LABORATORY 10

Họ tên: Vũ Quốc Bảo
MSSV: 20225694

Assignment 1:

```
.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

.data
n0: .byte 63
n1: .byte 6
n2: .byte 91
n3: .byte 79
n4: .byte 102
n5: .byte 109
n6: .byte 125
n7: .byte 7
n8: .byte 127
n9: .byte 111

.text
main:
 li $t1, 0       # Khởi tạo biến đếm $t1
 li $v0, 32
 li $t0, SEVENSEG_RIGHT
loop1:
 beq $t1, 10, loop2 # Nếu đã hiển thị hết các giá trị từ n0 đến n9 thì thoát khỏi vòng lặp
 lb $a0, n0($t1)   # Lấy giá trị tương ứng với $t1 (tức là n0, n1, n2,...)
 jal SHOW_7SEG_RIGHT # Hiển thị giá trị lấy được
 li $a0, 1000
 syscall
 nop
 addi $t1, $t1, 1   # Tăng biến đếm để lấy giá trị tiếp theo
 j loop1
loop2:
 addi $t1, $t1, -1   # Tăng biến đếm để lấy giá trị tiếp theo
 blt $t1, 0, loop2 # Nếu đã hiển thị hết các giá trị từ n0 đến n9 thì thoát khỏi vòng lặp
 lb $a0, n0($t1)   # Lấy giá trị tương ứng với $t1 (tức là n0, n1, n2,...)
 jal SHOW_7SEG_RIGHT # Hiển thị giá trị lấy được
 li $a0, 1000
 syscall
```

```
 nop
 j loop2
 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
```

Assignment 2:

* Trái sang phải:

```
.eqv MONITOR_SCREEN 0x10010000 # Địa chỉ bắt đầu của bộ nhớ màn hình
.eqv RED 0x00FF0000 # Các giá trị màu thường sử dụng
.eqv BLACK 0x00000000

.text
main:
 li $v0, 32
 li $k0, MONITOR_SCREEN # Nạp địa chỉ bắt đầu của màn hình
 li $t1, RED
 li $t2, BLACK

 # Dùng vòng lặp để lặp lại quá trình ghi màu
 li $t3, 60 # Bắt đầu từ offset 60
```

```
 li $t4, 0  # Khởi tạo biến đếm
 li $t5, 10  # Cho tối đa 10 lần đổi vị trí

loop:
 sw $t1, ($k0)  # Ghi màu đỏ vào địa chỉ $k0
 li $a0, 1000
 syscall
 sw $t2, ($k0)  # Ghi màu đen vào địa chỉ $k0 (tô đè lên màu đỏ)
 addi $k0, $k0, 4  # Dịch con trỏ đến ô nhớ tiếp theo
 addi $t4, $t4, 1  # Tăng biến đếm lên 1
 beq $t4, $t5, exit  # Nếu số lần đổi vị trí đạt giá trị tối đa như đã cho thì thoát
 j loop

exit:
 li $v0, 10
 syscall
```

* Phải sang trái:

```
.eqv MONITOR_SCREEN 0x10010000 # Địa chỉ bắt đầu của bộ nhớ màn hình
.eqv RED 0x00FF0000 # Các giá trị màu thường sử dụng
.eqv BLACK 0x00000000

.text
main:
 li $v0, 32
 li $k0, MONITOR_SCREEN # Nạp địa chỉ bắt đầu của màn hình
 addi $k0, $k0, 60
 li $t1, RED
 li $t2, BLACK

 # Dùng vòng lặp để lặp lại quá trình ghi màu
 li $t3, 60 # Bắt đầu từ offset 60
 li $t4, 0  # Khởi tạo biến đếm
 li $t5, 10  # Cho tối đa 10 lần đổi vị trí

loop:
 sw $t1, ($k0)  # Ghi màu đỏ vào địa chỉ $k0
 li $a0, 1000
 syscall
 sw $t2, ($k0)  # Ghi màu đen vào địa chỉ $k0 (tô đè lên màu đỏ)
 addi $k0, $k0, -4  # Dịch con trỏ đến ô nhớ tiếp theo
 addi $t4, $t4, 1  # Tăng biến đếm lên 1
 beq $t4, $t5, exit  # Nếu số lần đổi vị trí đạt giá trị tối đa như đã cho thì thoát
 j loop
```

```
exit:
 li $v0, 10
 syscall
```

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:
        addi $a0, $zero, 135 # Marsbot rotates 135* and start running
        jal ROTATE
        nop
        jal GO
        nop

sleep1: addi $v0, $zero, 32 # Keep running by sleeping in 5000 ms
        li $a0, 5000
        syscall
        jal TRACK # And draw new track line

edge1: addi $a0, $zero, 150 # Marsbot rotates 150*
        jal ROTATE

sleep2: addi $v0, $zero, 32 # Keep running by sleeping in 5000 ms
        li $a0, 5000
        syscall
        jal UNTRACK # Keep old track
        nop
        jal TRACK # And draw new track line
        nop

edge2: addi $a0, $zero, 270 # Marsbot rotates 270*
        jal ROTATE

sleep3: addi $v0, $zero, 32 # Keep running by sleeping in 5000 ms
        li $a0, 5000
```

```
        syscall
        jal UNTRACK # Keep old track
        nop
        jal TRACK # And draw new track line
        nop

edge3:  addi $a0, $zero, 30 # Marsbot rotates 30*
        jal ROTATE

sleep4: addi $v0, $zero, 32 # Keep running by sleeping in 5000 ms
        li $a0, 5000
        syscall
        jal UNTRACK # Keep old track
        jal STOP

exit:
        li $v0, 10
        syscall

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

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

.data
exit: .asciiz     "exit"

.text
 li      $k0, KEY_CODE
 li      $k1, KEY_READY
 li      $s0, DISPLAY_CODE
 li      $s1, DISPLAY_READY
 la      $s2, exit
 li      $t3, 0

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

checkExit:
 add $t4, $t3, $s2
 lb $t4, 0($t4)
 bne $t4, $t0, reset
 addi $t3, $t3, 1
 j ShowKey
reset:
 addi $t3, $0, 0
 beq $t0, 'e', checkExit
 j ShowKey

ShowKey:
 sw $t0, 0($s0) # show key
 nop
 beq $t3, 4, end
 j loop

end:
 li $v0, 10
 syscall
```