TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

TRƯỜNG CÔNG NGHỆ THÔNG TIN & TRUYỀN THÔNG

****

BÁO CÁO CUỐI KÌ

IT3280

THỰC HÀNH KIẾN TRÚC MÁY TÍNH

Nhóm sinh viên thực hiện: **Nhóm 6**

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

1. Hoàng Hương Giang - 20225619
2. Nguyễn Thị Huyền Trang - 20225674

MỤC LỤC

PHẦN 1: Hoàng Hương Giang - 20225619 3
1. Đề bài 3
2. Mã nguồn 4
3. Phân tích 21
3.1. Các bước thực hiện 21
3.2. Ý nghĩa các chuỗi được khai báo 22
3.3. Các thanh ghi sử dụng cố định 23
3.4. Ý nghĩa các chương trình con 23
4. Kết quả 31
PHẦN 2: Nguyễn Thị Huyền Trang - 20225674 35
1. Đề bài 35
2. Mã nguồn 36
3. Phân tích 44
3.1. Cách làm, thuật toán 44
3.2. Các thanh ghi sử dụng cố định 49
4. Kết quả 50

PHẦN 1: Hoàng Hương Giang - 20225619

1. Đề bài

Postscript CNC Marsbot

Máy gia công cơ khí chính xác CNC Marsbot được dùng để cắt tấm kim loại theo các đường nét được qui định trước. CNC Marsbot có một lưỡi cắt dịch chuyển trên tấm kim loại, với giả định rằng:

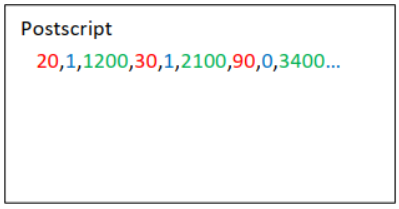
- Nếu lưỡi cắt dịch chuyển nhưng không cắt tấm kim loại, tức là Marsbot di chuyển nhưng không để lại vết (Track)
- Nếu lưỡi cắt dịch chuyển và cắt tấm kim loại, tức là Marsbot di chuyển và có để lại vết.

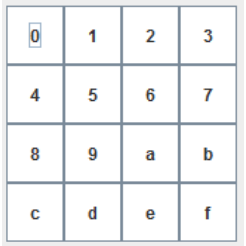
Để điều khiển Marsbot cắt đúng như hình dạng mong muốn, người ta nạp vào Marsbot một kịch bản là chuỗi ký tự bao gồm liên tiếp bộ 3 tham số (các giá trị phân cách nhau bởi dấu phẩy)

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

Hãy lập trình để CNC Marsbot có thể:

- Thực hiện cắt kim loại như đã mô tả
- Nội dung postscript được lưu trữ cố định bên trong mã nguồn
- 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.
- Một postscript chứa chữ DCE cần gia công. Hai script còn lại sinh viên tự đề xuất (tối thiểu 10 đường cắt)





2. Mã nguồn

```
.eqv  HEADING    0xffff8010

.eqv  MOVING     0xffff8050

.eqv  LEAVETRACK 0xffff8020

.eqv IN_ADDRESS_HEXA_KEYBOARD 0xFFFF0012

.eqv OUT_ADDRESS_HEXA_KEYBOARD 0xFFFF0014

.eqv  MASK_CAUSE_KEYMATRIX 0x00000800     # Bit 11: Key matrix interrupt

.data

        script0: .asciiz "161,1,3120,19,1,3120,90,0,1000,180,1,3000,90,0,1000,0,1,3000,90,1,1000,180,0,1500,270,1,1000,180,0,1500,90,1,1000,90,0,1500,0,1,3000,270,0,1000,90,1,2000,180,0,3000,90,0,1800,0,1,3000,147,1,3580,0,1,3000,90,0,2000,199,1,3150,90,0,2000,341,1,3150,199,0,2100,90,1,1330,161,0,1050,90,0,1200,0,1,3000,153,1,3200,27,1,3200,180,1,3000"

        script4: .asciiz "71,1,1700,37,1,1700,17,1,1700,0,1,1700,341,1,1700,320,1,1700,295,1,1700,180,1,8820,90,0,7000,270,1,2300,345,1,4520,15,1,4000,75,1,2500,90,0,2000,180,1,8820,90,1,2666,0,0,4410,270,1,2666,0,0,4410,90,1,2666"

        script8: .asciiz "180,1,4800,0,0,2400,90,1,2500,180,0,2400,0,1,4800,90,0,1200,180, 1, 3600, 157, 1,750,135,1,750,90,1,1000,45,1,750,23,1,750,0,1,3600,90,0,3600,180,0,700,315,1,1000,270,1,1000,225,1,1000,180,1,1000,135,1,1000,90,1,1000,135,1,1000,180,1,1000,225,1,1000,270,1,1000,315,1,1000,180,0,700,90,0,4700,0,1,4800,270,0,1500,90,1,3000"

        String0wrong: .asciiz "Postscript so 0 sai do "

        String4wrong: .asciiz "Postscript so 4 sai do "

        String8wrong: .asciiz "Postscript so 8 sai do "

        StringAllwrong: .asciiz "Tat ca Postscript deu sai"

        Reasonwrong1:      .asciiz "loi cu phap"
```

```
        Reasonwrong2:     .asciiz "thieu bo so"
        EndofProgram: .asciiz "Chuong trinh ket thuc!"
        ChooseAnotherScript: .asciiz "Vui long chon postscipt khac"
        NotCheck: .asciiz "Chua check xong. Xin hay doi mot lat"
        Done: .asciiz "Da cat xong!"
        Choose:       .asciiz "----------------------MENU----------------------\nVui long chon phim tren Digital Lab Sim\n0: VIETNAM\n4: DCE\n8: HUST\nc: Thoat chuong trinh"
        NotNormal: .asciiz "Xay ra loi bat thuong!"
        Array: .word

.text
main:       li $v0, 55
            la $a0, Choose
            li $a1, 1
            syscall
            li $t1, IN_ADDRESS_HEXA_KEYBOARD
            li $t2, OUT_ADDRESS_HEXA_KEYBOARD
            li $t3, 0x80 # bit 7 of = 1 to enable interrupt
            sb $t3, 0($t1)
            la $k0, Array
            li $s0, 4
            div $k0, $s0 #lay dia chi mang/4
            mfhi $s1 #gan s1 = dia chi mang mod 4
            beqz $s1, First_Ck #Neu s1 = 0=> dia chi mang chia het cho 4 => check
```

```
                sub $s0, $s0, $s1 #s0 = 4 - dia chi mang mod 4
                add $k0, $k0, $s0 #k0 = dia chi + (4 - dia chi mod4) => Dia chi o nho
tiep theo
First_Ck:       jal StringCheck
Loop:           nop
        addi $v0, $zero, 32
        li $a0, 200
        syscall
        nop
        nop
        b Loop # Wait for interrupt
        nop
        b Loop
end_of_main:        li $v0, 55
                la $a0, EndofProgram
                li $a1, 1
                syscall
                li $v0, 10
                syscall
#------------------------------
#StringCheck: Kiem tra du lieu dau vao
#a0: dia chi cac chuoi
#t7, t8, t9: giu gia tri 1 neu chuoi 0, 4, 8 sai
#a1: bit gia tri dung sai
#s0: dem so chuoi sai
#k0: dia chi mang
```

```
#-----------------------------
StringCheck:        li $s0, 0
SC_InSR:    addi  $sp,$sp,4    # Save $a0 because we may change it later
        sw    $ra,0($sp)
mainSC:             la $a0, script0
        jal Check
        addi $t7, $a0, 0 #t7 = gia tri dung/sai cua chuoi 0
        la $a0, String0wrong #Gan a0 = message khi chuoi 0 sai
        jal WrongMessage
        nop
Check_script4:      la $a0, script4
        jal Check
        addi $t8, $a0, 0 #t8 = gia tri dung/sai cua chuoi 4
        la $a0, String4wrong #Gan a0 = message khi chuoi 0 sai
                jal WrongMessage
                nop
Check_script8:      la $a0, script8
        jal Check
        addi $t9, $a0, 0 #t9 = gia tri dung/sai cua chuoi 8
        la $a0, String8wrong #Gan a0 = message khi chuoi 0 sai
                jal WrongMessage
                nop
                blt $s0, 3, SC_ResSR
                li $a1, 3
                la $a0, StringAllwrong
```

```
                jal WrongMessage
                j end_of_main
SC_ResSR: lw      $ra, 0($sp)     # Restore the registers from stack
        addi    $sp,$sp,-4
end_of_StringCheck:       addi $t6, $zero, 1 #luu t6 = 1 => da hoan thanh check
                        jr $ra
#------------------
WrongMessage:     li $v0, 59
                beq $a1, 0, end_of_WN
                beq $a1, 2, Reason2
                beq $a1, 3, Reason3
                la $a1, Reasonwrong1 #sai do ly do 1
                j call
Reason2:        la $a1,Reasonwrong2 #sai do ly do 2
                j call
Reason3:        li $v0, 55
                li $a1, 0
call:           addi $s0, $s0, 1
                syscall
end_of_WN:          jr $ra

#---------------
#Check: Kiem tra 1 chuoi co vi pham hay khong
#a0: dia chi ban dau cua script
#a1: Gia tri dung sai
```

```
#a2: byte duoc load
#a3: dem so dau phay
#v0: byte truoc byte hien tai
#Loi sai : Co chu hoac ky hieu ( 1), Khong du bo so(2)
#---------------
Check:          li $a3, 0 #gan a3 = 0
                lb $a2, 0($a0)
                beq $a2, 0x2C, wrong1 #khi ky tu dau tien la ',' thi sai
loop_Check:lb $a2, 0($a0)
                beq $a2, 0x2C, is_comma #Neu a2 = ',' thi thuc hien cong so dau phay
                beq $a2, 0x00, end_string #Neu a2 = /0 thi thuc hien ket thuc string
                beq $a2, 0x20, next_loop #neu a2 = dau cach thi bo qua
                blt $a2, 0x30, wrong1 #neu a2 < 30 => a2 la ky tu khac chu so => loi
                bgt $a2, 0x39, wrong1 #neu a2 > 39 => a2 la ky tu khac chu so => loi
                j next_loop
                nop
is_comma:   beq $v0, 0x2C, wrong1
                nop
                addi $a3, $a3, 1 #so dau phay cong 1
next_loop:  addi $a0, $a0, 1 #Tang a0 + 1 => chi den byte tiep theo
                addi $v0, $a2, 0 #v0 giu byte truoc
                j loop_Check
                nop
wrong1:         li $a1, 1 #gan a1 = 1, day sai do xuat hien chu hoac ky hieu
                li $a0, 1
```

```
                jr $ra #Quay ve ctr con goc
wrong2:         li $a1, 2 #a1= 2, day sai do thieu bo so
                li $a0, 2
                jr $ra
end_string:     beq $v0, 0x2C, wrong1 #Neu ky tu cuoi cung cua chuoi la , => sai
                li $a2, 3 #gan a2 = 3
                div $a3, $a2 #a3/3
                mfhi $a2 #a2 = a3 mod 3 = so dau phay mod 3
                bne $a2,2, wrong2 #neu a2 != a1 != 2 => so dau phay khong chia 3 du 2 => khong du bo so
                addi $a0, $k0, 0 #a1 = k0 => Chuoi dung va a0 chua dia chi mang cua chuoi dang xet
                addi $a3, $a3, 3 #a3= a3 + 1 + 2( A3 + 1 = so cac so co trong chuoi,
                #+1 de chua 1 o luu gia tri 0/1(da chuyen thanh mang/chua chuyen thanh mang)
                #+1 de chua 1 o luu gia tri ket thuc mang
                sll $a3, $a3, 2 #a3= a3*4
                add $k0, $k0, $a3 #k0 chi den dia chi moi de nhan vao chuoi tiep theo neu chuoi dung
                li $a2, -1
                sw $a2, -4($k0)
                li $a1, 0
                jr $ra
#------------------------
.ktext 0x80000180
Check_Cause:    mfc0  $t4, $13
```

```
            li    $t3, MASK_CAUSE_KEYMATRIX # if Cause value confirm Key..
        and   $at, $t4,$t3
        beq   $at,$t3, is_Check_ready #Neu ngat do ban phim thi tiep tuc check
ReasonNotNormal:        li $v0, 55
            la $a0, NotNormal #thong bao loi bat thuong
            li $a1, 0
            syscall
                li $v0, 10
                syscall
is_Check_ready:         beq $t6, 1, IntSR
                addi $sp, $sp, 4
                sw $v0, 0($sp)
                addi $sp, $sp, 4
                sw $a0, 0($sp)
                addi $sp, $sp, 4
                sw $a1, 0($sp)
                li $v0, 55
                la $a0, NotCheck #Chua check xong
                li $a1, 1
                syscall
                j ResValueforcheck
IntSR:
        li $t3, 0x81 # check hang 1: 0,1,2,3
        sb $t3, 0($t1) # Luu vao $t1
        lb $a0, 0($t2) # Doc phim
```

```
beq $a0, 0x11, Found_script0 #Neu chon 0 thi chay den Found_Script0
bne $a0, 0x00, PleaseAnother #Neu nhan duoc so khac 0 thi can doi postscript khac
li $t3, 0x82 # check hang 2: 4, 5, 6, 7
sb $t3, 0($t1)
lb $a0, 0($t2) # Doc
beq $a0, 0x12, Found_script4 #Neu chon 4 thi nhay den Found_script4
bne $a0, 0x00, PleaseAnother #Neu so khac 4 => Doi
li $t3, 0x84 # check hang 3: 8, 9, A, B
sb $t3, 0($t1)
lb $a0, 0($t2) # Doc phim
beq $a0, 0x14, Found_script8 #Neu chon 8 thi nhay den Found_script8
bne $a0, 0x00, PleaseAnother #Neu so khac 8 => Doi
li $t3, 0x88 # check hang 4: C, D, E, F
sb $t3, 0($t1)
lb $a0, 0($t2) # Doc phim
beq $a0, 0x18, end_of_main #neu chon c thi ket thuc ctr
bne $a0, 0x00, PleaseAnother
beq $a0, 0x00, ReasonNotNormal #Khong nhan duoc gia tri nao => loi bat thuong
#---------------------------
#a1: Luu dia chi mang neu chuoi dung, 1/2 neu chuoi sai => Lay du lieu tu $t7, $t8,$t9
#a0: Gan dia chi scriptXwrong(duoc goi neu chuoi sai)
#s3: Gan dia chi chuoi
#---------------------------
```

```
Found_script0: add $a1, $zero, $t7 #luu dia chi mang 0(neu dung ) tu t7 vao a1
               la $a0, String0wrong #gan a0 dia chi thong bao neu chuoi 0 sai
               la $s3, script0 #Gan s3 dia chi cua chuoi
               j Found #Nhay den ham xu ly
Found_script4: add $a1, $zero, $t8
               la $a0, String4wrong
               la $s3, script4
               j Found
Found_script8: add $a1, $zero, $t9
               la $a0, String8wrong
               la $s3, script8
Found:         beq $a1, 1, WrongScript
          beq $a1, 2, WrongScript #Neu a1 = 1 hoac 2 thi chuoi sai => Nhay den
WrongScript
          addi $a0, $s3, 0 #nap dia chi stringX vao $a0
          lw $s1, 0($a1)
          addi $a2, $a1, 0 #luu dia chi lai vao a2 do thanh ghi a1 bi thay doi khi goi
den StringSolve
          bne $s1, 0, StringRun #Nếu chuỗi chưa chuyển thành số thì nhảy đến hàm
chuyển #Nếu không thì chạy SCRIPT
          jal StringSolve
StringRun:   addi $s0, $a2, 4 #Chay mang bat dau tu phan tu t2
               jal MarsbotControl #Trinh dieu khien Marsbot
               j next_pc
WrongScript:        li $v0, 59
               beq $a1, 2, WS_Reason2
```

```
                la $a1, Reasonwrong1 #sai do ly do 1
                j WS_call
WS_Reason2:     la $a1,Reasonwrong2 #sai do ly do 2
WS_call:    syscall #voi a0 da duoc gan la thong bao chuoi sai tu truoc
#-----------------------------------
PleaseAnother:          li $v0, 55
                        la $a0, ChooseAnotherScript
                        li $a1, 1
                        syscall #In thong bao xin chon vi tri khac
                        j return
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
            beq $t6, 0, ResValueforcheck
            j return
ResValueforcheck:lw $a1, 0($sp)
                        addi $sp, $sp, -4
                        lw $a0, 0($sp)
                        addi $sp, $sp, -4
                        lw $v0, 0($sp)
                        addi $sp, $sp, -4
return:     eret # Return from exception

#-----------------
#StringSolve: Xu ly bien doi chuoi thanh so
```

```
#a0: dia chi chuoi(Tham so truyen vao)
#a1: dia chi mang (Tham so truyen vao)
#s0: byte duoc load
#s1: dem so truoc ','
#s2: So da duoc xu ly
#s3: 10
#s4: Dem tu 1 - s1 khi chuyen so
#s5: 10^i
#------------------
StringSolve:
        li $s0, 1 #Gan gia tri s0 khac 0 de bat dau ctr
        li $s3, 1
        sw $s3, 0($a1)      #Luu bit 1 vao pt dau tien mang de xac dinh chuoi da duoc chuyen
        addi $a1, $a1, 4 #Luu gia tri tu pt thu 2
mainSS:         li $s3, 10
        li $s2, 0
        li $s1, 1 #Bien dem bat dau tu 1
        li $s5, 1 #luu s5 = 10^0
SS_loop:    lb $s0, 0($a0) #s0 = byte duoc xet
        beq $s0, 0x20, SS_nextbyte #Neu gap dau ' ' thì bỏ qua
        beq $s0, 0x2C, Into_Array       #Neu gap dau ',' thi chuyen thanh so
        beq $s0, 0x00, Into_Array       #Neu gap ket thuc chuoi thì thuc hien chuyen thanh so lan cuoi
        addi $sp, $sp, 1
        sb $s0, 0($sp) #luu byte vao stack
```

```
                addi $s1, $s1, 1 #dem so chu so cua so do
SS_nextbyte:    addi $a0, $a0, 1
                j SS_loop
Into_Array:     li $s4, 1
                addi $v0, $s0, 0 #luu byte dau hieu dan toi chuyen sang so (0x2C,
0x00). Voi 0x00 thi de ket thuc ctr
Into_loop:      beq $s4, $s1, SaveArray #Neu da du so cac chu so thi luu vao mang
                lb $s0, 0($sp)
                addi $sp, $sp, -1
                addi $s0, $s0, -48 #Doi gia tri $s0 sang so
                mult $s0, $s5
                mflo $s0 #s0 = s0*10^i
                add $s2, $s0, $s2 #s2 + s0
                #Next
                mult $s5, $s3 #s5 nhan 10 sau moi lan chuyen
                mflo $s5 #s5 = 10^(i+1)
                addi $s4, $s4, 1
                j Into_loop
SaveArray:      sw $s2, 0($a1)
                add $a1, $a1, 4
                addi $a0, $a0, 1
                beq $v0, 0x00, end_of_StringSolve #neu v0 = 0x00 => ket thuc cau
                j mainSS
#_______________
end_of_StringSolve:     jr $ra
#---------------------------
```

```
#MarsbotControl   : Trinh dieu khien Marsbot
#s0: dia chi mang ( luu truoc khi vao ctrinh con)
#a1: rotate - goc xoay
#a0: tg chay
#a2: bit track - untrack
#------------------------
MarsbotControl:
MB_InSR:  addi  $sp,$sp,4    # Save $ra because we may change it later
        sw    $ra,0($sp)
FirstRun:     li $a1, 165
              li $a0, 10000
              jal ROTATE
              nop
              jal GO
              nop
              addi    $v0,$zero,32    # Dua Marsbot ra giua man hinh de de nhin hon
        syscall
TakeData:   lw $a1, 0($s0) #Load goc xoay
              addi $s0, $s0, 4
              beq $a1, -1, MB_EndScript #Neu gia tri load duoc = -1 => ket thuc
              lw $a2, 0($s0) #load bit track/untrack
              addi $s0, $s0, 4
              lw $a0, 0($s0) # Load tg chay
              addi $s0, $s0, 4
MB_Run:             jal ROTATE
```

```
                nop
                beq $a2, 0, Leave   #Neu a2 = 0 => Khong luu lai vet => thuc hien di
chuyen luon
                jal     TRACK           # and draw new track line
        nop
Leave:                  addi    $v0,$zero,32
        syscall         #a0 la tham so thoi gian
                jal     UNTRACK         # keep old track
        nop
MB_nextData:        j TakeData  #Tiep tuc lay du lieu tu mang
MB_EndScript:       jal STOP #Dung
MB_ResSR:lw         $ra, 0($sp)     # Restore the registers from stack
        addi    $sp,$sp,-4
        li $v0, 55
                la $a0, Done
                li $a1, 1
                syscall #Thong bao da ve xong
end_of_MarsbotControl: jr $ra


#-----------------------------------------------------------
# GO procedure, to start running
# param[in]    none
#-----------------------------------------------------------
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 procedure, to stop running
# param[in]    none
#-----------------------------------------------------------
STOP:   li    $at, MOVING     # change MOVING port to 0
        sb    $zero, 0($at)   # to stop
        nop
        jr    $ra
        nop
#-----------------------------------------------------------
# TRACK procedure, to start drawing line
# param[in]    none
#-----------------------------------------------------------
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 procedure, to stop drawing line
# param[in]    none
```

```
#-----------------------------------------------------------
UNTRACK:li    $at, LEAVETRACK # change LEAVETRACK port to 0
      sb    $zero, 0($at)   # to stop drawing tail
      nop
      jr    $ra
      nop
#-----------------------------------------------------------
# ROTATE procedure, to rotate the robot
# param[in]    $a1, 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    $a1, 0($at)     # to rotate robot
      nop
      jr    $ra
      nop
```

3. Phân tích

3.1. Các bước thực hiện

Bước 1: Thông báo MENU ra màn hình, người dùng có thể chọn 1 trong 4 phím trên Digital Lab Sim để chạy Marsbot cắt hình tương ứng. Chọn:

0: Cắt hình VIETNAM

4: Cắt hình DCE

8: Cắt hình HUST

C: Thoát chương trình

Lưu ý: *Ấn các phím khác Marsbot sẽ thông báo chọn lại.*

Bước 2: Kiểm tra các postscript có hợp lệ hay không, nếu không hợp lệ chương trình thông báo chuỗi không hợp lệ theo lỗi bằng popup trên màn hình phần mềm MARS. Nếu tất cả postscript đều không hợp lệ thì thông báo và kết thúc chương trình.

Bước 3: Đợi người dùng nhấn phím trên Digital Lab Sim.

Bước 4: Sau khi người dùng nhấn phím trên Digital Lab Sim, chương trình ngắt sẽ được thực hiện. Có nhiều kiểu ngắt:

- Ngắt do ấn phím trên Digital Lab Sim:
    - Trường hợp chương trình chưa kiểm tra postscript xong sẽ thông báo ra màn hình và quay lại chương trình chính tiếp tục kiểm tra.
    - Trường hợp đã kiểm tra postscript xong rồi chương trình sẽ kiểm tra phím nào được ấn và chạy postscript tương ứng với phím đấy. Nếu phím được bấm không thuộc các phím 0,4,8,c thì sẽ hiện thông báo "Vui lòng chọn postscript khác" và quay lại chương trình chính
- Ngắt trong trường hợp khác: Lỗi bất thường, thông báo ra màn hình rồi kết thúc chương trình.

Bước 5: Khi chạy postscript:

- Postscript được ấn nếu *sai* chương trình sẽ:
    - Thông báo ra màn hình lỗi sai và thông báo chọn phím khác.
    - Quay về chương trình chính đợi người dùng ấn phím khác.
- Postscript được ấn nếu *đúng* chương trình sẽ:
    - Kiểm tra xem postscript đã chuyển thành mảng hay chưa:

        - Chưa chuyển: chương trình sẽ gọi đến hàm chuyển chuỗi thành mảng.
        - Đã chuyển: chương trình sẽ di chuyển Marsbot theo postscript đã chọn.
    - Thông báo ra màn hình đã cắt xong hình.
    - Quay về chương trình chính đợi người dùng ấn phím khác.

Bước 6: Tiếp tục lặp lại các bước **3** đến **5** cho đến khi người dùng ấn phím C. Hiện thông báo kết thúc chương trình và kết thúc.

3.2. Ý nghĩa các chuỗi được khai báo

Tên chuỗi	Ý nghĩa
script0	Postscript tương ứng với phím 0 trên Digital Lab Sim
script4	Postscript tương ứng với phím 4 trên Digital Lab Sim
script8	Postscript tương ứng với phím 8 trên Digital Lab Sim
StringXWrong	Với X nhận các giá trị 0,4,8, kết hợp với Reasonwrong1 và Reasonwrong2 thông báo postscript số X sai và lý do ra màn hình
StringAllWrong	Thông báo tất cả postscript đều sai
Reasonwrong1	Thông báo lỗi sai do cú pháp(hai dấu phẩy, ký tự khác số hoặc dấu phẩy, dấu phẩy ở đầu chuỗi,…)
Reasonwrong2	Thông báo lỗi sai do thiếu bộ số (không đủ 3n số)
EndofProgram	Thông báo kết thúc chương trình
ChooseAnotherScript	Thông báo chọn script khác
NotCheck	Thông báo chưa check xong postscript
Done	Thông báo đã cắt hình xong
Choose	Hiện MENU những phím có thể chọn trên Digital Lab Sim và ý nghĩa của chúng.
NotNormal	Thông báo nếu có lỗi bất thường xảy ra
Array	Mảng lưu các giá trị số từ chuỗi chuyển thành. Các mảng tương ứng các chuỗi được lưu liên tiếp nhau.

3.3. Các thanh ghi sử dụng cố định

Thanh ghi	Vai trò
$t1	Lưu địa chỉ IN_ADDRESS_HEXA_KEYBOARD
$t2	Lưu địa chỉ OUT_ADDRESS_HEXA_KEYBOARD
$t6	Đánh dấu đã hoàn thành quá trình check tất cả các postscript hay chưa.
$t7	Bằng 1 hoặc 2 nếu postscript 0 lỗi tùy theo lỗi sai Là địa chỉ mảng nếu postscript 0 đúng
$t8	Bằng 1 hoặc 2 nếu postscript 4 lỗi tùy theo lỗi sai Là địa chỉ mảng nếu postscript 4 đúng
$t9	Bằng 1 hoặc 2 nếu postscript 8 lỗi tùy theo lỗi sai Là địa chỉ mảng nếu postscript 8 đúng

3.4. Ý nghĩa các chương trình con

StringCheck: Kiểm tra dữ liệu đầu vào

Input: Địa chỉ các postscript 0, 4, 8

Output:

- $t7, $t8, $t9:
  - Là 1 hoặc 2 nếu postscript 0, 4, 8 sai (tùy lỗi);
  - Là địa chỉ của mảng số nếu postscript đúng.
- Thông báo ra màn hình nếu postscript sai (Kết thúc chương trình nếu tất cả các postscript đều sai).

Các thanh ghi sử dụng:

- a0: Chứa địa chỉ các postscript ban đầu
  Sau khi chạy hàm *Check* thì bằng 1 hoặc 2 nếu postscript sai, là địa chỉ mảng nếu postscript đúng.
- a1: Chứa giá trị đúng/sai của postscript:
  0: đúng
  1: lỗi cú pháp
  2: lỗi thiếu bộ số
  3: tất cả các postscript đều sai
- t7, t8, t9: Postscript sai: Là 1 hoặc 2 (tùy lỗi)
  Postscript đúng: Là địa chỉ của mảng số

- s0: Đếm số postscript sai
- $k0: Địa chỉ mảng tương ứng.

Giải thích:

- Đầu tiên khởi tạo số postscript sai bằng 0 ($s0 = 0)
- Rồi lưu lại địa chỉ trở về ở thanh ghi $ra vào stack (do trong hàm này sẽ gọi đến hàm khác)
- Tiếp theo là kiểm tra script 0:
    - Gọi đến hàm *Check* với đầu vào là địa chỉ postsrcipt 0 được lưu trong $a0.
    - Lấy output $a0 của hàm Check gán cho $t7
    - Gán $a0 bằng địa chỉ của thông báo String0wrong.
    - Kiểm tra nếu $a1 ≠ 0 thì gọi đến hàm *WrongMessage* để thông báo lỗi sai với postscript sai được lưu trong $a0.
    - Tiếp tục kiểm tra postscript 4
- Tiếp đến là kiểm tra script 4:
    - Gọi đếm hàm *Check* với đầu vào là địa chỉ postsrcipt 4 được lưu trong $a0.
    - Lấy output $a0 của hàm Check gán cho $t8
    - Gán $a0 bằng địa chỉ của thông báo String0wrong.
    - Kiểm tra nếu $a1 ≠ 0 thì gọi đến hàm *WrongMessage* để thông báo lỗi sai với postscript sai được lưu trong $a0.
    - Tiếp tục kiểm tra postscript 8
- Kiểm tra script 8:
    - Gọi đếm hàm *Check* với đầu vào là địa chỉ postsrcipt 8 được lưu trong $a0.
    - Lấy output $a0 của hàm Check gán cho $t9
    - Gán $a0 bằng địa chỉ của thông báo String0wrong.
    - Kiểm tra nếu $a1 ≠ 0 thì gọi đến hàm *WrongMessage* để thông báo lỗi sai với postscript sai được lưu trong $a0.
    - Kiểm tra xem có phải tất cả các postscript đều sai hay không(với $s0 = 3 nếu tất cả đều sai), nếu đúng thì gán $a1 = 3 rồi gọi đến hàm *WrongMessage* để thông báo lỗi ra màn hình và nhảy đến end_of_main kết thúc chương trình.
- Khôi phục địa chỉ trả về vào thanh ghi $ra, gán $t6 = 1 (Đánh dấu đã check xong) rồi quay về chương trình chính.

WrongMessage: Thông báo lỗi sai

Input:

- Postscript sai ($a0)
- Lỗi sai của postscript đó ($a1)

Output: Thông báo postscript sai và lỗi sai ra màn hình

Các thanh ghi sử dụng:

- $a0: Chứa postscript sai (0, 4 hoặc 8)
- $a1: Chứa giá trị sai của postscript

  1: lỗi cú pháp

  2: lỗi thiếu bộ số

  3: tất cả các postscript đều sai
- $s0: Đếm số postscript sai.

Giải thích:

- Đầu tiên kiểm tra xem nếu $a1 = 0 (postscript đúng) thì nhảy đến end_of_WN và kết thúc chương trình con WrongMessage.
- Nếu $a1 = 1 thì thông báo "Postscript so X sai do loi cu phap" ra màn hình.
- Nếu $a1 = 2 thì thông báo "Postscript so X sai do thieu bo so" ra màn hình.
- Nếu $a1 = 3 thì thông báo "Tat ca Postscript deu sai" ra màn hình.
- Nếu postsrcipt sai thì tăng $s0 lên 1.
- Kết thúc hàm *WrongMessage*.

Check: Kiểm tra 1 postscript có hợp lệ hay không

Input: Địa chỉ chuỗi ($a0)

Output:

- $a0: Postscript đúng: là địa chỉ mảng của chuỗi đang xét

  Postscript sai: bằng 1 hoặc 2(tùy lỗi)
- $a1: Postscript đúng: bằng 0

  Postscript sai: bằng 1 hoặc 2(tùy lỗi)

Các thanh ghi sử dụng:

- a0: Địa chỉ ban đầu của postscript, gọi là s
- a1: 0: Postscript đúng

  1: Lỗi cú pháp

  2: Lỗi thiếu bộ số
- a2: Giá trị đang xét trên postscript, tương ứng là s[i]
- v0: Giữ giá trị trước của a2, tương ứng là s[i-1]
- a3: Đếm số dấu phẩy
- s0: Đếm số chuỗi sai
- k0: Địa chỉ mảng của postscript

Giải thích:

- Coi $a0 như s, $a2 như s[i], $v0 như s[i-1].
- Nếu s[0] = ‘,’ là lỗi cú pháp → nhảy đến *wrong1*.
- Vòng lặp:
  - Load s[i] vào $a2 từ địa chỉ $a0.
  - Nếu s[i] = ‘,’ thì kiểm tra xem s[i-1] = ‘,’ hay không. Nếu có (*2 dấu phẩy liên tiếp*) là lỗi cú pháp → nhảy đến *wrong1*.
  - Nếu s[i] = ‘ ’ (dấu cách) thì bỏ qua.
  - Nếu s[i] = ‘\0’ (NULL) thì:
    - Kiểm tra nếu ký tự cuối cùng s[i-1] = ‘,’ là lỗi cú pháp → nhảy đến *wrong1*.
    - Kiểm tra nếu số lượng dấu phẩy không phải 3n + 2 là thiếu bộ số → nhảy đến *wrong2*.
    - Nếu không phải 1 trong 2 lỗi trên thì postscript đúng:
      - Gán $a1 = 0.
      - Gán $a0 = địa chỉ mảng postscript đang xét lấy từ $k0.
    - Tăng $a3 lên 3 vì:
      - Số dấu phẩy + 1 = Số lượng số của postscript.
      - Dành vị trí đầu tiên để đánh dấu postscript đã chuyển thành mảng hay chưa.
      - Dành vị trí cuối cùng để đánh dấu kết thúc mảng.

      → Khi đó $a3 sẽ thành số phần tử của mảng.
    - Gán $k0 = địa chỉ mảng của postscript tiếp theo (Ngay sau phần tử cuối cùng mảng trước).

        - Lưu $a2 = -1 vào vị trí cuối cùng của mảng để đánh dấu kết thúc mảng.
        - Kết thúc hàm *Check*.
    - Nếu s[i] < '0' hay s[i] > '9' thì s[i] đang là một ký tự không hợp lệ, là lỗi cú pháp → nhảy đến *wrong1*.
    - Nếu không có lỗi nào thì trỏ đến phần tử tiếp theo của chuỗi ($a0++).
- Lặp lại vòng lặp cho đến khi gặp NULL hoặc lỗi.
- Giải thích wrong1 và wrong2:
    - wrong1: Gán $a1, $a0 = 1(*lỗi cú pháp*) rồi quay lại hàm *StringCheck*.
    - wrong2: Gán $a1, $a0 = 2(*lỗi thiếu bộ số*) rồi quay lại hàm *StringCheck*.

Check_Cause: Kiểm tra nguyên nhân gây ra ngắt

Giải thích:

- Đầu tiên kiểm tra nếu không phải ngắt do nhấn phím trên Digital Lab Sim thì thông báo xảy ra lỗi bất thường rồi thoát chương trình.
- Nếu là ngắt do nhấn phím trên Digital Lab Sim nhảy đến *is_Check_ready* kiểm tra $t6. Nếu $t6 ≠ 1 (chưa check xong) thì thông báo ra màn hình r quay lại chương trình chính.
  Lưu ý: Trước khi thông báo phải lưu lại các thanh ghi $v0, $a0, $a1 vào stack (do chương trình chính cũng sử dụng những thanh ghi này) và thông báo xong thì khôi phục lại những thanh ghi này.
- Nếu check postscript xong rồi thì nhảy đến *IntSR* để kiểm tra phím được nhấn.

IntSR: Kiểm tra phím được nhấn trên Digital Lab Sim

Giải thích: Kiểm tra từng hàng một:

- Nếu các phím 0,4,8 được ấn thì sẽ nhảy đến *Found_scriptX* với X = 0,4,8 gán $a1 bằng thanh ghi lưu địa chỉ mảng tương ứng từng chuỗi($t7, $t8, $t9), chuỗi StringXwrong vào $a0, địa chỉ chuỗi vào $s3. Tiếp tục nhảy đến *Found*
  - *Found:*
    - Nếu $a1 bằng 1 hoặc 2 (chuỗi sai) thì nhảy đến *WrongScript* thông báo lỗi sai và kết thúc chương trình con, quay lại chương trình chính đợi người dùng bấm phím khác.
    - Kiểm tra giá trị phần tử đầu tiên của mảng tương ứng của chuỗi. Nếu s1 = 0 thì chuỗi chưa chuyển sang mảng → Gọi hàm *StringSolve*. Ngược lại, nếu s1 = 1 thì chuỗi đã chuyển → nhảy đến *StringRun*.

- *StringRun:* Tăng địa chỉ được lưu trong $s0(Do mảng số được chuyển từ chuỗi được lưu từ phần tử thứ 2). Gọi hàm *MarsbotControl* và sau đó kết thúc chương trình con, trở về chương trình chính đợi người dùng ấn phím khác.

- Nếu phím C được ấn thì nhảy đến *end_of_main* và kết thúc chương trình.
- Nếu các phím khác được bấm sẽ nhảy đến *PleaseAnother*, thông báo "Vui lòng chọn Postscript khác" và quay lại chương trình chính.
- Nếu không nhận được phím nào ($a0 = 0x00) thì xảy ra lỗi bất thường, nhảy đến *ReasonNotNormal*, thông báo ra màn hình và kết thúc chương trình.

StringSolve: Biến chuỗi thành mảng số

Input: $a0: Địa chỉ chuỗi cần chuyển

$a1: Địa chỉ mảng của chuỗi đó

Output: Mảng đã chuyển xong (phần tử đầu tiên đổi từ 0 thành 1)

Các thanh ghi sử dụng:

- $a0: Địa chỉ chuỗi, gọi là s
- $a1: Địa chỉ mảng
- $s0: ký tự, gọi là s[k]
- $s1: Đếm số chữ số của 1 số
- $s2: Giá trị số (từ chuỗi chuyển sang)
- $s3: 10
- $s5: 10^i (bắt đầu từ 10^0 rồi tăng dần lên)

Ý tưởng:

Khi người dùng nhấn vào phím 0/4/8 *lần đầu tiên*, postscript tương ứng với phím được nhấn sẽ được chuyển thành mảng bằng cách gọi đến hàm này (Địa chỉ mảng đã được xác định (nếu postscript đúng) khi chạy hàm *StringCheck*). Những lần ấn sau không cần chạy hàm này nữa.

Giải thích:

- Đầu tiên gán a[0] = 1 (Đánh dấu chuỗi đã chuyển thành mảng).
- Khởi tạo các giá trị:
  - $s3 = 10 (để nhân 10 trong *Into_loop*).
  - $s2 = 0 (Giá trị số cuối cùng sau khi chuyển từ các ký tự sang).
  - $s1 = 0 (Đếm số chữ số).

    - $s5 = 1 ($10^k$ với k bắt đầu từ 0).
- Vòng lặp (*SS_loop*): *Dùng để đếm số chữ số của 1 số đằng trước dấu phẩy*
    - Lấy s[i] lưu vào $s0.
    - Nếu s[i] = ‘ ’ (dấu cách) thì bỏ qua.
    - Nếu s[i] = ‘,’ thì bắt đầu chuyển những ký tự số trước đó thành chuỗi. → nhảy đến *Into_Array.*
    - Nếu s[i] = ‘\0’ (NULL) thì chuyển những ký tự số trước đó thành chuỗi → nhảy đến *Into_Array* rồi kết thúc hàm *StringSolve*.
    - Nếu không phải những trường hợp trên thì s[i] thuộc [‘0’, ‘9’]. Lưu s[i] vào stack.
    - Tăng số chữ số lên 1 ($s1++).
    - Trỏ đến phần tử tiếp theo của chuỗi ($a0++).
- Lặp lại vòng lặp cho đến khi gặp dấu phẩy hoặc NULL.
- *Into_Array: Chuyển từng ký tự thành số rồi cộng vào thành 1 số hoàn chỉnh*
    - Gán $s4 = 1(Biến đếm số chữ số của số đang được chuyển)
    - Gán $v0 = s[i] (để kiểm tra ký tự dẫn tới chuyển sang số)
    - Vòng lặp (*Into_loop*):
        - Lấy từng ký tự ra khỏi stack (Lấy từ hàng đơn vị rồi tăng dần lên) gán cho $s0.
        - Chuyển $s0 từ ký tự thành số bằng cách trừ đi ‘0’(48).
        - Nhân $s0 với 10^k ($s5) (k bắt đầu từ 0 rồi tăng dần lên).
        - Cộng tất cả vào $s2.
        - Rồi nhân $s5 với 10 ($10^{k+1}$).
        - Tăng biến đếm $s4 lên 1.
        - Và lặp lại vòng lặp cho đến khi số chữ số $s4 = $s1.
- Sau khi chuyển ký tự thành số xong (số chuyển xong được lưu trong $s2) ta phải lưu số đó vào mảng.
- Sau đó trỏ mảng và chuỗi đến phần tử tiếp theo.
- Lúc này nếu $v0 = ‘\0’ (NULL) thì kết thúc hàm *StringSolve*. Nếu không thì tiếp túc lặp lại từ khởi tạo các giá trị cho đến khi gặp ký tự NULL.

MarsbotControl: Chạy Script

Input:

- $s0: Địa chỉ mảng tương ứng chuỗi được chọn tăng thêm 4, trỏ tới phần tử thứ 2 trong mảng

Output:

- Hình được cắt xong trên giao diện Marsbot

Các thanh ghi sử dụng:

- $a1: giá trị góc quay
- $a2: giá trị quyết định có lưu vết hay không
- $a0: Thời gian chạy

Giải thích:

- Lưu $ra vào stack (Do trong hàm này gọi đến hàm khác).
- Tiếp theo quay Marsbot 165° rồi cho nó chạy 10s đưa Marsbot cách lề để thực hiện cắt hình.
- Vòng lặp:
    - Lưu lần lượt các giá trị góc xoay, giá trị điều khiển lưu vết, thời gian chạy vào các thanh ghi $a1, $a2, $a0.
    - Nếu $a1 = -1 (đã lấy hết giá trị trong mảng) thì kết thúc cắt (Dừng Marsbot và thông báo đã cắt xong ra màn hình) rồi khôi phục $ra và kết thúc hàm *MarsbotControl*.
    - Nếu không thì quay Marsbot theo góc ($a1).
    - Nếu không lưu vết ($a2 = 0) thì nhảy đến Leave và thực hiện cắt với tham số thời gian lưu tại $a0. Nếu lưu vết thì gọi hàm *TRACK* và sau đó thực hiện tương tự.
    - Gọi hàm *UNTRACK* (Nếu không cắt thì vẫn *UNTRACK*).
- Quay lại vòng lặp cho đến khi $a1 = -1.

GO: Di chuyển Marsbot

STOP: Dừng Marsbot

TRACK: Để lại vết

UNTRACK: Không để lại vết

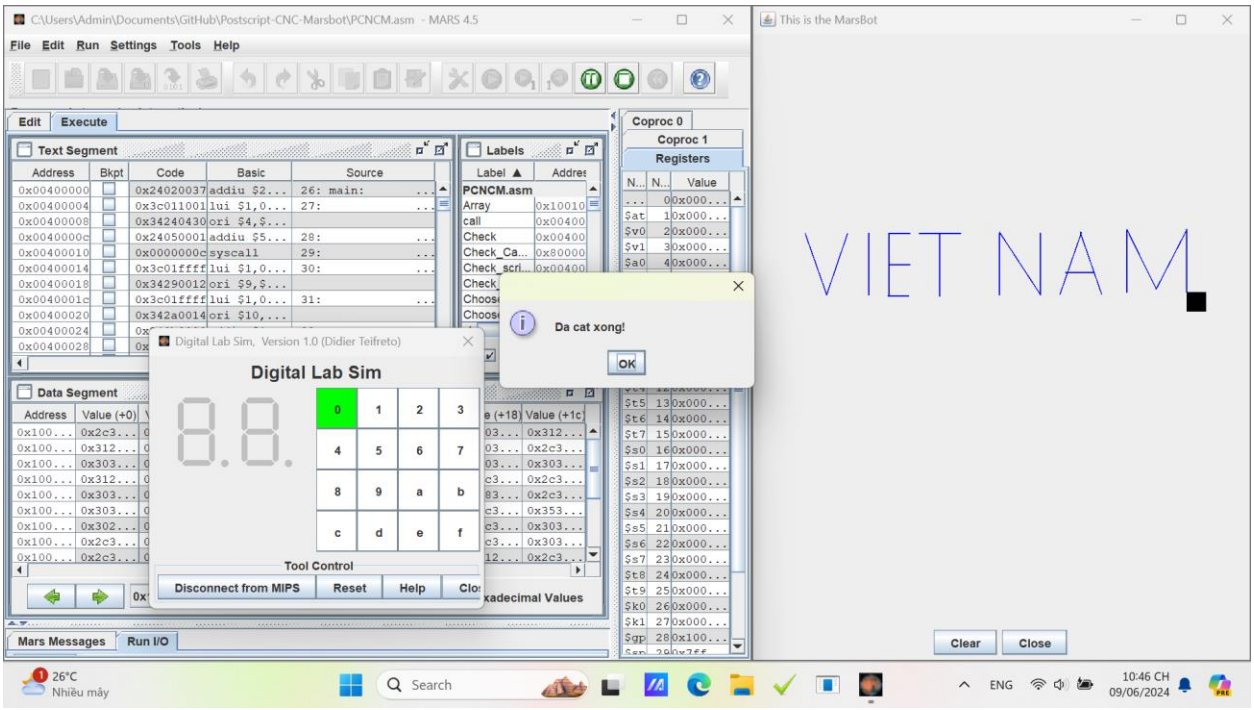
ROTATE: Quay Marsbot

Input: $a1: Góc quay từ 0 đến 359 với:

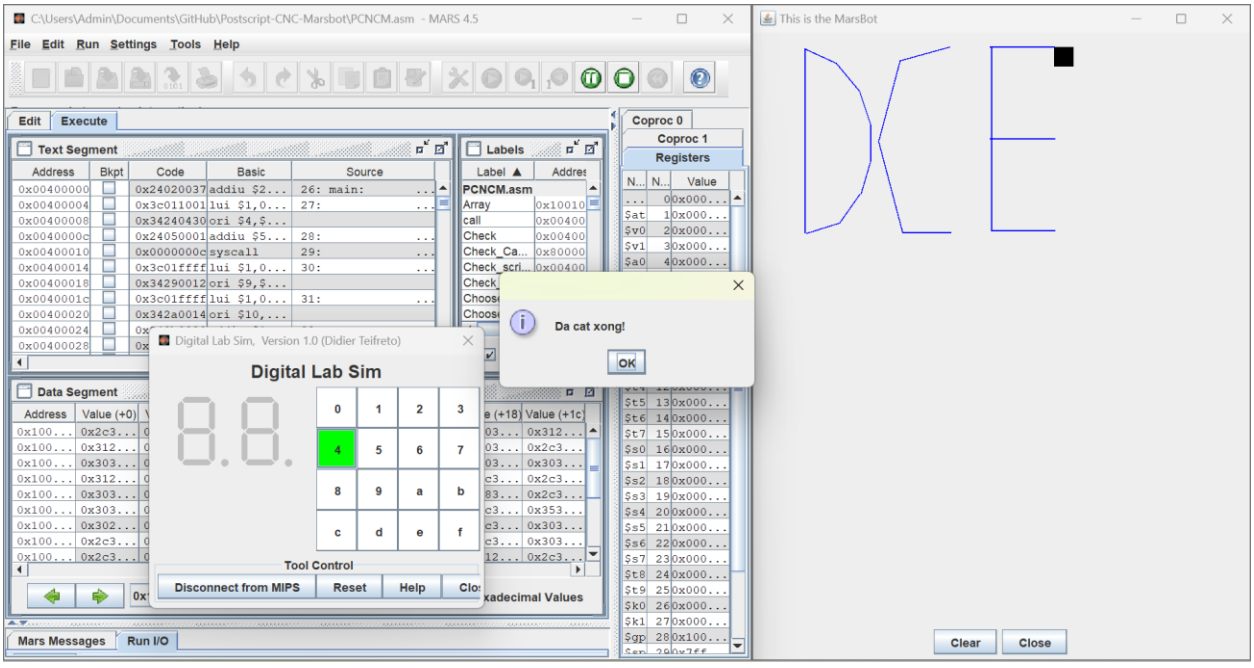
- 0 : North (up)
- 90: East (right)
- 180: South (down)
- 270: West (left)

4. Kết quả

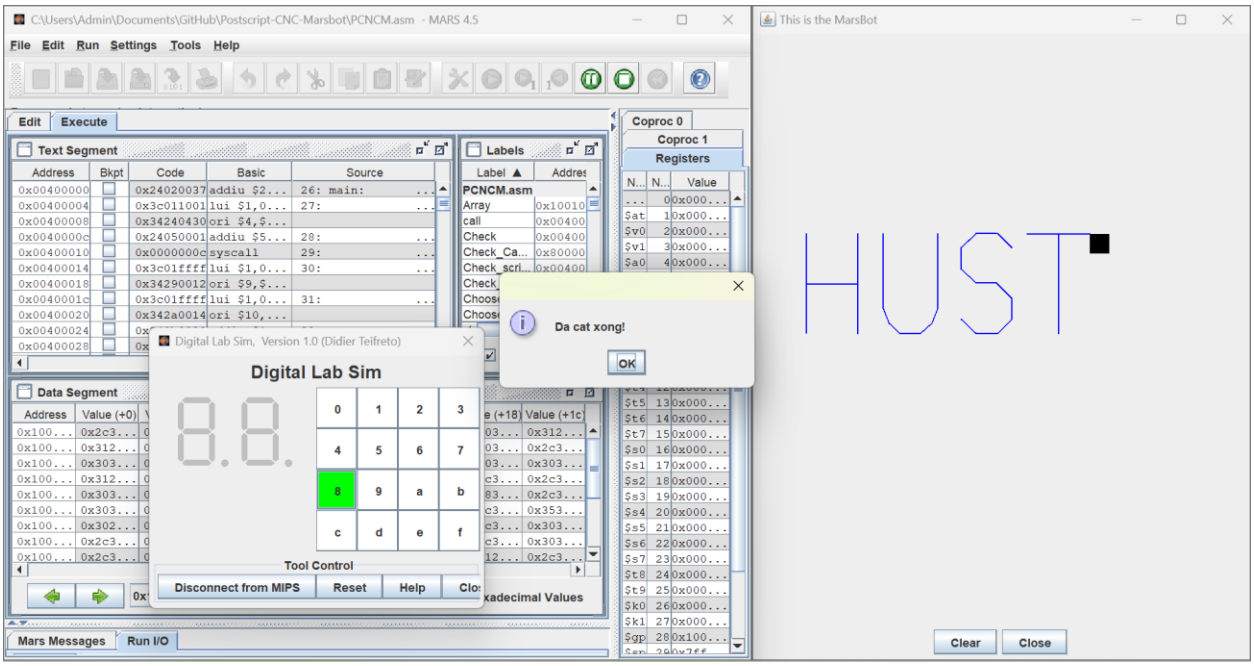
a. Postscript 0: VIETNAM



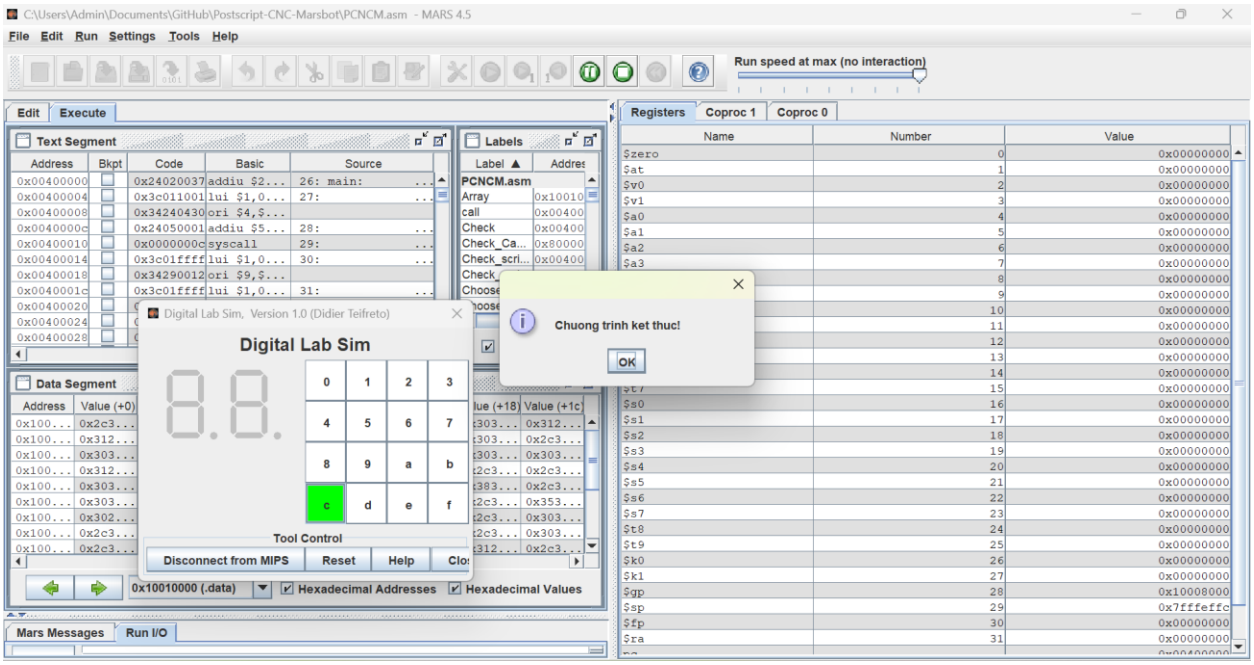
b. Postscript 4: DCE



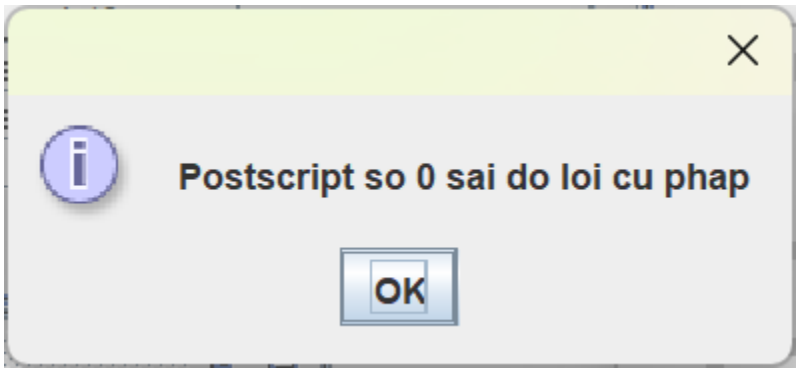
c. Postscript 8: HUST



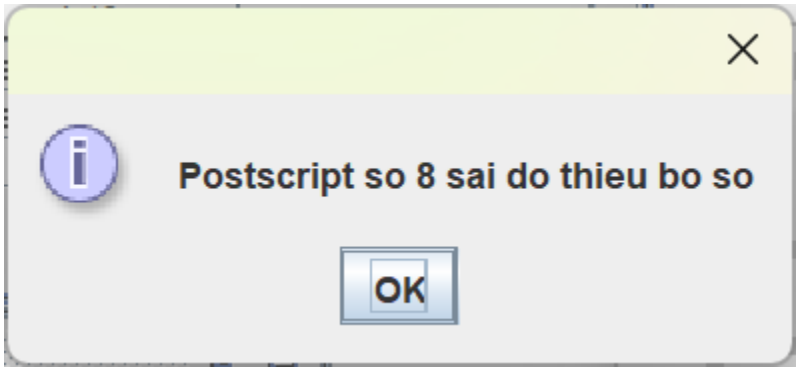
d. Khi ấn phím C



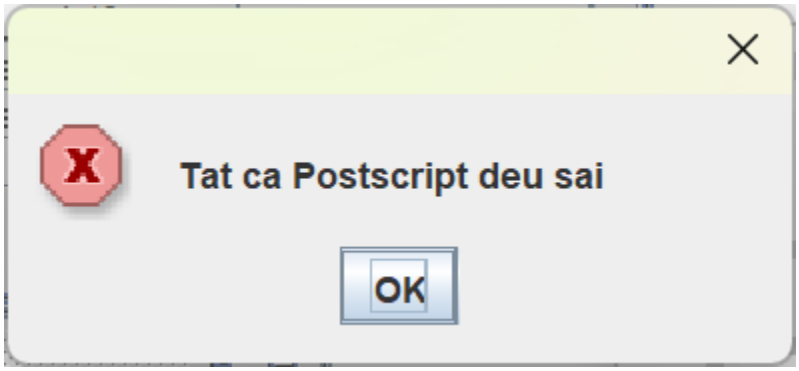
e. Lỗi cú pháp



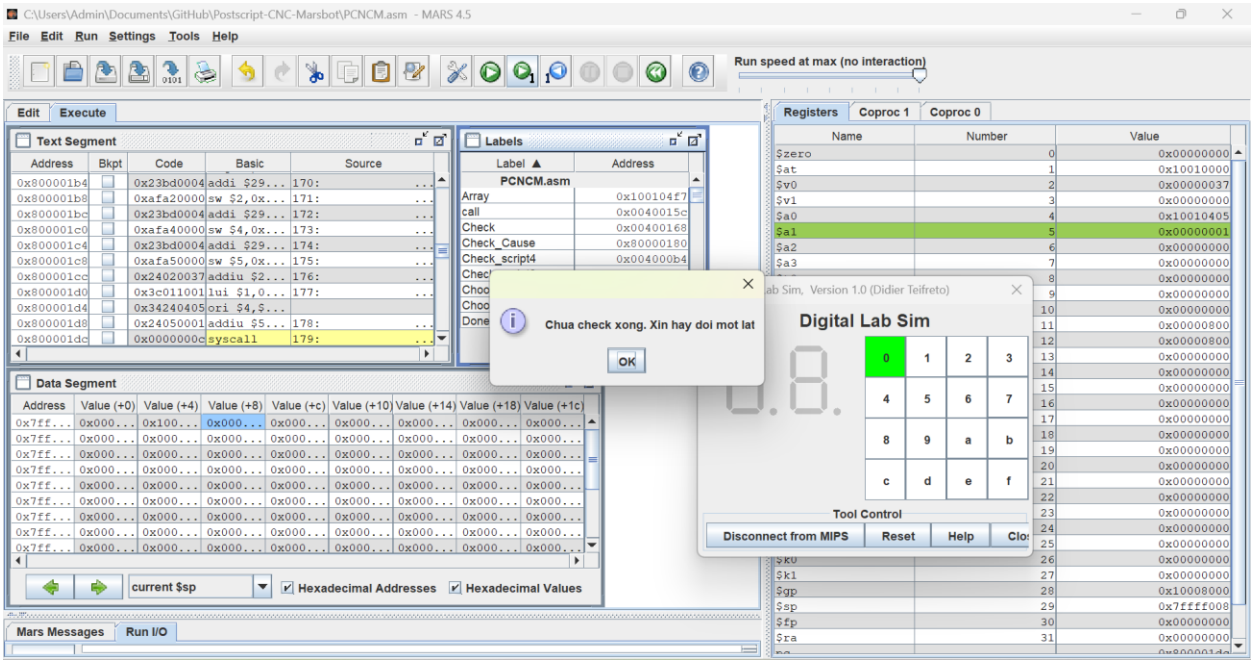
f. Lỗi thiếu bộ số



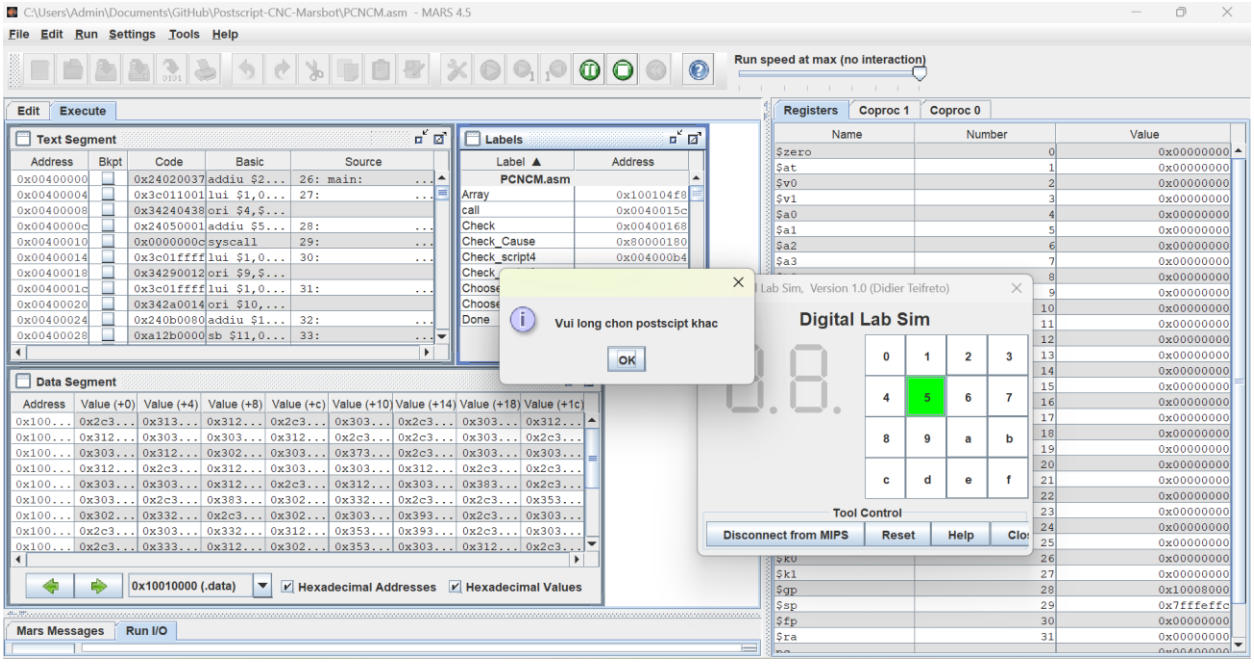
g. Lỗi tất cả các Postscript đều sai



h. Ấn phím khi chưa check xong



i. Chọn phím khác 0,4,8,c

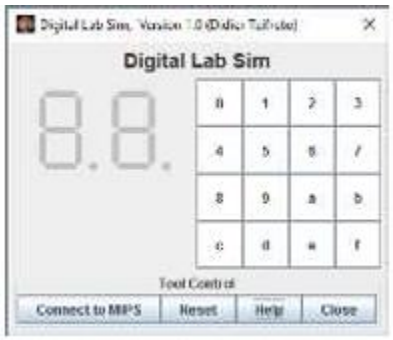


PHẦN 2: Nguyễn Thị Huyền Trang - 20225674

1. Đề bài

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.



2. 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 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
array: .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 " (giay)\nToc do go trung binh: "
message3: .asciiz " (tu/phut)\n"
message4: .asciiz "Chon Yes de tiep tuc kiem tra"

.text   # bien toan cuc : k0, k1, s0, s1, s2, s3, s4

main:
li      $k0, KEY_CODE
li      $k1, KEY_READY
la      $s0, string          # s0 = dia chi cua xau string
addi    $s1, $0, 0           # s1 = so ki tu dung
addi    $s2, $0, 0           # s2 = so tu
```

```
addi   $s3, $0, 0          # s3 = so lan xay ra counter interrupt
addi   $s4, $0, 0          # s4 = ki tu truoc
addi   $s5, $0, 0          # s5 = dem thoi gian


WaitForKey:
lw     $t1, 0($k1)         # $t1 = [$k1] = KEY_READY
nop
beq    $t1, $zero, WaitForKey # if $t1 == 0 then Polling
nop

# Enable the interrupt of TimeCounter of Digital Lab Sim
li     $t1, COUNTER
sb     $t1, 0($t1)

#------ vong lap doi keyboard interrupt ------------
loop:
lw     $t1, 0($k1)                 # $t1 = [$k1] = KEY_READY
bne    $t1, $zero, keyboard_interrupt   # Tao keyboard interrupt khi nhan duoc ky tu tu ban phim
addi   $v0, $0, 32                 # Neu khong nhap ky tu nao => sleep
li     $a0, 5                          # sleep 5 ms
syscall
b      loop                        # So lenh trong 1 vong lap = 6 => cu lap 5 lan thi tao 1 counter interrupt => dem 25 ms 1 counter interrupt
nop
```

```
#------- keyboard interrupt khi nhan duoc ky tu tu ban phim ----------

keyboard_interrupt:

teqi    $t1, 1                  # kiem tra neu t1 = 1 (co ky tu nhap vao tu ban
phim) thi ngat (thuc hien cau lenh o .ktext)

b       loop                    # Quay lai vong lap de cho doi su kien interrupt
tiep theo

nop


.ktext 0x80000180


#------ tam thoi vo hieu hoa ngat ------------

dis_int:

li      $t1, COUNTER                # BUG: must disable with Time Counter

sb      $zero, 0($t1)


#------ kiem tra loai interrupt o thanh ghi $13 -----------

get_cause:

mfc0  $t1, $13                  # $t1 = Coproc0.cause

is_counter:

li      $t2, MASK_CAUSE_COUNTER             # if Cause value confirm
Counter..

and     $at, $t1, $t2

bne     $at, $t2, keyboard_intr


#------ Counter interrupt -------------
```

```
counter_intr:
blt     $s3, 40, continue       # Neu so lan ngat do counter = 40 => du 1s -> khoi tao lai $s3, tang bien dem thoi gian len 1s
addi    $s3, $0, 0              # Khoi tao lai $s3 = 0
addi    $s5, $s5, 1             # Tang bien dem thoi gian len 1s
j       end_intr
nop
continue:
addi    $s3, $s3, 1             #Neu chua du 1s thi tang bien dem so lan ngat
j       end_intr
nop

#----- xu ly keyboard interrupt -----------
keyboard_intr:

process:                        # Kiem tra ky tu nhap vao
lb      $t0, 0($s0)             # t0 = string[i]
lb      $t1, 0($k0)             # t1 = ki tu nhap vao tu ban phim
beq     $t1, 10, end_program        # Ki tu la '\n' => in
bne     $t0, $t1, check_space       # Neu ki tu nhap vao # string[i] -> khong tang so ki tu dung
nop
addi    $s1, $s1, 1             # Tang so ky tu dung

check_space:                        # Kiem tra ki tu nhap vao co phai la ' ' khong (de xac dinh tu)
```

```
bne    $t1, ' ', end_process          # Neu ky tu nhap vao != ' ' => khong them tu
moi
nop
beq    $s4, ' ', end_process          # Neu co 2 dau cach lien tiep => khong
them tu moi
nop
addi   $s2, $s2, 1                    # Tang bien dem so tu da nhap
end_process:
beq    $t0, $0, update
addi   $s0, $s0, 1                    # Tang dia chi xau len 1
update:
addi   $s4, $t1, 0                    # Cap nhat lai ky tu truoc do
j      end_intr

# ---------------------- Ket thuc xu ly ngat ------------------------------------
end_intr:
# Enable the interrupt of TimeCounter of Digital Lab Sim
li     $t1, COUNTER
sb     $t1, 0($t1)
mtc0   $zero, $13                     # Must clear cause register
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
```

```
#------ Ket thuc chuong trinh khi nhan ky tu Enter ------------------------

end_program:
beq    $s4, ' ', print_digi_lab_sim        # neu ki tu cuoi cung khong la dau cach => them 1 tu
beq    $s4, $0, print_digi_lab_sim         # neu khong nhap ki tu nao => s2 = 0 => in ket qua
addi   $s2, $s2, 1

print_digi_lab_sim:                        # in so ky tu dung ra digital lab sim
li     $t1, 10
div    $s1, $t1
mfhi   $t1                    # t1 = chữ số bên phải
mflo   $t2                    # t2 = chữ số bên trái

la     $t0, array
add    $t1, $t0, $t1          # t1 = dia chi gia tri cho sevenseg_right
add    $t2, $t0, $t2          # t2 = dia chi gia tri cho sevenseg_left

lb     $t1, 0($t1)            # t1 = gia tri cho sevenseg_right
jal    show_7seg_right              # hien thi led ben phai
nop

lb     $t2, 0($t2)            # t2 = gia tri cho sevenseg_left
jal    show_7seg_left               # hien thi led ben trai
```

```
nop

j       print_rate
show_7seg_right:                    # in ket qua led ben phai
li      $t0, SEVENSEG_RIGHT             # assign right port's address
sb      $t1, 0($t0)             # assign new value for right led
nop
jr      $ra
nop
show_7seg_left:                     # in ket qua led ben trai
li      $t0, SEVENSEG_LEFT      # assign left port's address
sb      $t2, 0($t0)             # assign new value for left led
nop
jr      $ra
nop

print_rate:                         # in ra thoi gian va toc do go
li      $v0, 4
la      $a0, message1
syscall                             # in ra dong "Thoi gian hoan thanh: "

li      $v0, 1
addi    $a0, $s5, 0
syscall                             # in ra thoi gian
```

```
li      $v0, 4
la      $a0, message2
syscall                         # in ra dong " (giay)\nToc do go trung binh:
"

li      $v0, 1
li      $a0, 60
mult    $s2, $a0
mflo    $s2
div     $s2, $s5
mflo    $a0                     # toc do go 1 phut = so tu * 60 / thoi gian
syscall                         # in ra toc do go

li      $v0, 4
la      $a0, message3
syscall                         # in ra dong " (tu/phut)"

#----- Kiem tra xem co muon tiep tuc chuong trinh khong ------
check_back:
li      $v0, 50
la      $a0, message4
syscall

beq     $a0, 0, main
beq     $a0, 1, end
end:
```

3. Phân tích

3.1. Cách làm, thuật toán

Bước 1: Khai báo các biến cần thiết trong vùng data, khởi tạo các giá trị ban đầu trong main

```
.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 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
array: .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 " (giay)\nToc do go trung binh: "
message3: .asciiz " (tu/phut)\n"
message4: .asciiz "Chon Yes de tiep tuc kiem tra"

.text   # bien toan cuc : k0, k1, s0, s1, s2, s3, s4

main:
        li      $k0, KEY_CODE
        li      $k1, KEY_READY
        la      $s0, string             # s0 = dia chi cua xau string
        addi    $s1, $0, 0              # s1 = so ki tu dung
        addi    $s2, $0, 0              # s2 = so tu
        addi    $s3, $0, 0              # s3 = so lan xay ra counter interrupt
        addi    $s4, $0, 0              # s4 = ki tu truoc
        addi    $s5, $0, 0              # s5 = dem thoi gian

```

Bước 2: Tạo vòng lặp vô hạn WaitForKey đợi nhập ký tự đầu tiên (bắt đầu tính thời gian từ khi nhập ký tự đầu tiên đến khi nhấn Enter)

```

WaitForKey:
        lw      $t1, 0($k1)             # $t1 = [$k1] = KEY_READY
        nop
        beq     $t1, $zero, WaitForKey # if $t1 == 0 then Polling
        nop

```

Bước 3: Khi người dùng nhập ký tự đầu tiên, Enable ngắt do Counter trong Digital Lab Sim

```
35          # Enable the interrupt of TimeCounter of Digital Lab Sim
36          li      $t1, COUNTER
37          sb      $t1, 0($t1)
38
```

Bước 4: Vòng lặp chính của chương trình:

– Tạo 1 vòng lặp:

```
39  #------ vong lap doi keyboard interrupt ------------
40  loop:
41          lw      $t1, 0($k1)                 # $t1 = [$k1] = KEY_READY
42          bne     $t1, $zero, keyboard_interrupt # Tao keyboard interrupt khi nhan duoc ky tu tu ban phim
43          addi    $v0, $0, 32                 # Neu khong nhap ky tu nao => sleep
44          li      $a0, 5                      # sleep 5 ms
45          syscall
46          b       loop                        # So lenh trong 1 vong lap = 6 => cu lap 5 lan thi tao 1 counter interrupt => dem 25 ms 1 counter in
47          nop
48
49  #------- keyboard interrupt khi nhan duoc ky tu tu ban phim ----------
50  keyboard_interrupt:
51          teqi    $t1, 1                      # kiem tra neu t1 = 1 (co ky tu nhap vao tu ban phim) thi ngat (thuc hien cau lenh o .ktext)
52          b       loop                        # Quay lai vong lap de cho doi su kien interrupt tiep theo
53          nop
54
```

- Trong vòng lặp 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 có tín hiệu ngắt: 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

```
54
55  .ktext 0x80000180
56
57  #------ tam thoi vo hieu hoa ngat ------------
58  dis_int:
59          li      $t1, COUNTER                    # BUG: must disable with Time Counter
60          sb      $zero, 0($t1)
61
62  #------ kiem tra loai interrupt o thanh ghi $13 -----------
63  get_cause:
64          mfc0    $t1, $13                        # $t1 = Coproc0.cause
65  is_counter:
66          li      $t2, MASK_CAUSE_COUNTER         # if Cause value confirm Counter..
67          and     $at, $t1, $t2
68          bne     $at, $t2, keyboard_intr
```

- Nếu ngắt do bộ đếm Counter:

```
69
70  #------ Counter interrupt -------------
71  counter_intr:
72          blt     $s3, 40, continue               # Neu so lan ngat do counter = 40 => du 1s -> khoi tao lai $s3, tang bien dem thoi gian len 1s
73          addi    $s3, $0, 0                      # Khoi tao lai $s3 = 0
74          addi    $s5, $s5, 1                     # Tang bien dem thoi gian len 1s
75          j       end_intr
76          nop
77  continue:
78          addi    $s3, $s3, 1                     #Neu chua du 1s thi tang bien dem so lan ngat
79          j       end_intr
80          nop
```

  - Cứ mỗi 30 lệnh sau khi được enable, Counter Interrupt sẽ xảy ra. Mà vòng lặp đợi Keyboard có 6 lệnh, thiết lập thời gian 1 lần sleep là 5ms => sau 5 vòng lặp thời gian là 25ms thì ngắt do bộ đếm Counter xảy ra.
  - Khi ngắt do Counter xảy ra, kiểm tra nếu đủ 40 lần ngắt (tức là được 1s) thì tăng thời gian lên 1s và đặt biến đếm số lượng ngắt lại bằng 0, nếu chưa đủ 40 lần ngắt thì tăng biến đếm số lượng ngắt lên 1 và quay lại vòng lặp.
- Nếu ngắt do có dữ liệu nhập vào từ bàn phím:

```
81
82  #----- xu ly keyboard interrupt -----------
83  keyboard_intr:
84
85  process:                                        # Kiem tra ky tu nhap vao
86          lb      $t0, 0($s0)                     # t0 = string[i]
87          lb      $t1, 0($k0)                     # t1 = ki tu nhap vao tu ban phim
88          beq     $t1, 10, end_program            # Ki tu la '\n' => in
89          bne     $t0, $t1, check_space           # Neu ki tu nhap vao # string[i] -> khong tang so ki tu dung
90          nop
91          addi    $s1, $s1, 1                     # Tang so ky tu dung
92
93  check_space:                                    # Kiem tra ki tu nhap vao co phai la ' ' khong (de xac dinh tu)
94          bne     $t1, ' ', end_process           # Neu ky tu nhap vao != ' ' => khong them tu moi
95          nop
96          beq     $s4, ' ', end_process           # Neu co 2 dau cach lien tiep => khong them tu moi
97          nop
98          addi    $s2, $s2, 1                     # Tang bien dem so tu da nhap
99  end_process:
100         beq     $t0, $0, update
101         addi    $s0, $s0, 1                     # Tang dia chi xau len 1
102 update:
103         addi    $s4, $t1, 0                     # Cap nhat lai ky tu truoc do
104         j       end_intr
105
```

  - Ta kiểm tra xem ký tự mới nhập có phải là ký tự Enter hay không, 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 không phải ký tự Enter, 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 có từ mới không bằng cách so sánh ký tự mới nhập và ký tự nhập trước đó với khoảng trắng (space). Nếu 2 ký tự đó đều là khoảng trắng thì không có thêm từ mới. Nếu ký tự trước đó là khoảng trắng và ký tự hiện tại không phải khoảng trắng thì tăng số từ trong đoạn lên 1.
- Cập nhật lại các thông số cần thiết để đến vòng lặp tiếp theo

```
106 # ----------------------- Ket thuc xu ly ngat ---------------------------------------
107 end_intr:
108         # Enable the interrupt of TimeCounter of Digital Lab Sim
109         li      $t1, COUNTER
110         sb      $t1, 0($t1)
111         mtc0    $zero, $13                      # Must clear cause register
112 next_pc:
113         mfc0    $at, $14                        # $at <= Coproc0.$14 = Coproc0.epc
114         addi    $at, $at, 4                     # $at = $at + 4 (next instruction)
115         mtc0    $at, $14                        # Coproc0.$14 = Coproc0.epc <= $at
116 return:
117         eret                                    # Return from exception
118
```

Bước 5:

– Tính toán kết quả đã thu được ở trên và in ra Digital Lab Sim:

$t1, $t2 chứa chữ số cần in ra ở led bên phải và bên trái, sau đó $t1, $t2 được gán là địa chỉ cho giá trị tương ứng sẽ in ra led (bằng cách cộng số thứ tự với địa chỉ mảng số nguyên ứng với các giá trị in ra led 7 thanh từ 1 đến 9 đã khai báo ở đầu, chỉ cần cộng không cần x4 vào số thứ tự vì mảng được khai báo kiểu byte, mỗi phần tử trong mảng chỉ chiếm 1 byte). Tiếp tục gọi 2 chương trình con để in ra led bên phải và bên trái.

```
#------ Ket thuc chuong trinh khi nhan ky tu Enter ------------------------

end_program:
        beq     $s4, ' ', print_digi_lab_sim    # neu ki tu cuoi cung khong la dau cach => them 1 tu
        beq     $s4, $0, print_digi_lab_sim     # neu khong nhap ki tu nao => s2 = 0 => in ket qua
        addi    $s2, $s2, 1

print_digi_lab_sim:                             # in so ky tu dung ra digital lab sim
        li      $t1, 10
        div     $s1, $t1
        mfhi    $t1                             # t1 = chữ số bên phải
        mflo    $t2                             # t2 = chữ số bên trái

        la      $t0, array
        add     $t1, $t0, $t1                   # t1 = dia chi gia tri cho sevenseg_right
        add     $t2, $t0, $t2                   # t2 = dia chi gia tri cho sevenseg_left

        lb      $t1, 0($t1)                     # t1 = gia tri cho sevenseg_right
        jal     show_7seg_right                 # hien thi led ben phai
        nop

        lb      $t2, 0($t2)                     # t2 = gia tri cho sevenseg_left
        jal     show_7seg_left                  # hien thi led ben trai
        nop

        j       print_rate
```

- Hiển thị led bên phải

```
show_7seg_right:                        # in ket qua led ben phai
        li      $t0, SEVENSEG_RIGHT     # assign right port's address
        sb      $t1, 0($t0)             # assign new value for right led
        nop
        jr      $ra
        nop
```

- Hiển thị led bên trái

```
show_7seg_left:                         # in ket qua led ben trai
        li      $t0, SEVENSEG_LEFT      # assign left port's address
        sb      $t2, 0($t0)             # assign new value for left led
        nop
        jr      $ra
        nop
```

– Hiển thị thời gian hoàn thành và tốc độ gõ trung bình ra màn hình Run I/O:

Tốc độ gõ trung bình (wpm) = 60 * số từ gõ được / thời gian gõ

```
print_rate:                                  # in ra thoi gian va toc do go
        li      $v0, 4
        la      $a0, message1
        syscall                              # in ra dong "Thoi gian hoan thanh: "

        li      $v0, 1
        addi    $a0, $s5, 0
        syscall                              # in ra thoi gian

        li      $v0, 4
        la      $a0, message2
        syscall                              # in ra dong " (giay)\nToc do go trung binh: "

        li      $v0, 1
        li      $a0, 60
        mult    $s2, $a0
        mflo    $s2
        div     $s2, $s5
        mflo    $a0                          # toc do go 1 phut = so tu * 60 / thoi gian
        syscall                              # in ra toc do go

        li      $v0, 4
        la      $a0, message3
        syscall                              # in ra dong " (tu/phut)"

```

– Hiển thị lựa chọn tiếp tục hoặc kết thúc chương trình. Nếu tiếp tục thì quay trở lại main, thiết lập lại các thông số và tiến hành lần lặp tiếp theo.

```
#----- Kiem tra xem co muon tiep tuc chuong trinh khong ------
check_back:
        li      $v0, 50
        la      $a0, message4
        syscall

        beq     $a0, 0, main
        beq     $a0, 1, end
end:
```

3.2. Các thanh ghi sử dụng cố định

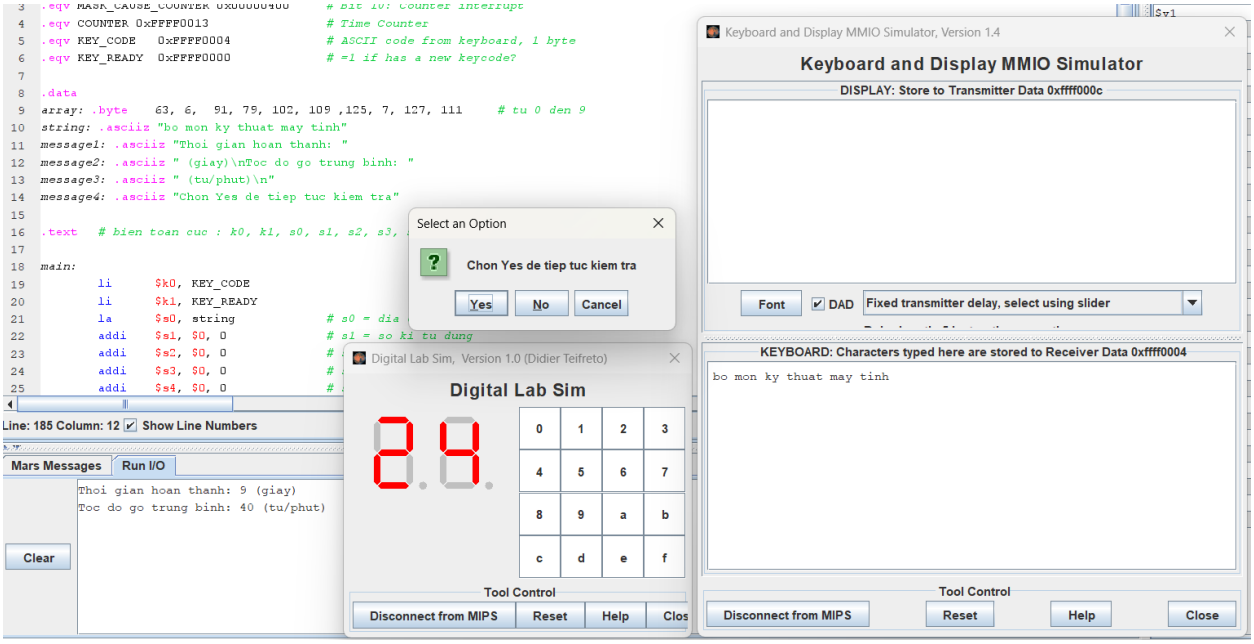
Thanh ghi	Vai trò
$k0	Lưu địa chỉ chứa trạng thái sẵn sàng của KEY_CODE
$k1	Lưu địa chỉ chứa ký tự được nhập vào Keyboard and Display MMIO Simulator
$s0	Lưu địa chỉ của xâu cố định
$s1	Chứa số ký tự nhập đúng

$s2	Chứa số từ nhập vào từ bàn phím
$s3	Đếm số lần xảy ra Counter Interrupt
$s5	Chứa thời gian hoàn thành đoạn ký tự

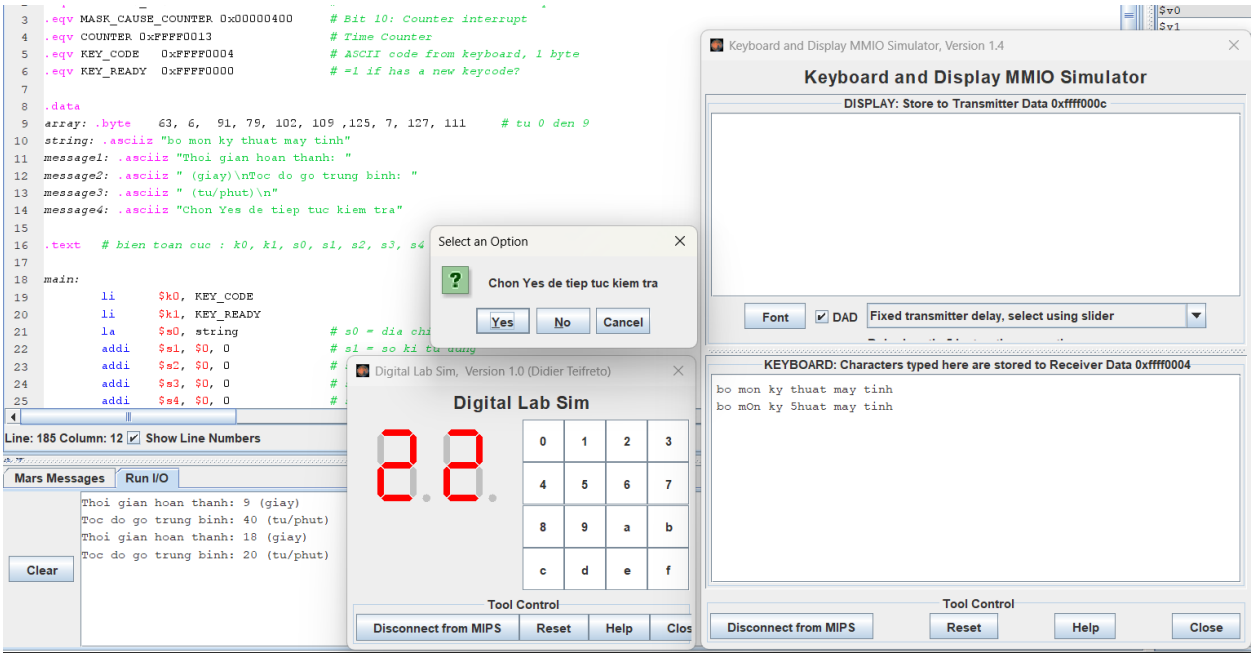
4. Kết quả

Đoạn mã cố định sẵn trong mã nguồn: “bo mon ky thuat may tinh”

- Đoạn ký tự gõ vào: “bo mon ky thuat may tinh” => số ký tự đúng là 24
  Với thời gian hoàn thành là 9s, đoạn văn bản có 6 từ => tốc độ gõ trung bình là 40 từ/phút

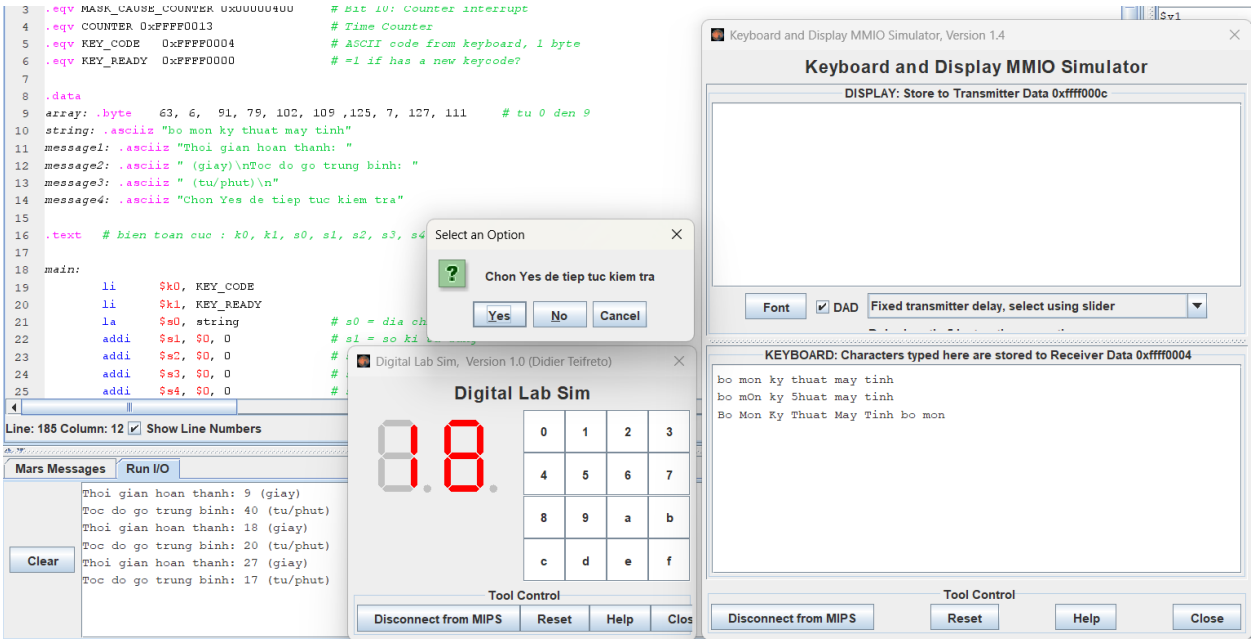


- Đoạn ký tự gõ vào; “bo mOn ky 5huat may tinh” => số ký tự đúng là 22
  Với thời gian hoàn thành là 18s, đoạn văn bản có 6 từ => tốc độ gõ trung bình là 20 từ/phút



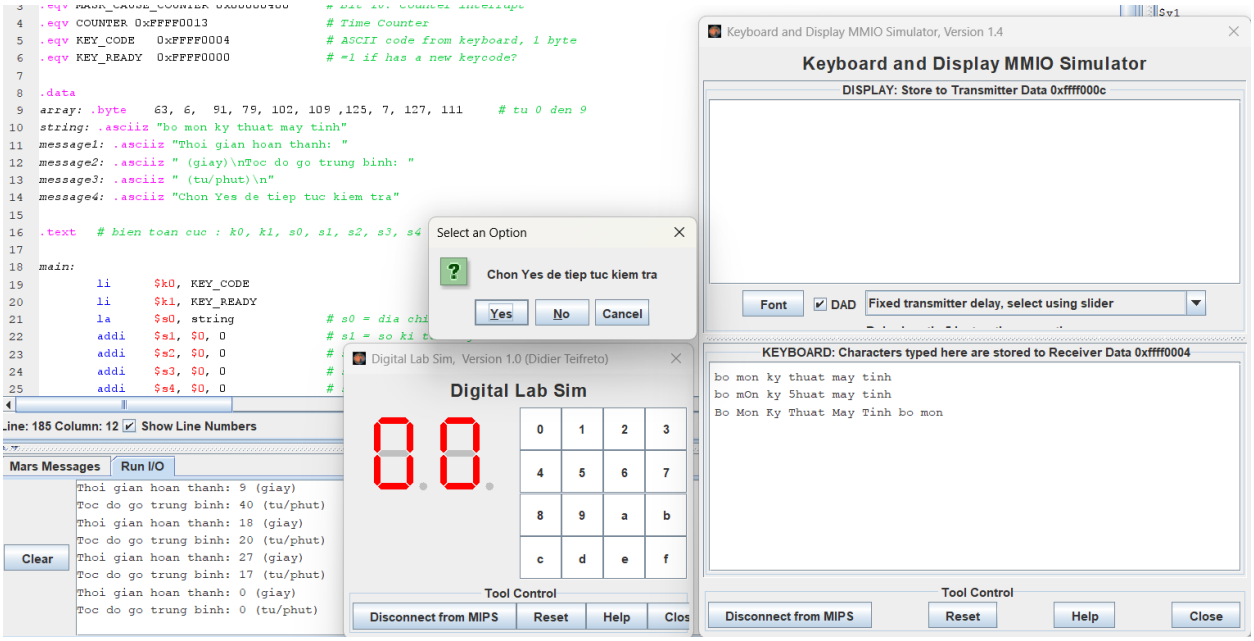
– Đoạn ký tự gõ vào “Bo Mon Ky Thuat May Tinh bo mon ” => số ký tự đúng là 18

Với thời gian hoàn thành là 27s, đoạn văn bản có 8 từ => tốc độ gõ trung bình là 17 từ/phút



– Đoạn ký tự gõ vào “” => số ký tự đúng là 0

Với thời gian hoàn thành là 0s, đoạn văn bản có 0 từ => tốc độ gõ trung bình là 0 từ/phút


.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
array: .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 " (giay)\nToc do go trung binh: "
message3: .asciiz " (tu/phut)\n"
message4: .asciiz "Chon Yes de tiep tuc kiem tra"
.text # bien toan cuc : k0, k1, s0, s1, s2, s3, s4
main:
li $k0, KEY_CODE
li $k1, KEY_READY
la $s0, string # s0 = dia chi
addi $s1, $0, 0 # s1 = so ki
addi $s2, $0, 0 #
addi $s3, $0, 0 #
addi $s4, $0, 0 #
Select an Option
Chon Yes de tiep tuc kiem tra
Yes
No
Cancel
Keyboard and Display MMIO Simulator, Version 1.4
Keyboard and Display MMIO Simulator
DISPLAY: Store to Transmitter Data 0xffff000c
Font
DAD
Fixed transmitter delay, select using slider
KEYBOARD: Characters typed here are stored to Receiver Data 0xffff0004
bo mon ky thuat may tinh
bo mOn ky 5huat may tinh
Bo Mon Ky Thuat May Tinh bo mon
Tool Control
Disconnect from MIPS
Reset
Help
Close
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
Clos
ine: 185 Column: 12
Show Line Numbers
Mars Messages
Run I/O
Clear
Thoi gian hoan thanh: 9 (giay)
Toc do go trung binh: 40 (tu/phut)
Thoi gian hoan thanh: 18 (giay)
Toc do go trung binh: 20 (tu/phut)
Thoi gian hoan thanh: 27 (giay)
Toc do go trung binh: 17 (tu/phut)
Thoi gian hoan thanh: 0 (giay)
Toc do go trung binh: 0 (tu/phut)