para determinar qual linha do teclado será lida.
`sem_led` é o valor usado para desligar todos os LEDS do display.

```
1  .data
2
3  # valores para ler cada linha
4  ler_linha1: .byte 1
5  ler_linha2: .byte 2
6  ler_linha3: .byte 4
7  ler_linha4: .byte 8
8  sem_led: .byte 0 # valor para nao mostrar nenhum led
```

`led_0, led_1, led_2,... led_F` são valores para acender os LEDs correspondentes para cada dígito hexadecimal.

```
10  # valores para acender os leds de cada numero
11  led_0: .byte 63
12  led_1: .byte 6
13  led_2: .byte 91
14  led_3: .byte 79
15  led_4: .byte 102
16  led_5: .byte 109
17  led_6: .byte 125
18  led_7: .byte 7
19  led_8: .byte 127
20  led_9: .byte 111
21  led_A: .byte 119
22  led_B: .byte 124
23  led_C: .byte 57
24  led_D: .byte 94
25  led_E: .byte 121
26  led_F: .byte 113
```

Carregar os endereços específicos informados pelo Mars nos registradores.

```
28  .text
29
30  li $s1, 0xFFFF0010 # armazena em $s1 o endereco do display da direita do Digital Lab Sim
31  li $s2, 0xFFFF0012 # armazena em $s1 o endereco que ordena qual linha checar
32  li $s3, 0xFFFF0014 # armazena em $s3 o endereco que reccebe linha e coluna do botao
```

Blocos LINHA1, LINHA2, LINHA3, LINHA4:
Ler as linhas do teclado e verificar qual botão foi pressionado em cada linha.
`lb $t2, ler_linha[i]` carrega o valor correspondente à linha atual do teclado no registrador.
`sb $t2, 0($s2)` escreve esse valor no endereço de controle que seleciona qual linha do teclado será lida.
`lb $t3, 0($s3)` lê o valor pressionado na linha atual do teclado.
`beqz $t3, LINHA[i+1]`, se nenhum botão for pressionado, pular para a próxima linha.
`beq $t3, valor, Display[X]`, se um botão específico for pressionado, pula para o código de exibição correspondente.

```
36  LINHA1:
37  lb $t2,ler_linha1 # escreve em $t2 comando para ler a 1ª linha
38  sb $t2, 0($s2) # passa o valor de $t2 para endereco de comandos
39  lb $t3, 0($s3) # poe linha-coluna pressionada em $t3
40  beqz $t3, LINHA2
41
42  # testa cada botao da linha 1
43  beq $t3, 0x11, Display0
44  beq $t3, 0x21, Display1
45  beq $t3, 0x41, Display2
46  beq $t3, 0xffffff81, Display3
```

Blocos Display0 até DisplayF:
Esses blocos escrevem os valores correspondentes nos LEDS do display de 7 segmentos, dependendo do botão pressionado.
`lb $t1, led_X` é carregado o valor correspondente ao dígito ou letra a ser exibido nos LEDs.
`sb $t1, 0($s1)` é escrito esse valor no endereço que controla os LEDs do display.

```
89  #escreve 0 no display
90  Display0: lb $t1, led_0
91  sb $t1, 0($s1)
92  j Loop
93
94  #escreve 1 no display
95  Display1: lb $t1, led_1
96  sb $t1, 0($s1)
97  j Loop
98
99  #escreve 2 no display
100 Display2: lb $t1, led_2
101 sb $t1, 0($s1)
102 j Loop
103
104 #escreve 3 no display
105 Display3: lb $t1, led_3
106 sb $t1, 0($s1)
107 j Loop
```

Caso nenhum botão tenha sido pressionado, desliga todos os LEDs do display.

```
85  SemDisplay: lb $t1, sem_led # no caso em que nenhum botao tenha sido pressionado, desliga o display e sai do loop
86  sb $t1, 0($s1) # passa o valor de $t1 para o display da direita
87  j Exit
```

Prints de tela: