ĐẠI HỌC BÁCH KHOA HÀ NỘI

BÁO CÁO BÀI TẬP LỚN

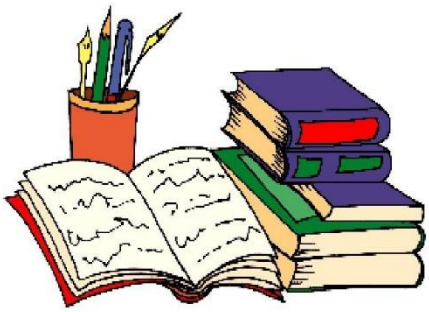
MÔN THỰC HÀNH KIẾN TRÚC MÁY TÍNH

Giảng viên hướng dẫn: Lê Bá Vui

Sinh viên thực hiện:

Lê Trường Giang -20205077 – Bài 1

Trần Minh Quân -20205015- Bài2



Bài 2:

I. Yêu cầu bài toán, mã nguồn

```
.eqv KEY_CODE 0xFFFF0004
.eqv KEY_READY 0xFFFF0000

.eqv SCREEN_MONITOR 0x10010000

.data
circle_end:     .word  1        # The end of the "circle" array
circle:         .word           # The pointer to the "circle" 2-dimentional array
.text
setup:
        addi    $s0, $0, 255    # x = 255
        addi    $s1, $0, 255    # y = 255
        addi    $s2, $0, 1      # dx = 1
        add     $s3, $0, $0     # dy = 0
        addi    $s4, $0, 20     # r = 20
        addi    $a0, $0, 50     # t = 50ms/frame
        jal     circle_data

input:
        li      $k0, KEY_READY      # Check whether there is input data
        lw      $t0, 0($k0)
        bne     $t0, 1, edge_check
        jal     direction_change

# Check whether the circle has touched the edge
edge_check:

right:
        bne     $s2, 1, left
        j       check_right

left:
        bne     $s2, -1, down
        j       check_left

down:
        bne     $s3, 1, up
        j       check_down

up:
```

```
        bne     $s3, -1, move_circle
        j       check_up

move_circle:
        add     $s5, $0, $0     # Set color to black
        jal     draw_circle     # Erase the old circle

        add     $s0, $s0, $s2   # Set x and y to the coordinates of the center of the new circle
        add     $s1, $s1, $s3
        li      $s5, 0x00FFFF00       # Set color to yellow
        jal     draw_circle     # Draw the new circle

loop:
        li $v0, 32              # Syscall value for sleep
        syscall
        j       input           # Renew the cycle

# Procedure below

circle_data:
        addi    $sp, $sp, -4    # Save $ra
        sw      $ra, 0($sp)
        la      $s5, circle     # $s5 becomes the pointer of the "circle" array
        mul     $a3, $s4, $s4   # $a3 = r^2
        add     $s7, $0, $0     # pixel x (px) = 0

pixel_data_loop:
        bgt     $s7, $s4, data_end
        mul     $t0, $s7, $s7   # $t0 = px^2
        sub     $a2, $a3, $t0   # $a2 = r^2 - px^2 = py^2
        jal     root            # $a2 = py

        add     $a1, $0, $s7    # $a1 = px
        add     $s6, $0, $0     # After saving (px, py), (-px, py), (-px, -py), (px, -py), we swap px
and py, then save (-py, px), (py, px), (py, -px), (-py, -px)

symmetric:
        beq     $s6, 2, finish
        jal     pixel_save      # px, py >= 0
        sub     $a1, $0, $a1
        jal     pixel_save      # px <= 0, py >= 0
        sub     $a2, $0, $a2
        jal     pixel_save      # px, py <= 0
        sub     $a1, $0, $a1
```

```
        jal     pixel_save      # px >= 0, py <= 0

        add     $t0, $0, $a1    # Swap px and -py
        add     $a1, $0, $a2
        add     $a2, $0, $t0
        addi    $s6, $s6, 1
        j       symmetric

finish:
        addi    $s7, $s7, 1
        j       pixel_data_loop

data_end:
        la      $t0, circle_end
        sw      $s5, 0($t0)     # Save the end address of the "circle" array
        lw      $ra, 0($sp)
        addi    $sp, $sp, 4
        jr      $ra

root:                           # Find the square root of $a2
        add     $t0, $0, $0     # Set $t0 = 0
        add     $t1, $0, $0     # $t1 = $t0^2

root_loop:
        beq     $t0, $s4, root_end      # If $t0 exceeds 20, 20 will be the square root
        addi    $t2, $t0, 1             # $t2 = $t0 + 1
        mul     $t2, $t2, $t2           # $t2 = ($t0 + 1)^2
        sub     $t3, $a2, $t1           # $t3 = $a2 - $t0^2
        bgez    $t3, continue           # If $t3 < 0, $t3 = -$t3
        sub     $t3, $0, $t3

continue:
        sub     $t4, $a2, $t2           # $t4 = $a2 - ($t0 + 1)^2
        bgez    $t4, compare            # If $t4 < 0, $t4 = -$t4
        sub     $t4, $0, $t4

compare:
        blt     $t4, $t3, root_continue         # If $t3 >= $t4, $t0 is not nearer to square root of
$a2 than $t0 + 1
        add     $a2, $0, $t0            # Else $t0 is the nearest number to square root of $a2
        jr      $ra

root_continue:
        addi    $t0, $t0, 1
```

```
        add     $t1, $0, $t2
        j       root_loop

root_end:
        add     $a2, $0, $t0
        jr      $ra

pixel_save:
        sw      $a1, 0($s5)     # Store px in the "circle" array
        sw      $a2, 4($s5)     # Store py in the "circle" array
        addi    $s5, $s5, 8     # Move the pointer to a null block
        jr      $ra

direction_change:
        li      $k0, KEY_CODE
        lw      $t0, 0($k0)

case_d:
        bne     $t0, 'd', case_a
        addi    $s2, $0, 1      # dx = 1
        add     $s3, $0, $0     # dy = 0
        jr      $ra

case_a:
        bne     $t0, 'a', case_s
        addi    $s2, $0, -1     # dx = -1
        add     $s3, $0, $0     # dy = 0
        jr      $ra

case_s:
        bne     $t0, 's', case_w
        add     $s2, $0, $0     # dx = 0
        addi    $s3, $0, 1      # dy = 1
        jr      $ra

case_w:
        bne     $t0, 'w', case_x
        add     $s2, $0, $0     # dx = 0
        addi    $s3, $0, -1     # dy = -1
        jr      $ra

case_x:
        bne     $t0, 'x', case_z
        addi    $a0, $a0, 10    # t += 10
```

```
        jr      $ra

case_z:
        bne     $t0, 'z', default
        beq     $a0, 0, default         # Only reduce t when t >= 0
        addi    $a0, $a0, -10  # t -= 10

default:
        jr      $ra

check_right:
        add     $t0, $s0, $s4   # Set $t0 to the right side of the circle
        beq     $t0, 511, reverse_direction  # Reverse direction if the side has touched the
edge
        j       move_circle    # Return if not

check_left:
        sub     $t0, $s0, $s4   # Set $t0 to the left side of the circle
        beq     $t0, 0, reverse_direction      # Reverse direction if the side has touched the
edge
        j       move_circle    # Return if not

check_down:
        add     $t0, $s1, $s4   # Set $t0 to the down side of the circle
        beq     $t0, 511, reverse_direction  # Reverse direction if the side has touched the
edge
        j       move_circle    # Return if not

check_up:
        sub     $t0, $s1, $s4   # Set $t0 to the up side of the circle
        beq     $t0, 0, reverse_direction      # Reverse direction if the side has touched the
edge
        j       move_circle    # Return if not

reverse_direction:
        sub     $s2, $0, $s2    # dx = -dx
        sub     $s3, $0, $s3    # dy = -dy
        j       move_circle

draw_circle:
        addi    $sp, $sp, -4    # Save $ra
        sw      $ra, 0($sp)
        la      $s6, circle_end
        lw      $s7, 0($s6)     # $s7 becomes the end address of the "circle" array
```

```
        la      $s6, circle     # $s6 becomes the pointer to the "circle" array

draw_loop:
        beq     $s6, $s7, draw_end  # Stop when $s6 = $s7
        lw      $a1, 0($s6)         # Get px
        lw      $a2, 4($s6)         # Get py
        jal     pixel_draw
        addi    $s6, $s6, 8         # Get to the next pixel
        j       draw_loop

draw_end:
        lw      $ra, 0($sp)
        addi    $sp, $sp, 4
        jr      $ra

pixel_draw:
        li      $t0, SCREEN_MONITOR
        add     $t1, $s0, $a1       # final x (fx) = x + px
        add     $t2, $s1, $a2       # fy = y + py
        sll     $t2, $t2, 9         # $t2 = fy * 512
        add     $t2, $t2, $t1       # $t2 += fx
        sll     $t2, $t2, 2         # $t2 *= 4
        add     $t0, $t0, $t2
        sw      $s5, 0($t0)
        jr      $ra
```

II. Tổng quát quá trình

B1: Set up các giá trị cơ bản: tâm đường tròn, đường kính, khoảng di chuyển dx dy (1px) tương ứng hướng di chuyển, $a0 chứa thời gian sleep.

B2: Tính toán khoảng cách tâm với các điểm thỏa mãn để tạo nên 1 hình tròn, lưu vào mảng circle chứa tất cả điểm có thể tạo thành 1 hình tròn.

B3: Đọc từ input và kiểm tra xem có tín hiệu đổi hướng di chuyển/ tăng giảm tốc độ không. Nếu không thay đổi thì kiểm tra chạm góc

B4: Kiểm tra vòng tròn đã chạm cạnh màn hình chưa (từ vòng lặp trước). Nếu chạm cạnh màn hình thì phải thay đổi hướng di chuyển

B5: Nếu đã kiểm tra góc xong tiến hành vẽ từng điểm ảnh của đường tròn bằng cách xóa hình tròn cũ, cập nhật giá trị mới và vẽ lại hình tròn

B6: Lặp quá trình từ bước 3 để có 1 hình tròn di chuyển.

III. Chi tiết quá trình:

Phần 1: Set up giá trị

```
setup:
addi    $s0, $0, 255    # x = 255
addi    $s1, $0, 255    # y = 255
addi    $s2, $0, 1      # dx = 1
add     $s3, $0, $0     # dy = 0
addi    $s4, $0, 20     # r = 20
addi    $a0, $0, 50     # t = 50ms/frame
jal     circle_data
```

-Giải thích: hình tròn có tâm nằm chính giữa map 512x512, mặc định là di chuyển sang phải nên dx=1, dy=0. Bán kính 20, tốc độ ngủ 50ms/frame

-Ý tưởng thay đổi tốc độ: Thay đổi thời gian ngủ ít-> tốc độ giảm

thời gian ngủ nhiều->tốc độ tăng

```
        direction_change:
    li      $k0, KEY_CODE
    lw      $t0, 0($k0)

case_d:
    bne     $t0, 'd', case_a
    addi    $s2, $0, 1      # dx = 1
    add     $s3, $0, $0     # dy = 0
    jr      $ra

case_a:
    bne     $t0, 'a', case_s
    addi    $s2, $0, -1     # dx = -1
    add     $s3, $0, $0     # dy = 0
    jr      $ra

case_s:
    bne     $t0, 's', case_w
    add     $s2, $0, $0     # dx = 0
    addi    $s3, $0, 1      # dy = 1
    jr      $ra

case_w:
    bne     $t0, 'w', case_x
    add     $s2, $0, $0     # dx = 0
```

```
        addi    $s3, $0, -1     # dy = -1
        jr      $ra

case_x:
        bne     $t0, 'x', case_z
        addi    $a0, $a0, 10    # t += 10
        jr      $ra

case_z:
        bne     $t0, 'z', default
        beq     $a0, 0, default         # Only reduce t when t >= 0
        addi    $a0, $a0, -10   # t -= 10

default:
        jr      $ra
```

Mỗi phím được tạo chức năng thay đổi hướng và tốc độ.

Phần 2: Tính toán khoảng cách để vẽ hình tròn

-Đầu tiên tạo con trỏ để lưu giá trị khoảng cách vào mảng circle.
-Tạo cặp (px, py) thỏa mãn thuộc đường tròn bằng cách:
+ Chọn px nguyên bắt đầu từ 0. Tính (py)^2=r^2- (px)^2
+ Chọn số nguyên z có bình phương gần (py)^2 nhất. Chọn z là py
+ Được cặp giá trị (px, z) chính là (px, py)
-Vì px, py dương nên ta sẽ tiến hành đảo để tạo các cặp (-px, py), (-px, -py), (px, -py) để có hình tròn
-Đảo ngược px cho py để chắc chắn hình tròn hoàn chỉnh nhất có thể. Vậy 1 lần tìm ra cặp (px, py) sẽ có 8 điểm sinh ra.
-Lưu các điểm trên vào mảng circle. Hoàn thành dữ liệu hình tròn

Phần 3: Kiểm tra xem có tín hiệu đổi hướng di chuyển/ tăng giảm tốc độ

-Mặc định của tín hiệu là di chuyển sang phải. Khi tiếp nhận phím điều hướng di chuyển/ tốc độ, giá trị của dx và dy sẽ được điều chỉnh để sẵn sàng cho việc di chuyển từ trạng thái cũ sang mới (như code giải thích phần 1 ở trên).

Phần 4: Kiểm tra vòng tròn đã chạm cạnh màn hình hay chưa

-Kiểm tra bằng cách tính tọa độ tâm (x, y) cộng thêm bán kính đã chạm tới rìa màn hình (0, 511) chưa. Nếu chạm đến thì tiến hành đổi ngược lại dx, dy tùy theo hướng để điều chỉnh lại trạng thái di chuyển

```
        check_right:
        add     $t0, $s0, $s4   # Set $t0 to the right side of the circle
        beq     $t0, 511, reverse_direction  # Reverse direction if the side has touched the
edge
        j       move_circle    # Return if not
```

```
check_left:
        sub     $t0, $s0, $s4   # Set $t0 to the left side of the circle
        beq     $t0, 0, reverse_direction      # Reverse direction if the side has touched the
edge
        j       move_circle   # Return if not

check_down:
        add     $t0, $s1, $s4   # Set $t0 to the down side of the circle
        beq     $t0, 511, reverse_direction  # Reverse direction if the side has touched the
edge
        j       move_circle   # Return if not

check_up:
        sub     $t0, $s1, $s4   # Set $t0 to the up side of the circle
        beq     $t0, 0, reverse_direction      # Reverse direction if the side has touched the
edge
        j       move_circle   # Return if not

reverse_direction:
        sub     $s2, $0, $s2    # dx = -dx
        sub     $s3, $0, $s3    # dy = -dy
        j       move_circle
```

Phần 5: Cập nhật giá trị mới và vẽ lại hình tròn

-Xóa hình tròn từ lần cũ bằng cách đổi màu sang màu đen, vẽ hình tròn, đổi lại sang màu vàng, cập nhật cách giá trị rồi vẽ thêm 1 lần nữa.

- Cách vẽ hình tròn mới:

+ Tính toán giá trị x, y sau khi cộng thêm (dx, dy) theo hướng phù hợp.

+ Cộng thêm (px, py) từ mảng hình tròn để được những điểm tạo nên hình tròn xung quanh tâm (x, y)

+Xác định vị trí trên bit maps và tiến hành vẽ.

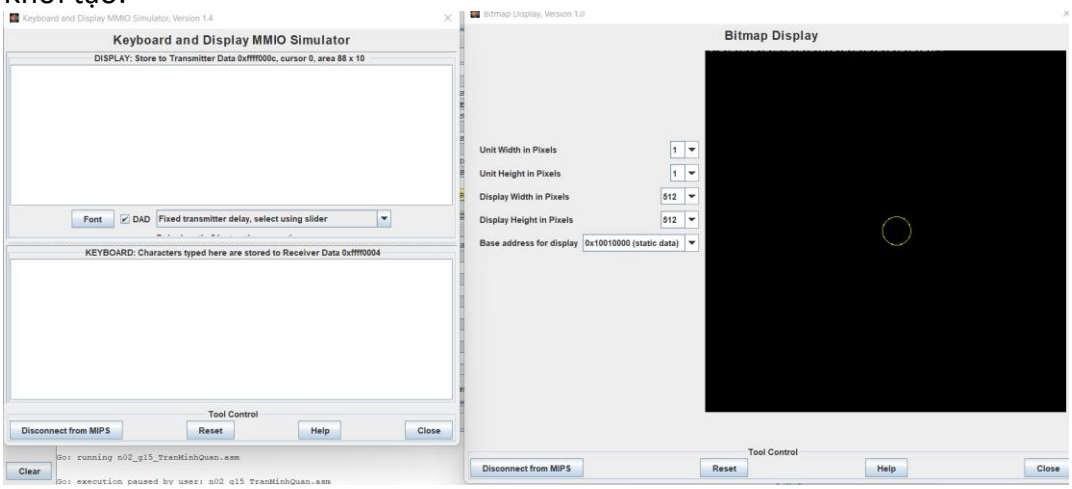
+Quá trình vẽ dừng lại khi mảng circle chạm đến phần tử cuối(đã được lưu địa chỉ trước đó vào circle-end)

```
        draw_circle:
        addi    $sp, $sp, -4    # Save $ra
        sw      $ra, 0($sp)
        la      $s6, circle_end
        lw      $s7, 0($s6)     # $s7 becomes the end address of the "circle" array
        la      $s6, circle     # $s6 becomes the pointer to the "circle" array

    draw_loop:
        beq     $s6, $s7, draw_end   # Stop when $s6 = $s7
        lw      $a1, 0($s6)          # Get px
        lw      $a2, 4($s6)          # Get py
```

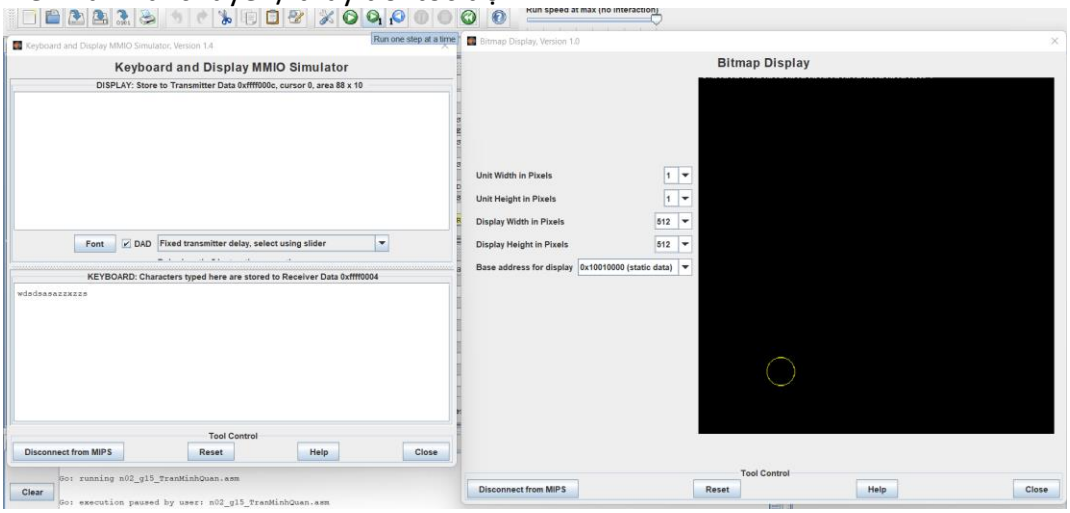
```
        jal     pixel_draw
        addi    $s6, $s6, 8             # Get to the next pixel
        j       draw_loop

draw_end:
        lw      $ra, 0($sp)
        addi    $sp, $sp, 4
        jr      $ra

pixel_draw:
        li      $t0, SCREEN_MONITOR
        add     $t1, $s0, $a1           # final x (fx) = x + px
        add     $t2, $s1, $a2           # fy = y + py
        sll     $t2, $t2, 9             # $t2 = fy * 512
        add     $t2, $t2, $t1           # $t2 += fx
        sll     $t2, $t2, 2             # $t2 *= 4
        add     $t0, $t0, $t2
        sw      $s5, 0($t0)
        jr      $ra
```

IV. Màn hình chạy chương trình

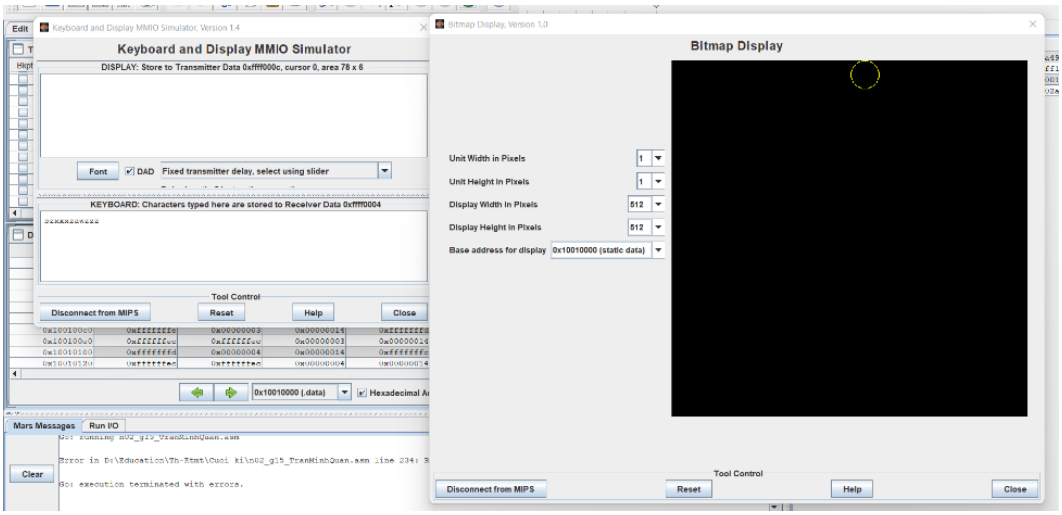
Khởi tạo:



Tiến hành di chuyển/ thay đổi tốc độ



Trường hợp đặc biệt: Hình tròn chạm vào cạnh màn hình



Bài 1:

I. Yêu cầu bài toán, mã nguồn

1)Yêu cầu bài toán

1. Curiosity Marsbot

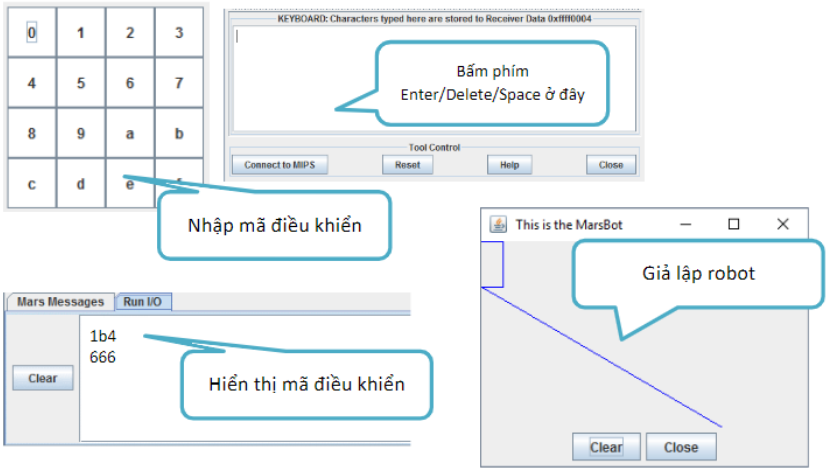
Xe tự hành Curiosity Marsbot chạy trên sao Hỏa, được vận hành từ xa bởi các lập trình viên trên Trái Đất. Bằng cách gửi đi các mã điều khiển từ một bàn phím ma trận, lập trình viên điều khiển quá trình di chuyển của Marbot như sau:

Mã điều khiển	Ý nghĩa
1b4	Marsbot bắt đầu chuyển động
c68	Marsbot đứng im
444	Rẽ trái 90 độ so với phương chuyển động gần nhất
666	Rẽ phải 90 độ so vơi phương chuyển động gần nhất
dad	Bắt đầu để lại vết trên đường
cbc	Chấm dứt để lại vết trên đường
999	Tự động đi theo lộ trình ngược lại. Không vẽ vết, không nhận mã khác cho tới khi kết thúc lộ trình ngược. Mô tả: Marsbot được lập trình để nhớ lại toàn bộ lịch sử các mã điều khiển và khoảng thời gian giữa các lần đổi mã. Vì vậy, nó có thể đảo ngược lại lộ trình để quay về điểm xuất phát.

Sau khi nhận mã điều khiển, Curiosity Marsbot sẽ không xử lý ngay, mà phải đợi lệnh kích hoạt mã từ bàn phím Keyboard & Display MMIO Simulator. Có 3 lệnh như vậy:

Kích hoạt mã	Ý nghĩa
Phím Enter	Kết thúc nhập mã và yêu cầu Marsbot thực thi.
Phím Delete	Xóa toàn bộ mã điều khiển đang nhập.
Phím Space	Lặp lại lệnh đã thực hiện trước đó.

Hãy lập trình để Marsbot có thể hoạt động như đã mô tả.
Đồng thời bổ sung thêm tính năng: mỗi khi gửi một mã điều khiển cho Marsbot, hiển thị mã đó lên màn hình console để người xem có thể giám sát lộ trình của xe.



2)Mã nguồn

```
.eqv IN_ADRESS_HEXA_KEYBOARD 0xFFFF0012
.eqv OUT_ADRESS_HEXA_KEYBOARD 0xFFFF0014
.eqv KEY_CODE 0xFFFF0004        # ma ASCII tu ban phim, 1 byte
.eqv KEY_READY 0xFFFF0000       # =1 neu co ma khoa moi
# tu dong xoa sau lw
#------------------------------------------------------------------------------
# Marsbot
.eqv HEADING 0xffff8010 # so nguyen: tu 0 den 359 do
```

```
# 0 : Di len ( Bac )
# 90: Sang phai ( Dong )
# 180: Di xuong ( Nam )
# 270: Sang Trai ( Tay )
.eqv MOVING 0xffff8050 # Boolean cho lenh kich hoat thuc thi
.eqv LEAVETRACK 0xffff8020 # Boolean (0 or non-0):

.eqv WHEREX 0xffff8030 # vi tri x cua MarsBot
.eqv WHEREY 0xffff8040 # vi tri y cua MarsBot


.data
# gia tri nhan vao
.eqv KEY_0 0x11
.eqv KEY_1 0x21
.eqv KEY_2 0x41
.eqv KEY_3 0x81
.eqv KEY_4 0x12
.eqv KEY_5 0x22
.eqv KEY_6 0x42
.eqv KEY_7 0x82
.eqv KEY_8 0x14
.eqv KEY_9 0x24
.eqv KEY_a 0x44
.eqv KEY_b 0x84
.eqv KEY_c 0x18
.eqv KEY_d 0x28
.eqv KEY_e 0x48
.eqv KEY_f 0x88
#------------------------------------------------------------------------------
#tao cac lenh chuc nang
DiChuyen: .asciiz "1b4"                 # di chuyen
DungLai: .asciiz "c68"                  # dung im
ReTrai: .asciiz "444"           # re trai 90 do
RePhai: .asciiz "666"           # re phai 90 do
DauVet: .asciiz "dad"           # ve
BoDauVet: .asciiz "cbc"         # ko ve
QuayXe: .asciiz "999"           # quay tro lai, ko ve. ko nhan ma toi khi het
msg_output: .asciiz "Da nhap ma dieu khien sai !"
#------------------------------------------------------------------------------
MaDieuKhien: .space 50 # dau vao ma dieu khien
MaTruocDo: .space 50 # dau vao ma dieu khien
DoDai: .word 0  # ma khiem soat do dai
nowHeading: .word 0
#------------------------------------------------------------------------------
dauvet: .space 600
DoDaiDauVet: .word 12                   #bytes

.text
main:
li $k0, KEY_CODE                # ma khoa
li $k1, KEY_READY               # khoa san sang

# ngat ma tran ban phim 4x4 trong Digital Lab Sim

li $t1, IN_ADRESS_HEXA_KEYBOARD
li $t3, 0x80 # bit 7 = 1 de kich hoat
```

```
sb $t3, 0($t1)
#-----------------------------------------------------------
loop:               nop
WaitForKey:
lw $t5, 0($k1)                  #$t5 = [$k1] = khoa san sang
beq $t5, $zero, WaitForKey      #neu $t5 == 0 quay lai WaitForKey cho tin hieu
nop
beq $t5, $zero, WaitForKey      #neu $t5 == 0 quay lai WaitForKey cho tin hieu
ReadKey:
lw $t6, 0($k0)                  #$t6 = [$k0] = KEY_CODE
beq $t6, 127 , delete       #if $t6 == Del thi xoa toan bo dau vao
#127 la ma Delete xong ASCII


beq $t6, ' ' ,laplai #neu $t6 == " " thi lap lai lenh truoc do

bne $t6, '\n' , loop            #neu t6 = \n thi quay lai loop de lay code dieu khien
nop
bne $t6, '\n' , loop
CheckMaDieuKhien:       #kiem tra ma
la $s2, DoDai       #ma kiem soat do dai
lw $s2, 0($s2)
#----------------
bne $s2, 3, pushErrorMess       #thong bao ma dieu khien dua vao la sai

la $s3, DiChuyen            #lenh di chuyen
jal SoSanh                  #kiem tra chuoi
beq $t0, 1, go

la $s3, DungLai             #lenh dung
jal SoSanh                  #kiem tra chuoi
beq $t0, 1, stop

la $s3, ReTrai              #lenh re trai
jal SoSanh                  #kiem tra chuoi
beq $t0, 1, goLeft

la $s3, RePhai              #lenh re phai
jal SoSanh                  #kiem tra chuoi
beq $t0, 1, goRight

la $s3, DauVet              #lenh ve
jal SoSanh                  #kiem tra chuoi
beq $t0, 1, track

la $s3, BoDauVet            #lenh khong ve
jal SoSanh                  #kiem tra chuoi
beq $t0, 1, untrack

la $s3, QuayXe              #lenh quay lai
jal SoSanh                  #kiem tra chuoi
beq $t0, 1, goBack

beq $t0, 0, pushErrorMess#ma dieu khien dua vao la sai

printMaDieuKhien:       # in ra ma dieu khien tren Mars Messages
li $v0, 4
```

```
la $a0, MaDieuKhien                    #dau vao cua code
syscall
nop

#sao chep MaDIeuKien vao MaTruocDo

la $s1, MaDieuKhien
la $s2, MaTruocDo
jal strcpy
la $s1, DoDai
lw $t7, 0($s1)


delete:                        #tiep theo
jal xoa    #Loai bo chuoi InputMaDieuKhien
nop
j loop                         #quay lai nhan key moi

laplai:


la $s1, MaTruocDo
la $s2, MaDieuKhien
jal strcpy
la $s1, DoDai
sw $t7, 0($s1)


j CheckMaDieuKhien

# ham luudauvet, luu tru duong dan cua MarsBot den bien duong dan

luudauvet:
#luu lai tung ki tu vao cac bien
#sp tro toi dinh cua stack
addi $sp,$sp,4
sw $t1, 0($sp)
addi $sp,$sp,4
sw $t2, 0($sp)
addi $sp,$sp,4
sw $t3, 0($sp)
addi $sp,$sp,4
sw $t4, 0($sp)

addi $sp,$sp,4
sw $s1, 0($sp)
addi $sp,$sp,4
sw $s2, 0($sp)
addi $sp,$sp,4
sw $s3, 0($sp)
addi $sp,$sp,4
sw $s4, 0($sp)

#thuat tuan
li $t1, WHEREX
lw $s1, 0($t1)                 #s1 = x
li $t2, WHEREY
```

```
lw $s2, 0($t2)              #s2 = y

la $s4, nowHeading
lw $s4, 0($s4)              #s4 = now heading

la $t3, DoDaiDauVet
lw $s3, 0($t3)              #$s3 = DoDaiDauVet (dv: byte)

la $t4, dauvet
add $t4, $t4, $s3  #vi tri de luu tru vao mang dauvet
#chuyen de lieu tu thanh ghi ra t4 ( 3 ki tu dieu khien )
sw $s1, 0($t4)              #store x
sw $s2, 4($t4)              #store y
sw $s4, 8($t4)              #store heading

addi $s3, $s3, 12  #cap nhat DoDaiDauVet
#12 = 3 (word) x 4 (bytes)
sw $s3, 0($t3)

#khoi phuc lai gia tri trong stack
lw $s4, 0($sp)
addi $sp,$sp,-4
lw $s3, 0($sp)
addi $sp,$sp,-4
lw $s2, 0($sp)
addi $sp,$sp,-4
lw $s1, 0($sp)
addi $sp,$sp,-4
lw $t4, 0($sp)
addi $sp,$sp,-4
lw $t3, 0($sp)
addi $sp,$sp,-4
lw $t2, 0($sp)
addi $sp,$sp,-4
lw $t1, 0($sp)
addi $sp,$sp,-4

jr $ra
nop
jr $ra

# ham con strcpy sao chep noi dung ben trong chuoi nay sang chuoi khac


strcpy:
add $s0, $zero, $zero # s0 = i = 0
L1:
add $t1,$s0, $s1 # t1 = s0 + a1 = dia chi y[i]
lb $t2, 0($t1) # t2 = gia tri tai dia chi t1 = gia tri cua y[i]
add $t3, $s2, $s0 # t3 = dia chi bat dau cua xau dich + index = dia chi cua x[i]
sb $t2, 0($t3) # Gan gia tri cua y[i] cho thanh ghi co dia chi t3 (x[i])
beq $t2, $zero, end_of_strcpy1 #Neu ki tu vua doc duoc la KI TU KET THUC CHUOI, KET THUC
nop
addi $s0, $s0, 1 # s0 = s0+1 = i+1
j L1
nop
end_of_strcpy1:
```

```
jr $ra
nop
jr $ra

# ham xoa , loai bo chuoi ma dieu kien dau vao

xoa:
#sao luu vao stack
addi $sp,$sp,4
sw $t1, 0($sp)
addi $sp,$sp,4
sw $t2, 0($sp)
addi $sp,$sp,4
sw $s1, 0($sp)
addi $sp,$sp,4
sw $t3, 0($sp)
addi $sp,$sp,4
sw $s2, 0($sp)

#thuat tuan
la $s2, DoDai
lw $t3, 0($s2)                  #$t3 = DoDai
addi $t1, $zero, -1             #$t1 = -1 = i
addi $t2, $zero, 0          #$t2 = '\0'
la $s1, MaDieuKhien
addi $s1, $s1, -1
loai_bo: addi $t1, $t1, 1       #i++
add $s1, $s1, 1                 #$s1 = MaDieuKhien + i
sb $t2, 0($s1)                  #MaDieuKhien[i] = '\0'

bne $t1, $t3, loai_bo           #if $t1 <=3 continue loop
nop
bne $t1, $t3, loai_bo

add $t3, $zero, $zero
sw $t3, 0($s2)                  #DoDai = 0

#khoi phuc
lw $s2, 0($sp)
addi $sp,$sp,-4
lw $t3, 0($sp)
addi $sp,$sp,-4
lw $s1, 0($sp)
addi $sp,$sp,-4
lw $t2, 0($sp)
addi $sp,$sp,-4
lw $t1, 0($sp)
addi $sp,$sp,-4

jr $ra
nop
jr $ra

# thuat tuan goBack, dieu khien MarsBot quay lai


goBack:  la $s7, dauvet         # ma tran
```

```
la $s5, DoDaiDauVet         # byte toi da
lw $s5, 0($s5)
add $s7, $s7, $s5
begin:    addi $s5, $s5, -12            #lui lai 1 cau truc
addi $s7, $s7, -12 #vi tri cua thong tin ve canh cuoi cung
lw $s6, 8($s7)              #huong cua canh cuoi cung
addi $s6, $s6, 180          #nguoc lai huong cua canh cuoi cung

la $t8, nowHeading          #marsbot quay nguoc lai
sw $s6, 0($t8)
jal ROTATE

go_to_first_point_of_edge:          #toi diem dau cua canh
lw $t9, 0($s7)              #toa do x cua diem dau tien cua canh
li $t8, WHEREX              #toa do x hien tai
lw $t8, 0($t8)

bne $t8, $t9, go_to_first_point_of_edge

lw $t9, 4($s7)              #toa do y cua diem dau tien cua canh
li $t8, WHEREY              #toa do y hien tai
lw $t8, 0($t8)

bne $t8, $t9, go_to_first_point_of_edge

beq $s5, 0, finish

j begin

finish:    jal STOP
la $t8, nowHeading
add $s6, $zero, $zero
sw $s6, 0($t8)              #cap nhat heading
la $t8, DoDaiDauVet
sw $s5, 0($t8)              #cap nhat DoDaiDauVet = 0
jal ROTATE
j printMaDieuKhien
#---------------------------------------------------------------
# thuat tuan track ,dieu khien MarsBot de theo doi va in ma dieu khien
# param[in] none
#---------------------------------------------------------------
track:    jal TRACK
j printMaDieuKhien

# quy trinh untrack, bo theo doi,
#dieu khien MarsBot de bo theo doi va in ma dieu khien

untrack: jal UNTRACK
j printMaDieuKhien

# thuat tuan go, dieu khien MarsBot de di va in ra ma dieu khien

go:        jal GO
j printMaDieuKhien

# thuat tuan stop, dieu khien MarsBot dung va in ra ma dieu khien
```

```
stop:      jal STOP
j printMaDieuKhien

# thuat tuan goRight , dieu khien MarsBot sang trai va in ma dieu khien

goRight:la $s5, nowHeading
lw $s6, 0($s5)              # $s6 = $s5 = n?Heading
addi $s6, $s6, 90  # ( tang len 90 )
sw $s6, 0($s5)              # cap nhat nowHeading
jal luudauvet               # luu tru duong dan cua MarsBot vao stack
jal ROTATE                  # dieu khien robot xoay
j printMaDieuKhien          # in ra ma dieu khien

# thuat tuan goLeft,dieu khien MarsBot sang phai va in ma dieu khien

goLeft:   la $s5, nowHeading
lw $s6, 0($s5)              # $6 = nowHeading
addi $s6, $s6, -90 # cong s6 them -90, xoay phai 90 do
sw $s6, 0($s5)              # cap nhat nowHeading
jal luudauvet               # luu tru duong dan cua MarsBot vao stack
jal ROTATE                  # dieu khien robot xoay
j printMaDieuKhien          # in ra ma dieu khien


# thuat tuan SoSanh , kiem tra MaDieuKhien
#                                   kiem tra co bang voi chuoi s ( luu trong $s3)
#                                   Do dai 2 chuoi la nhu nhau

SoSanh:
#sao luu vao stack
addi $sp,$sp,4
sw $t1, 0($sp)
addi $sp,$sp,4
sw $s1, 0($sp)
addi $sp,$sp,4
sw $t2, 0($sp)
addi $sp,$sp,4
sw $t3, 0($sp)

#thuat tuan
addi $t1, $zero, -1                 #$t1 = -1 = i
add $t0, $zero, $zero
la $s1, MaDieuKhien                 #$s1 = MaDieuKhien
so_sanh: addi $t1, $t1, 1           #i++

add $t2, $s1, $t1           #$t2 = MaDieuKhien + i
lb $t2, 0($t2)                      #$t2 = MaDieuKhien[i]

add $t3, $s3, $t1           #$t3 = s + i
lb $t3, 0($t3)                      #$t3 = s[i]

bne $t2, $t3, KhacNhau              #if $t2 != $t3 -> isNotequal

bne $t1, 2, so_sanh         #if $t1 <=2 tiep tuc toi loop
nop
bne $t1, 2, so_sanh
GiongNhau:
```

```
#sao luu vao stack
lw $t3, 0($sp)
addi $sp,$sp,-4
lw $t2, 0($sp)
addi $sp,$sp,-4
lw $s1, 0($sp)
addi $sp,$sp,-4
lw $t1, 0($sp)
addi $sp,$sp,-4

add $t0, $zero, 1  #cap nhat $t0
jr $ra
nop
jr $ra
KhacNhau:
#restore
lw $t3, 0($sp)
addi $sp,$sp,-4
lw $t2, 0($sp)
addi $sp,$sp,-4
lw $s1, 0($sp)
addi $sp,$sp,-4
lw $t1, 0($sp)
addi $sp,$sp,-4

add $t0, $zero, $zero        #cap nhat $t0
jr $ra
nop
jr $ra

# thuat tuan pushErrorMess , thong bao ma bi sai

pushErrorMess: li $v0, 4
la $a0, MaDieuKhien
syscall
nop

li $v0, 55
la $a0, msg_output
syscall
nop
nop
j delete
nop
j delete

# thuat tuan GO , bat dau chay

GO:       #sao luu vao stack
addi $sp,$sp,4
sw $at,0($sp)
addi $sp,$sp,4
sw $k0,0($sp)
#thuat tuan
li $at, MOVING                # lenh kich hoat duoc thuc thi
addi $k0, $zero,1 # to logic 1,
sb $k0, 0($at)                # bat dau chay
```

```
#khoi phuc
lw $k0, 0($sp)
addi $sp,$sp,-4
lw $at, 0($sp)
addi $sp,$sp,-4

jr $ra
nop
jr $ra

# thuat tuan STOP , dung chay

STOP:     #sao luu
addi $sp,$sp,4
sw $at,0($sp)
#thuat tuan
li $at, MOVING                  # thay doi sang MOVING
sb $zero, 0($at)                # dung lai
#khoi phuc
lw $at, 0($sp)
addi $sp,$sp,-4

jr $ra
nop
jr $ra

# thuat tuan TRACK , ve duong

TRACK:   #sao luu
addi $sp,$sp,4
sw $at,0($sp)
addi $sp,$sp,4
sw $k0,0($sp)
#thuat tuan
li $at, LEAVETRACK              # thay doi sang LEAVETRACK
addi $k0, $zero,1  # to logic 1,
sb $k0, 0($at)                  # to start tracking
#khoi phuc
lw $k0, 0($sp)
addi $sp,$sp,-4
lw $at, 0($sp)
addi $sp,$sp,-4

jr $ra
nop
jr $ra

# thuat tuan UNTRACK , dung ve duong

UNTRACK:
#sao luu
addi $sp,$sp,4
sw $at,0($sp)
#thuat tuan
li $at, LEAVETRACK              # thay doi sang cong LEAVETRACK va cho = 0
sb $zero, 0($at)    # dung ve
#khoi phuc
```

```
lw $at, 0($sp)
addi $sp,$sp,-4

jr $ra
nop
jr $ra

# thuat tuan ROTATE_RIGHT , dieu khien robot xoay

ROTATE:
#sao luu
addi $sp,$sp,4
sw $t1,0($sp)
addi $sp,$sp,4
sw $t2,0($sp)
addi $sp,$sp,4
sw $t3,0($sp)
#thuat tuan
li $t1, HEADING             # thay doi xong sang HEADING
la $t2, nowHeading
lw $t3, 0($t2)              # t3 = nowHeading
sw $t3, 0($t1)              # xoay robot
#khoi phuc
lw $t3, 0($sp)
addi $sp,$sp,-4
lw $t2, 0($sp)
addi $sp,$sp,-4
lw $t1, 0($sp)
addi $sp,$sp,-4

jr $ra
nop
jr $ra

#================================================================================
# GENERAL INTERRUPT SERVED ROUTINE cho tat ca interrupts
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
.ktext 0x80000180
#---------------------------------------------------
# luu cac tap tin vao stack
#---------------------------------------------------
backup:  #sao luu tu thanh ghi vao stack
addi $sp,$sp,4
sw $ra,0($sp)
addi $sp,$sp,4
sw $t1,0($sp)
addi $sp,$sp,4
sw $t2,0($sp)
addi $sp,$sp,4
sw $t3,0($sp)
addi $sp,$sp,4
sw $a0,0($sp)
addi $sp,$sp,4
sw $at,0($sp)
addi $sp,$sp,4
sw $s0,0($sp)
addi $sp,$sp,4
```

```
sw $s1,0($sp)
addi $sp,$sp,4
sw $s2,0($sp)
addi $sp,$sp,4
sw $t4,0($sp)
addi $sp,$sp,4
sw $s3,0($sp)
#------------------------------------------------------
# thuat tuan
#------------------------------------------------------
get_cod:            #chuyen dia chi
li $t1, IN_ADRESS_HEXA_KEYBOARD                 #dia chi vao
li $t2, OUT_ADRESS_HEXA_KEYBOARD                #dia chi ra
scan_row1:          #nhan tin hieu dong 1
li $t3, 0x11
sb $t3, 0($t1)
lbu $a0, 0($t2)             #dia chi ra
bnez $a0, get_code_in_char
scan_row2:          #nhan tin hieu dong 2
li $t3, 0x12
sb $t3, 0($t1)
lbu $a0, 0($t2)             #dia chi ra
bnez $a0, get_code_in_char
scan_row3:          #nhan tin hieu dong 3
li $t3, 0x14
sb $t3, 0($t1)
lbu $a0, 0($t2)             #dia chi ra
bnez $a0, get_code_in_char
scan_row4:          #nhan tin hieu dong 4
li $t3, 0x18
sb $t3, 0($t1)
lbu $a0, 0($t2)             #dia chi ra
bnez $a0, get_code_in_char
get_code_in_char:
beq $a0, KEY_0, case_0
beq $a0, KEY_1, case_1
beq $a0, KEY_2, case_2
beq $a0, KEY_3, case_3
beq $a0, KEY_4, case_4
beq $a0, KEY_5, case_5
beq $a0, KEY_6, case_6
beq $a0, KEY_7, case_7
beq $a0, KEY_8, case_8
beq $a0, KEY_9, case_9
beq $a0, KEY_a, case_a
beq $a0, KEY_b, case_b
beq $a0, KEY_c, case_c
beq $a0, KEY_d, case_d
beq $a0, KEY_e, case_e
beq $a0, KEY_f, case_f

#luu vao s0 nhung kieu char
case_0:  li $s0, '0'
j store_code
case_1:  li $s0, '1'
j store_code
case_2:  li $s0, '2'
```

```
j store_code
case_3:  li $s0, '3'
j store_code
case_4:  li $s0, '4'
j store_code
case_5:  li $s0, '5'
j store_code
case_6:  li $s0, '6'
j store_code
case_7:  li $s0, '7'
j store_code
case_8:  li $s0, '8'
j store_code
case_9:  li $s0, '9'
j store_code
case_a:  li $s0, 'a'
j store_code
case_b:  li $s0, 'b'
j store_code
case_c:  li $s0, 'c'
j store_code
case_d:  li $s0, 'd'
j store_code
case_e:  li $s0,     'e'
j store_code
case_f:   li $s0, 'f'
j store_code
store_code:
la $s1, MaDieuKhien
la $s2, DoDai
lw $s3, 0($s2)                          #$s3 = so ki tu trong mang MaDieuKhien
addi $t4, $t4, -1               #$t4 = i
for_loop_to_store_code:
addi $t4, $t4, 1
bne $t4, $s3, for_loop_to_store_code
add $s1, $s1, $t4  #$s1 = MaDieuKhien + i
sb  $s0, 0($s1)                 #MaDieuKhien[i] = $s0

addi $s0, $zero, '\n'           #them ky tu '\n' vao cuoi chuoi
addi $s1, $s1, 1                #them ky tu '\n' vao cuoi chuoi
sb  $s0, 0($s1)                 #them ky tu '\n' vao cuoi chuoi


addi $s3, $s3, 1
sw $s3, 0($s2)                  #cap nhat do dai cua InputMaDieuKhien

#---------------------------------------------------------
# Dang gia tra ve dia chi cua main routine
# epc <= epc + 4
#---------------------------------------------------------
next_pc:
mfc0 $at, $14                   # $at <= Coproc0.$14 = Coproc0.epc
addi $at, $at, 4                # $at = $at + 4 (next instruction)
mtc0 $at, $14                   # Coproc0.$14 = Coproc0.epc <= $at

# khoi phuc tap tin trong stack
```

```
restore:
lw $s3, 0($sp)
addi $sp,$sp,-4
lw $t4, 0($sp)
addi $sp,$sp,-4
lw $s2, 0($sp)
addi $sp,$sp,-4
lw $s1, 0($sp)
addi $sp,$sp,-4
lw $s0, 0($sp)
addi $sp,$sp,-4
lw $at, 0($sp)
addi $sp,$sp,-4
lw $a0, 0($sp)
addi $sp,$sp,-4
lw $t3, 0($sp)
addi $sp,$sp,-4
lw $t2, 0($sp)
addi $sp,$sp,-4
lw $t1, 0($sp)
addi $sp,$sp,-4
lw $ra, 0($sp)
addi $sp,$sp,-4
return: eret #tra lai ngoai le return Exception
```

