Bài thực hành số 10

Lớp : 139365 – Học phần: Thực hành Kiến trúc máy tính

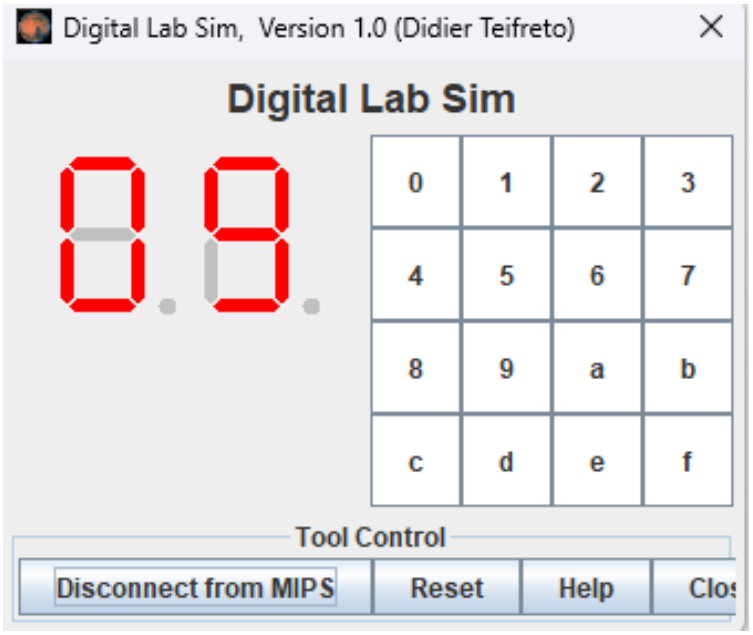
Họ và tên : Nguyễn Thị Thùy Dung MSSV : 20215009

Bài 1: Yêu cầu hiện ra 2 chữ số cuối mã số sinh viên (09)

```
.eqv SEVENSEG_LEFT 0xFFFF0011
.eqv SEVENSEG_RIGHT 0xFFFF0010
.text
main:
    li $a0, 0x3F
    jal SHOW_7SEG_LEFT
    nop
    li $a0, 0x6F
    jal SHOW_7SEG_RIGHT
    nop
exit:
    li $v0, 10
    syscall

SHOW_7SEG_LEFT:
    li $t0, SEVENSEG_LEFT
    sb $a0, 0($t0)
    nop
    jr $ra
    nop
SHOW_7SEG_RIGHT:
    li $t0, SEVENSEG_RIGHT
    sb $a0, 0($t0)
    nop
    jr $ra
    nop
```

Thực hiện gõ chương trình với công cụ MARS

- Kết quả:



- Giải thích :

  Gán địa chỉ của đèn bên trái là 0XFFFF0011, địa chỉ đèn bên phải là 0X FFFF0010

  Để hiện thị số 0 thì thanh g không được sáng

  . g f e d c b a

  0 0 1 1 1 1 1 1 -> 0x3F

  Để hiện thị số 9 thì thanh e không được sáng

  . g f e d c b a

  0 1 1 0 1 1 1 1 -> 0x6F

  Gán $a0 = 0x3F, sau đó nhảy đến chương trình con SHOW_7SEG_LEFT, khởi tạo $t0 có giá trị là địa chỉ của đèn bên trái, dùng sb (store byte) để lưu giá trị $a0 vào nơi có địa chỉ là giá trị của $t0, khi đó đèn bên trái hiện lên số 0 như kết quả

  Gán $a0 = 0x6F, sau đó nhảy đến chương trình con SHOW_7SEG_RIGHT, khởi tạo $t0 có giá trị là địa chỉ của đèn bên phải, dùng sb (store byte) để lưu giá trị $a0 vào nơi có địa chỉ là giá trị của $t0, khi đó đèn bên phải hiện lên số 9 như kết quả

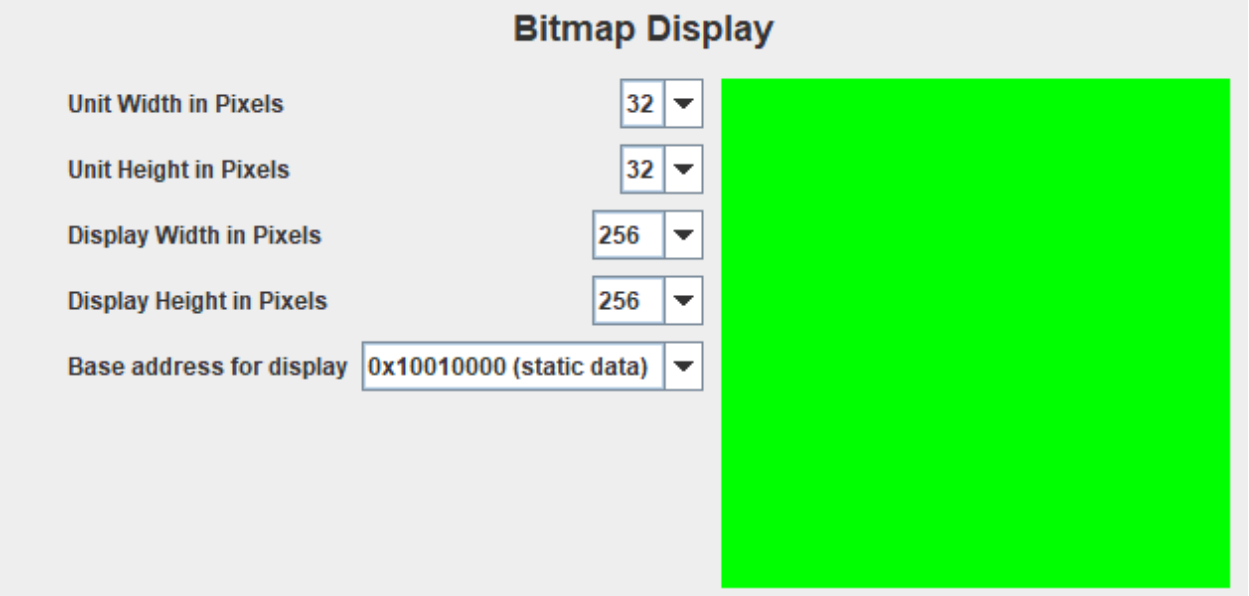
  Sau đó quay lại chương trình chính bằng lệnh jr $ra gán $v0 = 10 và gọi hệ thống để thoát chương trình

Bài 2:

```
1  .eqv MONITOR_SCREEN 0x10010000 #Dia chi bat dau cua bo nho man hinh
2  .eqv RED 0x00FF0000
3  .eqv GREEN 0x0000FF00
4  .eqv BLUE 0x000000FF
5  .eqv WHITE 0x00FFFFFF
6  .eqv YELLOW 0x00FFFF00
7  .text
8      li $k0, MONITOR_SCREEN #Nap dia chi bat dau cua man hinh
9      li $a0, GREEN
10     li $s0,0
11     li $s1,256
12 scan_one_row:
13     add $t0,$k0,$s0
14     sw $a0,0($t0)
15     addi $s0,$s0,4
16     bne $s0,$s1,scan_one_row
17
```

Thực hiện gõ chương trình với công cụ MARS

- Kết quả :



- Giải thích:
  Khởi tạo địa chỉ bộ nhớ mà hình là 0x10010000, khởi tạo các màu cơ bản là : red (0x00FF0000), green(0x0000FF00), blue (0x000000FF),white (0x00FFFFFF), yellow (0x00FFFF00)

Gán $k0 bằng địa chỉ bắt đầu màn hình (0X10010000), gán $a0 có giá trị màu xanh (0X0000FF00), $s0 = 0, $s1 = 256
Thực hiện vòng lặp scan_one_row: $t0 = $k0 + $s0 (bằng địa chỉ đầu màn hình cộng với biến chạy $s0), sau đó sw (store word) $a0 vào nơi có địa chỉ là giá trị của thanh ghi $t0 lưu giữ thì ô đó sẽ có màu xanh . Cứ thực hiện vòng lặp cho đến khi $s0 = $s1 = 256(*). Như vậy cả Bitmap sẽ có màu xanh như kết quả trên
(*): Ở Bitmap Display trên được khởi tạo : Unit Width in Pixel là 32 tức là 1 ô có chiều rộng 32 bit (4 byte), Unit Height in Pixels là 32 tức là 1 ô có chiều cao 32 bit (4 byte), Display Width : 256 tức là chiều rộng là 256 bit suy ra có 8 ô

Bài 3:

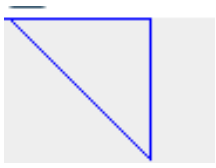
```
1  .eqv HEADING 0xffff8010
2  .eqv MOVING 0xffff8050
3  .eqv LEAVETRACK 0xffff8020
4  .eqv WHEREX 0xffff8030
5  .eqv WHEREY 0xffff8040
6  .text
7  main:
8     jal TRACK
9     nop
10    addi $a0, $zero, 90
11    jal ROTATE
12    nop
13    jal GO
14    nop
15 sleep1:
16    addi $v0,$zero,32
17    li $a0,3000
18    syscall
19    jal UNTRACK
20    nop
21    jal TRACK
22    nop
23 goDOWN:
24    addi $a0, $zero, 180
25    jal ROTATE
26    nop
```

```
27  sleep2:
28      addi $v0,$zero,32
29      li $a0,3000
30      syscall
31      jal UNTRACK
32      nop
33      jal TRACK
34      nop
35  goLEFT:
36      addi $a0, $zero, 315
37      jal ROTATE
38      nop
39
40  sleep3:
41      addi $v0,$zero,32
42      li $a0,4000
43      syscall
44      jal UNTRACK
45      nop
46
47
48
49  end_main:
50  GO: li $at, MOVING
51   addi $k0, $zero,1
52   sb $k0, 0($at)
```

```
53   nop
54   jr $ra
55   nop
56
57  STOP: li $at, MOVING
58   sb $zero, 0($at)
59   nop
60   jr $ra
61   nop
62
63  TRACK: li $at, LEAVETRACK
64   addi $k0, $zero,1
65   sb $k0, 0($at)
66   nop
67   jr $ra
68   nop
69  UNTRACK:li $at, LEAVETRACK
70   sb $zero, 0($at)
71   nop
72   jr $ra
73   nop
74  ROTATE: li $at, HEADING
75   sw $a0, 0($at)
76   nop
77   jr $ra
78   nop
79
```

Thực hiện gõ chương trình với công cụ MARS

- Kết quả :



- Giải thích :

Khởi tạo các địa chỉ của các chỉ lệnh : Heading (0xffff8010), moving (0xffff8050), leavestrack(0xffff8020), wherex(0xffff8030), wherey(0xffff8040)

Để vẽ hình tam giác, trước tiên em vẽ đường ngang bằng cách nhảy tới hàm track bắt đầu đường track mới lưu giá trị $k0 = 1 vào ô có địa chỉ leavestrack, sau đó sử dụng hàm rotate gán $a0 = 90 cho ô có địa chỉ heading để nó xoay sang ngang, tiếp đến là hàm Go gán giá trị $k0 =1 cho ô có địa chỉ Moving để nó bắt đầu di chuyển và để nó di chuyển trong 3s với $v0 =32 , $a0 = 3000 . Cuối cùng là hàm untrack gán $zero cho ô có địa chỉ Leavestrack để nó giữ đường track cũ

Tiếp tục vẽ đường dọc (go down) bằng cách tạo track mới bằng hàm track sử dụng hàm rotate gán $a0 = 180 cho ô có địa chỉ heading để nó quay xuống dưới và để nó di chuyển trong 3s với $v0 =32 , $a0 = 3000 . Cuối cùng là hàm untrack gán $zero cho ô có địa chỉ Leavestrack để gữ đường track cũ

Cuối cùng là vẽ đường chéo sang trái (goLEFT) bằng cách sử dụng hàm rotate gán $a0 = 315 để nó quay sang trái để nó di chuyển trong 3s với $v0 =32 , $a0 = 3000 . Cuối cùng là hàm untrack gán $zero cho ô có địa chỉ Leavestrack để giữ đường track

Bài 4:

