TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

Viện công nghệ Thông tin & Truyền thông

IT3280

Thực hành Kiến trúc máy tính

BÁO CÁO FINAL - PROJECT

Giảng viên hướng dẫn : ThS. Lê Bá Vui

Nhóm sinh viên :

1. Lê Việt Hùng - 20194580
2. Mai Đào Tuấn Thành - 20194675

Mục lục

1. Tổng quan

1.1. Đề tài được phân công

a) Project 3 – Kiểm tra tốc độ và độ chính xác khi gõ văn bản

Chương trình sau sẽ 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 Digi Lab Sim) để tạo ra khoảng thời gian để đo. Đây là thời gian giữa 2 lần ngắt, chu kì ngắt.

- Trong thời khoảng đó, người dùng nhập các kí tự từ bàn phím. Ví dụ nhập "*bo m**O**n ky **5**huat may tinh*".
Chương trình cần phải đếm số kí tự đúng (trong ví dụ trên thì người dùng gõ sai chữ **O** và **5**) mà người dùng đã gõ và hiển thị lên các đèn led.

b) Project 8 – Mô phỏng ổ đĩa RAID5

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      |
 --------------        --------------        --------------
```

1.2. Công cụ sử dụng

Mars 4.5

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

2.1. Phân tích

- Phân tích đề bài:
- Cho đoạn văn bản mẫu (ví dụ: “bo mon ky thuat may tinh”)
- Sử dụng bộ định thời Timer để tạo khoảng thời gian đo (chu kỳ ngắt)
- Phát hiện các ký tự sai so với văn bản gốc
- Đo tốc độ gõ của người dùng
- Hiển thị lên đèn led

2.2. Ý tưởng thuật toán

- Sử dụng vòng lặp để lưu các ký tự người dùng gõ vào 1 mảng cố định.
- Sử dụng syscall Sleep để đo thời gian từ lúc bắt đầu đến khi kết thúc chương trình
- Syscall được sử dụng trong Interupt của bộ đinh thời Timer của Digital Lab Sim
- So sánh chuỗi người dùng nhập vào với chuỗi mẫu để tìm ra số ký tự đúng
- Tốc độ gõ = Số phím gõ / Thời gian chương trình chạy
- Hiển thị kết quả lên trên đèn LED

2.3. Source code

```
.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 COUNTER_TIMER  0xFFFF0013  #Counter Timer's Address
.eqv MASK_CAUSE_COUNTER 0x00000400  # Bit 10: Counter interrupt

.data
bytedec     : .byte 63,6,91,79,102,109,125,7,127,111
inputString : .space 1000       #khoang trong de luu cac ky tu nhap tu ban phim.
SRCString : .asciiz "bo mon ky thuat may tinh"
msg_counter : .asciiz "\n So ky tu da nhap la :  "
numCorrectKey: .asciiz  "\n So ky tu nhap dung la: "
msg_request_loop: .asciiz "\n ban co muon quay lai chuong trinh? "
msg_speed: .asciiz "\nToc do danh may la: "
```

```
msg_unit: .asciiz " cps (character/second)"
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# MAIN Procedure
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
.text
    li   $k0,  KEY_CODE
    li   $k1,  KEY_READY
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# $s4: Sum of character from input's user
# $t4: Number 10
# $t9 - Boolean: For request loop program or not from user
# $s5 - Boolean: true if Enter key is press
# $s6: Counter Time from start to the end of program
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
main:
    li $s4,0
    li $t4,10
    li $t9,0
    li $s5, 0
    li $s6, 0
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Enable Counter (Digital Lab Sim)
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
START_COUNTER:
    li $s2, COUNTER_TIMER
    sb $s2, 0($s2)
LOOP:
WAIT_FOR_KEY:
    lw   $t1, 0($k1)                # $t1 = [$k1] = KEY_READY
    beq  $t1, $zero, WAIT_FOR_KEY        # if $t1 == 0 then Polling

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# 1. Store key in array 'inputString'
# 2. Add 1 for Sum of character from user input
# 3.  Check if Current character is Enter key
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
READ_KEY:
    lw $t0, 0($k0)
    la  $t7,inputString
    add $t7,$t7,$s4
    sb $t0,0($t7)
    addi $s4,$s4,1
    beq $t0,10,is_Entered
    nop
    j LOOP
```

```
is_Entered:
    li $s5, 1   #If $t0 is Enter key set $s5 = 1
    b END   #branch to END



END_MAIN:
    li $v0,10
    syscall

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# INTERUPT COUNTER
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
.ktext    0x80000180
#---------------------------------------------------------
# Temporary disable interrupt
#---------------------------------------------------------
dis_int:
    li $s2, COUNTER_TIMER
    sb $zero, 0($s2)
#---------------------------------------------------------
# Processing
#---------------------------------------------------------
get_caus:
    mfc0  $t8, $13
    li    $t2, MASK_CAUSE_COUNTER             # if Cause value confirm Counter..
    and   $at, $t8,$t2
    beq   $at,$t2, Counter_Intr
    j end_process

Counter_Intr:               # Processing Counter Interrupt
    li $v0, 4
    la $a0, msg_counter     #Just print msg about sum of input character
    syscall
    nop
    li $v0, 1 # service 1 is print integer
    add  $a0, $s4, $zero
    syscall # execute
    nop


end_process:
```

```
    mtc0 $zero, $13
check_WaitForKey:   #If on Loop Wait for key please dont next_pc
    beq  $t1, $zero, en_int
check_Entered:      #If on Loop main Please font next_pc
    beq $s5, 0, en_int

#-------------------------------------------------------
# Evaluate the return address of main routine
# epc <= epc + 4
#-------------------------------------------------------
next_pc:
    mfc0    $at, $14            # $at <= Coproc0.$14 = Coproc0.epc
    addi    $at, $at, 4  # $at = $at + 4 (next instruction)
    mtc0    $at, $14            # Coproc0.$14 = Coproc0.epc <= $at

#-------------------------------------------------------
# Re-enable interrupt
#-------------------------------------------------------
en_int:
    li $s2, COUNTER_TIMER
    sb $s2, 0($s2)

SLEEP:      addi $s6, $s6, 5
    addi    $v0,$zero,32
    li      $a0, 5      #Sleep 5ms
    syscall
    nop
#   b      LOOP     #Branch to LOOP

RETURN:
    eret                        # Return from exception

END:
Calc_Speed:
    mtc1  $s6, $f1
    cvt.s.w $f1, $f1

    mtc1 $t4, $f3
    cvt.s.w $f3, $f3

    mtc1 $s4, $f2
    cvt.s.w $f2, $f2

    div.s $f1, $f1, $f3
    div.s $f1, $f1, $f3
```

```
    div.s $f1, $f1, $f3

    div.s $f2, $f2, $f1

    li $v0,11
    li $a0,'\n'
    syscall
    li $t1,0        #Count the sum of character which is compared
    li $t3,0                        #Count the sum of correct character
    li $t8,24       #The lenght of Source String
    slt $t7,$s4,$t8     #Compare length of Source String with User's String
    bne $t7,1, CHECK_STRING #Which is lesser $t8 := that
    add $t8,$zero, $s4
    addi $t8,$t8,-1     #Skip Enter character from the end of string.
    beq $t8, 0, Print
CHECK_STRING:
    la $t2,inputString
    add $t2,$t2,$t1
    li $v0,11       #Get character of index $t1  of User's String to store in $t5
    lb $t5,0($t2)
    move $a0,$t5
    syscall         #Print User's String

    la $t4,SRCString
    add $t4,$t4,$t1
    lb $t6,0($t4)       #Get character of index $t1  of Source String to store in $t6
    bne $t6, $t5,CONTINUE   #Compare $t5, with $t6
    addi $t3,$t3,1
CONTINUE:
    addi $t1,$t1,1      #Add 1 for counter
    beq $t1,$t8,Print   #if no more character to compare go PRINT
    j CHECK_STRING      #else compare next character
Print:
    li $v0,4
    la $a0,numCorrectKey
    syscall
    li $v0,1
    add $a0,$0,$t3
    syscall

    li $v0,4
    la $a0,msg_speed
    syscall
    li $v0, 2
    mov.s $f12, $f2
```

```
    syscall
    li $v0,4
    la $a0,msg_unit
    syscall

    li $t9,1
    li $t6,0
    li $t4,10       # set $t4 = 10
    li $s6, 0
    add $t6,$0,$t3      #$t6 will be store the sum of correct character


DISPLAY_DIGITAL:
    div $t6,$t4
    mflo $t7        #$t7 = $t6 div $t4
    la $s2,bytedec      #convert to decimal
    add $s2,$s2,$t7
    lb $a0,0($s2)
    jal   SHOW_7SEG_LEFT        # show on left LED
#------------------------------------------------------------------------
    mfhi $t7        #$t7 = $t6 mod $t4
    la $s2,bytedec
    add $s2,$s2,$t7
    lb $a0,0($s2)
    jal  SHOW_7SEG_RIGHT        # show on right LED
#------------------------------------------------------------------------
-
    li    $t6,0     # Set $t6 to 0 for rerun program
    beq $t9,1,RequestForLoop

RequestForLoop:
    li $v0, 50
    la $a0, msg_request_loop
    syscall
    beq $a0,0,main
    nop
    b EXIT

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

EXIT:
.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 COUNTER_TIMER  0xFFFF0013  #Counter Timer's Address
.eqv MASK_CAUSE_COUNTER 0x00000400  # Bit 10: Counter interrupt

