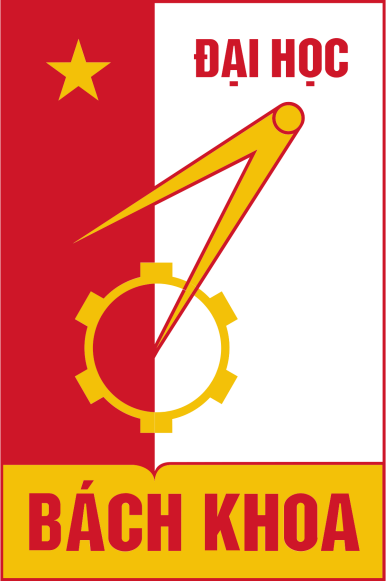
TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

Viện công nghệ Thông tin & Truyền thông



IT3280-130939

Thực hành Kiến trúc máy tính

**BÁO CÁO FINAL-PROJECT**

Giảng Viên Hướng Dẫn: ThS. Lê Bá Vui

Nhóm: 7

Họ và tên Sinh Viên 1: Phạm Quang Đức

MSSV: 20194518

Họ và tên Sinh Viên 2: Nguyễn Tiến Dũng

MSSV: 20194524

Mục lục

**Phần 1:** Project 01………………………….............................

1. Đề bài…………………………………………………
2. Định hướng……………………………………………
3. Ý nghĩa của các hàm…………………………………..
4. Mã nguồn……………………………………………...
5. Kết quả………………………………………………...

**Phần 2:** Project 04…………………………...........................

1. Đề bài…………………………………………………

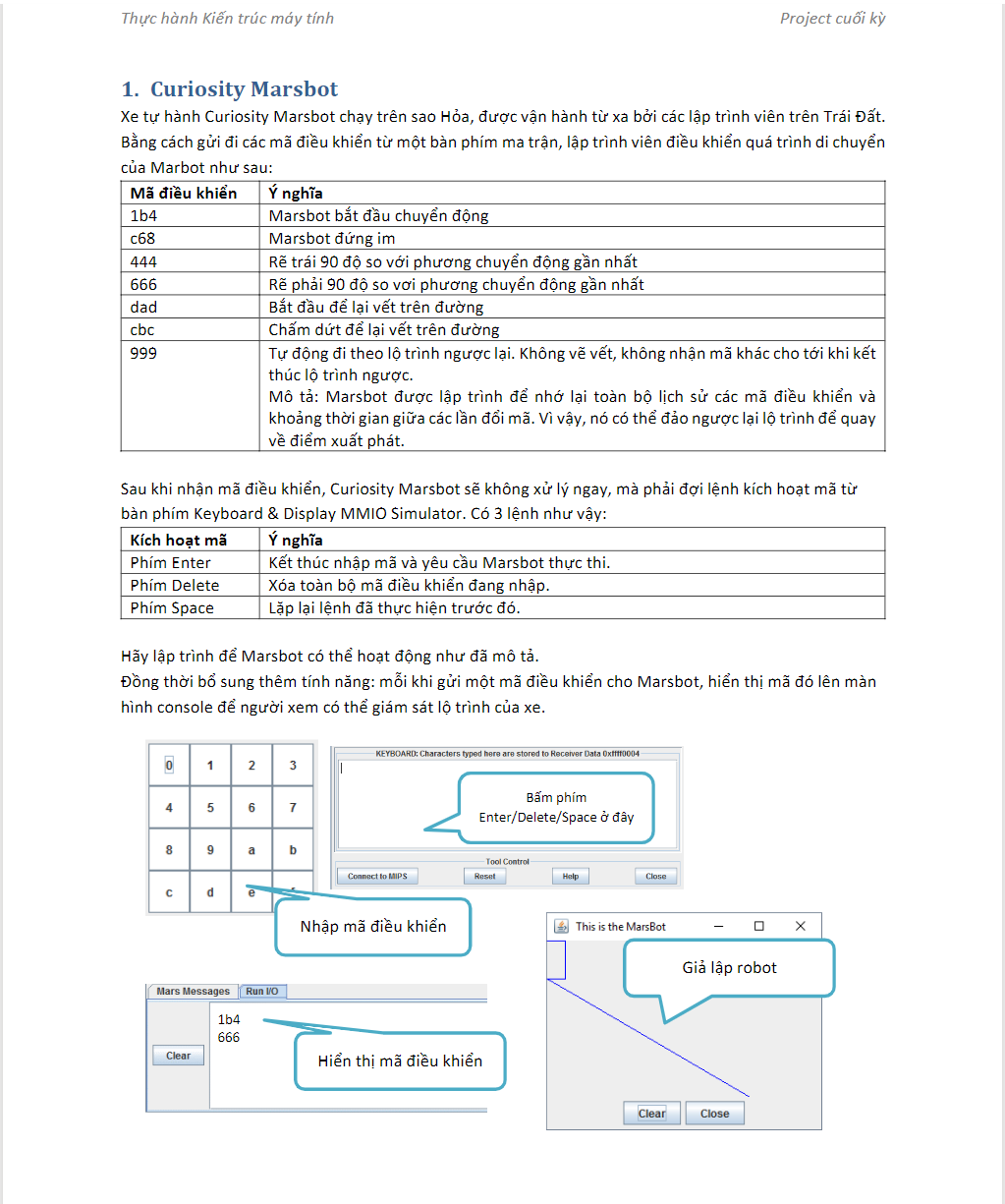
II. Phân tích cách thực hiện ………………………….........

III. Mã nguồn………………………………………………

IV. Kết quả.………………………………………………...

**Phần 1**: Project 1( Phạm Quang Đức)

1. **Đề bài**:

****

1. **Định hướng**

* **Bước 1**: Mỗi khi người dùng nhập 1 ký tự từ **Digital Lab Sim** sẽ tạo ra *interrupt* để lưu ký tự được nhập vào bộ nhớ, tạo lên đoạn code điều khiển.
* **Bước 2**: Kiểm tra xem ký tự **Enter** có được nhập ở **Keyboard & Display MMIO Simulator** hay không? Khi ký tự **Enter** được nhập, sẽ kiểm tra xem đoạn code có được hợp lệ hay không(gồm 3 ký tự), nếu không thì sẽ thông báo lỗi ***Nhap sai code*** và sang **bước 4**. Nếu có thì sang **bước 3**.
* **Bước 3**: Lần lượt kiểm tra xem code điều khiển được nhập vào có trùng với các đoạn code điều khiển đã quy định sẵn. Nếu không thì thông báo đoạn code bị lỗi. Ngược lại thì thực hiện thao tác theo quy định sẵn.
* **Bước 4**: In ra console code điều khiển đã nhập và xoá lưu trữ trong bộ nhớ.

1. **Ý nghĩa các hàm trong mã nguồn**

* **storePath**: Lưu lại thông tin về đường đi của Marsbot vào mảng path. Mảng **path** lưu lại thông tin về đường đi hay đúng hơn là thông tin về các cạnh của đường đi Marsbot. Mỗi cạnh gồm 3 thông tin là: Toạ độ x, y của điểm đầu tiên, hướng đi của cạnh đó.
* **BackCode**: điều khiển Marsbot đi ngược lại lộ trình mà nó đã đi và về điểm xuất phát.

Mảng **path** lưu thông tin đường đi. Mỗi thông tin về một cạnh gồm toạ độ x, y và hướng đi\_3 số nguyên. Do đó mỗi thông tin đường đi sẽ chiếm 12 byte. Do đó **lengthPath** sẽ có giá trị là bội 12.

Mỗi khi muốn quay ngược lại và đi về điểm đầu tiên của một cạnh trên đường đi, ta sẽ lấy hướng đi của cạnh đó và đi ngược lại, đến khi nào gặp điểm có toạ độ như đã lưu thì kết thúc việc đi lại trên cạnh đó, tiếp tục đi cạnh khác.

**GoLeftCode, GoRightCode**: điều khiển Marsbot quay và di chuyển sang phải (với hàm goRight) hoặc trái (với hàm goLeft) một góc 90\*.

Muốn di chuyển sang phải 90\* ta chỉ cần tăng biến **nowHeading** lên 90, đối với bên trái là giảm đi 90\*.

Sau đó thì gọi hàm **ROTATE** để thực hiện

* **ROTATE**: quay Marsbot theo hướng có số độ lưu trong **nowHeading.** Load biến **nowHeading** và lưu vào địa chỉ **HEADING** (0xffff8010) để Marsbot chuyển hướng.
* **DeLaiVet, StopDeLaiVet**: điều khiển Marsbot bắt đầu để lại vết (**DeLaiVet**) hoặc kết thúc để lại vết (**StopDeLaiVet**)

Load 1 vào địa chỉ **LEAVETRACK** (0xffff8020) nếu muốn để lại vết và load 0 nếu muốn kết thúc để lại vết

* **GO**, **STOP**: điều khiển Marsbot bắt đầu chuyển động (GO) hoặc dừng lại (STOP)

Load 1 vào địa chỉ **MOVE** (0xffff8050) nếu muốn di chuyển và load 0 nếu muốn dừng lại

* **errorMess**: hiện thông báo dialog khi người dùng nhập code điều khiển không đúng. Sử dụng các hàm syscall 4,55
* **isControlCode**: kiểm tra code điều khiển mà người dùng nhập vào với 1 code điều khiển nào đó có địa chỉ được đặt trong $s3.

Lần lượt so sánh các ký tự của 2 xâu này. Nếu 2 xâu bằng thanh ghi $t0 có giá trị là 1, ngược lại là 0.

* **removeControlCode**: xoá xâu **inputControlCode** (nơi lưu trữ code điều khiển nhập vào). Lần lượt gán các ký tự trong xâu bằng ‘\0’
* **Các thao tác trong phần xử lý interrupt**: Lần lượt quét các hàng của Digital Lab Sim để xem phím nào được bấm. Tiếp đó dựa vào mã được trả về ghi ký tự tương ứng vào bộ nhớ, cụ thể là ghi vào cuối xâu **inputControlCode**

**IV . Mã Nguồn**

.eqv IN\_ADRESS\_HEXA\_KEYBOARD 0xFFFF0012

.eqv OUT\_ADRESS\_HEXA\_KEYBOARD 0xFFFF0014 #dia chi ngat ket noi digital lab sim

.eqv KEY\_CODE 0xFFFF0004 # Lw keyboard

.eqv KEY\_READY 0xFFFF0000 # cho keyboard

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

# Key value

.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

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

# Marsbot

.eqv HEADING 0xffff8010 # Integer: Goc tu 0 den 359

.eqv MOVING 0xffff8050 # Boolean: di chuyen?

.eqv LEAVETRACK 0xffff8020 # de lai vet

.eqv WHEREX 0xffff8030 #Vi tri x hien tai cua MarsBot

.eqv WHEREY 0xffff8040 #Vi tri y hien tai cua MarsBot

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

.data

#Control code

MoveCode: .asciiz "1b4"

StopCode: .asciiz "c68"

GoLeftCode: .asciiz "444"

GoRightCode: .asciiz "666"

DeLaiVet: .asciiz "dad"

StopDeLaiVet: .asciiz "cbc"

BackCode: .asciiz "999"

WRONG\_CODE: .asciiz "Nhap sai code!!!"

inputControlCode: .space 10

lengthControlCode: .word 0

nowHeading: .word 0 # vi tri hien tai cua boot

path: .space 600 # luu trang thai khi thay doi cau lenh

lengthPath: .word 12 # do lon byte luu 3 gia tri x,y,z

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

.text

main: #khai bao

li $k0, KEY\_CODE

li $k1, KEY\_READY

li $t1, IN\_ADRESS\_HEXA\_KEYBOARD # Cho phep ngat ban phim cua Digital Lab Sim

li $t3, 0x80

sb $t3, 0($t1)

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

loop: nop

WAITKEY: lw $t5, 0($k1) # $t5 = [$k1] = KEY\_READY

beq $t5, $zero, WAITKEY # Neu $t5 == 0 thi lap lai WAITKEY, kiem tra key moi

nop

beq $t5, $zero, WAITKEY

nop

READ: lw $t6, 0($k0) # $t6 = [$k0] = KEY\_CODE

beq $t6, 127, delete # Neu $t6 == delete key thi xoa input, 127 la delete key trong ma ascii

nop

bne $t6, '\n', loop # Neu $t6 != '\n' thi lap lai loop, dung de doc enter khi nhap vao

nop

CHECKCONTROL:

la $s2, lengthControlCode

lw $s2, 0($s2)

bne $s2, 3, errorMessage

nop

la $s3, MoveCode

jal TEST\_CONTROL

nop

beq $t0, 1, go

nop

la $s3, StopCode

jal TEST\_CONTROL

nop

beq $t0, 1, stop

nop

la $s3, GoLeftCode

jal TEST\_CONTROL

nop

beq $t0, 1, goLeft

nop

la $s3, GoRightCode

jal TEST\_CONTROL

nop

beq $t0, 1, goRight

nop

la $s3, DeLaiVet

jal TEST\_CONTROL

nop

beq $t0, 1, track

nop

la $s3, StopDeLaiVet

jal TEST\_CONTROL

nop

beq $t0, 1, untrack

nop

la $s3, BackCode

jal TEST\_CONTROL

nop

beq $t0, 1, goBack

nop

beq $t0, 0, errorMessage

nop

PRINT\_CONTROL:

li $v0, 4

la $a0, inputControlCode

syscall

nop

delete:

jal REMOVE\_CONTROL

nop

j loop

nop

j loop

TEST\_CONTROL:# so sanh keyboard voi $s3( == -> $t0=1; != ->4st0=0)

#backup

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)

# processing

addi $t1, $zero, -1 # $t1 = -1 = i

add $t0, $zero, $zero

la $s1, inputControlCode # $s1 = inputControlCode

loopToCheckString:

addi $t1, $t1, 1 # i++

add $t2, $s1, $t1 # $t2 = inputControlCode + i

lb $t2, 0($t2) # $t2 = inputControlCode[i]

add $t3, $s3, $t1 # $t3 = s + i

lb $t3, 0($t3) # $t3 = s[i]

bne $t2, $t3, isNotControlCode # Neu $t2 != $t3 nhay den isNotControlCode

bne $t1, 2, loopToCheckString # Neu $t1 <= 2 tiep tuc vong lap

nop

bne $t1, 2, loopToCheckString

lw $t3, 0($sp) # restore

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 = 1

jr $ra

nop

jr $ra

isNotControlCode:

lw $t3, 0($sp) # restore

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 = 0

jr $ra

nop

jr $ra

errorMessage: #Hien thong bao dialog khi nhap code sai

li $v0, 4

la $a0, inputControlCode

syscall

nop

li $v0, 55

la $a0, WRONG\_CODE

syscall

nop

nop

j delete

nop

j delete

REMOVE\_CONTROL: #Xoa xau inputControlCode, lan luot gan cac ky tu cua inputControlCode = '\0'

#backup

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)

#processing

la $s2, lengthControlCode

lw $t3, 0($s2) #$t3 = lengthControlCode

addi $t1, $zero, -1 #$t1 = -1 = i

addi $t2, $zero, 0 #$t2 = '\0'

la $s1, inputControlCode

addi $s1, $s1, -1

forLoopToRemove:

addi $t1, $t1, 1 #i++

add $s1, $s1, 1 #$s1 = inputControlCode + i

sb $t2, 0($s1) #inputControlCode[i] = '\0'

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

nop

bne $t1, $t3, forLoopToRemove

add $t3, $zero, $zero

sw $t3, 0($s2) #lengthControlCode = 0

#restore

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

storePath: # luu lai thong tin ve duong di cua Marsbot vao mang path

#backup

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)

#processing

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, lengthPath

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

la $t4, path

add $t4, $t4, $s3 # Vi tri bat dau luu

sw $s1, 0($t4) # Luu x

sw $s2, 4($t4) # Luu y

sw $s4, 8($t4) # Luu heading

addi $s3, $s3, 12 # Cap nhat lengthPath

# 12 = 3 (word) x 4 (bytes)

sw $s3, 0($t3)

#restore

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

track: jal TRACK #track: Dieu khien Marsbot va in control conde

j PRINT\_CONTROL

untrack: jal UNTRACK #untrack: Dieu khien Marsbot va in control conde

j PRINT\_CONTROL

go:

jal GO

j PRINT\_CONTROL

stop: jal STOP

j PRINT\_CONTROL

goRight:

#backup

addi $sp,$sp,4

sw $s5, 0($sp)

addi $sp,$sp,4

sw $s6, 0($sp)

#restore

la $s5, nowHeading

lw $s6, 0($s5)

addi $s6, $s6, 90

sw $s6, 0($s5) # Cap nhat nowHeading = nowHeading + 90

#restore

lw $s6, 0($sp)

addi $sp,$sp,-4

lw $s5, 0($sp)

addi $sp,$sp,-4

jal storePath

jal ROTATE

j PRINT\_CONTROL

goLeft: #Dieu khien Marsbot quay va di chuyen sang trai mot goc 90

#backup

addi $sp,$sp,4

sw $s5, 0($sp)

addi $sp,$sp,4

sw $s6, 0($sp)

#processing

la $s5, nowHeading

lw $s6, 0($s5)

addi $s6, $s6, -90

sw $s6, 0($s5) # Cap nhat nowHeading = nowHeading - 90

#restore

lw $s6, 0($sp)

addi $sp,$sp,-4

lw $s5, 0($sp)

addi $sp,$sp,-4

jal storePath

jal ROTATE

j PRINT\_CONTROL

goBack: #goBack: Dieu khien Marsbot di nguoc lai theo lo trinh da di va ve lai diem xuat phat

#backup

addi $sp,$sp,4

sw $s5, 0($sp)

addi $sp,$sp,4

sw $s6, 0($sp)

addi $sp,$sp,4

sw $s7, 0($sp)

addi $sp,$sp,4

sw $t8, 0($sp)

addi $sp,$sp,4

sw $t9, 0($sp)

jal UNTRACK

jal GO

la $s7, path

la $s5, lengthPath

lw $s5, 0($s5)

add $s7, $s7, $s5

begin:

addi $s5, $s5, -12 # Lui lai 1 structure

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

goToFirstPointOfEdge:

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, goToFirstPointOfEdge

nop

bne $t8, $t9, goToFirstPointOfEdge

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, goToFirstPointOfEdge

nop

bne $t8, $t9, goToFirstPointOfEdge

beq $s5, 0, finish

nop

beq $s5, 0, finish

j begin

nop

j begin

finish:

jal STOP

la $t8, nowHeading

add $s6, $zero, $zero

sw $s6, 0($t8) # Cap nhat heading

la $t8, lengthPath

addi $s5, $zero, 12

sw $s5, 0($t8) # Cap nhat lengthPath = 12

lw $t9, 0($sp) #restore

addi $sp,$sp,-4

lw $t8, 0($sp)

addi $sp,$sp,-4

lw $s7, 0($sp)

addi $sp,$sp,-4

lw $s6, 0($sp)

addi $sp,$sp,-4

lw $s5, 0($sp)

addi $sp,$sp,-4

jal ROTATE

j PRINT\_CONTROL

GO: #GO: Dieu khien Marsbot chuyen dong

addi $sp,$sp,4 # backup

sw $at,0($sp)

addi $sp,$sp,4

sw $k0,0($sp)

# processing

li $at, MOVING # Thay doi MOVING port

addi $k0, $zero,1 # logic 1

sb $k0, 0($at) # Bat dau chuyen dong

lw $k0, 0($sp) # restore

addi $sp,$sp,-4

lw $at, 0($sp)

addi $sp,$sp,-4

jr $ra

nop

jr $ra

STOP: #STOP: Dieu khien Marsbot dung lai

addi $sp,$sp,4 # backup

sw $at,0($sp)

# processing

li $at, MOVING # Thay doi MOVING port thanh 0

sb $zero, 0($at) # Dung lai

lw $at, 0($sp) # restore

addi $sp,$sp,-4

jr $ra

nop

jr $ra

TRACK: #TRACK: Dieu khien Marsbot bat dau de lai vet

addi $sp,$sp,4 # backup

sw $at,0($sp)

addi $sp,$sp,4

sw $k0,0($sp)

# processing

li $at, LEAVETRACK # Thay doi LEAVETRACK port

addi $k0, $zero,1 # logic 1

sb $k0, 0($at) # Bat dau ve

lw $k0, 0($sp) # restore

addi $sp,$sp,-4

lw $at, 0($sp)

addi $sp,$sp,-4

jr $ra

nop

jr $ra

UNTRACK: #UNTRACK: Dieu khien Marsbot ket thuc de lai vet

#backup

addi $sp,$sp,4

sw $at,0($sp)

#processing

li $at, LEAVETRACK # Thay doi LEAVETRACK port thanh 0

sb $zero, 0($at) # Dung ve

#restore

lw $at, 0($sp)

addi $sp,$sp,-4

jr $ra

nop

jr $ra

ROTATE: # ROTATE: Quay Marsbot theo huong co so do luu trong nowHeading

addi $sp,$sp,4 #backup

sw $t1,0($sp)

addi $sp,$sp,4

sw $t2,0($sp)

addi $sp,$sp,4

sw $t3,0($sp)

#processing

li $t1, HEADING # Thay doi HEADING port

la $t2, nowHeading

lw $t3, 0($t2) # $t3 la heading hien tai

sw $t3, 0($t1) # Xoay bot

lw $t3, 0($sp) #restore

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 for all interrupts

.ktext 0x80000180

# SAVE the current REG FILE to stack

backup:

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)

# Processing

getCode:

li $t1, IN\_ADRESS\_HEXA\_KEYBOARD

li $t2, OUT\_ADRESS\_HEXA\_KEYBOARD

scanRow1:

li $t3, 0x81

sb $t3, 0($t1)

lbu $a0, 0($t2)

bnez $a0, getCodeInChar

scanRow2:

li $t3, 0x82

sb $t3, 0($t1)

lbu $a0, 0($t2)

bnez $a0, getCodeInChar

scanRow3:

li $t3, 0x84

sb $t3, 0($t1)

lbu $a0, 0($t2)

bnez $a0, getCodeInChar

scanRow4:

li $t3, 0x88

sb $t3, 0($t1)

lbu $a0, 0($t2)

bnez $a0, getCodeInChar

getCodeInChar:

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

# $s0 luu tru code kieu ky tu

case\_0: li $s0, '0'

j storeCode

case\_1: li $s0, '1'

j storeCode

case\_2: li $s0, '2'

j storeCode

case\_3: li $s0, '3'

j storeCode

case\_4: li $s0, '4'

j storeCode

case\_5: li $s0, '5'

j storeCode

case\_6: li $s0, '6'

j storeCode

case\_7: li $s0, '7'

j storeCode

case\_8: li $s0, '8'

j storeCode

case\_9: li $s0, '9'

j storeCode

case\_a: li $s0, 'a'

j storeCode

case\_b: li $s0, 'b'

j storeCode

case\_c: li $s0, 'c'

j storeCode

case\_d: li $s0, 'd'

j storeCode

case\_e: li $s0, 'e'

j storeCode

case\_f: li $s0, 'f'

j storeCode

storeCode:

la $s1, inputControlCode

la $s2, lengthControlCode

lw $s3, 0($s2) # $s3 = strlen(inputControlCode)

addi $t4, $t4, -1 # $t4 = i

loopToStoreCode:

addi $t4, $t4, 1 # i++

bne $t4, $s3, loopToStoreCode # neu $t4 != $s3 thi nhay lai tu dau

add $s1, $s1, $t4 # $s1 = inputControlCode + i

sb $s0, 0($s1) # inputControlCode[i] = $s0

addi $s0, $zero, '\n' # Them '\n' vao cuoi xau

addi $s1, $s1, 1

sb $s0, 0($s1)

addi $s3, $s3, 1

sw $s3, 0($s2) # cap nhat lai do dai cua inputControlCode

next\_pc: #Evaluate the return address of main routine, epc <= epc + 4

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

restore: # RESTORE the REG FILE from STACK

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 # Return from exception

**V .Kết quả**

Graphical user interface, application, Word

Description automatically generated

Điều khiển Marsbot bằng các câu lệnh

Graphical user interface

Description automatically generated

In ra dialog thông báo control code nhập vào bị lỗi

A screenshot of a computer

Description automatically generated with medium confidence

Graphical user interface, Word

Description automatically generated

Điều khiển Marsbot đi ngược lại trở về vị trí ban đầu

**Phần 2:** Project 04

1. **Đề bài**

**Graphical user interface

Description automatically generated with low confidence**

**II . Cách thực hiện:**

* Đọc từ bàn phím Key Matrix để để chọn postscript nào sẽ được gia công với hàm SELECT, nếu người dùng chọn phím 0 chương trình sẽ thực hiện postscript1, người dùng chọn phím 4 chương trình sẽ thực hiện postscript2, người dùng chọn phím 8 chương trình sẽ thực hiện postscript3, nếu chọn phím khác chương trình sẽ không thực hiện.
* Sau khi đã đọc được yêu cầu chạy postscript cụ thể sẽ chạy hàm READ\_PSCRIPT để đọc nội dung của postscript đó
* Trong hàm READ\_PSCRIPT có các hàm READ\_ROTATE để đọc giá trị rotate (quay), READ\_TIME để đọc giá trị thời gian Marsbot chạy, READ\_TRACK để đọc giá trị track (giá trị 1 - Marsbot ghi) hoặc untrack (giá trị 0 – Marsbot không ghi)
* Sau khi đã đọc được giá trị thì sẽ gọi các hàm để Marsbot thực hiện.

**III . Code:**

# Mars bot

.eqv HEADING 0xffff8010

.eqv LEAVETRACK 0xffff8020

.eqv WHEREX 0xffff8030

.eqv WHEREY 0xffff8040

.eqv MOVING 0xffff8050

# Key matrix

.eqv OUT\_ADRESS\_HEXA\_KEYBOARD 0xFFFF0014

.eqv IN\_ADRESS\_HEXA\_KEYBOARD 0xFFFF0012

.data

# postscript-DCE -> enter 0

# (motion angle, 0=untrack | 1=track,;)

pscript1: .asciiz "90,0,1000,180,0,3000,180,1,6000,80,1,2000,40,1,2000,0,1,2000,320,1,2000,280,1,2000,90,0,8000,270,1,2000,210,1,2000,180,1,2000,150,1,2000,90,1,2000,90,0,5000,270,1,3000,0,1,5800,90,1,3000,180,0,3000,270,1,3000,"

# postscript- Dung -> enter 4

pscript2: .asciiz "90,0,1000,180,0,3000,180,1,6000,80,1,2000,40,1,2000,0,1,2000,320,1,2000,280,1,2000,90,0,5000,180,1,5700,90,1,3000,0,1,5700,90,0,2000,180,1,5700,0,0,5700,135,1,8000,0,1,5700,90,0,5000,270,1,2000,210,1,2000,180,1,2000,150,1,2000,90,1,2000,0,1,2000,270,0,500,90,1,1000,"

# postscript- Duc -> enter 8

pscript3: .asciiz "90,0,1000,180,0,3000,180,1,6000,80,1,2000,40,1,2000,0,1,2000,320,1,2000,280,1,2000,90,0,5000,180,1,5700,90,1,3000,0,1,5700,90,0,5000,270,1,2000,210,1,2000,180,1,2000,150,1,2000,90,1,2000,"

.text

# Key Matrix

li $s1, IN\_ADRESS\_HEXA\_KEYBOARD

li $s2, OUT\_ADRESS\_HEXA\_KEYBOARD

SELECT:

li $t4, 0x01 # dong 1 cua key matrix

sb $t4, 0($s1)

lb $a0, 0($s2)

bne $a0, 0x11, NOT\_0

la $a1, pscript1

j START

NOT\_0:

li $t4, 0x02 # dong 2 cua key matrix

sb $t4, 0($s1)

lb $a0, 0($s2)

bne $a0, 0x12, NOT\_4

la $a1, pscript2

j START

NOT\_4:

li $t4, 0x04 # dong 3 cua key matrix

sb $t4, 0($s1)

lb $a0, 0($s2)

bne $a0, 0x14, BACK

la $a1, pscript3

j START

BACK: j SELECT # Vong lap cho den khi chon dung 1 trong 3 so 0, 4, 8

# end

# xu li mars bot

START:

jal GO

READ\_PSCRIPT:

addi $t1, $zero, 0 # luu gia tri goc chuyen dong

addi $t2, $zero, 0 # luu gia tri thoi gian

addi $t3, $zero, 0 # luu gia tri track

  READ\_ROTATE:

  add $t7, $a1, $t6

lb $t4, 0($t7)  # doc 1 ki tu cua pscript

beq $t4, 0, END # ket thuc pscript

  beq $t4, 44, READ\_ROTATE1 # gap ki tu ','

  mul $t1, $t1, 10

  addi $t4, $t4, -48 # So 0-9 co thu tu 48-57 trong bang ascii.

  add $t1, $t1, $t4 # cong cac chu so lai voi nhau-> goc can quay

  addi $t6, $t6, 1 # tang vi tri ki tu can doc them 1

  j READ\_ROTATE # quay lai doc tiep den khi gap dau ','

  READ\_ROTATE1:

  add $a0, $t1, $zero

jal ROTATE

  READ\_TRACK: # doc xem co track khong

  addi $t6, $t6, 1

  add $t7, $a1, $t6

lb $t4, 0($t7)

  addi $t4, $t4, -48

  add $t3, $zero, $t4

  addi $t6, $t6, 1

  READ\_TIME: # doc thoi gian chuyen dong.

  addi $t6, $t6, 1

  add $t7, $a1, $t6

lb $t4, 0($t7)

beq $t4, 44, RUN\_TRACK

mul $t2, $t2, 10

  addi $t4, $t4, -48

  add $t2, $t2, $t4

  j READ\_TIME # quay lai doc tiep den khi gap dau ','

  RUN\_TRACK:

  addi $v0,$zero,32 # thoi gian marbot track = $t2

  add $a0, $zero, $t2

  beq $t3, $zero, CHECK\_UNTRACK # 1=track | 0=untrack

  jal UNTRACK

jal TRACK

j INCREAMENT

CHECK\_UNTRACK:

jal UNTRACK

INCREAMENT:

syscall

  addi $t6, $t6, 1 # bo qua dau ','

  j READ\_PSCRIPT

GO:

  li $at, MOVING

  addi $k0, $zero,1

  sb $k0, 0($at)

  jr $ra

STOP:

li $at, MOVING

  sb $zero, 0($at)

  jr $ra

TRACK:

li $at, LEAVETRACK

  addi $k0, $zero,1

sb $k0, 0($at)

  jr $ra

UNTRACK:

li $at, LEAVETRACK

  sb $zero, 0($at)

  jr $ra

ROTATE:

li $at, HEADING

  sw $a0, 0($at)

  jr $ra

END:

jal STOP

li $v0, 10

syscall

j SELECT

# end

**IV . Kết Quả**

**Chart, box and whisker chart

Description automatically generated**

**Chart, line chart

Description automatically generated**

