HUST

TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

ONE LOVE. ONE FUTURE.

ASSEMBLY LANGUAGE & COMPUTER ARCHITECTURE

IT3280E

CLASS CODE: 131102

ONE LOVE. ONE FUTURE.

TRƯỜNG ĐẠI HỌC
BÁCH KHOA HÀ NỘI
HANOI UNIVERSITY
OF SCIENCE AND TECHNOLOGY

Group 4

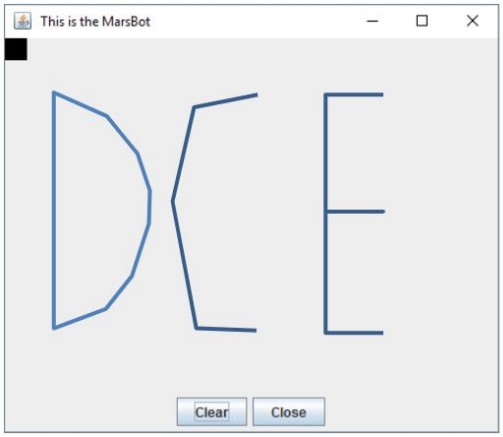
Instructor: Đỗ Công Thuần

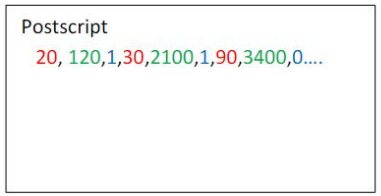
Group member: Đào Quang Dương - 20194747
Trần Chí Thành - 20194845

ONE LOVE. ONE FUTURE.

About our projects

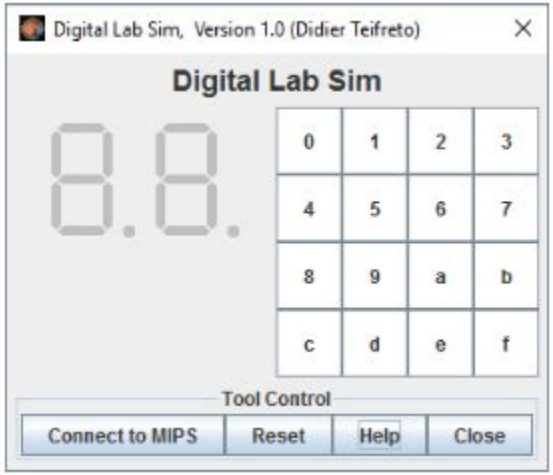
Postscript CNC Marsbot

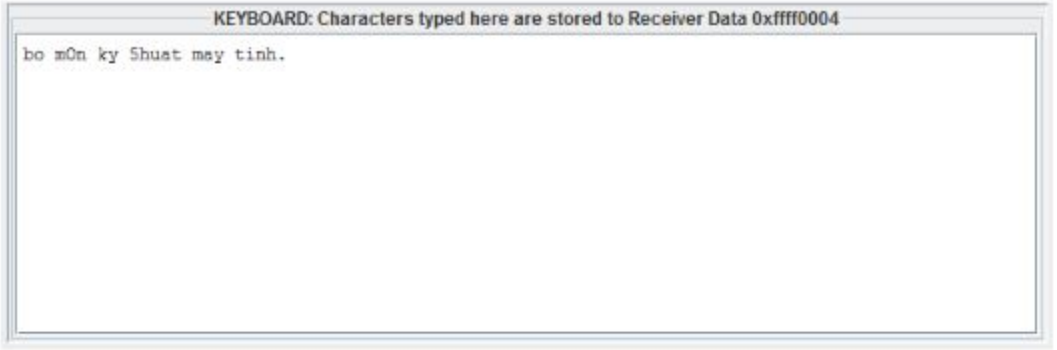




0	1	2	3
4	5	6	7
8	9	a	b
c	d	e	f

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





Postscript CNC Marsbot

Yêu cầu: Máy gia công cơ khí chính xác CNC Marsbot được dùng để cắt tấm kim loại theo các đường nét được quy định trước.

Để điều khiển Marsbot cắt đúng như hình dạng mong muốn, người ta nạp vào Marsbot một mảng cấu trúc gồm 3 phần tử:

<Góc chuyển động>, <Thời gian>, <Cắt/Không cắt>

- Mã nguồn chứa 3 postscript và người dùng sử dụng 3 phím 0, 4, 8 trên bàn phím Key Matrix để chọn postscript nào sẽ được gia công.

Key Matrix:

Digital Lab Sim is one of MARS tools for Controlling I/O devices.

The Hex keyboard of Digital Lab Sim is Memory-mapped I/O. It includes:

Byte value @0xFFFF0012 → used to command row number of hexadecimal keyboard.

Byte value @0xFFFF0014 → used to receive row and column number of the key pressed (0 = no key pressed).

```
.eqv    IN_ADDRESS_HEXA_KEYBOARD    0xFFFF0012
.eqv    OUT_ADDRESS_HEXA_KEYBOARD   0xFFFF0014
```

The program have to scan, one by one, each row and then observe if a key is pressed.

Problems and Solutions

Key Matrix:

load I/O address

```
li      $t7, IN_ADDRESS_HEXA_KEYBOARD
li      $t8, OUT_ADDRESS_HEXA_KEYBOARD
```

polling: example for row 1, if pressed, the keycode will store to $t6

```
li      $t6, 0x01                   # check row 1 with key 0, 1, 2, 3
sb      $t6, 0($t7)                 # must reassign expected row
lb      $t6, 0($t8)                 # read scan code of key button
```

and then check the value to get the corresponding postscript

```
beq     $t6, 0x11, key0_pressed
beq     $t6, 0x12, key4_pressed
beq     $t6, 0x14, key8_pressed
```

```
key0_pressed:
    la      $t0, postscript0
    j       Marbot_DRAW
```

Postscript:

Postscript is stored in a word array.

Postscript is the set of structs (góc chuyển động, thời gian, cắt hay không cắt)

<Góc chuyển động>, <Thời gian>, <Cắt/Không cắt>
Trong đó <Góc chuyển động> là góc của hàm HEADING của Marsbot
<Thời gian> là thời gian duy trì quá trình vận hành hiện tại
<Cắt/Không cắt> thiết lập lưu vết/không lưu vết

Postscript struct is read and store by using loop

```
lw      $s0, ($t0)      # $s0: goc chuyen dong
addi    $t0, $t0, 4
lw      $s1, ($t0)      # $s1: thoi gian
addi    $t0, $t0, 4
lw      $s2, ($t0)      # $s2: cat/khong cat
addi    $t0, $t0, 4
```

Marsbot:

The MarsBot is a virtual robot in the MARS. The MarsBot can travel in a 2D space, optionally leaving a trail (or track).

This program uses the HEADING,LEAVETRACK, and MOVE to control the movement of the 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
```

Problems and Solutions

Marsbot:

STOP and GO change the value in the MOVING location to 0 or 1

```
GO:
    li      $at, MOVING         # change MOVING port
    addi    $k0, $zero, 1       # to  logic 1,
    sb      $k0, 0($at)         # to start running
    nop
    jr      $ra
    nop
STOP:
    li      $at, MOVING         # change MOVING port TO 0
    sb      $zero, 0($at)       # to stop running
    nop
    jr      $ra
    nop
```

Marsbot:

TRACK and UNTRACK change the value in the LEAVETRACK location to 0 or 1

```
TRACK:
    li      $at, LEAVETRACK     # change LEAVETRACK port
    addi    $k0, $zero, 1       # to  logic 1,
    sb      $k0, 0($at)         # to start tracking
    nop
    jr      $ra
    nop

UNTRACK:
    li      $at, LEAVETRACK     # change LEAVETRACK port to 0
    sb      $zero, 0($at)       # to start tracking
    nop
    jr      $ra
    nop
```

Marsbot:

ROTATE change the value in the HEADING location to an angle between 0° and 359°

```
#-------------------------------------------------------------
# ROTATE procedure, to rotate the robot
# param[in]    $a0, An angle between 0 and 359
#                   0 : North (up)
#                   90: East  (right)
#                  180: South (down)
#                  270: West  (left)
#-------------------------------------------------------------
ROTATE:
    li      $at, HEADING    # change HEADING port
    sw      $a0, 0($at)     # to rotate robot
    nop
    jr      $ra
    nop
```

Marsbot:

In main program:

→ ROTATE

```
move    $a0, $s0
jal     ROTATE
```

→ Check whether or not to leave a TRACK

```
beqz    $s2, not_track
jal     TRACK
not_track:
```

→ start GO for a time by sleep system

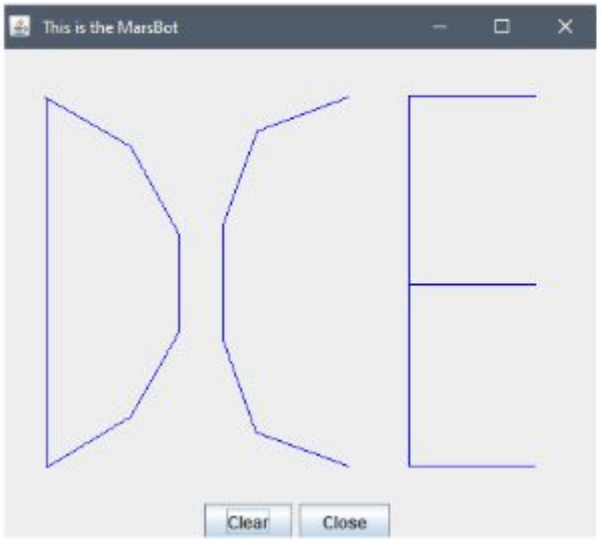
```
jal     GO
move    $a0, $s1
addi    $v0, $zero, 32
syscall
```

→ STOP and UNTRACK to keep old track

Postscript CNC Marsbot

Result

postscript 1 - key 0 - DCE - 15 tracks

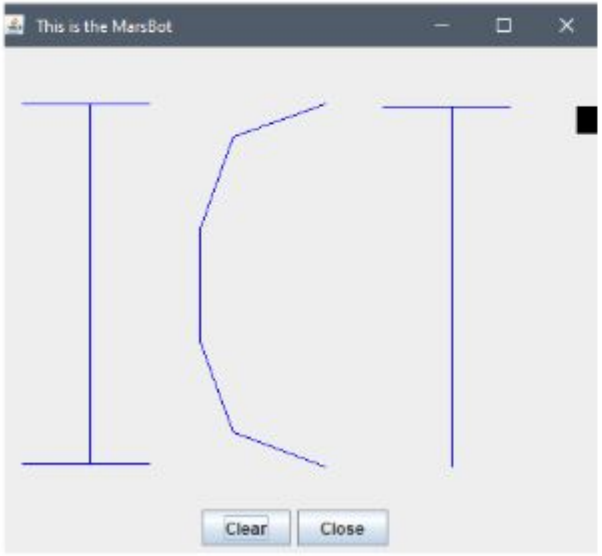


135,2000,0, 180,12000,1, 60,3000,1, 30,3000,1, 0,3000,1,
330,3000,1, 300,3000,1, 90,10000,0, 250,3000,1
200,3000,1, 180,3500,1, 160,3000,1, 110,3000,1, 90,6000,0,
270,4000,1, 0,12000,1, 90,4000,1, 180,6000,0, 270,4000,1,
90,10000,0

Postscript CNC Marsbot

Result

postscript 2 - key 4 - ICT - 10 tracks

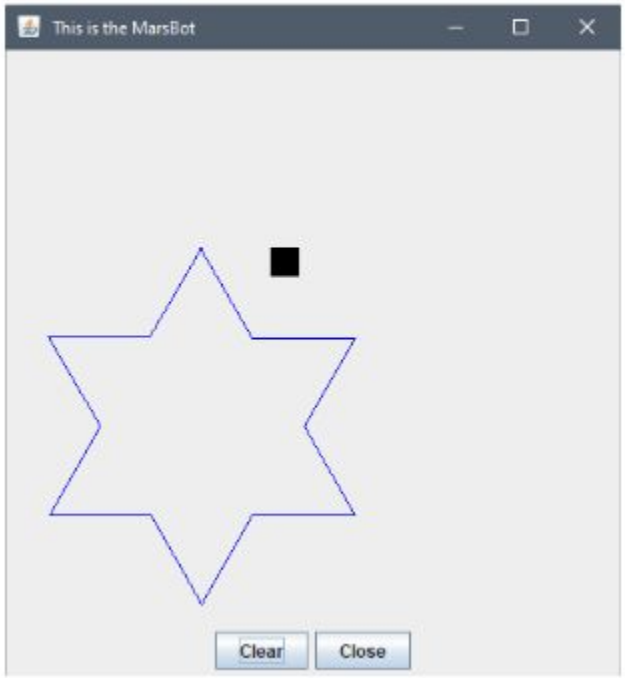


120,3500,0, 180,12000,1, 270,2000,0, 90,4000,1, 0,12000,0, 270,4000,1, 90,10000,0, 250,3000,1 200,3000,1, 180,3500,1, 160,3000,1, 110,3000,1, 90,4000,0, 0,12000,1, 270,2000,0, 90,4000,1, 90,2000,0

Postscript CNC Marsbot

Result

postscript 3 - key 8 - star - 12 tracks



90,6000,0, 180,6000,0, 210,3000,1, 270,3000,1, 150,3000,1,
210,3000,1, 90,3000,1, 150,3000,1, 30,3000,1, 90,3000,1,
330,3000,1, 30,3000,1, 270,3000,1, 330,3000,1, 90,2000,0

Differences

1. Postscript

   The number "-1" is added at the end of the array. It makes checking the last element of the array easier and faster while it doesn't affect anything. Easy to create new postscript without worrying about its length.

2. Key Scan

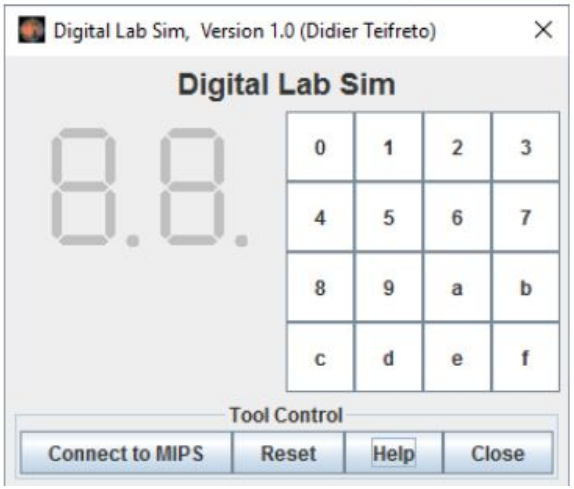
```
move    $t4, $t5                    # previously pressed key

sub     $t4, $t4, $t5
beqz    $t4, back_to_polling
```

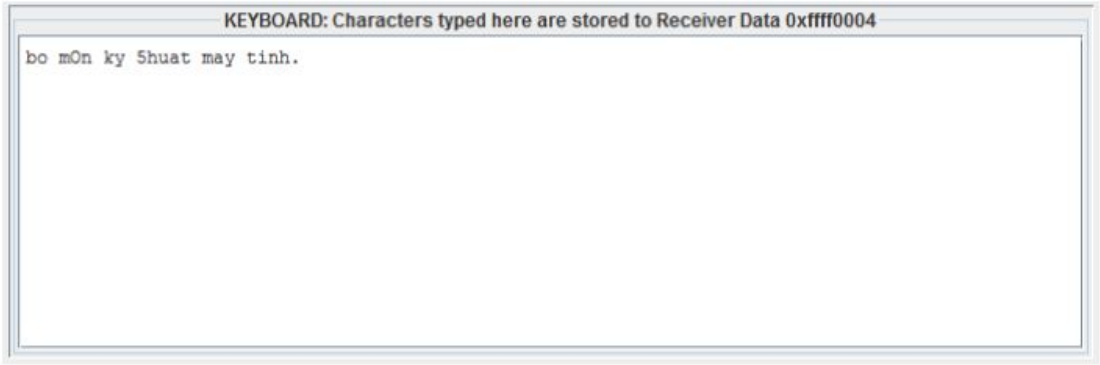
   When polling, the previously pressed key is stored and used to check whether or not it is different from the currently pressed key. If not, the program will back_to_polling. That will save the marsbot from having to redraw the previous postscript when the user forgets to reset the hex keyboard.

Typing accuracy and speed check

Chương trình sẽ đo tốc độ gõ bàn phím và hiện kết quả bằng 2 đèn led 7 đoạn:

- Cho 1 đoạn văn bản mẫu, cố định sẵn trong mã nguồn. VD: “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.
- Trong khoảng thời gian đó, người dùng nhập các kí tự từ bàn phím. Ví dụ nhập “Bo m0n ky 5huat may tinh”(sai chữ 0 và 5) mà người dùng đã gõ để hiển thị đèn LED.





Typing accuracy and speed check

Solution

1. Declaration

```
.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_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 DISPLAY_CODE   0xFFFF000C          # ASCII code to show, 1 byte
.eqv DISPLAY_READY  0xFFFF0008          # =1 if the display has already to do
                                        # Auto clear after sw
.eqv MASK_CAUSE_KEYBOARD   0x0000034    # Keyboard Cause

.data
byte_hex_led     : .byte 63,6,91,79,102,109,125,7,127,111 # gia tri cua byte tai
storestring : .space 1000                    #khoang trong de luu cac ky tu n
stringsource : .asciiz "bo mon ky thuat may tinh"
Message: .asciiz "\n So ky tu trong 1s :  "
resultMess: .asciiz  "\n So ky tu nhap dung la: "

.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 $t5,200                      #luu gia tri so vong lap.
        li $t6,0                        #bien dem so ky tu nhap duoc trong 1s
        li $t9,0
```

Variable

Typing accuracy and speed check

Solution

2. Counting the numbers of letter typed in 1s

- POLLING để xử lí dữ liệu nhập vào từ người dùng

```
POLLING:                        #neu da lap duoc 200 vong(1s) se nhay den xu ly so ky tu nhap trong 1s.
        addi    $s3, $s3, 1     #dem so ky tu nhap vao tu ban phim.
        div     $s3,$t5         #lay so vong lap chia cho 200 de xac dinh da duoc 1s hay chua
        mfhi    $t7             #luu phan du cua phep chia tren
        bne     $t7,0,SLEEP     #neu chua duoc 1s (s3/s5 = 1) nhay den label sleep
                                #neu da duoc 1s thi nhay den nhan SETCOUNT de thuc hien in ra man hinh
```

- SETCOUNT dùng để in số kí tự trong từng giây

```
SETCOUNT:                       #in ra man hinh so chu danh duoc trong 1s
        li      $s3,0           #tai lap gia tri cua $s3 ve 0 de dem lai so vong lap cho cac lan tiep theo
        li      $v0,4           #bat dau chuoi lenh in ra console so ky tu nhap duoc trong 1s
        la      $a0,Message
        syscall
        li      $v0,1                   #in ra so ky tu trong 1s
        add     $a0,$t6,$zero           #Nhan bien t6 dem so ky nhap duoc trong 1s
        syscall
```

- SLEEP

```
SLEEP:
        addi    $v0,$zero,32
        li      $a0,5                   #sleep 5 ms
        syscall
        nop                             #nop la can thiet sau trc khi branch sau syscall
        b       LOOP                    #Loop
```

Typing accuracy and speed check

Solution

3. Soft interrupt

- Đọc dữ liệu từ receiver data:

```
.ktext    0x80000180                    #chuong trinh con chay sau khi interupt duoc goi.
          mfc0    $t1, $13              #cho biet nguyen nhan lam tham chieu dia chi bo nh
          li      $t2, MASK_CAUSE_KEYBOARD
          and     $at, $t1,$t2
          beq     $at,$t2, COUNTER_KETYBOARD
          j        END_PROCESS

COUNTER_KETYBOARD:
READ_KEY:
          lw    $t0, 0($k0)                 #$t0 = [$k0] = KEY_CODE
```

- Kiểm tra nếu có BACKSPACE :

```
beq $t0,10,END                #Can chu y toi ki hieu xuong dong '\n' - ASCII: 10
                              #Neu xuat hien '\n' --> xong input process --> Chuyen toi END
```

- Xóa kí tự vừa nhập vào và giảm số lượng kí tự đi 1 đơn vị:

```
DELETE_CHAR:                  #viec xoa ki tu nhap vao tuong duong
                              #voi viec lam giam tong so ky tu nhap vao s4 di 1
          addi    $s4, $s4, -1
          sb      $t0, 0($s0)
```

Typing accuracy and speed check

Solution

3. Soft interrupt

- Nếu k gặp BACKSPACE và giá trị của display_ready là 1 thì in ra

```
SHOW_KEY:
        sb   $t0, 0($s0)                #hien thi ky tu vua nhap tu ban phim tren man hinh MMI
                                        #luu ky tu input vao mang storestring
        la   $t7,storestring            #lay $t7 con tro cua chuoi luu chuoi nhap vao
        add  $t7,$t7,$s4                #di chuyen con tro toi vi tri moi
        sb   $t0,0($t7)                 #luu lai tu vua nhap
        addi $s4,$s4,1                  #tang so luong tu nhap vao
        beq  $t0,10,END                 #Can chu y toi ki hieu xuong dong '\n' - ASCII: 10
                                        #Neu xuat hien '\n' --> xong input process --> Chuyen

        j  END_PROCESS

DELETE_CHAR:                            #viec xoa ki tu nhap vao tuong duong
```

- Nếu gặp xuống dòng thì chuyển đến cuối chương trình

```
END:
        li      $v0,11
        li      $a0,'\n'                #in xuong dong
        syscall
        li      $t1,0                   #reset bien dem t1 de dem so ky tu da duoc xet
        li      $t3,0                   #khai bao bien t3 de dem so ky tu nhap dung
        li      $t8,24                  #luu $t8 la do dai xau da luu tru trong ma nguon.
        slt     $t7,$s4,$t8             #so sanh xem do dai xau nhap tu ban phim (luu tai than
                                        #xau nao nho hon thi duyet theo do dai cua xau do (s4

        bne     $t7,1, CHECK_STRING
        add     $t8,$0,$s4              #trong TH xau input >= xau luu san, t8 chua do dai cua
        addi    $t8,$t8,-1              #tru 1 vi ky tu cuoi cung la dau enter thi khong can x
```

Typing accuracy and speed check

Solution

4. Typing accuracy check

- So sánh lần lượt các kí tự tương đương trong 2 chuỗi

```
CHECK_STRING:
        la      $t2,storestring
        add     $t2,$t2,$t1
        li      $v0,11                  #in ra cac ky tu da nhap tu ban phim da qua check.
        lb      $t5,0($t2)              #lay ky tu thu $t1 trong storestring luu vao $t5 de so sanh voi
        move    $a0,$t5                 #set noi dung thanh ghi a0 = noi dung thanh ghi t5
        syscall
        la      $t4,stringsource
        add     $t4,$t4,$t1
        lb      $t6,0($t4)              #lay ky tu thu $t1 trong stringsource luu vao $t6
        bne     $t6,$t5,CONTINUE        #so sanh hai ky tu luu trong t5 va t6, neu 2 ky tu ko giong nhau
        addi    $t3,$t3,1               #neu giong, tang so ki tu nhap dung t3 len them 1
```

- Sau khi so sánh xong thì in ra số kí tự đúng

```
PRINT:
        li      $v0,4
        la      $a0,resultMess
        syscall
        li      $v0,1
        add     $a0,$0,$t3              #in so tu nhap dung
        syscall
        li      $t9,1                   #thay doi de su dung lai qua trinh display_led o
        li      $t6,0                   #gan lai gia tri bien dem t6
        li      $t4,10                  #gan lai gia tri so chia = 10 cho t4
        add     $t6,$0,$t3
        b       DISPLAY_DIGITAL
```

Typing accuracy and speed check

Solution

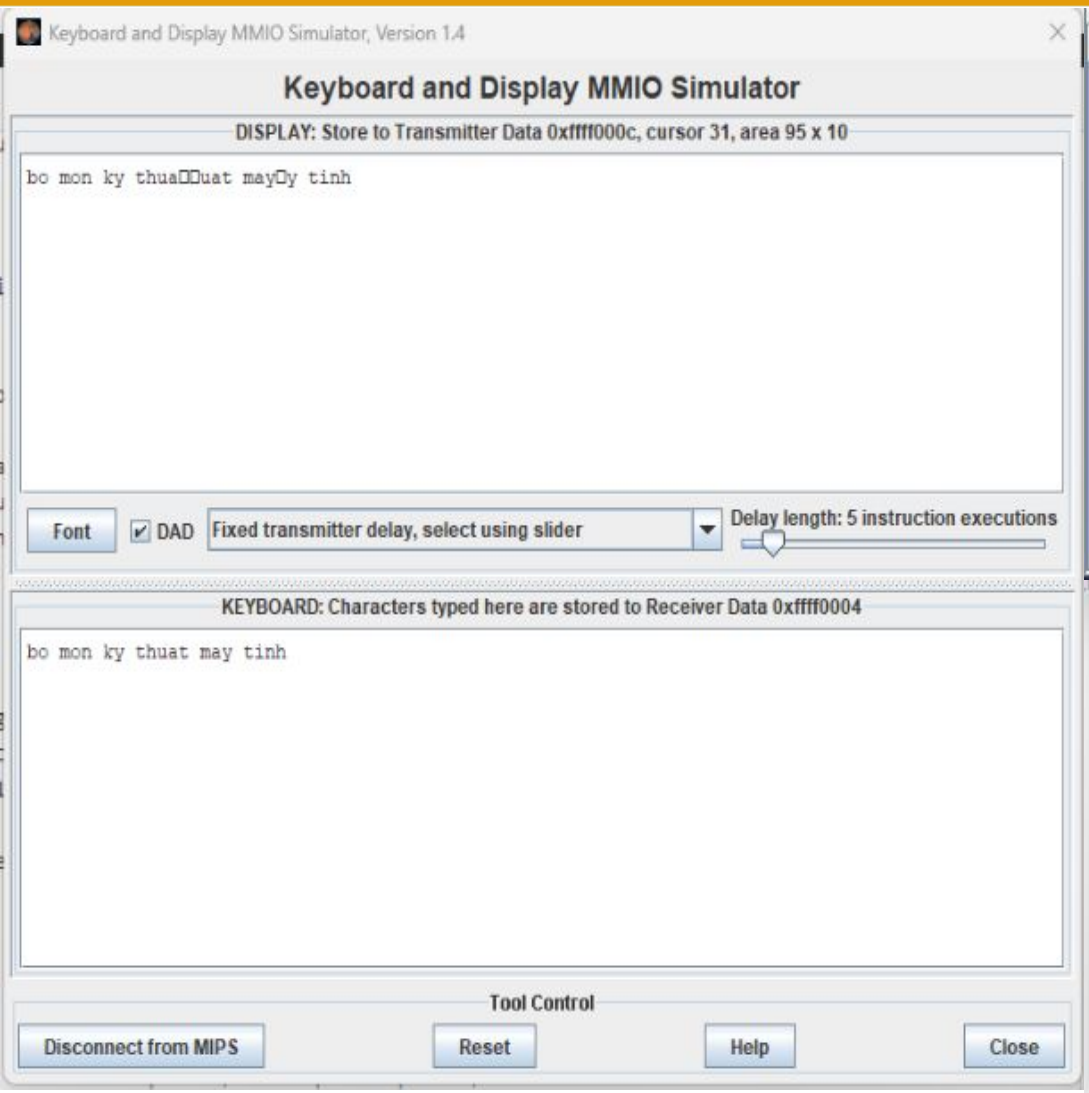
5. Print function to LED

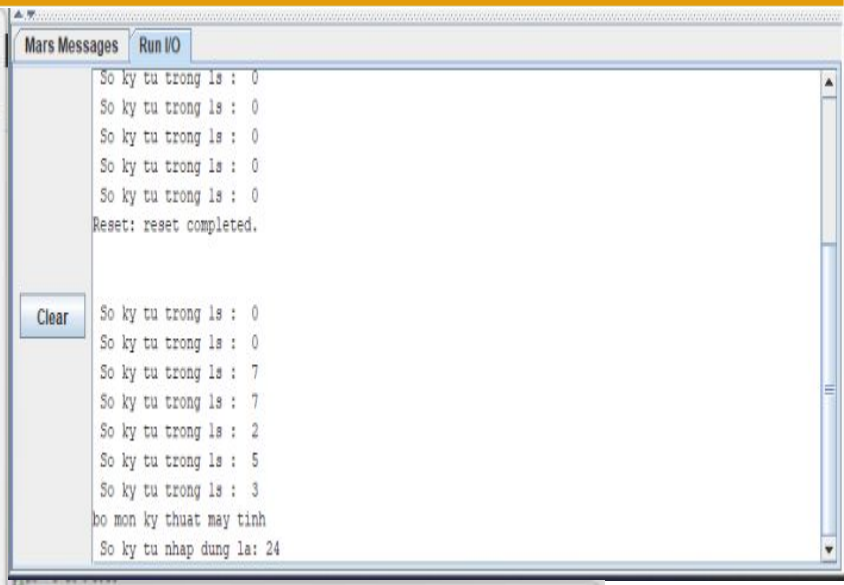
```
DISPLAY_DIGITAL:                        #Hien so chu nhap vao/1s tren 2 thanh led 7 thanh, ngoai
        div     $t6,$t4                 #lay so ky tu nhap duoc trong 1s chia cho 10
        mflo    $t7                     #luu gia tri phan nguyen, gia tri nay se duoc luu o den L
        la      $s2,byte_hex_led        #con tro toi array chua cac so hien thi tren led
        add     $s2,$s2,$t7             #xac dinh vi tri con tro =  cong them gia tri t7 vao con
        lb      $a0,0($s2)              #lay noi dung cho vao $a0
        jal     SHOW_7SEG_LEFT          #ngay den label den LED trai
#-------------------------------------------------------------------------------
        mfhi    $t7                     #luu gia tri phan du cua phep chia, gia tri nay se duoc i
        la      $s2,byte_hex_led
        add     $s2,$s2,$t7
        lb      $a0,0($s2)              #set value for segments
        jal     SHOW_7SEG_RIGHT         #show
#-------------------------------------------------------------------------------
        li      $t6,0                   #reset lai bien dem t6 trc kho quay l?i loop
        beq     $t9,1,ASK_LOOP          #bien t9 dung de kiem tra
```

```
SHOW_7SEG_LEFT:
        li      $t0,    SEVENSEG_LEFT   #gan dia chi cong nhu da khai bao
        sb      $a0,    0($t0)          #gan gia tri moi vao cong
        jr      $ra

SHOW_7SEG_RIGHT:
        li      $t0,    SEVENSEG_RIGHT
        sb      $a0,    0($t0)
        jr      $ra
```

Typing accuracy and speed check

Result





HUST

hust.edu.vn fb.com/dhbkhn

THANK YOU !