TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

Báo cáo bài tập lớn
Môn:Thực hành kiến trúc máy tính

Giáo viên hướng dẫn: Lê Bá Vui

Sinh viên thực hiện:

Phạm Trung Hiếu – 20204968 – Bài 3

Phạm Tuấn Anh – 20205054 – Bài 8

Bài 8: Mô phỏng ổ đĩa RAID 5 (Phạm Tuấn Anh)

8.1 Đề bài:

Hệ thống ổ đĩa RAID5 cần tối thiểu 3 ổ đĩa cứng, trong đó phần dữ liệu parity sẽ được chứa lần lượt lên 3 ổ đĩa như trong hình bên. Hãy viết chương trình mô phỏng hoạt động của RAID 5 với 3 ổ đĩa, với giả định rằng, mỗi block dữ liệu có 4 kí tự. Giao diện như trong minh họa dưới. Giới hạn chuỗi kí tự nhập vào có *độ dài là bội của 8.*

Trong ví dụ sau, chuỗi kí tự nhập vào từ bàn phím (DCE.****ABCD1234HUSTHUST) sẽ được chia thành các block 4 byte. Block 4 byte đầu tiên "DCE." sẽ được lưu trên Disk 1, Block 4 byte tiếp theo "****" sẽ lưu trên Disk 2, dữ liệu trên Disk 3 sẽ là 4 byte parity được tính từ 2 block đầu tiên với mã ASCII là

6e='D' xor '*' ; 69='C' xor '*'; 6f='E' xor '*' ; 04='.' xor '*'

```
Nhap chuoi ki tu : DCE.****ABCD1234HUSTHUST
     Disk 1                Disk 2                Disk 3
 --------------        --------------        --------------
|     DCE.     |      |     ****     |      [[ 6e,69,6f,04]]
|     ABCD     |      [[ 70,70,70,70]]      |     1234     |
[[ 00,00,00,00]]      |     HUST     |      |     HUST     |
 --------------        --------------        --------------
```

8.2 Phân tích cách làm:

- Bài gồm các phần chính : Menu (input) , kiểm tra input , in các disk trực tiếp , tính xor đổi ra asccii và in ra.
- Ý tưởng : dùng menu để nhập dữ liệu đầu vào , với những disk cần in trực tiếp từ dữ liệu nhập thì in ra , còn những disk cần tính xor thì tính rồi chuyển kết quả sang ascci ; vị trí disk in kết quả xor sẽ dựa vào thứ tự sắp xếp in thường và in kết quả xor.
- Cụ thể các phần như sau:
    a. Menu:
- Người dùng được chọn 1 là chạy chương trình , 2 là thoát
- Khi chạy chương trình người dùng sẽ nhập chuỗi kí tự và ta lưu địa chỉ dãy người dùng vừa nhập
- Nếu chọn thoát , chương trình sẽ kết thúc

b. Kiểm tra Input: chính là kiểm tra dãy có độ dài chia hết cho 4 không

- Chương trình sẽ kiểm tra :
  + Nếu k nhập gì -> độ dài = 0 => nhập lại
  + Chạy đến khi gặp kí tự kết thúc /n
  + Lấy ra byte cuối . Nếu byte cuối bằng 0 hoặc 8 thì sẽ chia hết cho 8 (vì khi kết hợp với những byte đằng trước đều chia hết cho 8 sẵn rồi)=> thỏa mãn . Nếu không thỏa mãn thì => nhập lại

c. In các kí tự và in ra giá trị tương ứng các disk:

- In các kí tự bằng syscall
- In các giá trị ứng với các disk bằng hàm print_word (hoạt động lấy địa chỉ rồi in ra)

d. In từng dòng (hết 3 dòng thì lặp lại)

- Với dòng thứ 1 thì print_word 2 lần rồi đến hàm show kết quả xor
- Với dòng thứ 2 thì print_word 1 lần , hàm show 1 lần , rồi print_word 1 lần nữa
- Với dòng thứ 3 thì hàm show trước , 2 hàm print_word sau

e. Hàm show_partition:

- Tính Xor từng cặp kí tự (4 cặp) rồi lấy kết quả . Kết quả sẽ thu được dưới dạng hexa , chương trình chạy hàm hex_to_string để chuyển thành kí tự trong bảng ascii

g. Hàm hex_to_string:

- TH1 : kết quả đó không lớn hơn 9 thì cộng với 0 (trong ascii) để thu được kết quả dưới dạng ascii
- TH2 : kết quả lớn hơn 9 thì cộng với a (trong ascii) rồi trừ 10 để thu được kết quả dưới dạng ascii

8.3 Mã nguồn:

x

.data

```
input:          .space 64
start:  .asciiz "Nhap chuoi ky tu : "
space:.asciiz          "     "
m0:    .asciiz "Chon chuc nang: 1. Chay  2. Thoat"
m1:    .asciiz "      Disk 1              Disk 2              Disk 3\n"
m2:    .asciiz " --------------       --------------       --------------\n"
m3:    .asciiz          "[[ "
m4:    .asciiz          "]]"
error_length: .asciiz "Do dai chuoi khong hop le! Hay nhap lai.\n"
.align 2
data:                   .space          4



.text
#Menu
        addi $t4, $0, 1
        addi $t5, $0, 2
Menu:
        li              $v0, 51
        la              $a0, m0
        syscall
        beq $t5, $a0, End
        bne $t4, $a0, Menu
```

```
Nhap:
        li      $v0, 4                                  # print start
        la      $a0, start
        syscall
        li      $v0, 8                                  # read input
        la      $a0, input                              # address of input buffer
        li      $a1, 64
        # max length
        syscall
        move $s0, $a0           # s0 chua dia chi xau moi nhap
#kiem tra do dai co chia het cho 8 khong
length: addi $s3, $zero, 0      # s3 = length
        addi $s6, $zero, 0  # s6 = index

check_char: add $s7, $s0, $s6   # s7 = address of string[i]
        lb $t6, 0($s7)          # t6 = string[i]
        nop
        beq $t6, 10, test_length    # t6 = '\n' ket thuc xau
        nop
        addi $s3, $s3, 1    # length++
        addi $s6, $s6, 1    # index++
        j check_char
```

```
        nop
test_length:
       beqz $s3, error1
        move $s5, $s3
        and $s7, $s3, 0x0000000f            # xoa het cac byte cua $s3 ve 0,
chi giu lai byte cuoi
        bne $s7, 0, test1                   # byte cuoi bang 0 hoac 8 thi so chia
het cho 8
        j Line1
test1:  beq $s7, 8, Line1
        j error1
error1:         li $v0, 4                   #Khong phai chuoi boi cua 8 thi
quay lai input
        la $a0, error_length
        syscall
        j Nhap
#ket thuc kiem tra do dai



Line1:
        li      $v0, 4                              # print m1
        la      $a0, m1
        syscall
```

```
        la      $a0, m2                         # print ----
        syscall
        la      $s0, input
loop:
        lw      $t0, 0($s0)
        lw      $t1, 4($s0)
        li      $t2, 10
        beq     $t0, $t2, end_loop              # Đến kí tự /n thì dừng
        move    $a1, $t0
        jal     print_word
        move    $a1, $t1
        jal     print_word
        jal     show_partition
        li      $v0, 11                         # new line
        li      $a0, 10
        syscall
        addi    $s0, $s0, 8
        lw      $t0, 0($s0)
        lw      $t1, 4($s0)
        li      $t2, 10
        beq     $t0, $t2, end_loop
        move    $a1, $t0
        jal     print_word
```

```
        jal     show_partition
        move $a1, $t1
        jal     print_word
        li      $v0, 11                         # new line
        li      $a0, 10
        syscall
        addi    $s0, $s0, 8
        lw      $t0, 0($s0)
        lw      $t1, 4($s0)
        li      $t2, 10
        beq     $t0, $t2, end_loop
        jal     show_partition
        move $a1, $t0
        jal     print_word
        move $a1, $t1
        jal     print_word
        li      $v0, 11                         # new line
        li      $a0, 10
        syscall
        addi    $s0, $s0, 8
        j       loop
end_loop:
        li      $v0, 4
```

```
        la      $a0, m2
        syscall
        j Menu


hex_to_string:
        and     $t8, $a0, 0xf           # Lay  vd b
        bgt     $t8, 0x9, condition1    # >9 hay k
        li      $v0, 0x30               # + 0 trong ascii
        add     $v0, $v0, $t8           # v0 = 0 + t8(<=9) = t8 trong ascii
        j       next
condition1:
        li      $v0, 0x61               # + "a" ascii
        subi    $t9, $t8, 0xa           # -10
        add     $v0, $v0, $t9                   # Ket qua xor theo ascii
next:
        sll     $v0, $v0, 8             # Lay cho ghi ki tu thu 2
        srl     $t8, $a0, 4             # Lay ki tu thu 2 de xet
        bgt     $t8, 0x9, condition2
        addi    $v0, $v0, 0x30
        add     $v0, $v0, $t8
        j       end_hts
condition2:
        addi    $v0, $v0, 0x61
```

```
        subi    $t9, $t8, 10
        add     $v0, $v0, $t9
end_hts:
        jr      $ra

print_word:
        li      $v0, 11                 # print |
        li      $a0, 124
        syscall
        li      $v0, 4              # print space
        la      $a0, space
        syscall
        la      $a0, data
        sw      $a1, 0($a0)         # print 4 ki tu
        syscall
        la      $a0, space
        syscall
        li      $v0, 11
        li      $a0, 124
        syscall
        li      $v0, 4
        la      $a0, space
        syscall
```

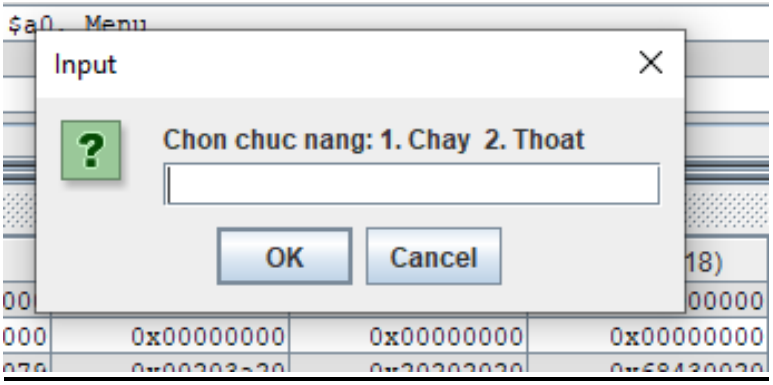
```
        jr      $ra


show_partition:
        li      $v0, 4
        sw      $ra, 0($sp)
        la      $a0, m3             #print [[
        syscall
        xor     $s1, $t0, $t1
        and     $a0, $s1, 0xff      # Ket qua xor tung bit   vs 0xab
        jal     hex_to_string       # Chuyen ket qua xor sang ascii de in ra
man hinh
        la      $a0, data           # in ra 2 chu
        sw      $v0, 0($a0)
        li      $v0, 4
        syscall
        li      $a0, ','
        li      $v0, 11
        syscall
        srl     $s1, $s1, 8
        and     $a0, $s1, 0xff
        jal     hex_to_string
        la      $a0, data
        sw      $v0, 0($a0)
```

```
li      $v0, 4
syscall
li      $a0, ','
li      $v0, 11
syscall
srl     $s1, $s1, 8
and     $a0, $s1, 0xff
jal     hex_to_string
la      $a0, data
sw      $v0, 0($a0)
li      $v0, 4
syscall
li      $a0, ','
li      $v0, 11
syscall
srl     $s1, $s1, 8
and     $a0, $s1, 0xff
jal     hex_to_string
la      $a0, data
sw      $v0, 0($a0)
li      $v0, 4
syscall
la      $a0, m4
```

```
        syscall
        la      $a0, space
        syscall
        lw      $ra, 0($sp)
        jr      $ra
End:
```

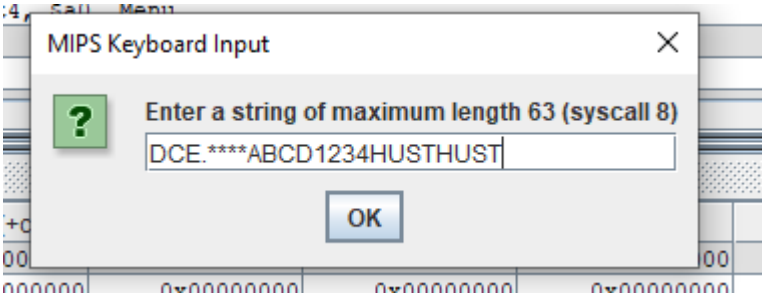
8.4. Kết quả chạy chương trình :

Khi khởi chạy chương trình , menu hiện ra cho người dùng chọn chức năng



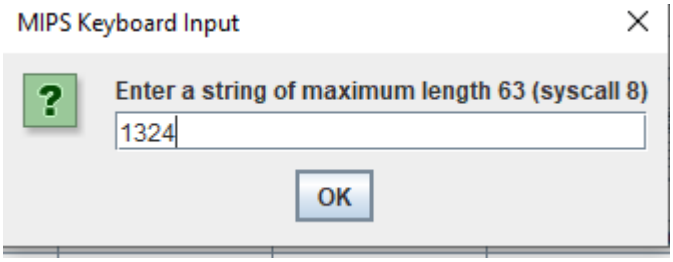
Khi người dùng chọn chức năng một , chương trình sẽ hoạt động và yêu cầu người dùng nhập dãy kí tự

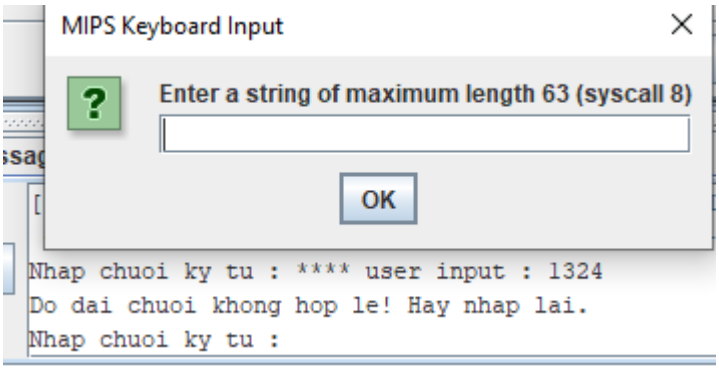
Khi người dùng nhập dãy có độ dài (là bội của 8) thỏa mãn thì sẽ in ra màn hình kết quả và hiện lại menu



```
Nhap chuoi ky tu : **** user input : DCE.****ABCD1234HUSTHUST
      Disk 1                Disk 2               Disk 3
 --------------       --------------       --------------
|      DCE.      |     |      ****      |     [[ 6e,69,6f,04]]
|      ABCD      |     [[ 70,70,70,70]]      |      1234      |
[[ 00,00,00,00]]     |      HUST      |     |      HUST      |
 --------------       --------------       --------------
```

Khi người dùng nhập dãy có độ dài không thỏa mãn , chương trình sẽ thông báo cho người dùng và cho người dùng nhập lại





Cuối cùng khi chọn chức năng 2 là thoát chương trình sẽ kết thúc hoàn toàn

Bài 3:Kiểm tra tốc độ và độ chính xác khi gõ văn bản(Phạm Trung Hiếu)

3.1.Đầu 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 DigitalLab Sim) để tạo ra khoảng thời gian để đo. Đây là thời gian giữa 2 lần ngắt, chu kì ngắt

Yêu cầu:

1.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 Ovà 5)mà người dùng đã gõ và hiển thị lên các đèn led.

2.Chương trình đồng thời cần tính được tốc độ gõ: thời gian hoàn thành và số từ trên một đơn vị thời gian.

3.2.Cách làm,thuật toán

_Chọn chu kì ngắt là: 1s

_Đọc ký tự người dùng rồi lưu vào trong bộ nhớ.Đọc được ký tự enter thì kết thúc nhập.Sau đó so sánh xâu vừa nhập với xâu nguồn rồi in ra số ký tự đúng.

_Sau mỗi 1s sẽ hiển thị số ký tự mà người dùng đã nhập.Kể từ khi người dùng nhập ký tự đầu tiên thì sau mỗi 1s tăng biến đếm liên tục đến khi kết thúc thì in ra màn hình.

_Kết thúc chương trình sẽ hiện ra hộp thoại hỏi người dùng có tiếp tục chương trình không

3.3.Mã nguồn

```
.eqv SEVENSEG_LEFT    0xFFFF0011 # Dia chi cua den led 7 doan trai
                                #Bit 0 = doan a
                                #Bit 1 = doan b
                                #Bit 7 = dau .
.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
Led_hex: .byte 63,6,91,79,102,109,125,7,127,111
```

```
String_space : .space 1000                    #khoang trong de luu cac
ky tu nhap tu ban phim.

stringsource : .asciiz "bo mon ky thuat may tinh"

Mess1: .asciiz "\n So ky tu trong 1s :  "

Mess2: .asciiz  "\n So ky tu nhap dung la: "

Mess3: .asciiz "\n Ban co muon quay lai chuong trinh? "

Mess4: .asciiz "\n Thoi gian hoan thanh la:  "

Mess5: .asciiz "\n Toc do go trung binh (ky tu/1s) la: "

Mess6: .asciiz "\n Vui long nhap xau!"

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

#

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

.text

        li   $k0,  KEY_CODE

        li   $k1,  KEY_READY

        li   $s0, DISPLAY_CODE

        li   $s1, DISPLAY_READY

Main:

        li $s2,0                    #Dung de dem toan bo so ky tu nhap
vao

        li $s3,0                     #Dung de dem so vong lap
```

```
        li $s4,10
        li $s5,200              #Luu gia tri so vong lap.
        li $s6,0                #Bien dem so ky tu nhap duoc
trong 1s
        li $s7,0                #Bien danh dau het chuong trinh
        li $a1,0                #Bien dem thoi gian go
Loop:
WAIT_FOR_KEY:
        lw   $t1, 0($k1)            # $t1 = [$k1] = KEY_READY
        beq  $t1, $zero,Check           # if $t1 == 0 then Polling
MAKE_INTER:
        addi $s6,$s6,1              #Tang bien dem ky tu nhap duoc
trong 1s len 1
        teqi $t1, 1                 # if $t0 = 1 then raise an Interrupt
#-------------------------------------------------------
#
#-------------------------------------------------------
Check:
        #Neu da lap duoc 200 vong( 1s) se nhay den xu ly so ky tu
nhap trong 1s.
        addi    $s3, $s3, 1       # Tang so vong lap len 1
        div $s3,$s5                 #Lay so vong lap chia cho 200 de
xac dinh da duoc 1s hay chua
```

```
        mfhi $t2                    #Luu phan du cua phep chia tren
        bnez $t2,Sleep              #Neu chua duoc 1s nhay den
Sleep
        #Neu da duoc 1s thi nhay den nhan SetCount de thuc hien in
ra man hinh
        beqz $s2,SetCount
        addi $a1,$a1,1              #Tang thoi gian go len 1s
SetCount:
        li $s3,0                    #Tao lai so vong lap cho 1s tiep
        li $v0,4                    #In Mess1
        la $a0,Mess1
        syscall
        li    $v0,1                 #In ra so ky tu trong 1s
        add   $a0,$zero,$s6
        syscall
DISPLAY_DIGITAL:
        div $s6,$s4                 #Lay so ky tu nhap duoc trong 1s
chia cho 10
        mflo $t2                    #Lay phan nguyen de hien thi o
Led trai
        la $t3,Led_hex              #Lay dia chi Led_hex
        add $t3,$t3,$t2             # Chi toi dia chi gia tri can hien
thi
```

```
        lb $a0,0($t3)                   #Lay gia tri cho vao $a0
        jal   SHOW_7SEG_LEFT            #Hien thi Led trai
#---------------------------------------------------------------------
        mfhi $t2                        #Lay phan du de hien thi o Led
phai
        la $t3,Led_hex                  #Lay dia chi Led_hex
        add $t3,$t3,$t2                 # Chi toi dia chi gia tri can hien
thi
        lb $a0,0($t3)                   #Lay gia tri cho vao $a0
        jal   SHOW_7SEG_RIGHT           #Hien thi Led phai
#---------------------------------------------------------------------
        li $s6,0                        #Khoi tao lai so ky tu trong 1s
cho 1s tiep
        beq $s7,1,Ask_Loop
Sleep:
        li $v0,32
        li $a0,5                        #Sleep 5ms
        syscall
        nop
        j Loop                          #Quay lai Loop
End_Main:
        li $v0,10
```

```
        syscall
        nop
SHOW_7SEG_LEFT:
        li   $t0,  SEVENSEG_LEFT        # assign port's address
        sb   $a0,  0($t0)          # assign new value
        jr   $ra
SHOW_7SEG_RIGHT:
        li   $t0,  SEVENSEG_RIGHT      # assign port's address
        sb   $a0,  0($t0)          # assign new value
        jr   $ra
#---------------------------------------------------------
#XU LY NGAT
#---------------------------------------------------------
.ktext0x80000180
        mfc0 $t1,$13                #Nguyen nhan ngat
        li   $t2, MASK_CAUSE_KEYBOARD        #Ngyen nhan
ngat do ban phim
        and  $at,$t1,$t2                #So sanh nguyen nhan ngat
        beq  $at,$t2,CountKey                #Neu do ban phim thi nhay
den CountKey
        j    End_Process                #Khong thi nhay den
End_Process
```

```
CountKey:
READ_KEY:
        lb   $t0, 0($k0)          # $t0 = [$k0] = KEY_CODE
WAIT_FOR_DIS:
        lw   $t2, 0($s1)                # $t2 = [$s1] =
DISPLAY_READY
        beq  $t2, $zero, WAIT_FOR_DIS       # if $t2 == 0 then
Polling
SHOW_KEY:
        sb $t0,0($s0)                   #Luu ki tu vua nhap vao vung
hien thi
        la $t5,String_space             #Dia chi luu xau duoc nhap
        add $t5,$t5,$s2                 #Nhay den dia can ghi cua chi ki
vua nhap
        sb $t0,0($t5)                   #Luu ki tu vua nhap
        addi $s2,$s2,1                  #Tang so ki tu len 1
        bne $t0,10,End_Process              #Neu khong phai ky tu la
enter nhay den End_Process
        beq $s2,1,Error                 #Neu la xau rong nhay den Error
        bnez $a1,End                    #Neu thoi gian hoan thanh >=1
thi nhay den End
        addi $a1,$a1,1                  #Neu khong thi lam tron roi nhay
        j End
End_Process:
```

```
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 lenh tiep theo trong Main
End:
        li $v0,11
        li $a0,'\n'             #In xuong dong
        syscall
        li $t1,0                        #Dem so ki tu da xet
        li $t2,0                        #Dem so ky tu dung

        li $t3,24               #Do dai xau cua ma nguon
        slt $t4,$s2,$t3                 #So sanh do dai 2 xau
        #Xau nao nho hon thi duyet theo xau do
        beqz $t4,Check_String
        add $t3,$zero,$s2               #Gan $t3=so ky tu da nhap de xet
        add $t3,$t3,-1                  #Bo qua ky tu enter
Check_String:
```

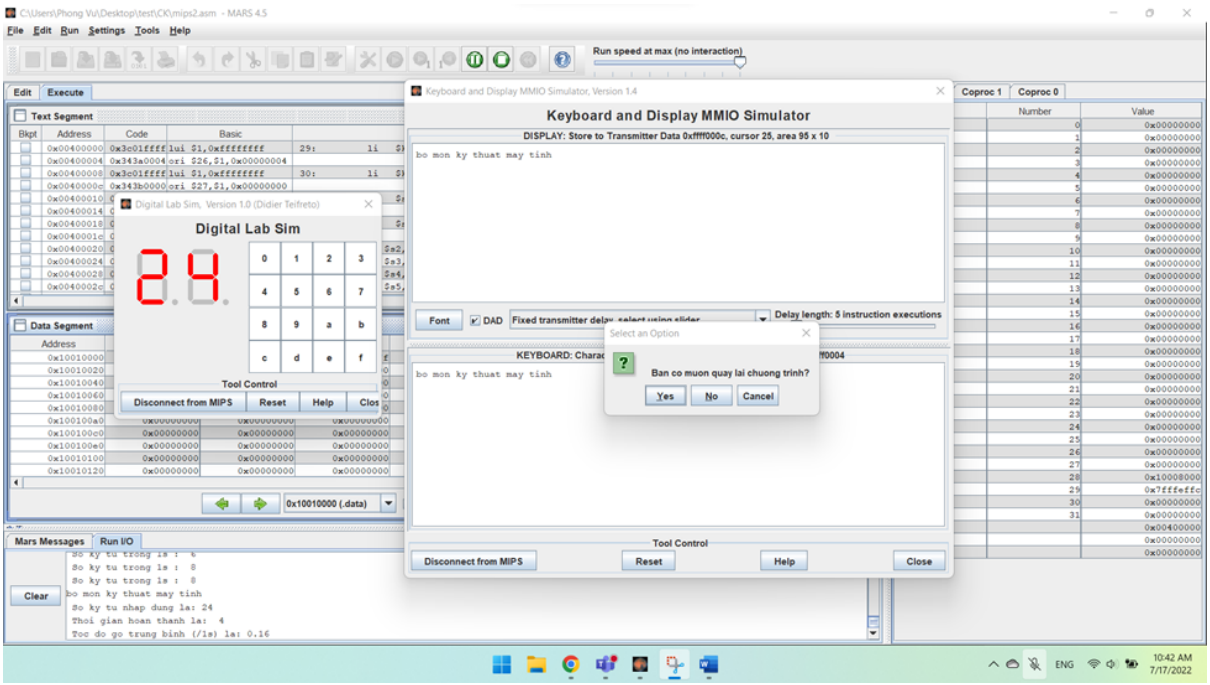
```
        la $t7 String_space         #Lay dia chi xau da nhap
        add $t7,$t7,$t1             #Nhay den dia chi ky tu dang xet
        li $v0,11                   #In ra ky tu dang xet
        lb $t4,0($t7)               #Lay ky tu dang xet de in
        add $a0,$zero,$t4
        syscall
        la $t5,stringsource         #Lay dia chi xau nguon
        add $t5,$t5,$t1             #Nhay den dia chi ky tu dang xet
        lb $t6,0($t5)               #Lay ky tu can xet de so sanh
        bne $t4,$t6,Continue        #Neu khac thi nhay den Continue
        add $t2,$t2,1               #Neu giong thi tang bien dem len
1
Continue:
        addi $t1,$t1,1              #Tang bien dem de xet ky tu tiep
theo
        beq $t1,$t3,Print           #Neu da duyet het ky tu thi nhay den
Print
        j Check_String              #Neu chua thi quay lai check tiep
Print:
        li $v0,4                    #In Mess2
        la $a0,Mess2
        syscall
```

```
        li $v0,1                    #In so ky tu dung
        add $a0,$zero,$t2
        syscall
        li $v0,4                    #In Mess4
        la $a0,Mess4
        syscall
        li $v0,1                    #In thoi gian hoan thanh
        add $a0,$zero,$a1
        syscall
        jal Tinh_TB                 #Tinh toc do go trung binh
        li $s7,1                    #Danh dau ket thuc chuong trinh
        add $s6,$zero,$t2           #Cho $s6=$t2 de hien thi tren led
        b DISPLAY_DIGITAL
Ask_Loop:
        li $v0,50               #In Mess3
        la $a0,Mess3
        syscall
        beqz $a0,Main
        b End_Main
Error:
        li $v0,4                    #In Mess6
```

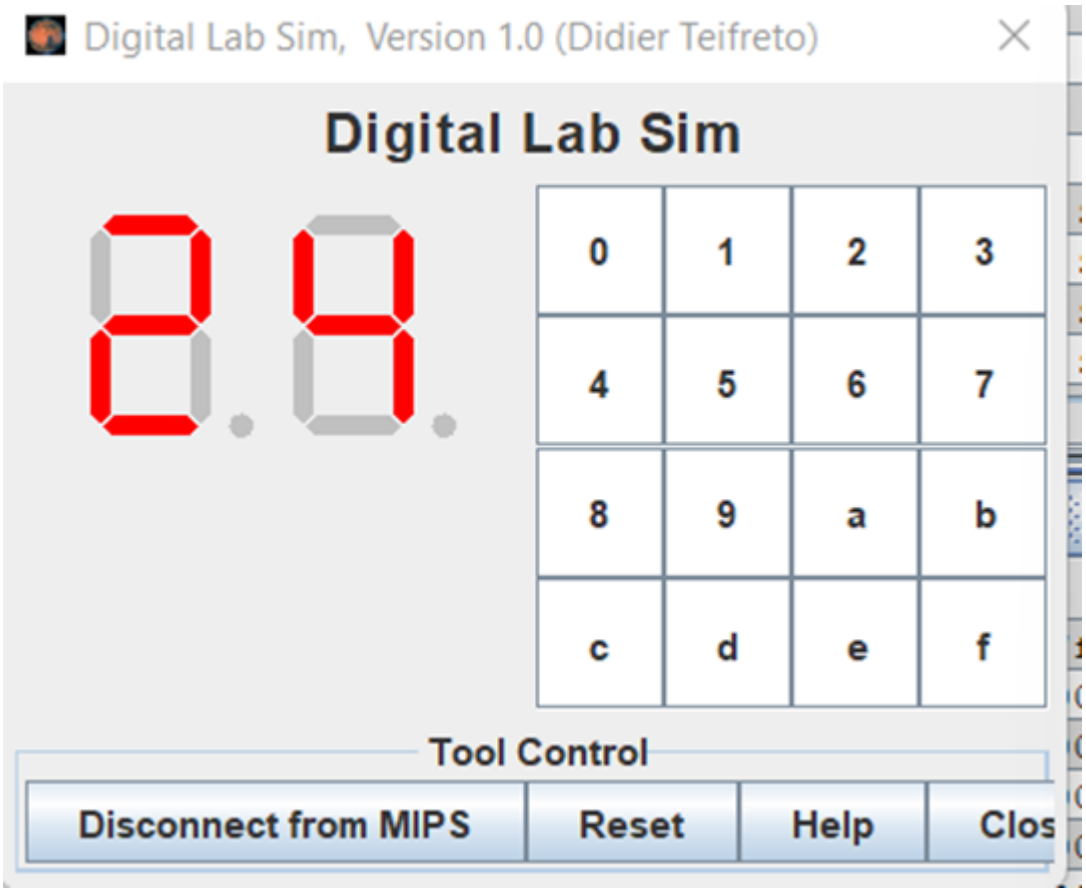
```
        la $a0,Mess6
        syscall
        b Main
Tinh_TB:
        li $v0,4                    #In Mess5
        la $a0,Mess5
        syscall
        div $s2,$a1                 #Lay so ky tu chia cho thoi gian
go
        li $t8,0                    #Bien dem so chu so sau dau
phay
        li $a2,0                    #Phan nguyen cua phep chia
Nguyen:
        mfhi $a3                    #Lay phan du de kiem tra
        beqz $a3,Show               #Neu phan du=0 thi la phep chia
het => in luon
        mflo $a0                    #Neu khac 0 thi xet phan nguyen
Show:
        mflo $a2
        li $v0,1                    #In phan nguyen
        add $a0,$zero,$a2
        syscall
```

```
        li $v0,11
        li $a0,'.'              #In dau '.'
        syscall
Thap_Phan:
        beq $t8,2,End1          #Lay 2 so sau dau phay
        mulo $a3,$a3,$s4            #Nhan phan du truoc voi 10
        div $a3,$a1             #Chia tiep de lay phan thap phan
        mflo $a3                #Lay phan nguyen de in
        li $v0,1                #In ra phan nguyen
        add $a0,$zero,$a3
        syscall
        mfhi $a3                #Lay phan du de kiem tra va cho
vong lap sau
        beqz $a3,End1           #Neu phan du =0 => Ket thuc
        addi $t8,$t8,1          #Neu khong thi tang bien dem
len 1 va lap tiep
        j Thap_Phan
End1:       jr $ra
```

3.4.Kết quả


Digital Lab Sim
Keyboard and Display MMIO Simulator
Tool Control
Disconnect from MIPS
Reset
Help
Close

_Số ký tự đúng:


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

_Tốc độ gõ trung bình và thời gian hoàn thành và số ký tự mỗi giây:

```
 So ky tu trong 1s :  0
 So ky tu trong 1s :  2
 So ky tu trong 1s :  6
 So ky tu trong 1s :  8
 So ky tu trong 1s :  8
oo mon ky thuat may tinh
 So ky tu nhap dung la: 24
 Thoi gian hoan thanh la:  4
 Toc do go trung binh (/1s) la: 0.16
-- program is finished running --
```

_Hộp thoai: