Báo cáo Project cuối kỳ

Thành viên: Nguyễn Hoàng Tùng – 20225948, Nguyễn Xuân Khuê - 20225643

2. Vẽ hình trên màn hình Bitmap

Đề bài :

- Viết một chương trình sử dụng MIPS để vẽ một quả bóng di chuyển trên màn hình mô phỏng Bitmap của Mars). Nếu đối tượng đập vào cạnh của màn hình thì sẽ di chuyển theo chiều ngược lại.
  Yêu cầu:
  - Thiết lập màn hình ở kích thước 512x512. Kích thước pixel 1x1.
  - Quả bóng là một đường tròn.
  Chiều di chuyển phụ thuộc vào phím người dùng bấm, gồm có (di chuyển lên (W), di chuyển xuống (S), Sang trái (A), Sang phải (D) trong bộ giả lập Keyboard and Display MMIO Simulator). Vị trí bóng ban đầu ở giữa màn hình. Tốc độ bóng di chuyển là có thay đổi không đổi. Khi người dùng giữ một phím nào đó (W, S, A, D) thì quả bóng sẽ tăng tốc theo hướng đó với gia tốc tuỳ chọn

Khai báo biến

```
.data
buffer: .space 0x100000
xv: .word 0 # vantoc x bat dau
yv: .word 0#van toc y bat dau
xp: .word 256 #vi tri x
yp: .word 256 #vi tri y

circleUp :.word 0x0000ff00 #green pixel khi di chuyen len
circleDown :.word 0x0100ff00 #green pixel khi di chuyen xuong
circleLeft :.word 0x0200ff00 #green pixel khi di chuyen sang trai
circleRight :.word 0x0300ff00 #green pixel khi di chuyen sang phai
xconvert: .word 512 #gia tri cua x dua xp vao bitmap
yconvert: .word 4 #gia tri cua y dua yp vao bitmap
```

Xconvert và y convert dùng để đưa từ vị trí tọa độ về địa chỉ trên screen
Circleup,circledown,circleleft,circleright chỉ hướng của quả cầu
Xp,yp : bắt đầu tại vị trí chính giữa screen
Buffer : screen address

Vẽ border để cho quả cầu đập lại

```
#border
la $t0,buffer #dia chi buffer
li $t1,512 #512*512
li $t2,0xFF4500 #red color
bordertop:
    sw $t2,0($t0) # mau do
    addi $t0,$t0,4 #di den pixel tiep theo
    addi $t1,$t1,-1 #giam pixelcount
    bnez $t1,bordertop #lap den khi pixelcount=0
### set up bat dau ve border bot
    la $t0,buffer #screen address
    addi,$t0,$t0,1046528 #pixel goc duoi ben trai
    addi $t1,$zero,512 #do dai row duoi cung
borderbot: #cac border con lai tuong tu
    sw $t2,0($t0)
    addi $t0,$t0,4
    addi $t1,$t1,-1
    bnez $t1,borderbot
#####set up bat dau ve border left
    la $t0,buffer
    addi $t1,$zero,511 #1+511 =512
borderleft:
    sw $t2,0($t0)
    addi $t0,$t0,2048
    addi,$t1,$t1,-1
    bnez $t1,borderleft
```

```
###set up bat dau ve border right
    la $t0,buffer
    addi,$t0,$t0,2044
    addi $t1,$zero,511
borderright:
    sw $t2,0($t0)
    addi $t0,$t0,2048
    addi ,$t1,$t1,-1
    bnez ,$t1,borderright
```

Thao tác nhập vào keyboard(1)

```
thaotacgame:

   lw $s1,0xffff0004
   addi $v0,$zero,32
   addi $a0,$zero,66
   syscall
   beq $s1,100,moveright #ma ascii cua cac chu cai lan luot la "d"
   beq $s1,97,moveleft #"a"
   beq $s1,119,moveup # "w"
   beq $s1,115,movedown #"s"

   beq $t3,0,moveup #bat dau bang viec di len
moveup:
   lw $s3,circleUp #huong len
   add $a0,$s3,$zero
   jal updatecircle
   jal updateCirclePosition
   j exitmoving
movedown:
   lw $s3,circleDown #huong xuong
   add $a0,$s3,$zero
   jal updatecircle
   jal updateCirclePosition
   j exitmoving
moveleft:
   lw $s3,circleLeft #huong sang trai
   add $a0,$s3,$zero
   jal updatecircle
   jal updateCirclePosition
```

```
exitmoving:
   li $k1, 0xFFFF0000
   lw $t1, 0($k1) # $t1 = [$k1] = KEY_READY
   beq $t1, $zero, thaotacgame
   j tangvantoc
```

Dùng mã ascii để thu đc tín hiệu từ keyboard
100-d,97-a,119-w,115-s
Mỗi lần ấn một chữ sẽ chuyển sang thủ tục con di chuyển theo 4 hướng.
Tại các thủ tục di chuyển hướng thì sẽ cập nhật hình tròn và cập nhật vị trí , vận tốc của đường tròn.
Các nút một khi đã ấn thì sẽ tiếp tục đi theo hướng đó cho đến khi gặp border.
Trước khi kết thúc một lần đường tròn chuyển động, ta đưa một đoạn code để check key ready từ keyboard, nếu không có j thì tiếp tục chạy , còn nếu có thì ta check để xét xem liệu nó có tăng gia tốc không

Thao tác nhập vào keyboard(1)

```
thaotacgame:

lw $s1,0xffff0004
addi $v0,$zero,32
addi $a0,$zero,66
syscall
beq $s1,100,moveright #ma ascii cua cac chu cai lan luot la "d"
beq $s1,97,moveleft #"a"
beq $s1,119,moveup # "w"
beq $s1,115,movedown #"s"

beq $t3,0,moveup #bat dau bang viec di len
moveup:
lw $s3,circleUp #huong len
add $a0,$s3,$zero
jal updatecircle
jal updateCirclePosition
j exitmoving
movedown:
lw $s3,circleDown #huong xuong
add $a0,$s3,$zero
jal updatecircle
jal updateCirclePosition
j exitmoving
moveleft:
lw $s3,circleLeft #huong sang trai
add $a0,$s3,$zero
jal updatecircle
jal updateCirclePosition
```

```
exitmoving:
li $k1, 0xFFFF0000
lw $t1, 0($k1) # $t1 = [$k1] = KEY_READY
beq $t1, $zero, thaotacgame
j tangvantoc
tangvantoc:
```

Dùng mã ascii để thu đc tín hiệu từ keyboard
100-d,97-a,119-w,115-s
Mỗi lần ấn một chữ sẽ chuyển sang thủ tục con di chuyển theo 4 hướng.
Tại các thủ tục di chuyển hướng thì sẽ cập nhật hình tròn và cập nhật vị trí , vận tốc của hình.
Các nút một khi đã ấn thì hình tròn sẽ tiếp tục đi theo hướng đó cho đến khi gặp border.
Trước khi kết thúc một lần đường tròn chuyển động, ta đưa một đoạn code để check key ready từ keyboard, nếu không có j thì tiếp tục chạy , còn nếu có thì ta check để xét xem liệu nó có tăng gia tốc không.

Thao tác nhập(2)

```
tangvantoc: #tang van toc neu co nut dc an li`
   beq $s1,119,input1 # "w"
   beq $s1,115,input2 #"s"
   beq $s1,97,input3 #"a"
   beq $s1,100,input4
```

```
speedupright:
   bgt $s0,40,Right #neu van toc =40 thi khong giam nua
   j moveright
Right:
   subi $s0,$s0,40#speedup bang cach giam thoi gian sleep di 40ms
   j moveright
```

```
input4:

   lw $s1,0xffff0004
   addi $v0,$zero,32
   addi $a0,$zero,1
   syscall
   beq $s1,100,speedupright #ma ascii cua cac chu cai lan luot la "d"
   li $s0,400#neu khong trung voi nut trc do reset lai van toc  400ms
   beq $s1,97,moveleft #"a"
   beq $s1,119,moveup # "w"
   beq $s1,115,movedown #"s"
```

Nếu KEY READY =1,ta check lại nút trước đó được ấn
Trùng với nút nào trong 1trong 4 nút w,s,a,d thì sang thủ tục để check nút tiếp theo liệu có trùng với nó không lần lượt là input1,input2,input3,input4
$\Rightarrow$ lấy ví dụ với input4 (những cái còn lại tương tự)
Ta check xem nút được bấm tiếp theo liệu có phải là nút "d"(sang phải) hay không, nếu có thì tiến hành tăng vận tốc bằng cách giảm thời gian sleep 40ms (với mỗi lần ấn) trong lúc vẽ hình tròn
Nếu không trùng thì ta đổi chiều hình tròn theo chiều nút được ấn , đưa thời gian sleep về lại ban đầu(ở đây em set up là 400ms) và tiếp tục các thủ tục con ở thủ tục thaotacgame(thủ tục nhập từ bàn phím).

Vẽ hình tròn

```
updatecircle:
    addiu $sp,$sp,-24 #allocate 24 bytes for stack
    sw $fp,0($sp) #store caller's frame pointer
    sw $ra,4($sp) #store caller's return address
    addiu $fp,$sp,20 #updatecircle frame pointer
    lw $t0,xp # tam x of circle
    lw $t1,yp #tamm y of circle
    lw $t2,xconvert #512
    mult $t1,$t2 # yp *512
    mflo $t3 #yp *512
    add $t3,$t3,$t0 #yp *512+ xp
    lw $t2,yconvert #4
    mult $t3,$t2 # (yp *512+xp)*4
    mflo $t0 #t0=(yp *512+xp)*4
    la $a1,tail
    sw $t0,0($a1)
    la $t1,buffer #screen address
    add $t0,$t1,$t0 #t0=(yp *512+xp)*4 +screen address
    li $t5,31
    li $t6,0x00FFFF00 #yellow
    beq $a0,0x0000FF00,getnextUp #xem huong di de doi chieu neu la canh tren
    beq $a0,0x0100FF00,getnextDown #xem huong di de doi chieu neu la canh duoi
    beq $a0,0x0200FF00,getnextLeft #xem huong di de doi chieu neu la canh trai
    beq $a0,0x0300FF00,getnextRight#xem huong di de doi chieu neu la canh phai
```

Chuyển tọa độ xp,yp sang địa chỉ trên bitmap sau mỗi lần cập nhật vị trí tâm mới

Vẽ hình tròn (2)

```
getnextUp:
    addi $t7,$t0,-32768 # pixel phia tren cach 16 don vi pixel 32768/4/512=16
    lw $t8,0($t7)
    beq $t8,0xFF4500,swapVelocityUp #neu mau do thi la border => dao chieu
    j loop
# tuong tu voi 3 huong con lai
getnextDown:
    addi $t7,$t0,30720
    lw $t8,0($t7)
    beq $t8,0xFF4500,swapVelocityDown
    j loop
getnextLeft:
    addi $t7,$t0,-64
    lw $t8,0($t7)
    beq $t8,0xFF4500,swapVelocityLeft
    j loop
getnextRight:
    addi $t7,$t0,64
    lw $t8,0($t7)
    beq $t8,0xFF4500,swapVelocityRight
    j loop
```

Duyệt màu của các pixel nằm cách tâm 16 đơn vị khoảng cách(cho bán kính quả cầu =15)
Nếu màu trùng với màu của border(màu đỏ) có nghĩa là cần phải đảo hướng còn không thì sang bước tiếp theo

```
swapVelocityUp:
li $t3,0xffff0004 #keyboard
li $t4,115 #s
sw $t4,0($t3)
j movedown
swapVelocityDown:
li $t3,0xffff0004
li $t4,119 #w
sw $t4,0($t3)
j moveup
swapVelocityLeft:
li $t3,0xffff0004
li $t4,100 #d
sw $t4,0($t3)
j moveright
swapVelocityRight:
li $t3,0xffff0004
li $t4,97 #a
sw $t4,0($t3)
j moveleft
```

Thủ tục con khi đảo chiều(sẽ quay về các thủ tục movedown,moveup,... và đảo lại xv,yv sao cho hướng nó ngược lại với hướng hiện tại)

Vẽ hình tròn (3)

```
loop: #hinh tron ban kinh 15 ,in mau cho cac pixel xung quanh cach tam 15 don vi pixel
#ra soat bat dau tu cac diem cach no 15 don vi pixel theo don phuong 1 chieu
   lw $t0,xp
   lw $t1,yp
   addi $t2,$t0,-15
   addi $t3,$t1,-15
   addi $t4,$t1,15
   li $t5,31 #15+15+1

circle: #xet xem toa do cua diem do co nam trong
   sub $t7,$t2,$t0 #toa do x
   mul $t7,$t7,$t7 #x binh phuong
   sub $t8,$t3,$t1 #toa do y
   mul $t8,$t8,$t8 #y binh phuong
   add $t9,$t7,$t8 #x binh + y binh
   addi $t9,$t9,-225 #
   bltzal $t9,circle15 # so sanh x binh + y binh voi 225 (15^2) , neu nho hon thi sang buoc tiep theo
   addi $t2,$t2,1 #tang x
   addi $t5,$t5,-1
   beqz $t5,reset
   bnez $t5,circle
reset: #tang y len 1 va dua x ve gia tri ban dau
   li $t5,31
   addi $t2,$t2,-31
   addi $t3,$t3,1
   beq $t3,$t4,sleep
   j circle
```

Ta kiểm tra các pixel nằm trong block 31*31 với xp,yp là tâm

B1: xét xem liệu khoảng cách các pixel đó tới tâm có <= 15 hay không

Nếu đã nằm trong đường tròn bán kính 15 rồi thì ta sang bước tiếp

Vẽ hình tròn(4)

```
circle15: #sau khi xet nam trong vong tron ban kinh 15 thi con phai xet nam ngoai duong tron ban kinh 14
   addi $t9,$t9,29 #15^2-14^2 =29
   bgezal $t9,print # in mau neu thoa man
   addi $t2,$t2,1
   addi $t5,$t5,-1
   j circle
print:
   lw $t7,xconvert #512
   mult $t3,$t7 #512*yp
   mflo $s3
   add $s3,$s3,$t2 #512*yp+xp
   lw $s4,yconvert #4
   mul $s5, $s3,$s4 #(512*yp+xp)*4
   la $s6,buffer #screen address
   add $s5,$s5,$s6 #(512*yp+xp)*4 +screen address
   sw $t6,0($s5)
   addi $t2,$t2,1
   addi $t5,$t5,-1
   j circle
sleep:
   move $a1,$a0
   addi $v0,$zero,32
   add $a0,$zero,$s0
   syscall
#end print circle
```

B2: xét các pixel đã nằm trong bán kính 15 , xem xem liệu có nằm ngoài đường tròn tâm xp,yp bán kính 14 không

⇒ nếu thỏa mãn thì ta tô màu vàng điểm đó

⇒ Duyệt tiếp các điểm tiếp theo

⇒ Cứ như vậy ta được đường tròn màu vàng

⇒ Sau khi đã vẽ được xong hình tròn ta delay một lúc bằng thủ tục sleep

Dịch chuyển của quả cầu

```
    lw $t2,circleUp
    beq $a0,$t2,setVelocityUp #if direction and color = circleup => setvelocityup
    lw $t2,circleDown
    beq $a0,$t2,setVelocityDown#if direction and color = circledown => setvelocitydown
    lw $t2,circleLeft
    beq $a0,$t2,setVelocityLeft#if direction and color = circleleft => setvelocityleft
    lw $t2,circleRight
    beq $a0,$t2,setVelocityRight#if direction and color = circleright => setvelocityright
setVelocityUp:
    addi $t5,$zero,0
    addi $t6,$zero,-20 #set velo y
    sw $t5,xv #update
    sw $t6,yv #update
    j removeCircle
setVelocityDown:
    addi $t5,$zero,0
    addi $t6,$zero,20 #set velo y
    sw $t5,xv#update
    sw $t6,yv#update
    j removeCircle
setVelocityLeft:
    addi $t5,$zero,-20 #set velo x
    addi $t6,$zero,0
    sw $t5,xv #update
    sw $t6,yv#update
    j removeCircle
setVelocityRight:
    addi $t5,$zero,20 #set velo x
    addi $t6,$zero,0
    sw $t5,xv #update
    sw $t6,yv #update
```

Kiểm tra hướng tại $a0
Nếu đi theo hướng nào thì ta sẽ update xv,yv theo hướng đó với vận tốc cố định là 5 pixel/frame(5*4=20)

Xóa hình tròn

```
removeCircle: #xoa di de them moi , thay bang mau "den"
#tuong tu nhu ve hinh tron nhung thay mau vang thanh mau den
    li $t5,31
    li $t6,0x00000000 #black
    lw $t0,xp
    lw $t1,yp
    addi $t2,$t0,-15
    addi $t3,$t1,-15
    addi $t4,$t1,15
circle_r:
    sub $t7,$t2,$t0
    mul $t7,$t7,$t7
    sub $t8,$t3,$t1
    mul $t8,$t8,$t8
    add $t9,$t7,$t8
    addi $t9,$t9,-225
    bltzal $t9,circle15_r
    addi $t2,$t2,1
    addi $t5,$t5,-1
    beqz $t5,reset_r
    bnez $t5,circle_r
reset_r:
    li $t5,31
    addi $t2,$t2,-31
    addi $t3,$t3,1
    beq $t3,$t4,sleep_r
```

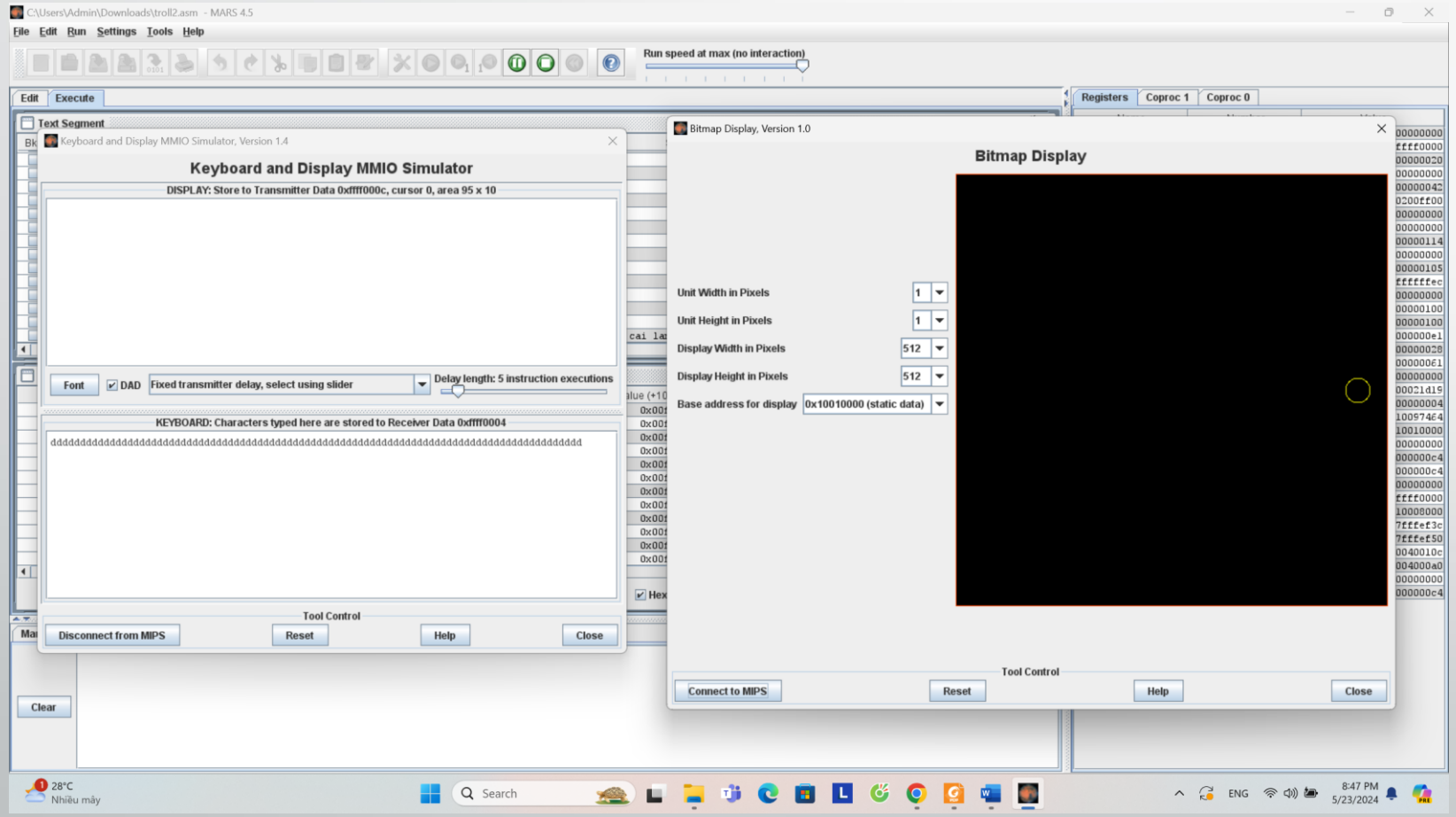
- Sau khi đã vẽ hình tròn và delay trong 1 khoảng, và cập nhật xv,yv ta sẽ thực hiện xóa đi cái hình tròn đó bằng cách làm y hệt như cách ta vẽ hình tròn nhưng thay vì màu vàng thì ta dùng màu đen

Cập nhật vị trí

```
exitUpdatecircle:
    lw $ra,4($sp)
    lw $fp,0($sp)
    addiu $sp,$sp,24
    jr $ra
updateCirclePosition:
    addiu $sp,$sp,-24
    sw $fp,0($sp)
    sw $ra,4($sp)
    addiu $fp,$sp,20
    lw $t3,xv
    lw $t4,yv
    lw $t5,xp
    lw $t6,yp
    add $t5,$t5,$t3 #vi tri tiep theo la bang vi tri hien tai + voi xv,yv
    add $t6,$t6,$t4
    sw $t5,xp #cap nhat lai vi tri cua duong tron tiep theo voi xp
    sw $t6,yp #yp tiep theo
    lw $ra,4($sp)
    lw $fp,0($sp)
    addiu $sp,$sp,24
    jr $ra
```

Trả lại sp ,fp của thủ tục updatecircle
Cập nhật vị trí của đường tròn tiếp theo bằng cách lấy xp,yp cộng với xv,yv
$\Rightarrow$ cứ lặp lại như vậy
$\Rightarrow$ HOÀN THÀNH chương trình

Chạy demo


C:\Users\Admin\Downloads\troll2.asm - MARS 4.5
File Edit Run Settings Tools Help
Run speed at max (no interaction)
Edit Execute
Text Segment
Registers Coproc 1 Coproc 0
Keyboard and Display MMIO Simulator, Version 1.4
Keyboard and Display MMIO Simulator
DISPLAY: Store to Transmitter Data 0xffff000c, cursor 0, area 95 x 10
Font
DAD
Fixed transmitter delay, select using slider
Delay length: 5 instruction executions
KEYBOARD: Characters typed here are stored to Receiver Data 0xffff0004
Tool Control
Disconnect from MIPS
Reset
Help
Close
Bitmap Display, Version 1.0
Bitmap Display
Unit Width in Pixels
Unit Height in Pixels
Display Width in Pixels
Display Height in Pixels
Base address for display
0x10010000 (static data)
Tool Control
Connect to MIPS
Reset
Help
Close
Clear
28°C
Nhiều mây
Search
ENG
8:47 PM
5/23/2024

Bài 3: Postscript CNC Marsbot

Đề bài:

3. Postscript CNC Marsbot

Máy gia công cơ khí chính xác CNC Marsbot được dùng để cắt tấm kim loại theo các đường nét được qui định trước. CNC Marsbot có một lưỡi cắt dịch chuyển trên tấm kim loại, với giả định rằng:

- Nếu lưỡi cắt dịch chuyển nhưng không cắt tấm kim loại, tức là Marsbot di chuyển nhưng không để lại vết (Track).
- Nếu lưỡi cắt dịch chuyển và cắt tấm kim loại, tức là Marsbot di chuyển và có để lại vết.

Để điều khiển Marsbot cắt đúng như hình dạng mong muốn, người ta nạp vào Marsbot một mảng cấu trúc gồm 3 phần tử:

- <Góc chuyển động>, <Thời gian>, <Cắt/Không cắt>
- Trong đó <Góc chuyển động> là góc của hàm HEADING của Marsbot.
- <Thời gian> là thời gian duy trì quá trình vận hành hiện tại.
- <Cắt/Không cắt> thiết lập lưu vết/không lưu vết.

Hãy lập trình để CNC Marsbot có thể:

- Thực hiện cắt kim loại như đã mô tả.
- Nội dung postscript được lưu trữ cố định bên trong mã nguồn.
- Mã nguồn chứa 3 postscript và người dùng sử dụng 3 phím 0, 4, 8 trên bàn phím Key Matrix để chọn postscript nào sẽ được gia công.
- Một postscript chứa chữ "DCE", một postscript chứa chữ "SoICT" cần gia công. Postscript còn lại sinh viên tự đề xuất (tối thiểu 10 đường cắt)

Bước 1: Polling và xử lí tín hiệu từ digital lab sim

```
polling:                                    #Scan signal on Digital Lab Sim
 reset:
        li $t3, 0x1
 loop:
        beq $t3, 0x8, reset
        nop
        sb $t3, 0($t1 ) # must reassign expected row
        lb $a0, 0($t2) # read scan code of key button
        bnez $a0, select_postscript
        nop
        sll $t3,$t3,1
        j loop
        nop
 select_postscript:
        beq $a0, 0x11, select1 #If press 0 -> run Postscript1
        nop
        beq $a0, 0x12, select2 #if press 4 -> run postscript2
        nop
        beq $a0, 0x14, select3 #if press 8 -> run postscript3
        nop
        j reset
        nop
```

Bước 2: Tạo các mảng để vẽ

```
postscript1: .word 90,3000,0,180,3000,0,180,6500,1,60,2000,1,30,1500,1,0,1500,1,340,2000,1,300,1520,1,270,500,1,90,8000,0,270,1500,1,240,1500,1,210,1500,1,180,1700,1,150,1500,1,120,1500,1,90,1500,
postscript2: .word 90,3000,0,180,3000,0,90,3000,1,270,3000,0,180,3000,1,90,3000,1,180,3000,1,270,3000,1,90,4500,0,0,3000,1,90,3000,1,180,3000,1,270,3000,1,90,4500,0,0,6000,1,90,1500,0,180,6000,1,9
postscript3: .word 90,3000,0,180,3000,0,180,6000,1,90,6000,1,0,6000,1,270,6000,1,90,1000,0,180,1000,0,90,1000,1,180,1000,1,270,1000,1,0,1000,1,90,3000,0,90,1000,1,180,1000,1,270,1000,1,0,1000,1,27
```

