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

**********


# BÁO CÁO MINI PROJECT

## THỰC HÀNH KIẾN TRÚC MÁY TÍNH

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

**Lớp:** Thực hành KTMT – 130937

**Nhóm 3:** Nguyễn Thị Diệu Linh 20205094

Lê Thị Nguyệt 20205109

Hà Nội, ngày 22 Tháng 07 năm 2022

# Mục lục

# 1. Project 01: Curiosity Mars Bot

*Sinh viên thực hiện*: Lê Thị Nguyệt – 20205109

## 1.1. Đề bài

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 với phương chuyển động gần nhất |
| 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.<br>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, Curioisity 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 |

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.

## 1.2. Cách thực hiện

Bước 1: Nhấn các phím ở Digital Data Lab Sim, lưu từng ký tự vào một xâu ký tự.

Bước 2: Nhập Enter/Delete/Space ở Keyboard & Display MMIO Simulator:

- Nếu nhập Enter: kiểm tra mã điều khiển nhập vào có hợp lệ về độ dài và có khớp với một trong các mã đã quy ước không.

- Nếu nhập Delete: xóa mã điều khiển đang nhập.

-Nếu nhập Space: copy lại mã đã thực hiện trước đó được lưu trong code_history và lưu vào mảng cmdCode

Bước 3: Thực hiện mã điều khiển được nhập vào.

Bước 4: Nếu mã được nhập vào là mã để rẽ trái (phải), lưu tọa độ x, y và góc trước khi rẽ vào lần lượt 03 mảng số nguyên (x_history, y_history, a_history)

Bước 5: In ra màn hình mã điều khiển được nhập vào, và lập lại Bước 1.

## 1.3. Các chương trình con

### 1.3.1. Hàm main

Các nhãn và công việc tương ứng của từng nhãn trong hàm main như sau:

- `setStartHeading`: set góc đầu tiên của Marsbot là góc 90 độ
- `printErrorMsg`: In ra thông báo lỗi
- `printCmd`: In ra mã điều khiển vừa nhập vào
- `resetInput`: xóa mã điều khiển đã nhập để chuẩn bị cho mã tiếp theo
- `waitForKey`: chờ phím được nhấn từ Digital Lab Sim
- `readKey`: Đọc ký tự được nhập vào từ Keyboard & Display MMIO Simulator
- `checkCmd`: kiểm tra mã điều khiển có hợp lệ về độ dài và khớp với một trong các mã đã được quy ước
- `go`, `stop`, `turnLeft`, `turnRight`, `track`, `untrack`, `goBackward`: thực thi mã điều khiển

### 1.3.2. Các hàm cho Marsbot

Các hàm và chức năng tương ứng của từng hàm như sau:

- `GO`, `STOP`: điều khiển Marsbot bắt đầu chuyển động (`GO`) hoặc dừng lại (`STOP`); lưu trạng thái đang chuyển động hay không vào `isGoing`
- `ROTATE`: điều khiển Marsbot quay theo góc lưu ở `a_current - TRACK`, `UNTRACK`: điều khiển Marsbot bắt đầu để lại vết (`TRACK`) hoặc dừng để lại vết (`UNTRACK`); lưu trạng thái đang ghi vết hay không vào `isTracking`
- `saveHistory`: lưu tọa độ x, y và góc hiện tại trước khi Marsbot thực hiện lệnh `ROTATE`

### 1.3.3. Các hàm để xử lý xâu

Các hàm và chức năng tương ứng của từng hàm như sau:

- `strcmp`: so sánh xâu ở `$s3` với mã điều khiển vừa nhập (`cmdCode`), trả về giá trị boolean ở `$t0`
- `strClear`: xóa mã điều khiển vừa nhập (`cmdCode`)

## 1.4. Mã nguồn

```
# eqv for Digital Lab Sim
.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

# eqv for 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

# eqv for Mars bot
.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: 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
      x_history:        .word 0 : 16
      y_history:        .word 0 : 16
      a_history:        .word 0 : 16
      l_history:        .word 4           # history length

      a_current:        .word 0           # current alpha

      isGoing:          .word 0
      isTracking:       .word 0

      cmdCode:          .space 8    # input command code
      code_history:           .space 8    # history code
      cmdLen:                 .word 0           # input command length

      MOVE_CODE:        .asciiz "1b4"     # command code
      STOP_CODE:        .asciiz "c68"
      TURN_LEFT_CODE:         .asciiz "444"
      TURN_RIGHT_CODE:  .asciiz "666"
      TRACK_CODE:       .asciiz "dad"
      UNTRACK_CODE:           .asciiz "cbc"
      GOBACKWARD_CODE:  .asciiz "999"

      invalidCmd_msg:         .asciiz "Invalid command code: "

#-------------------------------------------------------------------------
-------
.text
main: li $k0, KEY_CODE
      li $k1, KEY_READY

      li $t1, IN_ADRESS_HEXA_KEYBOARD     # enable the interrupt of
Digital Lab Sim
      li $t3, 0x80                  # bit 7 = 1 to enable
      sb $t3, 0($t1)

setStartHeading:
```

```
      lw    $t7, l_history          # l_history += 4
      addi  $t7, $zero, 4           # to save x = 0; y = 0; a = 90
      sw    $t7, l_history

      li    $t7, 90
      sw    $t7, a_current    # a_current = 90 -> heading down
      jal   ROTATE
      nop

      sw    $t7, a_history + 4      # a_history[1] = 90
                            # x_history[1] =0
                            # y_history[1] = 0
      j     waitForKey

printErrorMsg:
      li    $v0, 4
      la    $a0, invalidCmd_msg
      syscall

printCmd:
      li    $v0, 4
      la    $a0, cmdCode
      syscall

resetInput:
      jal   strClear
      nop

waitForKey:
      lw    $t5, 0($k1)             # $t5 = [$k1] = KEY_READY
      beq   $t5, $zero, waitForKey        # if $t5 == 0 -> Polling
      nop
      beq   $t5, $zero, waitForKey
readKey:
      lw    $t6, 0($k0)             # $t6 = [$k0] = KEY_CODE
      beq   $t6, 0x7f , resetInput        # if $t6 == 'DEL' -> reset
input
      beq   $t6,0x20,copy_code_history    # if $t6 =='SPACE' -> recontrol
      bne   $t6, 0x0a, waitForKey         # if $t6 != '\n' -> Polling
      nop
      bne   $t6, 0x0a, waitForKey
checkCmd:
      lw    $s2, cmdLen             # cmdLen != 3 -> invalid cmd
      bne   $s2, 3, printErrorMsg

      la    $s3, MOVE_CODE
      jal   strcmp
      beq   $t0, 1, case_go

      la    $s3, STOP_CODE
      jal   strcmp
      beq   $t0, 1, case_stop
```

```
      la    $s3, TURN_LEFT_CODE
      jal   strcmp
      beq   $t0, 1, case_turnLeft

      la    $s3, TURN_RIGHT_CODE
      jal   strcmp
      beq   $t0, 1, case_turnRight

      la    $s3, TRACK_CODE
      jal   strcmp
      beq   $t0, 1, case_track

      la    $s3, UNTRACK_CODE
      jal   strcmp
      beq   $t0, 1, case_untrack

      la    $s3, GOBACKWARD_CODE
      jal   strcmp
      beq   $t0, 1, goBackward
      nop

      j     printErrorMsg

      switch:
      case_go:    jal save_code
                  j     go
      case_stop:  jal save_code
                  j     stop
      case_turnLeft:    jal save_code
                  j     turnLeft
      case_turnRight:   jal save_code
                  j     turnRight
      case_track: jal save_code
                  j     track
      case_untrack:     jal save_code
                  j     untrack
      case_goBackWard:jal save_code
                  j     goBackward
      default:
      # j   printCmdCode

#-------------------------------------------------------------
go:   jal   GO
      j     printCmd

#-------------------------------------------------------------
stop:       jal   STOP
      j     printCmd

#-------------------------------------------------------------
track:      jal   TRACK
      j     printCmd
```

```
#-------------------------------------------------------------
untrack:    jal   UNTRACK
      j     printCmd

#-------------------------------------------------------------
turnRight:
      lw    $t7, isGoing
      lw    $s0, isTracking

      jal   STOP
      nop
      jal   UNTRACK
      nop

      la    $s5, a_current
      lw    $s6, 0($s5)       # $s6 is heading at now
      addi  $s6, $s6, 90            # increase alpha by 90*
      sw    $s6, 0($s5)       # update a_current

      jal   saveHistory
      jal   ROTATE

      beqz  $s0, noTrack1
      nop
      jal   TRACK
      noTrack1:   nop

      beqz  $t7, noGo1
      nop
      jal   GO
      noGo1:      nop

      j     printCmd

#-------------------------------------------------------------
turnLeft:
      lw    $t7, isGoing
      lw    $s0, isTracking

      jal   STOP
      nop
      jal   UNTRACK
      nop

      la    $s5, a_current
      lw    $s6, 0($s5)       # $s6 is heading at now
      addi  $s6, $s6, -90           # decrease alpha by 90*
      sw    $s6, 0($s5)       # update a_current

      jal   saveHistory
      jal   ROTATE
```

```
      beqz  $s0, noTrack2
      nop
      jal   TRACK
      noTrack2:   nop

      beqz  $t7, noGo2
      nop
      jal   GO
      noGo2:      nop

      j     printCmd

#-----------------------------------------------------------
goBackward:
      li    $t7, IN_ADRESS_HEXA_KEYBOARD  # Disable interrupts when going
backward
      sb    $zero, 0($t7)

      lw    $s5, l_history                # $s5 = length history
      jal   UNTRACK
      jal   GO

goBackward_turn:

      lw    $s6, a_history($s5)           # $s6 = a_history[l_history]
      addi  $s6, $s6, 180                 # $s6 = the reverse direction
of alpha
      sw    $s6, a_current
      jal   ROTATE
      nop

goBackward_toTurningPoint:
      lw    $t9, x_history($s5)           # $t9 = x_history[i]
      get_x:
      li    $t8, WHEREX       # $t8 = x_current
      lw    $t8, 0($t8)
      bne   $t8, $t9, get_x         # x_current == x_history[i]
      nop                     # -> get y

      lw    $t9, y_history($s5)           # $t9 = y_history[i]
      get_Y:
      li    $t8, WHEREY       # $t8 = y_current
      lw    $t8, 0($t8)
      bne   $t8, $t9, get_Y         # y_current == y_history[i]
      nop                     # -> turn or end

      beq $s5, 4, goBackward_end    # l_history == 4
      nop                     # -> end
      addi $s5, $s5, -4             # l_history--
      j goBackward_turn       # else -> turn

goBackward_end:
      jal   STOP
      sw    $zero, a_current        # update heading
```

```
      jal   ROTATE

      addi  $s5, $zero, 4
      sw    $s5, l_history          # reset l_history = 4

      j     printCmd

#-------------------------------------------------------------
# saveHistory()
#-------------------------------------------------------------
saveHistory:
      addi  $sp, $sp, 4             # backup
      sw    $t1, 0($sp)
      addi  $sp, $sp, 4
      sw    $t2, 0($sp)
      addi  $sp, $sp, 4
      sw    $t3, 0($sp)
      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)

      lw    $s1, WHEREX             # s1 = x
      lw    $s2, WHEREY             # s2 = y
      lw    $s4, a_current          # s4 = a_current

      lw    $t3, l_history          # $t3 = l_history
      addi  $t3,$t3,4
      sw    $s1, x_history($t3)           # store: x, y, alpha
      sw    $s2, y_history($t3)
      sw    $s4, a_history($t3)

      sw    $t3, l_history

      lw    $s4, 0($sp)             # restore backup
      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

saveHistory_end: jr     $ra
save_code:  addi  $sp, $sp, 4 # back up
            sw    $t0, 0($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    $s0, 0($sp)
            addi  $sp, $sp, 4
            sw    $s1, 0($sp)



            la $t0,cmdLen
            lw $t0,cmdLen
            la $s0, cmdCode
            la $s1, code_history
            li $t1,0          # i =0
save_code_loop:
            add $t2, $s0,$t1
            lb $t2,0($t2)           # $t2 = cmdcode[i]
            add $t3, $s1,$t1
            sb $t2,0($t3)           # code_history[i] = cmdcode[i]
            beq $t1,$t0,end_save_code_loop
            add $t1,$t1,1
            j save_code_loop
end_save_code_loop:



      lw    $s1, 0($sp)             # restore backup
      addi  $sp, $sp, -4
      lw    $s0, 0($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    $t0, 0($sp)
      addi  $sp, $sp, -4
      jr $ra
copy_code_history:      addi  $sp, $sp, 4 # back up
            sw    $t0, 0($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    $s0, 0($sp)
            addi  $sp, $sp, 4
            sw    $s1, 0($sp)


            la $s0, code_history
            lb $t0,0($s0)
            beq $t0,$0,waitForKey
            nop
            beq $t0,$0,waitForKey
            li $t0,3
            sb $t0, cmdLen
            la $s0, code_history
            la $s1, cmdCode
            li $t1,0


copy_code_history_loop:
            add $t2, $s0,$t1
            lb $t2,0($t2)
            add $t3, $s1,$t1
            sb $t2,0($t3)
            beq $t1,$t0,end_copy_code_history_loop
            add $t1,$t1,1
            j copy_code_history_loop
end_copy_code_history_loop:


      lw    $s1, 0($sp)             # restore backup
      addi  $sp, $sp, -4
      lw    $s0, 0($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    $t0, 0($sp)
      addi  $sp, $sp, -4
      j checkCmd
#=============================================================================
=======
# Procedure for Mars bot
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# GO()
```

```
#-------------------------------------------------------------
GO:   addi  $sp, $sp, 4             # backup
      sw    $at, 0($sp)
      addi  $sp, $sp, 4
      sw    $k0, 0($sp)

      li    $at, MOVING             # change MOVING port
      addi  $k0, $zero, 1           # to logic 1,
      sb    $k0, 0($at)             # to start running

      li    $t7, 1                  # isGoing = 0
      sw    $t7, isGoing

      lw    $k0, 0($sp)             # restore back up
      addi  $sp, $sp, -4
      lw    $at, 0($sp)
      addi  $sp, $sp, -4

GO_end:     jr    $ra

#-------------------------------------------------------------
# STOP()
#-------------------------------------------------------------
STOP:       addi  $sp, $sp, 4             # backup
      sw    $at, 0($sp)

      li    $at, MOVING             # change MOVING port to 0
      sb    $zero, 0($at)           # to stop

      sw    $zero, isGoing          # isGoing = 0

      lw    $at, 0($sp)             # restore back up
      addi  $sp, $sp, -4

STOP_end:   jr $ra

#-------------------------------------------------------------
# TRACK()
#-------------------------------------------------------------
TRACK:      addi  $sp, $sp, 4             # backup
      sw    $at, 0($sp)
      addi  $sp, $sp, 4
      sw    $k0, 0($sp)

      li    $at, LEAVETRACK         # change LEAVETRACK port
      addi  $k0, $zero,1            # to logic 1,
      sb    $k0, 0($at)             # to start tracking

      addi  $s0, $zero, 1
      sw    $s0, isTracking

      lw    $k0, 0($sp)             # restore back up
      addi  $sp, $sp, -4
      lw    $at, 0($sp)
```

```
      addi  $sp, $sp, -4

TRACK_end:  jr $ra

#-------------------------------------------------------------
# UNTRACK()
#-------------------------------------------------------------
UNTRACK:    addi  $sp, $sp, 4       # backup
      sw    $at, 0($sp)

      li    $at, LEAVETRACK   # change LEAVETRACK port to 0
      sb    $zero, 0($at)     # to stop drawing tail

      sw    $zero, isTracking

      lw    $at, 0($sp)       # restore back up
      addi  $sp, $sp, -4

UNTRACK_end: jr   $ra

#-------------------------------------------------------------
# ROTATE()
#-------------------------------------------------------------
ROTATE:     addi  $sp, $sp, 4       # backup
      sw    $t1, 0($sp)
      addi  $sp, $sp, 4
      sw    $t2, 0($sp)
      addi  $sp, $sp, 4
      sw    $t3, 0($sp)

      li    $t1, HEADING      # change HEADING port
      la    $t2, a_current
      lw    $t3, 0($t2)       # $t3 is heading at now
      sw    $t3, 0($t1)       # to rotate robot

      lw    $t3, 0($sp)       # restore back up
      addi  $sp, $sp, -4
      lw    $t2, 0($sp)
      addi  $sp, $sp, -4
      lw    $t1, 0($sp)
      addi  $sp, $sp, -4

ROTATE_end: jr    $ra

#=============================================================================
=======
# Procedure for string
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# strcmp()
# - input: $s3 = string to compare with cmdCode
# - output: $t0 = 0 if not equal, 1 if equal
```

```
#-------------------------------------------------------------

strcmp:     addi  $sp, $sp, 4             # back up
      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)

      li    $t0,0       # $t0  = 0
      li    $t1,0       # $t1 = i = 0

strcmp_loop:
      beq   $t1, 3, strcmp_equal          # if i = 3 -> end loop -> equal
      nop

      lb    $t2, cmdCode($t1)       # $t2 = cmdCode[i]

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

      beq   $t2, $t3, strcmp_next         # if $t2 == $t3 -> continue the
loop
      nop

      j     strcmp_end

strcmp_next:
      addi  $t1, $t1, 1             # i++
      j     strcmp_loop

strcmp_equal: add $t0, $zero, 1           # $t0 = 1

strcmp_end: lw    $t3, 0($sp)             # restore the backup
      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

      jr $ra

#-------------------------------------------------------------
# strClear()
#-------------------------------------------------------------

strClear:
      addi  $sp, $sp, 4             # backup
      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)

      lw    $t3, cmdLen             # $t3 = cmdLen
      addi  $t1, $zero, -1          # $t1 = -1 = i

strClear_loop:
      addi  $t1, $t1, 1             # i++
      sb    $zero, cmdCode($t1)           # cmdCode[i] = '\0'

      bne   $t1, $t3, strClear_loop # if $t1 <=3 resetInput loop
      nop

      sw    $zero, cmdLen           # reset cmdLen = 0

strClear_end:
      lw    $s2, 0($sp)             # restore backup
      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

#=============================================================================
=======
# 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)
```

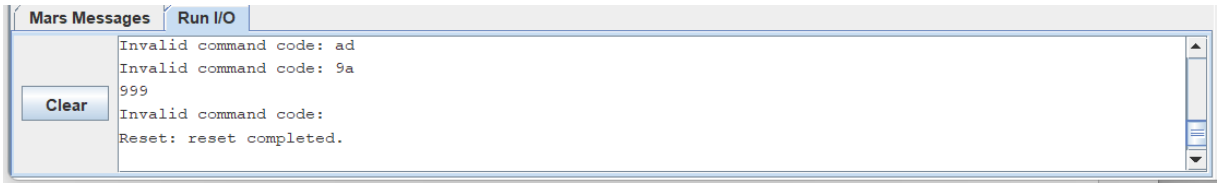
```
      addi  $sp, $sp, 4
      sw    $at, 0($sp)
      addi  $sp, $sp, 4
      sw    $s0, 0($sp)
      addi  $sp, $sp, 4
      sw    $s1, 0($sp)
      addi  $sp, $sp, 4
      sw    $s2, 0($sp)
      addi  $sp, $sp, 4
      sw    $t4, 0($sp)
      addi  $sp, $sp, 4
      sw    $s3, 0($sp)
#-------------------------------------------------------------
# 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

case_0:     li    $s0, '0'          # $s0 store code in char type
      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, cmdCode
      la    $s2, cmdLen
      lw    $s3, 0($s2)             # $s3 = strlen(cmdCode)
      add   $t4, $s3, $0            # $t4 = cmdLen
      add   $s1, $s1, $t4           # $s1 = cmdCode + i
      sb    $s0, 0($s1)             # cmdCode[cmdLen] = $s0
      addi  $s0, $zero, '\n'        # add '\n' character to end of string
      addi  $s1, $s1, 1
      sb    $s0, 0($s1)       #cmdCode[cmdLen+1] = '\n'

      addi  $s3, $s3, 1
      sw    $s3, 0($s2)             # update cmdLen
```

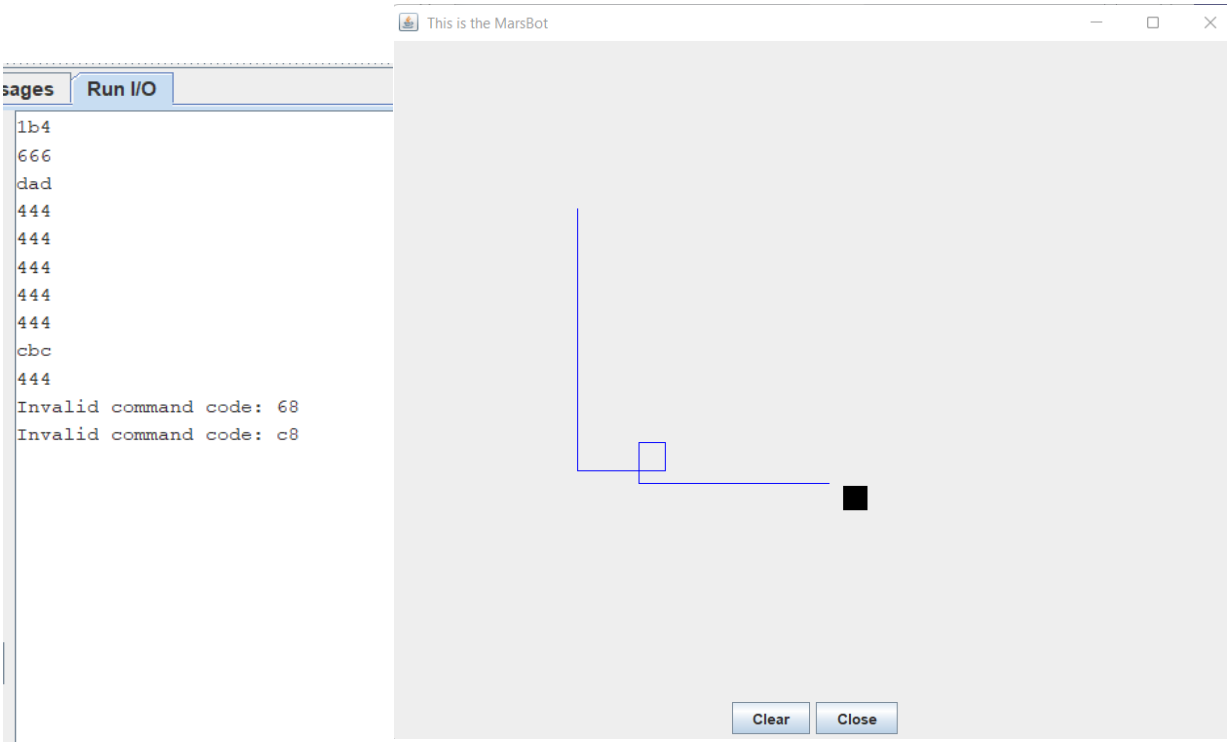
```
#-----------------------------------------------------
# Evaluate the return address of main routine
# epc <= epc + 4
#-----------------------------------------------------
next_pc:
      mfc0  $at, $14          # $at <= Coproc0.$14 = Coproc0.epc
      addi  $at, $at, 4       # $at = $at + 4 (next 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
```

## 1.5. Thực thi chương trình

Thông báo nhập mã điều khiển không hợp lệ




Marbot thực thi các mã điều khiển



# 2. Project 03: Kiểm tra tốc độ và độ chính xác khi gõ văn bản

*Sinh viên thực hiện*: Nguyễn Thị Diệu Linh – 20205094

## 2.1. Đề bài

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 m**O**n ky **5**huat 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.

## 2.2. Cách thực hiện

Lưu các kí tự mà người dùng nhập vào trong Keyboard thành 1 chuỗi và so sánh chuỗi được nhập với chuỗi mẫu. Ta so sánh 2 mã ASCII của 2 ký tự ở cùng vị trí với nhau trên 2 chuỗi. Nếu 2 kí tự có cùng mã ASCII ở cùng vị trí thì ta tăng biến đếm số ký tự đúng của nó thêm 1.

Cách tính số kí tự trong 01 giây: Tạo 250 chu kì, mỗi chu kỳ dài 4ms để đo được số kí tự nhập được trong 01 giây.

Cách tính tốc độ gõ: Tổng số kí tự gõ / Tổng số thời gian

## 2.3. Các chương trình con

### 2.3.1. In ra màn hình và thể hiện trên LED 7 đoạn số kí tự nhập được trong 01 giây

Vòng lặp giữa các chương trình con để tính số kí tự thực hiện trong 01 giây:

`WAIT_FOR_KEY`: Kiểm tra xem có ký tự mới nào được nhập vào hay không.
`MAKE_INTER`: Cập nhật số kí tự đã được nhập trong 1 giây.

`CK`: Kiểm tra xem đã chương trình chạy được 1 giây hay chưa.

SETCOUNT: Để in ra số kí tự đã được nhập trong 1 giây.

DISPLAY_SPEED: Thể hiện số kí tự được nhập vào trong 1 giây và tổng số kí tự đúng trên đèn LED 7 đoạn trái và phải.

### 2.3.2. In ra màn hình và LED 7 đoạn số kí tự nhập đúng

COUNTER_KEYBOARD:

READ_KEY: Đọc kí tự người dùng nhập vào.

WAIT_FOR_DIS: Hiển thị kí tự được nhập vào trên DISPLAY.

SHOW_KEY: Tính toán tổng số kí tự đã nhập và xử lí khi người dùng bấm Enter.

END, COMPARE_STRING, CONTINUE: So sánh 2 xâu nhập vào và tính toán ra số kí tự đúng. Kí tự được coi là đúng khi nằm ở cùng vị trí trên 2 xâu và giống nhau.

PRINT: In ra màn hình và hiển thị trên đèn LED số kí tự nhập đúng của người dùng.

## 2.4. 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 KEY_CODE         0xFFFF0004    # Dia chi luu ma Ascii cua ky tu duoc
nhap vao
.eqv KEY_READY        0xFFFF0000    # Gia tri o dia chi nay bang 1 neu co
phim moi duoc danh. Duoc tu dong xoa sau khi load word
.eqv DISPLAY_CODE     0xFFFF000C    # Dia chi de hien thi ky tu ra man hinh
.eqv DISPLAY_READY    0xFFFF0008    # Bang 1 neu man hinh san sang cho hien
thi, tu dong xoa sau khi store word
.eqv MASK_CAUSE_KEYBOARD   0x0000034# Keyboard cause

.data
            bytehex     : .byte
0x3F,0x6,0x5B,0x4F,0x66,0x6D,0x7D,0x7,0x7F,0x6F
            storestring : .space 1000
            enterstring : .asciiz "Bo mon ky thuat may tinh"
            mess: .asciiz "\nSo ky tu trong 1s :  "
            speed: .asciiz "\nToc do go tren giay:  "
            don_vi: .asciiz " time/char"
            numkeyright: .asciiz  "\nSo ky tu nhap dung la: "
            notification: .asciiz "\nBan co muon nhap lai tu dau khong? "
```

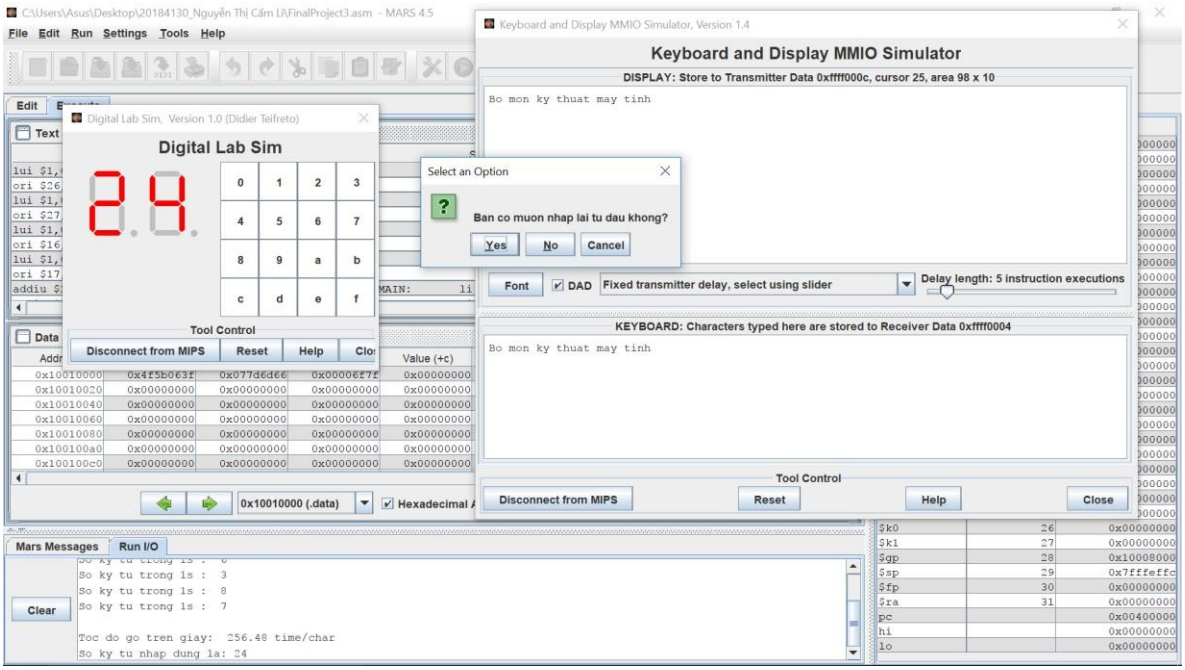
```
.text
            li   $k0,  KEY_CODE
            li   $k1,  KEY_READY
            li   $s0, DISPLAY_CODE
            li   $s1, DISPLAY_READY

MAIN:       li $s4,0                #dung de dem toan bo so ky tu nhap vao
            li $s3,0                #dung de dem so vong lap
            li $t4,10
            li $t5,250              #luu gia tri so vong lap.
            li $t6,0                #bien dem so ky tu nhap duoc trong 1s
            li $t9,0                #moi lan nguoi dung nhap xong, thanh ghi
t9 duoc gan thanh 1, de chuong trinh biet va goi thu tuc ASK_LOOP (hoi nguoi
dung co muon nhap lai ko)
            li $s5,0                # so giay la 0ms
LOOP:
WAIT_FOR_KEY:lw   $t1, 0($k1)        #Doc gia tri o KEY_READY neu co gia tri
la 1 thi co ky tu moi duoc nhap vao, 0 neu khong co ky tu moi
             beq  $t1, $zero,CK      #Neu khong doc duoc ky tu nao thi tiep
tuc chu ky moi ma khong tao interrupt
#-----------------------------------------------------------------
#Bo su ly MIPS cho phep tao ra ngat mem, bang lenh teq, hoac teqi
#Thiet bi Keyboard khong tu tao ra ngat khi co mot phim duoc bam ma nguoi
#ma nguoi lap trinh phai tu tao ngat mem
#-----------------------------------------------------------------
MAKE_INTER: addi $t6,$t6,1          #tang bien dem ky tu nhap duoc trong 1s
len 1
            teqi $t1, 1             #Tao ngat mem


CK:         addi $s3, $s3, 1        #dem so chu ky trong s hien tai, duoc
tra ve 0 sau moi 1s
            addi  $s5,$s5,4         # so giay la +4ms
            div $s3,$t5             #moi chu ky nghi 4ms nen neu du 250 chu
ky tuc la du 1s
            mfhi $t7                #luu phan du cua phep chia tren
            bne $t7,0,SLEEP         #neu chua duoc 1s tiep tuc cho chuong
trinh nghi 4ms

#neu da duoc 1s thi nhay den nhan SETCOUNT de thuc hien in ra man hinh
SETCOUNT:
            li $s3,0                #tai lap gia tri cua $t3 ve 0 de dem lai
so vong lap cho cac lan tiep theo
            li $v0,4                #in ra console thong bao so ky tu nhap
duoc trong 1s
            la $a0,mess
            syscall
            nop
            li    $v0,1             #in ra so ky tu trong 1s
            add   $a0,$t6,$zero
            syscall
```

```
DISPLAY_SPEED:
        div $t6,$t4                     #lay so ky tu nhap duoc trong 1s chia
cho 10
        mflo $t7                        #luu gia tri phan nguyen, gia tri nay
se duoc luu o den LED ben trai
        la $s2,bytehex                  #lay dia chi cua danh sach luu gia tri
cua tung chu so den LED
        add $s2,$s2,$t7
        lb $a0,0($s2)                   #lay noi dung cho vao $a0
        li   $t0,  SEVENSEG_LEFT        # Lay dia chi hien thi den leg bay doan
trai
        sb   $a0,  0($t0)               # gian gia tri can hien thi

        mfhi $t7                        #luu gia tri phan du cua phep chia, gia
tri nay se duoc in ra trong den LED ben phai
        la $s2,bytehex
        add $s2,$s2,$t7
        lb $a0,0($s2)
        li   $t0,  SEVENSEG_RIGHT   # Lay dia chi hien thi den leg bay doan
phai
        sb   $a0,  0($t0)           # gian gia tri can hien thi

        li    $t6,0                 #reset $t6 = 0
        beq $t9,1,ASK_LOOP
SLEEP:
        addi    $v0,$zero,32
        li      $a0,4                # sleep 4 ms
        syscall
        nop
        b       LOOP                # Loop


#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# XU LY NGAT
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
.ktext  0x80000180                      #chuong trinh con chay sau khi
interupt duoc goi.
        mfc0  $t1, $13                  # cho biet nguy�n nh�n l�m tham
chieu dia chi bo nho khong hop
        li    $t2, MASK_CAUSE_KEYBOARD
        and   $at, $t1,$t2
        beq   $at,$t2, COUNTER_KETYBOARD
        j    END_PROCESS

COUNTER_KETYBOARD:
READ_KEY:    lb   $t0, 0($k0)                 #doc ky tu duoc nhap vao

WAIT_FOR_DIS:lw   $t2, 0($s1)               #doc gia tri tai dia chi
DISPLAY_READY, neu gia tri la 1 thi hien thi ky tu nhap vao, nguoc tai tiep
tuc cho
             beq  $t2, $zero, WAIT_FOR_DIS  #tiep tuc cho
```

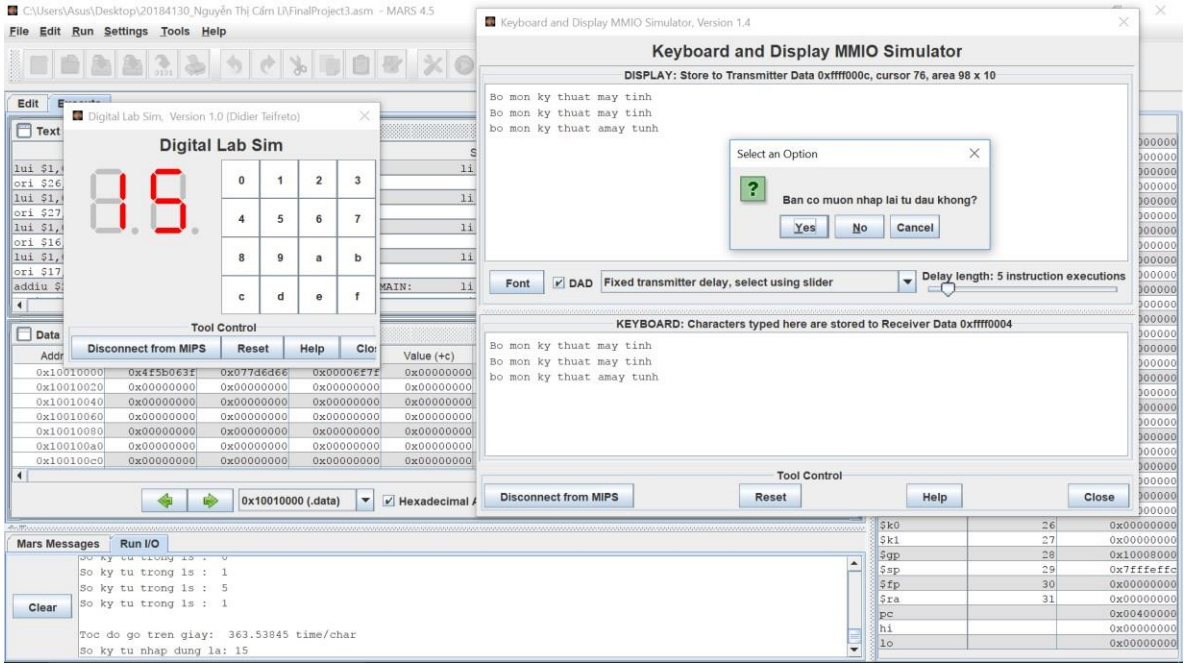
```
SHOW_KEY:    sb $t0, 0($s0)                  #hien thi ky tu vua nhap tu ban
phim tren man hinh MMIO
             la  $t7,storestring             #luu dia chi cua mang nhap vao
thanh ghi t7
             add $t7,$t7,$s4
             sb $t0,0($t7)                 #luu ky tu vua doc duoc vao mang
             addi $s4,$s4,1                        #tang so ky tu nhap duoc
them 1
             beq $t0,'\n',END                 #neu ky tu nhap vao la ky tu
xuong dong thi nhay den label end
END_PROCESS:
#-----------------------------------------------------
#Tuy nhien, luu y rang, trong MARS, thanh ghi PC van chua dia chi cua lenh
#ma ngat xay ra, tuc l� lenh d� thuc hien xong, chu kh�ng chua dia chi cua
#lenh ke tiep. Boi v�y phi tu lap tr�nh de tang dia chi chua trong thanh
ghi
#epc bang c�ch su dung 2 lenh mfc0 (de doc thanh ghi trong bo dong xu l�
#C0) v� mtc0 (de ghi gia tri vao thanh ghi trong bo dong xu ly C0)
#-----------------------------------------------------
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                            # tro ve len ke tiep cua chuong
trinh chinh


END:        li $v0,11
            li $a0,'\n'                   #in xuong dong
            syscall
               nop
            li $t1,0                      #i = 0
            li $t3,0                      #bien dem so ky tu nhap dung
            li $t8,24                     #luu $t8 la do dai xau "Bo mon ky
thuat may tinh"
            slt $t7,$s4,$t8               #kiem tra do dai xau da cho va xau
nhap vao xau nao ngan hon thi duyet theo xau do
            bne $t7,1, COMPARE_STRING
            add $t8,$0,$s4
            addi $t8,$t8,-1               #ky tu cuoi cung nhap vao la '\n'
khong can xet

COMPARE_STRING: la $t2,storestring
                add $t2,$t2,$t1                 #lap qua cac ky tu
                lb $t5,0($t2)                #lay ky tu duoc luu tru ra
                la $t4,enterstring
                add $t4,$t4,$t1
                lb $t6,0($t4)                 #lay ky tu thu $t1 trong
enterstring luu vao $t6
                bne $t6,$t5,CONTINUE
                addi $t3,$t3,1                 #hai ky tu dang xet giong
nhau tang bien dem
```

```
CONTINUE:   addi $t1,$t1,1                      #i++
            beq $t1,$t8,PRINT                   #i = n
            j COMPARE_STRING                        #tiep tuc check ki tu
tiep theo

PRINT:
        li $v0, 4
        la $a0, speed
       syscall          # In xau speed

      mtc1 $s4, $f1           # Convert $s4(integer) -> $f1(float): so luong
phim da nhap
      mtc1 $s5, $f2           # Convert $s5(integer) -> $f2(float): so luong
time (so chu ky)
      div.s $f12, $f2, $f1    # Toc do trung binh (so chu ky ngat/1 ky tu)
      li $v0, 2
      syscall

      li $v0, 4
      la $a0, don_vi
      syscall                 # In message don_vi

      li $v0,4
      la $a0,numkeyright
      syscall
      li $v0,1
      add $a0,$0,$t3
      syscall
      li $v0,4


            li $t9,1
            li $t4,10                       #gan lai gia tri cho t4 = 10 de
phuc vu cho viec hien thi den leg
            add $t6,$0,$t3
            b DISPLAY_SPEED

ASK_LOOP:   li $v0, 50                              #tao thong bao hoi nguoi
dung
         la $a0, notification
         syscall
         beq $a0,0,MAIN                     #neu nguoi dung chon yes quay lai
MAIN
         nop
         li $v0,10
         syscall
```

## 2.5. Thực thi chương trình




Kết quả khi nhập đúng toàn bộ chuỗi




Kết quả khi nhập sai chuỗi