RELATÓRIO LABORATÓRIO 2

ORGANIZAÇÃO DE COMPUTADORES I

Emanuelle Guse - 23100486

EXERCÍCIO 1

1. Inicialmente, foi criada uma variável responsável por armazenar os valores que cada número tem que passar para ser mostrado no display, de forma que ***bytes_array[i]*** representa ***i***.

```
.data
  bytes_array: .byte 0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07, 0x7F, 0x6F  # códigos do 0 ao 9
```

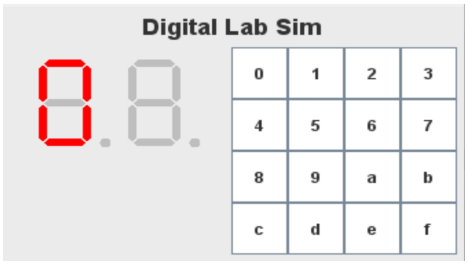
Data Segment

Address	Value (+0)	Value (+4)	Value (+8)
268500992	0x4f5b063f	0x077d6d66	0x00006f7f
268501024	0x00000000	0x00000000	0x00000000

2. Após isso, foram carregados os endereços do display esquerdo e direito, já sendo conhecidos previamente, uma vez que são sempre o mesmo. Eles foram armazenados nos registradores $s0 e $s1.

$s0	16	0xffff0011
$s1	17	0xffff0010

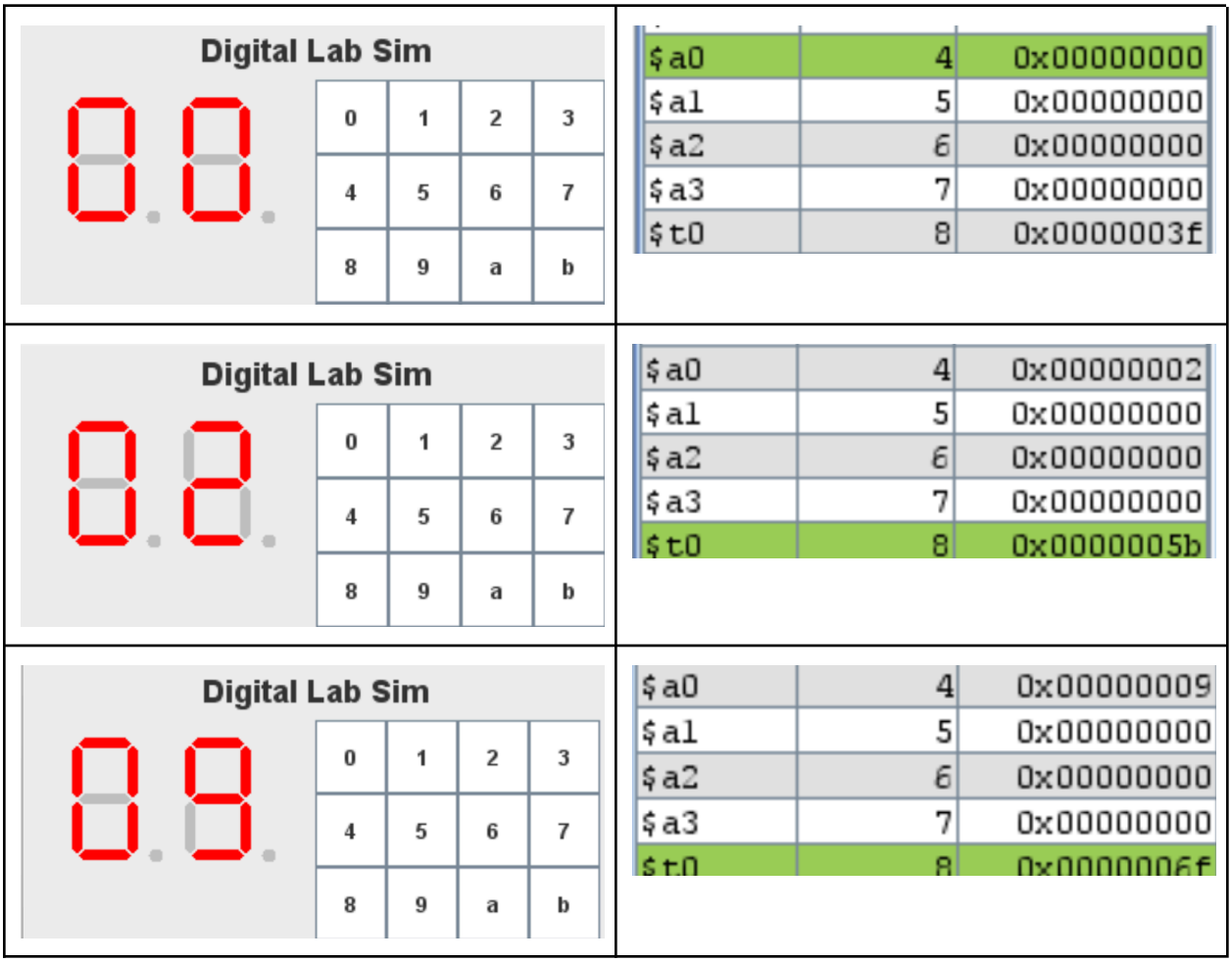
Além disso, definimos o display esquerdo como 0, já que isso nunca vai mudar. Para isso, passamos 0x3f (0) para um registrador temporário que é, por fim, passado para o valor do display.



3. São criados três marcadores: *loop*, *inner_loop* e *delay*.

3.1) Dentro de loop, a única coisa que ocorre é que o registrador $a0 é definido como zero. Ele servirá como nosso contador.

3.2) Já dentro do *inner_loop*, mais coisas acontecem. O registrador $t0 recebe o valor de *bytes_array* na posição atual, armazenada em $a0. Esse valor é passado para o display direito, no qual é escrito. Após isso, o contador é incrementado. O registrador $t1 recebe o valor 0, servindo como um contador do delay. Enquanto o contador $a0 for menor que 11 (números variando entre 0 e 9, já que ele é incrementado antes de fazer o branch), ele vai passar para o delay.

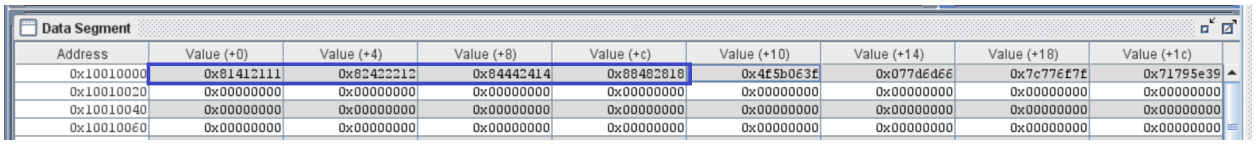


3.3) O delay serve, basicamente, para que a mudança fique mais visível a olho humano. Nele, ocorre uma contagem até um número específico e, enquanto for menor que ele, volta ao delay e *incrementa*. Quando isso não ocorre, ele faz o jump para loop, reiniciando o processo e zerando o contador $a0.

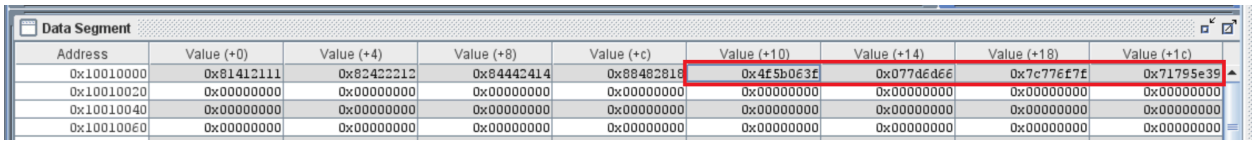
O código possui 33 linhas, considerando quebras, comentários e que pseudo-instruções são apenas uma. Ignorando comentários e transformando as pseudos-instruções em instruções, o exercício 1 conta com 1 linha em .text e 24 linhas em .data.

EXERCÍCIO 2

1. Inicialmente, assim como feito no primeiro exercício, os valores do array em cada posição são armazenados em uma variável. Dessa vez, também armazenamos as possíveis entradas do teclado.



valores de recebimento do teclado



valores do display

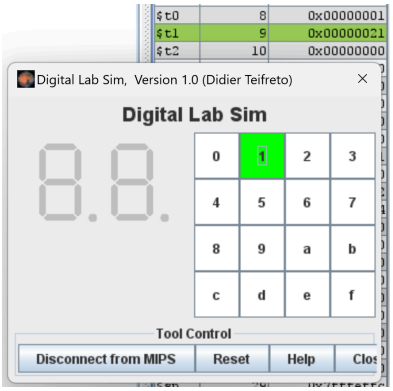
2. Carregamos os valores dos endereços dos displays direito e esquerdo, selecionador e leitor de teclado nos registradores de $s0 a $s3. Novamente, iniciamos o display da esquerda como zero.

$s0	16	0xffff0011
$s1	17	0xffff0010
$s2	18	0xffff0012
$s3	19	0xffff0014

3. Para esse exercício, dividimos em 6 marcadores: loop, row, key_pressed, key_search, change_display e delay.

3.1) O loop apenas inicializa o valor de $t0 como 1. Esse valor será usado como contagem das linhas.

3.2) Em row, começamos armazenando no selecionador de linhas a linha atual. A partir disso, é possível receber o valor da leitura, que é armazenado em $t1. Na imagem, vamos que $t0 está checando a primeira linha (0, 1, 2, 3) e conseguiu receber 21 (1) em $t1. Como $t1 não é mais zero, ele parte para o



key_pressed. Caso esse não fosse o caso, ele verificaria se a linha é a última possível (8) e, se não for, adiciona o próprio valor, dobrando ele. Se for, ele reinicia a busca em row, agora na próxima linha.

3.3) Se ele verificou que uma tecla foi pressionada (entrou em key_pressed), inicia em $t2 um contador com valor 0 e entra em *key_search*.

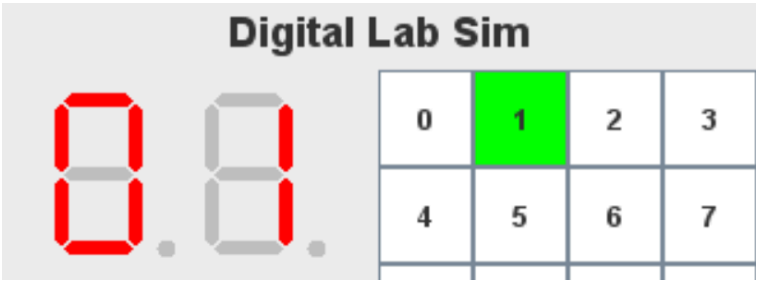
$t1	9	0x00000021
$t2	10	0x00000000
$t3	11	0x00000011

3.4) Inicialmente, é carregada em $t3 o valor das possíveis entradas na posição atual. Nesse exemplo, o primeiro valor será 0x11. É verificado se esse valor é igual ao pressionado ($t1). Se o valor da contagem chegar a 16 (testar todas as possibilidades de input), ele volta para key_pressed e reinicia a contagem e o processo. Para o contador sendo igual a 0, o valor será diferente, sendo incrementado e voltando para o loop em busca da próxima tecla.

$t1	9	0x00000021
$t2	10	0x00000001
$t3	11	0x00000021

Após isso, será feito o mesmo processo, mas, nesse exemplo, a comparação será verdadeira, passando para a mudança do display.

3.5) O registrador $t4 recebe o valor de entrada no display na posição em que foi encontrada a correspondência. Esse valor é passado para $s1 (display direito), fazendo o número aparecer na tela. Após isso, é iniciado em $t5 um contador, que servirá para a o delay.



3.6) O delay funciona da mesma forma que o anterior, apenas incrementando 1 a cada iteração até chegar em um determinado valor. Quando isso ocorre, ele volta para loop e reinicia o processo.

O código possui 51 linhas, considerando quebras, comentários e que pseudo-instruções são apenas uma. Ignorando comentários e transformando as pseudos-instruções em instruções, o exercício 2 conta com 2 linhas em .text e 36 linhas em .data.