Bước 3: Tạo vòng lặp để vẽ

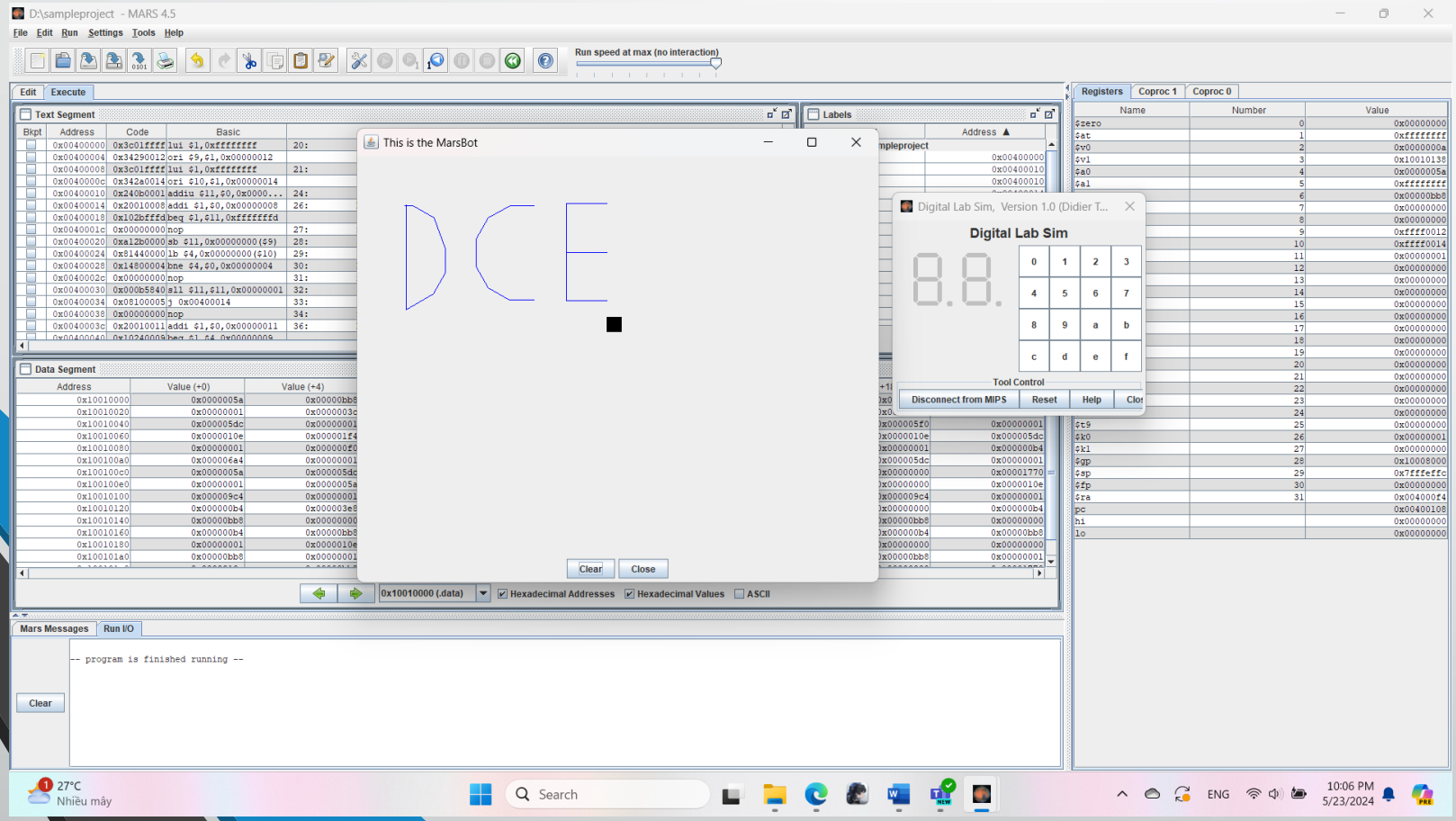
```
        la $v1, postscript3
CNC:                            #Start cut
        lw $a1, 0($v1) #Rotation
        lw $a0, 4($v1) #Time
        lw $a2, 8($v1) #Cut/No Cut
        addi $v1, $v1, 12
 check_postscript:
        beq $a1, -1, end_main #Mark -1 to end
        nop
        beq $a0, -1, end_main
        nop
        beq $a2, -1, end_main
        nop
 run_postscript:
        jal ROTATE              #Turn with rotation
        nop
        beq $a2, 1, TRACK       #Select cut/no cut
        nop
  cont:
        jal GO                  #Robot run in orbit
        nop
        li $v0, 32              #Time that robot run in orbit
        syscall
        jal STOP                #Robot stop
        nop
        jal UNTRACK             #Stop cutting
        nop
        j CNC                   #Update new orbit
        nop
end_main:
        li $v0,10
        syscall
```