II. Tổng quát quá trình

B1: Nhập Mã Điều Khiển vào Digital Lap Sim

B2: Sau đó nhập phím vào Keyboard & Display MMIO Simulator tùy vào phím ta nhập sẽ có những chức năng khác nhau:

- **Enter: Bắt đầu xử lí vào thực thi lệnh**
- **Delete:Xóa hết những mã lệnh đang nhập**
- **Space:Thực thi lại lệnh trước đó**

B3: Sau khi thực thi lệnh Marbot sẽ hoạt động tùy vào mã điều khiển ta nhập trước đó (Nếu lệnh không tồn tại trong đề bài sẽ báo lỗi) và chờ lệnh tiếp theo

III. Chi tiết quá trình:

1. Ý tưởng:

Tạo 1 biến MaDieuKhien để nhận mã lệnh từ Digital Lap Simp. Ta sử dụng Key_Code và Key_Ready để xác nhận và đối chiếu key nhập từ Keyboard & Display MMIO Simulator với '/n', 'del', ' ' để thực hiện chức năng tương ứng. Sau đó sẽ đối chiếu MaDieuKhien với những mã điều khiển có sẵn trong yêu cầu đề bài để thực thi lệnh tương ứng.

2.Chi tiết thuật toán

-Khởi tạo dữ liệu cần thiết cho những Công cụ (Tools)

```
.eqv IN_ADRESS_HEXA_KEYBOARD 0xFFFF0012
.eqv OUT_ADRESS_HEXA_KEYBOARD 0xFFFF0014
.eqv KEY_CODE 0xFFFF0004      # ma ASCII tu ban phim, 1 byte
.eqv KEY_READY 0xFFFF0000     # =1 neu co ma khoa moi
# tu dong xoa sau lw
#-------------------------------------------------------------------------------
# Marsbot
.eqv HEADING 0xffff8010 # so nguyen: tu 0 den 359 do
# 0 : Di len ( Bac )
# 90: Sang phai ( Dong )
# 180: Di xuong ( Nam )
# 270: Sang Trai ( Tay )
.eqv MOVING 0xffff8050 # Boolean cho lenh kich hoat thuc thi
.eqv LEAVETRACK 0xffff8020 # Boolean (0 or non-0):

.eqv WHEREX 0xffff8030 # vi tri x cua MarsBot
.eqv WHEREY 0xffff8040 # vi tri y cua MarsBot


.data
# gia tri nhan vao
.eqv KEY_0 0x11
.eqv KEY_1 0x21
.eqv KEY_2 0x41
.eqv KEY_3 0x81
.eqv KEY_4 0x12
.eqv KEY_5 0x22
.eqv KEY_6 0x42
.eqv KEY_7 0x82
.eqv KEY_8 0x14
.eqv KEY_9 0x24
.eqv KEY_a 0x44
.eqv KEY_b 0x84
.eqv KEY_c 0x18
.eqv KEY_d 0x28
.eqv KEY_e 0x48
.eqv KEY_f 0x88
#-------------------------------------------------------------------------------
#tao cac lenh chuc nang
DiChuyen: .asciiz "1b4"                    # di chuyen
DungLai: .asciiz "c68"              # dung im
ReTrai: .asciiz "444"    # re trai 90 do
RePhai: .asciiz "666"   # re phai 90 do
DauVet: .asciiz "dad"  # ve
BoDauVet: .asciiz "cbc"             # ko ve
QuayXe: .asciiz "999"  # quay tro lai, ko ve. ko nhan ma toi khi het
```

```
msg_output: .asciiz "Da nhap ma dieu khien sai !"
#-------------------------------------------------------------------------------
MaDieuKhien: .space 50 # dau vao ma dieu khien
MaTruocDo: .space 50 # dau vao ma dieu khien
DoDai: .word 0  # ma khiem soat do dai
nowHeading: .word 0
#-------------------------------------------------------------------------------
dauvet: .space 800
DoDaiDauVet: .word 20           #bytes
```

3.Các chương trình con

- Các chương trình chức năng theo mã điều khiển

- **go: Bắt đầu di chuyển**
- **Stop: Ngừng di chuyển**
- **MoveLeft, movRight, Rotate: các hàm dùng cùng với nhau để có thể đổi hướng Marbot sang trái / phải**
- **Track: Để lại dấu vết**
- **Untrack : ngừng để lại dấu vết**
- **GoBack: Di chuyển ngược lại**

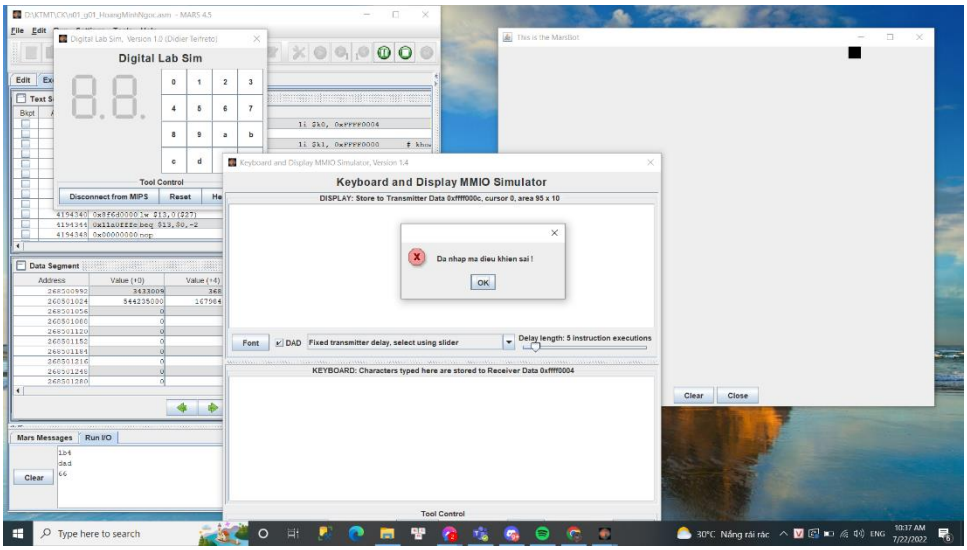
- Các chương trình nhập 1 phím phất kì vào keyboard & display MMIO simulator.

- **CheckMaDieuKhien: Kiểm tra MaDieuKhien mình nhập vào có trùng với Mã điều khiển cho sẵn hay không. Nếu đúng thực hiện chức năng đó**
- **Delete: Xóa toàn bộ nội dung bên trong mã điều khiển bằng cách ghi đè '/0' vào từng byte**
- **LapLai: Sau khi in xong 1 mã lệnh ta sao chép nội dung mã lệnh đó cùng với độ dài vào các biến có sẵn. Khi ta cần dùng phím ' ' sẽ sao chép nội dung trong các biến đó và MaDieuKien va DoDai để thực hiện chức năng như bình thường**

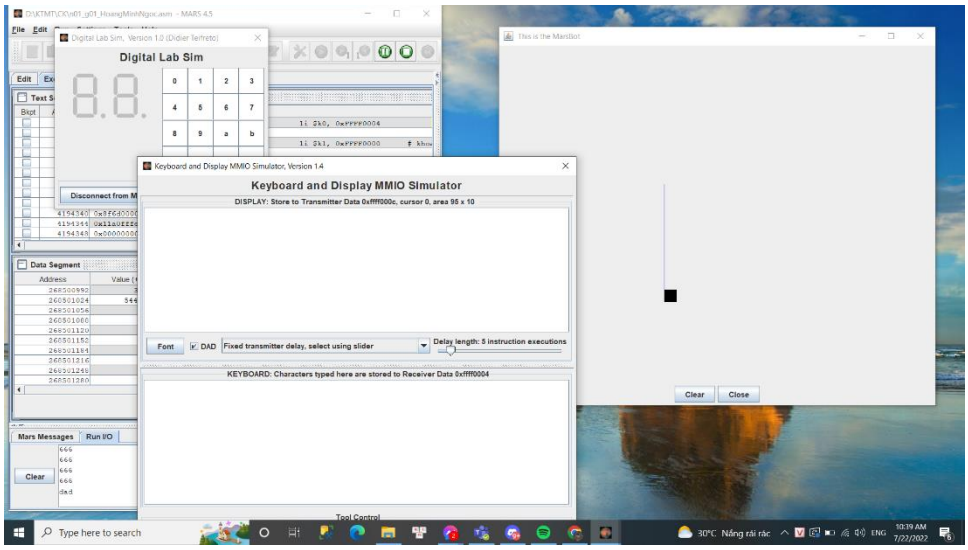
Ngoài ra còn một số chương trình con khác để các chương trình con liệt kê ở trên hoạt động

4. Chạy thử:

1 số hình ảnh minh họa:



Nhập mã lệnh sai cú pháp



Để lại dấu vết