Bài thực hành số 10

Lớp: 139365 – Học phần: Thực Hành Kiến Trúc Máy Tính
Đào Minh Nhật – 20215107

Assignment 1:

Code:

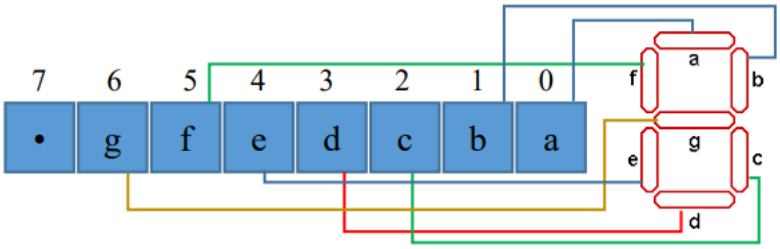
```
# Lab 10, Assignment 1
.eqv SEVENSEG_LEFT 0xFFFF0011 # Dia chi cua den led 7 doan trai.

.eqv SEVENSEG_RIGHT 0xFFFF0010 # Dia chi cua den led 7 doan phai

.text
main:
        li $a0, 0x3F # set value for segments
        jal SHOW_7SEG_LEFT # show
        nop
        li $a0, 0x07 # set value for segments
        jal SHOW_7SEG_RIGHT # show
        nop
exit: li $v0, 10
        syscall
endmain:

SHOW_7SEG_LEFT:
        li $t0, SEVENSEG_LEFT # assign port's address
        sb $a0, 0($t0) # assign new value
        nop
        jr $ra
        nop

SHOW_7SEG_RIGHT:
        li $t0, SEVENSEG_RIGHT # assign port's address
        sb $a0, 0($t0) # assign new value
        nop
        jr $ra
        nop
```

Giải thích:

Địa chỉ của đèn led 7 bên trái: 0xFFFF0011

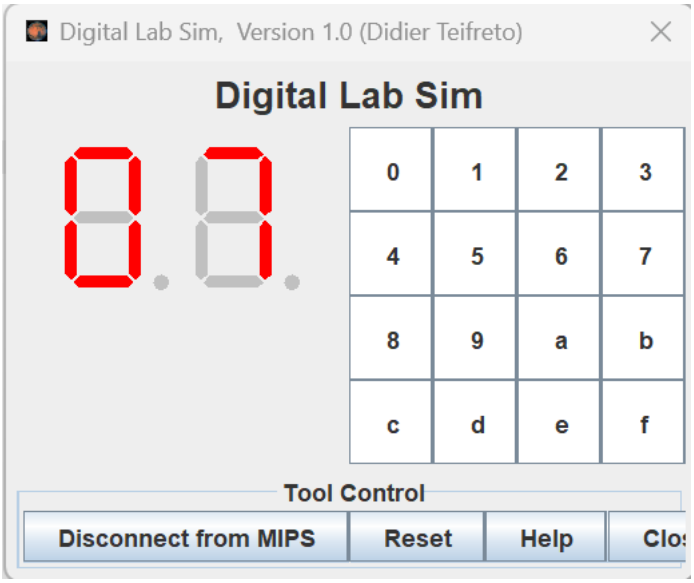
Địa chỉ của đèn led 7 bên phải: 0xFFFF0010

$a0: giá trị hiện thị các thanh led



Với $a0 = 0x3F (00111111) thì đèn sẽ hiển thị số 0

Với $a0 = 0x07 (00000111) thì đèn sẽ hiển thị số 7



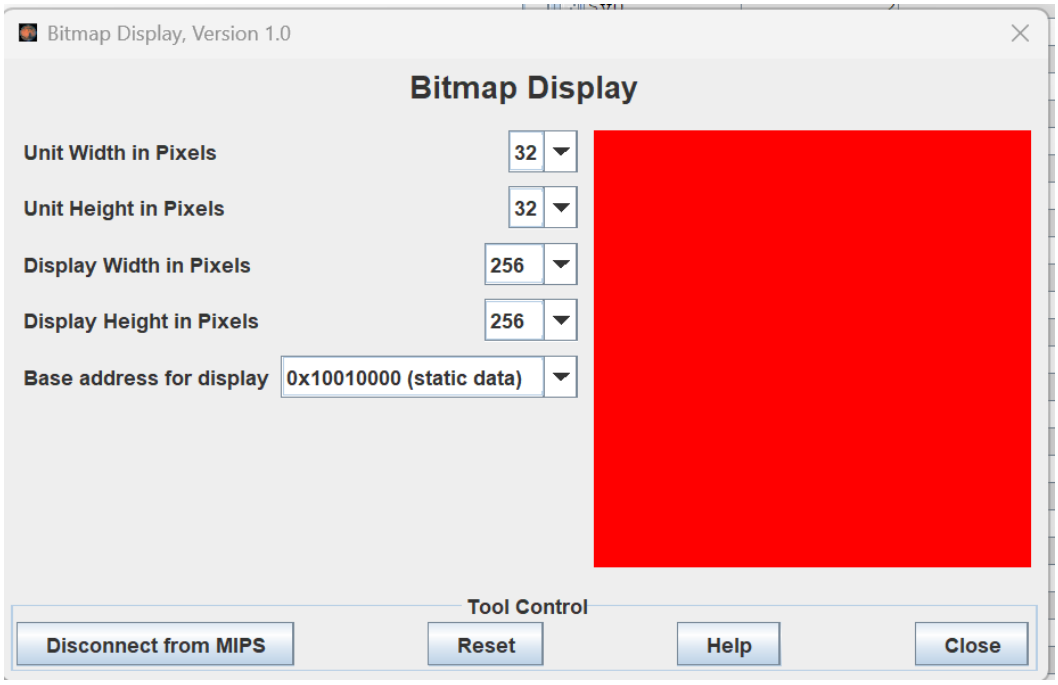
Assignment 2

Code:

Chương trình in ra toàn màn hình màu đỏ

```
# Lab 10, Assignment 2
.eqv MONITOR_SCREEN 0x10010000 #Dia chi bat dau cua bo nho man hinh
.eqv RED 0x00FF0000

.text
        li $k0, MONITOR_SCREEN #Nap dia chi bat dau cua man hinh

        li $t0, RED
        li $s0, 0
        li $s1, 256

scan_full_red:
        add $t1, $k0, $s0

        sw $t0, 0($t1)

        addi $s0, $s0, 4
        bne $s0, $s1, scan_full_red
end_scan_full_red:
```

Kết quả:



Assignment 3

Code:

```
# Lab 10, Assignment 3
.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

```

```
48  sleep3: addi $v0,$zero,32 # Keep running by sleeping in 1000 ms
49          li $a0,1000
50          syscall
51          jal UNTRACK # keep old track
52          nop
53          jal TRACK # and draw new track line
54          nop
55
56  goASKEW: addi $a0, $zero, 120 # Marsbot rotates 120*
57          jal ROTATE
58          nop
59
60  sleep4: addi $v0,$zero,32 # Keep running by sleeping in 2000 ms
61          li $a0,2000
62          syscall
63
64          jal UNTRACK # keep old track
65          nop
66          jal TRACK # and draw new track line
67          nop
68  end_main:
69
```

```
70  GO:     li $at, MOVING # change MOVING port
71          addi $k0, $zero,1 # to logic 1,
72          sb $k0, 0($at) # to start running
73          nop
74          jr $ra
75          nop
76
77  STOP:   li $at, MOVING # change MOVING port to 0
78          sb $zero, 0($at) # to stop
79          nop
80          jr $ra
81          nop
82
83  TRACK:  li $at, LEAVETRACK # change LEAVETRACK port
84          addi $k0, $zero,1 # to logic 1,
85          sb $k0, 0($at) # to start tracking
86          nop
87          jr $ra
88          nop
89
90  UNTRACK:li $at, LEAVETRACK # change LEAVETRACK port to 0
91          sb $zero, 0($at) # to stop drawing tail
92          nop
93          jr $ra
```

```
94          nop
95
96  ROTATE: li $at, HEADING # change HEADING port
97          sw $a0, 0($at) # to rotate robot
98          nop
99          jr $ra
100         nop
```

Giải thích:

Dòng 15: Thực hiện hàm TRACK để vẽ đường kẻ

Hàm TRACK (dòng 83-88):

Thay đổi đến cổng LEAVETRACK và truyền vào logic 1 ($k0 = 1) thực hiện vẽ đường kẻ.

Dòng 17-18: Khởi tạo $a0 = 90 để thực hiện hướng di chuyển sang phải với hàm rotate.

Hàm ROTATE (Dòng 96-100):

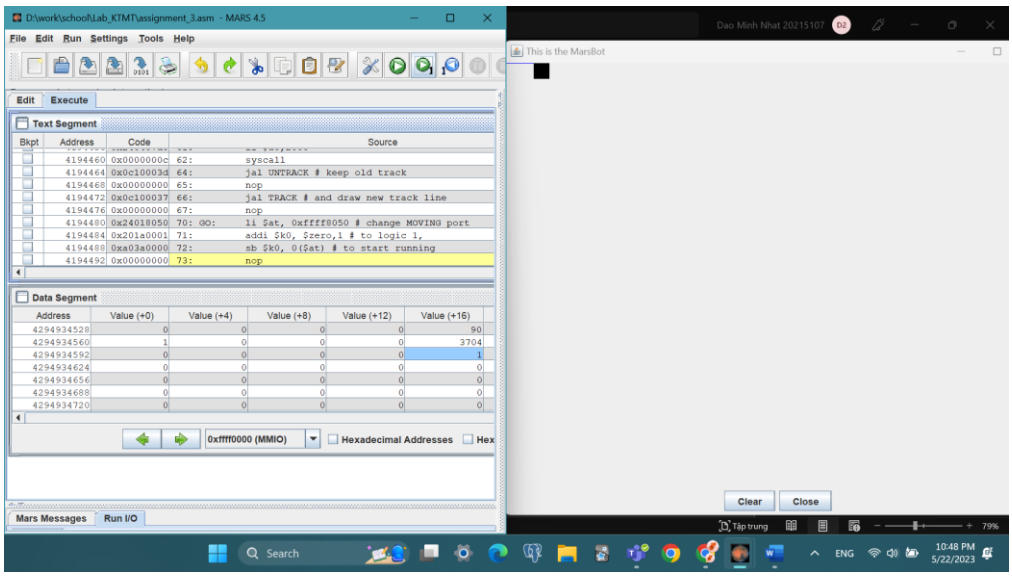
Thay đổi đến cổng HEADING

Dòng 20: Thực hiện hàm GO để bắt đầu chạy

Hàm GO (dòng 70-75):

Thay đổi đến cổng MOVING, và truyền vào logic 1 ($k0 = 1) để bắt đầu di chuyển

⇨ Sau khi thực hiện 3 hàm TRACK để vẽ đường, ROTATE (90) để chọn hướng di chuyển sang phải và GO để chạy thì ta sẽ thấy con robot di chuyển sang bên phải với vạch kẻ xanh trên đường đã di chuyển qua.



Dòng 23-25: hàm sleep với v0 = 32 và a0 = 1000 sẽ giúp cho chương trình nghỉ 1000ms.

Dòng 27: Thực hiện hàm UNTRACK để bỏ đường kẻ.

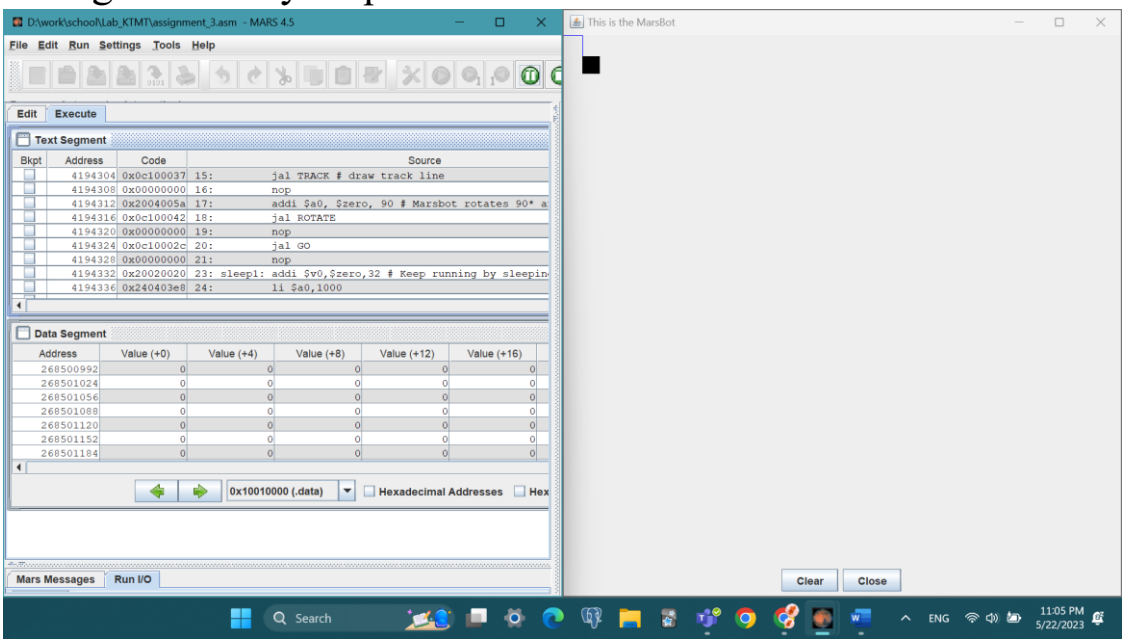
Hàm UNTRACK:

Dòng 90-94: Thay đổi đến cổng LEAVETRACK và truyền vào logic 0 để dừng in ra đường kẻ.

Dòng 29: Thực hiện hàm TRACK để vẽ đường kẻ.

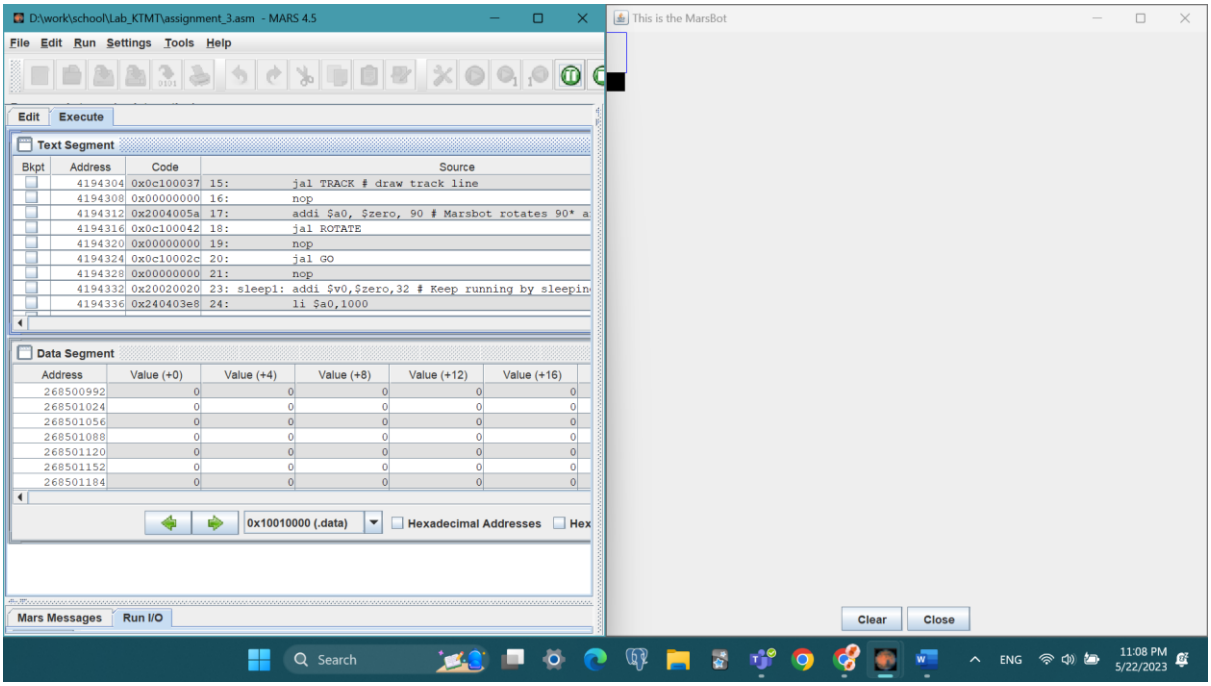
Dòng 32: Thực hiện hàm goDOWN với hàm ROTATE tham số truyền vào là $a0 = 180 để robot di chuyển xuống dưới.

⇨ Sau khi thực hiện hàm sleep1 để có thời gian nghỉ để chuyển hướng, ROTATE (180) để chọn hướng di chuyển xuống dưới thì ta sẽ thấy con robot di chuyển xuống dưới với vạch kẻ xanh trên đường đã di chuyển qua.

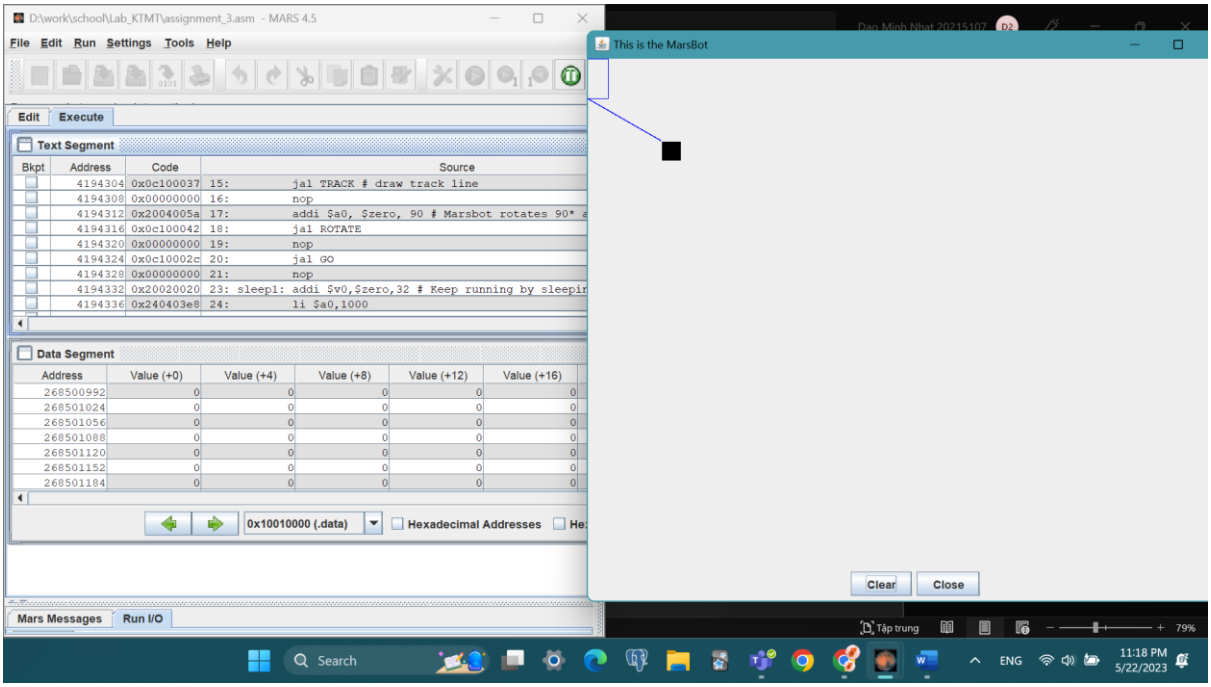


Dòng 36-46: tương tự với lệnh sleep2 (2000ms thời gian nghỉ) và goLEFT(với ROTATE (270) để di chuyển sang trái)

⇨ Khi đó robot sẽ di chuyển sang trái với đường kẻ.



Dòng 48-58: Hàm sleep3 và goASEW(với ROTATE (120)) , thì robot sẽ di chuyển theo đường chéo từ trên xuống dưới và từ trái sang phải



Assignment 4

Code:

```
# Lab 10, Assignment 4
.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
        beq $t0, 'd', Exit

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
Exit:
 li $v0, 10
 syscall
```

Giải thích:

Hàm WaitForKey là hàm kiểm tra xem đã có gì được nhập chưa.

Hàm ReadKey là để đọc kí tự vừa nhập, và kiểm tra xem kí tự nhập vào có phải là kí tự ‘d’ không nếu có thì chuyển tới exit.

Hàm WaitForDis để kiểm tra xem kí tự đã có thể hiển thị chưa.

Hàm Encrypt để tăng giá trị vừa nhập vào 1 rồi in ra màn hình.

Kết quả: