ĐẠI HỌC BÁCH KHOA HÀ NỘI
TRƯỜNG CNTT & TT

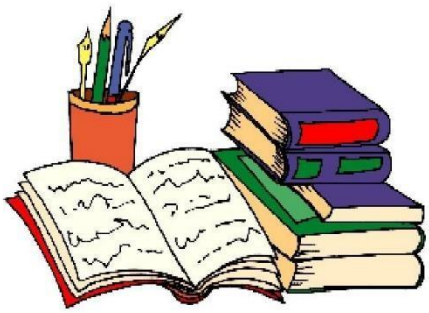
***

BÁO CÁO PROJECT NHÓM MÔN

THỰC HÀNH KIẾN TRÚC MÁY TÍNH

Học phần : Thực hành kiến trúc máy tính
Mã lớp học : 147799
Giảng viên hướng dẫn : **Lê Bá Vui**

Nhóm sinh viên thực hiện: Nhóm 4
Đỗ Hoàng Đông 20225807
Lương Anh Minh 20225650



Nội Dung

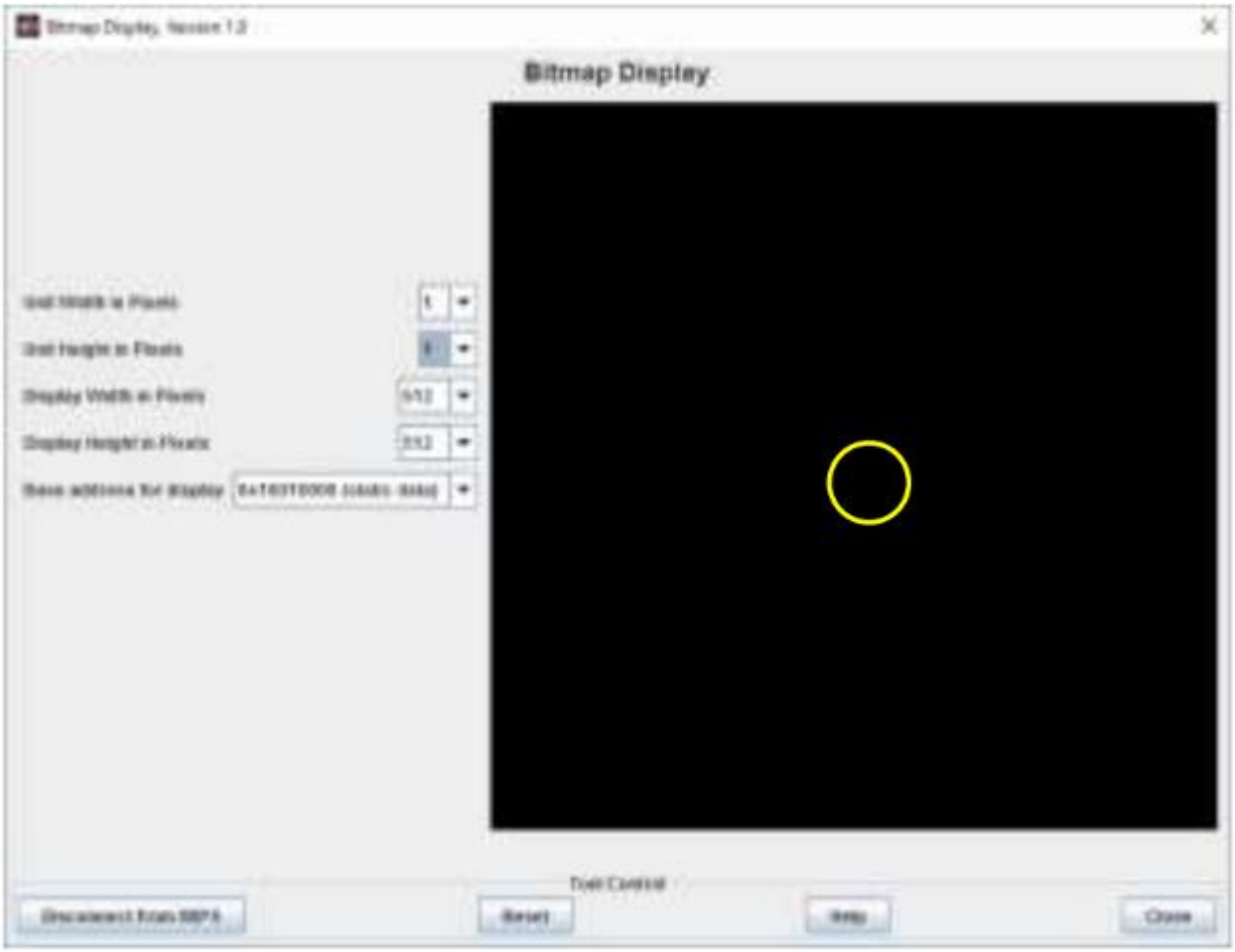
Vấn đề 2: Vẽ hình trên màn hình Bitmap

Vấn đề 4: Postscript CNC Marsbot

Vấn đề 2: Vẽ hình trên màn hình Bitmap

1. Yêu cầu bài toán

- Viết chương trình vẽ một quả bóng hình tròn 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.
  Thiết lập màn hình ở kích thước 512x512. Kích thước pixel 1x1.
- 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), tăng tốc độ (Z), giảm tốc độ (X) trong bộ giả lập Keyboard and Display MMIO Simulator).
- Vị trí bóng ban đầu ở giữa màn hình.



2. Mô tả thuật toán

Bước 1: Khởi tạo các giá trị cho chương trình

- Thiết lập tâm hình tròn I(x, y) = (256, 256) là vị trí đầu tiên của quả bóng
- R = 15 (bán kính quả bóng)
- x = y = 0(độ dịch chuyển của quả bóng)
- $t8 lưu giá trị của màu vẽ.
- Thời gian ngủ của chương trình là 100ms.

```
17  main:
18          # circle:  tam I(a,b) ;bk r
19          addi $s1,$0,256 # a
20          addi $s2,$0,256 # b
21          addi $s3,$0,15 # r
22
23          li $t8,YELLOW # Draw_color
24          jal Draw_circle # ve hinh tron ban dau
25          nop
26
27          li $a0,100  # 100ms
28          li $a1,100
```

Bước 2: Vẽ hình tròn ban đầu
Thuật toán Midpoint để vẽ hình tròn

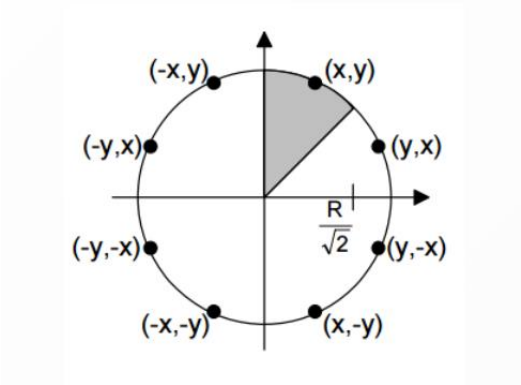
Đường tròn có tâm O(xc, yc) = (0, 0), bán kính r có phương trình:

$x^2 + y^2 = r^2 \Rightarrow x^2 + y^2 - r^2 = 0$

Đặt $f(x, y) = x^2 + y^2 - r^2$

Với mọi điểm P(x, y) nằm trong hệ tọa độ Oxy:

- P(x, y) nằm trên đường tròn O nếu f(x, y) = 0
- P(x, y) nằm ngoài đường tròn O nếu f(x, y) > 0
- P(x, y) nằm trong đường tròn O nếu f(x, y) < 0



- Do đường tròn có tính đối xứng qua các cung 1/8, nghĩa là ứng với một điểm có tọa độ (x, y) thuộc 1 cung bất kỳ, có thể hoàn toàn xác định được tọa độ 7 điểm còn lại bằng cách lấy đối xứng qua các cung.

```
putpixel(x + xc, y + yc, color);
putpixel(-x + xc, y + yc, color);
putpixel(x + xc, -y + yc, color);
putpixel(-x + xc, -y + yc, color);
putpixel( y + xc, x + yc, color);
putpixel(-y + xc, x + yc, color);
putpixel(y + xc, -x + yc, color);
putpixel(-y + xc, -x + yc, color);
```

- Từ tính chất đó nên chỉnh cần vẽ 1/8 đường tròn là đủ, sau đó sẽ lấy đối xứng để được đường tròn hoàn chỉnh.
- Điểm đầu tiên vẽ là điểm (x = 0, y = R)
- Trong cung 1/8 thứ nhất do khoảng biến thiên của x lớn hơn khoảng biến thiên của y, nên $x_{i+1} = x_i + 1$.

Bước 3: Đọc input từ người dùng

Sử dụng thanh ghi $t9 để lưu hướng di chuyển

- Nếu người dùng nhập “w”: thiết lập $t9 = 1 (lên trên).
- Nếu người dùng nhập “s”: thiết lập $t9 = 2(xuống dưới).
- Nếu người dùng nhập “a”: thiết lập $t9 = 3 (sang trái).
- Nếu người dùng nhập “d”: thiết lập $t9 = 4 (sang phải).
- Nếu người dùng nhập “z”: thiết lập giảm thời gian ngủ để tăng tốc độ.
- Nếu người dùng nhập “x”: thiết lập tăng thời gian ngủ để giảm tốc độ.

- Nếu người dùng nhập kí tự khác : thiết lập $t9 = 0 (quả bóng dừng di chuyển).
    - *Đọc input từ người dùng*

```
nop
lb $t3, 0($k0) #luu ki tu duoc an vao $t3
#Doi chieu phim duoc an voi cac ki tu w,s,a,d,z,x
beq $t3, 119, up_direction      # w
beq $t3, 115, down_direction    # s
beq $t3, 97, left_direction     # a
beq $t3, 100, right_direction   # d
beq $t3, 122, Speed     # z
beq $t3, 120, Slow      # x
add $t9,$0,$0
j end_process
nop
up_direction:
addi $t9, $zero, 1
j end_process
nop

down_direction:
addi $t9, $zero, 2
j end_process
nop

left_direction:
addi $t9, $zero, 3
j end_process
nop

right_direction:
addi $t9, $zero, 4
j end_process
nop
Speed:
beq $al,20,end_process
nop
addi $al,$al,-20
j end_process
nop

Slow:
beq $al,200,end_process
nop
addi $al,$al,20
j end_process
nop
```

Bước 4: Kiểm tra xem quả bóng có đập vào cạnh của màn hình hay không bằng cách

- Kiểm tra hoành độ điểm bên phải có bằng 512 – 17 = 495 (chạm tường) hay không?
- Kiểm tra hoành độ điểm bên trái 16 (chạm tường)
- Kiểm tra tung độ điểm dưới cùng (495)
- Kiểm tra tung độ điểm trên cùng 16 (chạm tường).
- Nếu quả bóng đập vào cạnh của màn hình, ta sẽ đảo ngược hướng của quả bóng, bằng cách lưu $t9 là hướng ngược lại
- Nếu không, ta sẽ di chuyển quả bóng theo hướng cũ.

```
Up:
li $t8,BLACK
jal Draw_circle # delete OLD_Circlev # I , r
nop
sub $s2,$s2,1 # y -= 1
li $t8,YELLOW
jal Draw_circle  # draw new circle
nop
bne $s2,16,Up1
nop
# To_Down
addi $t9,$0,2
Up1:
j Moving
nop
```

Nếu quả bóng đang đi lên ($t9 = 1) tới tâm có tung độ = 16 thì sẽ chuyển hướng ngược lại (đổi $t9 = 2 to down). Tương tự với Down, Left, Right.

Bước 5: Di chuyển quả bóng tới vị trí mới

- Xóa quả bóng ở vị trí cũ bằng cách set giá trị màu sắc của mọi điểm thuộc viền của quả bóng thành màu đen. ($t8 = BLACK).
- Cập nhật lại vị trí của tâm quả bóng tới tọa độ mới

  (đi lên y = y - 1 , đi xuống y = y + 1, sang trái x = x -1, sang phải x = x + 1).
- Vẽ lại quả bóng ở vị trí mới bằng cách set giá trị màu sắc của mọi điểm thuộc viền của quả bóng thành màu vàng. ($t8 = YELLOW).
- Sau đó, ta quay lại bước 3.

```
Continue:
beq $t9, 1, Up
nop
beq $t9, 2, Down
nop
beq $t9, 3, Left
nop
beq $t9, 4, Right
nop
j Moving
nop

Up:
li $t8,BLACK
jal Draw_circle # delete OLD_Circlev # I , r
nop
sub $s2,$s2,1 # y -= 1
li $t8,YELLOW
jal Draw_circle  # draw new circle
nop

bne $s2,16,Up1
nop
# To_Down
addi $t9,$0,2
Up1:
j Moving
nop

Down:
li $t8,BLACK
jal Draw_circle
nop
add $s2,$s2,1 # y += 1
li $t8,YELLOW
jal Draw_circle
nop
bne $s2,495,L2
nop
# To_Up
addi $t9,$0,1
L2:
j Moving
```

Vấn đề 4: Postscript CNC Marsbot

4. 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:

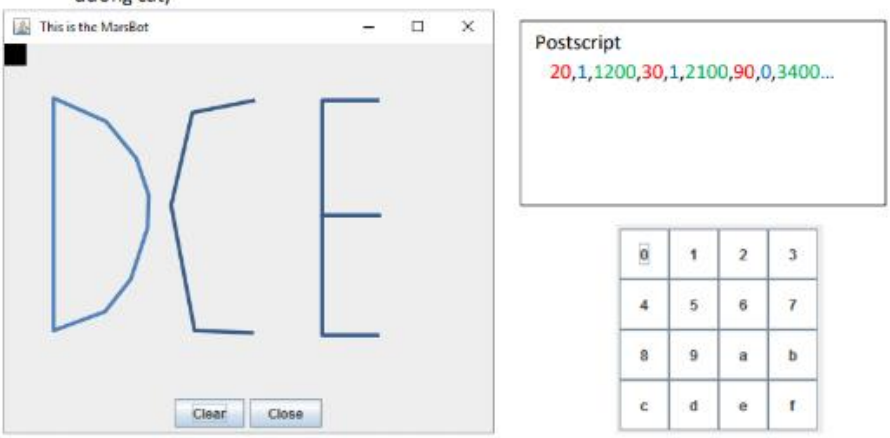
- Nếu lưỡi cắt dịch chuyển nhưng không cắt tấm kim loại, tức là Marsbot di chuyển nhưng không để lại vết (Track)
- Nếu lưỡi cắt dịch chuyển và cắt tấm kim loại, tức là Marsbot di chuyển và có để lại vết.

Để điều khiển Marsbot cắt đúng như hình dạng mong muốn, người ta nạp vào Marsbot một kịch bản là chuỗi ký tự bao gồm liên tiếp bộ 3 tham số (các giá trị phân cách nhau bởi dấu phẩy)

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

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 cần gia công. Hai script còn lại sinh viên tự đề xuất (tối thiểu 10 đường cắt)



1. Mô tả vấn đề

Yêu cầu bài toán:

- Sử dụng MarsBot Tool trong phần mềm MARS MIPS để vẽ các từ DCE, SoICT và một hình dạng tự chọn (tối thiểu 10 đường cắt)
- Tạo postscript theo cấu trúc <góc chuyển động>, <thời gian>, <cắt/không cắt> được lưu trữ cố định bên trong mã nguồn để vẽ các từ nói trên
- Mã nguồn chứa 3 postscript và người dùng sử dụng 3 phím 0, 4, 8 trên bàn phím Key Matrix của Digital Lab Sim để chọn postscript được gia công:

  Phím 0: chương trình sẽ thực thi message1 (vẽ từ DCE)

  Phím 4: chương trình sẽ thực thi message2 (vẽ từ SoICT)

  Phím 8: chương trình sẽ thực thi message3 (vẽ từ I Love HUST)

2. Ý tưởng

B1: Nhập mã lệnh từ Digital Lab Sim

B2: Chờ nhận tín hiệu phím từ bàn phím vào Digital Lab Sim:

+Nếu nhận phím 0 thì Marsbot sẽ chạy vẽ Postscript DCE

+Nếu nhận phím 4 thì Marsbot sẽ chạy vẽ Postscript SoICT

+Nếu nhận phím 8 thì Marsbot sẽ chạy vẽ Postscript ILoveHUST

B3: Quan sát Marsbot chạy và vẽ postscript. Sau khi vẽ xong chương trình sẽ

hiển thị câu hỏi “Do you want to continue?” Sau khi trả lời yes có thể tiếp tục ấn phím 0, 4 hay 8 để Mars Bot có thể vẽ tiếp.

3. Triển khai vấn đề

1. Cách thức hoạt động:

Bước 1: Khởi tạo

- Kích hoạt các Interrupt cho Key Matrix

```
# Enable interrupts you expect
#------------------------------------------------------
# Enable the interrupt of Keyboard matrix 4x4 of Digital Lab Sim
        li $t1, IN_ADRESS_HEXA_KEYBOARD
        li $t2, OUT_ADRESS_HEXA_KEYBOARD
"
```

- Bắt đầu vòng lặp vô hạn để chờ các ngắt từ bàn phím

```
"
Loop:
        nop
        nop
        nop
        nop
        b Loop # Wait for interrupt
end_main:
#------------------------------ ----------------------------------
```

Bước 2: Xác định phím:

- Xác định phím nào đã được bấm bằng cách kiểm tra từng hàng của Key Matrix

```
Check:
        li $t3, 0x01 # start with row 0x1
        row_loop:
        sb $t3, 0($t1) # assign expected column
        lb $a0, 0($t2) # read scan code of key button
        beqz $a0, next_row# if scan code is 0, no key pressed, go to next row
#-------------------------------------------------------------------------------
        nop

      next_row:
      sll $t3, $t3, 1 # multiply row value by 2
      blt $t3, 0x10, row_loop # if row value is less than 0x10 (16), repeat the row loop
```

Bước 3: Logic vẽ MarsBot

```
    lw $a0,0($s0)
    beq $a0,13,end2
    jal ROTATE  # at 180
    nop
    jal TRACK  # at 0
    nop
    nop
    jal GO
    nop
    addi $v0,$zero,32
    addi $s0,$s0,-4       # Decrementing $s0 here causes issues
    lw $a0,0($s0)
    syscall
    jal UNTRACK
    nop
    jal STOP
    addi $s0,$s0,8        # Incrementing $s0 here
    j  loop
    nop
```

1. Logic vẽ được điều khiển bởi một script được lưu trữ trong bộ nhớ. Mỗi script bao gồm một chuỗi các hướng dẫn với ba tham số: góc, thời gian và cắt/không cắt.
2. Thuật toán xử lý từng bước di chuyển trong postscript

- Góc: Xoay MarsBot đến góc chỉ định
- Thời gian: Di chuyển MarsBot trong khoảng thời gian chỉ định

- Cắt/Không cắt: Xác định liệu MarsBot có nên để lại dấu vết hay không

- Đầu tiên ta lấy giá trị từ vùng nhớ mà $s0 trỏ tới và lưu vào thanh ghi $a0

```
loop:
    lw $a0,0($s0)
```

- So sánh $a0 với 13, nếu giống nhau sẽ nhảy tới end2
- **jal ROTATE; jal TRACK; jal GO:** Gọi các hàm "ROTATE", "TRACK" và "GO". "jal" là lệnh nhảy và lưu trạng thái trở lại (jump and link), nó lưu địa chỉ trở lại của lệnh tiếp theo vào thanh ghi $ra (return address register) trước khi nhảy đến hàm.
- **addi $s0, $s0, -4, addi $s0, $s0, 8 và lw $a0, 0($s0)**: Điều chỉnh con trỏ $s0, giảm giá trị của nó đi 4 và sau đó tăng lên 8 trong khi sử dung **lw** để Lấy giá trị từ vùng nhớ mà $s0 đang trỏ tới sau khi được giảm giá trị và lưu vào thanh ghi $a0. Sau đó, nó quay lại vòng lặp để thực hiện lại quá trình này.
- Sau khi đọc đến kí tự cuối là 13 thì Marsbot sẽ dừng chạy và hiển thị end2 .

```
end2:
        li $v0,50
        la $a0,notification
        syscall
        bne $a0,0,else
        nop
        j point
        else: # End the program
        li $v0,10
        syscall
```

- Tăng địa chỉ chứa trong thanh ghi epc bằng cách sử dụng 2 lệnh mfc0 để đọc thanh ghi trong bộ đồng xử lý C0 và mtc0 để ghi giá trị vào thanh ghi trong bộ đồng xử lý C0
- Khôi phục giá trị của các thanh ghi đã lưu từ Stack và lệnh eret giúp quay trở về thực hiện chương trình chính sau khi xử lý xong các đoạn mã Exception bằng cách gán nội dung thanh ghi PC bằng giá trị trong thanh ghi $14(epc). Khi đó thanh ghi $14 (epc) sẽ chứa địa chỉ kế tiếp của chương trình chính.

```
point:
#--------------------------------------------------------
# Evaluate the return address of main routine
# epc <= epc + 4
#--------------------------------------------------------
next_pc:
        mfc0 $at, $14 # $at <= Coproc0.$14 = Coproc0.epc
        li $at,0x00400010
        mtc0 $at, $14 # Coproc0.$14 = Coproc0.epc <= $at
return:
        eret # Return from exception
```

*Các chương trình chức năng:

Check:

- Thủ tục này kiểm tra các phím được nhấn từ bàn phím ma trận và thực hiện các thao tác dựa trên mã quét của phím.

```
Check:
        li $t3, 0x01 # start with row 0x1
        column_loop:
        sb $t3, 0($t1) # assign expected column
        lb $a0, 0($t2) # read scan code of key button
        beqz $a0, next_column # if scan code is 0, no key pressed, go to next row
#--------------------------------------------------------------------------------
        nop
        bne $a0,0x11,else1
        la $s0,message1
        j loop
else1:
        bne $a0,0x12,else2
        la $s0,message2
        j loop
else2:
        bne $a0,0x14,end1
        la $s0,message3
```

GO:

- Bắt đầu di chuyển robot bằng cách đặt cổng MOVING thành 1.

```
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:

- Dừng di chuyển robot bằng cách đặt cổng MOVING thành 0.

```
STOP:
        li $at, MOVING # change MOVING port to 0
        sb $zero, 0($at) # to stop
        nop
        jr $ra
        nop
```

TRACK:

- Bắt đầu vẽ đường (để lại dấu vết) bằng cách đặt cổng LEAVETRACK.

```
TRACK:
        li $at, LEAVETRACK # change LEAVETRACK port
        addi $s0,$s0,8
        lw $k0,0($s0)
        sb $k0, 0($at) # to start tracking
        nop
        jr $ra
        nop
```

UNTRACK:

- Dừng vẽ đường bằng cách đặt cổng LEAVETRACK thành 0.

```
UNTRACK:
        li $at, LEAVETRACK # change LEAVETRACK port to 0
        sb $zero, 0($at) # to stop drawing tail
        nop
        jr $ra
        nop
```

ROTATE:

- Xoay robot đến góc được chỉ định (0-359 độ) bằng cách đặt giá trị của góc vào cổng HEADING.

```
ROTATE:
        li $at, HEADING # change HEADING port
        sw $a0, 0($at) # to rotate robot
        nop
        jr $ra
        nop
```

end1 và end2:

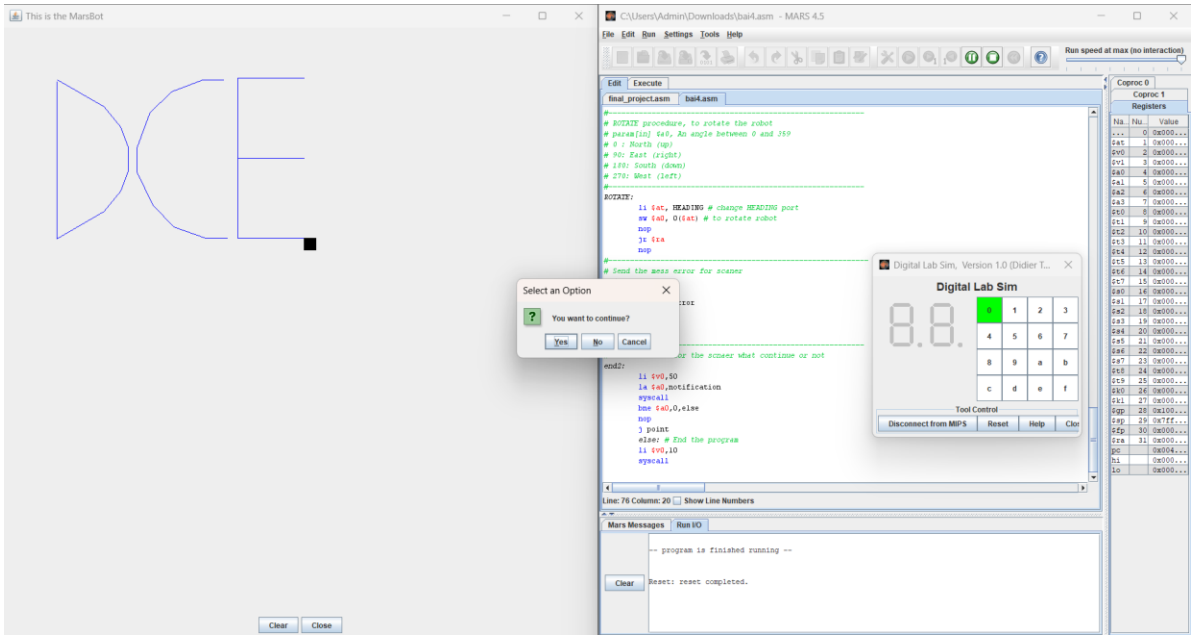
- Gửi thông báo lỗi hoặc thông báo cho máy quét, và xác định xem có tiếp tục hay không.

```
end1:
        li $v0,55
        la $a0, error
        syscall
        nop
        j point
#-----------------------------------------------------------------
# Send the mess for the scnaer what continue or not
end2:
        li $v0,50
        la $a0,notification
        syscall
        bne $a0,0,else
        nop
        j point
        else: # End the program
        li $v0,10
        syscall
```

4. Kết quả

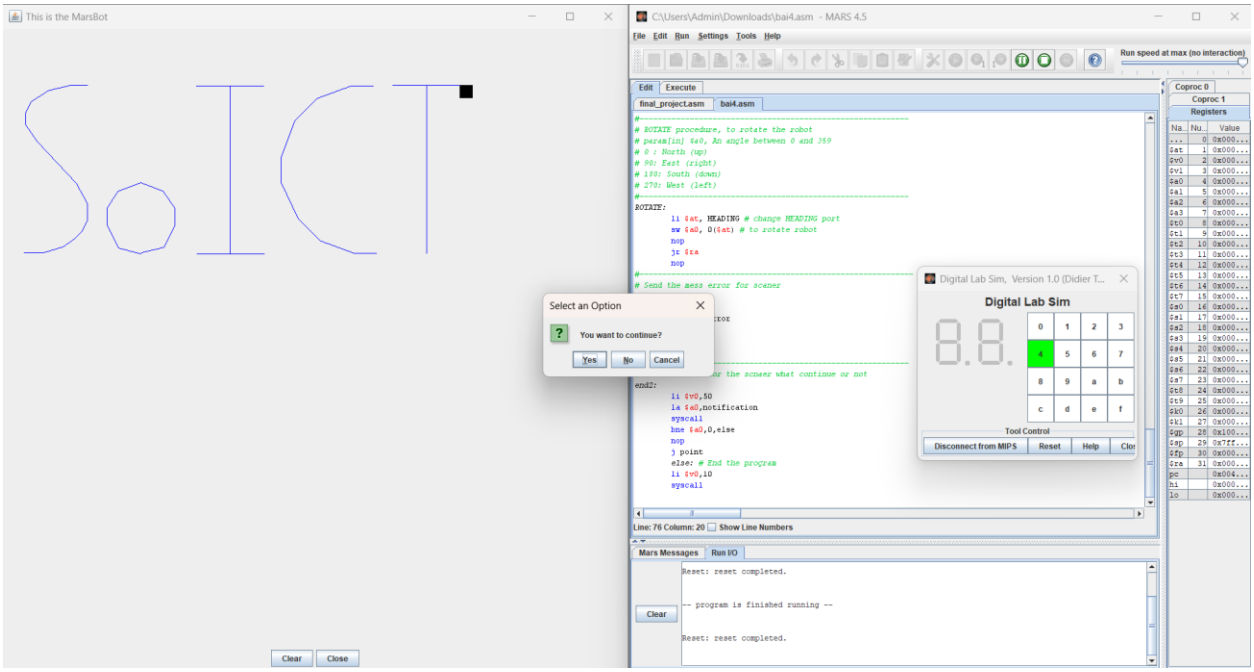
- Khi ấn phím 0

```
#-----------------------------D-------------------------------C-------------------------------E-------------------------------||
message1: .word 180 4000 0 90 4000 0 180 11800 1 60 4000 1 40 2000 1 20 1600 1 0 1800 1 -20 1600 1 -40 2000 1
                -60 3990 1 90 12500 0 270 1600 1 250 2520 1 225 2260 1 200 2800 1 180 1800 1 155 2800 1 135 2260 1
                 110 2520 1 90 1600 1 90 800 0 0 12000 1 90 5000 1 180 6000 0 270 5000 1 180 6000 0 90 5000 1 13
```



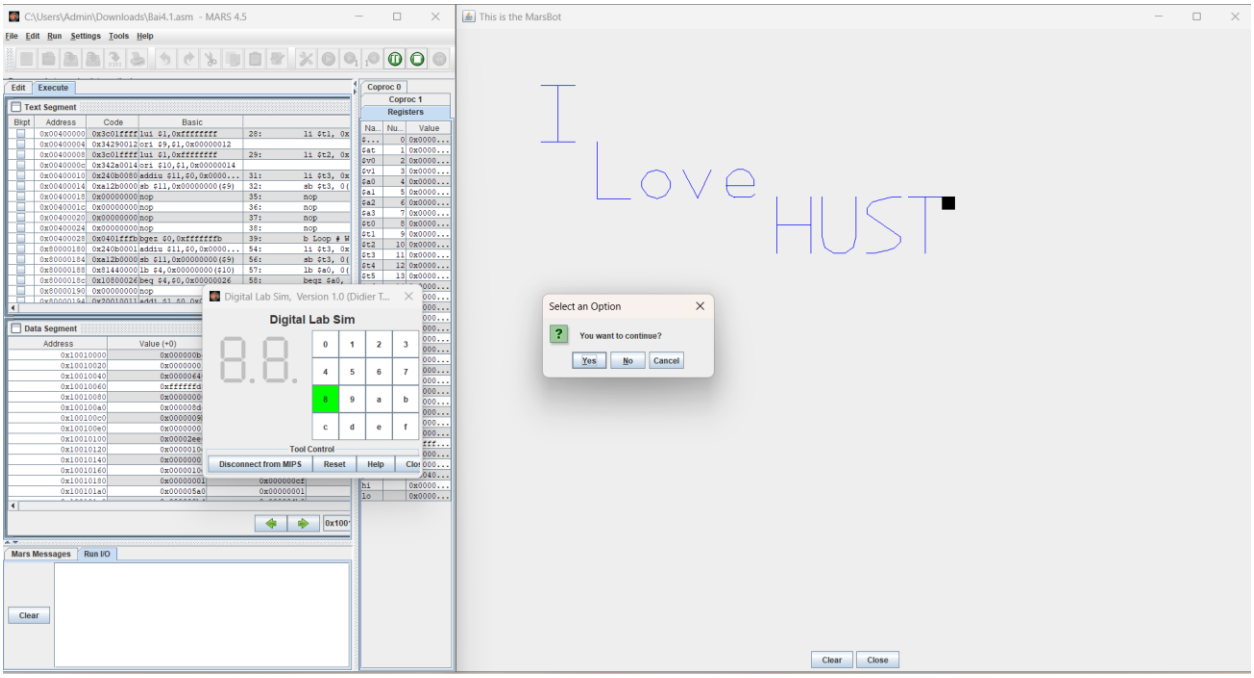
- Khi ấn phím 4

```
            #-----------------S---------------o---------------I---------------C---------------T-------------||
message2: .word 180 4000 0 90 6000 0 270 1200 1 256 1640 1 236 1440 1 207 900 1 180 800 1 146 1440 1 137 4100 1
                150 1600 1 180 1200 1 207 900 1 225 720 1 233 1000 1 254 1460 1 270 1400 1 90 8400 0 72 1280 1 34 1440 1 0 800 1 326 1440 1 288 1280 1 252 1280 1 214 1440 1 180 800 1
                 146 1440 1 108 1280 1 90 4000 0 90 4800 1 270 2400 0 0 12000 1 270 2400 0 90 4800 1 90 7800 0 270 1600 1 250 2520 1
                  225 2260 1 200 2800 1 180 1800 1 155 2800 1 135 2260 1 110 2520 1 90 1600 1 90 3600 0 0 12000 1 270 2400 0 90 4800 1 13
```



- Khi ấn phím 8:

```
            #-------------------------I------L------o--------v--------e---------H-------U----------S----------T-------------------
message3: .word 180 4000 0 90 6000 0 90 2400 1 270 1200 0 180 4000 1 270 1200 0 90 2400 1 90 1600 0 180 4000 1
                90 2400 1 90 1800 0 72 640 1 34 720 1 0 400 1 326 720 1 288 640 1 252 640 1 214 720 1 180 400 1
                 146 720 1 108 640 1 90 3000 0 333 2240 1 90 2000 0 205 2200 1  90 2600 0 297 440 1 333 440 1
                  0 800 1 45 560 1 63 440 1 90 400 1 117 440 1 146 720 1 180 400 1 270 2000 1 180 800 0 90 600 0
                   90 1200 1 45 280 1 225 280 0 90 1800 0 180 4000 1 0 2000 0 90 2400 1 180 2000 0 0 4000 1 90 800 0
                    180 2800 1 166 820 1 124 720 1 90 800 1 56 720 1 14 820 1 0 2800 1 90 3200 0 270 600 1 256 820 1
                    252 640 1 225 560 1 180 400 1 124 2880 1 180 400 1 225 560 1 252 640 1 256 820 1 270 600 1 90 4200 0 0 4000 1 270 1200 0 90 2400 1 13
```


C:\Users\Admin\Downloads\Bai4.1.asm - MARS 4.5
File Edit Run Settings Tools Help
Edit Execute
Text Segment
Data Segment
Mars Messages Run I/O
Clear
Digital Lab Sim, Version 1.0 (Didier T...
Digital Lab Sim
Tool Control
Disconnect from MIPS
Reset
Help
Coproc 0
Coproc 1
Registers
This is the MarsBot
I Love HUST
Select an Option
You want to continue?
Yes
No
Cancel
Clear
Close