Bài thực hành số 10

Lớp: 139365 – Học phần: Thực Hành Kiến Trúc Máy Tính

Họ và Tên: Nguyễn Anh Thứ MSSV: 20215144

Bài 1:

Với yêu cầu hiển thị 2 số cuối mssv, ta cần hiển thị số 44

```
.eqv SEVENSEG_LEFT 0xFFFF0011 # Dia chi cua den led 7 doan trai.
 # Bit 0 = doan a;
 # Bit 1 = doan b; ...
# Bit 7 = dau .
.eqv SEVENSEG_RIGHT 0xFFFF0010 # Dia chi cua den led 7 doan phai
.text
main:
        li $a0, 0x8 # set value for segments
        jal SHOW_7SEG_LEFT # show
         nop
        li $a0, 0x1F # set value for segments
        jal SHOW_7SEG_RIGHT # show
        nop
exit:   li $v0, 10
        syscall
endmain:
#---------------------------------------------------------------
# Function SHOW_7SEG_LEFT : turn on/off the 7seg
# param[in] $a0 value to shown
# remark $t0 changed
#---------------------------------------------------------------
SHOW_7SEG_LEFT: li $t0, SEVENSEG_LEFT # assign port's address
        sb $a0, 0($t0) # assign new value
        nop
        jr $ra
        nop
#---------------------------------------------------------------
```

```
# Function SHOW_7SEG_RIGHT : turn on/off the 7seg
# param[in] $a0 value to shown
# remark $t0 changed
#---------------------------------------------------------------
SHOW_7SEG_RIGHT: li $t0, SEVENSEG_RIGHT # assign port's address
         sb $a0, 0($t0) # assign new value
         nop
         jr $ra
         nop
```

.eqv SEVENSEG_LEFT 0xFFFF0011, .eqv SEVENSEG_RIGHT 0xFFFF0010: lần lượt đặt 2 biến SEVENSEG_LEFT và SEVENSEG_RIGHT để lưu địa chỉ dung để hiển thị led 7 đoạn trái và phải

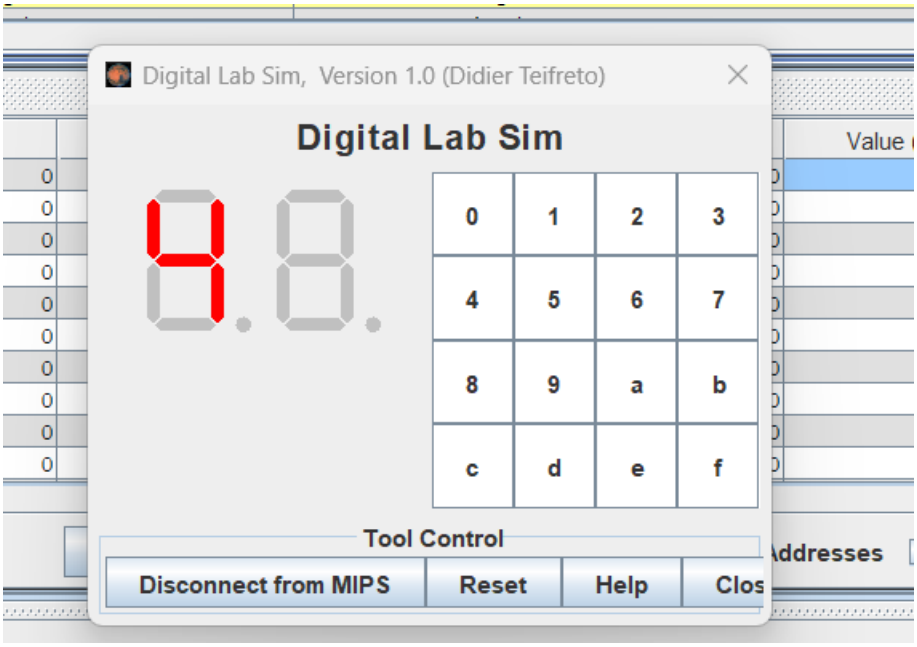
li $a0, 0x66 # set value for segments: lưu giá trị 102 vào $a0 để tạo ra được số 4

jal SHOW_7SEG_RIGHT: chạy đến hàm SHOW_7SEG_LEFT, lưu lại địa chỉ vừa ở vào $ra

Trong hàm SHOW_7SEG_LEFT:

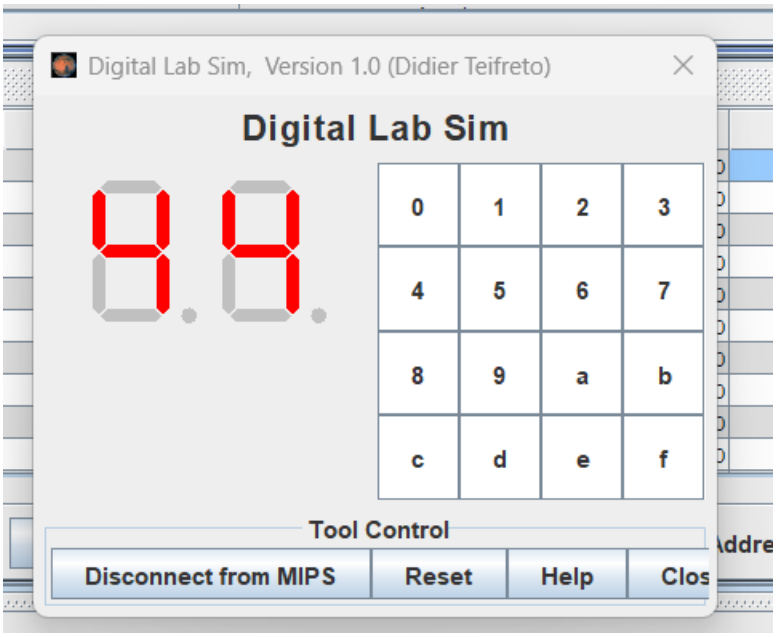
li $t0, SEVENSEG_LEFT: lưu vào $t0 giá trị của biến SEVENSEG_LEFT

sb $a0, 0($t0) : lưu a0 vào giá trị tham chiếu của t0, ở đây là 0x66 (để có thể hiển thị ra số 4)



jr $ra: quay lại vị trí vừa đứn ở hàm main

Tương tự với hàm SHOW_7SEG_RIGHT, ta được thêm 1 số 4 và hiển thị tổng thể là 44



Bài 2: vẽ cả màn hình 1 màu

```
.eqv MONITOR_SCREEN 0x10010000 #Dia chi bat dau cua bo nho man hinh
.eqv RED 0x00FF0000 #Cac gia tri mau thuong su dung
.eqv GREEN 0x0000FF00
.eqv BLUE 0x000000FF
.eqv WHITE 0x00FFFFFF
.eqv YELLOW 0x00FFFF00
.text
 li $k0, MONITOR_SCREEN #Nap dia chi bat dau cua man hinh
 li $a0, RED
 li $s0, 0
 li $s1, 256
loop1:
        add $t0, $k0, $s0
        sw $a0, 0($t0)
        addi $s0, $s0, 4
        bne $s0, $s1, loop1
```

.eqv MONITOR_SCREEN 0x10010000: tạo địa chỉ bắt đầu của màn hình

.eqv RED 0x00FF0000: tạo biến của giá trị màu thường sử dụng với các biến còn lại tương tự

li $k0, MONITOR_SCREEN: Nap dia chi bat dau cua man hinh

$t9	25	0
$k0	26	268500992
$k1	27	0

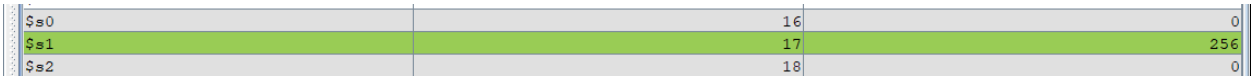
li $a0, RED: lưu giá trị của màu đỏ vào $a0

$v1	3	0
$a0	4	16711680
$a1	5	0

li $s0, 0: lưu biến đếm của địa chỉ tô màu, địa chỉ tô màu được bắt đầu từ 0

$t7	15	0
$s0	16	0
$s1	17	0

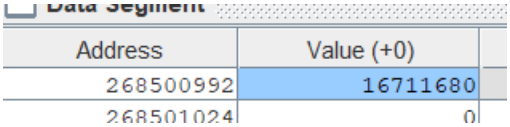
li $s1, 256: lưu giá trị lớn nhất của địa chỉ màn hình tô màu



loop1:

add $t0, $k0, $s0: tạo địa chỉ của ô cần tô, bắt đầu từ ô đầu tiên trên cùng bên trái

sw $a0, 0($t0): lưu giá trị của $a0 vào $t0, ở đây là màu đỏ



addi $s0, $s0, 4: tang giá trị của biến đếm lên 4

bne $s0, $s1, loop1: so sánh biến đếm với giá trị max. nếu vẫn nhỏ hơn, ta chạy lại loop1 để tiếp tục tô màu cho các ô tiếp theo

Bài 3: vã hình tam giác

```
.eqv HEADING 0xffff8010 # Integer: An angle between 0 and 359
 # 0 : North (up)
# 90: East (right)
 # 180: South (down)
 # 270: West (left)
.eqv MOVING 0xffff8050 # Boolean: whether or not to move
.eqv LEAVETRACK 0xffff8020 # Boolean (0 or non-0):
 # whether or not to leave a track
.eqv WHEREX 0xffff8030 # Integer: Current x-location of MarsBot
.eqv WHEREY 0xffff8040 # Integer: Current y-location of MarsBot
.text
main: jal TRACK # draw track line
 nop
 addi $a0, $zero, 90 # Marsbot rotates 90* and start running
 jal ROTATE
 nop
 jal GO
 nop
 sleep1: addi $v0,$zero,32 # Keep running by sleeping in 1000 ms
 li $a0,1000
 syscall

 jal UNTRACK # keep old track
 nop
 jal TRACK # and draw new track line
 nop
goDOWN: addi $a0, $zero, 180 # Marsbot rotates 180*
```

```
 jal ROTATE
 nop

sleep2: addi $v0,$zero,32 # Keep running by sleeping in 2000 ms
 li $a0,2000
 syscall
 jal UNTRACK # keep old track
 nop
 jal TRACK # and draw new track line
 nop
goLEFT: addi $a0, $zero, 270 # Marsbot rotates 270*
 jal ROTATE
 nop

sleep3: addi $v0,$zero,32 # Keep running by sleeping in 1000 ms
 li $a0,1000
 syscall
 jal UNTRACK # keep old track
 nop
 jal TRACK # and draw new track line
 nop

goASKEW:addi $a0, $zero, 120 # Marsbot rotates 120*
 jal ROTATE
 nop

sleep4: addi $v0,$zero,32 # Keep running by sleeping in 2000 ms
```

```
 li $a0,2000
 syscall

 jal UNTRACK # keep old track
 nop
 jal TRACK # and draw new track line
 nop
end_main:

#-----------------------------------------------------------
# GO procedure, to start running
# param[in] none
#-----------------------------------------------------------
GO: li $at, MOVING # change MOVING port
 addi $k0, $zero,1 # to logic 1,
 sb $k0, 0($at) # to start running
 nop
 jr $ra
 nop
#-----------------------------------------------------------
# STOP procedure, to stop running
# param[in] none
#-----------------------------------------------------------
STOP: li $at, MOVING # change MOVING port to 0
 sb $zero, 0($at) # to stop
 nop
 jr $ra
```

```
 jr $ra
 nop
#-----------------------------------------------------------
# TRACK procedure, to start drawing line
# param[in] none
#-----------------------------------------------------------
TRACK: li $at, LEAVETRACK # change LEAVETRACK port
 addi $k0, $zero,1 # to logic 1,
 sb $k0, 0($at) # to start tracking
 nop
 jr $ra
 nop
#-----------------------------------------------------------
# UNTRACK procedure, to stop drawing line
# param[in] none
#-----------------------------------------------------------
UNTRACK:li $at, LEAVETRACK # change LEAVETRACK port to 0
 sb $zero, 0($at) # to stop drawing tail
 nop
 jr $ra
 nop
#-----------------------------------------------------------
# ROTATE procedure, to rotate the robot
# param[in] $a0, An angle between 0 and 359
# 0 : North (up)
# 90: East (right)
# 180: South (down)
```

```
# 270: West (left)
#-----------------------------------------------------------
ROTATE: li $at, HEADING # change HEADING port
 sw $a0, 0($at) # to rotate robot
 nop
 jr $ra
 nop

```

HEADING (0xffff8010): Đây là một hằng số nguyên dùng để đại diện cho góc quay của robot. Góc quay này có giá trị từ 0 đến 359, trong đó:

0: Hướng Bắc (lên trên)

90: Hướng Đông (phải)

180: Hướng Nam (xuống dưới)

270: Hướng Tây (trái)

MOVING (0xffff8050): Đây là một hằng số boolean (0 hoặc khác 0) đại diện cho trạng thái di chuyển của robot. Nếu giá trị là khác 0, robot đang di chuyển. Ngược lại, nếu giá trị là 0, robot đang dừng.

LEAVETRACK (0xffff8020): Đây cũng là một hằng số boolean dùng để quyết định xem robot có để lại vết vẽ hay không. Nếu giá trị là 0, robot không để lại vết vẽ. Ngược lại, nếu giá trị khác 0, robot sẽ để lại vết vẽ.

WHEREX (0xffff8030): Đây là một biến số nguyên đại diện cho tọa độ x hiện tại của robot trên bản đồ.

WHEREY (0xffff8040): Đây cũng là một biến số nguyên đại diện cho tọa độ y hiện tại của robot trên bản đồ.

Trong main:

jal TRACK: Gọi hàm TRACK để vẽ đường đi của robot trên bản đồ.

addi $a0, $zero, 90: Thiết lập giá trị góc quay của robot là 90 độ (hướng Đông).

jal ROTATE: Gọi hàm ROTATE để xoay robot theo góc quay mới.

jal GO: Gọi hàm GO để bắt đầu di chuyển robot.

sleep1: Gọi hệ thống để ngừng chương trình chạy trong 1000ms (1 giây).

jal UNTRACK: Gọi hàm UNTRACK để không vẽ thêm đường đi của robot.

jal TRACK: Gọi hàm TRACK để vẽ đường đi mới của robot.

Tiếp tục với các bước điều khiển khác như goDOWN, goLEFT, goASKEW, và các hàm còn lại.

Sửa lại cho bot vẽ một hình tam giác:

Ta cần thay đổi 1 chút ở hàm main:

main:

```
jal TRACK       # Vẽ đường đi
nop
addi $a0, $zero, 0     # Đặt góc quay về phía Bắc (0 độ)
jal ROTATE
nop
jal GO        # Bắt đầu di chuyển
nop
sleep1:
   addi $v0, $zero, 32   # Ngừng chương trình trong 1000ms (1 giây)
   li $a0, 1000
   syscall
jal UNTRACK     # Dừng vẽ đường đi
```

```
nop
jal TRACK       # Vẽ đường đi mới
nop
addi $a0, $zero, 120   # Quay robot 120 độ
jal ROTATE
nop
jal GO        # Di chuyển
nop
sleep2:
   addi $v0, $zero, 32   # Ngừng chương trình trong 1000ms (1 giây)
   li $a0, 1000
   syscall
jal UNTRACK     # Dừng vẽ đường đi
nop
jal TRACK       # Vẽ đường đi mới
nop
addi $a0, $zero, 240   # Quay robot 240 độ
jal ROTATE
nop
jal GO        # Di chuyển
no
sleep3:
   addi $v0, $zero, 32   # Ngừng chương trình trong 1000ms (1 giây)
   li $a0, 1000
   syscall
jal UNTRACK     # Dừng vẽ đường đi
nop
jal TRACK       # Vẽ đường đi mới
nop
```

```
addi $a0, $zero, 0   # Quay robot về phía Bắc (0 độ)
jal ROTATE
nop
jr $ra
nop
```

Trong đoạn mã trên, robot sẽ di chuyển theo hình tam giác với các bước như sau:

- Bắt đầu từ một điểm bất kỳ, robot sẽ vẽ đường đi.
- Đặt góc quay về phía Bắc (0 độ) bằng cách gọi ROTATE với a0 là 0.
- Gọi GO để robot bắt đầu di chuyển.
- Dừng chương trình trong 1000ms bằng sleep1.
- Gọi UNTRACK để dừng vẽ đường đi.
- Gọi TRACK để vẽ đường đi mới.
- Quay robot 120 độ bằng ROTATE với a0 là 120.
- Gọi GO để robot di chuyển tiếp.
- Dừng chương trình trong

Bài 4: chạy để hiểu là được

```
.eqv KEY_CODE 0xFFFF0004 # ASCII code from keyboard, 1 byte
.eqv KEY_READY 0xFFFF0000 # =1 if has a new keycode ?
 # Auto clear after lw
.eqv DISPLAY_CODE 0xFFFF000C # ASCII code to show, 1 byte
.eqv DISPLAY_READY 0xFFFF0008 # =1 if the display has already to do
 # Auto clear after sw
.text
 li $k0, KEY_CODE
 li $k1, KEY_READY

 li $s0, DISPLAY_CODE
 li $s1, DISPLAY_READY
loop: nop

WaitForKey: lw $t1, 0($k1) # $t1 = [$k1] = KEY_READY
 nop
 beq $t1, $zero, WaitForKey # if $t1 == 0 then Polling
 nop
 #---------------------------------------------------
ReadKey: lw $t0, 0($k0) # $t0 = [$k0] = KEY_CODE
 nop
 #---------------------------------------------------
WaitForDis: lw $t2, 0($s1) # $t2 = [$s1] = DISPLAY_READY
 nop
beq $t2, $zero, WaitForDis # if $t2 == 0 then Polling
 nop
 #---------------------------------------------------
Encrypt: addi $t0, $t0, 1 # change input key
 #---------------------------------------------------
ShowKey: sw $t0, 0($s0) # show key
 nop
 #---------------------------------------------------
 j loop
 nop

```

Định nghĩa các hằng số:

KEY_CODE: Địa chỉ của biến lưu trữ mã ASCII từ bàn phím (1 byte).

KEY_READY: Địa chỉ của biến kiểm tra xem có mã ASCII mới từ bàn phím hay không (giá trị = 1 nếu có).

DISPLAY_CODE: Địa chỉ của biến lưu trữ mã ASCII để hiển thị (1 byte).

DISPLAY_READY: Địa chỉ của biến kiểm tra xem màn hình đã sẵn sàng để hiển thị hay chưa (giá trị = 1 nếu đã sẵn sàng).

Chuẩn bị các thanh ghi:

$k0 và $k1 được khởi tạo với giá trị của KEY_CODE và KEY_READY tương ứng.

Vòng lặp chính (loop):

WaitForKey: Lệnh lw được sử dụng để tải giá trị từ địa chỉ $k1 vào thanh ghi $t1. Đây là bước kiểm tra biến KEY_READY để xem có mã ASCII mới từ bàn phím hay không.

beq $t1, $zero, WaitForKey: Nếu $t1 bằng 0 (không có mã ASCII mới), lệnh nhảy beq sẽ quay lại WaitForKey để tiếp tục kiểm tra.

ReadKey: Lệnh lw được sử dụng để tải giá trị từ địa chỉ $k0 vào thanh ghi $t0. Đây là bước đọc mã ASCII từ bàn phím.

WaitForDis: Lệnh lw được sử dụng để tải giá trị từ địa chỉ $s1 vào thanh ghi $t2. Đây là bước kiểm tra biến DISPLAY_READY để xem màn hình đã sẵn sàng để hiển thị hay chưa.

beq $t2, $zero, WaitForDis: Nếu $t2 bằng 0 (màn hình chưa sẵn sàng), lệnh nhảy beq sẽ quay lại WaitForDis để tiếp tục kiểm tra.

Encrypt: Lệnh addi được sử dụng để tăng giá trị trong thanh ghi $t0 lên 1. Đây là bước thay đổi khóa đầu vào.

ShowKey: Lệnh sw được sử dụng để lưu giá trị trong thanh ghi $t0 vào địa chỉ $s0. Đây là bước hiển thị mã ASCII trên màn hình.

j loop: Lệnh nhảy