1. Hai số cuối mã số sinh viên
2. Bitmap display: vẽ hình

Ngoài là hình vuông 32x32

Ở giữa là hình vuông 8x8 màu đỏ

1. Vẽ hình tam giác đều
2. Kiểm tra 4 kí tự gần nhất nhập vào nếu là “exit” thì thoát

Laboratory Exercise 10

­­

Assignment 1:

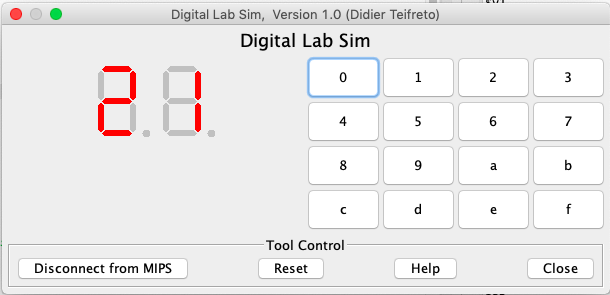
Hai số cuối mã số sinh viên (20176821) là 21

main: li $a0, 0x6 # set value for segments 1

jal SHOW\_7SEG\_LEFT # show

li $a0, 0x5B # set value for segments 2

jal SHOW\_7SEG\_RIGHT # show



Assignment 2:

.eqv RED 0x00FF0000

.text

li $k0, MONITOR\_SCREEN # load screen address

li $s0, 1 # chỉ số cột

li $s1, 1 # chỉ số hàng

li $t1, 8 # chiều dài hàng và cột

addi $k0, $k0, 1584 # vị trí bắt đầu hình vuông ở giữa

# 12\*32 + 12 = 396, 396 \* 4 = 1584

loop: # vòng lặp từng hàng một

beq $s0, $t1, next\_row # nếu chỉ số cột = 8 -> dòng tiếp theo

li $t0, RED # lấy màu đỏ

add $k0, $k0, 4 # sang cột tiếp theo

sw $t0, 0($k0) # in màu

addi $s0, $s0, 1 # tăng chỉ số cột thêm 1

j loop

next\_row:

addi $s1, $s1, 1 # tăng chỉ số hàng thêm 1

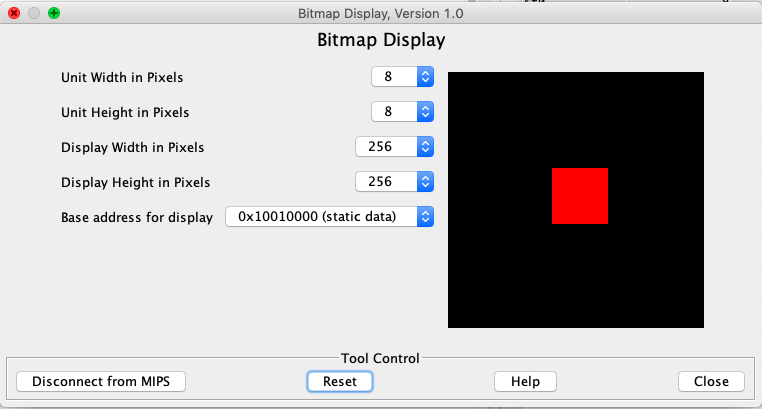
beq $s1, $t1, end # nếu số hàng = 8 -> end

addi $k0, $k0, 100 # chuyển sang toạ độ pixel ở đầu hàng tiếp theo

li $s0, 1 # reset chỉ số cột = 0

j loop

end:



Assignment 3:

**Code:**

.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 trackline

addi $a0, $zero, 90 # Marsbot rotates 90\* and start running

jal ROTATE

jal GO

sleep1: addi $v0,$zero, 32 # Keep running by sleeping in 5000 ms

li $a0,5000

syscall

jal UNTRACK # keep old track

jal TRACK # and draw new track line

goDOWN: addi $a0, $zero, 210 # Marsbot rotates 210\*

jal ROTATE

sleep2: addi $v0,$zero,32 # Keep running by sleeping in 5000 ms

li $a0,5000

syscall

jal UNTRACK # keep old track

jal TRACK # and draw new track line

goLEFT: addi $a0, $zero, 330 # Marsbot rotates 330\*

jal ROTATE

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

li $a0,5000

syscall

jal UNTRACK # keep old track

jal TRACK # and draw new track line

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)

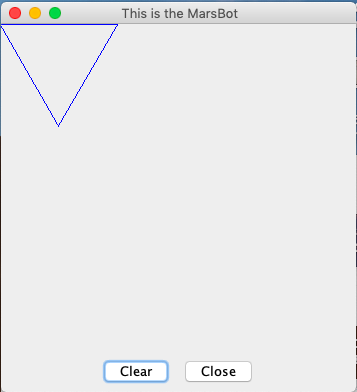
#-----------------------------------------------------------

ROTATE: li $at, HEADING # change HEADING port

sw $a0, 0($at) # to rotate robot

jr $ra

**Result:**



Assignment 4:

**Code:**

.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

.eqv e 0x65

.eqv x 0x78

.eqv i 0x69

.eqv t 0x74

.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

beq $t1, $zero, WaitForKey # if $t1 == 0 then Polling

ReadKey: lw $t0, 0($k0) # $t0 = [$k0] = KEY\_CODE

j check\_e

WaitForDis: lw $t2, 0($s1) # $t2 = [$s1] = DISPLAY\_READY

beq $t2, $zero, WaitForDis # if $t2 == 0 then Polling

ShowKey: sw $t0, 0($s0) # show key

nop

j loop

check\_e: beq $t3, e, check\_x # if e exist, check x

bne $t0, e, WaitForDis # if: ki tu hien tai khac e, continue

add $t3, $t0, $zero # else: $t3 = 'e'

j WaitForDis

check\_x: beq $t4, x, check\_i # if x exist, check i

bne $t0, x, reset # if: ki tu hien tai khac i, continue

add $t4, $t0, $zero # else: $t4 = 'x'

j WaitForDis

check\_i: beq $t5, i, check\_t # if i exist, check t

bne $t0, i, reset # if: ki tu hien tai khac t, continue

add $t5, $t0, $zero # else: $t5 = 't'

j WaitForDis

check\_t: beq $t0, t, exit # neu gap tiep t, exit

j reset # if: ki tu hien tai khac t, continue

reset: li $t3, 0 # reset 'e' to 0

li $t4, 0 # reset 'x' to 0

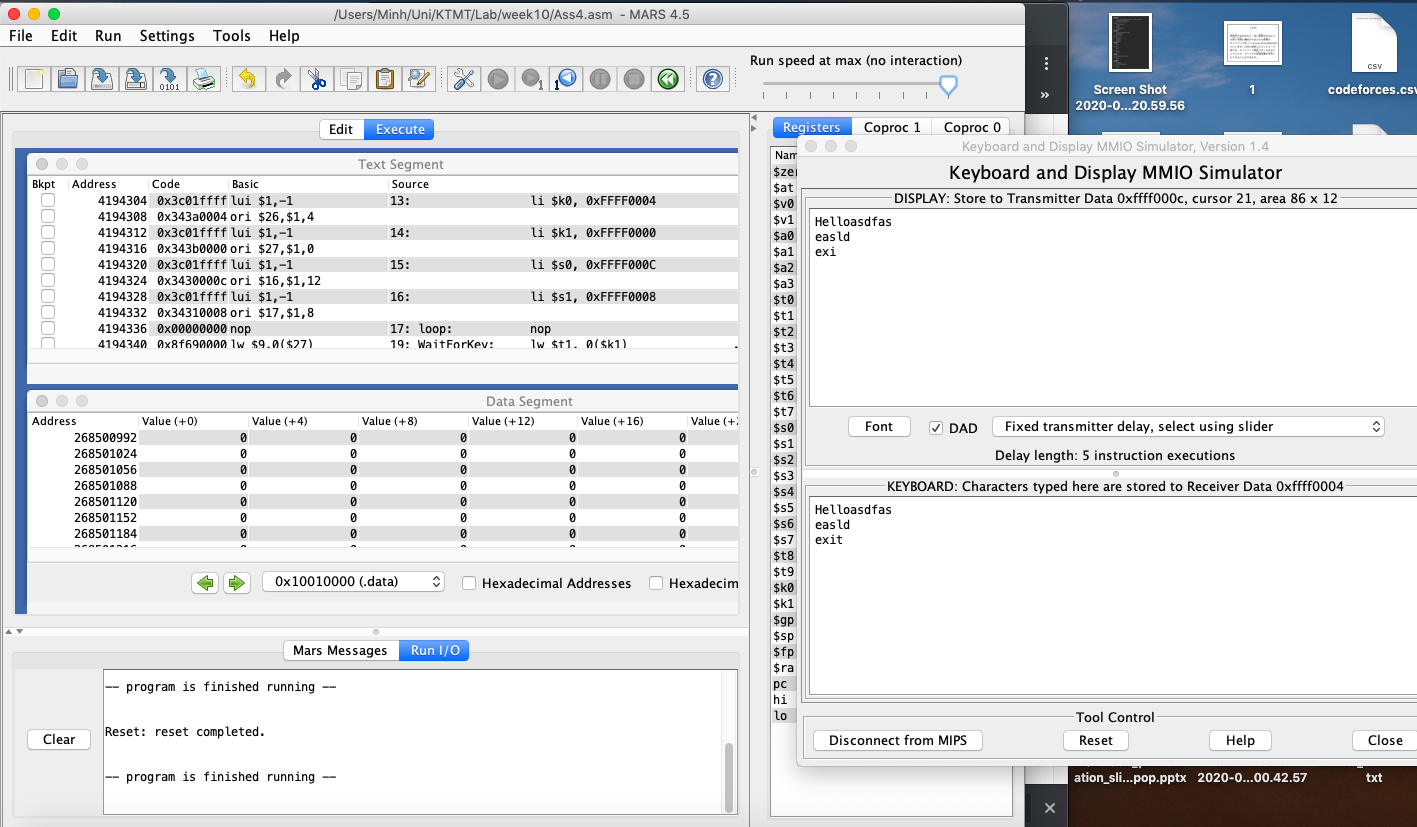
li $t5, 0 # reset 'i' to 0

j WaitForDis

exit: li $v0, 10

syscall

**Result:**

****