**TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI**

Viện Công nghệ thông tin và Truyền thông

\*\*\*\*\*\*\*∞\*\*\*\*\*\*\*

A red and yellow sign

Description automatically generated with low confidence

**BÁO CÁO BÀI TẬP CUỐI KÌ 20212**

*Môn Học:* **Thực Hành Kiến Trúc Máy Tính**

*Mã HP:*  IT3280

*Mã Lớp:* 130937

*Giảng viên hướng dẫn:* **ThS. Lê Bá Vui**

*Sinh viên thực hiện:* **Kiều Phương Thảo – 20205128 (**Đề 1**)**

**Ngô Thị Hồng Hạnh – 20204964 (**Đề 5**)**

# ***ĐỀ 1***

## **Đề bài : Curiosity Marsbot**

Table

Description automatically generated

## **Phân tích cách làm**

* Cần có 3 công cụ mô phỏng hỗ trợ:

+ Digital Lab Sim (để nhập key control)

+ Keyboard and Display MMIO (nhập Delete/ Enter/ Space)

+ Mars Bot

* Sử lý từng công cụ:

+ Với Digital Lab Sim cần input control code

+ Với MMIO cần kiểm tra khi nào người dùng nhập vào cáo phím Delete/Enter/Space để tiến hành check mã code vừa mới input từ Digital Lab Sim

+ Sau khi nhận lệnh điều khiển thì tiến hành cho Marsbot thực thi

## **Thuật toán**

* Bước 1: Mỗi khi người dùng nhập 1 kí tự từ Digital Lab Sim sẽ tạo ra interrupt để lưu kí tự được nhập vào bộ nhớ, tạo nên đoạn code điều khiển
* Bước 2: Kiểm tra liên tục xem kí tự được nhập ở Keyboard & Display MMIO Simulator.

+ Nếu kí tự được nhập là phím Delete thì chương trình sẽ nhảy đến notprint (không print control, đồng thời xóa mã code người dùng đang nhập)

+ Nếu kí tự là phím Space thì chương trình sẽ nhảy đến recontrol (thực hiện lại câu lệnh trước đó)

+ Nếu kí tự Enter được nhập thì kiểm tra xem mã code vừa nhập có đủ 3 ký tự hay không. Nếu có thì chuyển sang bước 3, nếu không thì chuyển sang bước 4.

* 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 chuyển sang bước 4. Nếu có thì thực thi marsbot.
* Bước 4:

+Mã code đúng: In ra control code điều khiển đã nhập và xóa lưu trữ trong bộ nhớ.

+Mã code sai: In ra control code, và thông báo “Wrong! Again”, đồng thời xóa lưu trữ trong bộ nhớ

## **Mã nguồn**

## Mã nguồn đầy đủ

.eqv IN\_ADRESS\_HEXA\_KEYBOARD 0xFFFF0012

.eqv OUT\_ADRESS\_HEXA\_KEYBOARD 0xFFFF0014

.eqv KEY\_CODE 0xFFFF0004

.eqv KEY\_READY 0xFFFF0000

#Key of Digital Lab Sim

.eqv input\_0 0x11

.eqv input\_1 0x21

.eqv input\_2 0x41

.eqv input\_3 0x81

.eqv input\_4 0x12

.eqv input\_5 0x22

.eqv input\_6 0x42

.eqv input\_7 0x82

.eqv input\_8 0x14

.eqv input\_9 0x24

.eqv input\_a 0x44

.eqv input\_b 0x84

.eqv input\_c 0x18

.eqv input\_d 0x28

.eqv input\_e 0x48

.eqv input\_f 0x88

# Marsbot

.eqv HEADING 0xffff8010

.eqv MOVING 0xffff8050

.eqv LEAVETRACK 0xffff8020

.eqv WHEREX 0xffff8030 #toa do x cua marsbot

.eqv WHEREY 0xffff8040 #toa do y cua marsbot

#===============================================================================

.data

#Control code mac dinh

MOVE\_CODE: .asciiz "1b4"

STOP\_CODE: .asciiz "c68"

LEFT\_CODE: .asciiz "444"

RIGHT\_CODE: .asciiz "666"

TRACK\_CODE: .asciiz "dad"

UNTRACK\_CODE: .asciiz "cbc"

BACK\_CODE: .asciiz "999"

WRONG\_CODE: .asciiz "Wrong! Again"

#Khai bao cac bien de luu tru code nhap vao,bo nho dem truoc khi xoa de sau nay space, do dai code, vitri cua marsbot

INPUT: .space 50

SAVE: .space 50

LENGTH: .word 0

SAVE\_LENGTH: .word 0

NOW\_HEAD: .word 0

PATH: .space 600

LENGTH\_PATH: .word 12

.text

main:

li $k0, KEY\_CODE

li $k1, KEY\_READY

li $t1, IN\_ADRESS\_HEXA\_KEYBOARD

li $t2, 0x80 # bit 7 = 1 to enable

sb $t2, 0($t1)

#-----------------------------------------------------------KeyBoard and Display MMIO--------------------------------------------

loop: nop

WaitForKey:# nhap vao cac phim Enter/ Delete/ Space

lw $t8, 0($k1) #$t8 = [$k1] = KEY\_READY

beq $t8, $zero, WaitForKey #if $t8 == 0 then Polling

nop

beq $t8, $zero, WaitForKey

lw $t8, 0($sp)

addi $sp,$sp,-4

ReadKey:# Doc ky tu vua nhap tu Keyboard

lw $t9, 0($k0) #$t9 = [$k0] = KEY\_CODE

beq $t9, 127 , notPrint

nop

beq $t9, 127 , notPrint #if $t9 == delete key then remove input

nop

beq $t9,32,reControl #if $t9 == space key then reControl again

bne $t9, '\n' , loop #if $t9 != '\n' then Polling

nop

bne $t9, '\n' , loop

Check:# Kiem tra xem ma code vua nhap tu Digital Lab Sim co dung khong

la $s2, LENGTH #neu khong phai 3 ky tu thi printError

lw $s2, 0($s2)

bne $s2, 3, printError

#neu dúng 3 ky tu thi check xem input la ma dieu khien nao

la $s3, MOVE\_CODE

jal soSanh2chuoi

beq $t0, 1, go

la $s3, STOP\_CODE

jal soSanh2chuoi

beq $t0, 1, stop

la $s3, LEFT\_CODE

jal soSanh2chuoi

beq $t0, 1, goLeft

la $s3, RIGHT\_CODE

jal soSanh2chuoi

beq $t0, 1, goRight

la $s3, TRACK\_CODE

jal soSanh2chuoi

beq $t0, 1, track

la $s3, UNTRACK\_CODE

jal soSanh2chuoi

beq $t0, 1, untrack

la $s3, BACK\_CODE

jal soSanh2chuoi

beq $t0, 1, goBack

beq $t0, 0, printError

print:# in ra code vua nhap dung len man hinh

li $v0, 4

la $a0, INPUT

syscall

nop

saveCode:# luu tru lai code dung truoc khi xoa de reControl neu nguoi dung an Saoce

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)

la $s1,INPUT

la $s2, LENGTH

la $s3, SAVE

la $s4, SAVE\_LENGTH

lw $t4,0($s2)

sw $t4, 0($s4)

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

add $t0, $zero, $zero

for\_loop\_to\_save:

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

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

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

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

sb $t2, 0($t3) #$t3 = s[i]

bne $t1, 3, for\_loop\_to\_save #if $t1 <=2 continue loop

nop

bne $t1, 3, for\_loop\_to\_save

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

notPrint:# khong in va chi xoa

jal removeControl

nop

j loop

nop

j loop

printError:#print code bi nhap sai

li $v0, 4

la $a0, INPUT

syscall

nop

li $v0, 55

la $a0, WRONG\_CODE

syscall

nop

nop

j notPrint

nop

j notPrint

removeControl: #xoa code cu de nhap code moi

#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, LENGTH

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

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

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

la $s1, INPUT

addi $s1, $s1, -1

for\_loop\_to\_remove:

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

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

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

bne $t1, $t3, for\_loop\_to\_remove #if $t1 <=3 continue loop

nop

bne $t1, $t3, for\_loop\_to\_remove

add $t3, $zero, $zero

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

reControl:#khi nguoi dung nhap Space se in ra cau lenh truoc do

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)

la $s1,INPUT

la $s2, LENGTH

la $s3, SAVE

la $s4, SAVE\_LENGTH

lw $t4,0($s4)

sw $t4, 0($s2)

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

add $t0, $zero, $zero

for\_to\_save:

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

add $t2, $s3, $t1 #$t2 = INPUT + i

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

add $t3, $s1, $t1 #$t3 = s + i

sb $t2, 0($t3) #$t3 = s[i]

bne $t1, 3, for\_to\_save #if $t1 <=2 continue loop

nop

bne $t1, 3, for\_to\_save

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

nop

j Check

nop

j Check

soSanh2chuoi:#so sanh 2 chuoi xem co bang nhau khong

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)

#chuong trinh con

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

add $t0, $zero, $zero

la $s1, INPUT

for\_loop\_to\_check\_equal:

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

add $t2, $s1, $t1

lb $t2, 0($t2)

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

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

bne $t2, $t3, haiChuoiKhongBangNhau #if $t2 != $t3 -> khong bang nhau

bne $t1, 2, for\_loop\_to\_check\_equal #if $t1 <=2 continue loop

nop

bne $t1, 2, for\_loop\_to\_check\_equal

haiChuoiBangNhau:

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 #update $t0

jr $ra

nop

jr $ra

haiChuoiKhongBangNhau:

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 #update $t0

jr $ra

nop

jr $ra

#-------------------------------------------------------------------Mar Bot ------------------------------------------------

inputPath:# nhap vao mang path toa do x,y, huong xoay

#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, NOW\_HEAD

lw $s4, 0($s4) #s4 = NOW\_HEAD

la $t3, LENGTH\_PATH

lw $s3, 0($t3) #$s3 = LENGTH\_PATH (dv: byte)

la $t4, PATH

add $t4, $t4, $s3 #position to store

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

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

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

addi $s3, $s3, 12 #update LENGTH\_PATH

#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

goBack:

#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 $t6, 0($sp)

addi $sp,$sp,4

sw $t5, 0($sp)

jal UNTRACK

jal GO

la $s7, PATH

la $s5, LENGTH\_PATH

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 $t6, NOW\_HEAD #marsbot quay nguoc lai

sw $s6, 0($t6)

jal ROTATE

go\_to\_first\_point\_of\_edge:

lw $t5, 0($s7) #toa do x cua diem dau tien cua canh

li $t6, WHEREX #toa do x hien tai

lw $t6, 0($t6)

bne $t6, $t5, go\_to\_first\_point\_of\_edge

nop

bne $t6, $t5, go\_to\_first\_point\_of\_edge

lw $t5, 4($s7) #toa do y cua diem dau tien cua canh

li $t6, WHEREY #toa do y hien tai

lw $t6, 0($t6)

bne $t6, $t5, go\_to\_first\_point\_of\_edge

nop

bne $t6, $t5, go\_to\_first\_point\_of\_edge

beq $s5, 0, finish

nop

beq $s5, 0, finish

j begin

nop

j begin

finish:

jal STOP

la $t6,NOW\_HEAD

add $s6, $zero, $zero

sw $s6, 0($t6) #update heading

la $t6, LENGTH\_PATH

addi $s5, $zero, 12

sw $s5, 0($t6) #update LENGTH\_PATH = 12

#restore

lw $t5, 0($sp)

addi $sp,$sp,-4

lw $t6, 0($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

j print

track: jal UNTRACK

jal TRACK

j print

untrack: jal UNTRACK

j print

go:

jal GO

j print

stop: jal STOP

j print

goRight:

#backup

addi $sp,$sp,4

sw $s5, 0($sp)

addi $sp,$sp,4

sw $s6, 0($sp)

addi $sp,$sp,4

sw $at,0($sp)

addi $sp,$sp,4

sw $t0,0($sp)

#chuong trinh con

li $at, LEAVETRACK

lb $t0, 0($at)

beq $t0, $zero, countinue\_right

jal UNTRACK

jal TRACK

countinue\_right:

la $s5, NOW\_HEAD

lw $s6, 0($s5) #$s6 is heading at now

addi $s6, $s6, 90 #increase heading by 90\*

sw $s6, 0($s5) # update nowHeading

#

lw $t0,0($sp)

addi $sp,$sp,4

lw $at, 0($sp)

addi $sp,$sp,-4

lw $s6, 0($sp)

addi $sp,$sp,-4

lw $s5, 0($sp)

addi $sp,$sp,-4

jal inputPath

jal ROTATE

j print

goLeft:

addi $sp,$sp,4

sw $s5, 0($sp)

addi $sp,$sp,4

sw $s6, 0($sp)

addi $sp,$sp,4

sw $at,0($sp)

addi $sp,$sp,4

sw $t0,0($sp)

#chuong trinh con

li $at, LEAVETRACK

lb $t0, 0($at)

beq $t0, $zero, countinue\_left

jal UNTRACK

jal TRACK

countinue\_left:

la $s5, NOW\_HEAD

lw $s6, 0($s5) #$s6 is heading at now

addi $s6, $s6, -90 #increase heading by 90\*

sw $s6, 0($s5) # update NOW\_HEAD

#

lw $t0,0($sp)

addi $sp,$sp,4

lw $at, 0($sp)

addi $sp,$sp,-4

lw $s6, 0($sp)

addi $sp,$sp,-4

lw $s5, 0($sp)

addi $sp,$sp,-4

jal inputPath

jal ROTATE

j print

GO: #backup

addi $sp,$sp,4

sw $at,0($sp)

addi $sp,$sp,4

sw $k0,0($sp)

#processing

li $at, MOVING # change MOVING port

addi $k0, $zero,1 # to logic 1,

sb $k0, 0($at) # to start running

#restore

lw $k0, 0($sp)

addi $sp,$sp,-4

lw $at, 0($sp)

addi $sp,$sp,-4

jr $ra

nop

jr $ra

STOP: #backup

addi $sp,$sp,4

sw $at,0($sp)

#processing

li $at, MOVING # change MOVING port to 0

sb $zero, 0($at) # to stop

#restore

lw $at, 0($sp)

addi $sp,$sp,-4

jr $ra

nop

jr $ra

TRACK: #backup

addi $sp,$sp,4

sw $at,0($sp)

addi $sp,$sp,4

sw $k0,0($sp)

#processing

li $at, LEAVETRACK # change LEAVETRACK port

addi $k0, $zero,1 # to logic 1,

sb $k0, 0($at) # to start tracking

#restore

lw $k0, 0($sp)

addi $sp,$sp,-4

lw $at, 0($sp)

addi $sp,$sp,-4

jr $ra

nop

jr $ra

UNTRACK:

addi $sp,$sp,4

sw $at,0($sp)

#chuong trinh con

li $at, LEAVETRACK

sb $zero, 0($at)

#

lw $at, 0($sp)

addi $sp,$sp,-4

jr $ra

nop

jr $ra

ROTATE:

addi $sp,$sp,4

sw $t1,0($sp)

addi $sp,$sp,4

sw $t2,0($sp)

addi $sp,$sp,4

sw $t3,0($sp)

#chuong trinh con

li $t1, HEADING

la $t2, NOW\_HEAD

lw $t3, 0($t2)

sw $t3, 0($t1)

#

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

#-------------------------------------------------------------Digital Lab Sim----------------------------------------------

.ktext 0x80000180

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)

#chuong trinh

get\_cod:

li $t1, IN\_ADRESS\_HEXA\_KEYBOARD

li $t2, OUT\_ADRESS\_HEXA\_KEYBOARD

row1:

li $t3, 0x81

sb $t3, 0($t1)

lbu $a0, 0($t2)

bnez $a0, get\_code\_in\_char

row2:

li $t3, 0x82

sb $t3, 0($t1)

lbu $a0, 0($t2)

bnez $a0, get\_code\_in\_char

row3:

li $t3, 0x84

sb $t3, 0($t1)

lbu $a0, 0($t2)

bnez $a0, get\_code\_in\_char

row4:

li $t3, 0x88

sb $t3, 0($t1)

lbu $a0, 0($t2)

bnez $a0, get\_code\_in\_char

get\_code\_in\_char:

beq $a0, input\_0, so0

beq $a0, input\_1, so1

beq $a0, input\_2, so2

beq $a0, input\_3, so3

beq $a0, input\_4, so4

beq $a0, input\_5, so5

beq $a0, input\_6, so6

beq $a0, input\_7, so7

beq $a0, input\_8, so8

beq $a0, input\_9, so9

beq $a0, input\_a, a

beq $a0, input\_b, b

beq $a0, input\_c, c

beq $a0, input\_d, d

beq $a0, input\_e, e

beq $a0, input\_f, f

so0: li $s0, '0'

j inputCode

so1: li $s0, '1'

j inputCode

so2: li $s0, '2'

j inputCode

so3: li $s0, '3'

j inputCode

so4: li $s0, '4'

j inputCode

so5: li $s0, '5'

j inputCode

so6: li $s0, '6'

j inputCode

so7: li $s0, '7'

j inputCode

so8: li $s0, '8'

j inputCode

so9: li $s0, '9'

j inputCode

a: li $s0, 'a'

j inputCode

b: li $s0, 'b'

j inputCode

c: li $s0, 'c'

j inputCode

d: li $s0, 'd'

j inputCode

e: li $s0, 'e'

j inputCode

f: li $s0, 'f'

j inputCode

inputCode:#luu code nhap tu digital lab sim

la $s1, INPUT

la $s2, LENGTH

lw $s3, 0($s2)

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

sb $s0, 0($s1)

addi $s0, $zero, '\n'

addi $s1, $s1, 1

sb $s0, 0($s1)

addi $s3, $s3, 1

sw $s3, 0($s2)

restore:# tra lai gia tri thanh ghi

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

### Giải thích một số hàm:

1. KeyBoard and Display MMIO

+ ***Check***: Kiểm tra control code vừa nhập với mã điều khiên thiết lập sẵn

+ ***saveCode****:* Lưu trữ code trước khi xóa, để có thể restore nếu người dùng ấn phím space

+ ***reControl***: restore lại mã code trước đó

+ ***soSanh2chuoi***: so sánh mã code vừa nhập với mã điều khiển đã được thiết lập sẵn ở $s3

1. Mars Bot

+ ***inputPath***: -Lưu lại thông tin về đường đi của Marsbot vào mảng path

biến NOW\_HEAD, LENGTH\_PATH

-Mảng path lưu thông tin về đường đi hay đúng hơn là thông tin về các cạnh của đường đi của Marsbot. Mỗi một cạnh gồm 3 thông tin: tọa độ x và y của điểm đầu tiên, hướng đi của cạnh đó.

+ ***goBack:*** -điều khiển Marsbot đi ngược lại theo lộ trình nó đã đi và về điểm xuất phát

-mảng path lưu thông tin đường đi, biến lengthPath lưu kích cỡ của mảng path theo byte.

-Mảng path lưu thông tin đường đi. Mỗi thông tin về 1 cạnh gồm tọa độ x và 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 của 12. Mỗi khi muốn quay ngược lại và đi về điểm đầu tiên của 1 cạnh trên đường đi, ta sẽ lấy hướng đi của cạnh đó và đi ngược lại, đến khi nào gặp điểm có tọa độ như đã lưu thì kết thúc việc đi ngược trên cạnh đó, tiếp tục trên cạnh khác.

+ ***goRight và goLeft***:- điều khiển Marsbot sang phải hoặc trái

-Muốn di chuyển sang phải hoặc trái, ta chỉ cần tăng hoặc giảm biến NOW\_HEAD

+ ***ROTATE***: -quay Marsbot theo hướng có số độ lưu trong NOW\_HEAD

-Load biến NOW\_HEAD và lưu vào địa chỉ HEADING (0xffff8010) để Marsbot chuyển hướng.

+ ***TRACK, UNTRACK***:- điều khiển Marsbot bắt đầu để lại vết hoặc không

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

+***GO, STOP***:

điều khiển Marsbot bắt đầu chuyển động (GO) hoặc dừng lại (STOP) Load 1 vào địa chỉ MOVING (0xffff8050) nếu muốn để lại vết và load 0 nếu muốn kết thúc vết

1. Digital Lab Sim

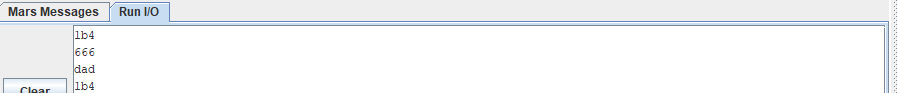
Lần lượt quét các hàng của Digital Lab Sim để xem phím nào được bấm. Tiếp đó dựa vào mã được trả về ghi kí tự tương ứng vào bộ nhớ, cụ thể là ghi vào cuối xâu INPUT.

## **Kết quả**

* Mars Bot bắt đầu chạy và in vết:

Graphical user interface, application, Word

Description automatically generated



* Mars Bot quay sang phải:

Chart

Description automatically generated with low confidence

* Mars Bot quay sang trái:

Graphical user interface

Description automatically generated with low confidence

* Thực hiện ngược lại quá trình:

Graphical user interface, application, Word

Description automatically generated with medium confidence

# ***ĐỀ 5***

## **Đề bài : Biểu thức trung tố, hậu tố**

Graphical user interface, text, application

Description automatically generated

## **Phân tích cách làm**

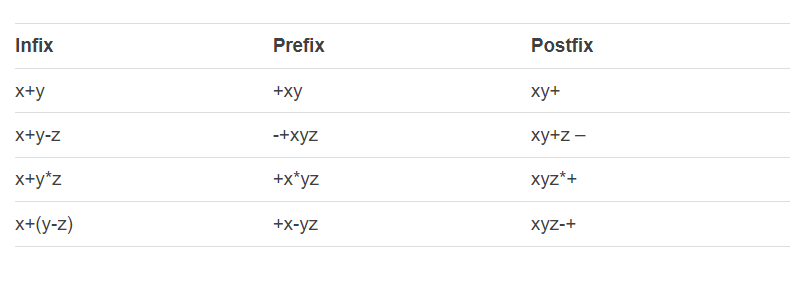
**Đổi biểu thức trung tố sang hậu tố.**

+ Biểu thức hậu tố (postfix) : các toán tử sẽ được đặt sau các toán hạng

+ Mức độ ưu tiên của các toán tử:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Toán tử | + | - | \* | / | % | ( | ) |
| Mức độ ưu tiên | 1 | 1 | 2 | 2 | 2 | 0 | 0 |

+ Ví dụ về cách chuyển đổi:



## **Thuật toán**

Ý tưởng: Sử dụng ngăn xếp và xâu

Các bước thực hiện:

* B1:Đổi biểu thức trung tố sang hậu tố.

+Tạo 1 ngăn xếp .

+ Khi biểu thức đầu vào không có lỗi và chưa duyệt tới cuối biểu thức thì lấy giá trị kí tự tại vị trí i ( char c = str(i)).

-Nếu c == ‘(‘ => cho vào stack.

-Nếu c là chữ số => Lưu vào xâu

-Nếu c là một toán tử => Nếu stack không rỗng,ta kiểm tra mức độ ưu tiên của toán tử c với toán tử top của stack. Nếu mức độ ưu tiên của c <= mức độ ưu tiên của top thì in top ra màn hình và tiến hành lặp lại bước so sánh trên. Ngược lại, t cho c vào trong stack.

-Nếu c == ‘)’ => In các phần tử trong stack ra màn hình cho tới khi gặp dấu ‘(‘ . Không xuất ra dấu ‘(‘

* B2: Tính giá trị biểu thức hậu tố:

+ Duyệt biểu thức dạng chuỗi từ trái sang phải:

+Tạo 1 ngăn xếp mới.

-Nếu là toán hạng thì đưa vào ngăn xếp .

-Nếu là toán tử thì Pop 2 toán hạng trong ngăn xếp ra, sau đó tính toán giá trị của chúng dựa vào toán tử này, sau đó Push lại vào ngăn xếp.

-Thực hiện cho đến khi kết thúc chuỗi.

+Kết quả của biểu thức chính là phần tử còn lại cuối cùng trong ngăn xếp.

## **Mã nguồn**

.data

infix: .space 256

postfix: .space 256

stack: .space 256

Mess1: .asciiz "Nhap bieu thuc trung to:\n(Luu y!) Cac so nhap vao nam trong khoang 0-99"

Xuong\_dong: .asciiz "\n"

Mess2: .asciiz "Bieu thuc hau to: "

Mess4: .asciiz "Ket qua: "

Mess3: .asciiz "Bieu thuc trung to: "

# Khoi tao bieu thuc trung to

.text

#-------Nhap bieu thuc trung to---------

li $v0, 54

la $a0, Mess1

la $a1, infix # nhap BT trung to

la $a2, 256

syscall

#--------In ra BT trung to--------------

la $a0, Mess3

li $v0, 4 # In mess

syscall

la $a0, infix #In Infix

li $v0, 4

syscall

#----------Chuyen doi BT trung to sang hau to------

li $s6, -1 # i Bien dem trong BT trung to

li $s7, -1 # k Bien dem trong stack

li $t7, -1 # j Bien diem trong bieu thuc hau to

while:

la $s1, infix # $s1 = address (infix)

la $t5, postfix # $t5 = address(postfix )

la $t6, stack # $t6 = address(stack)

li $s2, '+' # gan $s2 = '+'

li $s3, '-' # gan $s3 = '-'

li $s4, '\*' # gan $s4 = '\*'

li $s5, '/' # gan $s5 = '/'

li $a3, '%' # gan $a3 = '%'

addi $s6, $s6, 1 # i = i ++

# Xet Infix[i]

add $s1, $s1, $s6 #$s1= address (infix(i))

lb $t1, 0($s1) # $t1 = gia tri infix(i)

beq $t1, $s2, toan\_tu # '+'

nop

beq $t1, $s3, toan\_tu # '-'

nop

beq $t1, $s4, toan\_tu # '\*'

nop

beq $t1, $s5, toan\_tu # '/'

nop

beq $t1, $a3, toan\_tu # '%'

nop

beq $t1, 10, khong\_la\_toan\_tu # '\n' #(ki tu \0) ket thuc infix

nop

beq $t1, 32, khong\_la\_toan\_tu # ' ' # dau cach

nop

beq $t1, $zero, endWhile # Neu Infix[] khong con phan tu nao thi ket thuc

nop

# ---------Dua toan hang vao postfix--------------

addi $t7, $t7, 1 # j = j++

add $t5, $t5, $t7 # $t5 = address(Postfix[j])

sb $t1, 0($t5) # $t1 = postfix[j]

lb $a0, 1($s1) # $a0 = infix[i+1]

jal check\_number # $v0 = 1 khi infix[i] la toan hang

beq $v0, 1, khong\_la\_toan\_tu # neu no la toan hang=> xet ki tu tiep theo

nop

add\_space: # Them dau cach giua cac phan tu

add $t1, $zero, 32

sb $t1, 1($t5)

addi $t7, $t7, 1

j khong\_la\_toan\_tu

nop

toan\_tu:

# add to stack ...

beq $s7, -1, pushToStack #neu stack rong => push to stack

nop

add $t6, $t6, $s7

lb $t2, 0($t6) # t2 = value of stack[counter]

#-------------------Kiem tra muc do uu tien cua toan tu ----------

# $t1 là infix[i] ,$t2 là top(stack) $s2 la '+',$s3 là '-'

beq $t1, $s2, Muc\_1 #Neu toan tu la '+' hoac '-'=> muc 1

nop

beq $t1, $s3, Muc\_1

nop

li $t3, 2 # Neu phan tu dang xet ko phai '+' hay '-' thi co muc do uu tien la 2

#$t3: Muc do uu tien cua Infix(i)

j check\_t2 #Kiem tra muc do uu tien phan tu top(stack)

nop

Muc\_1:

li $t3, 1

# Xet do uu tien cua top (stack)

check\_t2:

beq $t2, $s2, Muc1 #Neu toan tu la '+' hoac '-'=> muc 1

nop

beq $t2, $s3, Muc1

nop

li $t4, 2 # Neu phan tu dang xet ko phai '+' hay '-' thi co muc do uu tien la 2

#$t4: Muc do uu tien cua Top(stack)

j So\_sanh\_do\_uu\_tien

nop

Muc1:

li $t4, 1

#So sanh muc do uu tien cua phan tu top(stack) vs phan tu dang xet trong Infix

So\_sanh\_do\_uu\_tien:

beq $t3, $t4, Dong\_muc\_do

nop

slt $s1, $t3, $t4 #$t3 <= $t4 = 1:0

beqz $s1, t3\_hon\_t4 # $t3 >= $t4 => Dua t1 vao trong stack

nop

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

# t3 < t4

# pop t2 from stack va push t2 vao postfix

sb $zero, 0($t6)

addi $s7, $s7, -1 # scounter ++

addi $t6, $t6, -1

la $t5, postfix #postfix = $t5

addi $t7, $t7, 1

add $t5, $t5, $t7

sb $t2, 0($t5)

j toan\_tu # So sanh $t1 vs phan tu tiep stack

nop

t3\_hon\_t4:

# push t1 to stack

j pushToStack

nop

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

Dong\_muc\_do:

# pop t2 tu stack va luu vao postfix

# push to stack

sb $zero, 0($t6) #xoa phan tu $t2 khoi stack

addi $s7, $s7, -1 # k = k-- (bien dem trong stack )

la $t5, postfix # $t5 = address(postfix[0])

addi $t7, $t7, 1 # j = j++ (Bien diem postfix)

add $t5, $t5, $t7 #tinh dia chi postfix[k]

sb $t2, 0($t5) #luu $t2 vao Postfix

j pushToStack

nop

#---------Cho ki tu vao trong stack

pushToStack:

la $t6, stack #stack = $t6

addi $s7, $s7, 1 # scounter ++

add $t6, $t6, $s7 # $t6 = address(stack(j))

sb $t1, 0($t6) #Luu gia tri vao stack

khong\_la\_toan\_tu: # xet phan tu ke tiep

j while

nop

endWhile: # lay ra dia chi top cua stack ($t6)

#Khi gap \0 thi ket thuc

addi $s1, $zero, 32

add $t7, $t7, 1

add $t5, $t5, $t7 # $t5 = top (postfix)

la $t6, stack

add $t6, $t6, $s7

popallstack:

lb $t2, 0($t6) # t2 = stack[i]

beq $t2, 0, endPostfix

sb $zero, 0($t6)

addi $s7, $s7, -2

add $t6, $t6, $s7

sb $t2, 0($t5)

add $t5, $t5, 1

j popallstack

nop

endPostfix:

# ----------------In ra bieu thuc hau to ---------------------------

la $a0, Mess2

li $v0, 4

syscall

la $a0, postfix

li $v0, 4

syscall

la $a0, Xuong\_dong #Xuong dong

li $v0, 4

syscall

#-----------------------TINH GIA TRI BIEU THUC HAU TO------------------------

#Sau khi pop het ra postfix => stack rong

li $s3, 0 # $s3 = 0 ,Bien dem cua stack (i)

la $s2, stack #stack = $s2

# Postfix to stack

while\_P\_to\_S:

la $s1, postfix #postfix = $s1

add $s1, $s1, $s3 # lay ra dia chi cua postfix[i]

lb $t1, 0($s1) # $t1 = postfix[i]

# Neu xau postfix ket thuc

beqz $t1 end\_while\_p\_to\_S

nop

add $a0, $zero, $t1 # $a0 = postfix[i]

jal check\_number # $v0 = 1 => La toan hang

nop # $v0 = 0 => la toan tu

beqz $v0, la\_toan\_tu

nop

jal Them\_so\_vao\_stack # Neu postfix(i) la toan hang => Them vao stack

nop

j continue

nop

#Neu postfix(i) la toan tu thi pop ra

la\_toan\_tu:

jal pop #pop toan hang tu stack

nop

add $a1, $zero, $v0 #$a1 la toan hang t1

jal pop #pop toan hang tu stack

nop

add $a0, $zero, $v0 #$a0 la toan hang t2

add $a2, $zero, $t1 # toan tu

jal Tinh

continue: # bo qua dau cach

add $s3, $s3, 1 # i++

j while\_P\_to\_S

nop

#-----------------------------Tinh--------------------------

#$t1 = postfix(i) khi la toan tu

Tinh:

sw $ra, 0($sp) # luu $ra vao $sp

li $v0, 0 # gan $v0 = 0

beq $t1, '\*', Phep\_nhan

nop

beq $t1, '/', Phep\_chia

nop

beq $t1, '+', Phep\_cong

nop

beq $t1, '-', Phep\_tru

nop

beq $t1 , '%',Chia\_du

nop

Phep\_nhan:

mul $v0, $a0, $a1

j Push\_KQ

Phep\_chia:

div $a0, $a1

mflo $v0

j Push\_KQ

Phep\_cong:

add $v0, $a0, $a1

j Push\_KQ

Phep\_tru:

sub $v0, $a0, $a1

j Push\_KQ

Chia\_du:

div $a0,$a1

mfhi $v0

j Push\_KQ

Push\_KQ: # day gia tri tinh duoc vao stack

add $a0, $v0, $zero #$a0 = $v0

jal push

nop

lw $ra, 0($sp)

jr $ra #Quay lai vong lap while\_P\_to\_S

nop

#----------------------------Them so vao stack----------------

#$s3 : bien dem cua postfix[] ,$s2 = stack

#$s1 = postfix[i]

# $t1 :toan hang dang xet

Them\_so\_vao\_stack:

sw $ra, 0($sp) #luu $ra vao $sp

li $v0, 0

While\_add: # while add number to stack

beq $t1, '0', case\_0

nop

beq $t1, '1', case\_1

nop

beq $t1, '2', case\_2

nop

beq $t1, '3', case\_3

nop

beq $t1, '4', case\_4

nop

beq $t1, '5', case\_5

nop

beq $t1, '6', case\_6

nop

beq $t1, '7', case\_7

nop

beq $t1, '8', case\_8

nop

beq $t1, '9', case\_9

nop

case\_0:

j end\_case

case\_1:

addi $v0, $v0, 1

j end\_case

nop

case\_2:

addi $v0, $v0, 2

j end\_case

nop

case\_3:

addi $v0, $v0, 3

j end\_case

nop

case\_4:

addi $v0, $v0, 4

j end\_case

nop

case\_5:

addi $v0, $v0, 5

j end\_case

nop

case\_6:

addi $v0, $v0, 6

j end\_case

nop

case\_7:

addi $v0, $v0, 7

j end\_case

nop

case\_8:

addi $v0, $v0, 8

j end\_case

nop

case\_9:

addi $v0, $v0, 9

j end\_case

nop

end\_case:

add $s3, $s3, 1 # counter++

la $s1, postfix #postfix = $s1

add $s1, $s1, $s3 #Tinh dia chi postfix[i]

lb $t1, 0($s1) # Luu $t1 = postfix[i]

beq $t1, $zero, end\_While\_add

beq $t1, ' ', end\_While\_add

mul $v0, $v0, 10 #$v0 = $v0 \* 10

j While\_add

end\_While\_add:

add $a0, $zero, $v0 #luu $a0 = $v0

jal push

lw $ra, 0($sp) # get $ra

jr $ra

nop

#-----------------Kiem tra ki tu nhap vao co phai so hay khong?-------------------

check\_number:

li $t8, '0' # $t8 = 0

li $t9, '9' #$t9 = 9

beq $t8, $a0, toan\_hang # Kiem tra xem $a0 co thuoc khoang tu 0 - 9 khong?

beq $t9, $a0, toan\_hang

slt $v0, $t8, $a0 # 0 < $a0=> $v0 = 1 : 0

beqz $v0, khong\_la\_toan\_hang # $v0 = 0 => ko phai so

slt $v0, $a0, $t9 # $a0 < 9 => $v0 = 1: 0

beqz $v0, khong\_la\_toan\_hang # $v0 = 0 => ko phai so

toan\_hang:

li $v0, 1

jr $ra

nop

khong\_la\_toan\_hang:

li $v0, 0

jr $ra

nop

#---------------------Pop toan hang tu stack ------------------------------------------

#$s2 = stack

#$v0 : gia tri pop

pop:

lw $v0, -4($s2) #$v0 = top

sw $zero, -4($s2) #Lay ra gia tri top va xoa gtri top

add $s2, $s2, -4 #giam dia chi top

jr $ra

nop

#----------------------Push gia tri sau khi tinh vao stack----------------------------------

#$a0: gia tri push

push:

sw $a0, 0($s2) # Luu top = $a0

add $s2, $s2, 4 # tang dia chi

jr $ra

nop

end\_while\_p\_to\_S:

# add null to end of stack

# -------------In ra gia tri bieu thuc hau to----------------------------

la $a0, Mess4

li $v0, 4

syscall

jal pop

add $a0, $zero, $v0

li $v0, 1

syscall

la $a0, Xuong\_dong

li $v0, 4

syscall

# **Kết quả**

