TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG
ĐẠI HỌC BÁCH KHOA HÀ NỘI

———————— * ————————

FINAL-PROJECT

MÔN: KIẾN TRÚC MÁY TÍNH

ĐỀ TÀI: 1, 3

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

Sinh viên: Lê Minh Đức 20194511

Đinh Thị Thu Hà 20194543

Lớp: IT3280-130938

Hà Nội, tháng 7 năm 2022

MỤC LỤC

A: Lê Minh Đức

Bài 1: Curiosity Marsbot

I, Đề bài

1. Curiosity Marsbot

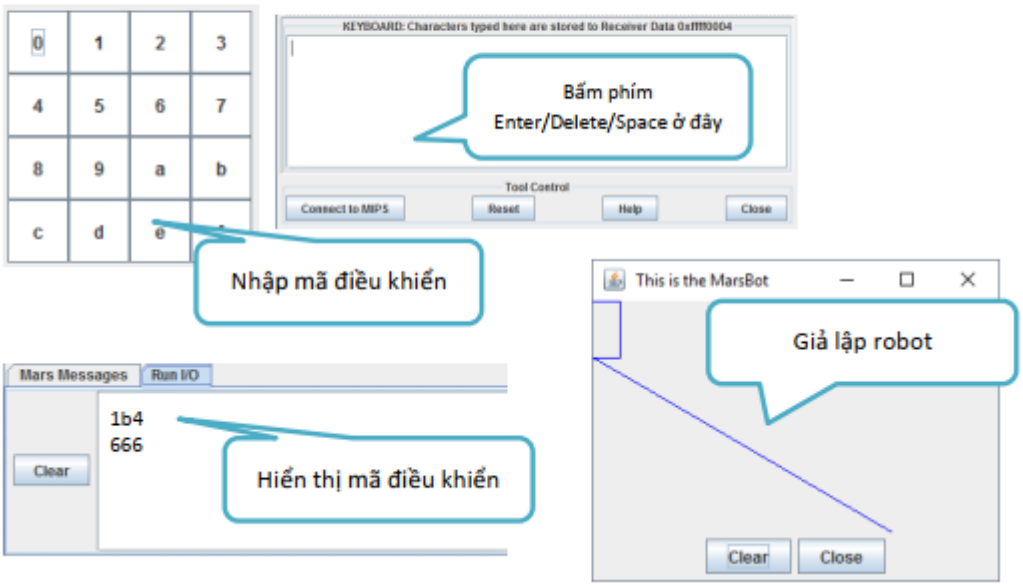
Xe tự hành Curiosity Marsbot chạy trên sao Hỏa, được vận hành từ xa bởi các lập trình viên trên Trái Đất. Bằng cách gửi đi các mã điều khiển từ một bàn phím ma trận, lập trình viên điều khiển quá trình di chuyển của Marbot như sau:

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.



II, Định hướng

1. Mỗi khi người dùng nhập 1 kí tự từ Digital Lab Sim sẽ tạo ra Interrup để lưu kí tự được nhập vào bộ nhớ, tạo nên đoạn code điều khiển
2. Kiểm tra liên tục xem kí tự Enter có được nhập ở Keyboard & Display MMIO Simulator hay không. Khi kí tự Enter được nhập, kiểm tra xem đoạn code điều khiển đó có hợp lệ hay không, nếu không sẽ thông báo code bị lỗi và chuyển tiếp sang bước 4. Nếu lỗi thì chuyển bước 3
3. Lần lượt kiểm tra xem code điều khiển được nhập vào có trùng khớp với các đoạn code điều khiển đã quy định sẵn không. Nếu không thì thông báo đoạn code đó bị lỗi. Ngược lại thực hiện theo thao tác đã quy định sẵn
4. In ra màn hình console code điều khiển đã nhập và xóa lưu trữ trong bộ nhớ.

III, Ý nghĩa của các hàm

1) storePath

Ý nghĩa: Lưu lại thông tin về đường đi của Marsbot vào mảng path.
Đầu vào: biến nowHeading và lengthPath
Mảng path lưu thông tin về đường đi của Marsbot, mỗi cạnh đường đi gồm 3 thông tin về các cạnh tọa độ x,y của điểm đầu tiên, hướng đi của cạnh đó.

2) goBack

Ý nghĩa: Điều khiển Marsbot đi ngược lại theo lộ trình nó đã đi và về điểm xuất phát.
Đầu vào: mảng path lưu thông tin đường đi, lengthPath lưu kích cỡ của mảng path theo byte
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

3) goRight va goLeft

Ý nghĩa: điều khiển Marsbot quay và di chuyển sang phải hoặc trái một góc 90 độ.
Đầu vào: biến nowHeading
Muốn di chuyển sang phải 90 độ ta chỉ cần tăng biến nowHeading lên 90 độ, đối với di chuyển sang trái là giảm biến nowHeading đi 90 độ

4) ROTATE

Ý nghĩa: quay Marsbot theo hướng có số độ lưu trong biến nowHeading
Đầu vào: biến nowHeading
Load biến nowHeading và lưu địa chỉ Heading (0xffff8010) để Marsbot chuyển hướng

5) TRACK, UNTRAK

Ý nghĩa: điều khiển Marbots bắt đầu để lại vết (TRACK) và kết thúc việc để lại vết (UNTRACK)
Load 1 vào địa chỉ LEAVETRACK (0xffff8020) nếu muốn để lại vết và load 0 vào địa chỉ đó nếu muốn kết thúc vết

6) GO,STOP

Ý nghĩa: 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.

7) pushErrorMess

Ý nghĩa: hiện thông báo dialog khi người dùng nhập code điều khiển không đúng.
Sử dụng các hàm syscall 4,55.

8) isEqualString

Ý nghĩa: so sánh 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 kí tự trong 2 xâu này. Nếu 2 xâu bằng nhau thì thanh ghi $t0 có giá trị là 1, ngược lại là 0

9) removeControlCode

Ý nghĩa: xóa xâu inputControlCode (nơi lưu trữ ode điều khiển khi nhập vào). Lần lượt gán các kí tự trong xâu bằng ‘\0’.

10) Các thao tác trong xử lý interrupt

Lần lượt quét các hàng của Digital Lab Sim để xem phím nào được bấm tiếp. Tiếp đó dựa vào mã được trả về ghi kí 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
.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 "Wrong control code!"
#-------------------------------------------------------------------------------

        inputControlCode: .space 50
        lengthControlCode: .word 0
        nowHeading: .word 0
#-----------------------------------------------------
# duong di cua masbot duoc luu tru vao mang path
# moi 1 canh duoc luu tru duoi dang 1 structure
# 1 structure co dang {x, y, z}
# trong do:     x, y la toa do diem dau tien cua canh
#               z la huong cua canh do
# mac dinh:     structure dau tien se la {0,0,0}
# do dai duong di ngay khi bat dau la 12 bytes (3x 4byte)
#-----------------------------------------------------
        path: .space 600
        lengthPath: .word 12    #bytes

#===============================================================================
#===============================================================================
.text
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 isEqualString
                beq $t0, 1, go
```

```
                la $s3, STOP_CODE
                jal isEqualString
                beq $t0, 1, stop


                la $s3, GO_LEFT_CODE
                jal isEqualString
                beq $t0, 1, goLeft

                la $s3, GO_RIGHT_CODE
                jal isEqualString
                beq $t0, 1, goRight

                la $s3, TRACK_CODE
                jal isEqualString
                beq $t0, 1, track


                la $s3, UNTRACK_CODE
                jal isEqualString
                beq $t0, 1, untrack


                la $s3, GO_BACK_CODE
                jal isEqualString
                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
# param[in] none
#-------------------------------------------------------------
track:  jal TRACK
        j printControlCode
#-------------------------------------------------------------
# untrack procedure, control marsbot to untrack and print control code
# param[in] none
```

```
#-----------------------------------------------------------
untrack: jal UNTRACK
        j printControlCode
#-----------------------------------------------------------
# go procedure, control marsbot to go and print control code
# param[in] none
#-----------------------------------------------------------
go:
        jal GO
        j printControlCode
#-----------------------------------------------------------
# stop procedure, control marsbot to stop and print control code
# param[in] none
#-----------------------------------------------------------
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
#---------------------------------------------------------------
# isEqualString 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
#---------------------------------------------------------------
isEqualString:
#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
#---------------------------------------------------------------
```

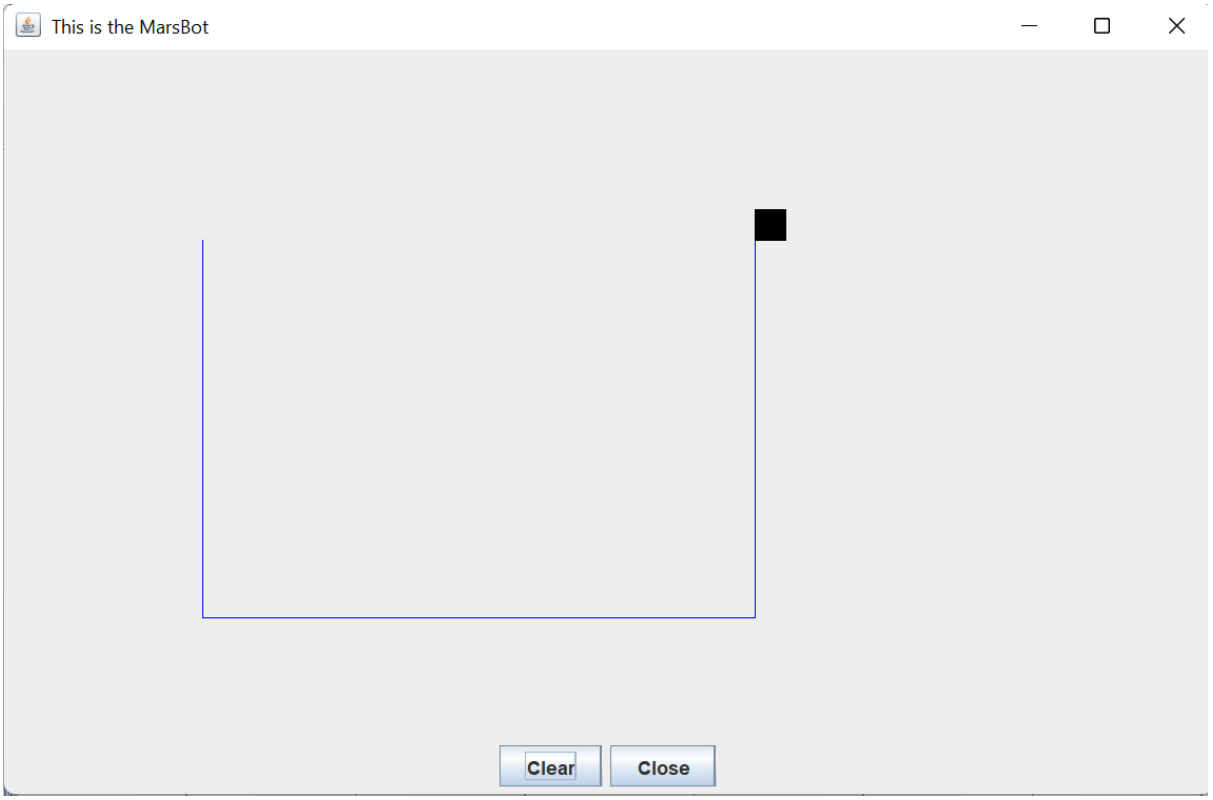
```
# GO procedure, to start running
# param[in] none
#-------------------------------------------------------------
GO:
#backup
        addi $sp,$sp,4
        sw $at,0($sp)
        addi $sp,$sp,4
        sw $k0,0($sp)
#processing
        li $at, MOVING          #change MOVING port
        addi $k0, $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
# param[in] none
#-------------------------------------------------------------
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:
#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)
#processing
        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
#restore
        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
```

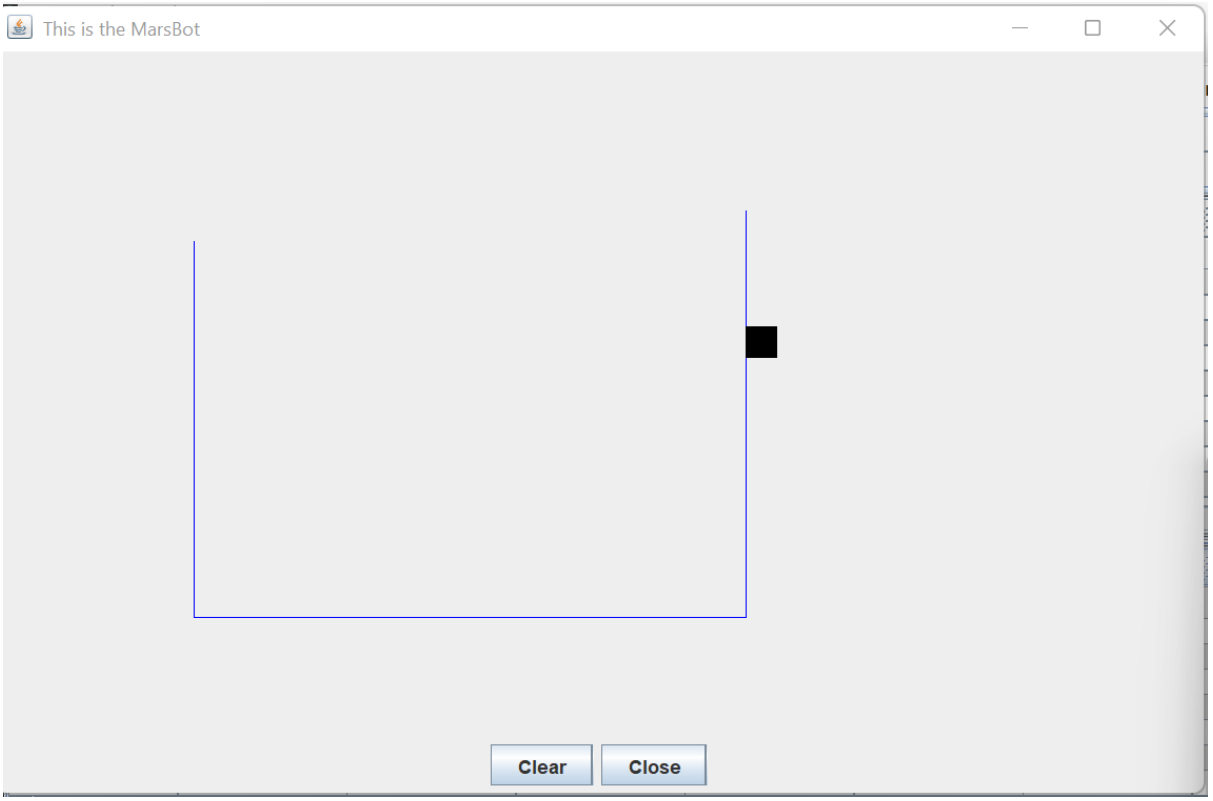
V, Kết quả chạy mô phỏng

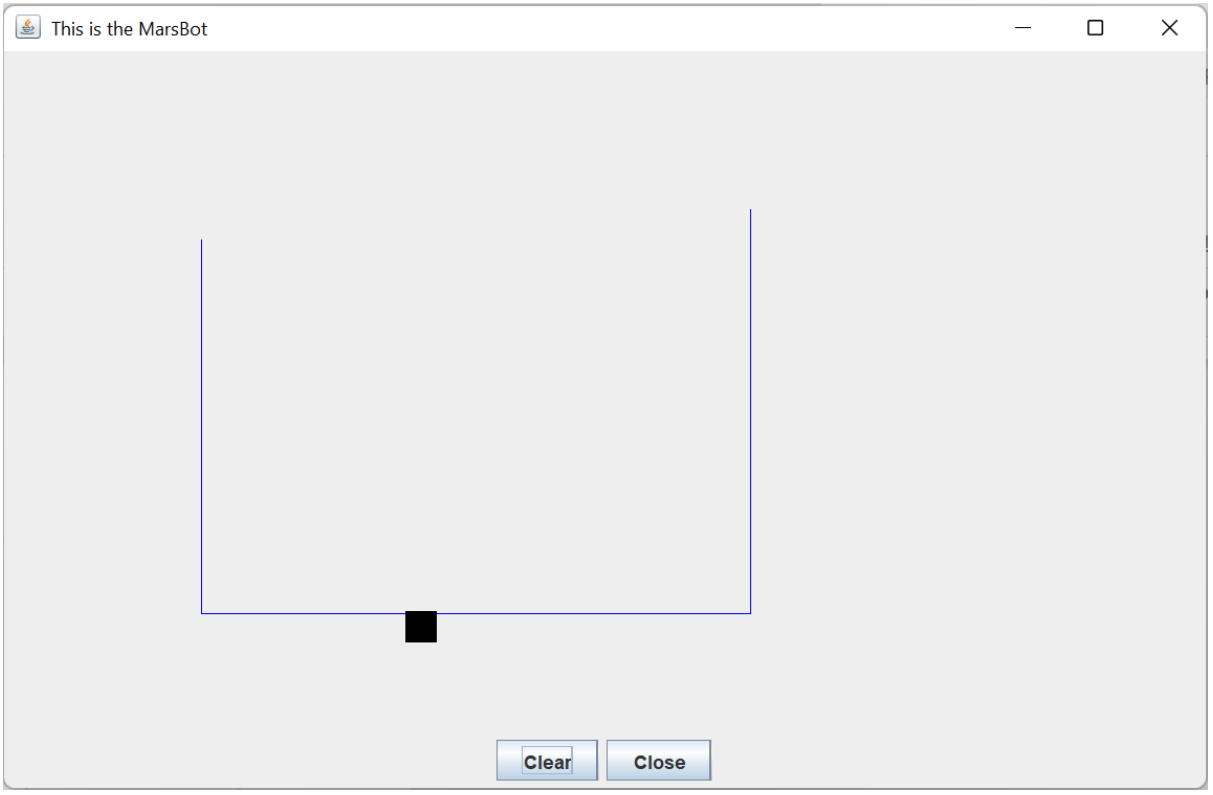
Hình ảnh Marsbot khi chạy các câu lệnh



Các câu lệnh được hiển thị ra màn hình Run I/O

Marsbot đi ngược lại vị trí ban đầu





Hình ảnh báo lỗi khi viết sai câu lệnh

B: Đinh Thị Thu Hà

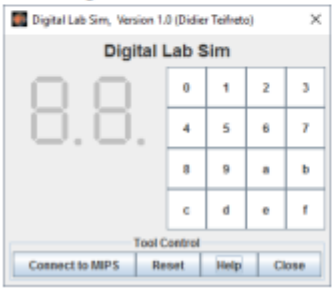
Bài 3: Kiểm tra tốc độ và độ chính xác khi gõ văn bản

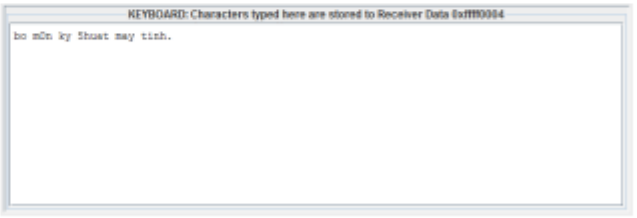
I, Đề bài

3. Kiểm tra tốc độ và độ chính xác khi gõ văn bản

Thực hiện chương trình đo tốc độ gõ bàn phím và hiển thị kết quả bằng 2 đèn led 7 đoạn. Nguyên tắc:

- Cho một đoạn văn bản mẫu, cố định sẵn trong mã nguồn. Ví dụ *"bo mon ky thuat may tinh"*
- Sử dụng bộ định thời Timer (trong bộ giả lập Digital Lab Sim) để tạo ra khoảng thời gian để đo. Đây là thời gian giữa 2 lần ngắt, chu kì ngắt.
- Người dùng nhập các kí tự từ bàn phím. Ví dụ nhập *"bo mOn ky 5huat may tinh"*. Chương trình cần phải đếm số kí tự đúng (trong ví dụ trên thì người dùng gõ sai chữ **O** và **5**) mà người dùng đã gõ và hiển thị lên các đèn led.
- Chương trình đồng thời cần tính được tốc độ gõ: thời gian hoàn thành và số từ trên một đơn vị thời gian.





II, Phân tích cách làm và thuật toán

- Sử dụng 1 vòng lặp vô hạn.
- Trong vòng lặp có kiểm tra giá trị tại địa chỉ KEY_READY nếu bằng 1 thì thực hiện tạo ngắt bằng teqi.
- Đồng thời chương trình cũng cho phép ngắt bằng bộ đếm time counter(timer)
- Khi đã bắt được exception và con trỏ $pc nhảy đến vùng phục vụ ngắt .ktext.
- Bên trong vùng .ktext ta sẽ lấy giá trị bên trong thanh ghi Coproc0.cause($13) để kiểm tra đây là loại ngắt nào.
- Trong trường hợp lệnh ngắt được thực hiện bởi teqi(tạo ra khi nhận được ký tự từ bàn phím).
- Ta kiểm tra xem ký tự thứ i của string đã cho có phải là ký tự kết thúc hay không('\0'), nếu đúng thì kết thúc chương trình, hiển thị ra số ký tự đúng lên Digital Lab Sim và in ra thời gian hoàn thành, tốc độ gõ lên màn hình.
- Nếu chương trình chưa kết thúc, ta so sánh ký tự vừa nhập với ký tự string[i] nếu bằng nhau -> tăng biến đếm số ký tự đúng lên 1.
- Tiếp tục kiểm tra xem ký tự vừa nhập vào == ' ' && ký nhập vào trước đó(prv) != ' ' nếu đúng thì tăng biến đếm số từ đã nhập lên 1.
- Sau đó tăng số ký tự nhập vào trong 1s lên 1, cập nhật giá trị của prv bằng ký tự vừa nhập vào và chuyển con trỏ của string lên 1 để phục vụ cho lần sau

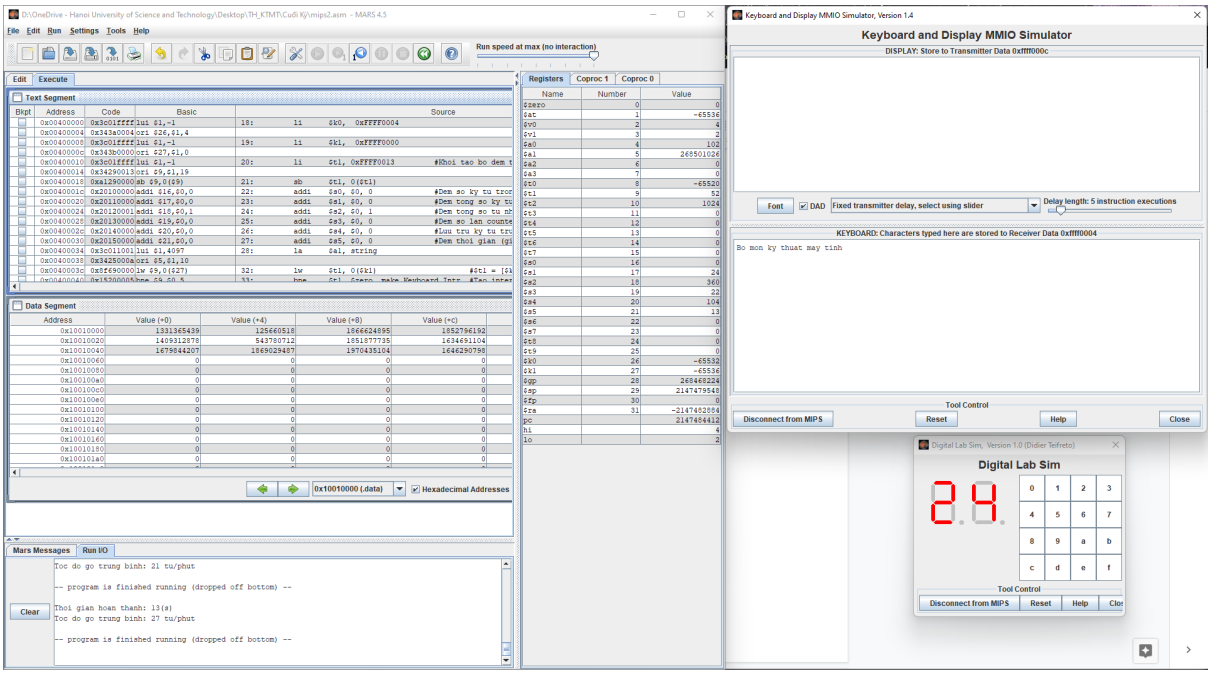
- Trong trường hợp lệnh ngắt được thực hiện bởi bộ đếm time counter(timer)
- Kiểm tra xem số lần tạo lệnh ngắt của timer đã đủ chưa (1s), nếu chưa đủ thì tăng biến đếm lên, nếu đã đủ thì hiển thị số ký tự đã gõ trong 1s lên Digital Lab Sim và khởi tạo lại biến đếm ký tự trong 1s, đồng thời tăng biến đếm thời gian hoàn thành nhập lên 1s.
- Sau khi hoàn thành các câu lệnh trong vùng .ktext.
- Thực hiện các hàm cần thiết để thiết lập lại các thông số để đón nhận lần ngắt tiếp theo.

III, Mã nguồn

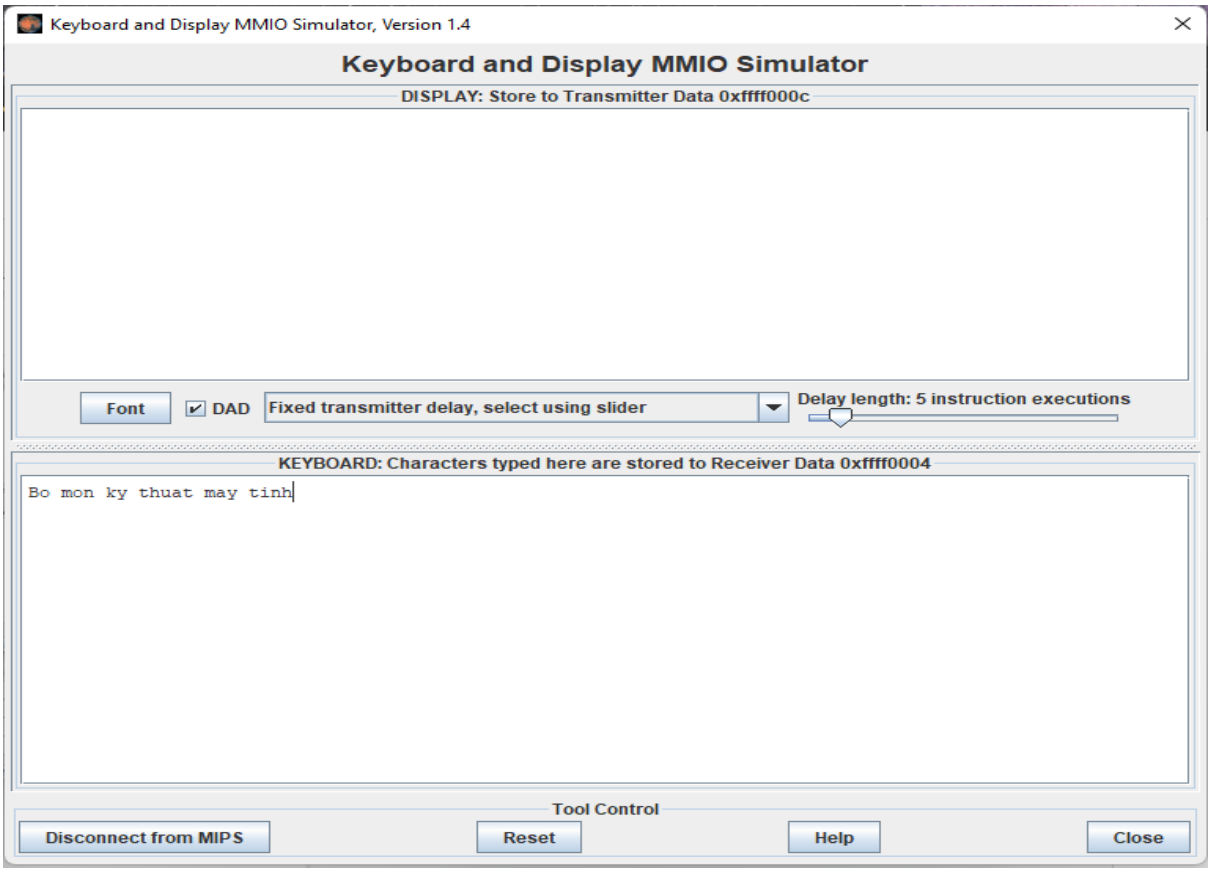
```
.eqv SEVENSEG_LEFT 0xFFFF0011           #Dia chi cua den led 7 doan trai
.eqv SEVENSEG_RIGHT 0xFFFF0010          #Dia chi cua den led 7 doan phai
.eqv IN_ADDRESS_HEXA_KEYBOARD       0xFFFF0012
.eqv MASK_CAUSE_COUNTER 0x00000400      #Bit 10: Counter interrupt
.eqv COUNTER 0xFFFF0013                 #Time Counter
.eqv KEY_CODE   0xFFFF0004              #ASCII code from keyboard, 1 byte
.eqv KEY_READY  0xFFFF0000              #=1 if has a new keycode?
.data
mang_so: .byte  63, 6,  91, 79, 102, 109 ,125, 7, 127, 111      #tu 0 den 9
string: .asciiz "Bo mon ky thuat may tinh"
message1:       .asciiz "Thoi gian hoan thanh: "
message2:       .asciiz "(s) \nToc do go trung binh: "
message3:       .asciiz " tu/phut\n"
#===============================================================
# MAIN Procsciiz ciiz edure
#===============================================================
.text   #bien toan cuc : k0, k1, s0, s1, s2, s3, s4, s5, a1
        li      $k0,  KEY_CODE
        li      $k1,  KEY_READY
        li      $t1, COUNTER            #Khoi tao bo dem timer
        sb      $t1, 0($t1)
        addi    $s0, $0, 0              #Dem so ky tu trong 1s
        addi    $s1, $0, 0              #Dem tong so ky tu dung
        addi    $s2, $0, 1              #Dem tong so tu nhap vao
        addi    $s3, $0, 0              #Dem so lan counter_intr
        addi    $s4, $0, 0              #Luu tru ky tu truoc do
        addi    $s5, $0, 0              #Dem thoi gian (giay)
        la      $a1, string
#===============================================================
#VONG LAP VO HAN DE DOI INTERRUPT
loop:
        lw      $t1, 0($k1)                     #$t1 = [$k1] = KEY_READY
        bne     $t1, $zero, make_Keyboard_Intr  #Tao interrupt khi nhan duoc ky tu tu ban phim
        addi    $v0, $0, 32
        li      $a0, 5
        syscall
        b       loop                                    #So lenh trong 1 vong lap = 6 => cu lap 5 lan thi tao 1 counter interrupt
        nop
#===============================================================
make_Keyboard_Intr:
        teqi    $t1, 1
        b       loop                            #Quay lai vong lap de cho doi su kien interrupt tiep theo
        nop
end_Main:
```

```
#-----------------------------------------------------------------
#PHAN PHUC VU NGAT
#-----------------------------------------------------------------
.ktext 0x80000180

dis_int:li      $t1, COUNTER                    #BUG: must disable with Time Counter
        sb      $zero, 0($t1)
#-----------------------------------------------------------------
#LAY GIA TRI CUA THANH GHI C0.cause DE KIEM TRA LOAI INTERRUPT
get_Caus:mfc0   $t1, $13                        #$t1 = Coproc0.cause
isCount:li      $t2, MASK_CAUSE_COUNTER         #if Cause value confirm Counter..
         and    $at, $t1,$t2
         bne    $at,$t2, keyboard_Intr
#-----------------------------------------------------------------
#NGAT DO BO DEM COUNTER
counter_Intr:
        blt     $s3, 40, continue               #Neu so lap ngat do counter = 40 : da du 1s -> khoi tao lai $s3, chieu toc do go ra DLS, tang bien dem thoi gian len 1
        jal     hien_thi
        addi    $s3, $0, 0                      #Khoi tao lai $s3
        addi    $s5, $s5, 1                     #Tang bien dem thoi gian(s)
        j       en_int
        nop
continue:
        addi    $s3, $s3, 1                     #Neu chua du 1s thi tang bien dem so lan ngat
        j       en_int
        nop
keyboard_Intr:
#-----------------------------------------------------------------
#NGAT DO BAN PHIM
check_Matching:                                 #Kiem tra ky tu nhap vao
        lb      $t0, 0($a1)                     #Lay ki tu thu i trong mang da cho
        beq     $t0, $0, end_Program            #Dung chuong trinh neu gap ki tu '\0'
        lb      $t1, 0($k0)                     #Lay ki tu nhap vao tu ban phim
        beq     $t1, $0, en_int                 #Loi
        bne     $t0, $t1, check_Space           #Neu ki tu nhap vao va ki tu thu i trong mang da cho bang nhau -> $s1++(dem so ki tu dung)
        nop
        addi    $s1, $s1, 1                     #Tang bien dem so ky tu dung
check_Space:                                    #Kiem tra ki tu nhap vao co phai la ' ' hay ko
        bne     $t1, ' ', end_Process           #Neu ky tu nhap vao == ' ' && ky tu truoc do != ' ' thi dem so tu da nhap
        nop
        beq     $s4, ' ', end_Process
        nop
        addi    $s2, $s2, 1                     #Tang bien dem so tu da nhap
end_Process:
        addi    $s0, $s0, 1                     #Tang so ky tu trong 1s len 1
        addi    $s4, $t1, 0                     #Cap nhat lai thanh ghi chua ky tu nhap vao ban phim truoc do
        addi    $a1, $a1, 1                     #Tang con tro len 1 <=> string+1
#-----------------------------------------------------------------

en_int:
        li      $t1, COUNTER
        sb      $t1, 0($t1)
        mtc0    $zero, $13                      #Must clear cause reg
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
return: eret                                    #Return from exception
#-----------------------------------------------------------------
# CHIEU RA MAN HINH DIGITAL LAB SIM GIA TRI CUA $s0
#-----------------------------------------------------------------
hien_thi:
        addi    $sp, $sp, -4
        sw      $ra, ($sp)
        addi    $t0, $0, 10
        div     $s0, $t0
        mflo    $v1                             #Lay so hang chuc
        mfhi    $v0                             #Lay so hang don vi
        la      $a0, mang_so
        add     $a0, $a0, $v1
        lb      $a0, 0($a0)                     #Set value for segments
        jal     SHOW_7SEG_LEFT                  #Show
        la      $a0, mang_so
        add     $a0, $a0, $v0
        lb      $a0, 0($a0)                     #Set value for segments
        jal     SHOW_7SEG_RIGHT                 #Show
        addi    $s0, $0, 0                      #Sau khi chieu ra man hinh thi khoi tao lai bien dem
        lw      $ra, ($sp)
        addi    $sp, $sp, 4
        jr      $ra
SHOW_7SEG_LEFT:
        li      $t0, SEVENSEG_LEFT              #Assign port's address
        sb      $a0, 0($t0)                     #Assign new value
        jr      $ra
SHOW_7SEG_RIGHT:
        li      $t0, SEVENSEG_RIGHT             #Assign port's address
        sb      $a0, 0($t0)                     #Assign new value
        jr      $ra
        nop
#-----------------------------------------------------------------
# KET THUC CHUONG TRINH VA HIEN THI SO KY TU DUNG
#-----------------------------------------------------------------

end_Program:
        addi    $v0, $0, 4
        la      $a0, message1
        syscall
        addi    $v0, $0, 1
        addi    $a0, $s5, 0
        syscall
        addi    $v0, $0, 4
        la      $a0, message2
        syscall
        addi    $v0, $0, 1
        addi    $a0, $0, 60
        mult    $s2, $a0
        mflo    $s2
        div     $s2, $s5
        mflo    $a0
        syscall
        addi    $v0, $0, 4
        la      $a0, message3
        syscall
        addi    $s0, $s1, 0
        jal     hien_thi
```

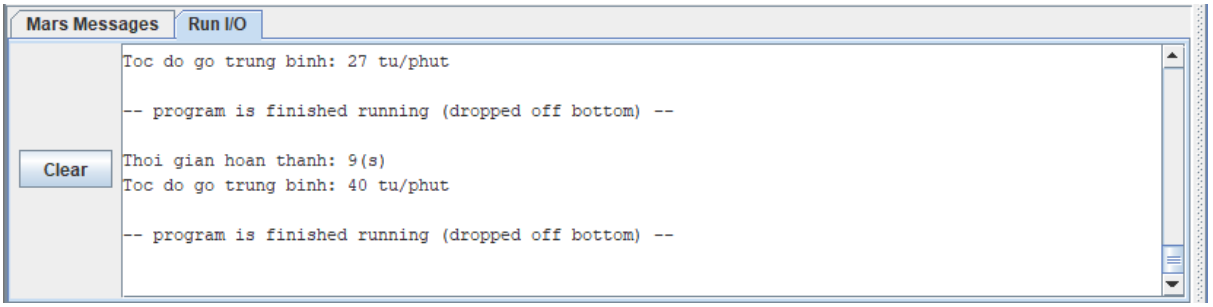
IV. Kết quả chạy mô phỏng

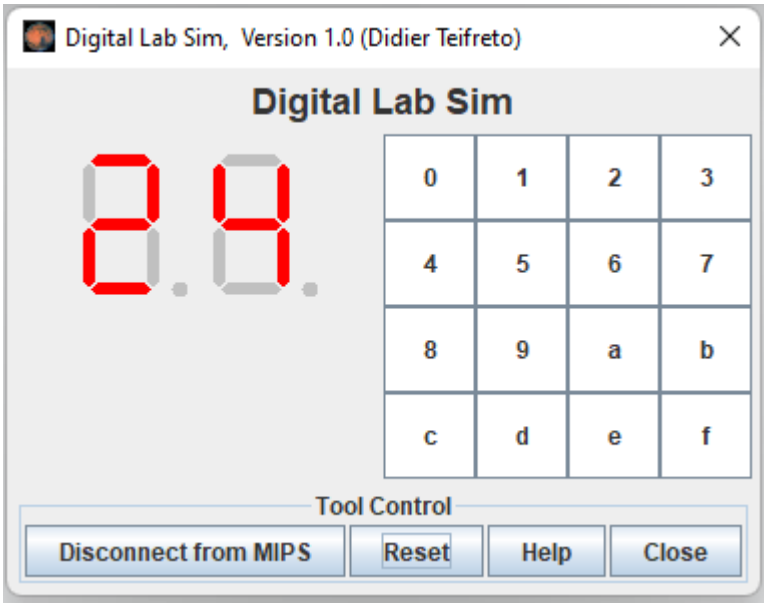


- Khi nhập vào một xâu “Bo mon ky thuat may tinh”

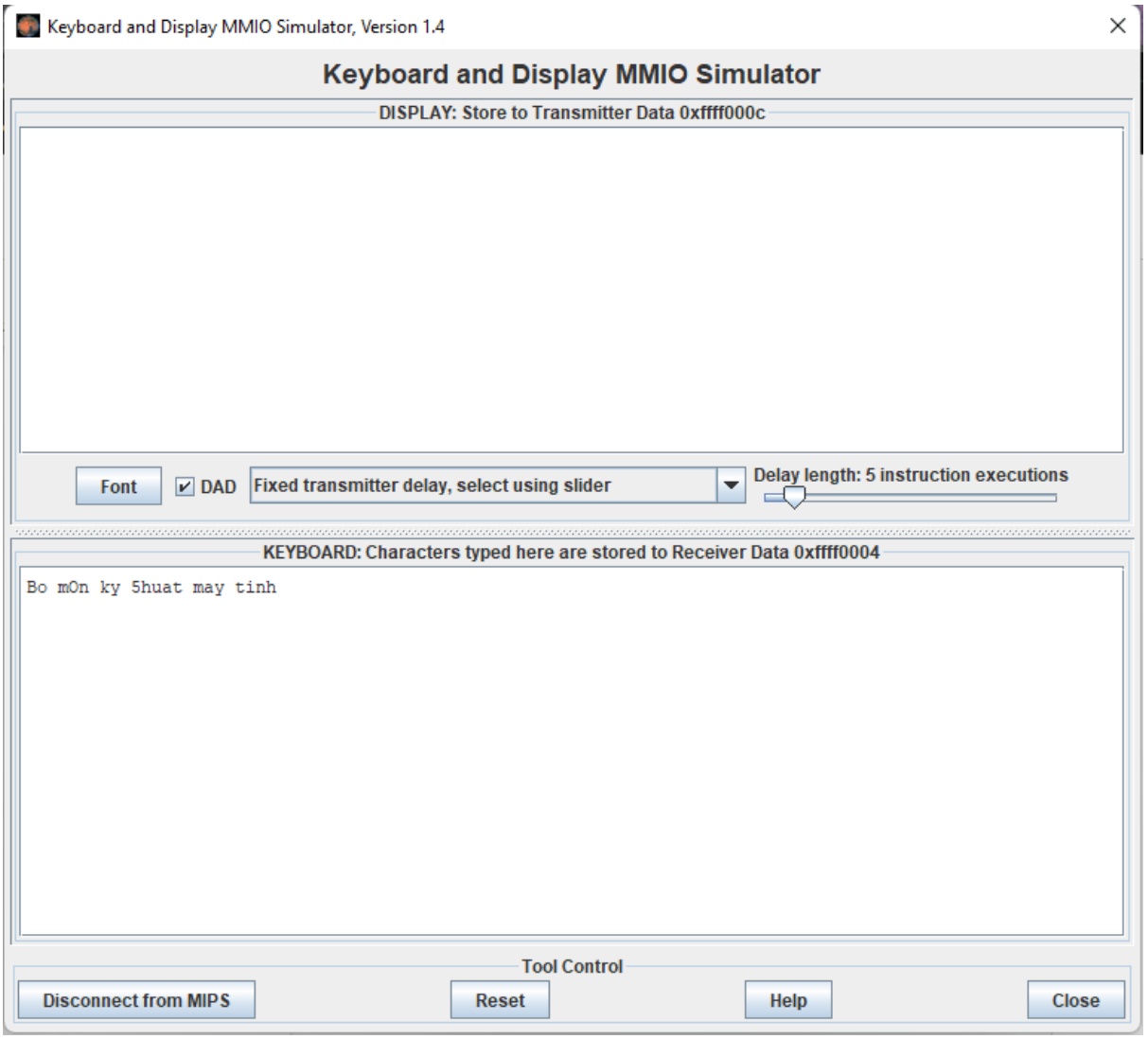


- Kết quả

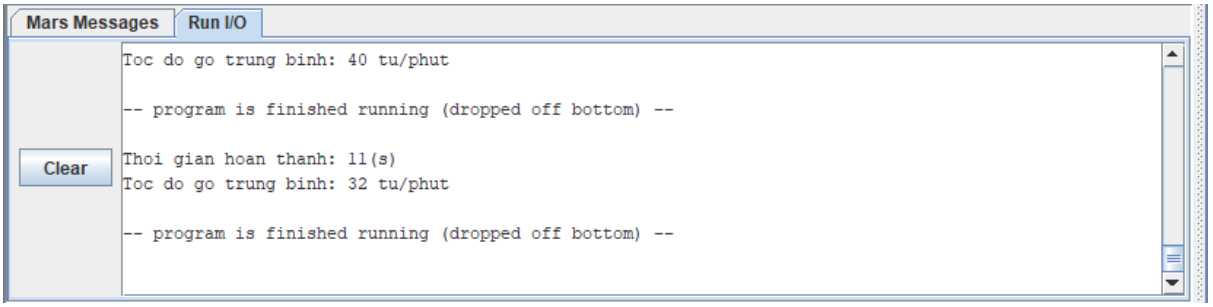

Mars Messages
Run I/O
Clear
Toc do go trung binh: 27 tu/phut
-- program is finished running (dropped off bottom) --
Thoi gian hoan thanh: 9(s)
Toc do go trung binh: 40 tu/phut
-- program is finished running (dropped off bottom) --

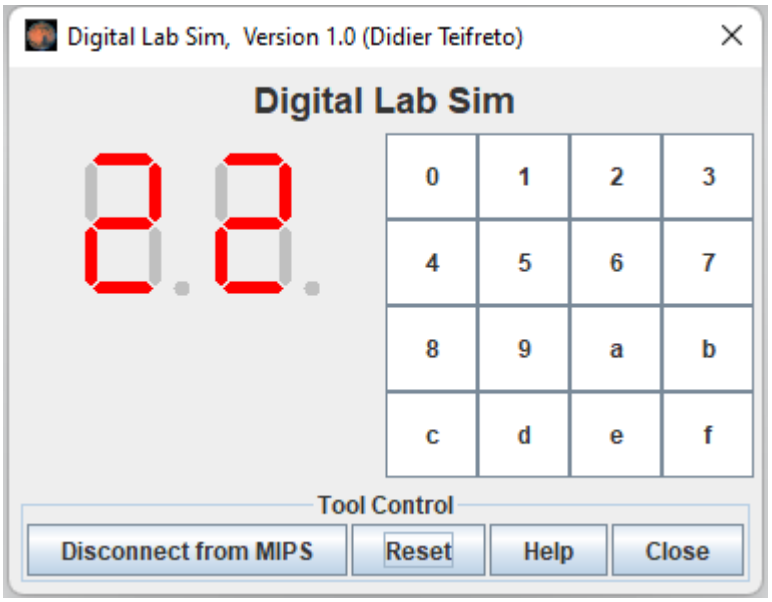

Digital Lab Sim, Version 1.0 (Didier Teifreto)
Digital Lab Sim
2.8.
0
1
2
3
4
5
6
7
8
9
a
b
c
d
e
f
Tool Control
Disconnect from MIPS
Reset
Help
Close

- Khi nhập vào một xâu “Bo mOn ky 5huat may tinh”



- Kết quả


Mars Messages
Run I/O
Clear
Toc do go trung binh: 40 tu/phut
-- program is finished running (dropped off bottom) --
Thoi gian hoan thanh: 11(s)
Toc do go trung binh: 32 tu/phut
-- program is finished running (dropped off bottom) --


Digital Lab Sim, Version 1.0 (Didier Teifreto)
Digital Lab Sim
0 1 2 3
4 5 6 7
8 9 a b
c d e f
Tool Control
Disconnect from MIPS
Reset
Help
Close