TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

———————— * ————————

BÁO CÁO

THỰC HÀNH KIẾN TRÚC MÁY TÍNH

Giảng viên hướng dẫn: **Lê Bá Vui**

Lớp: 113834 – K62

Sinh viên thực hiện:

Lương Đức Minh – 20176821

Nguyễn Thanh Hà – 20176742

Hà Nội, tháng 6 năm 2020

Mục lục

Table of Contents

Thông tin đề tài

Đề số 6: Hàm cấp phát bộ nhớ malloc()

Chương trình cho bên dưới là hàm malloc(), kèm theo đó là ví dụ minh họa, được viết bằng hợp ngữ MIPS, để cấp phát bộ nhớ cho một biến con trỏ nào đó. Hãy đọc chương trình và hiểu rõ nguyên tắc cấp phát bộ nhớ động.
Trên cơ sở đó, hãy hoàn thiện chương trình như sau. Lưu ý, ngoài viết các hàm đó, cần viết thêm một số ví dụ minh họa để thấy việc sử dụng hàm đó như thế nào.

1. Việc cấp phát bộ nhớ kiểu word/mảng word có 1 lỗi, đó là chưa bảo đảm qui tắc địa chỉ của kiểu word phải chia hết cho 4. Hãy khắc phục lỗi này.
2. Viết hàm lấy giá trị Word /Byte của biến con trỏ (tương tự như *CharPtr, *BytePtr, *WordPtr)
3. Viết hàm lấy địa chỉ biến con trỏ (tương tự như &CharPtr, &BytePtr, *WordPtr)
4. Viết hàm thực hiện copy 2 con trỏ xâu kí tự (Xem ví dụ về CharPtr)
5. Viết hàm tính toàn bộ lượng bộ nhớ đã cấp phát cho các biến động
6. Hãy viết hàm Malloc2 để cấp phát cho mảng 2 chiều kiểu .word với tham số vào gồm:
    1. Địa chỉ đầu của mảng
    2. Số dòng
    3. Số cột
7. Tiếp theo câu 6, hãy viết 2 hàm GetArray[i][j] và SetArray[i][j] để lấy/thiết lập giá trị cho phần tử ở dòng I cột j của mảng.

Đề số 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     |
 --------------        --------------        --------------
```

Đề số 6

Phương hướng giải quyết

Bước 1: Khởi tạo menu 7 options tương ứng với 7 yêu cầu của đề bài.

Bước 2: Xử lý từng yêu cầu một như sau

- **Sửa lỗi cấp phát kiểu .word của chương trình ví dụ:** Khi người dùng lựa chọn cấp phát con trỏ kiểu word (WordPtr) thì sẽ kiểm tra xem địa chỉ còn trống đầu tiên có chia hết cho 4 không?
  - Nếu chia hết thì thực hiện cấp phát như bình thường.
  - Nếu dư thì có 3 trường hợp chia 4 dư {3, 2, 1}, cần cộng thêm vào địa chỉ này (4 – số dư) bytes.
    VD: Địa chỉ bắt đầu cấp phát là 0x90000007 chia 4 dư 3. Cần cộng thêm (4-3) = 1 byte. Địa chỉ mới sẽ là 0x90000008. Thực hiện cấp phát tiếp như bình thường.
- **Viết hàm lấy giá trị Word / Byte của biến con trỏ:** Truy cập địa chỉ của các biến con trỏ bằng lệnh la và thực hiện lệnh lw để lấy ra giá trị tương ứng. Gọi Syscall 34 để in giá trị đó dưới dạng mã Hex.
- **Viết hàm lấy địa chỉ của biến con trỏ:** Chỉ cần truy cập địa chỉ các biến con trỏ bằng lệnh la và in ra màn hình với Syscall 34.
- **Viết hàm thực hiện copy 2 con trỏ xâu kí tự:**
  - Cho phép người dùng nhập xâu kí tự từ bàn phím (giới hạn 100 kí tự), lưu vào vùng nhớ trỏ bởi con trỏ CharPtr1.
  - Cho con trỏ CharPtr2 trỏ đến địa chỉ đầu tiên còn trống để thực hiện cấp phát.
  - Lưu lần lượt từng kí tự trong xâu kí tự được trỏ bởi CharPtr1 sang vùng nhớ được trỏ bởi CharPtr2. Số kí tự đếm được (độ dài xâu) chính là lượng bộ nhớ cần cấp phát cho con trỏ CharPtr2.
- **Viết hàm tính toàn bộ lượng bộ nhớ đã cấp phát cho các biến động:**
  - Tổng lượng bộ nhớ đã sử dụng cho việc cấp phát = "Used_Total"
    = (Địa chỉ đầu tiên còn trống – địa chỉ vùng nhớ dùng để cấp phát)
    = Sys_TheTopOfFree – Sys_MyFreeSpace
  - Tuy nhiên trong quá trình cấp phát kiểu .word có thực hiện làm tròn để đảm bảo địa chỉ là bội của 4, nên tổng lượng bộ nhớ thực tế cấp phát cho các biến có thể nhỏ hơn hoặc bằng "Used_Total"
  - Vì vậy, em tạo thêm 1 biến toàn cục $s2 để lưu bộ nhớ đã cấp phát, mỗi lần cấp phát chính xác bao nhiêu thì cập nhật lại giá trị thanh ghi này.
- **Viết hàm Malloc2 để cấp phát cho mảng 2 chiều kiểu .word:**
  - Thực chất có thể coi mảng 2 chiều kích thước m*n là mảng 1 chiều với (m*n) phần tử
  - Bên trong hàm malloc2 chỉ cần gọi đến hàm malloc với tham số đầu vào là (m*n) phần tử, mỗi phần tử 4 bytes (1 word)
- **Viết 2 hàm Get / Set cho mảng 2 chiều trên**
  - Phần tử A[i][j] có thể được truy cập theo công thức: index = i * ncols + j
  - nrows, ncols là số hàng, cột của mảng. Nếu i, j vượt quá giá trị này sẽ báo lỗi. Sau khi Set xong giá trị thì sẽ in ra toàn bộ mảng để dễ kiểm tra.

Source code (mã nguồn)

```
.data
#----------Pointer----------
CharPtr: .word  0     # Bien con tro, tro toi kieu asciiz
BytePtr: .word  0     # Bien con tro, tro toi kieu Byte
WordPtr: .word  0     # Bien con tro, tro toi mang kieu Word
CharPtr1: .word  0
CharPtr2: .word  0
ArrayPtr: .word  0    # Bien con tro 2D array, tro toi mang kieu Word
#----------Menu String----------
option_menu: .asciiz "1. Cap phat bo nho\n2. Lay gia tri word/byte cua bien con
tro\n3. Lay dia chi bien con tro\n4. Copy 2 xau con tro ki tu\n5. Tinh toan
luong bo nho da cap phat co cac bien dong\n6. Ham malloc2 cap phat mang 2
chieu\n7. GetArray[i][j] và SetArray[i][j]\n\nELSE: Exit"
malloc_menu: .asciiz "1. CharPtr\n2. BytePtr\n3. WordPtr\n4. Return main
menu\n\nELSE: Exit"
getset_menu: .asciiz "1. GetArray[i][j]\n2. SetArray[i][j]\n3. Return main
menu\n\nELSE: Exit"
#----------Dump Messages----------
malloc_success: .asciiz "\nMalloc succesfully. "
charPtr_Add: .asciiz "\nCharPtr address: "
bytePtr_Add: .asciiz "\nBytePtr address: "
wordPtr_Add: .asciiz "\nWordPtr address: "
arrayPtr_Add: .asciiz "\nArrayPtr address: "
charPtr_Val: .asciiz "\nCharPtr value: "
bytePtr_Val: .asciiz "\nBytePtr value: "
wordPtr_Val: .asciiz "\nWordPtr value: "
arrow: .asciiz " --> "
tab: .asciiz "\t"
new_line: .asciiz "\n"
dash_line: .asciiz "\n---------------------------------------\n"
amount:.asciiz "\nNhap so phan tu can cap phat: "
CharPtr1_Val: .asciiz "\nCharPtr1: "
oldCharPtr2_Val: .asciiz "CharPtr2 (before): "
newCharPtr2_Val: .asciiz "\nCharPtr2 (after): "
input_message: .asciiz "Nhap 1 string bat ky de thuc hien copy: "
used_total: .asciiz "\nTong luong bo nho da su dung (tinh ca lam tron de sua
loi): "
allocated_total: .asciiz "\nTong luong bo nho da cap phat: "
byte: .asciiz " byte(s)"
unallocated: .asciiz "\nMang 2 chieu chua duoc cap phat!!!"
input_row: .asciiz "\nNhap so hang: "
input_col: .asciiz "\nNhap so cot: "
input_i: .asciiz "\nNhap vi tri hang: "
input_j: .asciiz "\nNhap vi tri cot: "
```

```
assign_val: .asciiz "\nNhap gia tri can gan: "
out_of_bound: .asciiz "\nIndex out of bound."
string_copy: .space  100

.kdata
# Bien chua dia chi dau tien cua vung nho con trong
Sys_TheTopOfFree: .word  1
# Vung khong gian tu do, dung de cap bo nho cho cac bien con tro
Sys_MyFreeSpace:

.text
       jal    SysInitMem            # Khoi tao vung nho cap phat dong

       #------>[ Cac thanh ghi luu tru bien toan cuc (global variable) ]<------
       li     $s2, 0                # Tong so byte(s) da cap phat
       li     $s3, 0                # So hang cua array (nrows)
       li     $s4, 0                # So cot cua array (ncols)

menu:  la     $a0, dash_line        # start new action
       li     $v0, 4
       syscall
       la     $a0, option_menu      # In Menu String
       li     $v0, 51
       syscall
       move   $s0, $a0              # 7 options ung voi 7 yeu cau trong de bai
       beq    $s0, 1, extra_menu1
       beq    $s0, 2, case2
       beq    $s0, 3, case3
       beq    $s0, 4, case4
       beq    $s0, 5, case5
       beq    $s0, 6, case6
       beq    $s0, 7, extra_menu7
       j      end

extra_menu1:
       la     $a0, malloc_menu      # 3 lua chon tuong ung voi yeu cau 1
       li     $v0, 51
       syscall
       move   $s0, $a0              # switch case
       beq    $s0, 1, case1.1       # Malloc CharPtr
       beq    $s0, 2, case1.2       # Malloc BytePtr
       beq    $s0, 3, case1.3       # Malloc WordPtr
       beq    $s0, 4, menu          # return menu
       j      end

extra_menu7:
```

```
        la      $a0, getset_menu      # 2 lua chon get/set tuong ung yeu cau 7
        li      $v0, 51
        syscall
        move    $s0, $a0              # switch case
        beq     $s0, 1, case7.1       # getArray[i][j]
        beq     $s0, 2, case7.2       # setArray[i][j]
        beq     $s0, 3, menu          # return menu
        j       end

end:    li      $v0, 10
        syscall

#-----------------------
#  Cap phat cho bien con tro CharPtr, gom 3 phan tu, moi phan tu 1 byte
#-----------------------
case1.1:
        la      $a0, amount           # In "Nhap so phan tu can cap phat: "
        li      $v0, 51
        syscall
        move    $a1, $a0
        la      $a0, CharPtr
        la      $a3, charPtr_Add
        addi    $a2, $zero, 1
        jal     malloc
        j       menu
#-----------------------
#  Cap phat cho bien con tro BytePtr, gom 6 phan tu, moi phan tu 1 byte
#-----------------------
case1.2:
        la      $a0, amount           # In ra thong bao "Nhap so phan tu can cap
phat: "
        li      $v0, 51
        syscall
        move    $a1, $a0
        la      $a0, BytePtr
        la      $a3, bytePtr_Add
        addi    $a2, $zero, 1
        jal     malloc
        j       menu
#-----------------------
#  Cap phat cho bien con tro WordPtr, gom 5 phan tu, moi phan tu 4 byte
#-----------------------
case1.3:
        la      $a0, amount           # In ra thong bao "Nhap so phan tu can cap
phat: "
        li      $v0, 51
```

```
        syscall
        move    $a1, $a0
        la      $a0, WordPtr
        la      $a3, wordPtr_Add
        addi    $a2, $zero, 4
        jal     malloc
        j       menu
#----------------------------------------
#  In ra gia tri cac bien con tro
#----------------------------------------
case2:
        #----------->[ charPtr ]<-----------
        la      $a0, charPtr_Val
        li      $v0, 4
        syscall
        la      $a0, CharPtr
        jal     ptr_val
        #----------->[ bytePtr ]<-----------
        la      $a0, bytePtr_Val
        li      $v0, 4
        syscall
        la      $a0, BytePtr
        jal     ptr_val
        #----------->[ wordPtr ]<-----------
        la      $a0, wordPtr_Val
        li      $v0, 4
        syscall
        la      $a0, WordPtr
        jal     ptr_val
        j       menu
#----------------------------------------
#  In ra dia chi cac bien con tro
#----------------------------------------
case3:
        #----------->[ charPtr ]<-----------
        la      $a0, charPtr_Add
        li      $v0, 4
        syscall
        la      $a0, CharPtr
        jal     ptr_add
        #----------->[ bytePtr ]<-----------
        la      $a0, bytePtr_Add
        li      $v0, 4
        syscall
        la      $a0, BytePtr
        jal     ptr_add
```

```
        #----------->[ wordPtr ]<-----------
        la      $a0, wordPtr_Add
        li      $v0, 4
        syscall
        la      $a0, WordPtr
        jal     ptr_add
        j       menu
#-----------------------------------------
#  Ham thuc hien copy 2 xau ki tu
#-----------------------------------------
case4:
        # --------------->[ Cac lenh syscall de in ra man hinh ]<---------------
        li      $v0, 54               # System call for InputDialogString
        la      $a0, input_message    # In ra thong bao "Input a string: "
        la      $a1, string_copy      # Dia chi luu string dung de copy
        li      $a2, 100              # So ki tu toi da co the doc duoc = 100
        syscall
        la      $a1, string_copy      # Load lai 1 lan
        la      $s1, CharPtr1         # Load dia chi cua CharPtr1
        sw      $a1, 0($s1)           # Luu string vua nhap vao CharPtr1
        la      $a0, CharPtr1_Val     # In ra thong bao "CharPtr1: "
        li      $v0, 4
        syscall
        la      $a0, CharPtr1
        lw      $a0, 0($a0)           # Lay gia tri luu trong word nho CharPtr1
        li      $v0, 4                # In so integer ra man hinh duoi dang hexa
        syscall
        la      $a0, oldCharPtr2_Val  # In ra thong bao "CharPtr2 (before): "
        li      $v0, 4
        syscall
        la      $a0, CharPtr2
        lw      $a0, 0($a0)           # Lay gia tri luu trong word nho CharPtr2
        li      $v0, 34               # In so integer ra man hinh duoi dang hexa
        syscall
        la      $a0, newCharPtr2_Val  # In ra thong bao "CharPtr2 (after): "
        li      $v0, 4
        syscall

        # ----------------->[ Khoi tao gia tri de thuc hien copy ]<-------------
----
        la      $a0, CharPtr2         # Load dia chi cua CharPtr2
        la      $t9, Sys_TheTopOfFree #
        lw      $t8, 0($t9)           # Lay dia chi dau tien con trong
        sw      $t8, 0($a0)           # Cat dia chi do vao bien con tro
        lw      $t4, 0($t9)           # Dem so luong ki tu trong string
        lw      $t1, 0($s1)           # Load gia tri con tro CharPtr1
```

```
        lw      $t2, 0($a0)           # Load gia tri con tro CharPtr2
copy_loop:
        lb      $t3, 0($t1)           # Load 1 ki tu (tren cung) cua $t1 vao $t3
        sb      $t3, 0($t2)           # Luu 1 ki tu cua $t3 vao $t2
        addi    $t4, $t4, 1           # so luong ki tu trong string += 1
        addi    $t1, $t1, 1           # Dia chi ki tu tiep theo cua CharPtr1
        addi    $t2, $t2, 1           # Dia chi ki tu tiep theo cua CharPtr2
        beq     $t3, '\0', end_copy   # Check null = end string
        j       copy_loop
end_copy:
        sw      $t4, 0($t9)           # Kich thuoc cap phat = do dai string
        lw      $a0, 0($a0)           # Lay noi dung con tro CharPtr2
        li      $v0, 4                # In ra gia tri vung nho CharPtr2 tro den

        syscall
        j       menu

#---------------------------------------------------------
#  Tinh toan bo luong bo nho da cap phat cho cac bien dong
#---------------------------------------------------------
case5:
        la      $a0, used_total       # In "Tong luong bo nho da su dung: "
        li      $v0, 4
        syscall
        la      $t9, Sys_TheTopOfFree
        lw      $t9, 0($t9)           # Load dia chi luu o Sys_TheTopOfFree
        la      $t8, Sys_MyFreeSpace
        sub     $a0, $t9, $t8         # Sys_TheTopOfFree - Sys_MyFreeSpace
        li      $v0, 1
        syscall
        la      $a0, byte             # In ra don vi " byte(s)"
        li      $v0, 4
        syscall

        la      $a0, allocated_total  # In "Tong luong bo nho da cap phat: "
        li      $v0, 4
        syscall
        move    $a0, $s2
        li      $v0, 1
        syscall
        la      $a0, byte             # In ra don vi " byte(s)"
        li      $v0, 4
        syscall

        j       menu
```

```
#-----------------------------------------------------------
#  Tao ham Malloc2 cap phat mang 2 chieu kieu .word
#-----------------------------------------------------------
case6:
       la     $a0, input_row        # In thong bao "Nhap so cot: "
       li     $v0, 51               # Syscall to input dialog
       syscall
       addi   $s3, $a0, 0           # Luu so hang (row) thanh bien toan cuc
       la     $a0, input_col        # In thong bao "Nhap so cot: "
       li     $v0, 51               # Syscall to input dialog
       syscall
       addi   $s4, $a0, 0           # Luu so cot (cot) thanh bien toan cuc

       addi   $a1, $s3, 0           # Luu so hang (row) de thuc hien malloc
       addi   $a2, $s4, 0           # Luu so cot (col) de thuc hien malloc
       la     $a0, ArrayPtr         # Dia chi con tro cua array
       jal    malloc2
       j      menu
#-----------------------------------------------------------
#  Tao ham get/set cho mang 2 chieu
#-----------------------------------------------------------
case7.1:                            # GetArray[i][j]
       la     $a0, ArrayPtr
       lw     $s1, 0($a0)
       beqz   $s1, null             # Neu *ArrayPtr = 0 → mang chua cap phat
       la     $a0, input_i          # In thong bao "Nhap vi tri hang: "
       li     $v0, 51               # Syscall to input dialog
       syscall
       bge    $a0, $s3, invalid_idx # Neu so hang > nrows -> error
       move   $t1, $a0              # Luu lai vi tri hang
       la     $a0, input_j          # In thong bao "Nhap vi tri cot: "
       li     $v0, 51               # Syscall to input dialog
       syscall
       bge    $a2, $s4, invalid_idx # Neu so cot > ncols -> error
       move   $a1, $t1              # Luu vi tri hang vao $a1 de thuc hien get
       move   $a2, $a0              # Luu vi tri cot vao $a2 de thuc hien get
       la     $a0, ArrayPtr         # Load dia chi ArrayPtr de thuc hien get
       jal    get
       move   $a0, $v0              # Luu kqua tra ve vao $a0 de in ra man hinh
       li     $v0, 34               # Syscall to print integer
       syscall
       j      menu

case7.2:                            # SetArray[i][j]
       la     $a0, ArrayPtr
       lw     $s1, 0($a0)
```

```
        beqz    $s1, null               # Neu *ArrayPtr = 0 → mang chua cap phat
        la      $a0, input_i            # In thong bao "Nhap vi tri hang: "
        li      $v0, 51                 # Syscall to input dialog
        syscall
        move    $t1, $a0                # Luu lai vi tri hang
        la      $a0, input_j            # In thong bao "Nhap vi tri cot: "
        li      $v0, 51                 # Syscall to input dialog
        syscall
        move    $a2, $a0                # Luu vi tri cot vao $a2 de thuc hien get
        la      $a0, assign_val         # In thong bao "Nhap gia tri can gan: "
        li      $v0, 51                 # Syscall to input dialog
        syscall
        move    $a3, $a0
        move    $a1, $t1                # Luu vi tri hang vao $a1 de thuc hien get
        la      $a0, ArrayPtr           # Load dia chi ArrayPtr de thuc hien get
        jal     set
print_array:
        li      $t1, 0                  # row index
        li      $t2, 0                  # col index
        j       loop_j
loop_i:
        addi    $t1, $t1, 1             # row index
        beq     $t1, $s3, end_print     # For i = 0 -> nrows
        la      $a0, new_line           # new line
        li      $v0, 4
        syscall
        li      $t2, 0                  # reset col index
loop_j:
        beq     $t2, $s4, loop_i        # For j = 0 -> ncols
        la      $a0, ArrayPtr
        addi    $a1, $t1, 0
        addi    $a2, $t2, 0
        jal     get
        move    $a0, $v0                # Luu gia tri vua get duoc vao $a0 de thuc
hien syscall 1
        li      $v0, 1                  # Syscall to print integer
        syscall
        la      $a0, tab                # space between integer
        li      $v0, 4
        syscall
        addi    $t2, $t2, 1             # col index
        j       loop_j
end_print:
        j       menu

null:   la      $a0, unallocated
```

```
        li      $v0, 4
        syscall
        j       extra_menu7
invalid_idx:
        la      $a0, out_of_bound
        li      $v0, 4
        syscall
        j       extra_menu7
#-----------------------------------------
#  Ham khoi tao cho viec cap phat dong
#  @param    khong co
#  @detail   Danh dau vi tri bat dau cua vung nho co the cap phat duoc
#-----------------------------------------
SysInitMem:
        la      $t9, Sys_TheTopOfFree # Con tro den dia chi dau tien con trong
        la      $t7, Sys_MyFreeSpace  # Lay dia chi dau tien con trong, khoi tao
        sw      $t7, 0($t9)           # Luu lai
        jr      $ra
#-----------------------------------------
#  Ham cap phat bo nho dong cho cac bien con tro
#  @param  [in/out]   $a0   Chua dia chi cua bien con tro can cap phat
#                           Khi ham ket thuc, dia chi vung nho duoc cap phat se
luu tru vao bien con tro
#  @param  [in]       $a1   So phan tu can cap phat
#  @param  [in]       $a2   Kich thuoc 1 phan tu, tinh theo byte
#  @return            $v0   Dia chi vung nho duoc cap phat
#-----------------------------------------
malloc: la      $t9, Sys_TheTopOfFree
        lw      $t8, 0($t9)           # Lay dia chi dau tien con trong
        li      $t1, 4                # Do dai 1 word nho
        bne     $a2, $t1, valid       # Neu khong phai cap phat kieu WORD thi OK
        divu    $t8, $t1              # Dia chi bat dau cap phat chia het cho 4?
        mfhi    $t2                   # Luu phan du (remainder) vao $t2
        beqz    $t2, valid            # If Phan du = 0 -> Kich thuoc hop le
        sub     $t3, $t1, $t2         # Else can cap phat them (4-remainder) bits
        add     $t8, $t8, $t3         # Dia chi bat dau cap phat
valid:  sw      $t8, 0($a0)           # Cat dia chi do vao bien con tro
        addi    $v0, $t8, 0           # Dong thoi la ket qua tra ve cua ham
        mul     $t7, $a1, $a2         # Tinh kich thuoc cua mang can cap phat
        add     $t6, $t8, $t7         # Tinh dia chi dau tien con trong
        sw      $t6, 0($t9)           # Luu tro lai Sys_TheTopOfFree
        add     $s2, $s2, $t7         # Cap nhap tong luong bo nho da cap phat

        #----------->[ Cap phat thanh cong. In ra man hinh ]<-----------
        la      $a0, malloc_success   # In ra thong bao malloc successfully
        li      $v0, 4
```

```
        syscall
        move    $a0, $a3            # In ra kieu malloc
        li      $v0, 4
        syscall
        addi    $a0, $t8, 0         # Malloc start address
        li      $v0, 34             # In so integer ra man hinh duoi dang hexa
        syscall
        la      $a0, arrow          # In ra man hinh " --> "
        li      $v0, 4
        syscall
        addi    $a0, $t6, 0         # Malloc end address
        li      $v0, 34             # In so integer ra man hinh duoi dang hexa
        syscall
        jr      $ra
#----------------------------------------
#  Ham cap phat bo nho dong cho bien con tro mang 2 chieu
#  @param  [in/out]   $a0   Chua dia chi cua bien con tro can cap phat
#                           Khi ham ket thuc, dia chi vung nho duoc cap phat se
luu tru vao bien con tro
#  @param  [in]       $a1   So hang cua array (Number of row)
#  @param  [in]       $a2   So cot cua array (Number of col)
#  @return            $v0   Dia chi vung nho duoc cap phat
#----------------------------------------
malloc2:
        addiu   $sp, $sp, -4        # them 1 phan tu vao stack
        sw      $ra, 4($sp)         # push $ra
        mul     $a1, $a1, $a2       # tra ve so phan tu cua Array
        li      $a2, 4              # Mang kieu .word (4 bytes)
        la      $a3, arrayPtr_Add
        jal     malloc              # Cap phat mang co so phan tu = row x col
        lw      $ra, 4($sp)         # pop $ra khoi stack de return
        addiu   $sp, $sp, 4         # xoa bo nho stack da cap phat
        jr      $ra
#----------------------------------------
#  Ham get/set gia tri cua 1 phan tu trong mang 2 chieu
#  @param  [in]       $a0   Chua dia chi cua mang 2 chieu
#  @param  [in]       $a1   Vi tri hang (ROW INDEX)
#  @param  [in]       $a2   Vi tri cot (COL INDEX)
#  @param  [in]       $a3   Gia tri can gan cho A[i][j]: input cua ham set
#  @param  [in]       $s4   So cot cua mang 2 chieu (NCOLS)
#  @return            $v0   Gia tri A[i][j]: ket qua tra ve cua ham get
#----------------------------------------
get:    mul     $t0, $s4, $a1       # Cong thuc xac dinh vi tri cua A[i][j]:
        addu    $t0, $t0, $a2       # Index = i * ncols + j
        sll     $t0, $t0, 2         # Imm = index * 4 (bytes)
        lw      $t3, 0($a0)         # Load dia chi ArrayPtr tro den
```

```
        addu    $t0, $t0, $t3         # Dia chi A[i][j] = Dia chi co so + Imm
        lw      $v0, 0($t0)
        jr      $ra

set:    mul     $t0, $s4, $a1         # Cong thuc xac dinh vi tri cua A[i][j]:
        addu    $t0, $t0, $a2         # index = i * ncols + j
        sll     $t0, $t0, 2           # imm = index * 4 (bytes)
        lw      $t3, 0($a0)           # Load dia chi ArrayPtr tro den
        addu    $t0, $t0, $t3         # Dia chi A[i][j] = Dia chi co so + imm
        sw      $a3, 0($t0)
        jr      $ra

#----------------------------------------
#  2 ham in ra dia chi va gia tri cua pointer
#  @detail     ptr_val: in gia tri
#              ptr_add: in dia chi
#----------------------------------------
ptr_val:
        lw      $a0, 0($a0)           # Lay gia tri luu trong word nho
ptr_add:
        li      $v0, 34               # In so integer ra man hinh duoi dang hexa
        syscall
        jr      $ra
```

Ý nghĩa các thanh ghi và hàm

- **Vùng nhớ .data**

```
CharPtr:            Con trỏ kiểu Char. 1 phần tử = 1 byte
BytePtr:            Con trỏ kiểu Byte. 1 phần tử = 1 byte
WordPtr:            Con trỏ kiểu Word. 1 phần tử = 4 bytes
ArrayPtr:           Con trỏ mảng 2 chiều kiểu Word
CharPtr1, CharPtr2: 2 con trỏ xâu kí tự để thực hiện copy
string_copy:        Vùng nhớ 100 bytes để lưu xâu kí tự nhập từ bàn phím
# Còn lại là các chuỗi asciiz để thực hiện lệnh syscall.
```

- **Thanh ghi (một số thanh ghi đặc biệt)**

```
$s2         # tổng lượng bộ nhớ đã cấp phát cho các biến con trỏ
$s3         # số hàng của mảng 2 chiều ở option 6
$s4         # số cột của mảng 2 chiều ở option 6
$t9         # lưu địa chỉ Sys_TheTopOfFree
$t8         # lưu địa chỉ đầu tiên còn trống để thực hiện cấp phát
$a0         # thường dùng để load địa chỉ các con trỏ
$a1         # thường dùng để lưu số phần tử cấp phát hoặc số hàng
$a2         # thường dùng để lưu độ dài 1 phần tử hoặc số cột
```

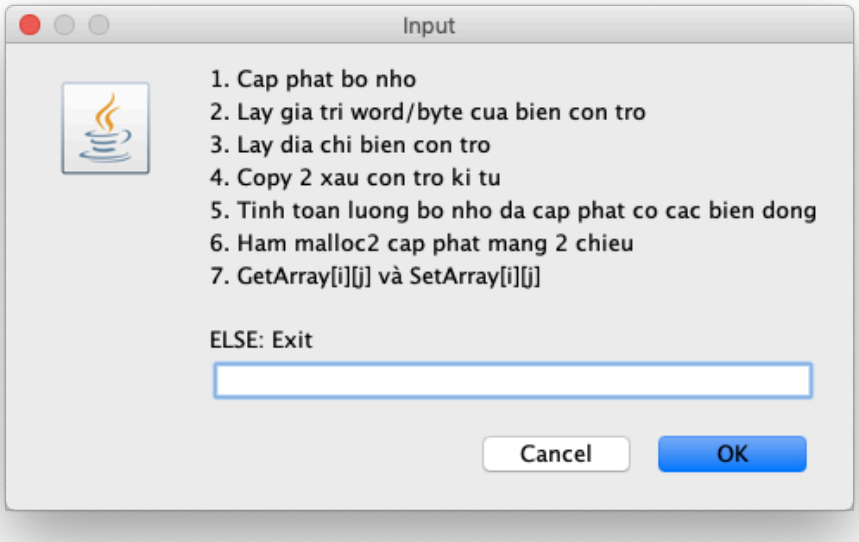
```
$a3          # thường dùng để lưu các giá trị gán như SetArray[i][j]
```

- **Hàm**

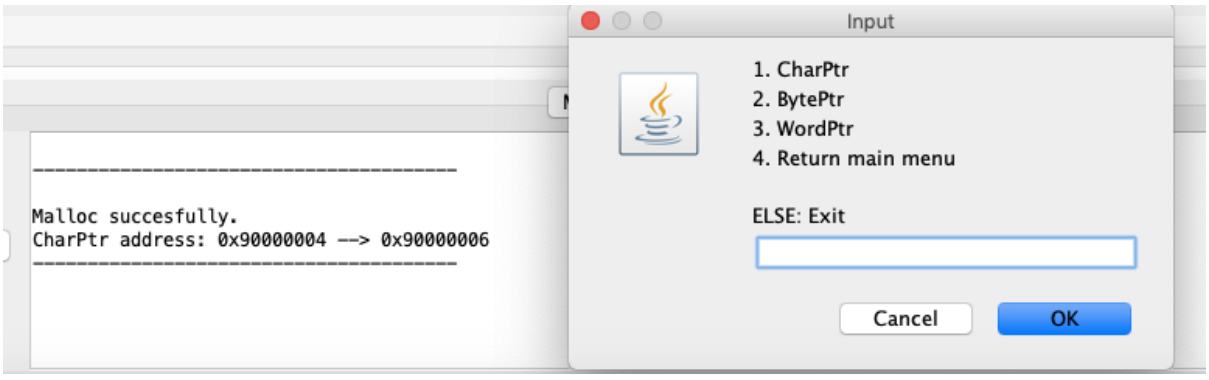
```
SysInitMem:         Hàm khởi tạo cho việc cấp phát động.
malloc:             Hàm cấp phát bộ nhớ động cho các biến con trỏ.
malloc2:            Hàm cấp phát bộ nhớ động cho mảng 2 chiều
get:                GetArray[i][j], lấy ra giá trị 1 phần tử trong mảng
set:                SetArray[i][j], lưu giá trị vào phần tử trong mảng
copy_loop:          Copy từng kí tự trong CharPtr1 lưu vào CharPtr2
ptr_val:            In ra giá trị của biến con trỏ
ptr_add:            In ra địa chỉ của biến con trỏ
# Còn lại là các hàm xử lý từng option của menu
```

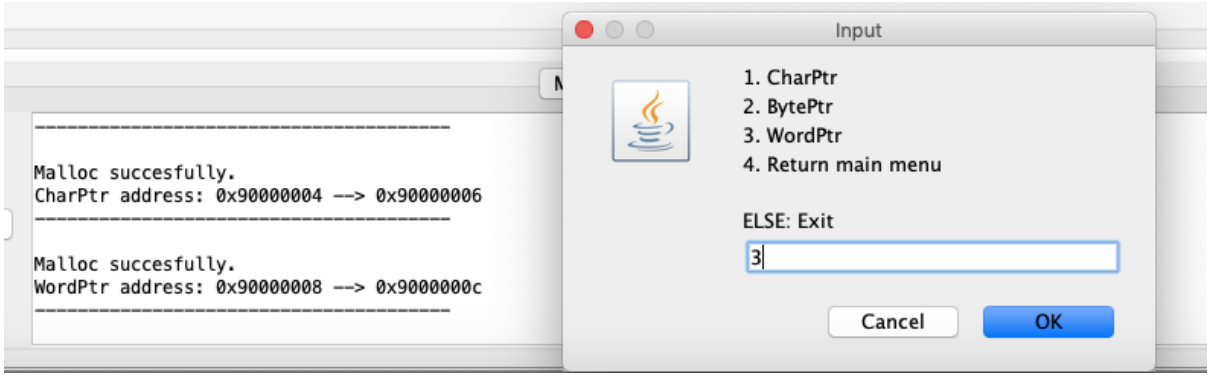
Kết quả demo

- **Menu**



- **Yêu cầu 1:**





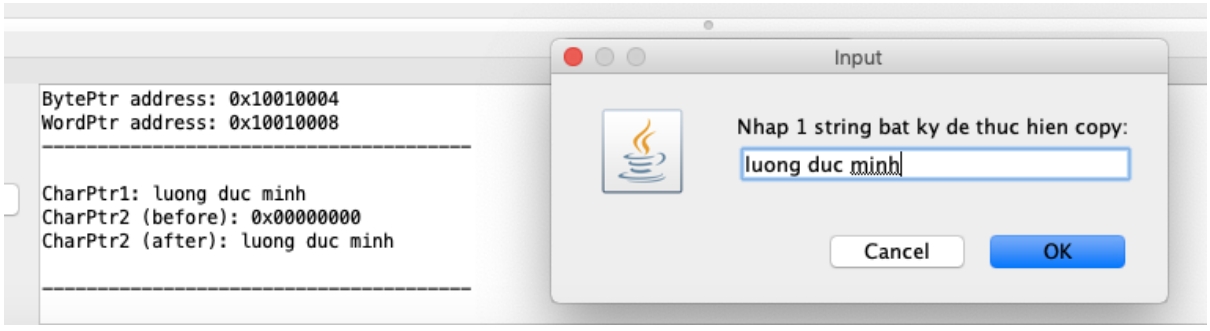
- **Yêu cầu 2:**

```
----------------------------------------

CharPtr value: 0x90000004
BytePtr value: 0x00000000
WordPtr value: 0x90000008
----------------------------------------
```

- **Yêu cầu 3:**

```
----------------------------------------

CharPtr address: 0x10010000
BytePtr address: 0x10010004
WordPtr address: 0x10010008
----------------------------------------
```

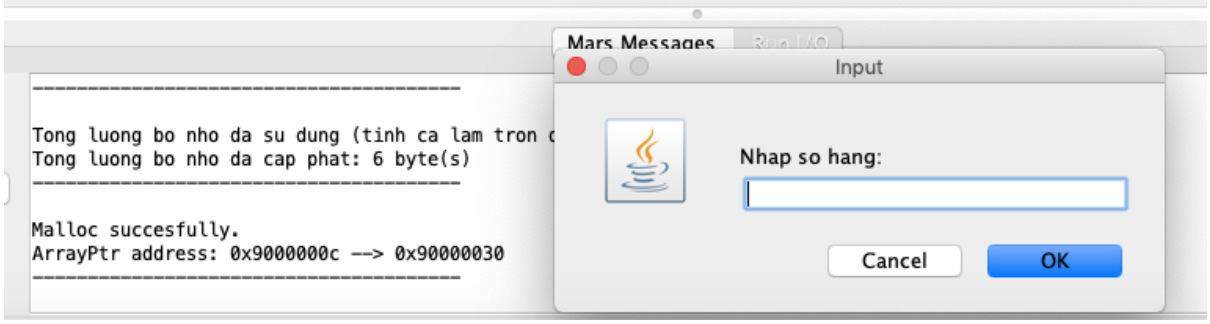
- **Yêu cầu 4:**



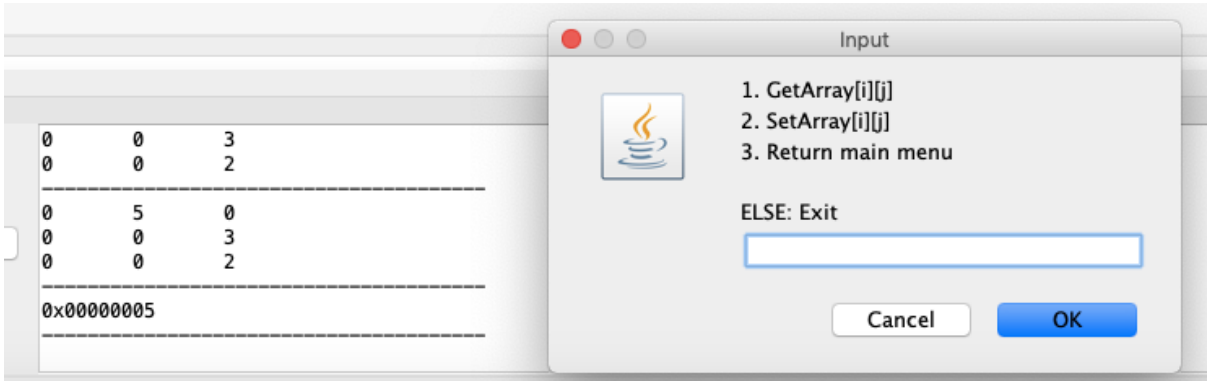
- **Yêu cầu 5:**

```
----------------------------------------

Tong luong bo nho da su dung (tinh ca lam tron de sua loi): 8 byte(s)
Tong luong bo nho da cap phat: 6 byte(s)
----------------------------------------
```

- **Yêu cầu 6:**



- **Yêu cầu 7:**



Đề số 8

Phương hướng giải quyết

Bước 1: Nhập vào một string, kiểm tra xem string đó có phải string hợp lệ hay không (không phải là xâu rỗng hay độ dài không phải bội của 8), nếu không hợp lệ thì báo lỗi.

Bước 2: Chia string nhập vào thành các khối 8 byte. Mỗi khối gồm 2 block 4 byte và lưu vào 2 ổ đĩa. Sau đó thực hiện phép xor sau đó in vào đĩa còn lại.

Bước 3: Tiến hành thực hiện lặp đi lặp lại để in ra màn hình.

Source code (mã nguồn)

```
.data

start:  .asciiz
"     Disk1              Disk2              Disk3      \n "
```

```
string: .asciiz
" --------------     --------------     -------------\n"
a1:  .asciiz "|     "
a2: .asciiz "     |   "
b1: .asciiz "[[ "
b2: .asciiz "]]   "
prompt: .asciiz "Nhap chuoi ki tu : "
contPrompt: .asciiz "\nAn ENTER de thuc hien lai..."
message1: .asciiz "\n\n----------- Let's start -----------\n"
error: .asciiz "\nDo dai xau phai la boi cua 8\n"
error2:.asciiz "\nXau khong duoc rong\n"
hex:  .byte
'0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F'
      .align 2
buffer: .space  4000
buff: .word 0
.text
      li    $v0, 4            # in thong diep start
      la    $a0, message1     # pointer to start message in memory
      syscall
begin:
      la $a0, prompt
      li $v0,4
      syscall                 #v0,4 la in string.
      la $a0, buffer
      li $a1, 4000
      li $v0 ,8               #nhap vao xau co do dai 4000
      syscall
      la $s0, buffer          #$s0: luu tru dia chi dau tien cua buffer
#----------------kiem tra do dai cua chuoi-----------------------
#---------------dem tung ki tu trong buffer-------------
count:
      lb $t0, 0($s0)          #load gia tri cua byte dau tien = t0
                              #load tung byte 1
      beq $t0, '\n', check    # neu t0=ki tu xuong dong,ket thuc chuoi
      addi $s0,$s0,1          #tang s0++ , tang den phan tu tiep theo
      j count
check:
      sub $s0, $s0, $a0       #s0 :do dai cua xau nhap vao
      srl $t0, $s0, 3         #dich phai 3 bit,xet truong hop xau rong
                              #VD 0000
      andi $s0, $s0, 7        #kiem tra so co chia het cho 8 khong
      bne $s0, $zero, printerror    # neu khong thi bao loi
      beq $t0, $zero, printerror2   # ktra xem nhap vao xau hay chua
      la $a0, start
      li $v0,4
      syscall                 #in string ben tren
```

```
      jal print               #goi den print
      move $a0,$0             #gan a0 = zero
#----------------------in cac block ra man hinh-------------------------
-
loop:
      jal printline           #goi ham print_line voi tham so a0(=0)
      addi $a0,$a0,1
      bne $a0,$t0,loop        #t0 la luong cac khoi 2 block
      jal print
      j readEnter


printline:                    #a0 = 0
      addiu $sp,$sp,-8        #con tro stack
      sw $ra, 4($sp)
      sw $a0, 8($sp)          # luu gia tri a0 vao stack
      rem $t6,$a0,3           #chia 3 lay du
      sll $t1, $a0, 3         #dich phai 3 bit
      la $t2,buffer           #load dia chi cua buffer vao t2
      add $t2,$t2,$t1         #t2: luu dia chi cua cac khoi
      lw $t3, 0($t2)          #load 4 byte dau t2 luu t3
      lw $t4, 4($t2)          #load 4 byte tiep theo cua t2 luu t4
      xor $t5,$t3,$t4         #luu KQ phep xor vao t5
      beqz $t6,row0
      beq $t6, 1, row1
      j row2

#------------------in cac block ra man hinh--------------------
row0:
      move $a1,$t3
      jal printblock          #goi ham print_block voi tham so a1
      move $a1,$t4            # truyen t4 vao a1
      jal printblock
      jal printxor
      j endswitch
row1:
      move $a1,$t3
      jal printblock
      jal printxor
      move $a1,$t4
      jal printblock
      j endswitch
row2:
      jal printxor
      move $a1,$t3
      jal printblock
      move $a1,$t4
```

```
      jal printblock
#-------------xuong dong-------------------
endswitch:
      li $a0,'\n'
      li $v0,11
      syscall                 # in ra ky tu '\n'
      lw $a0, 8($sp)
      lw $ra, 4($sp)
      addiu $sp,$sp,8
      jr $ra
#---------------in tung block--------------------------
printblock:                   # nhan gia tri $a1 dau vao
      la $a0, a1              #string a1 la dau | vao dong dau cach
      li $v0,4
      syscall                 #in string
      la $a0, buff
      sw $a1, 0($a0)
      li $v0,4
      syscall
      la $a0, a2              #string a2: in ra dau cach roi dau
|
      li $v0,4
      syscall
      jr $ra                  # nhay den gia tri thanh ghi $ra
#-------------in ra ket qua cua phep xor-----------------
printxor:
      la $a0, b1
      li $v0,4
      syscall                 #in ra [[    .
      la $s1,hex

      srl $s0, $t5, 4         # dich phai 4 bit
      and $s0,$s0,0xf         # giu lai gia tri cuoi cua s0;
      add $s2,$s1,$s0         # lay dia chi ki tu ascii tuong ung
      lb $a0,0($s2)           # load kí tu ascii tu dia chi tren
      li $v0,11
      syscall                 #in ra ky tu duy nhat

      and $s0,$t5,0xf         # in ra man hinh ki tu tiep theo
      add $s2,$s1,$s0
      lb $a0,0($s2)
      li $v0,11
      syscall

      li $a0,','
      li $v0,11
      syscall
```

```
srl $s0, $t5, 12
and $s0,$s0,0xf
add $s2,$s1,$s0
lb $a0,0($s2)
li $v0,11
syscall

srl $s0, $t5, 8
and $s0,$s0,0xf
add $s2,$s1,$s0
lb $a0,0($s2)
li $v0,11
syscall

li $a0,','
li $v0,11
syscall

srl $s0, $t5, 20
and $s0,$s0,0xf
add $s2,$s1,$s0
lb $a0,0($s2)
li $v0,11
syscall

srl $s0, $t5, 16
and $s0,$s0,0xf
add $s2,$s1,$s0
lb $a0,0($s2)
li $v0,11
syscall

li $a0,','
li $v0,11
syscall

srl $s0, $t5, 28
add $s2,$s1,$s0
lb $a0,0($s2)
li $v0,11
syscall

srl $s0, $t5, 24
and $s0,$s0,0xf
add $s2,$s1,$s0
```

```
        lb $a0,0($s2)
        li $v0,11
        syscall

        la $a0, b2
        li $v0,4
        syscall
        jr $ra

print:
        la $a0, string          #print dau ngan cach ------
        li $v0,4
        syscall
        jr $ra                  #quay lai neu phia tren co lenh jal

printerror:
        la $a0,error
        li $v0,4
        syscall                 #in loi chuoi khong phai boi cua 8
        j readEnter
printerror2:
        la $a0,error2
        li $v0,4
        syscall
        j readEnter             #in bao loi xau rong
#---------------------ket thuc ctrinh ------------------------
end:
        li $v0,10
        syscall                 #exit
readEnter:                      #yeu cau nguoi dung an enter de tiep tuc
        li    $v0, 4
        la    $a0, contPrompt
        syscall
        li    $v0, 12
        syscall
        move  $t0, $v0
        bne   $t0, 10, end      #thoat khoi ctr neu khong an enter
        j     begin             #quay lai begin
```

Ý nghĩa các thanh ghi và hàm:

- **Hàm:**
  - count: đếm độ dài của chuỗi
  - check: kiểm tra xem chuỗi có thỏa mãn yêu cầu đề bài không
  - printline: load giá trị vào các block, thực hiện phép xor
  - row1/row2/row3: in từng trường hợp.
  - printblock: in 1 block ra màn hình.
  - printxor: in kết quả của phép xor.

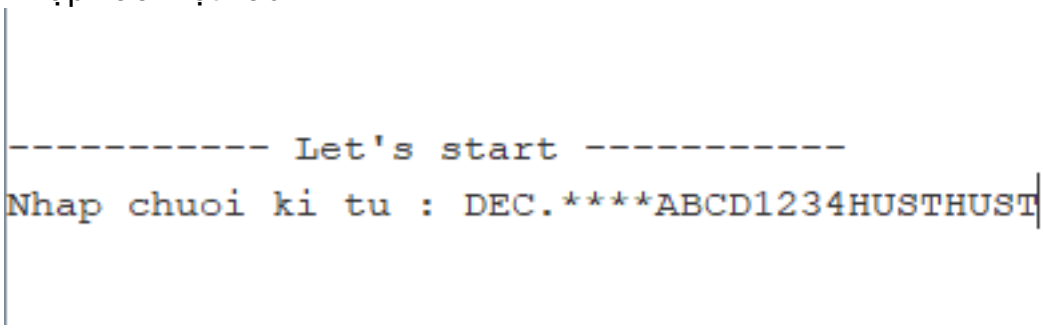
- readEnter: đọc kí tự '\n' để restart chương trình

- **Thanh ghi**

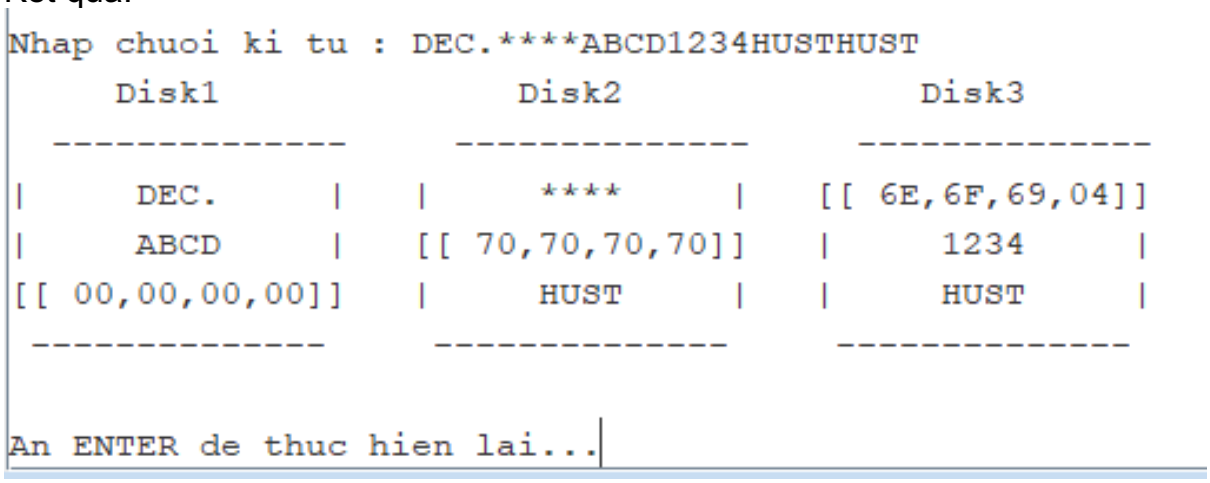
```
$t3         # lưu giá trị của block thứ nhất.
$t4         # lưu giá trị của block thứ hai.
$t5         # lưu kết quả phép xor giữa 2 block.
$t0         # số lượng các khối.
$t2         # lưu địa chỉ của buffer
```

Demo chương trình

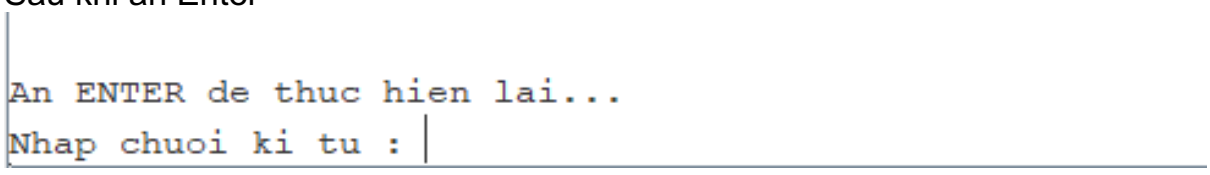
- Nhập vào một xâu:



- Kết quả:



- Sau khi ấn Enter



- Khi nhập vào 1 chuỗi không phải bội của 8:

```
An ENTER de thuc hien lai...
Nhap chuoi ki tu : 1234567

do dai xau phai la boi cua 8

An ENTER de thuc hien lai...
```

- Nếu không nhập gì cả:

```
An ENTER de thuc hien lai...
Nhap chuoi ki tu :

Xau khong duoc rong

An ENTER de thuc hien lai...
```

- Nhập một kí tự khác Enter:

```

An ENTER de thuc hien lai...h
-- program is finished running --
```