Bước 4: Thêm các hàm con và các giá trị cần thiết

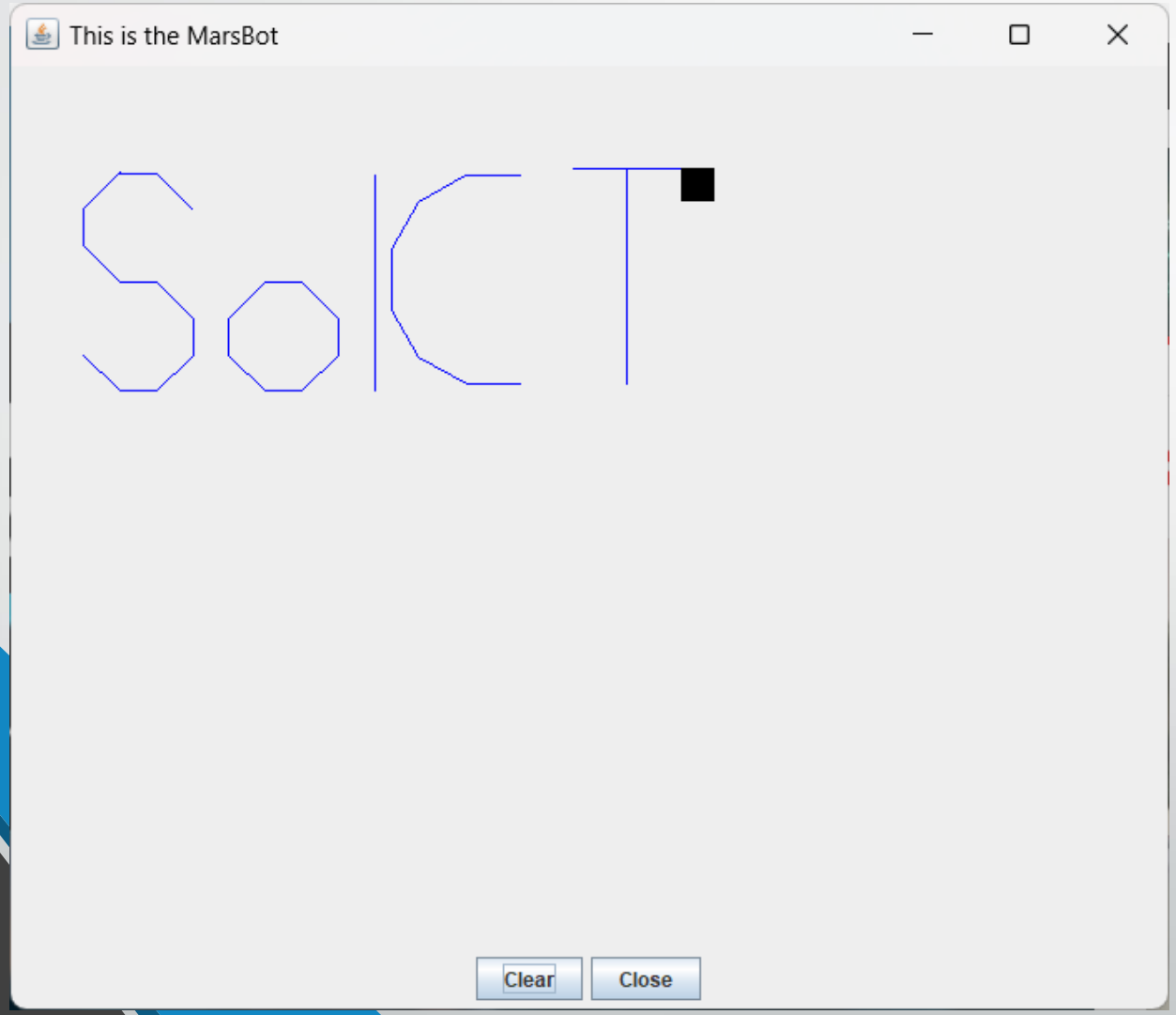
```
#-----------------------------------------------------------
STOP:
        li $at, MOVING # change MOVING port to 0
        sb $zero, 0($at) # to stop
        nop
        jr $ra
        nop
#-----------------------------------------------------------
# TRACK procedure, to start drawing line
# param[in] none
#-----------------------------------------------------------
TRACK:
        li $at, LEAVETRACK # change LEAVETRACK port
        addi $k0, $zero,1 # to logic 1,
        sb $k0, 0($at) # to start tracking
        nop
        j cont
        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] $a1, An angle between 0 and 359
# 0 : North (up)
# 90: East (right)
```

```
.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
.eqv IN_ADRESS_HEXA_KEYBOARD 0xFFFF0012
.eqv OUT_ADRESS_HEXA_KEYBOARD 0xFFFF0014
```

Kết quả demo 1:



Kết quả demo 2:



Kết quả demo 3: