Võ Anh Khôi 20225870

Bài 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
.text
count_up:
li $v0, 32
li $t0, SEVENSEG_LEFT
 li $a0, 0x3F # 0
 jal SHOW_7SEG_LEFT # show
 li $a0 , 1000
 syscall
 li $a0, 0x06 #1
 jal SHOW_7SEG_LEFT
  li $a0 , 1000
 syscall
  li $a0, 0x5B #2
 jal SHOW_7SEG_LEFT
  li $a0 , 1000
 syscall
  li $a0, 0x4F #3
 jal SHOW_7SEG_LEFT
```

```
 li $a0 , 1000
syscall
 li $a0, 0x66    #4
jal SHOW_7SEG_LEFT
 li $a0 , 1000
syscall
li $a0, 0x6D    #5
jal SHOW_7SEG_LEFT
 li $a0 , 1000
syscall
li $a0, 0x7D    #6
jal SHOW_7SEG_LEFT
 li $a0 , 1000
syscall
li $a0, 0x07    #7
jal SHOW_7SEG_LEFT
 li $a0 , 1000
syscall
li $a0, 0x7F    #8
jal SHOW_7SEG_LEFT
 li $a0 , 1000
syscall
li $a0, 0x6F    #9
jal SHOW_7SEG_LEFT
 li $a0 , 1000
syscall
```

```
 li $a0, 0x7F    #8
 jal SHOW_7SEG_LEFT
  li $a0 , 1000
 syscall
 li $a0, 0x07    #7
jal SHOW_7SEG_LEFT
 li $a0 , 1000
 syscall
li $a0, 0x7D    #6
jal SHOW_7SEG_LEFT
 li $a0 , 1000
 syscall
 li $a0, 0x6D    #5
 jal SHOW_7SEG_LEFT
  li $a0 , 1000
 syscall
 li $a0, 0x66     #4
jal SHOW_7SEG_LEFT
 li $a0 , 1000
 syscall
  li $a0, 0x4F #3
 jal SHOW_7SEG_LEFT
  li $a0 , 1000
 syscall
   li $a0, 0x5B #2
 jal SHOW_7SEG_LEFT
```

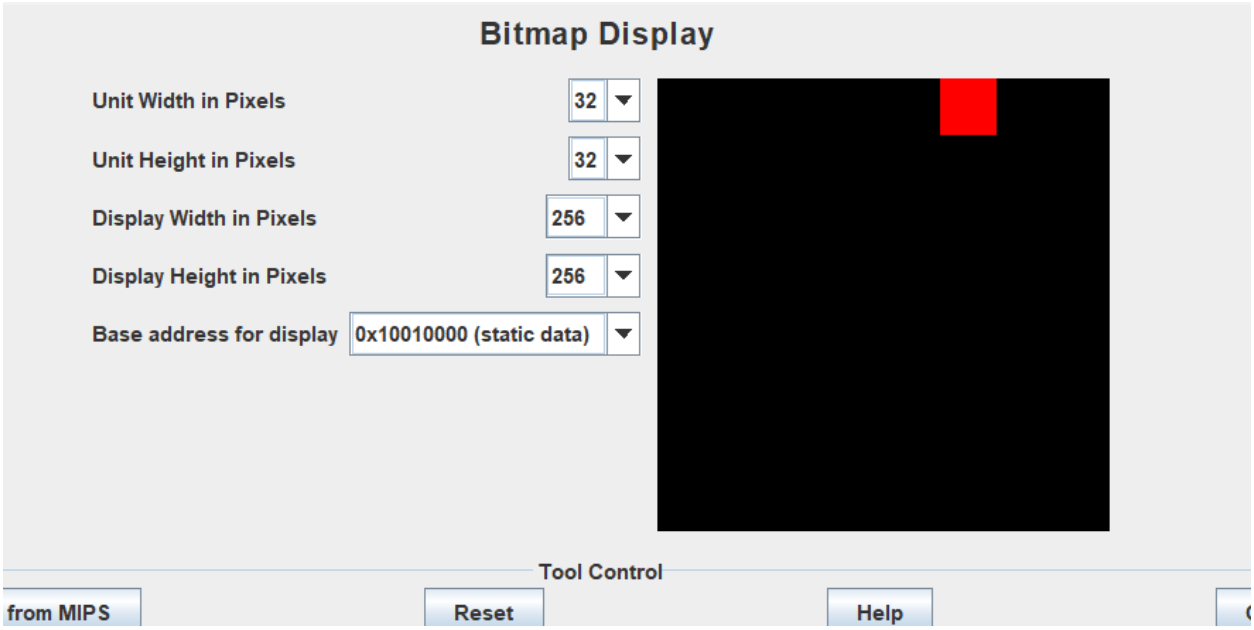
```
 li $a0 , 1000
syscall
 li $a0, 0x06 #1
jal SHOW_7SEG_LEFT
 li $a0 , 1000
syscall
j count_up
 #li $a0, 0x7D # set value for segments
 #jal SHOW_7SEG_RIGHT # show
exit: li $v0, 10
 syscall
endmain:
#---------------------------------------------------------------
# Function SHOW_7SEG_LEFT : turn on/off the 7seg
# param[in] $a0 value to shown
# remark $t0 changed
#---------------------------------------------------------------
SHOW_7SEG_LEFT:
# assign port's address
 sb $a0, 0($t0) # assign new value
jr $ra


#---------------------------------------------------------------
# Function SHOW_7SEG_RIGHT : turn on/off the 7seg
# param[in] $a0 value to shown
# remark $t0 changed
```

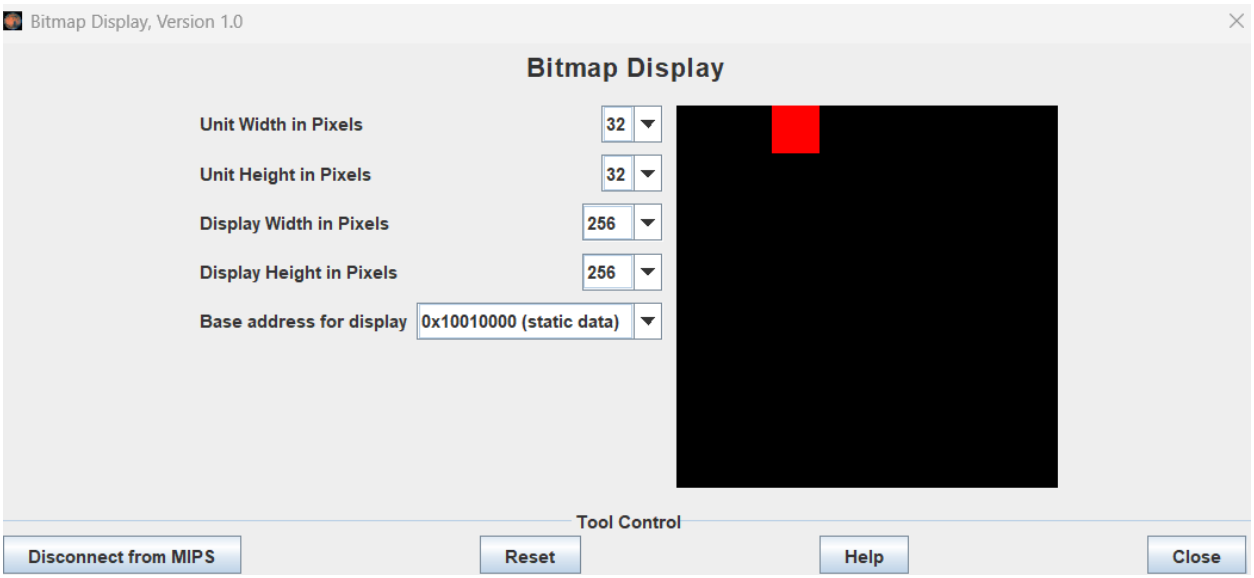
```
#---------------------------------------------------------------
SHOW_7SEG_RIGHT:
li $t0, SEVENSEG_RIGHT # assign port's address
 sb $a0, 0($t0) # assign new value
jr $ra
```

Bài 2

```
.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
.eqv NO 0x00000000
.text
 li $k0, MONITOR_SCREEN #Nap dia chi bat dau cua man hinh
  li $k1, MONITOR_SCREEN
li $t1,0
li $t2,28
loop:
        sll $t3, $t1, 2
        addi $t1, $t1, 1
        sub $t3, $t2, $t3
        bgt $zero, $t3, endloop
        li $t0, NO
```

```
        sw $t0, 0($k1)
        nop
        add $k1, $k0, $t3
        li $t0, RED
        sw $t0, 0($k1)
        nop

        j loop
endloop:
```





Bài 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, 90 # Marsbot rotates 90* and start running
 jal ROTATE
```

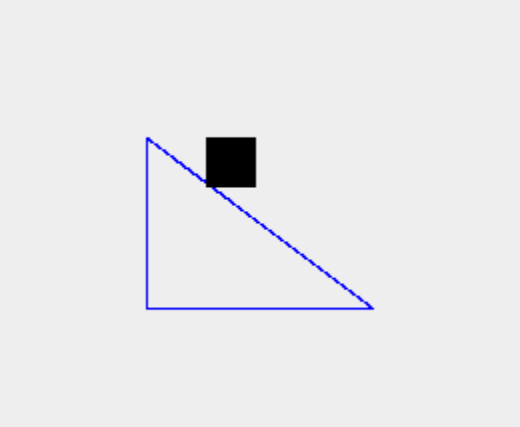
```
nop
jal GO
nop
sleep11: addi $v0,$zero,32 # Keep running by sleeping in 1000 ms
li $a0,3000
syscall
 addi $a0, $zero, 180 # Marsbot rotates 90* and start running
jal ROTATE
nop
jal GO
nop
sleep12: addi $v0,$zero,32 # Keep running by sleeping in 1000 ms
li $a0,3000
syscall


jal TRACK # draw track line
nop
addi $a0, $zero, 180 # 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 $a0,3000
syscall
```

```
 jal UNTRACK # keep old track
 nop
 jal TRACK # and draw new track line
 nop
goDOWN: addi $a0, $zero, 90 # Marsbot rotates 180*
 jal ROTATE
 nop

sleep2: addi $v0,$zero,32 # Keep running by sleeping in 2000 ms
 li $a0,4000
 syscall
 jal UNTRACK # keep old track
 nop
 jal TRACK # and draw new track line
 nop

goASKEW:addi $a0, $zero, 307 # Marsbot rotates 120*
 jal ROTATE
 nop
 jal UNTRACK # keep old track
 nop
 jal TRACK # and draw new track line
 nop

sleep4: addi $v0,$zero,32 # Keep running by sleeping in 2000 ms
 li $a0,5000
```

```
 syscall


 addi $a0, $zero, 90 # Marsbot rotates 120*
 jal ROTATE
 nop
 jal UNTRACK # keep old track
 nop
 addi $v0,$zero,32 # Keep running by sleeping in 2000 ms
 li $a0,1000
 syscall
 jal STOP
 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
 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
 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
```



Bài 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
 #-----------------------------------------------------
WaitForDis: lw $t2, 0($s1) # $t2 = [$s1] = DISPLAY_READY
 nop
 beq $t2, $zero, WaitForDis # if $t2 == 0 then Polling
 nop
 #-----------------------------------------------------
```

```
 #--------------------------------------------------
check:
checkE:
 beq $t3, 1, CheckX
 beq $t0, 101, Having
CheckX:
 beq $t3, 2, CheckI
 beq $t0, 120, Having
CheckI:
 beq $t3, 3, CheckT
 beq $t0, 105, Having
CheckT:
 beq $t3, 4, Exit
 beq $t0, 116, Having
Not:
addi $t3, $zero, 0
 #--------------------------------------------------
 ShowKey: sw $t0, 0($s0) # show key
 nop
 beq $t3, 4, Exit
 j loop
 nop

 Having: addi $t3, $t3, 1
 j ShowKey
 Exit:
```

li $v0, 10

syscall