BÁO CÁO THỰC HÀNH KIẾN TRÚC MÁY TÍNH (IT3280) GIỮA KỲ

Họ và tên: Trịnh Minh Đức

MSSV: 20225813

Assignment 1:

- Code:

```
.data
x: .word 0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07, 0x7F, 0x6F
.eqv SEVENSEG_RIGHT 0xFFFF0010 # Dia chi cua den led 7 doan phai
.text
main: la $s0,x #luu dia chi x vao s0
    addi $t2,$0,0 #i=0
Loop: slti $t1,$t2,40
    beq $t1,$0,Reset
    add $s1,$t2,$s0 #con tro vao x
    lb $s2,0($s1)
    add $a0,$s2,$03
    jal SHOW_7SEG_RIGHT
    nop
    addi $t2,$t2,1 #i+=1
    j Loop
Reset: addi $t2,$t2,-5
     j Loop1
Loop1: beq $t2,$0,Loop
     addi $t2,$t2,-1 #i-=1
     add $s1,$t2,$s0 #con tro vao x
     lb $s2,0($s1)
     add $a0,$s2,$03
     jal SHOW_7SEG_RIGHT
     nop
     j Loop1
endmain:
SHOW_7SEG_RIGHT: li $t0, SEVENSEG_RIGHT # assign port's address
            sb $a0, 0($t0) # assign new value
            nop
            jr $ra
            nop
```

Assignment 2:

- Code:

```
.eqv MONITOR_SCREEN        0x10010000
.eqv RED                   0x00FF0000
.eqv BLACK                 0x00000000
.text
    li      $k0, MONITOR_SCREEN
    addi    $k1, $k0, 256      # cần tô 64 ô => phải có 4 * 64 = 256 lần nhảy
    addi    $a0, $zero, 0
-
- LOOP:
    beq     $k0, $k1, END
    beq     $a0, 4, REVERSE    # $a0 = 4 thì xuống hàng dưới
    li      $t0, RED
    sw      $t0, 0($k0)        # tô ô màu đỏ
    li      $t0, BLACK
    sw      $t0, 0($k0)    # Chuyển ô màu đỏ sang màu đen ngay sau khi tô
    addi    $a0, $a0, 1        # $a0++
    addi    $k0, $k0, 4        # $k0 +=4 để tô ô tiếp theo
    j       LOOP
-
- REVERSE:
    # xuống hàng dưới và tô màu đỏ, sau đó chuyển sang màu đen
    beq     $k0, $k1, END
    beqz    $a0, LOOP      # $a0 = 0 thì xuống hàng dưới
    li      $t0, RED
    sw      $t0, 0($k0)
    li      $t0, BLACK
    sw      $t0, 0($k0)    # Chuyển ô màu đỏ sang màu đen ngay sau khi tô
    addi    $a0, $a0, -1  # $a0--
    addi    $k0, $k0, 4        # $k0 +=4 để tô ô tiếp theo
    j       REVERSE

END:
    # kết thúc chương trình
    li      $v0, 10
    syscall
```

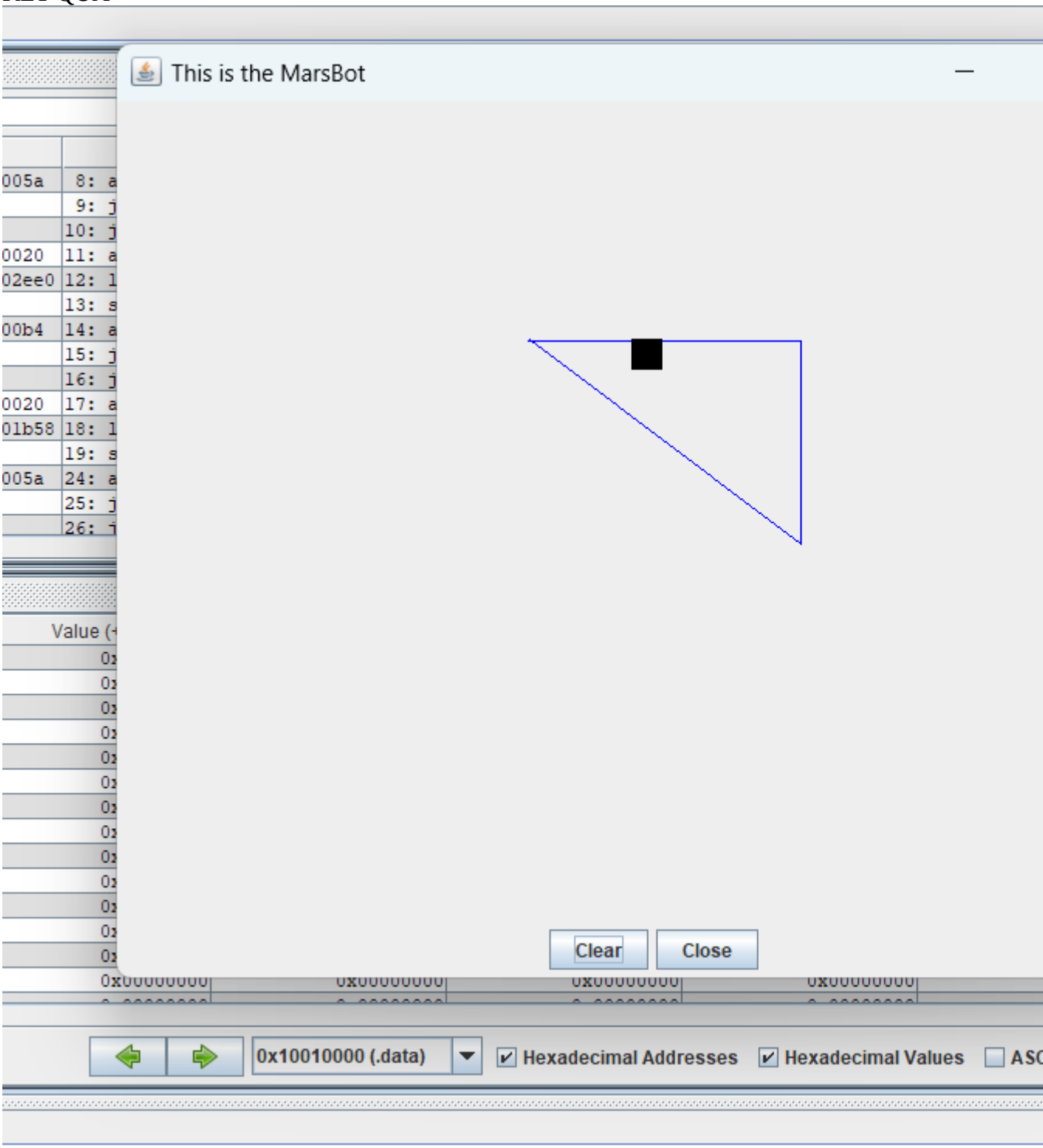
Assignment 3:

- Code:

```
.eqv HEADING 0xffff8010
.eqv MOVING 0xffff8050
.eqv LEAVETRACK 0xffff8020
.eqv WHEREX 0xffff8030
.eqv WHEREY 0xffff8040
.text
main:
addi $a0, $zero, 90
jal ROTATE
jal GO
addi $v0,$zero,32 # Keep running by sleeping in 1000 ms
li $a0,12000
syscall
addi $a0, $zero, 180
jal ROTATE
jal GO
addi $v0,$zero,32 # Keep running by sleeping in 1000 ms
li $a0,7000
syscall
#_____________________________________________________________
#Mio sleep la 1 doan, ve hay khong tuy nguoi lap trinh
#_____________________________________________________________
sleep1:
addi $a0, $zero, 90
jal ROTATE
jal GO
jal UNTRACK # keep old track
jal TRACK # and draw new track line
addi $v0,$zero,32 # Keep running by sleeping in 1000 ms
li $a0,8000
syscall
sleep2:
addi $a0, $zero, 180
jal ROTATE
jal GO
jal UNTRACK # keep old track
jal TRACK # and draw new track line
addi $v0,$zero,32 # Keep running by sleeping in 1000 ms
li $a0,6000
syscall
sleep3:
```

```
addi $a0, $zero, 307
jal ROTATE
jal GO
jal UNTRACK # keep old track
jal TRACK # and draw new track line
addi $v0,$zero,32 # Keep running by sleeping in 1000 ms
li $a0,10000
syscall
end_main:
jal UNTRACK # keep old track
addi $a0, $zero, 90
jal ROTATE
jal GO
addi $v0,$zero,32 # Keep running by sleeping in 1000 ms
li $a0,3000
syscall
jal STOP
li $v0, 10
syscall
GO:
li $at, MOVING # change MOVING port
addi $k0, $zero,1 # to logic 1,
sb $k0, 0($at) # to start running
jr $ra
ROTATE:
li $at, HEADING # change HEADING port
sw $a0, 0($at) # to rotate robot
jr $ra
STOP:
li $at, MOVING # change MOVING port to 0
sb $zero, 0($at) # to stop
jr $ra
TRACK:
li $at, LEAVETRACK # change LEAVETRACK port
addi $k0, $zero,1 # to logic 1,
sb $k0, 0($at) # to start tracking
jr $ra
UNTRACK:
li $at, LEAVETRACK # change LEAVETRACK port to 0
sb $zero, 0($at) # to stop drawing tail
jr $ra
```

KẾT QUẢ



Assigment 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 is already to do
                               # Auto clear after sw



.text
        li $a0, 'e'                #$a0 = e
        li $a1, 'x'                #$a1 = x
        li $a2, 'i'                #$a2 = i
        li $a3, 't'                #$a0 = t

        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
        bne $t0, $a3, SkipProcess
        add $t4, $0, $0
        addi $t5, $0, 3
        lw $t1, 0($sp)
        lw $t2, 4($sp)
        lw $t3, 8($sp)
        seq $t1, $t1, $a2
        seq $t2, $t2, $a1
        seq $t3, $t3, $a0
        add $t4, $t4, $t1
        add $t4, $t4, $t2
        add $t4, $t4, $t3
        bne $t4, $t5, SkipProcess
        addi $v0, $0, 10
        syscall
SkipProcess:
        addi $sp, $sp, -4
        sw $t0, 0($sp)
```

```
#-------------------------------------------------
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
```