

**TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI**

**VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG**



**BÁO CÁO**

**Final projects Course 2019-02**

**Môn học: Thực hành Kiến trúc máy tính – IT3238**

*Giảng viên hướng dẫn:*

ThS. Lê Bá Vui

*Sinh viên thực hiện:*

*Nhóm 16:*

Kiều Đăng Nam - 20176830

Lê Minh Quang - 20176856

*Mã lớp:* 113834

Hà Nội, tháng 6 năm 2020

**Mục Lục**

[**I. Bài 1** 4](#_Toc43718741)

[1. Đề bài 4](#_Toc43718742)

[2. Phân tích cách thực hiện 4](#_Toc43718743)

[3. Ý nghĩa các chương trình con 5](#_Toc43718744)

[4. Kết quả chạy chương trình. 6](#_Toc43718745)

[**II. Bài 10** 8](#_Toc43718746)

[1. Phân tích cách thực hiện: 8](#_Toc43718747)

[2. Ý nghĩa các thanh ghi: 8](#_Toc43718748)

[3. Ý nghĩa chương trình con 9](#_Toc43718749)

[4. Kết quả thực hiện 10](#_Toc43718750)

[**III. Source Code** 12](#_Toc43718751)

[1. Bài 1 12](#_Toc43718752)

[2. Bài 10 25](#_Toc43718753)

**Báo cáo Final - project giữa kỳ 20192**

**Môn học: Thực hành Kiến trúc máy tính**

Trong báo cáo này , nhóm em xin được trình bày nội dụng thực hiện các chủ đề đã được phân công như sau:

Phần 1: Bài 1 do Kiều Đăng Nam thực hiện.

Phần 2: Bài 10 do Lê Minh Quang thực hiện.

# **I. Bài 1**

1. **Đề bài**

Curiosity Mars Bot: Xe tự hành Curiository MarsBot chay jtreen sao Hỏa, được vận hành từ xa bởi các lập trình viên trên Trái Đất bằng cách gửi các mã điều khiển từ một bàn phím ma trận. Các mã điều khiển quá trình di chuyển của MarsBot như sau:

|  |  |
| --- | --- |
| **Mã điều khiển** | **Ý nghĩa** |
| 1b4 | Marbot bắt đầu chuyển động |
| c68 | Marbot đứng im |
| 444 | Rẽ trái 90\* so vơi phương chuyển động gần đây và giữ hướng mới |
| 666 | Rẽ phải 90\* so vơi phương chuyển động gần đây và giữ hướng mới |
| dad | Bắt đầu để lại vết trên đường |
| cbc | Chấm dứt để lại vết trên đường |
| 999 | Tự động quay trở lại 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 (dù có thể lệch một chút do hàm syscall sleep không thực sự thời gian thực) |

Sau khi nhận mã điều khiển, Curioisity Marsbot sẽ không xử lý ngay, mà phải đợi lệnh kích hoạt mã từ bàn phím Keyboard & Display MMIO Simulator. Có 2 lệnh như vậy:

|  |  |
| --- | --- |
| **Kích hoạt mã** | **Ý nghĩa** |
| Phím Enter | Kết thúc nhập mã và yêu cầu Marbot thực thi. |
| Phím Del | Xóa toàn bộ mã điều khiển đang nhập |

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

1. **Phân tích cách thực hiện**

Ý tưởng thực hiện: bài tập được lấy ý tưởng từ việc tổng hợp của 3 home assignment trong các bài thực hành week 10, 11. (Home Assignment 3, 4 – Week 10, Home Assignment 3 – Week 11)

* Mỗi khi người dùng nhập một ký vào trong Data Lab Sim sẽ tạp ra interrupt để lưu ký tự được nhập vào bộ nhớ, tạo nên CodeControl (Mã điều khiển)
* Tiếp theo là kiểm tra liên tục xem ký tự Enter/Delete có được nhập trong cửa sổ Keyboard & Display MMIO Simulator hay không.

+ Trường hợp là enter, chương trình sẽ tiến hành kiểm tra xem mã Code được nhập có hợp lệ hay không, về độ dài (gồm 3 ký tự), nếu không sẽ thông báo lỗi WRONG CODE. Nếu thỏa mãn về độ dài sẽ tiế tục kiểm tra xem mã code đó có trùng khớp với các mã code điều khiển đã được quy định trước đó. Nếu không thì thông báo WRONG CODE. Ngược lại sẽ chuyển đến hàm thực hiện các hành động của bot tương ứng với mã code điều khiển.

+ Trường hợp Delete, chương trình sẽ xóa toàn bộ mã điều khiển đang nhập.

* In ra console code điều khiển.

1. **Ý nghĩa các chương trình con**
   1. ***Các hàm thực hiện di chuyển của Marsbot***
2. **storePath**

* Chức năng: Lưu lại thông tin về đường đi của Marsbot vào mảng path, mỗi phần tử được lưu dưới dạng cấu trúc (x, y, z) với x,y là điểm tạo độ điểm đầu tiên, z là hướng đi của cạnh đó.
* Input: nowHeading, lengthPath.

1. **goBack**

* Chức năng: Marsbot đi ngược lại theo lộ trình nó đã đi, về điểm xuất phát. Mỗi lần quay ngược lại và đi về điểm đầu tiên của một cạnh trên đường đi, chương trình lấy hướng đi cạnh đó và đi ngược lại đến khi gặp điểm có tọa độ như đã lưu thì kế thúc việc đi ngược lại trên cạnh và tiếp tục trên các cạnh khác
* Input: mảng path lưu thông tin về đường đi, biến lengthPath lưu kích cỡ của mảng Path ( =12bytes)

1. **goRight, goLeft**

* Chức năng: Điều khiển Marsbot di chuyển quay sang phải (trái) một góc 900.
* Input: biến nowHeading. Khi di chuyển sang phải ta chỉ cần tăng biến nowHeading lên 900 và bên trái sẽ giảm 900. Và gọi hàm ROTATE để thực hiện.

1. **ROTATE**

* Chức năng: quay Marsbot theo hướng có góc được lưu trong nowHeading.
* Input: biến nowHeading.

1. **TRACK, UNTRACK**

* Chức năng: điều khiển Marsbot bắt đầu để lại vết (TRACK) hoặc dừng việc ghi vết (UNTRACK).
* Thực hiện: Load 1 vào địa chỉ LEAVETRACK nếu muốn để lại vết và 0 nếu kết thúc ghi vết.

1. **GO, STOP**

* Chức năng: điều khiển Marsbot bắt đầu chuyển động (GO) hoặc dừng lại (STOP).
* Thực hiện: Load 1 vào địa chỉ MOVING nếu muốn đi và 0 khi dừng lại
  1. ***Các hàm liên quan đến ControlCode (mã điều khiển)***

1. **checkCodeControl**

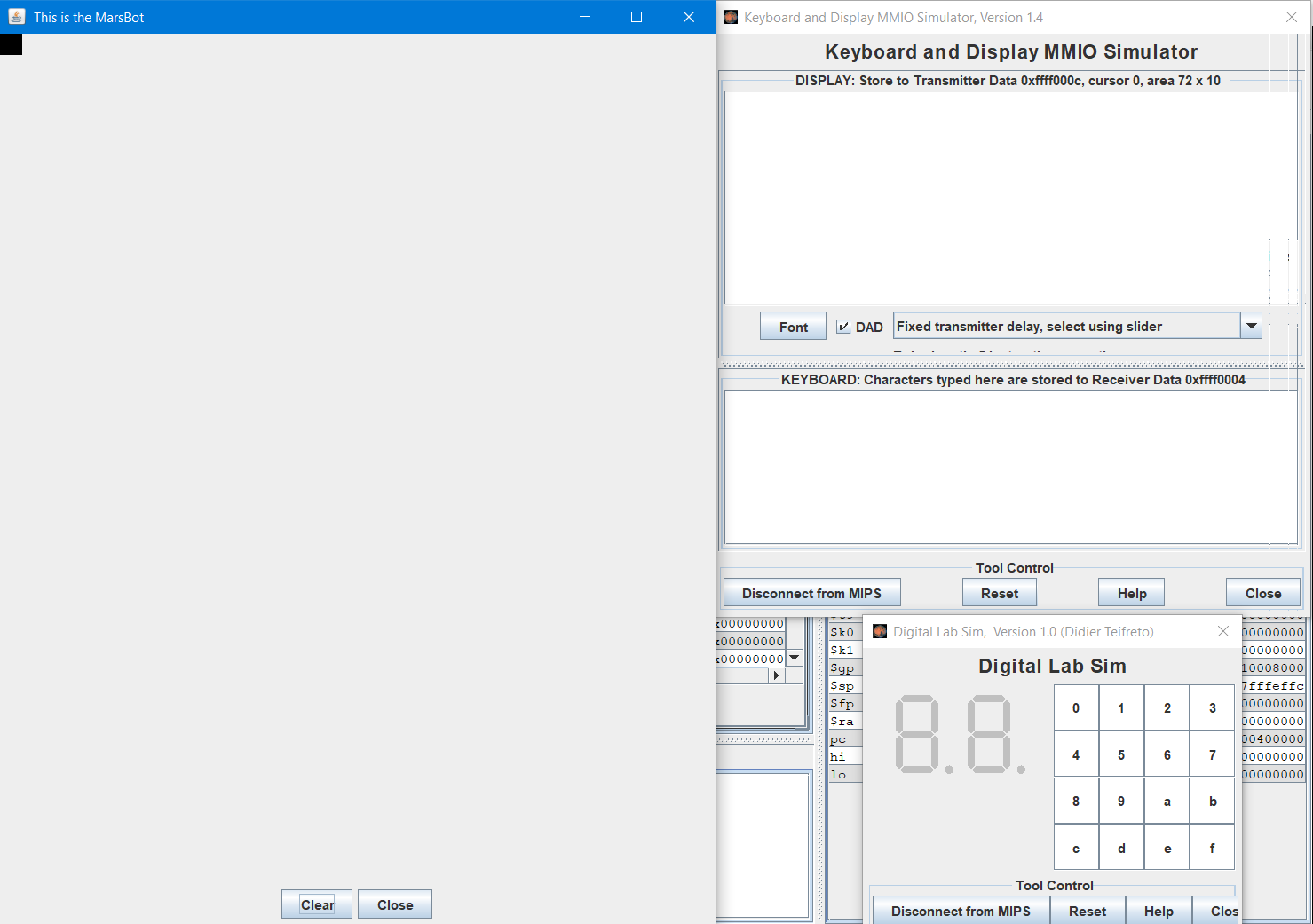
* Chức năng: Kiểm tra mã code điều khiển người dùng nhập (lưu trong $s1) có trùng với các mã code đã được định nghĩa sẵn (lưu trong $s3) hay không.

Nếu 2 xâu này bằng nhau thì thanh ghi $s0 = 1, ngược lại = 0.

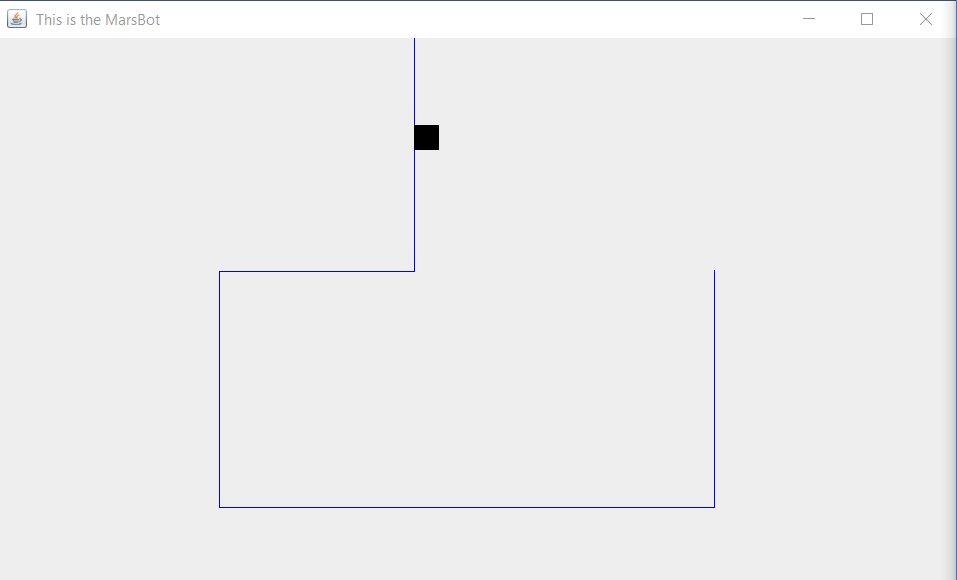
1. **removeControlCode**

* Chức năng: xóa xâu inputControlCode (địa chỉ lưu trữ mã code điều khiển nhập vào)
* Thực hiện bằng cách gán các ký tự trong xâu = ‘\0’

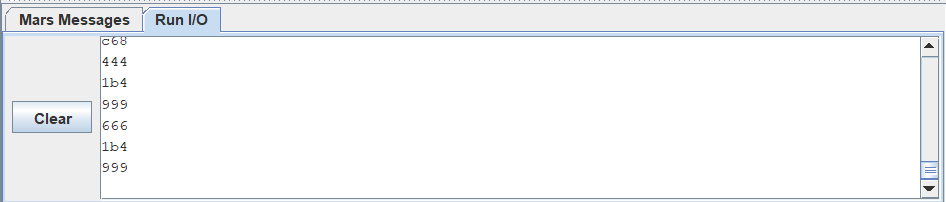
1. **Kết quả chạy chương trình.**
2. Bắt đầu khởi chương trình



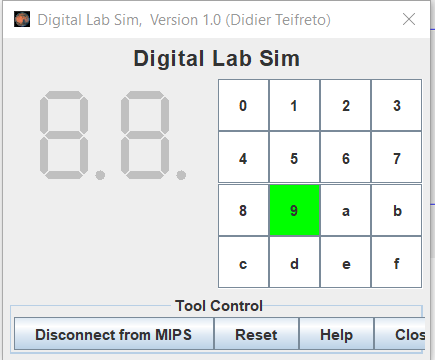
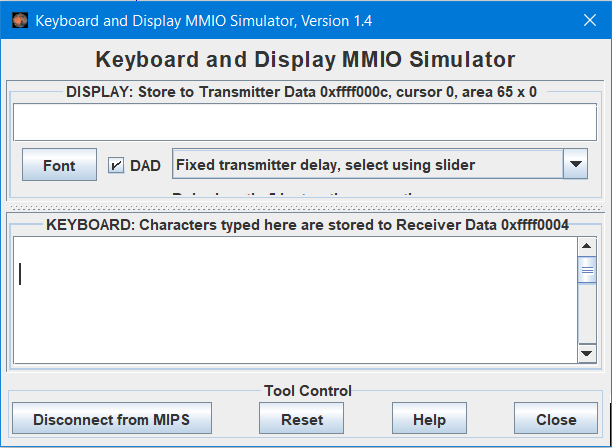
1. Kết quả điều khiển MarsBot qua các mã điều khiển



*Màn hình MarsBot*

**

*Màn hình Console để hiện thị path thông qua các mã điều khiển*



*Digital Lab Sim để nhập mã điều khiển Keyboard để xác nhận việc nhập lệnh*

# **II. Bài 10**

**Đề bài:**

Sử dụng 2 ngoại vi là bàn phím và led 7 thanh để xây dựng một máy tính bỏ túi đơn giản. Hỗ trợ các phép toán +, -, \*, /. Do trên bàn phím không có các phím trên nên sẽ dùng các phím

* Bấm phím a để nhập phép tính +
* Bấm phím b để nhập phép tính –
* Bấm phím c để nhập phép tính \*
* Bấm phím d để nhập phép tính /
* Bấm phím f để nhập phép =

Yêu cầu cụ thể như sau:

* Khi nhấn các phím số, hiển thị lên LED, do chỉ có 2 LED nên chỉ hiện thị 2 số cuối cùng. Ví dụ khi nhấn phím 1 → hiện thị 01. Khi nhấn thêm phím 2 → hiển thị 12. Khi nhấn thêm phím 3 → hiển thị 23.
* Sau khi nhập số, sẽ nhập phép tính + - \* /
* Sau khi nhấn phím f (dấu =) , tính toán và hiển thị kết quả lên LED.

1. **Phân tích cách thực hiện:**

* Khi nhấn nút đầu tiên, lưu giá trị vào thanh ghi $s3. Nếu nút tiếp theo cũng nhập số, cập nhật lại giá trị $s3. Ngược lại nếu nút tiếp theo là phép toán, sẽ chuyển lưu giá trị sang thanh ghi $s5( số thứ 2 ). Thực hiện phép toán giữa 2 thanh ghi $s3, $s5.
* In kết quả ra cả màn hình console và cả trên LED

1. **Ý nghĩa các thanh ghi:**

|  |  |
| --- | --- |
| **Thanh ghi** | **Ý nghĩa** |
| $a0 | Lưu mã hiển thị, kết quả tính |
| $t0 | Mã phím quét được |
| $t1 | Địa chỉ input của bàn phím |
| $t2 | Địa chỉ output của bàn phím |
| $t3 | Mã của hàng chứa phím C, D, E, F |
| $t4 | Mã của hàng chứa phím 0, 1, 2, 3 |
| $t5 | Mã của hàng chứa phím 4, 5, 6, 7 |
| $t6 | Mã của hàng chứa phím 8, 9, A, B |
| $s0, $a1 | Mã hiển thị số 0 |
| $s2 | Lưu giá trị phím được ấn |
| $s1 | Đèn báo. 1 nếu phím đang được ấn, 0 nếu phím được thả |
| $s3 | Giá trị toán hạng 1 |
| $s5 | Giá trị toán hạng 2 |
| $s4 | 0 nếu đang xử lý toán hạng 1. 1 nếu đang xử lý toán hạng 2 |

1. **Ý nghĩa chương trình con**

**polling**: Dùng để quét các hàng

**checkButton**: Kiểm tra giá trị của $t0 để xem phím nào đang được bấm. Nếu có phím được bấm nhảy sang pressedButton tương ứng. Nếu không nhảy sang freeButton

**button( 0 – F ):** Nạp mã hiển thị tương ứng vào $a0. Gán giá trị phím được ấn vào $s2

**free**: Đặt lại mã hiển thị số 0 cho $a0, đồng thời đặt lại $s1 = 0 ( phím đang ấn đã được thả hay chưa có phím mới nào được ấn)

**processButton**: Xử lý phím đang được ấn

* Nếu phím hiện tại vẫn đang được ấn 🡪 không làm gì cả. Nếu có phím mới được ấn, chuyển mã hiển thị hàng đơn vị sang cho hàng chục
* Thực hiện xử lý toán hạng 1 hoặc 2

**secondNumber**: Xử lý toán hạng thứ 2

**printInt**: In số vừa nhấn ra màn hình console

**processMath**: Khi ấn các phím +, - , \*, /

* Đặt lại mã hiển thị cho đèn LED về 0
* In dấu phép toán ra màn hình console
* Đặt lại giá trị $s2 = 0, $s4 = 1( Kết thúc xử lý toán hạng thứ nhất )

**isOldNumber**: Chưa có phím mới được ấn( $s1 = 1 )

**showResult**:

* In kết quả ra màn hình console thông qua thanh ghi $a0
* Lưu số dư khi chia kết quả cho 10 vào $t0

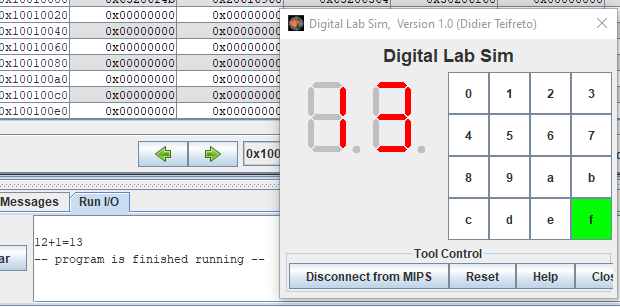
**SHOW\_LED\_RIGHT**: In LED phải

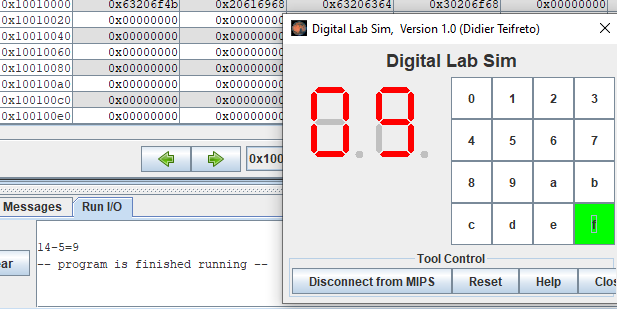
**SHOW\_LED\_LEFT**: In LED trái

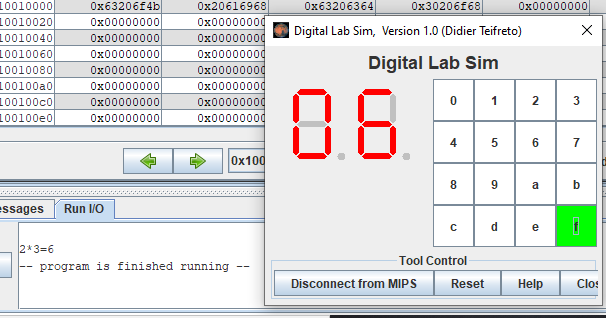
**DATA\_FOR\_LED**: Kiểm tra thanh ghi $t0 để đặt lại mã hiển thị đèn LED tương ứng

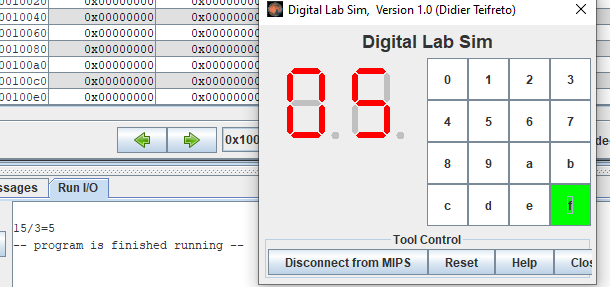
1. **Kết quả thực hiện**

**Phép cộng**: 12 + 1

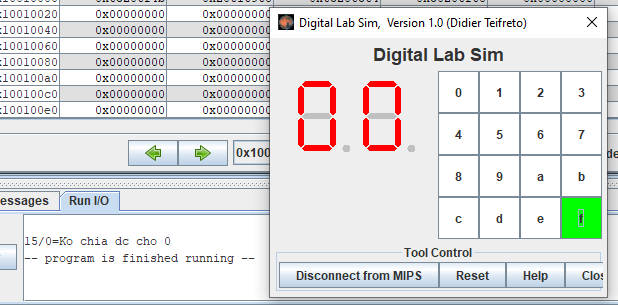


**Phép trừ**: 14-5

**Phép nhân:** 2 \* 3

**Phép chia**: 15 / 3

**Nếu số chia = 0** : 15 / 0



# **III. Source Code**

1. **Bài 1**

# FinalExam - Curiosity Marsbot, author: Kieu Dang Nam - 20176830

# Keyboard----------------------------------------------------------------------

.eqv IN\_ADRESS\_HEXA\_KEYBOARD 0xFFFF0012

.eqv OUT\_ADRESS\_HEXA\_KEYBOARD 0xFFFF0014

.eqv KEY\_CODE 0xFFFF0004 # ASCII code from keyboard, 1 byte

.eqv KEY\_READY 0xFFFF0000 # =1 if has a new keycode ?

# Auto clear after lw

# 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: An angle between 0 and 359

# 0 : North (up)

# 90: East (right)

# 180: South (down)

# 270: West (left)

.eqv MOVING 0xffff8050 # Boolean: whether or not to move

.eqv LEAVETRACK 0xffff8020 # Boolean (0 or non-0):

# whether or not to leave a track

.eqv WHEREX 0xffff8030 # Integer: Current x-location of MarsBot

.eqv WHEREY 0xffff8040 # Integer: Current y-location of MarsBot

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

.data

#Control code-------------------------------------------------------------------

MOVE\_CODE: .asciiz "1b4"

STOP\_CODE: .asciiz "c68"

GO\_LEFT\_CODE: .asciiz "444"

GO\_RIGHT\_CODE: .asciiz "666"

TRACK\_CODE: .asciiz "dad"

UNTRACK\_CODE: .asciiz "cbc"

GO\_BACK\_CODE: .asciiz "999"

WRONG\_CODE: .asciiz "Invalid code control ! Please enter again !"

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

inputControlCode: .space 50

lengthControlCode: .word 0

nowHeading: .word 0

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

# duong di cua marsbot duoc luu tru vao mang path

# moi 1 canh duoc luu tru duoi dang 1 structure {x, y, z}

# trong do: x, y la toa do diem dau tien cua canh

# z la huong cua canh do

# do dai duong di ngay khi bat dau la 12 bytes (3x 4byte)

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

path: .space 600

lengthPath: .word 12 #bytes

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

.text

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# MAIN PROCEDURE

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

main:

li $k0, KEY\_CODE

li $k1, KEY\_READY

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

# Enable the interrupt of Keyboard matrix 4x4 of Digital Lab Sim

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

li $t1, IN\_ADRESS\_HEXA\_KEYBOARD

li $t3, 0x80 # bit 7 = 1 to enable

sb $t3, 0($t1)

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

loop: nop

WaitForKey:

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

beq $t5, $zero, WaitForKey #if $t5 == 0 then Polling

nop

beq $t5, $zero, WaitForKey

ReadKey:

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

beq $t6, 127 , continue #if $t6 == delete key then remove input

#127 is delete key in ascii

bne $t6, '\n' , loop #if $t6 != '\n' then Polling

nop

bne $t6, '\n' , loop

CheckControlCode:

la $s2, lengthControlCode

lw $s2, 0($s2)

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

bne $s2, 3, pushErrorMess

la $s3, MOVE\_CODE

jal checkCodeControl

beq $t0, 1, go

la $s3, STOP\_CODE

jal checkCodeControl

beq $t0, 1, stop

la $s3, GO\_LEFT\_CODE

jal checkCodeControl

beq $t0, 1, goLeft

la $s3, GO\_RIGHT\_CODE

jal checkCodeControl

beq $t0, 1, goRight

la $s3, TRACK\_CODE

jal checkCodeControl

beq $t0, 1, track

la $s3, UNTRACK\_CODE

jal checkCodeControl

beq $t0, 1, untrack

la $s3, GO\_BACK\_CODE

jal checkCodeControl

beq $t0, 1, goBack

beq $t0, 0, pushErrorMess

printControlCode:

li $v0, 4

la $a0, inputControlCode

syscall

nop

continue:

jal removeControlCode

nop

j loop

nop

j loop

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

# storePath procedure, store path of marsbot to path variable

# param[in] nowHeading variable

# lengthPath variable

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

storePath:

#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 #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 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

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

# goBack procedure, control marsbot go back

# param[in] path array, lengthPath array

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

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 $t8, 0($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

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

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

nop

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

nop

bne $t8, $t9, go\_to\_first\_point\_of\_edge

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) #update heading

la $t8, lengthPath

addi $s5, $zero, 12

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

#restore

lw $t9, 0($sp)

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 printControlCode

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

# track procedure, control marsbot to track and print control code

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

track: jal TRACK

j printControlCode

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

# untrack procedure, control marsbot to untrack and print control code

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

untrack: jal UNTRACK

j printControlCode

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

# go procedure, control marsbot to go and print control code

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

go: jal GO

j printControlCode

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

# stop procedure, control marsbot to stop and print control code

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

stop: jal STOP

j printControlCode

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

# goRight procedure, control marsbot to go left and print control code

# param[in] nowHeading variable

# param[out] nowHeading variable

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

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) #$s6 is heading at now

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

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

#restore

lw $s6, 0($sp)

addi $sp,$sp,-4

lw $s5, 0($sp)

addi $sp,$sp,-4

jal storePath

jal ROTATE

j printControlCode

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

# goLeft procedure, control marsbot to go left and print control code

# param[in] nowHeading variable

# param[out] nowHeading variable

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

goLeft:

#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) #$s6 is heading at now

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

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

#restore

lw $s6, 0($sp)

addi $sp,$sp,-4

lw $s5, 0($sp)

addi $sp,$sp,-4

jal storePath

jal ROTATE

j printControlCode

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

# removeControlCode procedure, to remove inputControlCode string

# inputControlCode = ""

# param[in] none

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

removeControlCode:

#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

for\_loop\_to\_remove:

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

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

sb $t2, 0($s1) #inputControlCode[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) #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

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

# checkCodeControl procedure, to check inputControlCode string

# is equal with string s (store in $s3 )

# Length of two string is the same

# param[in] $s3, store address of a string

# param[out] $t0, 1 if equal, 0 is not equal

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

checkCodeControl:

#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

for\_loop\_to\_check\_equal:

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, isNotEqual #if $t2 != $t3 -> not equal

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

nop

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

isEqual:

#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, 1 #update $t0

jr $ra

nop

jr $ra

isNotEqual:

#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 #update $t0

jr $ra

nop

jr $ra

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

# pushErrorMess procedure, to announce the inputed control code is wrong

# param[in] none

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

pushErrorMess:

li $v0, 4

la $a0, inputControlCode

syscall

nop

li $v0, 55

la $a0, WRONG\_CODE

syscall

nop

nop

j continue

nop

j continue

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

# GO procedure, to start running

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

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 procedure, to stop running

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

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 procedure, to start drawing line

# param[in] none

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

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 procedure, to stop drawing line

# param[in] none

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

UNTRACK:

#backup

addi $sp,$sp,4

sw $at,0($sp)

#processing

li $at, LEAVETRACK # change LEAVETRACK port to 0

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

#restore

lw $at, 0($sp)

addi $sp,$sp,-4

jr $ra

nop

jr $ra

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

# ROTATE\_RIGHT procedure, to control robot to rotate

# param[in] nowHeading variable, store heading at present

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

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)

addi $sp,$sp,4

sw $ra,0($sp)

#processing

la $t7,LEAVETRACK

lb $t8, 0($t7)

li $t1, HEADING # change HEADING port

la $t2, nowHeading

lw $t3, 0($t2) #$t3 is heading at now

sw $t3, 0($t1) # to rotate robot

beqz $t8, restoreRotate

jal UNTRACK

jal TRACK

restoreRotate:

lw $ra, 0($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

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

# 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

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

get\_cod:

li $t1, IN\_ADRESS\_HEXA\_KEYBOARD

li $t2, OUT\_ADRESS\_HEXA\_KEYBOARD

scan\_row1:

li $t3, 0x81

sb $t3, 0($t1)

lbu $a0, 0($t2)

bnez $a0, get\_code\_in\_char

scan\_row2:

li $t3, 0x82

sb $t3, 0($t1)

lbu $a0, 0($t2)

bnez $a0, get\_code\_in\_char

scan\_row3:

li $t3, 0x84

sb $t3, 0($t1)

lbu $a0, 0($t2)

bnez $a0, get\_code\_in\_char

scan\_row4:

li $t3, 0x88

sb $t3, 0($t1)

lbu $a0, 0($t2)

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

#$s0 store code in char type

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, inputControlCode

la $s2, lengthControlCode

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

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 = inputControlCode + i

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

addi $s0, $zero, '\n' #add '\n' character to end of string

addi $s1, $s1, 1 #add '\n' character to end of string

sb $s0, 0($s1) #add '\n' character to end of string

addi $s3, $s3, 1

sw $s3, 0($s2) #update length of input control code

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

# Evaluate the return address of 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

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

# RESTORE the REG FILE from 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 # Return from exception

1. **Bài 10**

.data

Message: .asciiz "Ko chia dc cho 0"

.eqv IN\_ADRESS\_HEXA\_KEYBOARD 0xFFFF0012

.eqv OUT\_ADRESS\_HEXA\_KEYBOARD 0xFFFF0014

.eqv SEVENSEG\_RIGHT 0xFFFF0010 # dia chi den LED phai

.eqv SEVENSEG\_LEFT 0xFFFF0011 # dia chi den LED trai

.text

main:

li $t1, IN\_ADRESS\_HEXA\_KEYBOARD

li $t2, OUT\_ADRESS\_HEXA\_KEYBOARD

li $t3, 0x08 # duyet hang chua phim C, D, E, F

li $t4, 0x01 # duyet hang chua phim 0,1,2,3

li $t5, 0x02 # duyet hang chua phim 4,5,6,7

li $t6, 0x04 # duyet hang chua phim 8, 9, A, B

li $s0, 0x3f # Ma hien thi hang don vi

li $a1, 0x3f # Ma hien thi hang chuc

polling:

sb $t3, 0($t1 ) # Quét hàng C, D, E, F

lbu $t0, 0($t2) # $t0: ma phim quet duoc. $t0=0 neu ko có phím nào duoc an

bnez $t0, checkButton

nop

sb $t4, 0($t1 ) # Quét hàng 0,1,2,3

lbu $t0, 0($t2)

bnez $t0, checkButton

nop

sb $t5, 0($t1 ) # Quét hàng 4,5,6,7

lbu $t0, 0($t2)

bnez $t0, checkButton

nop

sb $t6, 0($t1 ) # Quét hàng 8, 9, A, B

lbu $t0, 0($t2)

checkButton:

beq $t0, 0x00, free # Ko an phím nào

beq $t0, 0x11, button0 # An phím 0

beq $t0, 0x21, button1 # An phím 1

beq $t0, 0x41, button2 # An phím 2

beq $t0, 0x81, button3 # An phím 3

beq $t0, 0x12, button4 # An phím 4

beq $t0, 0x22, button5 # An phím 5

beq $t0, 0x42, button6 # An phím 6

beq $t0, 0x82, button7 # An phím 7

beq $t0, 0x14, button8 # An phím 8

beq $t0, 0x24, button9 # An phím 9

beq $t0, 0x44, buttonA # An phím A

beq $t0, 0x84, buttonB # An phím B

beq $t0, 0x18, buttonC # An phím C

beq $t0, 0x28, buttonD # An phím D

beq $t0, 0x88, buttonF # An phím F

nop

button0:

li $a0, 0x3f # Nap ma hien thi so 0 vao $a0

li $s2, 0 # $s2: luu gia tri phim duoc an

j processButton

button1:

li $a0, 0x06 # Nap ma hien thi so 1 vao $a0

li $s2, 1

j processButton

button2:

li $a0, 0x5B # Nap ma hien thi so 2 vao $a0

li $s2, 2

j processButton

button3:

li $a0, 0x4f # Nap ma hien thi so 3 vao $a0

li $s2, 3

j processButton

button4:

li $a0, 0x66 # Nap ma hien thi so 4 vao $a0

li $s2, 4

j processButton

button5:

li $a0, 0x6D # Nap ma hien thi so 5 vao $a0

li $s2, 5

j processButton

button6:

li $a0, 0x7d # Nap ma hien thi so 6 vao $a0

li $s2, 6

j processButton

button7:

li $a0, 0x07 # Nap ma hien thi so 7 vao $a0

li $s2, 7

j processButton

button8:

li $a0, 0x7f # Nap ma hien thi so 8 vao $a0

li $s2, 8

j processButton

button9:

li $a0, 0x6f # Nap ma hien thi so 9 vao $a0

li $s2, 9

j processButton

buttonA:

li $t7, '+' # Gán $t7 phep tinh cong

j processMath # Nhay xuong phan xu ly khi phep tinh duoc bam

buttonB:

li $t7, '-' # Phép tru

j processMath

buttonC:

li $t7, '\*' # Phép nhân

j processMath

buttonD:

li $t7, '/' # Phép chia

j processMath

buttonF:

j result # Neu an dau =

free:

move $a0, $s0 # Chuyen Ma phim ve lai $a0

li $s1, 0 # Dat den bao = 0, phim da duoc tha ra

j showNumber # Chuyen den phan hien thi

processButton:

bnez $s1, showNumber # Kiem tra trang thai cua phim

move $a1, $s0 # Chuyen ma hien thi sang hang chuc

bnez $s4, secondNumber # Neu la toan hang thu 2 chuyen den xu ly toan hang 2

mul $s3, $s3, 10 # Xu ly toan hang 1

add $s3, $s3, $s2 # $s3 luu gia tri so hang thu nhat

j printInt

secondNumber:

mul $s5, $s5, 10 # Xu ly toan hang 2

add $s5, $s5, $s2 # $s5 luu gia tri so hang thu 2

printInt:

move $a0, $s2 #

li $v0, 1 #

syscall # In so vua nhan ra console

j isOldNumber

processMath:

li $a0, 0x3f # Dat lai ma hien thi so 0

li $a1, 0x3f #

bnez $s1, isOldNumber # Kiem tra co phim moi duoc bam chua

move $a0, $t7 #

li $v0, 11 #

syscall # Hien thi phep tinh len man hinh console

li $s2, 0 # Dat $s2 (gia tri phim bam hien tai) ve 0

li $s4, 1 # $s4 = 1, Xu ly Toan hang thu 2

isOldNumber:

li $s1, 1 # gan $s1 = 1 : Chua co phim moi duoc bam

showNumber:

move $s0, $a0 # Luu lai gia tri ma hien thi hang don vi

jal SHOW\_LED\_RIGHT

jal SHOW\_LED\_LEFT

delay:

li $a0, 100 # sleep 100ms

li $v0, 32

syscall

nop

back\_to\_polling:

j polling

result:

li $v0, 11

li $a0, '='

syscall #In ra dau bang ra man hinh van ban

beq $t7, '+', add # Neu la dau cong, chuyen den phan xu ly phep cong

beq $t7, '-', sub # Neu la dau tru, chuyen den phan xu ly phep tru

beq $t7, '\*', mul # Neu la dau nhan, chuyen den phan xu ly phep nhan

beq $t7, '/', div # Neu la dau chia, chuyen den phan xu ly phep chia

add: add $a0, $s3, $s5 # Cong hai so hang

j showResult # chuyen den phan hien thi ket qua

sub: sub $a0, $s3, $s5

j showResult

mul: mul $a0, $s3, $s5

j showResult

div: beq $s5, 0 , printMessage

div $a0, $s3, $s5

showResult:

li $v0, 1

syscall # In ket qua ra console (Ket qua tinh duoc luu o $a0)

div $s7, $a0, 10 # $s7 = ketqua($a0) chia 10

mfhi $t0 # Lay so du khi chia ket qua cho 10 (Hang don vi)

jal DATA\_FOR\_LED # Chuyen gia tri don vi cua ket qua sang ma hien thi cua LED

move $s0, $a0 # Luu ma hien thi vao $s0

div $s7, $s7, 10 # Chia $s7 cho 10

mfhi $t0 # Lay so du (Chinh la hang chuc cua ket qua)

jal DATA\_FOR\_LED # Chuyen gia tri hang chuc cua ket qua sang ma hien thi cua LED

move $a1, $a0 # Gan ma hien thi hang chuc cho $a1

jal SHOW\_LED\_RIGHT # show LED phai

jal SHOW\_LED\_LEFT # Show LED trai

li $v0, 10

syscall # Ket thuc chuong trinh

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

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

SHOW\_LED\_RIGHT:

sb $s0, SEVENSEG\_RIGHT # assign new value

jr $ra

SHOW\_LED\_LEFT:

sb $a1, SEVENSEG\_LEFT # assign new value

jr $ra

DATA\_FOR\_LED:

beq $t0, 0, setNumber0

beq $t0, 1, setNumber1

beq $t0, 2, setNumber2

beq $t0, 3, setNumber3

beq $t0, 4, setNumber4

beq $t0, 5, setNumber5

beq $t0, 6, setNumber6

beq $t0, 7, setNumber7

beq $t0, 8, setNumber8

beq $t0, 9, setNumber9

nop

setNumber0:

li $a0, 0x3f

j END\_\_F

setNumber1:

li $a0, 0x06

j END\_\_F

setNumber2:

li $a0, 0x5B

j END\_\_F

setNumber3:

li $a0, 0x4f

j END\_\_F

setNumber4:

li $a0, 0x66

j END\_\_F

setNumber5:

li $a0, 0x6D

j END\_\_F

setNumber6:

li $a0, 0x7d

j END\_\_F

setNumber7:

li $a0, 0x07

j END\_\_F

setNumber8:

li $a0, 0x7f

j END\_\_F

setNumber9:

li $a0, 0x6f

j END\_\_F

END\_\_F:

jr $ra

# Trong truong hop so chia = 0

printMessage:

li $v0, 4

la $a0, Message

syscall

li $v0, 10

syscall