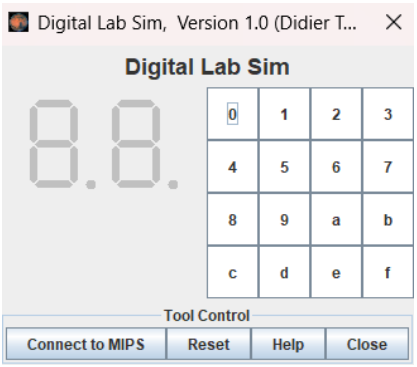
Relatório - Laboratório 02
Disciplina: INE5411 – Organização de Computadores I
Integrantes: Lucas Pastre de Souza e Rodrigo Martins dos Santos.

Introdução:

O presente relatório busca mostrar o funcionamento dos exercícios 1 e 2, referentes à segunda atividade de laboratório da disciplina Organização e Arquitetura de Computadores do curso de Ciências da Computação na Universidade Federal de Santa Catarina (UFSC).

Exercício 1:

O intuito do exercício 1 do Laboratório 02 é a implementação de um programa em Assembly para o MARS que escreva os números de 0 à 9 em um dos displays de sete segmentos da ferramenta Digital Lab Sim. Segue imagem da ferramenta:



Para isso, é necessário mapear os números de 0 à 9 para um vetor como demonstra o código abaixo:

```
1  .data
2      display: .byte 0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07, 0x7F, 0x6F
```

Em seguida, é necessário carregar o endereço do nosso vetor “display’ para um registrador $s0, também iremos carregar o display da esquerda para um registrador $s1 e o da direita para um registrador $s2. Após isso, carregaremos um valor máximo (10) em um registrador $t0 para ser usado mais tarde e carregaremos um contador “i” em registrador $t1.

```
4   .text
5           la $s0, display # Carrega o endereço do array de segmentos
6           addi $s1, $zero, 0xFFFF0011 # Endereço do display da esquerda
7           addi $s2, $zero, 0xFFFF0010 # Endereço do display da direita
8
9           addi $t0, $zero, 10     # Número máximo (9)
10          addi $t1, $zero, 0      # Inicializa o contador i
```

Para deixar o display mais “apresentável” no Digital Sim deixaremos o display da esquerda com o valor fixado em 0, como mostra o código abaixo:

```
12  # Exibir o valor fixo 0 no display da esquerda
13          addi $t3, $zero 0x3F
14          sb $t3, 0($s1)
```

Em seguida, iniciaremos o loop que imprime a sequência de números no display da direita no Digital Lab Sim, para isso iremos checar se o valor de $t1 (i) é menor do que o valor máximo armazenado em $t0 e carregaremos em um registrador $t2 o valor do segmento correspondente ao contador i ($t1) , logo após escreveremos o dado armazenado em $t2 no display da direita ($s2) e adicionaremos 1 ao valor de $t1, por fim inicializaremos uma variável no registrador $t4 para ser um “delay”, para que a transição dos números no display possa ser visível ao olho humano. Segue o trecho de código abaixo:

```
16  loop:
17          beq $t1, $t0, end           # Se t1 == 10, sai do loop
18
19  # Exibir o valor da unidade no display da direita
20
21          lb $t2, display($t1)        # Carrega o valor do segmento correspondente à unidade
22          sb $t2, 0($s2)              # Escreve o valor da unidade no display da direita
23
24          addi $t1, $t1, 1            # Incrementa o contador i
25
26          addi $t4, $zero, 250000
```

Por fim, implementamos um loop chamado “delay” para decrementar o delay, primeiro subtraímos 1 do valor registrado em $t4 e usamos uma função “branch if not equal” (bne) entre o valor de $t4 e o valor do registrador $zero, com um label para o próprio delay, após isso nós damos um “jump” para o loop onde o ciclo se repete até que o contador (i) seja igual a 10 (valor máximo) e o programa pule para o label “end’ onde o código se encerra. Segue o código abaixo:

```
27  delay:
28          addi $t4, $t4, -1
29          bne $t4, $zero, delay
30
31          j loop
32
33  end:
34          li $v0, 10
35          syscall
36
```

Exercício 2:

O objetivo do exercício 2 é criar um programa que lê o teclado alfanumérico do Digital Lab Sim no Mars e mostra os valores (de 0 a f) no display de 7 segmentos dessa ferramenta.

A ideia é ver esse teclado como uma matriz, varrendo linha por linha para ver se alguma tecla foi pressionada, e então mostrar o padrão correspondente no display.

Primeiramente, declaramos os endereços necessários para o programa funcionar, de acordo com as instruções de “Help” no Mars. Então na seção de texto, carregamos esses endereços em dois registradores.

```
1   .data
2           end_linha:.word 0xFFFF0012
3           end_teclado:.word 0xFFFF0014
4           end_display:.word 0xFFFF0010
5
6   .text
7
8   main:
9           # Carrega os endereços necessários
10
11          lw $s2, end_linha
12          lw $s3, end_teclado
13
```

A primeira parte do código dentro do rótulo main é responsável pela leitura das teclas do teclado. O processo ocorre em um loop onde cada linha do teclado é ativada uma por vez, e em seguida é feita a leitura do valor da tecla pressionada. Conforme as imagens do código abaixo:

```
14          # Condição para cada linha da "matriz" no digital lab sim
15
16          add $s0, $zero, 1       # 0001 em binário (linha 1)
17          sb  $s0, 0($s2)
18          lw  $s1, 0($s3)
19          bne $s1, $zero, display
20
21          add $s0, $zero, 2       #0010 em binario (linha 2)
22          sb  $s0, 0($s2)
23          lw  $s1, 0($s3)
24          bne $s1, $zero, display
25
```

```
26          add $s0, $zero, 4       #0100 em binário (linha 3)
27          sb  $s0, 0($s2)
28          lw  $s1, 0($s3)
29          bne $s1, $zero, display
30
31          add $s0, $zero, 8       #1000 em binário (linha 4)
32          sb  $s0, 0($s2)
33          lw  $s1, 0($s3)
34          bne $s1, $zero, display
35
```

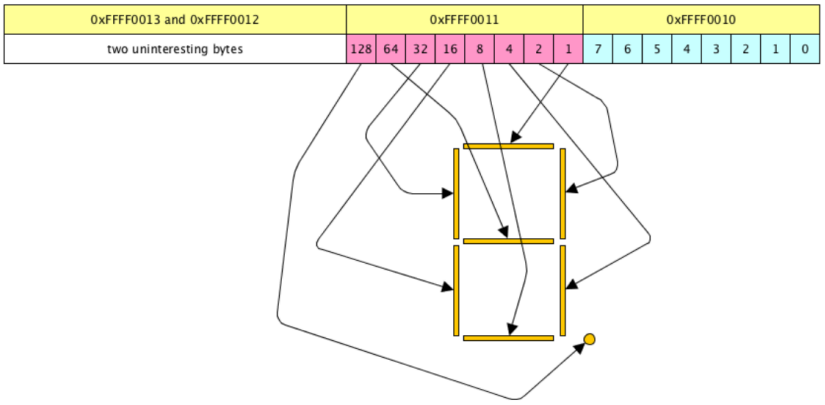
Quando uma tecla é detectada, passamos para a label display, onde são utilizadas instruções branch if equal (beq) para verificar qual tecla foi pressionada, de acordo com a comparação com o conteúdo de $s1. Veja no código abaixo:

```
36  display:
37          lw $s4, end_display
38
39          beq $s1, 0x11, zero
40          beq $s1, 0x21, um
41          beq $s1, 0x41, dois
42          beq $s1, 0x81, tres
43          beq $s1, 0x12, quatro
44          beq $s1, 0x22, cinco
45          beq $s1, 0x42, seis
46          beq $s1, 0x82, sete
47          beq $s1, 0x14, oito
48          beq $s1, 0x24, nove
49          beq $s1, 0x44, alfa_a
50          beq $s1, 0x84, alfa_b
51          beq $s1, 0x18, alfa_c
52          beq $s1, 0x28, alfa_d
53          beq $s1, 0x48, alfa_e
54          beq $s1, 0x88, alfa_f
55          beq $s1, 0x3F, main
56
```

Cada valor de $s1 (como 0x11, 0x21, etc.) corresponde a uma combinação de tecla e linha. Dependendo do valor detectado, o programa "desvia" para a label que contém o código correspondente ao número ou letra a ser exibido.

Uma vez identificada a tecla, o código hexadecimal correspondente ao número ou letra é carregado no registrador $t0 e armazenado no endereço do display ($s4)

O valor 0x3F é o código que representa o número 0 no display de 7 segmentos, e assim por diante para os números e letras subsequentes. A imagem abaixo mostra como é formado o binário para determinado padrão em 7 segmentos. Para formar o “e”, por exemplo, temos que identificar o seu binário, que seria 0111 1011, e então transformá-lo para valor hexadecimal, ou seja 0x7B. Em seguida, temos também imagens do código para todas as labels necessárias, para cada padrão no display de 7 segmentos.




```
56
57          zero:
58                  li $t0, 0x3F
59                  sw $t0, 0($s4)
60                  j main
61
62          um:
63                  li $t0, 0x06
64                  sw $t0, 0($s4)
65                  j main
66
67          dois:
68                  li $t0, 0x5B
69                  sw $t0, 0($s4)
70                  j main
71
72          tres:
73                  li $t0, 0x4F
74                  sw $t0, 0($s4)
75                  j main
76
```

```
77          quatro:
78                  li $t0, 0x66
79                  sw $t0, 0($s4)
80                  j main
81
82          cinco:
83                  li $t0, 0x6D
84                  sw $t0, 0($s4)
85                  j main
86
87          seis:
88                  li $t0, 0x7D
89                  sw $t0, 0($s4)
90                  j main
91
92          sete:
93                  li $t0, 0x07
94                  sw $t0, 0($s4)
95                  j main
96
```

```
97          oito:
98                  li $t0, 0x7F
99                  sw $t0, 0($s4)
100                 j main
101
102         nove:
103                 li $t0, 0x6F
104                 sw $t0, 0($s4)
105                 j main
106
107         alfa_a:
108                 li $t0, 0x77
109                 sw $t0, 0($s4)
110                 j main
111
112         alfa_b:
113                 li $t0, 0x7C
114                 sw $t0, 0($s4)
115                 j main
116
```

```
117         alfa_c:
118                 li $t0, 0x39
119                 sw $t0, 0($s4)
120                 j main
121
122         alfa_d:
123                 li $t0, 0x5E
124                 sw $t0, 0($s4)
125                 j main
126
127         alfa_e:
128                 li $t0, 0x7B
129                 sw $t0, 0($s4)
130                 j main
131
132         alfa_f:
133                 li $t0, 0x71
134                 sw $t0, 0($s4)
135                 j main
136
```

Por fim, a label “end” implica no fim do programa.

```
137          end:
138                 li $v0, 10
139                 syscall
140
```

Segue uma imagem de como fica o display de 7 segmentos após pressionar a tecla “e”, por exemplo, com o programa em funcionamento: