ĐẠI HỌC BÁCH KHOA HÀ NỘI
Trường SOICT
---ꕥ🕮ꕥ---

TIỂU LUẬN

Đề tài: Báo cáo cuối kỳ.

Giáo viên hướng dẫn: Lê Bá Vui
Sinh viên thực hiện: Ngô Vĩnh Khánh, Nguyễn Đình Trung

Nhóm: 15
Lớp:ICT-01

Computer Architecture
Final exam

Full name: Ngo Vinh Khanh, Problem 8
Student ID: 20215211
Full name: Nguyen Dinh Trung, Problem 6
Student ID: 20215249

Assignment 6:
Đề bài:

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 kiểu 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 của biến con trỏ.
3) Viết hàm lấy địa chỉ biến con trỏ.
4) Viết hàm thực hiện copy 2 con trỏ xâu kí tự.
5) Viết hàm giải phóng bộ nhớ đã cấp phát cho các biến con trỏ
6) Viết hàm tính toàn bộ lượng bộ nhớ đã cấp phát.
7) Hãy viết hàm malloc2 để cấp phát cho mảng 2 chiều kiểu .word với tham số vào gồm:
    a. Địa chỉ đầu của mảng
7) Hãy viết hàm malloc2 để cấp phát cho mảng 2 chiều kiểu .word với tham số vào gồm:
    a. Địa chỉ đầu của mảng
    b. Số dòng
    c. Số cột
8) Tiếp theo câu 7, 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.

Source code:

```
.data
CharPtr1:       .word   0
CharPtr2:       .word   0
ArrayPtr:       .word   0
Array2Ptr:      .word   0
mess1: .asciiz  "\n\n1. Mang mot chieu\n"
mess2: .asciiz  "2. Sao chep mang ky tu\n"
mess3: .asciiz  "3. Mang hai chieu\n"
```

```
mess4: .asciiz  "4. Giai phong bo nho\n"
mess5: .asciiz  "5. Hien thi bo nho\n"
mess6:  .asciiz "6. Ket thuc chuong trinh\n"
mess0.1:        .asciiz "So phan tu: "
mess0.2:        .asciiz "So byte moi phan tu (1 hoac 4): "
mess0.3:        .asciiz "Nhap phan tu: \n"
mess1.1:        .asciiz  "Gia tri cua con tro: "
mess1.2:        .asciiz  "\nDia chi cua con tro: "
mess1.3:        .asciiz "\nTong bo nho da cap phat: "
mess2.1:        .asciiz "So ky tu toi da: "
mess2.2:        .asciiz "\nNhap chuoi ky tu: "
mess2.3:        .asciiz  "\nChuoi ky tu duoc copy: "
mess3.1:        .asciiz  "\nSo hang: "
mess3.2:        .asciiz  "\nSo cot: "
mess3.3:        .asciiz  "\n1. getArray[i][j]\n"
mess3.4:        .asciiz  "2. setArray[i][j]\n"
mess3.5:        .asciiz  "3. Thoat\n"
mess3.6:        .asciiz "\nGia tri cua phan tu: "
mess3.01:       .asciiz  "i = "
mess3.02:       .asciiz  "j = "
mess4.1:        .asciiz  "Da giai phong toan bo bo nho cap phat.\n"
select:         .asciiz  "Lua chon: "
errmess:        .asciiz  "\nSo vua nhap khong hop le.\n"

.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
#Khoi tao vung nho cap phat dong
        jal     SysInitMem
#Hien thi menu
menu:
        li      $v0, 4
        la      $a0, mess1
        syscall
        la      $a0, mess2
        syscall
        la      $a0, mess3
        syscall
        la      $a0, mess4
        syscall
        la      $a0, mess5
        syscall
```

```
        la      $a0, mess6
        syscall
        la      $a0, select
        syscall
        li      $v0, 5 #Nhap lua chon
        syscall
#1. Sua loi bo nho cua vi du
case_1:
        bne     $v0, 1, case_2
        li      $v0, 4
        la      $a0, mess0.1
        syscall
        li      $v0, 5 #Nhap so phan tu cua day 1 chieu
        syscall
        bltz    $v0, error #Kiem tra so da nhap vao, neu v0 < 0 thi bao loi va yeu cau
nhap lai
        move    $a1, $v0 #Luu so phan tu vao a1
        li      $v0, 4
        la      $a0, mess0.2
        syscall
        li      $v0, 5 #Nhap kich thuc moi phan tu cua day
        syscall
is1:    beq     $v0, 1, ready
is4:    beq     $v0, 4, ready
        j       error    #Kiem tra kich thuoc nhap vao, neu không phai 1 hay 4 thi bao loi
va yeu cau nhap lai
ready:  move    $a2, $v0 #Luu kich thuoc moi phan tu vao a2
        la      $a0, ArrayPtr #Luu dia chi bat dau cua chuoi mot chieu
        jal     malloc #Chay ham malloc cho chuoi 1 chieu
        move    $t0, $v0 #Dat dia chi bat dau tra tu malloc vao t0
        li      $v0, 4
        la      $a0, mess0.3
        syscall
        move    $a0, $t0 #Dat a0 = t0
        li      $t0, 0 #Dat t0 = 0
input_loop:
        beq     $t0, $a1, input_end #Bat dau vong lap nhap du lieu, ket thuc khi t0 = a1
        li      $v0, 5
        syscall
        bne     $a2, 1, byte_4
byte_1:
        sb      $v0, 0($a0)
        addi    $a0, $a0, 1 #Neu kich thuoc phan tu bang 1 thi con tro tien 1 don vi
        addi    $t0, $t0, 1
        j       input_loop
byte_4:
```

```
        sw      $v0, 0($a0)
        addi    $a0, $a0, 4 #Neu kich thuoc phan tu bang 4 thi con tro tien 4 don vi
        addi    $t0, $t0, 1
        j       input_loop
input_end:
#2. Gia tri cua con tro
        li      $v0, 4
        la      $a0, mess1.1
        syscall
        la      $a0, ArrayPtr
        jal     getValue
        move    $a0, $v0
        li      $v0, 34
        syscall
#3. Dia chi cua con tro
        li      $v0, 4
        la      $a0, mess1.2
        syscall
        la      $a0, ArrayPtr
        jal     getAddress
        move    $a0, $v0
        li      $v0, 34
        syscall

        j       menu
#4. Vi?t hàm th?c hi?n copy 2 con tr? xâu kí t?.
case_2:
        bne     $v0, 2, case_3
        li      $v0, 4
        la      $a0, mess2.1
        syscall
        li      $v0, 5 #Nhap vao so ky tu toi da cua chu?i
        syscall
        move    $a1, $v0 #Luu so ky tu toi da vao a1
        li      $a2, 1 #Dat a2 = 1
        la      $a0, CharPtr1 #Dat dia chi cua chuoi 1 vao a0 va goi malloc
        jal     malloc
        move    $s0, $v0 #Dat s0 lam bien con tro cua chuoi 1
        la      $a0, CharPtr2#Dat dia chi cua chuoi 2 vao a0 va goi malloc
        jal     malloc
        move    $s1, $v0 #Dat s0 lam bien con tro cua chuoi 2
        li      $v0, 4
        la      $a0, mess2.2
        syscall
        move    $a0, $s0 #Nhap vao chuoi thu nhat
        li      $v0, 8
```

```
        syscall
        move   $a1, $s1 #Dat a1 la bien con tro cua chu?i 2.
        jal    strcpy
        li     $v0, 4  #In ra hai chuoi
        la     $a0, mess2.3
        syscall
        move   $a0, $s1
        syscall
        j      menu
#7. Viet ham malloc 2:
case_3:
        bne    $v0, 3, case_4
        li     $v0, 4
        la     $a0, mess3.1
        syscall
        li     $v0, 5 #Nhap vao so hang
        syscall
        move   $a1, $v0
        li     $v0, 4
        la     $a0, mess3.2
        syscall
        li     $v0, 5 #Nhap vao so cot
        syscall
        move   $a2, $v0
        la     $a0, Array2Ptr #Luu vao a0 dia chi cua mang 2 chieu
        jal    malloc2
        move   $t0, $v0 #Gan t0 bien con tro
        li     $v0, 4
        la     $a0, mess0.3
        syscall
        move   $a0, $t0 #Gan a0 thanh bien con tro
        add    $t0, $0, $0#Khoi tâo t0 = 0
        move   $t1, $a1 #t1 la so hang
        mul    $a1, $a1, $a2 #a1 la so phan tu
input_loop2:
        beq    $t0, $a1, input_end2 #su dung chuoi de nhap vao day, ket thuc khi t0 ==
so phan tu
        li     $v0, 5
        syscall
        sw     $v0, 0($a0)
        addi   $a0, $a0, 4
        addi   $t0, $t0, 1
        j      input_loop2
input_end2:
        move   $a1, $t1 #tra lai so hang ve a1
#8. Ham getArray va setArray
```

```
sub_menu:
        li      $v0, 4
        la      $a0, mess3.3
        syscall
        la      $a0, mess3.4
        syscall
        la      $a0, mess3.5
        syscall
        la      $a0, select
        syscall
        li      $v0, 5
        syscall
sub_case_1:
        bne     $v0, 1, sub_case_2
        li      $v0, 4
        la      $a0, mess3.01
        syscall
        li      $v0, 5 #Nhap so hang
        syscall
        move    $s0, $v0 #Luu vao s0
        li      $v0, 4
        la      $a0, mess3.02
        syscall
        li      $v0, 5 #Nhap so cot
        syscall
        move    $s1, $v0 #Luu vao s1
        la      $t0, Array2Ptr
        lw      $a0, 0($t0)
        jal     getArray
        move    $s2, $v0
        li      $v0, 4
        la      $a0, mess3.6
        syscall
        li      $v0, 1
        move    $a0, $s2
        syscall
        j       sub_menu
sub_case_2:
        bne     $v0, 2, sub_case_3
        li      $v0, 4
        la      $a0, mess3.01
        syscall
        li      $v0, 5
        syscall
        move    $s0, $v0
        li      $v0, 4
```

```
        la      $a0, mess3.02
        syscall
        li      $v0, 5
        syscall
        move    $s1, $v0
        move    $s2, $v0
        li      $v0, 4
        la      $a0, mess0.3
        syscall
        li      $v0, 5
        syscall
        la      $t0, Array2Ptr
        lw      $a0, 0($t0)
        jal     setArray
        j       sub_menu
sub_case_3:
        bne     $v0, 3, error
        j       menu
#5. Gi?i phóng b? nh?
case_4:
        bne     $v0, 4, case_5
        jal     free
        li      $v0, 4
        la      $a0, mess4.1
        syscall
        li      $v0, 4
        la      $a0, mess1.3
        syscall
        jal     memoryCalculate
        move    $a0, $v0
        li      $v0, 1
        syscall
        j       menu
#6. Vi?t hàm tính toàn b? l??ng b? nh? ?ã c?p phát.
case_5:

        bne     $v0, 5, case_6
        li      $v0, 4
        la      $a0, mess1.3
        syscall
        jal     memoryCalculate
        move    $a0, $v0
        li      $v0, 1
        syscall
        j       menu
case_6:
```

```
        bne $v0, 6, error
        li $v0, 10
        syscall
error:
        li      $v0, 4
        la      $a0, errmess
        syscall
        j       menu
#----------------------------------------

SysInitMem:
        la      $t9, Sys_TheTopOfFree       # Lay con tro chua dau tien con trong, khoi
tao
        la      $t7, Sys_MyFreeSpace        # Lay dia chi dau tien con trong, khoi tao
        sw      $t7, 0($t9)             # Luu lai
        jr      $ra
#----------------------------------------

malloc:
        la      $t9, Sys_TheTopOfFree
        lw      $t8, 0($t9)             # Lay dia chi dau tien con trong
        bne     $a2, 4, initialize      # Neu mang khoi tao co kieu Word, kiem tra dia chi
dau co dam bao quy tac khong
        andi    $t0, $t8, 0x03          # Lay so du khi chia dia chi trong cho 4
        beq     $t0, 0, initialize      # Neu khong co du, bo qua
        addi    $t8, $t8, 4             # Neu co, tien toi dia chi chia het cho 4 tiep theo
        subu    $t8, $t8, $t0
initialize:
        sw      $t8, 0($a0)     # Cat dia chi do vao bien con tro
        move    $v0, $t8        # 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 dia chi dau tien do vao bien
Sys_TheTopOfFree
        jr      $ra
#----------------------------------------

getValue:
        lw      $v0, 0($a0)     # Lay gia tri cua bien con tro trong o nho co dia chi luu
trong $a0
        jr      $ra
#----------------------------------------

getAddress:
        add     $v0, $0, $a0    # Lay dia chi tu $a0
```

```
        jr      $ra
#----------------------------------------

strcpy:
        add     $t0, $0, $a0    # Khoi tao $t0 o dau xau ky tu nguon
        add     $t1, $0, $a1    # Khoi tao $t1 o dau xau ky tu dich
        addi    $t2, $0, 1      # Khoi tao $t2 la ky tu khac '\0' de chay vong lap
cpyLoop:
        beq     $t2, 0, cpyLoopEnd    # Neu ky tu duoc copy trong vong lap truoc la '\0',
dung vong lap
        lb      $t2, 0($t0)           # Doc ky tu o xau ky tu nguon
        sb      $t2, 0($t1)           # Luu ky tu vua doc vao xau ky tu dich
        addi    $t0, $t0, 1           # Chuyen $t0 tro sang vi tri cua phan tu tiep theo
trong xau ky tu nguon
        addi    $t1, $t1, 1           # Chuyen $t1 tro sang vi tri cua phan tu tiep theo
trong xau ky tu dich
        j       cpyLoop
cpyLoopEnd:
        jr      $ra
#----------------------------------------

free:
        addi    $sp, $sp, -4    # Khoi tao 1 vi tri trong stack
        sw      $ra, 0($sp)     # Luu $ra vao stack
        jal     SysInitMem      # Tai lap lai vi tri cua con tro luu dia chi dau tien con trong
        lw      $ra, 0($sp)     # Tra gia tri cho $ra
        addi    $sp, $sp, 4     # Xoa stack
#----------------------------------------

memoryCalculate:
        la      $t0, Sys_MyFreeSpace        # Lay dia chi dau tien duoc cap phat
        la      $t1, Sys_TheTopOfFree       # Lay dia chi luu dia chi dau tien con trong
        lw      $t2, 0($t1)             # Lay dia chi dau tien con trong
        sub     $v0, $t2, $t0           # Tru hai dia chi cho nhau
        jr      $ra
#----------------------------------------

malloc2:
        addi    $sp, $sp, -12   # Luu cac gia tri can thiet de thuc hien 1 chuong trinh con
malloc trong chuong trinh con nay
        sw      $ra, 8($sp)
        sw      $a1, 4($sp)
        sw      $a2, 0($sp)
        mul     $a1, $a1, $a2   # $a1 = so phan tu = so hang * so cot
        addi    $a2, $0, 4      # $a2 = so byte cua 1 phan tu kieu word = 4
        jal     malloc          # Chuyen mang 2 chieu thanh mang 1 chieu, khoi tao
```

```
        lw      $ra, 8($sp)     # Tra lai gia tri cho cac thanh ghi
        lw      $a1, 4($sp)
        lw      $a2, 0($sp)
        addi    $sp, $sp, 12
        jr      $ra
#----------------------------------------

getArray:
        mul     $t0, $s0, $a2   # Vi tri cua phan tu = i * so cot + j
        add     $t0, $t0, $s1
        sll     $t0, $t0, 2     # Do phan tu co kieu word nen phai * 4 de ra khoang cach
dia chi tuong doi so voi dia chi dau
        add     $t0, $t0, $a0   # Cong dia chi dau de ra dia chi phan tu
        lw      $v0, 0($t0)     # Lay gia tri phan tu
        jr      $ra
#----------------------------------------

setArray:
        mul     $t0, $s0, $a2   # Vi tri cua phan tu = i * so cot + j
        add     $t0, $t0, $s1
        sll     $t0, $t0, 2     # Do phan tu co kieu word nen phai * 4 de ra khoang cach
dia chi tuong doi so voi dia chi dau
        add     $t0, $t0, $a0   # Cong dia chi dau de ra dia chi phan tu
        sw      $v0, 0($t0)     # Dat gia tri phan tu
        jr      $ra
```

Phân tích code và kết quả:

```
SysInitMem:
        la      $t9, Sys_TheTopOfFree   # Lay con tro chua dau tien con trong, khoi tao
        la      $t7, Sys_MyFreeSpace    # Lay dia chi dau tien con trong, khoi tao
        sw      $t7, 0($t9)             # Luu lai
        jr      $ra
#----------------------------------------
```

Hàm SysInitMem, bắt đầu gắn địa chỉ còn trống đầu tiên vào t7, sau đó lưu địa chỉ đó vào bộ nhớ của t9.

1. Sửa lỗi không chia hết cho 4 của kiểu word trong ví dụ:

```
#1. Sua loi bo nho cua vi du
case_1:
        bne     $v0, 1, case_2
        li      $v0, 4
        la      $a0, mess0.1
        syscall
        li      $v0, 5 #Nhap so phan tu cua day 1 chieu
        syscall
        bltz    $v0, error #Kiem tra so da nhap vao, neu v0 < 0 thi bao loi va yeu cau nhap lai
        move    $a1, $v0 #Luu so phan tu vao a1
        li      $v0, 4
        la      $a0, mess0.2
        syscall
        li      $v0, 5 #Nhap kich thuc moi phan tu cua day
        syscall
is1:    beq     $v0, 1, ready
is4:    beq     $v0, 4, ready
        j       error   #Kiem tra kich thuoc nhap vao, neu khong phai 1 hay 4 thi bao loi va yeu cau nhap lai
ready:  move    $a2, $v0 #Luu kich thuoc moi phan tu vao a2
        la      $a0, ArrayPtr #Luu dia chi bat dau cua chuoi mot chieu
        jal     malloc #Chay ham malloc cho chuoi 1 chieu
```

Kiểm tra số phần tử và kích thước phần tử của dãy, nếu không thỏa mãn số phần tử lớn không hay kích thước phần tử bằng 1 hoặc 4 thì báo lỗi, còn không thì di chuyển đến hàm malloc.

```
malloc:
        la      $t9, Sys_TheTop0fFree
        lw      $t8, 0($t9)             # Lay dia chi dau tien con trong
        bne     $a2, 4, initialize      # Neu mang khoi tao co kieu Word, kiem tra dia chi dau co dam bao quy tac khong
        andi    $t0, $t8, 0x03          # Lay so du khi chia dia chi trong cho 4
        beq     $t0, 0, initialize      # Neu khong co du, bo qua
        addi    $t8, $t8, 4             # Neu co, tien toi dia chi chia het cho 4 tiep theo
        subu    $t8, $t8, $t0
initialize:
        sw      $t8, 0($a0)     # Cat dia chi do vao bien con tro
        move    $v0, $t8        # 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 dia chi dau tien do vao bien Sys_TheTop0fFree
        jr      $ra
```

Hàm malloc, trước hết gán địa chỉ bắt đầu của bộ nhớ còn trống vào biến con trỏ t8. Nếu mảng khởi tạo là kiểu word (kích thước phần tử bằng 4 byte) thì kiểm tra xem con trỏ có thỏa mãn quy tắc. Đặt t0 là số dư của con trỏ cho 4. Nếu không dư thì bỏ qua còn nếu có thì tiến tới cho đến khi địa chỉ chia hết cho 4.
Lưu địa chỉ đã qua kiểm tra vào biến con trỏ. Tính kích thước của mảng cần cập nhật rồi khởi tạo lại địa chỉ bắt đầu của bộ nhớ còn trống rồi quay về.

```
        move    $t0, $v0 #Dat dia chi bat dau tra tu malloc vao t0
        li      $v0, 4
        la      $a0, mess0.3
        syscall
        move    $a0, $t0 #Dat a0 = t0
        li      $t0, 0 #Dat t0 = 0
input_loop:
        beq     $t0, $a1, input_end #Bat dau vong lap nhap du lieu, ket thuc khi t0 = a1
        li      $v0, 5
        syscall
        bne     $a2, 1, byte_4
byte_1:
        sb      $v0, 0($a0)
        addi    $a0, $a0, 1 #Neu kich thuoc phan tu bang 1 thi con tro tien 1 don vi
        addi    $t0, $t0, 1
        j       input_loop
byte_4:
        sw      $v0, 0($a0)
        addi    $a0, $a0, 4 #Neu kich thuoc phan tu bang 4 thi con tro tien 4 don vi
        addi    $t0, $t0, 1
        j       input_loop
input_end:
```

Nhập dữ liệu, với kích thước 1 byte, con trỏ tiến 1 đơn vị mỗi vòng lặp, với kích thước 4 byte, con trỏ tiến 4 đơn vị mỗi vòng lặp.

2. + 3. Viết hàm lấy giá trị của biến con trỏ + Viết hàm lấy địa chỉ biến con trỏ.

```
#2. Gia tri cua con tro
        li      $v0, 4
        la      $a0, mess1.1
        syscall
        la      $a0, ArrayPtr
        jal     getValue
        move    $a0, $v0
        li      $v0, 34
        syscall
#3. Dia chi cua con tro
        li      $v0, 4
        la      $a0, mess1.2
        syscall
        la      $a0, ArrayPtr
        jal     getAddress
        move    $a0, $v0
        li      $v0, 34
        syscall

        j       menu
getValue:
        lw      $v0, 0($a0)     # Lay gia tri cua bien con tro trong o nho co dia chi luu trong $a0
        jr      $ra
#---------------------------------------

getAddress:
        add     $v0, $0, $a0    # Lay dia chi tu $a0
        jr      $ra
#---------------------------------------
```

Hai hàm getValue và getAddress lần lượt lấy giá trị lưu trong a0 và địa chỉ của a0.

Kết quả:

```
1. Mang mot chieu
2. Sao chep mang ky tu
3. Mang hai chieu
4. Giai phong bo nho
5. Hien thi bo nho va ket thuc chuong trinh
6. Ket thuc chuong trinh
Lua chon: 1
So phan tu: 3
So byte moi phan tu (1 hoac 4): 1
Nhap phan tu:
2
5
7
Gia tri cua con tro: 0x90000004
Dia chi cua con tro: 0x10010008
```

4. Viết hàm thực hiện copy 2 con trỏ xâu kí tự.

```
#4. Viết hàm thực hiện copy 2 con trỏ xâu kí tự.
case_2:
        bne     $v0, 2, case_3
        li      $v0, 4
        la      $a0, mess2.1
        syscall
        li      $v0, 5 #Nhap vao so ky tu toi da cua chuỗi
        syscall
        move    $a1, $v0 #Luu so ky tu toi da vao a1
        li      $a2, 1 #Dat a2 = 1
        la      $a0, CharPtr1 #Dat dia chi cua chuoi 1 vao a0 va goi malloc
        jal     malloc
        move    $s0, $v0 #Dat s0 lam bien con tro cua chuoi 1
        la      $a0, CharPtr2#Dat dia chi cua chuoi 2 vao a0 va goi malloc
        jal     malloc
        move    $s1, $v0 #Dat s0 lam bien con tro cua chuoi 2
        li      $v0, 4
        la      $a0, mess2.2
        syscall
        move    $a0, $s0 #Nhap vao chuoi thu nhat
        li      $v0, 8
        syscall
        move    $a1, $s1 #Dat a1 la bien con tro cua chuỗi 2.
        jal     strcpy
```

Trước hết, nhập vào số ký tự tối đa của các chuỗi, lưu vào trong a1. Đặt a2 = 1. Cấp phát bộ nhớ cho 2 chuỗi bằng hàm malloc sau đó đặt lại a0 là biến con trỏ chuỗi 1 rồi nhập vào chuỗi 1 và đặt lại a1 là biến con trỏ chuỗi 2.

```
strcpy:
        add     $t0, $0, $a0    # Khoi tao $t0 o dau xau ky tu nguon
        add     $t1, $0, $a1    # Khoi tao $t1 o dau xau ky tu dich
        addi    $t2, $0, 1      # Khoi tao $t2 la ky tu khac '\0' de chay vong lap
cpyLoop:
        beq     $t2, 0, cpyLoopEnd      # Neu ky tu duoc copy trong vong lap truoc la '\0', dung vong lap
        lb      $t2, 0($t0)             # Doc ky tu o xau ky tu nguon
        sb      $t2, 0($t1)             # Luu ky tu vua doc vao xau ky tu dich
        addi    $t0, $t0, 1             # Chuyen $t0 tro sang vi tri cua phan tu tiep theo trong xau ky tu nguon
        addi    $t1, $t1, 1             # Chuyen $t1 tro sang vi tri cua phan tu tiep theo trong xau ky tu dich
        j       cpyLoop
cpyLoopEnd:
        jr      $ra
```

Hàm strcpy chạy vòng lặp để sao chép chuỗi 1 sang chuỗi 2 thông qua vòng lặp.

```
1. Mang mot chieu
2. Sao chep mang ky tu
3. Mang hai chieu
4. Giai phong bo nho
5. Hien thi bo nho
6. Ket thuc chuong trinh
Lua chon: 2
So ky tu toi da: 4

Nhap chuoi ky tu: str
Chuoi ky tu duoc copy: str
```

7. Malloc 2:

```
#7. Viet ham malloc 2:
case_3:
        bne     $v0, 3, case_4
        li      $v0, 4
        la      $a0, mess3.1
        syscall
        li      $v0, 5 #Nhap vao so hang
        syscall
        move    $a1, $v0
        li      $v0, 4
        la      $a0, mess3.2
        syscall
        li      $v0, 5 #Nhap vao so cot
        syscall
        move    $a2, $v0
        la      $a0, Array2Ptr #Luu vao a0 dia chi cua mang 2 chieu
        jal     malloc2
```

Nhập vào số hàng và số cột của mảng 2 chiều, được lưu ở a1 và a2.

```
malloc2:
        addi    $sp, $sp, -12   # Luu cac gia tri can thiet de thuc hien 1 chuong trinh con malloc trong chuong trinh con na
        sw      $ra, 8($sp)
        sw      $a1, 4($sp)
        sw      $a2, 0($sp)
        mul     $a1, $a1, $a2   # $a1 = so phan tu = so hang * so cot
        addi    $a2, $0, 4      # $a2 = so byte cua 1 phan tu kieu word = 4
        jal     malloc          # Chuyen mang 2 chieu thanh mang 1 chieu, khoi tao
        lw      $ra, 8($sp)     # Tra lai gia tri cho cac thanh ghi
        lw      $a1, 4($sp)
        lw      $a2, 0($sp)
        addi    $sp, $sp, 12
        jr      $ra
```

Hàm malloc2, trước hết lưu lại giá trị của ra, số hàng, số cột để chạy được hàm con malloc. Số phần tử trong mang sẽ là hàng x cột, lưu tại a1. Coi kích thước của mỗi phần tử là 4 (kiểu word). Sử dụng hàm malloc với các thông số như trên, biến mảng 2 thành 1 chiều rồi trả lại các giá trị của thanh ghi.

Kết quả:

```
1. Mang mot chieu
2. Sao chep mang ky tu
3. Mang hai chieu
4. Giai phong bo nho
5. Hien thi bo nho
6. Ket thuc chuong trinh
Lua chon: 3

So hang: 3

So cot: 3
Nhap phan tu:
1
2
3
4
5
6
7
8
9
```

8.

getArray:

```
sub_case_1:
        bne     $v0, 1, sub_case_2
        li      $v0, 4
        la      $a0, mess3.01
        syscall
        li      $v0, 5 #Nhap so hang
        syscall
        move    $s0, $v0 #Luu vao s0
        li      $v0, 4
        la      $a0, mess3.02
        syscall
        li      $v0, 5 #Nhap so cot
        syscall
        move    $s1, $v0 #Luu vao s1
        la      $t0, Array2Ptr
        lw      $a0, 0($t0)
        jal     getArray
```

Nhập vào số hàng và số cột.

```
getArray:
        mul     $t0, $s0, $a2   # Vi tri cua phan tu = i * so cot + j
        add     $t0, $t0, $s1
        sll     $t0, $t0, 2     # Do phan tu co kieu word nen phai * 4 de ra khoang cach dia chi tuong doi so voi dia chi da
        add     $t0, $t0, $a0   # Cong dia chi dau de ra dia chi phan tu
        lw      $v0, 0($t0)     # Lay gia tri phan tu
        jr      $ra
#----------------------------------------
```

Hàm getArray, phần tử thứ A[i][j] của mảng 2 chiều là phần tử thứ A[i*số cột + j] của mảng 1 chiều. Do mỗi phần tử cách nhau 4 byte nên phải nhân 4 để tính khoảng cách từ vị trí đầu.
setArray:

```
        j       sub_menu
sub_case_2:
        bne     $v0, 2, sub_case_3
        li      $v0, 4
        la      $a0, mess3.01
        syscall
        li      $v0, 5
        syscall
        move    $s0, $v0
        li      $v0, 4
        la      $a0, mess3.02
        syscall
        li      $v0, 5
        syscall
        move    $s1, $v0
        move    $s2, $v0
        li      $v0, 4
        la      $a0, mess0.3
        syscall
        li      $v0, 5
        syscall
        la      $t0, Array2Ptr
        lw      $a0, 0($t0)
        jal     setArray
```

```
#-----------------------------------------

setArray:
        mul     $t0, $s0, $a2   # Vi tri cua phan tu = i * so cot + j
        add     $t0, $t0, $s1
        sll     $t0, $t0, 2     # Do phan tu co kieu word nen phai * 4 de ra khoang cach dia chi tuong doi so voi dia chi da
        add     $t0, $t0, $a0   # Cong dia chi dau de ra dia chi phan tu
        sw      $v0, 0($t0)     # Dat gia tri phan tu
        jr      $ra
```

Dùng cách tương tự với getArray để tìm được địa chỉ muốn tìm, sau đó thay vì in ra, ta lưu giá trị mới nhập vào.
Kết quả:

```
4
5
6
7
8
9

1. getArray[i][j]
2. setArray[i][j]
3. Thoat
Lua chon: 2
i = 2
j = 2
Nhap phan tu:
12

1. getArray[i][j]
2. setArray[i][j]
3. Thoat
Lua chon: 1
i = 2
j = 2

Gia tri cua phan tu: 12
1. getArray[i][j]
2. setArray[i][j]
3. Thoat
Lua chon:
```

6) Tính lượng bộ nhớ đã cấp phát

```
#6. Viết hàm tính toàn bộ lượng bộ nhớ đã cấp phát.
case_5:

        bne     $v0, 5, case_6
        li      $v0, 4
        la      $a0, mess1.3
        syscall
        jal     memoryCalculate
        move    $a0, $v0
        li      $v0, 1
        syscall
        j       menu

memoryCalculate:
        la      $t0, Sys_MyFreeSpace    # Lay dia chi dau tien duoc cap phat
        la      $t1, Sys_TheTopOfFree   # Lay dia chi luu dia chi dau tien con trong
        lw      $t2, 0($t1)             # Lay dia chi dau tien con trong
        sub     $v0, $t2, $t0           # Tru hai dia chi cho nhau
        jr      $ra
```

Ta dùng địa chỉ của vị trí trống đầu mới nhất trừ đi vị trí trống đầu ban đầu.
Kết quả:

```
3. Mang hai chieu
4. Giai phong bo nho
5. Hien thi bo nho va ket thuc chuong trinh
6. Ket thuc chuong trinh
Lua chon: 1
So phan tu: 4
So byte moi phan tu (1 hoac 4): 4
Nhap phan tu:
1
5
2
3
Gia tri cua con tro: 0x90000008
Dia chi cua con tro: 0x10010008

1. Mang mot chieu
2. Sao chep mang ky tu
3. Mang hai chieu
4. Giai phong bo nho
5. Hien thi bo nho va ket thuc chuong trinh
6. Ket thuc chuong trinh
Lua chon: 5

Tong bo nho da cap phat: 20
```

5. Giải phóng bộ nhớ:

```
#5. Giải phóng bộ nhớ
case_4:
        bne     $v0, 4, case_5
        jal     free
        li      $v0, 4
        la      $a0, mess4.1
        syscall
        li      $v0, 4
        la      $a0, mess1.3
        syscall
        jal     memoryCalculate
        move    $a0, $v0
        li      $v0, 1
        syscall
        j       menu

free:
       addi    $sp, $sp, -4    # Khoi tao 1 vi tri trong stack
       sw      $ra, 0($sp)     # Luu $ra vao stack
       jal     SysInitMem      # Tai lap lai vi tri cua con tro luu dia chi dau tien con trong
       lw      $ra, 0($sp)     # Tra gia tri cho $ra
       addi    $sp, $sp, 4     # Xoa stack
#---------------------------------------
```

Chạy hàm SysInitMem bên trong để tái lập vị trí của con trỏ lưu địa chỉ đầu tiên còn trống.

Kết quả:

```
1. Mang mot chieu
2. Sao chep mang ky tu
3. Mang hai chieu
4. Giai phong bo nho
5. Hien thi bo nho
6. Ket thuc chuong trinh
Lua chon: 2
So ky tu toi da: 4

Nhap chuoi ky tu: str
Chuoi ky tu duoc copy: str

1. Mang mot chieu
2. Sao chep mang ky tu
3. Mang hai chieu
4. Giai phong bo nho
5. Hien thi bo nho
6. Ket thuc chuong trinh
Lua chon: 4
Da giai phong toan bo bo nho cap phat.

Tong bo nho da cap phat: 0

1. Mang mot chieu
2. Sao chep mang ky tu
3. Mang hai chieu
4. Giai phong bo nho
5. Hien thi bo nho
6. Ket thuc chuong trinh
Lua chon:
```

Assignment 8:

Đề bài:

8. Mô phỏng ổ đĩa RAID 5

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

Source code:

```
.data
        start: .asciiz "Nhap chuoi ky tu : "
        hex: .byte '0','1','2','3','4','5','6','7','8','9','a','b','c','d','e','f'
        d1: .space 4
        d2: .space 4
        d3: .space 4
        array: .space 32
        string: .space 5000
        enter: .asciiz "\n"
        error_length: .asciiz "Do dai chuoi khong chia het cho 8! Nhap lai.\n"
        m: .asciiz "      Disk 1              Disk 2              Disk 3\n"
        m2: .asciiz "----------------      ----------------      ----------------\n"
        m3: .asciiz "|     "
        m4: .asciiz "     |       "
        m5: .asciiz "[[ "
        m6: .asciiz "]]       "
        comma: .asciiz ","
        ms: .asciiz "Try again?"

.text
        la      $s1, d1                 # Tuong ung disk 1
        la      $s2, d2                 # Tuong ung disk 2
        la      $s3, d3                 # Tuong ung disk 3
        la      $a2, array                      # dia chi mang chua parity

input:  li      $v0, 4                          # nhap ten (chuoi)
        la      $a0, start
        syscall
        li      $v0, 8
        la      $a0, string
        li      $a1, 1000
        syscall
        move    $s0, $a0                        # s0 chua dia chi xau moi nhap
        li      $v0, 4
        la      $a0, m
        syscall
        li      $v0, 4
        la      $a0, m2
        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:
        add $t1, $s0, $t0               # t1 = address of string[i]
        lb      $t2, 0($t1)                     # t2 = string[i]
        nop
        beq     $t2, 10, test_length    # t2 = '\n' ket thuc xau
        nop
        addi    $t3, $t3, 1                     # length++
        addi    $t0, $t0, 1                     # index++
        j       check_char
        nop
test_length: move $t5, $t3
        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       split1
test1:  beq     $t1, 8, split1
        j       error1
error1:li       $v0, 4
        la      $a0, error_length
        syscall
        j       input
#------------------------------ket thuc kiem tra do dai---------------------------------

#-------------------------lay parity----------------------------
HEX:  li        $t4, 7
loopH: blt      $t4, $0, endloopH
        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
        add     $t7, $t7, $a0
        bgt     $t4, 1, nextc
        lb      $a0, 0($t7)                     # print hex[a0]
        li      $v0, 11
        syscall
        # lb $t6, 0($t7)
        # beq $t6, 48, in
nextc:  addi    $t4,$t4,-1
        j       loopH
# in:   bgt $t4, 1, loopH
        # move $a0, $t6
        # li $v0, 11
        # syscall
endloopH: jr    $ra
```

```
#-------------------------------------------------------

#-----------------------------Mo phong RAID 5-----------------------------
# xet 6 khoi dau
# lan 1: luu vao 2 khoi 1,2; xor vao 3-------
split1:
        addi    $t0, $zero, 0                   # so byte duoc in ra (4 byte)
        addi    $t9, $zero, 0
        addi    $t8, $zero, 0
        la      $s1, d1
        la      $s2, d2
        la      $a2, array
print11:li      $v0, 4
        la      $a0, m3
        syscall
b11:    lb      $t1, ($s0)              #B?t ??u vòng l?p
        addi    $t3, $t3, -1
        sb      $t1, ($s1)
b21:    add     $s5, $s0, 4
        lb      $t2, ($s5)                      # t2 chua dia chi tung byte cua dick 2
        addi    $t3, $t3, -1
        sb      $t2, ($s2)
b31:    xor     $a3, $t1, $t2
        sw      $a3, ($a2)
        addi    $a2, $a2, 4
        addi    $t0, $t0, 1
        addi    $s0, $s0, 1
        addi    $s1, $s1, 1
        addi    $s2, $s2, 1
        bgt     $t0, 3, reset
        j       b11
reset:  la      $s1, d1
        la      $s2, d2
print12:lb      $a0, ($s1)
        li      $v0, 11
        syscall
        addi    $t9, $t9, 1
        addi    $s1, $s1, 1
        bgt     $t9, 3, next11
        j       print12
next11:li       $v0, 4
        la      $a0, m4
        syscall
        li      $v0, 4
        la      $a0, m3
        syscall
```

```
print13:lb      $a0, ($s2)
        li      $v0, 11
        syscall
        addi    $t8, $t8, 1
        addi    $s2, $s2, 1
        bgt     $t8, 3, next12
        j       print13
next12:li       $v0, 4
        la      $a0, m4
        syscall
        li      $v0, 4
        la      $a0, m5
        syscall

        la      $a2, array
        addi    $t9, $zero, 0
print14:lb      $t8, ($a2)
        jal     HEX
        li      $v0, 4
        la      $a0, comma
        syscall
        addi    $t9, $t9, 1
        addi    $a2, $a2, 4
        bgt     $t9, 2, end1                    # in ra 3 parity dau co dau ",", parity cuoi
cung k co
        j       print14
end1:   lb      $t8, ($a2)
        jal HEX
        li $v0, 4
        la $a0, m6
        syscall
        li $v0, 4
        la $a0, enter
        syscall
        beq $t3, 0, exit1
#-------------------------------------------------------------
split2: la $a2, array
        la $s1, d1
        la $s3, d3
        addi $s0, $s0, 4
        addi $t0, $zero, 0
print21:li $v0, 4
        la $a0, m3
        syscall
b12:    lb $t1, ($s0)
        addi $t3, $t3, -1
```

```
        sb $t1, ($s1)
b32:    add $s5, $s0, 4
        lb $t2, ($s5)
        addi $t3, $t3, -1
        sb $t2, ($s3)
b22:    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 b12
reset2: la $s1, d1
        la $s3, d3
        addi $t9, $zero, 0
print22:lb $a0, ($s1)
        li $v0, 11
        syscall
        addi $t9, $t9, 1
        addi $s1, $s1, 1
        bgt $t9, 3, next21
        j print22
next21:li $v0, 4
        la $a0, m4
        syscall
        la $a2, array
        addi $t9, $zero, 0
        li $v0, 4
        la $a0, m5
        syscall
print23:lb $t8, ($a2)
        jal HEX
        li $v0, 4
        la $a0, comma
        syscall
        addi $t9, $t9, 1
        addi $a2, $a2, 4
        bgt $t9, 2, next22
        j print23
next22:lb $t8, ($a2)
        jal HEX
        li $v0, 4
        la $a0, m6
        syscall
```

```
        li $v0, 4
        la $a0, m3
        syscall
        addi $t8, $zero, 0
print24:lb $a0, ($s3)
        li $v0, 11
        syscall
        addi $t8, $t8, 1
        addi $s3, $s3, 1
        bgt $t8, 3, end2
        j print24

end2:   li $v0, 4
        la $a0, m4
        syscall
        li $v0, 4
        la $a0, enter
        syscall
        beq $t3, 0, exit1
#-------------------------------------------------------------
split3:la $a2, array
        la      $s2, d2
        la      $s3, d3
        addi    $s0, $s0, 4
        addi    $t0, $zero, 0
print31:li      $v0, 4
        la      $a0, m5
        syscall
b23:    lb      $t1, ($s0)
        addi    $t3, $t3, -1
        sb      $t1, ($s1)
b33:    add     $s5, $s0, 4
        lb      $t2, ($s5)
        addi    $t3, $t3, -1
        sb      $t2, ($s3)
b13:    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, reset3
        j       b23
reset3:la       $s2, d2
        la      $s3, d3
```

```
        la      $a2, array
        addi    $t9, $zero, 0
print32:lb      $t8, ($a2)
        jal     HEX
        li      $v0, 4
        la      $a0, comma
        syscall
        addi    $t9, $t9, 1
        addi    $a2, $a2, 4
        bgt     $t9, 2, next31
        j       print32
next31:lb       $t8, ($a2)
        jal HEX
        li      $v0, 4
        la      $a0, m6
        syscall
        li      $v0, 4
        la      $a0, m3
        syscall
        addi    $t9, $zero, 0
print33:lb      $a0, ($s2)
        li      $v0, 11
        syscall
        addi    $t9, $t9, 1
        addi    $s2, $s2, 1
        bgt     $t9, 3, next32
        j       print33
next32:addi     $t9, $zero, 0
        addi    $t8, $zero, 0
        li      $v0, 4
        la      $a0, m4
        syscall
        li      $v0, 4
        la      $a0, m3
        syscall
print34:lb      $a0, ($s3)
        li      $v0, 11
        syscall
        addi    $t8, $t8, 1
        addi    $s3, $s3, 1
        bgt     $t8, 3, end3
        j       print34

end3:   li      $v0, 4
        la      $a0, m4
        syscall
```

```
        li      $v0, 4
        la      $a0, enter
        syscall
        beq     $t3, 0, exit1
#-------------------------------end 6 khoi dau-------------------------------------
# chuyen sang 6 khoi tiep theo
nextloop: addi $s0, $s0, 4
        j split1

exit1:  li      $v0, 4
        la      $a0, m2
        syscall
        j       ask
#-------------------------------ket thuc mo phong RAID 5---------------------------

#-------------------------------------try again-------------------------------------
ask:    li      $v0, 50
        la      $a0, ms
        syscall
        beq     $a0, 0, clear
        nop
        j       exit
        nop
# clear: dua string ve trang thai ban dau de thuc hien lai qua trinh
clear:  la      $s0, string
        add     $s3, $s0, $t5   # s3: dia chi byte cuoi cung duoc su dung trong string
        li      $t1, 0
goAgain: 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
```

Phân tích code và kết quả:

```
.text
        la      $s1, d1                 # Tuong ung disk 1
        la      $s2, d2                 # Tuong ung disk 2
        la      $s3, d3                 # Tuong ung disk 3
        la      $a2, array                      # dia chi mang chua parity
```

Khởi tạo đĩa 1, đĩa 2, đĩa 3 và mảng chứa parity vào s1, s2, s3 và a2

```
input:  li      $v0, 4                          # nhap ten (chuoi)
        la      $a0, start
        syscall
        li      $v0, 8
        la      $a0, string
        li      $a1, 1000
        syscall
        move    $s0, $a0                        # s0 chua dia chi xau moi nhap
        li      $v0, 4
        la      $a0, m
        syscall
        li      $v0, 4
        la      $a0, m2
        syscall
```

Nhập chuỗi vào. Lưu địa chỉ của chuỗi vào s0

```
#----------------------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: add $t1, $s0, $t0           # t1 = address of string[i]
        lb      $t2, 0($t1)                     # t2 = string[i]
        nop
        beq     $t2, 10, test_length    # t2 = '\n' ket thuc xau
        nop
        addi    $t3, $t3, 1                     # length++
        addi    $t0, $t0, 1                     # index++
        j       check_char
        nop
test_length: move $t5, $t3
        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       split1
test1:  beq     $t1, 8, split1
        j       error1
error1:li       $v0, 4
        la      $a0, error_length
```

Kiểm tra độ dài chuỗi mới nhập. Dùng check_char để tính độ dài của chuỗi qua vòng lặp rồi sau đó xóa hết các byte của t3 trừ byte cuối để kiểm tra xem t3 có bằng 0 hoặc 8 hay không. Nếu không thì báo lỗi và nhập lại.

```
#-----------------------------Mo phong RAID 5-------------------------------
# xet 6 khoi dau
# lan 1: luu vao 2 khoi 1,2; xor vao 3-------
split1:addi     $t0, $zero, 0                   # so byte duoc in ra (4 byte)
       addi     $t9, $zero, 0
       addi     $t8, $zero, 0
       la       $s1, d1
       la       $s2, d2
       la       $a2, array
print11:li      $v0, 4
       la       $a0, m3
       syscall
b11:   lb       $t1, ($s0)
       addi     $t3, $t3, -1
       sb       $t1, ($s1)
```

Khởi tạo t0, t9, t8, đĩa 1, 2 và mảng chứa parity.
Bắt đầu vòng lặp, lấy giá trị đầu tiên từ chuỗi rồi lưu vào đĩa 1.

```
b21:    add     $s5, $s0, 4
        lb      $t2, ($s5)                  # t2 chua dia chi tung byte cua dick 2
        addi    $t3, $t3, -1
        sb      $t2, ($s2)
b31:    xor     $a3, $t1, $t2
        sw      $a3, ($a2)
        addi    $a2, $a2, 4
        addi    $t0, $t0, 1
        addi    $s0, $s0, 1
        addi    $s1, $s1, 1
        addi    $s2, $s2, 1
        bgt     $t0, 3, reset
        j       b11
reset:  la      $s1, d1
        la      $s2, d2
```

Tiến con trỏ của chuỗi lên 4 đơn vị (do mỗi đĩa chứa 4 byte) và bắt đầu lưu giá trị tại vị chí đó vào chuỗi.
Tính giá trị của xor a3, t1, t2 sau đó lưu vào trong mảng chứa parity đồng thời tiến index lên 1 đơn vị. Lặp lại cho đến khi index = 3.

```
reset:  la      $s1, d1
        la      $s2, d2
print12:lb      $a0, ($s1)
        li      $v0, 11
        syscall
        addi    $t9, $t9, 1
        addi    $s1, $s1, 1
        bgt     $t9, 3, next11
        j       print12
next11:li       $v0, 4
        la      $a0, m4
        syscall
        li      $v0, 4
        la      $a0, m3
        syscall
```

Đặt lại s1 và s2 về đầu địa chỉ của đĩa. In ra màn hình các giá trị của đĩa 1.

```
print13:lb      $a0, ($s2)
        li      $v0, 11
        syscall
        addi    $t8, $t8, 1
        addi    $s2, $s2, 1
        bgt     $t8, 3, next12
        j       print13
next12:li       $v0, 4
        la      $a0, m4
        syscall
        li      $v0, 4
        la      $a0, m5
        syscall

        la      $a2, array
        addi    $t9, $zero, 0
```

Tương tự, in ra màn hình các giá trị của s2.

```
print14:lb      $t8, ($a2)
        jal     HEX
        li      $v0, 4
        la      $a0, comma
        syscall
        addi    $t9, $t9, 1
        addi    $a2, $a2, 4
        bgt     $t9, 2, end1                    # in ra 3 parity dau co dau ",", parity cuoi cung k co
        j       print14
end1:   lb      $t8, ($a2)
        jal HEX
        li $v0, 4
        la $a0, m6
        syscall
        li $v0, 4
        la $a0, enter
        syscall
        beq $t3, 0, exit1
#-------------------------------------------------------
```

```
HEX:    li      $t4, 7
loopH:  blt     $t4, $0, endloopH
        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
        add     $t7, $t7, $a0
        bgt     $t4, 1, nextc
        lb      $a0, 0($t7)                     # print hex[a0]
        li      $v0, 11
        syscall
        # lb $t6, 0($t7)
        # beq $t6, 48, in
nextc:  addi    $t4,$t4,-1
        j       loopH
# in:   bgt $t4, 1, loopH
        # move $a0, $t6
        # li $v0, 11
        # syscall
endloopH: jr    $ra
```

Sau đó, ta gọi hàm HEX để in ra kết quả của mảng chứa parity dưới dạng hex. Kiểm tra xem nếu toàn bộ chuỗi đã được in hết ra chưa. Rồi thì hỏi lại người dùng có muốn tiếp tục, nếu có thì đưa làm mới chuỗi nhập vào, không thì kết thúc.

```
exit1:  li      $v0, 4
        la      $a0, m2
        syscall
        j       ask
#------------------------------ket thuc mo phong RAID 5-----------------------------------

#---------------------------------------try again-----------------------------------------
ask:    li      $v0, 50
        la      $a0, ms
        syscall
        beq     $a0, 0, clear
        nop
        j       exit
        nop
# clear: dua string ve trang thai ban dau de thuc hien lai qua trinh
clear:  la      $s0, string
        add     $s3, $s0, $t5   # s3: dia chi byte cuoi cung duoc su dung trong string
        li      $t1, 0
goAgain: 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-------------------------------------------
```

```
#---------------------------------end try again-------------------------------------

exit:   li      $v0, 10
        syscall
```

Nếu chuỗi chưa kết thúc ta tiến hành tìm và in tiếp các khối sau. Ta làm tương tự như khối một nhưng ở khối 2 thì sẽ là khởi tạo đĩa 1, 3 còn khối 3 sẽ là đĩa 2, 3. Sau khi kết thúc khối thứ 3 mà chưa duyệt hết chuỗi thì ta quay lại chuỗi một.

Kết quả:

```
Nhap chuoi ky tu : 123
      Disk 1                  Disk 2                Disk 3
----------------        ----------------        ----------------
Do dai chuoi khong chia het cho 8! Nhap lai.
Nhap chuoi ky tu :
```

```
Nhap chuoi ky tu : 123
      Disk 1                  Disk 2                Disk 3
----------------        ----------------        ----------------
Do dai chuoi khong chia het cho 8! Nhap lai.
Nhap chuoi ky 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      |
----------------        ----------------        ----------------
```


```
Nhap chuoi ky tu : DCE.****ABCD1234HUSTHUSTfirelewi
      Disk 1                  Disk 2                Disk 3
----------------        ----------------        ----------------
|     DCE.      |       |     ****      |       [[ 6e,69,6f,04]]
|     ABCD      |       [[ 70,70,70,70]]        |     1234      |
[[ 00,00,00,00]]        |     HUST      |       |     HUST      |
|     fire      |       |     lewi      |       [[ 0a,0c,05,0c]]
----------------        ----------------        ----------------

-- program is finished running --
```