


Trường Đại học Bách khoa Hà Nội
Trường Công nghệ thông tin và Truyền thông

Môn thực hành
Kiến trúc máy tính

Báo cáo
Bài tập lớn

Giáo viên hướng dẫn:
ThS. Lê Bá Vui

Sinh viên thực hiện:

Vũ Minh Hiếu	20194563	Đề tài số 2
Nguyễn Tiến Dũng	20204957	Đề tài số 8

Mục lục

1.
Đề tài 2:
Vẽ hình trên màn hình Bitmap

Viết chương trình vẽ một quả bóng hình tròn di chuyển trên màn hình mô phỏng Bitmap của Mars. Nếu đối tượng đập vào cạnh của màn hình thì sẽ di chuyển theo chiều ngược lại.

1.1. Yêu cầu

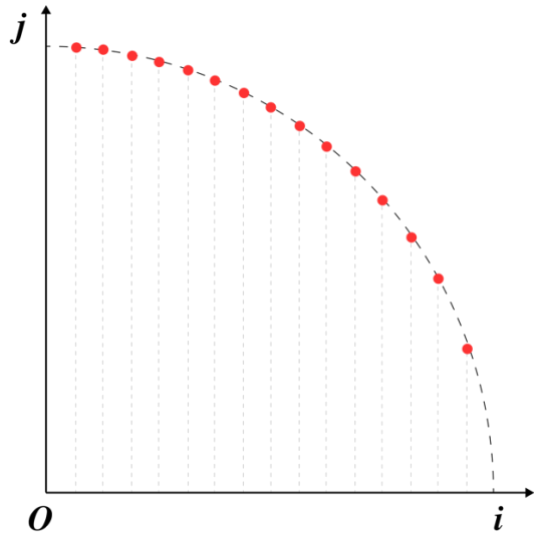
- Thiết lập màn hình ở kích thước 512x512.
  Kích thước pixel 1x1.
- Chiều di chuyển phụ thuộc vào phím người dùng bấm trong bộ giả lập Keyboard and Display MMIO Simulator, gồm có:
  - Di chuyển lên (W)
  - Di chuyển xuống (S)
  - Sang trái (A)
  - Sang phải (D)
  - Tăng tốc độ (Z)
  - Giảm tốc độ (X)
- Vị trí bóng ban đầu ở giữa màn hình.

1.2. Ý tưởng thực hiện

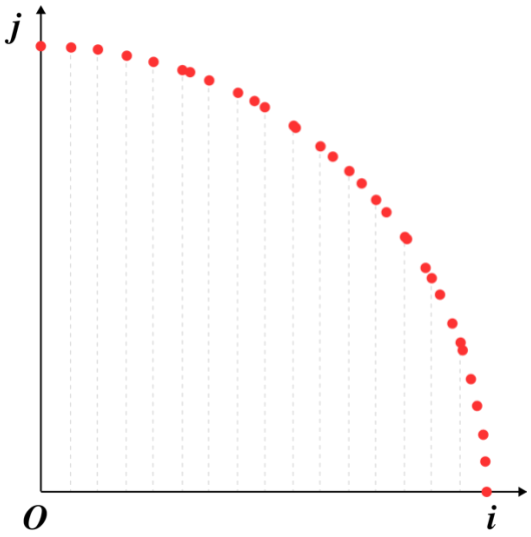
- Để làm một đối tượng di chuyển thì chúng ta sẽ xóa đối tượng ở vị trí cũ và vẽ đối tượng ở vị trí mới.
- Để xóa đối tượng chúng ta chỉ cần vẽ đối tượng đó với màu là màu nền.
- Để thay đổi tốc độ di chuyển, ta giảm thời gian chờ (delay) giữa mỗi lần thay đổi vị trí đường tròn.

1.3. Phân tích cách thực hiện

- Khởi tạo 3 tập lệnh bó macro gồm: `delay`, `branchIfLessOrEqual` và `setColorAndDrawCirle`.
- Hàm `circleInit` dùng để tạo mảng dữ liệu và lưu trữ tọa độ các điểm trên đường tròn vào mảng dữ liệu `CIRCLE_DATA`



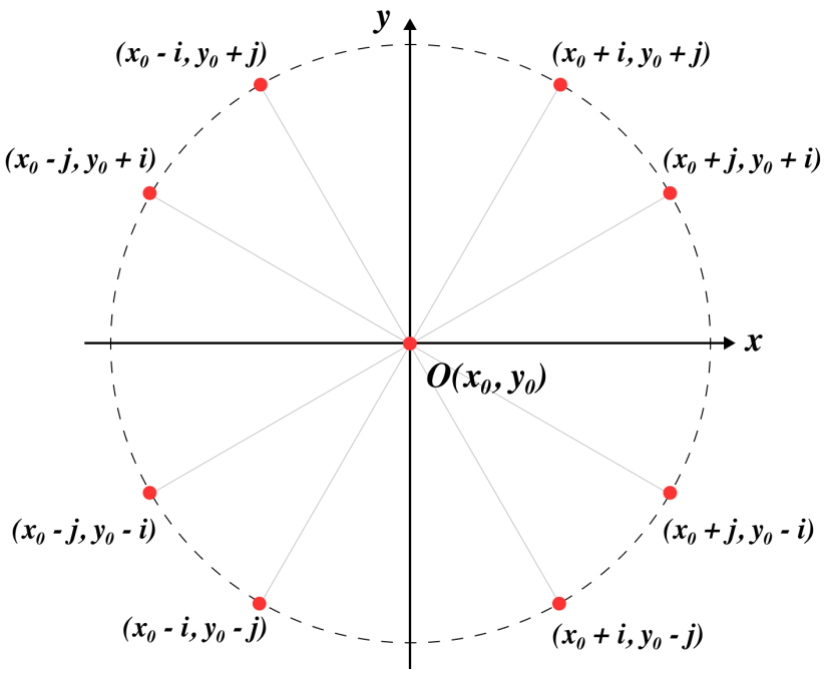
Tập hợp điểm j trong CIRCLE_DATA



Cung một phần tư khi vẽ lên

- Hàm `readKeyboard` dùng để đọc dữ liệu người dùng nhập vào từ bàn phím. Đầu tiên, dùng hàm `positionCheck` để kiểm tra xem đã có ký tự nào được nhập vào hay chưa? Nếu chưa nhập thì cho phép người dùng nhập vào từ bàn phím. Nếu đã nhập thì nhảy xuống hàm `positionCheck` rồi lần lượt vào các hàm `checkRightEdge`, `checkLeftEdge`, `checkTopEdge` và `checkBottomEdge` để kiểm tra xem đường tròn đã chạm các mép màn hình hay chưa?
- Nếu các điều kiện trên không thỏa mãn nghĩa là đường tròn chưa chạm mép nào thì nhảy xuống hàm `draw`, đổi màu đường tròn hiện tại sang màu nền, cập nhập vị trí mới và đổi màu đường tròn sang màu đỏ.

- Hàm `drawCircle` dùng để vẽ đường tròn, ta dùng hàm hàm `drawCirclePoint` để vẽ đồng thời 8 điểm ảnh đống xứng qua 2 trục, tạo nên một đường tròn.



Trong đó:

1.4. Ý nghĩa của các thanh ghi khi được sử dụng

Global	$s0	Toạ độ x0 của tâm đường tròn
	$s1	Toạ độ y0 của tâm đường tròn
	$s2	Bán kính R đường tròn
	$s3	Chiều rộng màn hình
	$s4	Chiều cao màn hình
	$s5	Màu của viền đường tròn
	$s7	dx
	$t8	dy
	$t6	Khoảng thời gian delay giữa các lần di chuyển hình tròn
sqrt	$v0	Số cần căn bậc hai
	$a0	Kết quả căn bậc hai
readKeyboard	$k1	Ready bit của Keyboard MMIO Simulator

	$k0	Mã ASCII của ký tự vừa nhập
drawCirclePoint	$a0	i
	$a1	j
	$t1	Kết quả Xi = x0 [i,j]
	$t4	Kết quả Yi = y0 [i,j]

1.5. Mã nguồn

```
.eqv SCREEN     0x10010000      # Base Addresss ở cài đặt Bitmap Display
.eqv BORDER     0xFFC72A        # Màu của viền đường tròn
.eqv RADIUS     24              # Bán kính đường tròn
.eqv BACKGROUND 0x000000        # Màu nền

.eqv KEY_a      0x61
.eqv KEY_s      0x73
.eqv KEY_d      0x64
.eqv KEY_w      0x77
.eqv KEY_z      0x7A
.eqv KEY_x      0x78

.eqv KEY_A      0x41
.eqv KEY_S      0x53
.eqv KEY_D      0x44
.eqv KEY_W      0x57
.eqv KEY_Z      0x5A
.eqv KEY_X      0x58

.eqv KEY_ENTER  0x0a

.eqv DELTA_X    10
.eqv DELTA_Y    10
.eqv DELAY_TIME 150
.eqv KEY_CODE   0xFFFF0004
.eqv KEY_READY  0xFFFF0000


#-------------------------------------------------------------------------
# Delay chương trình
```

```
# Khoảng thời gian delay giữa các lần di chuyển hình tròn (ms)

.macro delay
        li      $v0, 32         # Gọi sleep
        add     $a0, $t6, 0
        syscall
.end_macro

.macro branchIfLessOrEqual(%r1, %r2, %branch)
        sle     $v0, %r1, %r2
        bnez    $v0, %branch
.end_macro

.macro setColorAndDrawCirle(%color)
        li      $s5, %color     # Đặt màu viền theo màu nền
                                # để xoá hình tròn cũ
        jal     drawCircle
.end_macro

.kdata
CIRCLE_DATA:    .space 512      # Mảng gồm 512 phần tử

.text
        li      $s0, 256        # x0 = 256      Toạ độ x của tâm đường tròn
        li      $s1, 256        # y0 = 256      Toạ độ y của tâm đường tròn
        li      $s2, RADIUS     # R = 24        Bán kính đường tròn
        li      $s3, 512        # SCREEN_WIDTH = 512    Chiều rộng màn hình
        li      $s4, 512        # SCREEN_HEIGHT = 512   Chiều cao màn hình
        li      $s5, BORDER     # Màu của viền hình tròn (Màu vàng)
        li      $s7, 0          # dx = 0
        li      $t8, 0          # dy = 0
        li      $t6, DELAY_TIME # currentDelay = 150


#---------------------------------------------------------------------------
# circleInit()
# Hàm khởi tạo đường tròn
# Tạo array lưu tạo độ các điểm của đường tròn
# A[i] = j

circleInit:
```

```
        la      $t5, CIRCLE_DATA        # Trỏ vào địa chỉ
                                        # mảng dữ liệu của đường tròn
        li      $t0, 0                  # i = 0

loop:                                   # for loop i -> R
        slt     $v0, $t0, $s2           # if (i < R)
        beqz    $v0, end_circleInit
        mul     $s6, $s2, $s2           # R^2
        mul     $t3, $t0, $t0           # i^2
        sub     $t3, $s6, $t3           # $t3 = R^2 - i^2
        move    $v0, $t3
        jal     sqrt
                                #         /|
                                #       /  |
                                #  R  /    |  j
                                #   /______|
                                # O    i
        sw      $a0, 0($t5)     # Lưu j = sqrt(R^2 - i^2) vào array
        addi    $t0, $t0, 1     # i++
        add     $t5, $t5, 4     # Đi đến vị trí tiếp theo của array CIRCLE_DATA
        j       loop
end_circleInit:


#-------------------------------------------------------------------------
# readKeyboard()
# Hàm xử lý ký tự nhập vào từ bàn phím

programLoop:
readKeyboard:
        lw      $k1, KEY_READY          # Kiểm tra xem đã nhập ký tự nào chưa
        beqz    $k1, positionCheck
        lw      $k0, KEY_CODE
        beq     $k0, KEY_a, case_a
        beq     $k0, KEY_A, case_a
        beq     $k0, KEY_s, case_s
        beq     $k0, KEY_S, case_s
        beq     $k0, KEY_d, case_d
        beq     $k0, KEY_D, case_d
        beq     $k0, KEY_w, case_w
        beq     $k0, KEY_W, case_w
```

```
        beq     $k0, KEY_Z, case_z
        beq     $k0, KEY_z, case_z
        beq     $k0, KEY_X, case_x
        beq     $k0, KEY_x, case_x
        beq     $k0, KEY_ENTER, case_enter
        j       positionCheck
        nop
case_a:
        jal     moveToLeft
        j       positionCheck
case_s:
        jal     moveToDown
        j       positionCheck
case_d:
        jal     moveToRight
        j       positionCheck
case_z:
        jal     speedUp
        j       positionCheck
case_x:
        jal     speedDown
        j       draw
case_w:
        jal     moveToUp
        j       draw
case_enter:
        j       endProgram

positionCheck:
checkRightEdge:
        add     $v0, $s0, $s2    # x0 + R
        add     $v0, $v0, $s7    # if (x0 + R + DELTA_X >= SCREEN_WIDTH) then
    moveToLeft
        branchIfLessOrEqual($v0, $s3, checkLeftEdge)    # else check left edge
        jal     moveToLeft
        nop
checkLeftEdge:
        sub     $v0, $s0, $s2
        add     $v0, $v0, $s7    # if (x0 - R + DELTA_X <= 0) then moveToRight
        branchIfLessOrEqual($zero, $v0, checkTopEdge)   # else check top edge
        jal     moveToRight
```

```
        nop
checkTopEdge:
        sub     $v0, $s1, $s2
        add     $v0, $v0, $t8   # if (y0 - R + DELTA_Y <= 0) then moveToDown
        branchIfLessOrEqual($zero, $v0, checkBottomEdge) # else check bottom edge
        jal     moveToDown
        nop
checkBottomEdge:
        add     $v0, $s1, $s2
        add     $v0, $v0, $t8   # if (y0 + R + DELTA_Y >= SCREEN_HEIGHT) then
    moveToUp
        branchIfLessOrEqual($v0, $s4, draw)     # else vẽ đường tròn
        jal     moveToUp
        nop


#------------------------------------------------------------------------
# Hàm vẽ đường tròn

draw:
        setColorAndDrawCirle(BACKGROUND)        # Vẽ đường tròn trùng màu nên
        add     $s0, $s0, $s7                   # Cập nhật toạ độ mới của đường tròn
        add     $s1, $s1, $t8

        setColorAndDrawCirle(BORDER)            # Vẽ đường tròn mới
        delay                                   # Dừng chương trình 1 lúc
        j       programLoop

endProgram:
        setColorAndDrawCirle(BACKGROUND)
        li      $v0, 10                         # Gọi exit
        syscall


# ^^^^^^^^^^^^^^^^^^^^^^
#------------------------------------------------------------------------
# Hàm vẽ đường tròn
# Sử dụng data của array CIRCLE_DATA tạo ở circleInit

drawCircle:
        add     $sp, $sp, -4
```

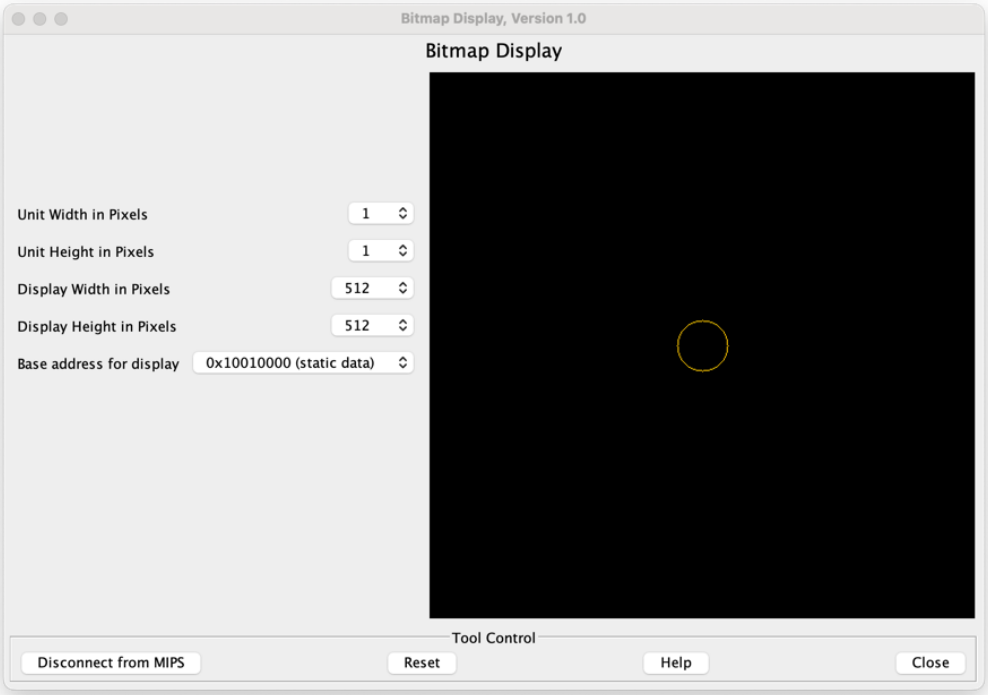
```
        sw      $ra, 0($sp)
        li      $t0, 0                  # Khởi tạo i = 0
loop_drawCircle:
        slt     $v0, $t0, $s2           # if (i < R)
        beqz    $v0,  end_drawCircle

        sll     $t5, $t0, 2             # CIRCLE_DATA[i]
        lw      $t3, CIRCLE_DATA($t5)   # Load j to $t3

        move    $a0, $t0                # i = $a0
        move    $a1, $t3                # j = $a1
        jal     drawCirclePoint # Draw (x0 + i, y0 + j), (x0 + j, y0 + i)
        sub     $a1, $zero, $t3
        jal     drawCirclePoint # Draw (x0 + i, y0 - j), (x0 + j, y0 - i)
        sub     $a0, $zero, $t0
        jal     drawCirclePoint # Draw (x0 - i, y0 - j), (x0 - j, y0 - i)
        add     $a1, $zero, $t3
        jal     drawCirclePoint # Draw (x0 - i, y0 + j), (x0 - j, y0 + i)

        addi    $t0, $t0, 1
        j       loop_drawCircle
end_drawCircle:
        lw      $ra, 0($sp)
        add     $sp, $sp, 0
        jr      $ra


#---------------------------------------------------------------------------
# Hàm vẽ điểm trên đường tròn
# Vẽ đồng thời 2 điểm
# (x0 + i, y0 + j) và (x0 + j, x0 + i)
# i = $a0, j = $a1
# Xi = $t1, Yi = $t4

drawCirclePoint:
        add     $t1, $s0, $a0           # Xi = x0 + i
        add     $t4, $s1, $a1           # Yi = y0 + j
        mul     $t2, $t4, $s3           # Yi * SCREEN_WIDTH
        add     $t1, $t1, $t2           # Xi + Yi * SCREEN_WIDTH
                                        # (Toạ độ 1 chiều của điểm ảnh)
        sll     $t1, $t1, 2             # Địa chỉ tương đối của điểm ảnh
```

```
        sw      $s5, SCREEN($t1)        # Draw ảnh

        add     $t1, $s0, $a1           # Xi = x0 + j
        add     $t4, $s1, $a0           # Yi = Y0 + i
        mul     $t2, $t4, $s3           # Yi * SCREEN_WIDTH
        add     $t1, $t1, $t2           # Xi + Yi * SCREEN_WIDTH
                                        # (Toạ độ 1 chiều của điểm ảnh)
        sll     $t1, $t1, 2             # Địa chỉ tương đối của điểm ảnh
        sw      $s5, SCREEN($t1)        # Draw ảnh

        jr      $ra


#---------------------------------------------------------------------------
# Các hàm di chuyển

moveToLeft:
        li      $s7, -DELTA_X
        li      $t8, 0
        jr      $ra
moveToRight:
        li      $s7, DELTA_X
        li      $t8, 0
        jr      $ra
moveToUp:
        li      $s7, 0
        li      $t8, -DELTA_Y
        jr      $ra
moveToDown:
        li      $s7, 0
        li      $t8, DELTA_Y
        jr      $ra
speedUp:
        addi    $v0, $0, 20
        branchIfLessOrEqual($t6, $v0, end_speedUp)
        addi    $t6, $t6, -10
end_speedUp:
        jr      $ra
speedDown:
        addi    $t6, $t6, 10
        jr      $ra
```

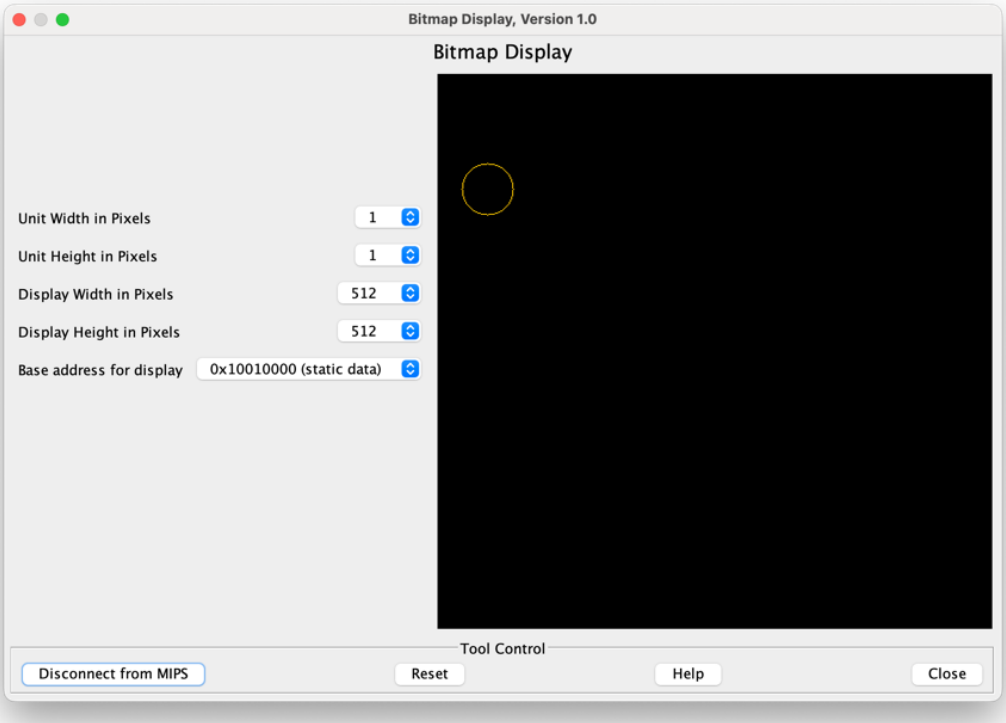
```
#-------------------------------------------------------------------------
# Square Root
# Để sử dụng floating point thì phải chuyển sang coprocessor
# $v0 = S, $a0 = sqrt(S)

sqrt:
        mtc1    $v0, $f0        # Đưa từ $v0 vào $f0
        cvt.s.w $f0, $f0
        sqrt.s  $f0, $f0
        cvt.w.s $f0, $f0
        mfc1    $a0, $f0        # Đưa lại từ $f0 vào $a0
        jr      $ra
```

1.6. Ảnh chụp kết quả

- Chưa nhập dữ liệu

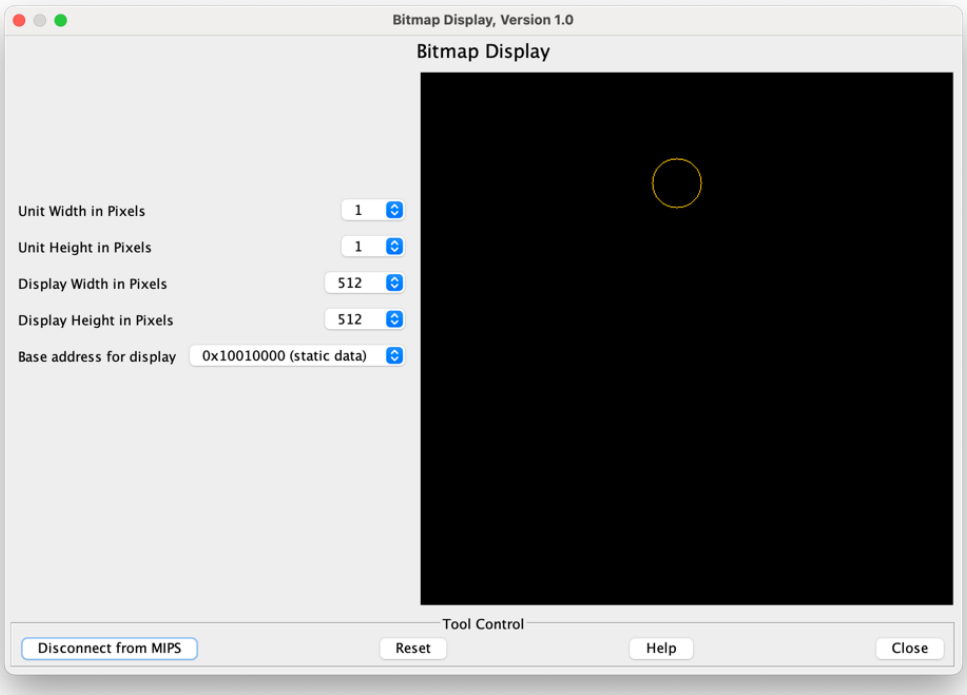


- Di chuyển trái khi nhập a hay A

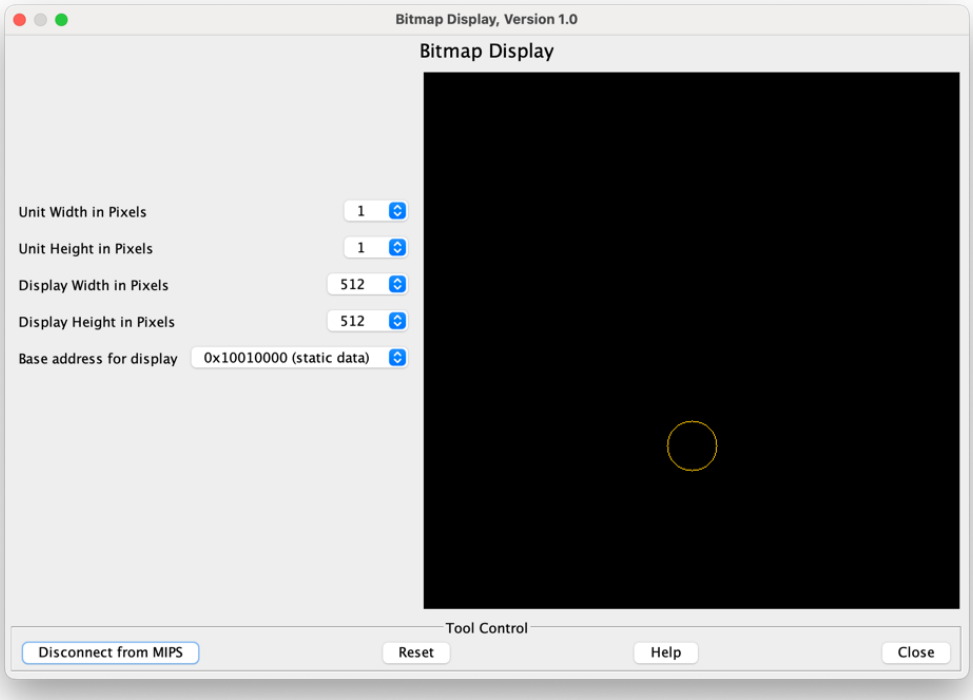


- Di chuyển phải khi nhập d hay D

- Di chuyển lên khi nhập w hay W



- Di chuyển xuống khi nhập s hay S



2.
Đề tài 8:
Mô phỏng ổ đĩa RAID 5

Hệ thống ổ đĩa RAID 5 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 dưới.

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

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 trên, 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 '*'`

2.1. 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:
  - Block 1: 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
  - Block 2: 4 byte parity được tính sẽ lưu vào Disk 2. Nếu còn chuỗi cần lưu, hàm sẽ tiếp tục block 3
  - Block 3: 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.

2.2. 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

2.3. Ý nghĩa các thanh ghi được 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

2.4. Mã nguồn

```
.data
start: .asciiz "Nhap chuoi ky tu : "
# dung chuyen doi ma ascii sang hexa
hex: .byte '0','1','2','3','4','5','6','7','8','9','a','b','c','d','e','f'
d1: .space 4            # tuong trung cho disk1,disk2,disk3 co 4 byte
d2: .space 4
d3: .space 4
array: .space 32        # Lưu trữ các ký tự được XOR
string: .space 5000     # Input string
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:
# H□m kiem tra k□ tu:  k□ tu ket th□c: "\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 k□ t? k?t th□c
        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
```

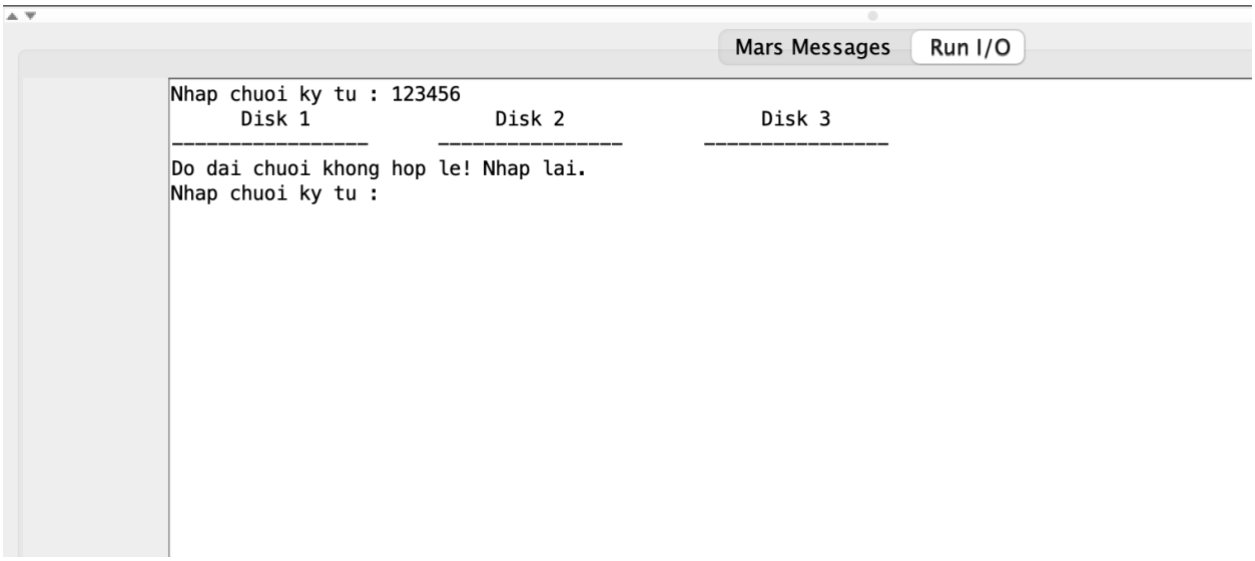
```
print32:
# Ham' print parity byte duoi dang ASCII
        lb $t8, ($a2)                   # luu chuoi can chuyen duoi ASCII
        jal HEX                         # dung ham HEX de chuyen duoi ve ASCII
        nop
        li $v0, 4                       # print
        la $a0, comma
        syscall

        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
```

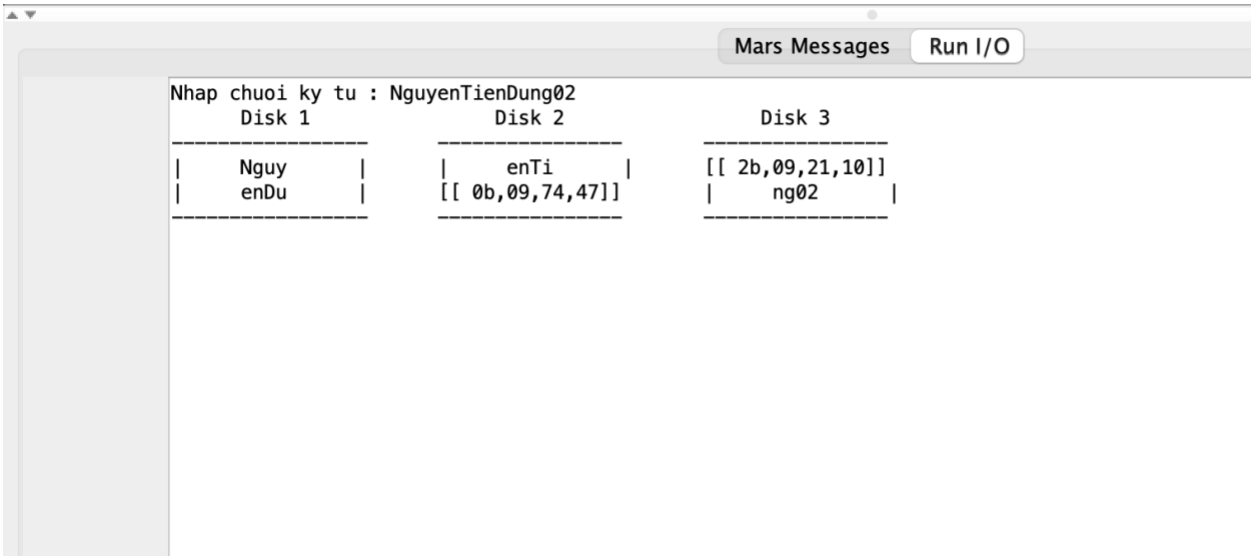
```
        la $a0, ms1
        syscall
        j ask
        nop

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

# H□m 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
```

2.5. Ảnh chụp kết quả

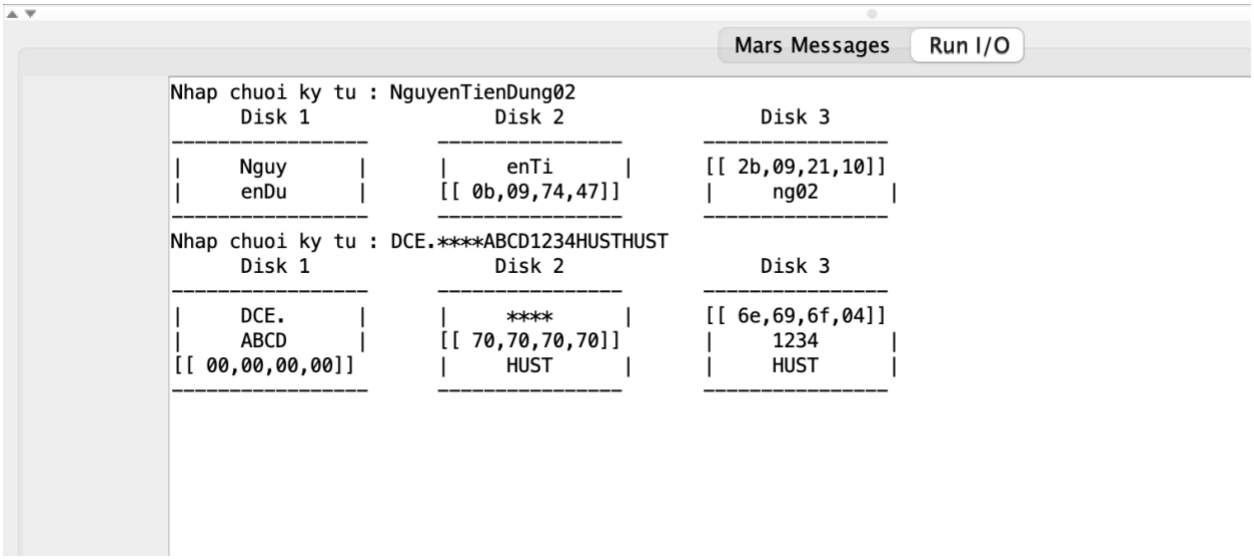
- Chuỗi nhập vào: 123456



- Chuỗi nhập vào: NguyenTienDung02



- Chuỗi nhập vào: DCE.****ABCD1234HUSTHUST



- Chuỗi nhập vào: HUSTHUSTHUSTHUSTHUSTHUSTHUSTHUST