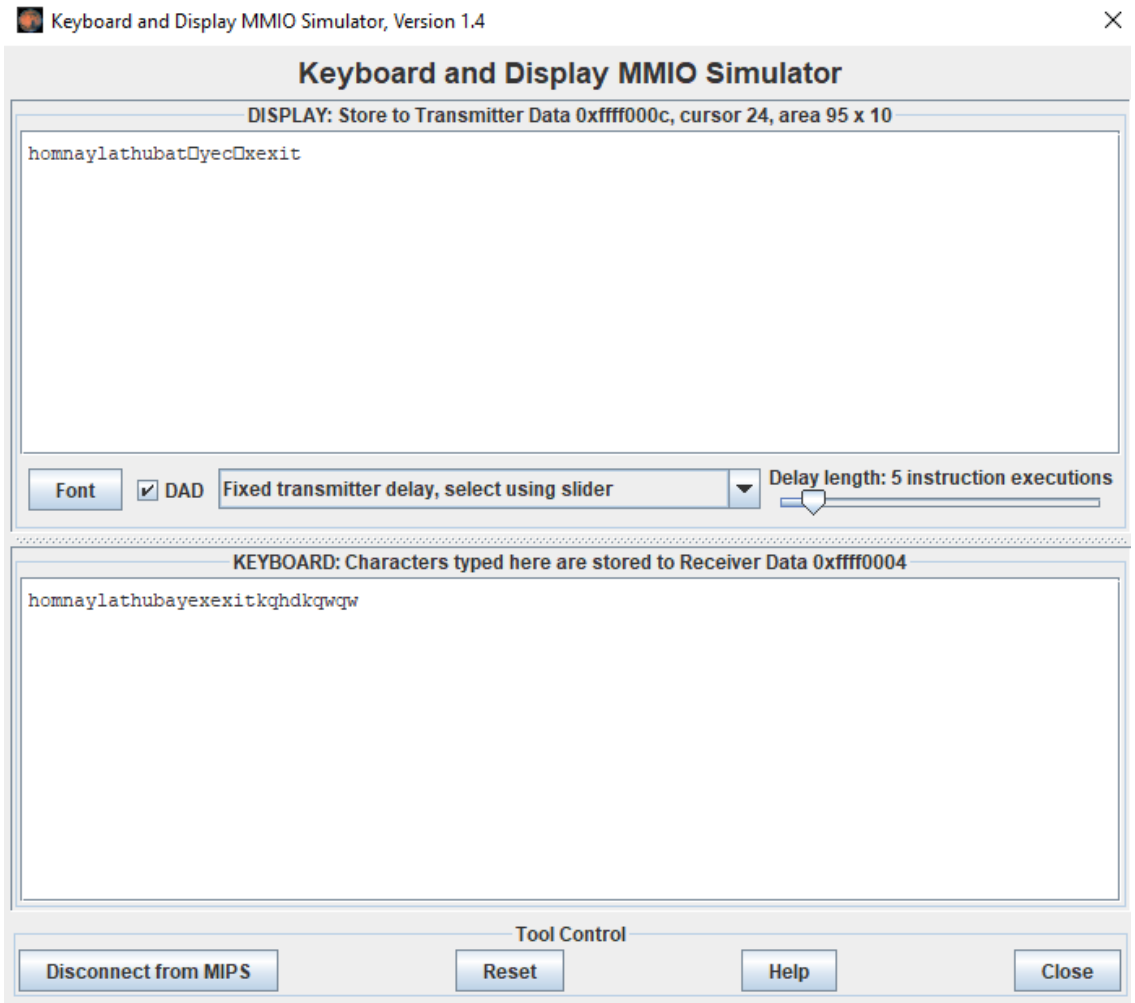
```
.eqv KEY_CODE 0xFFFF0004
.eqv KEY_READY 0xFFFF0000
.eqv DISPLAY_CODE 0xFFFF000C
.eqv DISPLAY_READY 0xFFFF0008
.data
.text
    li $k0, KEY_CODE
    li $k1, KEY_READY
    li $s0, DISPLAY_CODE
    li $s1, DISPLAY_READY
main:
lan1:
    jal loop
    beq $t0,101,lan2
    j lan1
lan2:
    jal loop
    beq $t0,120,lan3
    beq $t0,101,lan2
    j lan1
lan3:
    jal loop
    beq $t0,105,lan4
    beq $t0,101,lan2
    j lan1
```

```
lan4:
    jal loop
    beq $t0,116,end
    beq $t0,101,lan2
    j lan1
end:
    li $v0,10
    syscall

loop:
    nop
WaitForKey:
    lw $t1, 0($k1)
    nop
    beq $t1, $zero, WaitForKey
    nop

ReadKey:
   lw $t0, 0($k0)
   nop
WaitForDis:
   lw $t2, 0($s1)
   nop
   beq $t2, $zero, WaitForDis
```

```
49      beq $t2, $zero, WaitForDis
50      nop
51  ShowKey:
52      sw $t0, 0($s0)
53      nop
54      jr $ra
```

Thực hành chương trình bằng công cụ MARS

- Kết quả :



- Giải thích:
  Ban đầu khởi tạo các địa chỉ key_code(0xFFFF0004), key_ready(0xFFFF0000), display_code(0xFFFF000C), display_ready(0xFFFF0008) và lần lượt gán các địa chỉ lần lượt cho $k0, $k1, $s0, $s1
  Thực hiện vòng lặp : hàm đợi việc nhập WaitForKey lấy giá trị của từ ô có địa chỉ $k1 (key_ready) lưu vào $t1, nếu $t1 = 0 thì vẫn quay lại waitforkey, nếu $t1 = 1 thì nhảy đến ReadKey, lấy giá trị từ ô có địa chỉ $k0 (Key_code) lưu vào $t0, rồi nhảy đến WaitForDis lấy giá trị từ ô có địa chỉ $s1 (display_ready) lưu vào $t2. Nếu $t2 = 0 thì lại chạy WaitForDis. Ngược lại sẽ chạy hàm ShowKey để lưu giá trị $t0 vào $s0 (Display_code) để in ra
  Ở mỗi vòng lặp như vậy kiểm tra ký tự vừa nhập $t0 với 101 (e), nếu được thì kiểm tra ký tự tiếp theo $t0 với 120 (x), nếu đúng tiếp tục kiểm tra $t0 với 105 (i) , cuối cùng kiểm tra $t0 với 116 (t). Nếu đúng tất cả sẽ kết thúc nhảy đến end và kết thúc chương trình với $v0 = 10 và syscall
  Nếu không thỏa mãn ở kí tự vào sẽ quay về kiểm tra lại.