.data
bytedec     : .byte 63,6,91,79,102,109,125,7,127,111
inputString : .space 1000       #khoang trong de luu cac ky tu nhap tu ban phim.
SRCString : .asciiz "bo mon ky thuat may tinh"
msg_counter : .asciiz "\n So ky tu da nhap la :  "
numCorrectKey: .asciiz  "\n So ky tu nhap dung la: "
msg_request_loop: .asciiz "\n ban co muon quay lai chuong trinh? "
msg_speed: .asciiz "\nToc do danh may la: "
msg_unit: .asciiz " cps (character/second)"
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# MAIN Procedure
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
.text
    li   $k0,  KEY_CODE
    li   $k1,  KEY_READY
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# $s4: Sum of character from input's user
# $t4: Number 10
# $t9 - Boolean: For request loop program or not from user
# $s5 - Boolean: true if Enter key is press
# $s6: Counter Time from start to the end of program
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
main:
    li $s4,0
    li $t4,10
    li $t9,0
    li $s5, 0
    li $s6, 0
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
# Enable Counter (Digital Lab Sim)
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
START_COUNTER:
    li $s2, COUNTER_TIMER
    sb $s2, 0($s2)
LOOP:
WAIT_FOR_KEY:
    lw   $t1, 0($k1)                # $t1 = [$k1] = KEY_READY
    beq  $t1, $zero, WAIT_FOR_KEY        # if $t1 == 0 then Polling

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# 1. Store key in array 'inputString'
# 2. Add 1 for Sum of character from user input
# 3.  Check if Current character is Enter key
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
READ_KEY:
    lw $t0, 0($k0)
    la  $t7,inputString
    add $t7,$t7,$s4
    sb $t0,0($t7)
    addi $s4,$s4,1
    beq $t0,10,is_Entered
    nop
    j LOOP


is_Entered:
    li $s5, 1   #If $t0 is Enter key set $s5 = 1
    b END   #branch to END




END_MAIN:
    li $v0,10
    syscall

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# INTERUPT COUNTER
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
.ktext    0x80000180
#-----------------------------------------------------------
# Temporary disable interrupt
#-----------------------------------------------------------
dis_int:
```

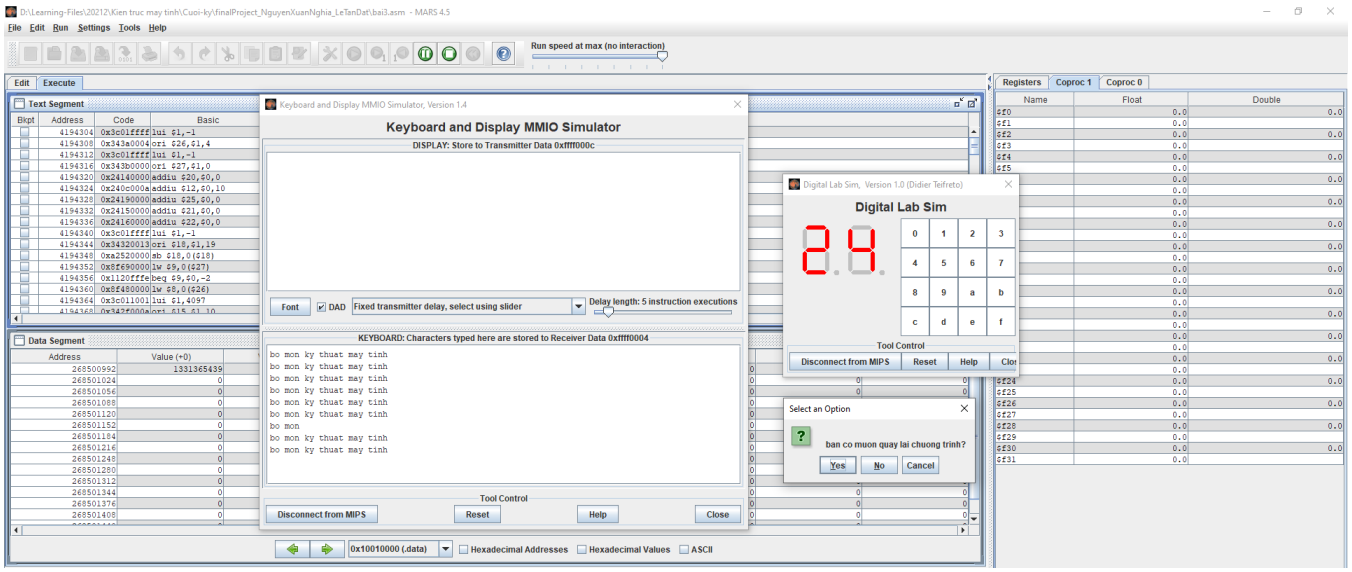
```
    li $s2, COUNTER_TIMER
    sb $zero, 0($s2)
#-------------------------------------------------------
# Processing
#-------------------------------------------------------
get_caus:
    mfc0  $t8, $13
    li    $t2, MASK_CAUSE_COUNTER             # if Cause value confirm Counter..
    and   $at, $t8,$t2
    beq   $at,$t2, Counter_Intr
    j end_process

Counter_Intr:               # Processing Counter Interrupt
    li $v0, 4
    la $a0, msg_counter     #Just print msg about sum of input character
    syscall
    nop
    li $v0, 1 # service 1 is print integer
    add  $a0, $s4, $zero
    syscall # execute
    nop


end_process:
    mtc0 $zero, $13
check_WaitForKey:   #If on Loop Wait for key please dont next_pc
    beq  $t1, $zero, en_int
check_Entered:      #If on Loop main Please font next_pc
    beq $s5, 0, en_int

#-------------------------------------------------------
# Evaluate the return address of main routine
# epc <= epc + 4
#-------------------------------------------------------
next_pc:
    mfc0    $at, $14            # $at <= Coproc0.$14 = Coproc0.epc
    addi    $at, $at, 4  # $at = $at + 4 (next instruction)
    mtc0    $at, $14            # Coproc0.$14 = Coproc0.epc <= $at

#-------------------------------------------------------
# Re-enable interrupt
#-------------------------------------------------------
en_int:
    li $s2, COUNTER_TIMER
    sb $s2, 0($s2)
```

```
SLEEP:      addi $s6, $s6, 5
    addi    $v0,$zero,32
    li      $a0, 5      #Sleep 5ms
    syscall
    nop
#   b      LOOP     #Branch to LOOP

RETURN:
    eret                        # Return from exception

END:
Calc_Speed:
    mtc1  $s6, $f1
    cvt.s.w $f1, $f1

    mtc1 $t4, $f3
    cvt.s.w $f3, $f3

    mtc1 $s4, $f2
    cvt.s.w $f2, $f2

    div.s $f1, $f1, $f3
    div.s $f1, $f1, $f3
    div.s $f1, $f1, $f3

    div.s $f2, $f2, $f1

    li $v0,11
    li $a0,'\n'
    syscall
    li $t1,0        #Count the sum of character which is compared
    li $t3,0                        #Count the sum of correct character
    li $t8,24       #The lenght of Source String
    slt $t7,$s4,$t8     #Compare length of Source String with User's String
    bne $t7,1, CHECK_STRING #Which is lesser $t8 := that
    add $t8,$zero, $s4
    addi $t8,$t8,-1     #Skip Enter character from the end of string.
    beq $t8, 0, Print
CHECK_STRING:
    la $t2,inputString
    add $t2,$t2,$t1
    li $v0,11       #Get character of index $t1  of User's String to store in $t5
    lb $t5,0($t2)
    move $a0,$t5
```

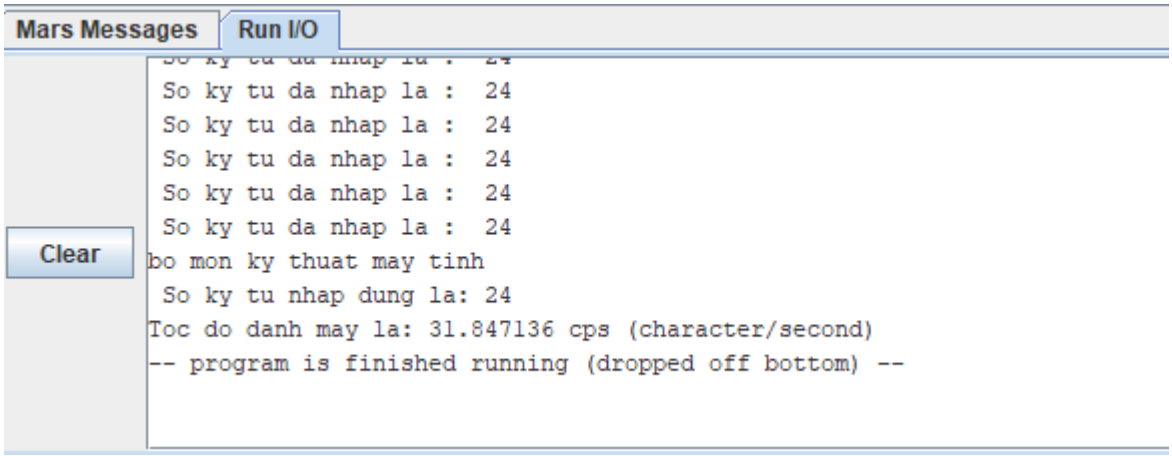
```
    syscall         #Print User's String

    la $t4,SRCString
    add $t4,$t4,$t1
    lb $t6,0($t4)       #Get character of index $t1  of Source String to store in $t6
    bne $t6, $t5,CONTINUE   #Compare $t5, with $t6
    addi $t3,$t3,1
CONTINUE:
    addi $t1,$t1,1      #Add 1 for counter
    beq $t1,$t8,Print   #if no more character to compare go PRINT
    j CHECK_STRING      #else compare next character
Print:
    li $v0,4
    la $a0,numCorrectKey
    syscall
    li $v0,1
    add $a0,$0,$t3
    syscall

    li $v0,4
    la $a0,msg_speed
    syscall
    li $v0, 2
    mov.s $f12, $f2
    syscall
    li $v0,4
    la $a0,msg_unit
    syscall

    li $t9,1
    li $t6,0
    li $t4,10       # set $t4 = 10
    li $s6, 0
    add $t6,$0,$t3      #$t6 will be store the sum of correct character


DISPLAY_DIGITAL:
    div $t6,$t4
    mflo $t7        #$t7 = $t6 div $t4
    la $s2,bytedec      #convert to decimal
    add $s2,$s2,$t7
    lb $a0,0($s2)
    jal   SHOW_7SEG_LEFT        # show on left LED
#-------------------------------------------------------------------------
    mfhi $t7        #$t7 = $t6 mod $t4
```

```
    la $s2,bytedec
    add $s2,$s2,$t7
    lb $a0,0($s2)
    jal  SHOW_7SEG_RIGHT        # show on right LED
#------------------------------------------------------------------------
-
    li    $t6,0     # Set $t6 to 0 for rerun program
    beq $t9,1,RequestForLoop

RequestForLoop:
    li $v0, 50
    la $a0, msg_request_loop
    syscall
    beq $a0,0,main
    nop
    b EXIT

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

EXIT:
```

2.4. Demo



Output:



3. Project 8 – Mô phỏng ổ đĩa RAID5

3.1. Phân tích cách thức thực hiện

Phân tích đề bài :

- mô phỏng hoạt động của RAID 5 với 3 ổ đĩa, với giả định rằng, mỗi block dữ liệu có 4 kí tự.

- Giới hạn chuỗi kí tự nhập vào có độ dài là bội của 8.
- 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      |
 --------------        --------------        --------------
```

Phân tích cách thực hiện

* Chương trình chính được chia làm 3 hàm chính:

- Nhập chuỗi kí tự và kiểm tra chuỗi đó có số kí tự là bội của 8 hay có rỗng không.
- Có hàm RAID5 được chia làm 3 phần:

    + block1 : 4 byte parity được tính từ 2 block đầu tiên sẽ được lưu vào Disk 3. Nếu còn chuỗi cần lưu, hàm sẽ tiếp tục block2

    + block2 : 4 byte parity được tính sẽ lưu vào Disk 2. Nếu còn chuỗi cần lưu, hàm sẽ tiếp tục block3

    + block3 : 4 byte parity được tính sẽ lưu vào Disk 1. Nếu còn chuỗi cần lưu,hàm sẽ quay trở về block1

- Hàm hex để chuyển 4 byte parity từ chuẩn ASCII sang Hexa.

* Cách chạy chương trình :

- Nhập vào 1 chuỗi có số kí tự là bội của 8

- Output sẽ hiển thị các dữ liệu được lưu ở các disk theo cách lưu trữ của RAID.

3.2. Ý nghĩa các thanh ghi sử dụng

- $s1:địa chỉ của Disk1
- $s2:địa chỉ của Disk2
- $s3:địa chỉ của Disk3
- $t3:độ dài chuỗi input
- $t0:index
- $t1:địa chỉ của chuỗi nhập vào
- $t2:string[i];
- $t4:gán giá trị bằng 7 (cho lặp tới 0 để đủ 8 bit)
- $t7:địa chỉ của hex
- $a0:chỉ số của mảng hex - $t8:địa chỉ của chuỗi parity

3.3. Source code

```
.data
        start:  .asciiz "Nhap chuoi ky tu : "
        hex: .byte  '0','1','2','3','4','5','6','7','8','9','a','b','c','d','e','f'
#dung chuyen doi ma ascii sang hexa
        d1: .space 4 #tuong trung cho disk1,disk2,disk3 co 4 byte
        d2: .space 4
        d3: .space 4
        array: .space 32 #luu tru các kí tu dc XOR
        string: .space 1000 #chuoi ki input
        enter: .asciiz "\n"
        error_length: .asciiz "Do dai chuoi khong hop le! Nhap lai.\n"
        disk: .asciiz  "      Disk 1                Disk 2                Disk 3\n"
```

```
        ms1: .asciiz   "-----------------      ---------------       ----------------
\n"
        ms2: .asciiz   "|     "
        ms3: .asciiz   "      |      "
        ms4: .asciiz  "[[ "
        ms5: .asciiz "]]       "
        comma: .asciiz ","
        message: .asciiz "Try again?"

.text
main:
    la $s1, d1              # s1 = address of disk 1
    la $s2, d2              # s2 = address of disk 2
    la $s3, d3              # s3 = address of disk 3
    la $a2, array               # dia chi mang chua parity

    j input
    nop

input:  li $v0, 4               # print " Nhap chuoi ky tu"
    la $a0, start
    syscall

    li $v0, 8               # Get string
    la $a0, string
    li $a1, 1000
    syscall

    move $s0, $a0               # s0 chua dia chi xau moi nhap

    li $v0, 4               # print " Disk1 Disk2 Disk3"
    la $a0, disk
    syscall
    li $v0, 4               # print " ------ "
    la $a0, ms1
    syscall

#-----------------------kiem tra do dai co chia het cho 8 khong-----------------------
--
length:
    addi $t3, $zero, 0          # t3 = length
    addi $t0, $zero, 0          # t0 = index

check_char:
# Hàm kiem tra kí tu:  kí tu ket thúc: "\n"
```

```
    add $t1, $s0, $t0           # t1 = address of string[i]
    lb  $t2, 0($t1)                 # t2 = string[i]
    beq $t2, 10, test_length        # khi stirng[i]  = '\n' ket thuc kiem tra kí t? k?t
thúc
    nop

    addi $t3, $t3, 1            # length++
    addi $t0, $t0, 1                # index++
    j check_char
    nop

test_length:
    move $t5, $t3               # t5 chua dia chi length
    beq $t0,0,error                 # if has only "\n" -> error

    and $t1, $t3, 0x0000000f            # xoa het cac byte cua $t3 ve 0, chi giu lai
byte cuoi
    bne $t1, 0, test1           # byte cuoi bang 0 hoac 8 thi so chia het cho 8
    j block1
    nop

test1:  beq $t1, 8, block1          # neu byte cuoi  != 8 va != 0 => error
    j error
    nop

error:  li $v0, 4               # Ham in loi thong bao
    la $a0, error_length
    syscall
    j input                 # bat nguoi dung nhap lai input
    nop

#-------------------------------ket thuc kiem tra do dai-------------------------------
---


HEX:
# Ham lay parity
# Co 1 dau vao la t8 chua parity string roi chuyen tu ascii sang hexa
    li $t4, 7               #t4 = 7

loopH:
    blt $t4, $0, endloopH           # t4 < 0  -> endloop
    sll $s6, $t4, 2             # s6 = t4*4
    srlv $a0, $t8, $s6          # a0 = t8>>s6
```

```
    andi $a0, $a0, 0x0000000f       # a0 = a0 & 0000 0000 0000 0000 0000 0000 0000 1111
=> lay byte cuoi cung cua a0
    la $t7, hex                 # t7 = adrress of hex
    add $t7, $t7, $a0           # t7 = t7 + a0
    bgt $t4, 1, nextc           # if t4 > 1 , jump to nextC
    lb $a0, 0($t7)              # print hex[a0]
    li $v0, 11
    syscall


nextc:  addi $t4,$t4,-1             # t4 --
    j loopH
    nop

endloopH:
    jr $ra
    nop


# Ham mo phong RAID 5
# xet 6 khoi dau -
#lan 1: luu  2 khoi 4-byte vao  disk 1,2; xor vao disk 3
RAID5:
# RAID 5 gom 3 phan,
#block 1 : byte parity luu vao disk 3
#block 2 : byte parity luu vao disk 2
#block 3 : byte parity luu vao disk 1
block1:
#Funtion block1:Lan thu nhat xet 2 khoi 4 byte  luu vao Disk 1 , Disk 2 ;
#Byte parity luu vao Disk 3;

    addi $t0, $zero, 0          # so byte duoc in ra (4 byte)
    addi $t9, $zero, 0
    addi $t8, $zero, 0
    la $s1, d1              # s1 = adress of d1
    la $s2, d2              # s2 = address of d2
    la $a2, array               #

print11:
    li $v0, 4               # print message2 : "|     "
    la $a0, ms2
    syscall

#   vi du DCE.****
b11:
```

```
# luu DCE. vao disk 1
    lb $t1, ($s0)               # t1 = first value of input string
    addi $t3, $t3, -1           # t3 = length -1,giam do dai sau can xet
    sb $t1, ($s1)               # store t1 to disk 1
b12:
# luu **** vao disk 2
    add $s5, $s0, 4             # s5 = s0 +4
    lb $t2, ($s5)               # t2 = inputstring[5]
    addi $t3, $t3, -1           # t3 = t3  - 1  , giam do dai xau can xet
    sb $t2, ($s2)               # store t2 vao disk 2
b13:
# luu ket qua xor vao disk 3
    xor $a3, $t1, $t2           # a3 = t1 xor t2
    sw $a3, ($a2)               # luu a3 vao dia chi chuoi a2
    addi $a2, $a2, 4                #   parity string
    addi $t0, $t0, 1                # xet char tiep theo
    addi $s0, $s0, 1                # loai bo ki tu vua xet , Vi du : "D"
    addi $s1, $s1, 1                # tang dia chi disk 1 len 1
    addi $s2, $s2, 1                # tang dia chi disk 2 len 1
    bgt $t0, 3, reset           #  da xet duoc 4 byte , reset disk
    j b11
    nop
reset:
    la $s1, d1              # reset con tro ve  disk 1 VD : "D" trong "DCE."
    la $s2, d2              # reset con tro ve disk 2

print12:                    #in Disk 1
    lb $a0, ($s1)               #print each char  in Disk 1
    li $v0, 11
    syscall
    addi $t9, $t9, 1
    addi $s1, $s1, 1
    bgt $t9, 3, next11          # sau khi in du 4 lan => in het Disk 1
    j print12
    nop

next11:                     #Ham chuan bi bat dau de print Disk 2    "|         |"
    li $v0, 4
    la $a0, ms3
    syscall
    li $v0, 4
    la $a0, ms2
    syscall

print13:                        # Ham print disk 2
```

```
    lb $a0, ($s2)
    li $v0, 11
    syscall
    addi $t8, $t8, 1
    addi $s2, $s2, 1
    bgt $t8, 3, next12          # in dc 4 byte => xong Disk 2
    j print13
    nop

next12:                     # ham chuan bi in Disk 3
    li $v0, 4
    la $a0, ms3
    syscall
    li $v0, 4
    la $a0, ms4
    syscall
    la $a2, array           # a2 = address of parity string[i]
    addi $t9, $zero, 0      # t9 = i

print14:                    # Ham chuyen doi  parity string -> ma ASCII va in ra man
hinh
    lb $t8, ($a2)           # t8 = adress of parity string[i]
    jal HEX
    nop
    li $v0, 4
    la $a0, comma           # print  " , "
    syscall

    addi $t9, $t9, 1        # parity string's index  + 1
    addi $a2, $a2, 4        # bo qua parity string da xet'
    bgt $t9, 2, end1        # in ra 3 parity dau co dau ",", parity cuoi cung k co
    j print14
end1:               # in ra parity cuoi cung va hoan thanh Disk 3
    lb $t8, ($a2)
    jal HEX
    nop
    li $v0, 4
    la $a0, ms5
    syscall

    li $v0, 4           # xuong dong , bat dau khoi block moi
    la $a0, enter
    syscall
    beq $t3, 0, exit1       # neu length string con lai can xet = 0 , exit
    j block2            # neu con lai ki tu can xet => block2
```

```
    nop

#-------------------------------------------------------------------------------

block2:
#Ham block 2 :
# xet 2 khoi 4  byte tiep theo  vao Disk 1 va Disk 3;  byte parity vao Disk 2

    la $a2, array
    la $s1, d1              # s1 = address of Disk 1
    la $s3, d3              # s3 =            Disk 3
    addi $s0, $s0, 4
    addi $t0, $zero, 0

print21:
# print "|     "
    li $v0, 4
    la $a0, ms2
    syscall

b21:
# xet tung byte trong 4 byte dau vao Disk 1
    lb $t1, ($s0)               # t1 = address of Disk 1
    addi $t3, $t3, -1           # length con' phai kiem tra   -1
    sb $t1, ($s1)
b23:
# xet 4 byte ke tiep vao Disk 3
    add $s5, $s0, 4
    lb $t2, ($s5)
    addi $t3, $t3, -1
    sb $t2, ($s3)

b22:
#Tinh 4 byte parity vao Disk 2
    xor $a3, $t1, $t2
    sw $a3, ($a2)
    addi $a2, $a2, 4
    addi $t0, $t0, 1
    addi $s0, $s0, 1
    addi $s1, $s1, 1
    addi $s3, $s3, 1
    bgt $t0, 3, reset2
    j b21
    nop
reset2:
```

```
    la $s1, d1          # reset de chuan bi print ra Disk 1
    la $s3, d3          # reset de chuan bi print ra Disk 3
    addi $t9, $zero, 0      # index

print22:
# print Disk 1
    lb $a0, ($s1)
    li $v0, 11
    syscall
    addi $t9, $t9, 1
    addi $s1, $s1, 1
    bgt $t9, 3, next21
    j print22
    nop

next21:     # print khoang cach
    li $v0, 4
    la $a0, ms3
    syscall
    la $a2, array
    addi $t9, $zero, 0
    li $v0, 4
    la $a0, ms4
    syscall

print23:    # print Disk 2 chua byte parity
    lb $t8, ($a2)
    jal HEX             # chuyen doi ve ASCII
    nop
    li $v0, 4
    la $a0, comma           #print ","
    syscall
    addi $t9, $t9, 1
    addi $a2, $a2, 4
    bgt $t9, 2, next22
    j print23
    nop

next22:
#print Disk 2 theo ACSII
    lb $t8, ($a2)
    jal HEX
    nop

    li $v0, 4
```

```
    la $a0, ms5
    syscall

    li $v0, 4
    la $a0, ms2
    syscall
    addi $t8, $zero, 0

print24:
# print Disk 3
    lb $a0, ($s3)
    li $v0, 11
    syscall
    addi $t8, $t8, 1
    addi $s3, $s3, 1
    bgt $t8, 3, end2
    j print24
    nop

end2:
# Neu string can xet da het thi nhay den nhan exit1
# chua het thi tiep tuc block3
    li $v0, 4
    la $a0, ms3
    syscall
    li $v0, 4
    la $a0, enter
    syscall
    beq $t3, 0, exit1
#-------------------------------------------------------------
block3:
# Byte parity duoc luu o Disk1
# 2 block 4 byte dc luu vao Disk 2 , Disk 3
    la $a2, array
    la $s2, d2
    la $s3, d3
    addi $s0, $s0, 4            # xet den vi tri 4 byte hien tai
    addi $t0, $zero, 0          # index
print31:
# chuan bi print parity print: "[[ "
    li $v0, 4
    la $a0, ms4
    syscall
b32:
#  byte stored in Disk 2
```

```
#Vi du DCE.****ABCD1234HUSTHUST
    lb $t1, ($s0)           # in first loop, t1 = first H
    addi $t3, $t3, -1
    sb $t1, ($s2)
b33:
    # store in Disk 3 first
    add $s5, $s0, 4         #
    lb $t2, ($s5)           # in first loop , t2 = the second "H"
    addi $t3, $t3, -1       # stored in disk 3
    sb $t2, ($s3)           # stored t2 in disk 3

b31:
# ham xor tinh parity
    xor $a3, $t1, $t2       # a3 = parity number
    sw $a3, ($a2)           # stored in parity string
    addi $a2, $a2, 4        # parity string's index + 4
    addi $t0, $t0, 1        # index so char dang xet
    addi $s0, $s0, 1        # loai bo ki tu da xet , VD: "H", string dang xet la
"USTHUST"
    addi $s2, $s2, 1        #   disk2 +1
    addi $s3, $s3, 1        #   disk 3 +1
    bgt $t0, 3, reset3      # net xet duoc 4 lan , thoat khoi vong lap
    j b32               # neu chua xet du 4 byte , tiep tuc xet
    nop
reset3:
# to first of disk2 , disk 3
    la $s2, d2
    la $s3, d3
    la $a2, array
    addi $t9, $zero, 0      #index

print32:
# Ham' print parity byte duoi dang ASCII
    lb $t8, ($a2)           # luu chuoi can chuyen duoi ASCII
    jal HEX             # dung ham HEX de chuyen duoi ve ASCII
    nop
    li $v0, 4           # print
    la $a0, comma
    syscall

    addi $t9, $t9, 1
    addi $a2, $a2, 4        # loai bo parity string da duoc xet
    bgt $t9, 2, next31      # neu in du 3 lan dau phay -> next31
    j print32
    nop
```

```
next31:
# print 1 byte parity con' lai
    lb $t8, ($a2)
    jal HEX
    nop

    li $v0, 4
    la $a0, ms5
    syscall
    li $v0, 4
    la $a0, ms2
    syscall
    addi $t9, $zero, 0

print33:
#print disk 2, print 4 byte from Disk 2
    lb $a0, ($s2)
    li $v0, 11
    syscall
    addi $t9, $t9, 1
    addi $s2, $s2, 1
    bgt $t9, 3, next32
    j print33
    nop

next32:
# print ki tu ngan cach
    addi $t9, $zero, 0
    addi $t8, $zero, 0
    li $v0, 4
    la $a0, ms3
    syscall
    li $v0, 4
    la $a0, ms2
    syscall
print34:
#  print  4 byte from Disk 3
    lb $a0, ($s3)
    li $v0, 11
    syscall
    addi $t8, $t8, 1
    addi $s3, $s3, 1
    bgt $t8, 3, end3
    j print34
```

```
    nop

end3:
#in ra cac ki tu ket thuc khi liet ke Disk 3
    li $v0, 4
    la $a0, ms3         # ki tu : "      |"
    syscall

    li $v0, 4
    la $a0, enter           # ki tu xuong dong
    syscall
    beq $t3, 0, exit1       # neu ko con ki tu can xet -> exit
                    #neu con ki tu can xet -> tro ve block1

#-end 6 block 4-byte dau
# chuyen sang 6 block 4-byte tiep theo

nextloop: addi $s0, $s0, 4      #bo qua 4 ki tu da xet roi
    j block1
    nop

exit1:  # in ra dong ------ va ket thuc mo phong RAID
    li $v0, 4
    la $a0, ms1
    syscall
    j ask
    nop

#ket thuc mo phong RAID 5

#try again
ask:    #li $v0, 50         #ask if try again
    #la $a0, message
    #syscall
    #beq $a0, 0, clear      # a0 :     0 = yes;  1 = NO ;  2 = cancel
    #nop
    j exit
    nop

# Hàm clear: dua string ve trang thai ban dau
clear:
    la $s0, string
    add $s3, $s0, $t5   # s3: dia chi byte cuoi cung duoc su dung trong string
    li $t1, 0       # set t1 = 0
```

```
goAgain:        # Dua string ve trang thai rong~ de bat dau lai .
    sb $t1, ($s0)       # set byte o dia chi s0 thanh 0
    nop
    addi $s0, $s0, 1
    bge $s0, $s3, input
    nop
    j goAgain
    nop
#end try again

exit:   li $v0, 10
    syscall
```

3.4. Kết quả chạy chương trình