ĐẠI HỌC BÁCH KHOA HÀ NỘI

TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

———————— * ————————

BÁO CÁO CUỐI KỲ

THỰC HÀNH KIẾN TRÚC MÁY TÍNH

Giảng viên hướng dẫn: Thầy Lê Bá Vui

Nhóm sinh viên thực hiện: Nhóm 5

Họ và tên	MSSV	Bài tập số
Hoàng Văn Thể	20205129	8
Nguyễn Doãn Anh Thái	20205021	9

Hà Nội, tháng 07 năm 2022

MỤC LỤC

MỤC LỤC 2

PHẦN I. MÔ PHỎNG Ổ ĐĨA RAID 5 3

PHẦN II. VẼ HÌNH BẰNG KÍ TỰ ASCII 20

PHẦN I. MÔ PHỎNG Ổ ĐĨA RAID 5

1. Đề bài:

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

2. Phân tích cách làm:

- Với chuỗi ký tự input ta kiểm tra xem chuỗi kí tự nhập vào có độ dài là bội của 8 không. Nếu không thì yêu cầu nhập lại, nếu đúng thì bắt đầu chạy chương trình.
- Lấy lần lượt các 4 bytes in vào các Disk 1 và 2, in Disk 3 = Disk 1 XOR Disk 2 bằng hàm tính XOR. Tương tự với lần lượt với các dòng tiếp.
- Mã hoá kết quả từ dạng binary sang dạng hex theo bảng asciiz.
- Khi chạy xong thì đưa ra yêu cầu có muốn chạy lại không?

3. Thuật toán:

- Kiểm tra độ dài của chuỗi ký tự nhập vào:

```
#Check if the length of the input string is divisible by 8?
length: li      $t0, 0                          # t0 = length
        li      $t1, 0                          # t1 = index
check_char:
        add     $t2, $s0, $t1                   # t2 = address of string[i]
        lb      $t3, 0($t2)                     # t3 = string[i]
        nop
        beq     $t3, 10, test_len               # t3 = "\n" ends string
        nop
        addi    $t0, $t0, 1                     # length++
        addi    $t1, $t1, 1                     # index++
        j       check_char
        nop
test_len:
        beqz    $t0, er
        move    $t4, $t0
        and     $t2, $t0, 0xf                   # Get the last byte of $t0
        bne     $t2, 0, test1                   # If it equals 0 or 8, $t0 is divisble by 8
        j       START
test1:  beq     $t2, 8, START
        j       er
er:     li      $v0, 4                          # The length of the input string isn't appropriate
        la      $a0, er_len
        syscall
```

- Đo độ dài của chuỗi (dừng lại khi gặp "\n") bằng cách lấy byte cuối cùng của độ dài chuỗi nếu byte đó bằng 0 hoặc 8 thì khởi chạy nếu không thì báo lỗi và yêu cầu nhập lại.

- Mô phỏng RAID 5:

```
loop:
#Disk 1 XOR Disk 2 = Disk 3
        lw      $s1, 0($s0)                     # Load first 4 bytes from $s0
        lw      $s2, 4($s0)                     # Load next 4 bytes from $s0
        li      $k0, 10
        beq     $s1, $k0, end                   # If $s1 = "/n" then ends loop

        move    $a1, $s1                        # Print Disk 1
        jal     PRINT

        move    $a1, $s2                        # Print Disk 2
        jal     PRINT

        jal     PARITY                          # Print Disk 3

        li      $v0, 11                         # Print "\n"
        li      $a0, 10
        syscall
        addi    $s0, $s0, 8                     # Return $s0 8 bytes
```

- Load 4 bytes đầu tiên từ $s0 vào $s1, load 4 bytes đầu tiên từ $s0 vào $s2 nếu $s1 là "\n" thì dừng vòng lặp và kết thúc. Gán $a1 chứa địa chỉ của $s1, gọi hàm PRINT in ra 4 ký tự của $s1, gán $a1 tương tự với $s2, in 4 ký tự tiếp theo, rồi gọi hàm PARITY in ra kết quả XOR.
- Tương tự với các lần tiếp theo.

- Hàm PRINT:

```
PRINT:
        li      $v0, 11                         # print '|'
        li      $a0, '|'
        syscall
        li      $v0, 4                          # print "     "
        la      $a0, space
        syscall
        la      $a0, d
        sw      $a1, 0($a0)                     # print 4 characters
        syscall
        la      $a0, space                      # print "     "
        syscall
        li      $v0, 11                         # print '|'
        li      $a0, '|'
        syscall
        li      $v0, 4                          # print "     "
        la      $a0, space
        syscall
        jr      $ra
```

Nhớ 4 ký tự (4 bytes) từ $a1 vào $a0 rồi in 4 ký tự ra.

- Hàm PARITY:

```
PARITY:
        li      $v0, 4
        sw      $ra, 0($sp)
        la      $a0, opn                    # print "[[ "
        syscall

        xor     $s3, $s1, $s2
        and     $a0, $s3, 0xff              # Get 2 bytes from result of $s0 XOR $s1
        jal     HEX                         # Turn result into asciiz to print
        la      $a0, d                      # Print 2 characters in hex
        sw      $v0, 0($a0)
        li      $v0, 4
        syscall

        li      $v0, 11                     # print ","
        li      $a0, ','
        syscall
```

- Lưu tạm $ra vào $sp vì trong hàm này có hàm con HEX sử dụng chung thanh ghi $ra.
- $s3 = $s1 XOR $s2, lấy ra 2 bytes từ kết quả $s3 và dùng hàm HEX chuyển sang dạng 2 ký tự asciiz.

```
        srl     $s3, $s3, 8                 # Shift right 8 bits
        and     $a0, $s3, 0xff              # Get 2 bytes from result of $s0 XOR $s1
        jal     HEX                         # Turn result into asciiz to print
        la      $a0, d
        sw      $v0, 0($a0)
        li      $v0, 4
        syscall
        li      $a0, ','
        li      $v0, 11
        syscall
```

- Dịch phải 8 bits để in tiếp kết quả của phép XOR dưới dạng 2 ký tự asciiz. Tương tự với 2 lần còn lại.

- Hàm HEX:

```
HEX:
        and     $k1, $a0, 0xf                   # Get 1 byte first
        bgt     $k1, 9, char1                   # if $k1 < 9
        li      $v0, 0x30                       # $v0 = 0 in asciiz
        add     $v0, $v0, $k1                   # $v0 = 0 + $k1 by number form in ascii
        j       next
char1:
        li      $v0, 0x61                       # $v0 = "a" ascii
        subi    $k1, $k1, 0xa                   # minus 10
        add     $v0, $v0, $k1                   # print a, b, c, d in ascii

next:
        sll     $v0, $v0, 8                     # Shift right 8 bits
        srl     $k1, $a0, 4                     # Take next byte
        bgt     $k1, 0x9, char2
        addi    $v0, $v0, 0x30
        add     $v0, $v0, $k1
        j       endH
char2:
        addi    $v0, $v0, 0x61
        subi    $k1, $k1, 10
        add     $v0, $v0, $k1
endH:   jr      $ra
```

- Từ kết quả XOR của từng đôi một 2 bytes, lấy byte đầu tiên kiểm tra xem có <= 9 không thì in dưới dạng chữ số, nếu lớn hơn thì in dưới dạng chữ cái a, b, c, d, e, f. Dịch trái 8 bits để ghi ký tự tiếp, và lấy 4 bits (1 byte) còn lại, kiểm tra tương tự.

- Hàm REPEAT:

```
# Try again?
REPEAT: li $v0, 50
        la $a0, try
        syscall
        beq $a0, 0, clear
        nop
        j exit
clear:  la      $s0, string
        add     $t5, $s0, $t4                   # t5: address of the last byte used in the string
        li      $t2, 0
Again:
        sb      $t2, ($s0)                              # set byte at address $s0 = 0
        nop
        addi    $s0, $s0, 1
        bge     $s0, $t5, INPUT
        j       Again
#-----------------------------EXIT--------------------------------
```

- Nếu bạn muốn chạy lại thì xoá dữ liệu của string trước và set địa chỉ $s0 = 0, rồi quay lại INPUT.

4. Mã nguồn:

```
.data

string: .space     100
```

```
m0:  .asciiz  "Nhap chuoi ky tu : "
er_len: .asciiz "Do dai chuoi ki tu khong hop le. Hay thu lai!.\n"
m1:  .asciiz "     Disk 1               Disk 2               Disk 3\n"
m2:  .asciiz "----------------     ----------------     ---------------- \n"
opn:     .asciiz "[[ "
cls:     .asciiz "]]"
space:   .asciiz  "     "
try:     .asciiz "Do you want to try again?"
# "\n" = 10 in asciiz
.align 2
d:   .space   4

.text
INPUT:   li   $v0, 4                # Print "Nhap chuoi ky tu : "
     la   $a0, m0
     syscall
     li   $v0, 8
     la   $a0, string            # Read the input string
     li   $a1, 100
     syscall
     move     $s0, $a0           # s0 contains address of the input string
```

```
#Check if the length of the input string is divisible by
8?
length: li    $t0, 0                # t0 = length
     li   $t1, 0                 # t1 = index
check_char:
     add  $t2, $s0, $t1               # t2 = address of
string[i]
     lb   $t3, 0($t2)            # t3 = string[i]
     beq  $t3, 10, test_len      # t3 = "\n" ends string
     addi      $t0, $t0, 1            # length++
     addi      $t1, $t1, 1            # index++
     j    check_char
test_len:
     beqz $t0, er
     move      $t4, $t0
     and  $t2, $t0, 0xf          # Get the last byte of
$t0
     bne  $t2, 0, test1          # If it equals 0 or 8,
$t0 is divisble by 8
     j    START
test1:    beq  $t2, 8, START
     j    er
er:  li   $v0, 4                 # The length of the
input string isn't appropriate
     la   $a0, er_len
```

```
     syscall
     j    INPUT                  # Retype the input
string


#Stimulate RAID5
START:   li   $v0, 4
     la   $a0, m1
     syscall
     la   $a0, m2
     syscall
loop:


#Disk 1 XOR Disk 2 = Disk 3
     lw   $s1, 0($s0)            # Load first 4 bytes
from $s0
     lw   $s2, 4($s0)            # Load next 4 bytes from
$s0
     li   $k0, 10
     beq  $s1, $k0, end          # If $s1 = "/n" then
ends loop


     move $a1, $s1          # Print Disk 1
     jal  PRINT


     move $a1, $s2          # Print Disk 2
```

```
    jal  PRINT

    jal  PARITY                 # Print Disk 3

    li   $v0, 11                # Print "\n"
    li   $a0, 10
    syscall
    addi $s0, $s0, 8            # Return $s0 8 bytes

#Disk 1 XOR Disk 3 = Disk 2
    lw   $s1, 0($s0)            # Load first 4 bytes
from $s0
    lw   $s2, 4($s0)            # Load next 4 bytes from
$s0
    li   $k0, 10
    beq  $s1, $k0, end          # If $s1 = "/n" then
ends loop

    move $a1, $s1           # Print Disk 1
    jal  PRINT

    jal  PARITY                 # Print Disk 2

    move $a1, $s2           # Print Disk 3
    jal  PRINT
```

```
    li   $v0, 11                # Print "\n"
    li   $a0, 10
    syscall
    addi $s0, $s0, 8            # Return $s0 8 bytes

#Disk 2 XOR Disk 3 = Disk 1
    lw   $s1, 0($s0)            # Load first 4
characters from $s0
    lw   $s2, 4($s0)            # Load next 4 characters
from $s0
    li   $k0, 10
    beq  $s1, $k0, end          # If $s1 = "/n" then
ends loop

    jal  PARITY                 # Print Disk 1

    move $a1, $s1           # Print Disk 2
    jal  PRINT

    move $a1, $s2           # Print Disk 3
    jal  PRINT

    li   $v0, 11                # Print "\n"
    li   $a0, 10
```

```
    syscall
    addi $s0, $s0, 8            # Return $s0 8 bytes
    j    loop
end:
    li   $v0, 4
    la   $a0, m2
    syscall
    j    REPEAT
#------------------------------Functions-----------------
------------------
PRINT:
    li   $v0, 11                # print '|'
    li   $a0, '|'
    syscall
    li   $v0, 4                 # print "     "
    la   $a0, space
    syscall
    la   $a0, d
    sw   $a1, 0($a0)            # print 4 characters
    syscall
    la   $a0, space             # print "     "
    syscall
    li   $v0, 11                # print '|'
    li   $a0, '|'
    syscall
```

```
    li   $v0, 4                 # print "     "
    la   $a0, space
    syscall
    jr   $ra
# XOR processing
PARITY:
    li   $v0, 4
    sw   $ra, 0($sp)
    la   $a0, opn                   # print "[[ "
    syscall

    xor  $s3, $s1, $s2
    and  $a0, $s3, 0xff             # Get 2 bytes from
result of $s0 XOR $s1
    jal  HEX                        # Turn result into
asciiz to print
    la   $a0, d                     # Print 2 characters
in hex
    sw   $v0, 0($a0)
    li   $v0, 4
    syscall

    li   $v0, 11                # print ","
    li   $a0, ','
    syscall
```

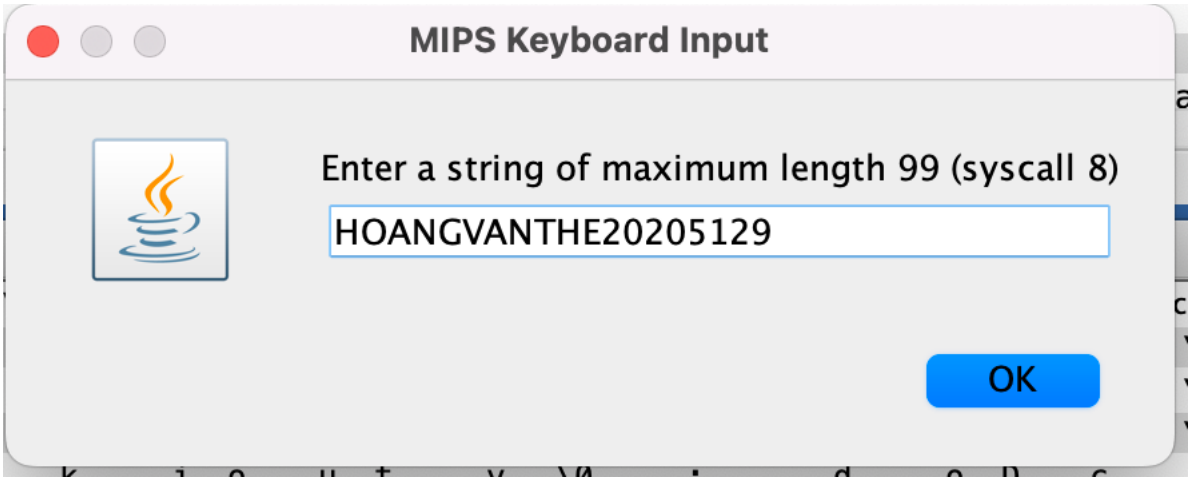
```
    srl  $s3, $s3, 8            # Shift right 8 bits
    and  $a0, $s3, 0xff             # Get 2 bytes from
result of $s0 XOR $s1
    jal  HEX                # Turn result into asciiz to
print
    la   $a0, d
    sw   $v0, 0($a0)
    li   $v0, 4
    syscall
    li   $a0, ','
    li   $v0, 11
    syscall
    srl  $s3, $s3, 8            # Shift right 8 bits
    and  $a0, $s3, 0xff             # Get 2 bytes from
result of $s0 XOR $s1
    jal  HEX                # Turn result into asciiz to
print
    la   $a0, d
    sw   $v0, 0($a0)
    li   $v0, 4
    syscall
    li   $a0, ','
    li   $v0, 11
    syscall
    srl  $s3, $s3, 8            # Shift right 8 bits
```

```
    and  $a0, $s3, 0xff              # Get 2 bytes from
result of $s0 XOR $s1
    jal  HEX               # Turn result into asciiz to
print
    la   $a0, d
    sw   $v0, 0($a0)
    li   $v0, 4
    syscall
    la   $a0, cls           # print "]]"
    syscall
    la   $a0, space             # print "       "
    syscall
    lw   $ra, 0($sp)
    jr   $ra

# Turn binary XOR result into hex by asciiz
HEX:
    and  $k1, $a0, 0xf          # Get 1 byte first
    bgt  $k1, 9, char1          # if $k1 < 9
    li   $v0, 0x30                   # $v0 = 0 in asciiz
    add  $v0, $v0, $k1               # $v0 = 0 + $k1 by
number form in ascii
    j    next
char1:
    li   $v0, 0x61                   # $v0 = "a" ascii
```

```
     subi $k1, $k1, 0xa              # minus 10
     add  $v0, $v0, $k1          # print a, b, c, d,...
by ascii


next:
     sll  $v0, $v0, 8                # Shift left 8 bits
     srl  $k1, $a0, 4            # Take next byte
     bgt  $k1, 0x9, char2
     addi $v0, $v0, 0x30
     add  $v0, $v0, $k1
     j    endH
char2:
     addi $v0, $v0, 0x61
     subi $k1, $k1, 10
     add  $v0, $v0, $k1
endH:    jr   $ra


# Try again?
REPEAT:  li $v0, 50
     la $a0, try
     syscall
     beq $a0, 0, clr
     nop
     j exit
clr: la   $s0, string
```

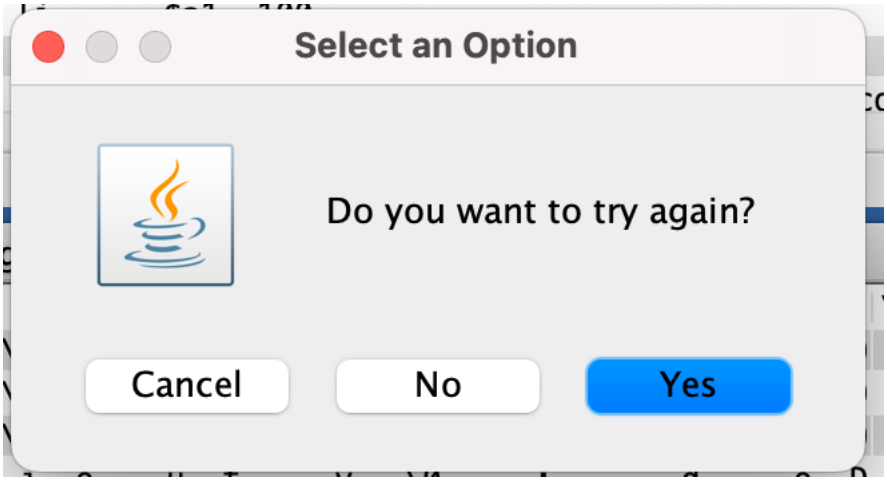
```
    add  $t5, $s0, $t4           # t5: address of the
last byte used in the string

    li   $t2, 0

Again:

    sb   $t2, ($s0)                  # set byte at
address $s0 = 0

    nop

    addi     $s0, $s0, 1

    bge  $s0, $t5, INPUT

    j    Again

#------------------------------EXIT----------------------
-----------------

exit:    li $v0, 10

    syscall
```

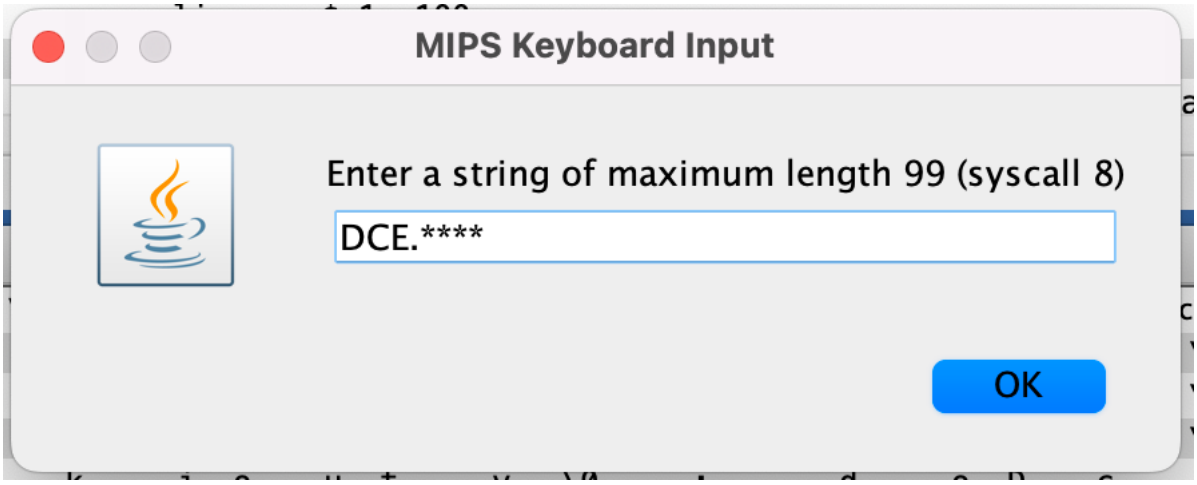
5. Kết quả chạy thử:



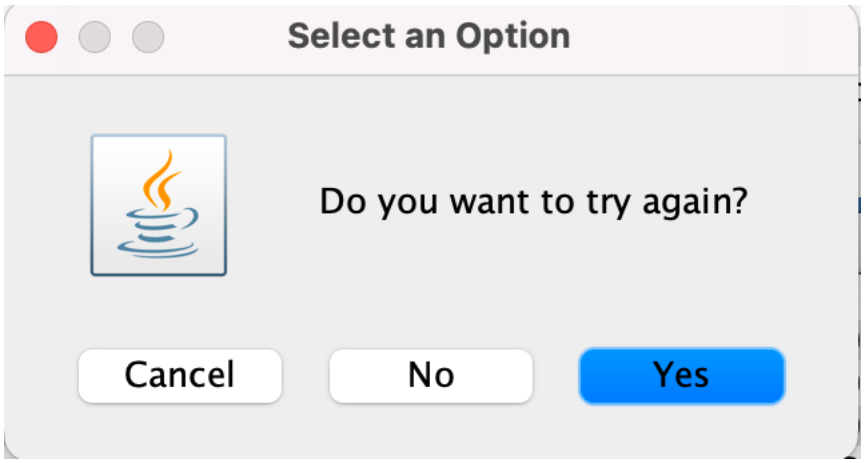

```
Nhap chuoi ky tu : **** user input : HOANGVANTHE20205129
Do dai chuoi ki tu khong hop le. Hay thu lai!.
Nhap chuoi ky tu :
```

```
Nhap chuoi ky tu : **** user input : HOANGVANTHE20205
     Disk 1              Disk 2              Disk 3
-----------------   -----------------   -----------------
|      HOAN     |   |      GVAN     |   [[ 0f,19,00,00]]
|      THE2     |   [[ 64,7a,75,07]]  |      0205     |
-----------------   -----------------   -----------------
```





```
|      THE2      |      [[ 64,7a,75,07]]      |      0205      |
-----------------  -----------------  -----------------
Nhap chuoi ky tu : **** user input : DCE.****
     Disk 1              Disk 2              Disk 3
-----------------  -----------------  -----------------
|      DCE.      |      |  ****  |      [[ 6e,69,6f,04]]
-----------------  -----------------  -----------------
```




```
     Disk 1              Disk 2              Disk 3
-----------------  -----------------  -----------------
|      DCE.      |      |  ****  |      [[ 6e,69,6f,04]]
-----------------  -----------------  -----------------

-- program is finished running --
```

PHẦN II. VẼ HÌNH BẰNG KÍ TỰ ASCII

1. Đề bài:

Cho hình ảnh đã được chuyển thành các kí tự ASCIInhư hình vẽ. Đây là hình của chữ DCE có viền * và màu là các con số.

- Hãy hiển thị hình ảnh trên lên giao diện console (hoặc giao diện Display trong công cụ giả lập Keyboard and Display MMIO Simulator).
- Hãy sửa ảnh để các chữ cái DCE chỉ còn lại viền, không còn màu số ở giữa, và hiển thị.
- Hãy sửa ảnh để hoán đổi vị trí của các chữ, thành ECD, và hiển thị. Để đơn giản, các hoạ tiết đính kèm cũng được phép di chuyển theo.
- Hãy nhập từ bàn phím kí tự màu cho chữ D, C, E, rồi hiển thị ảnh trên với màu mới.

Chú ý: ngoài vùng nhớ lớn chứa ảnh được chứa sẵn trong code, không được tạo thêm vùng nhớmới đểchứa ảnh hiệu chỉnh.

2. Phân tích cách làm, thuật toán:

- Bài toán có 5 chức năng nên em đưa vào Switch-Case để thực hiện các chức năng sau đó quay lại menu ban đầu để thực hiện chức năng khác.

- Ở bài này, phần khó nhất chính là đọc dữ liệu từ xâu data để in ra lần lượt các byte. Bài toán yêu cầu in 16 dòng tất cả, mỗi dòng có 57 byte. Độ dài của chữ D C E lần lượt là 22 byte, 20 byte, 15 byte. Dựa vào đây ta có thể in ra từng chữ cái với từng vị trí theo ý muốn.

- Chức năng 1: Hiện thị hình ảnh DCE lên giao diện console

+ Như đã nói ở trên, chúng ta sẽ in lần lượt các chữ D C E. Sau đây là phần phân tích thuật toán in ra chữ D

```
printD:
        mul $t1,$t0,$s1 # t1 = 57 * (0,1,...15)
        add $t2,$t1,0 # t2 = t1: độ dài min của chữ D
        addi $t3,$t1,22 # độ dài max của chữ D theo từng dòng (chữ D dài 22 byte)
        loopPrintD:
                beq $t2,$t3,endLoopPrintD # t2 = t3 thì dừng
                add $t4,$s0,$t2 # t4 = s0 + t2 : lấy từng byte trong xâu data
                lb $a0,0($t4) # giá trị của địa chỉ t4
                beq $a0,'2',printAsS2
                continueD:
                li,$v0,11
                syscall # in ra màn hình
                add $t2,$t2,1 # tăng t2
                j loopPrintD
        endLoopPrintD:
                jr $ra
        printAsS2:
                addi $a0,$s2,0
                j continueD
```

+ Ở mỗi dòng sẽ lấy 22 byte đầu tiên, vì mỗi dòng là 57 byte nên để xuống dòng ta sẽ lấy địa chỉ của xâu cộng với 57. Vòng lặp loopPrintD sẽ dừng khi đã đạt đủ độ dài của chữ D tức $t2=$t3.

+ Tương tự với C và E

- Chức năng 2: In viền của DCE

```
F2:
        # thay màu bằng kí tự space
        li $s2,' '
        li $s3,' '
        li $s4,' '
        jal print
        j main
```

+ Gán giá trị màu của các chữ bằng kí tự ‘ ’

```
            beq $a0,'2',printAsS2
            continueD:
            li,$v0,11
            syscall # in ra màn hình
            add $t2,$t2,1 # tăng t2
            j loopPrintD
    endLoopPrintD:
            jr $ra
    printAsS2:
            addi $a0,$s2,0
            j continueD
```

+ Nếu kí tự ở a0 là kí tự màu thì nhảy đến nhãn printAsS2 để đổi màu thành kí tự ‘ ’

- Chức năng 3: Thay đổi vị trí D C E

+ Tương tự như chức năng 1, chỉ cần thay đổi thứ tự in thành E C D

```
childF3:

        jal printE # in E
        jal printC # in C
        jal printD # in D
        j continue # quay lại hàm print để in dòng mới
```

- Chức năng 4: Thay đổi màu D C E

+ Tương tự như chức năng 2, ta nhập màu muốn thay đổi vào thanh ghi $s2,$s3,$s4

```
# Them mau cho D

li $v0, 4
la $a0, ChuD
syscall
li $v0, 12
syscall
add $s2,$v0,0

# Them mau cho C
li $v0, 4
la $a0, ChuC
syscall
li $v0, 12
syscall
add $s3,$v0,0


# Them mau cho E
li $v0, 4
la $a0, ChuE
syscall
li $v0, 12
syscall
add $s4,$v0,0
```

3. Mã nguồn:

```
.data
    Menu: .asciiz "------------IN CHU-----------\n"
    Func1:.asciiz "1. In DCE\n"
    Func2:.asciiz "2. In DCE(giu lai vien khong in mau)\n"
    Func3:.asciiz "3. Thay doi vi tri D, C, E\n"
    Func4:.asciiz "4. Doi mau\n"
    Func5:.asciiz "5. Thoat\n"
    Thank:.asciiz "------------THANK YOU-----------\n"
    Nhap: .asciiz "Nhap chuc nang: "
```

```
    ChuD: .asciiz "Nhap mau cho chu D(0->9): "
    ChuC: .asciiz "Nhap mau cho chu C(0->9): "
    ChuE: .asciiz "Nhap mau cho chu E(0->9): "
    Warning: .asciiz "Vui long chon 1 trong 5 chuc
nang\n"
    data: .asciiz "
************* **************
*3333333333333**222222222222*
*33333******** *22222******222222*
*33333*        *22222*      *22222*
*33333******** *22222*       *22222*      *************
*3333333333333**22222*       *22222*    **11111*****111*
*33333******** *22222*       *22222*  **1111**       **
*33333*        *22222*      *222222*  *1111*
*33333******** *22222*******222222*  *11111*
*3333333333333**2222222222222222*    *11111*
************* ***************       *11111*
---               *1111**                              /
o o \\              *1111****   *****
\\   > /              **111111***111*
-----                ***********    dce.hust.edu.vn"
.text
main:
    la $s0,data # địa chỉ của xâu data
    li $s1, 57 # chiều dài 1 dòng
    li $s2,'2' # màu khởi tạo của chữ D
    li $s3,'1' # màu khởi tạo của chữ C
    li $s4,'3' # màu khởi tạo của chữ E
```

```
li $s5,0 # s5 = 0 thì in DCE s5 = 1 thì in ECD

la $a0, Menu
li $v0, 4
syscall

la $a0, Func1 # In chức năng 1
li $v0, 4
syscall

la $a0, Func2      # In chức năng 2
li $v0,4
syscall

la $a0,Func3  # In chức năng 3
li $v0,4
syscall

la $a0,Func4  # In chức năng 4
li $v0,4
syscall

la $a0,Func5  # Thoát
li $v0,4
```

```
syscall


la $a0,Nhap   # Nhập
li $v0,4
syscall


li $v0,5
syscall


Case1:
     li $v1,1 # v1 = 1
     bne $v0,$v1,Case2  # v0 != v1 thì nhảy đến Case2
     j F1 # v0 = v1 thì thực hiện Func1
Case2:
     li $v1,2 # v1 = 2
     bne $v0,$v1,Case3 # v0 != 2 thì nhảy đến Case3
     j F2 # v0 = 2 thì thực hiện Func2
Case3:
     li $v1,3 # v1 = 3
     bne $v0,$v1,Case4 # v0 != 3 thì nhảy đến Case4
     j F3 # v0 = 3 thì thực hiện Func3
Case4:
     li $v1,4 # v1 = 4
     bne $v0,$v1,Case5 # v0 != 4 thì nhảy đến Case5
```

```
          j F4 # v0 = 4 thì thực hiện Func4
     Case5:
          li $v1,5 # v1 = 5
          bne $v0,$v1,default # v0 != 5 thì nhảy đến
default
          j F5 # v0 = 5 thì thực hiện Func5
     default:
          li $v0,4
          la $a0,Warning
          syscall
          j main
#########Func1#############
F1:
     jal print
     j main
#########Func2#############
F2:
     # thay màu bằng kí tự space
     li $s2,' '
     li $s3,' '
     li $s4,' '
     jal print
     j main
#########Func3#############
```

```
F3:
     li $s5,1 # gán $ s5 = 1 để in ECD
     jal print
     j main
#########Func4#############
F4:
     # Them mau cho D

     li $v0, 4
     la $a0, ChuD
     syscall
     li $v0, 12
     syscall
     add $s2,$v0,0

     # Them mau cho C
     li $v0, 4
     la $a0, ChuC
     syscall
     li $v0, 12
     syscall
     add $s3,$v0,0

     # Them mau cho E
```

```
	li $v0, 4
	la $a0, ChuE
	syscall
	li $v0, 12
	syscall
	add $s4,$v0,0

	jal print
	j main
#########Func5#############
F5:
	la $a0,Thank  # In chức năng 4
	li $v0,4
	syscall
	li $v0,10
	syscall
print:
	li $t0,0 # gán t0 = 0
	addi $sp, $sp, -4
	sw $ra, 0($sp)
	loopPrint:
	# cho t0 chạy từ 0->15 để in 16 dòng
		beq $t0,16,endLoopPrint # t0 = 16 thì dừng loop
```

```
          beq $s5,1,childF3 # nếu s5 = 1 thì thực hiện
Func3 (in ECD)
          jal printD # in D
          jal printC # in C
          jal printE # in E
          continue:
          li $v0,11 # xuống dòng mới
          la $a0,'\n'
          syscall
          addi $t0,$t0,1 # t0 = t0 + 1
          j loopPrint
     endLoopPrint:
          lw $ra,0($sp)
          addi $sp, $sp, 4
          jr $ra
childF3:

     jal printE # in E
     jal printC # in C
     jal printD # in D
     j continue # quay lại hàm print để in dòng mới

printD:
     mul $t1,$t0,$s1 # t1 = 57 * (0,1,...15)
```

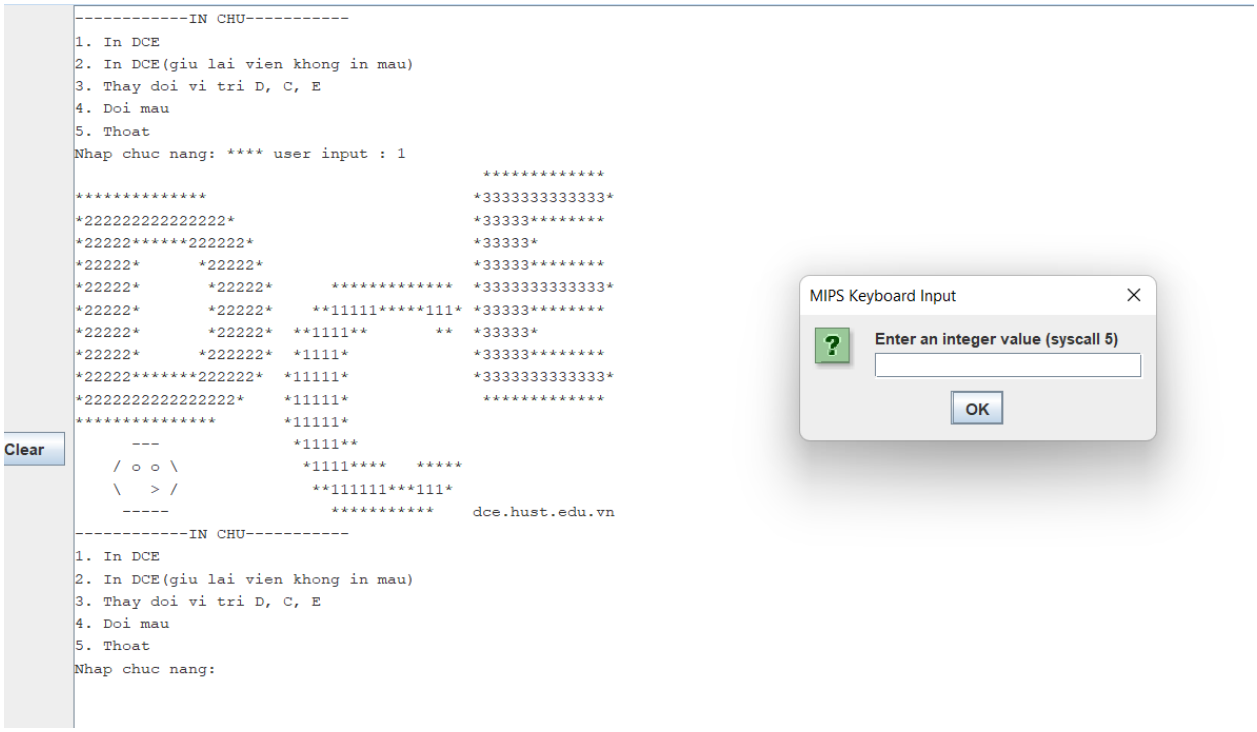
```
    add $t2,$t1,0 # t2 = t1: độ dài min của chữ D
    addi $t3,$t1,22 # độ dài max của chữ D theo từng dòng (chữ D dài 22 byte)
    loopPrintD:
        beq $t2,$t3,endLoopPrintD # t2 = t3 thì dừng
        add $t4,$s0,$t2 # t4 = s0 + t2 : lấy từng byte trong xâu data
        lb $a0,0($t4) # giá trị của địa chỉ t4
        beq $a0,'2',printAsS2
        continueD:
        li,$v0,11
        syscall # in ra màn hình
        add $t2,$t2,1 # tăng t2
        j loopPrintD
    endLoopPrintD:
        jr $ra
    printAsS2:
        addi $a0,$s2,0
        j continueD
printC:
    mul $t1,$t0,$s1 # t1 = 57 * (0,1,...15)
    add $t2,$t1,22 # t2 = t1 + 22: độ dài min của chữ C
    addi $t3,$t1,42 # độ dài max của chữ D theo từng dòng (chữ D dài 20 byte)
    loopPrintC:
```

```
        beq $t2,$t3,endLoopPrintC # t2 = t3 thì dừng
        add $t4,$s0,$t2 # t4 = s0 + t2 : lấy từng byte
trong xâu data
        lb $a0,0($t4) # giá trị của địa chỉ t4
        beq $a0,'1',printAsS3
        continueC:
        li,$v0,11
        syscall # in ra màn hình
        add $t2,$t2,1 # tăng t2
        j loopPrintC
    endLoopPrintC:
        jr $ra
    printAsS3:
        addi $a0,$s3,0
        j continueC
printE:
    mul $t1,$t0,$s1 # t1 = 57 * (0,1,...15)
    add $t2,$t1,42 # t2 = t1 + 42 : độ dài min của chữ E
    addi $t3,$t1,57 # độ dài max của chữ E theo từng
dòng (chữ D dài 15 byte)
    loopPrintE:
        beq $t2,$t3,endLoopPrintE # t2 = t3 thì dừng
        add $t4,$s0,$t2 # t4 = s0 + t2 : lấy từng byte
trong xâu data
        lb $a0,0($t4) # giá trị của địa chỉ t4
```

```
        beq $a0,'3',printAsS4
        continueE:
        li,$v0,11
        syscall # in ra màn hình
        add $t2,$t2,1 # tăng t2
        j loopPrintE
    endLoopPrintE:
        jr $ra
    printAsS4:
        addi $a0,$s4,0
        j continueE
```

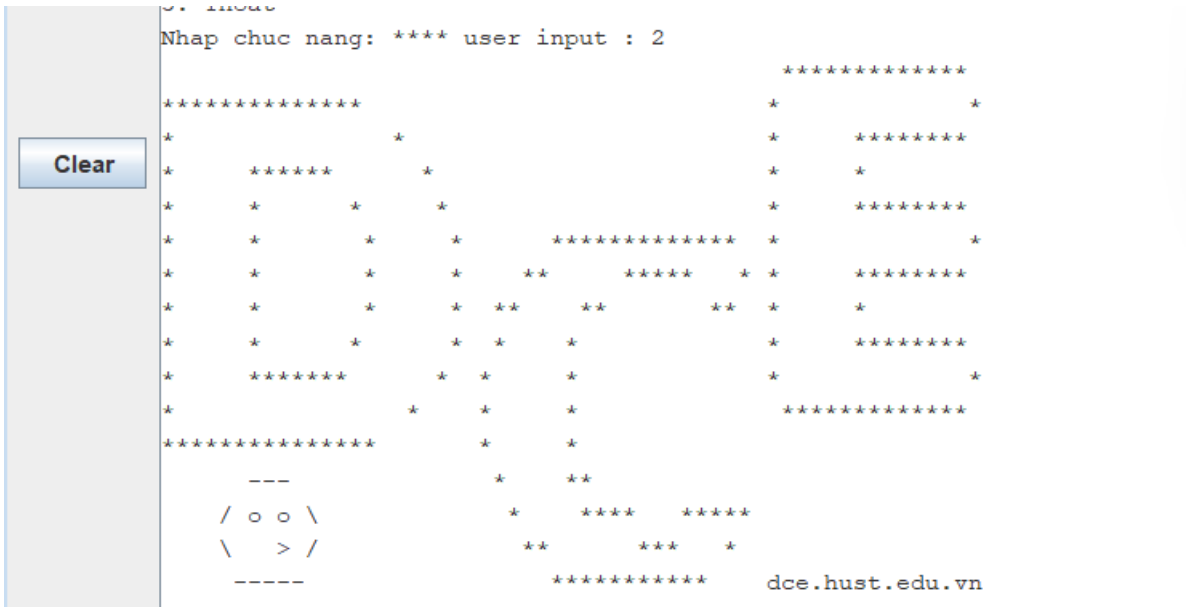
4. Kết quả chạy thử:

- Chức năng 1:

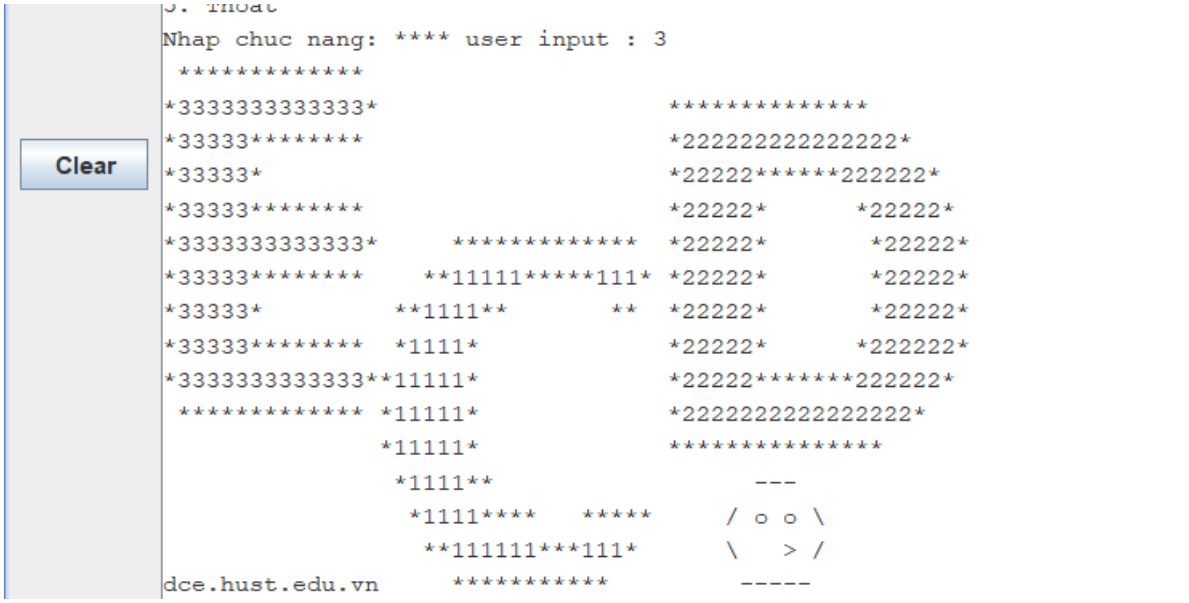



```
------------IN CHU-----------
1. In DCE
2. In DCE(giu lai vien khong in mau)
3. Thay doi vi tri D, C, E
4. Doi mau
5. Thoat
Nhap chuc nang: **** user input : 1
                                          *************
**************                           *3333333333333*
*222222222222222*                        *33333********
*22222******222222*                      *33333*
*22222*      *22222*                     *33333********
*22222*       *22222*      *************  *3333333333333*
*22222*       *22222*    **11111****111*  *33333********
*22222*       *22222*  **1111**       **  *33333*
*22222*      *222222*  *1111*             *33333********
*22222*******222222*  *11111*             *3333333333333*
*2222222222222222*    *11111*              *************
***************       *11111*
      ---              *1111**
    / o o \             *1111****   *****
    \   > /              **111111***111*
     -----                 ***********    dce.hust.edu.vn
------------IN CHU-----------
1. In DCE
2. In DCE(giu lai vien khong in mau)
3. Thay doi vi tri D, C, E
4. Doi mau
5. Thoat
Nhap chuc nang:
```

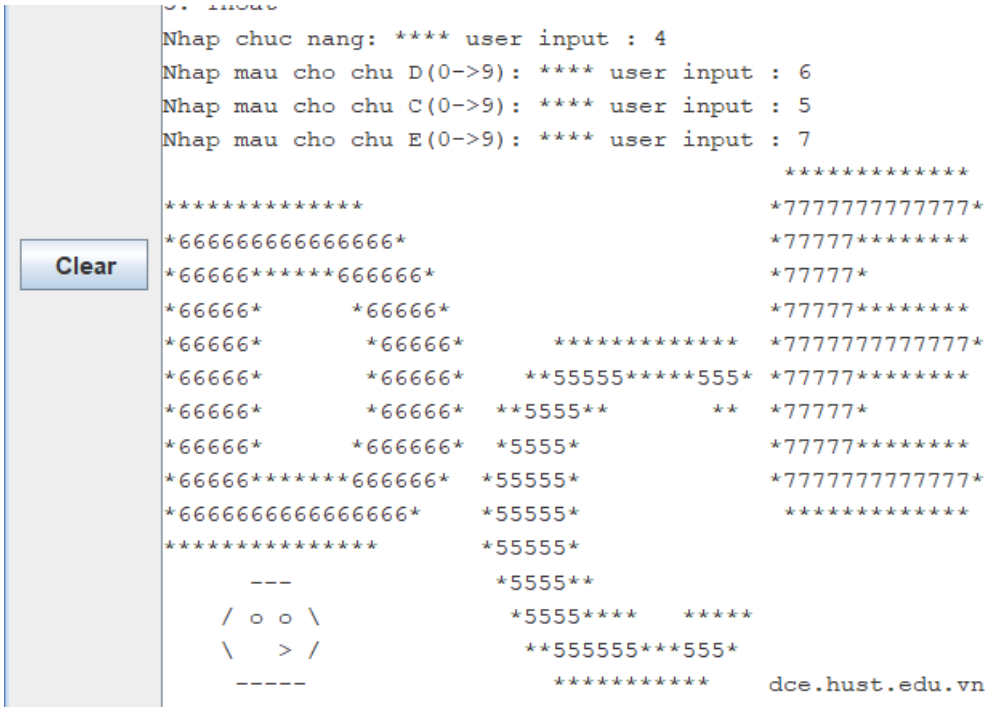
- Chức năng 2:


Clear
Nhap chuc nang: **** user input : 2
dce.hust.edu.vn

- Chức năng 3:


Clear
Nhap chuc nang: **** user input : 3
dce.hust.edu.vn

- Chức năng 4:



Trân trọng cảm ơn thầy và các bạn!