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

BÀI TẬP LỚN

MÔN: Kiến trúc máy tính

ĐỀ TÀI: **2 và 6**

Giảng viên hướng dẫn: ThS. Lê Bá Vui

Nhóm sinh viên thực hiện:

1. Nguyễn Huy Linh – 20194604
2. Vũ Tuấn Kiệt – 20194599

Hà Nội, tháng 7 năm 2022

Bài số 2: Vẽ hình trên màn hình Bitmap

Vẽ hình quả bóng hình tròn di chuyển trên màn hình mô phỏng Bitmap của Mars. Nếu đối tượng đập vào cạnh của màn hình thì sẽ di chuyển ngược lại.

Yêu cầu:

- Màn hình ở kích thước 512x512, kích thước pixel 1x1.
- Di chuyển trái (A), di chuyển phải (D), di chuyển lên (W), di chuyển xuống (S), tăng tốc độ (Z), giảm tốc độ (X).
- Vị trí bóng ban đầu ở giữa màn hình.

Bài làm

Phân tích cách làm:

- Ta sẽ vẽ hình tròn thông qua tọa độ tâm đường tròn và chiều dài bán kính cho trước.
- Đầu tiên ta sẽ thực hiện vẽ hình tròn từ trên xuống dưới từ phải sang trái theo chiều kim đồng hồ.
- Với việc di chuyển quả bóng thì ta đầu tiên sẽ tô lại quả bóng với màu của màn hình, sau đó cập nhật lại tọa độ của tâm rồi vẽ lại quả bóng với tâm mới(bán kính vẫn như cũ).
- Di chuyển quả bóng sang trái là dịch tâm sang trái một khoảng d.
- Di chuyển quả bóng sang phải là dịch tâm sang phải một khoảng d.
- Di chuyển quả bóng lên trên là dịch tâm lên trên một khoảng d.
- Di chuyển quả bóng xuống dưới là dịch tâm xuống dưới một khoảng d.
- Khoảng cách dịch tâm d chính là tốc độ. d càng lớn độ dịch càng cao, tốc độ càng cao và ngược lại.
- Tăng tốc độ bằng cách tăng d, giảm tốc độ bằng cách giảm d.
- Khi dịch trái ta sẽ kiểm tra xem quả bóng có chạm vào cạnh màn hình hay không. Nếu có thì dịch phải.
- Khi dịch phải ta sẽ kiểm tra xem quả bóng có chạm vào cạnh màn hình hay không. Nếu có thì dịch trái.
- Khi dịch lên ta sẽ kiểm tra xem quả bóng có chạm vào cạnh màn hình hay không. Nếu có thì dịch xuống.
- Khi dịch xuống ta sẽ kiểm tra xem quả bóng có chạm vào cạnh màn hình hay không. Nếu có thì dịch lên.

Phân tích thuật toán vẽ hình:

- Tọa độ tâm đường tròn O(Rx, Ry), bán kính R=32 cho trước.
- Phương trình đường tròn (x-Rx)^2+(y-Ry)^2=R^2. Với góc phần tư thứ nhất và thứ tư x=Rx+sqrt(R^2-(y-Ry)^2). Với góc phần tư thứ hai và ba x=Rx- sqrt(R^2-(y-Ry)^2). x và y lấy số nguyên.
- Để truy cập tới một vị trí trên Bitmap với tọa độ (x,y) cho trước bằng cách (y*512+x)*4(vì kích cỡ màn hình đặt là 512x512).
- Vị trí trung tâm màn hình ban đầu quả bóng xuất hiện là (256,256).
- Để vẽ hình tròn ta chia làm 4 phase vẽ lần lượt từ góc phần tư thứ tư, thứ nhất thứ hai và thứ ba(theo chiều kim đồng hồ). Khởi tạo tọa độ ban đầu là (x,y)=(Rx,Ry-32)(đỉnh đường tròn).
- Với phase 1 ta sẽ lần lượt tô màu các pixel với tọa độ (x,y) (x được xác định qua y thông qua biểu thức ở trên). y lần lượt tăng lên 1, dừng khi y bằng Ry.
- Phase 2 tương tự phase 1 nhưng dừng khi y bằng Ry+32.
- Phase 3 tương tự nhưng dừng khi y bằng Ry.
- Phase 4 tương tự nhưng dừng khi y bằng Ry-32.

Phân tích thuật toán di chuyển:

- Ta sẽ đọc keycode từ bàn phím bằng Keyboard and Display MMIO Simulator.

- Với keycode = w = 119 ta sẽ di chuyển quả bóng lên trên bằng cách giảm Ry của tâm quả bóng xuống một khoảng d. Ta kiểm tra nếu Ry < 32 (chạm cạnh trên màn hình) thì di chuyển xuống.
- Với keycode = s = 115 ta sẽ di chuyển quả bóng đi xuống bằng cách tăng Ry của tâm quả bóng lên một khoảng d. Ta kiểm tra nếu Ry > 480(chạm cạnh dưới màn hình) thì di chuyển lên.
- Với keycode = a = 97 ta sẽ di chuyển quả bóng sang trái bằng cách giảm Rx của tâm quả bóng xuống một khoảng d. Ta kiểm tra Rx < 32(chạm cạnh trái màn hình) thì di chuyển phải.
- Với keycode = d = 100 ta sẽ di chuyển quả bóng sang phải bằng cách tăng Rx của tâm quả bóng lên một khoảng d. Ta kiểm tra Rx > 480(chạm cạnh phải màn hình) thì di chuyển trái.
- Ban đầu khởi tạo tốc độ d = 1.
- Với keycode = z = 122 tăng tốc độ d = d*2. Nếu d > 8 đặt d = 8.
- Với keycode = x = 120 giảm tốc độ d= d/2. Nếu d < 1 đặt d = 1.

Mã nguồn:

```
.eqv    MONITOR_SCREEN          0x10010000
.eqv    RED             0x00FF0000
.eqv    BLACK                   0x00000000

.eqv    KEY_CODE   0xFFFF0004   # ASCII code from keyboard, 1 byte
.text
main:
        li $k0, MONITOR_SCREEN                  # load screen
        li $t0, RED                             # load red color
        li $a2, KEY_CODE
        li $s0, 256                             # init x in center screen: 256
        li $s1, 256                             # init y in center screen: 256
        li $s4, 1                               # distance move <=> speed
        jal print                               # display circle
        j moving                                # start moving
exit:
        li $v0, 10                              # exit program
        syscall
endmain:

print:
        sw $ra, 0($sp)
        add $t1, $s0, $0                        # point x = Rx
        subi $t2, $s1, 32                       # point y = Ry-32
        phase1:
                beq $t2, $s1, endphase1                 # if point y == y stop
                sll $t9, $t2, 9                 # y*512
                add $t9, $t9, $t1               # y*512+x
                sll $t9, $t9, 2                 # 4*(y*512+x)
                add $t9, $t9, $k0               # access to screen
                sw $t0, 0($t9)                  # print pixel

                addi $t2, $t2, 1                # point point y ++
                sub $t1, $t2, $s1               # point y-Ry
                mul $t1, $t1, $t1               # (point y-Ry)^2
                subi $t1, $t1, 1024             # (point y-Ry)^2-R^2
                sub $t1, $0, $t1                # R^2-(point y-Ry)^2
```

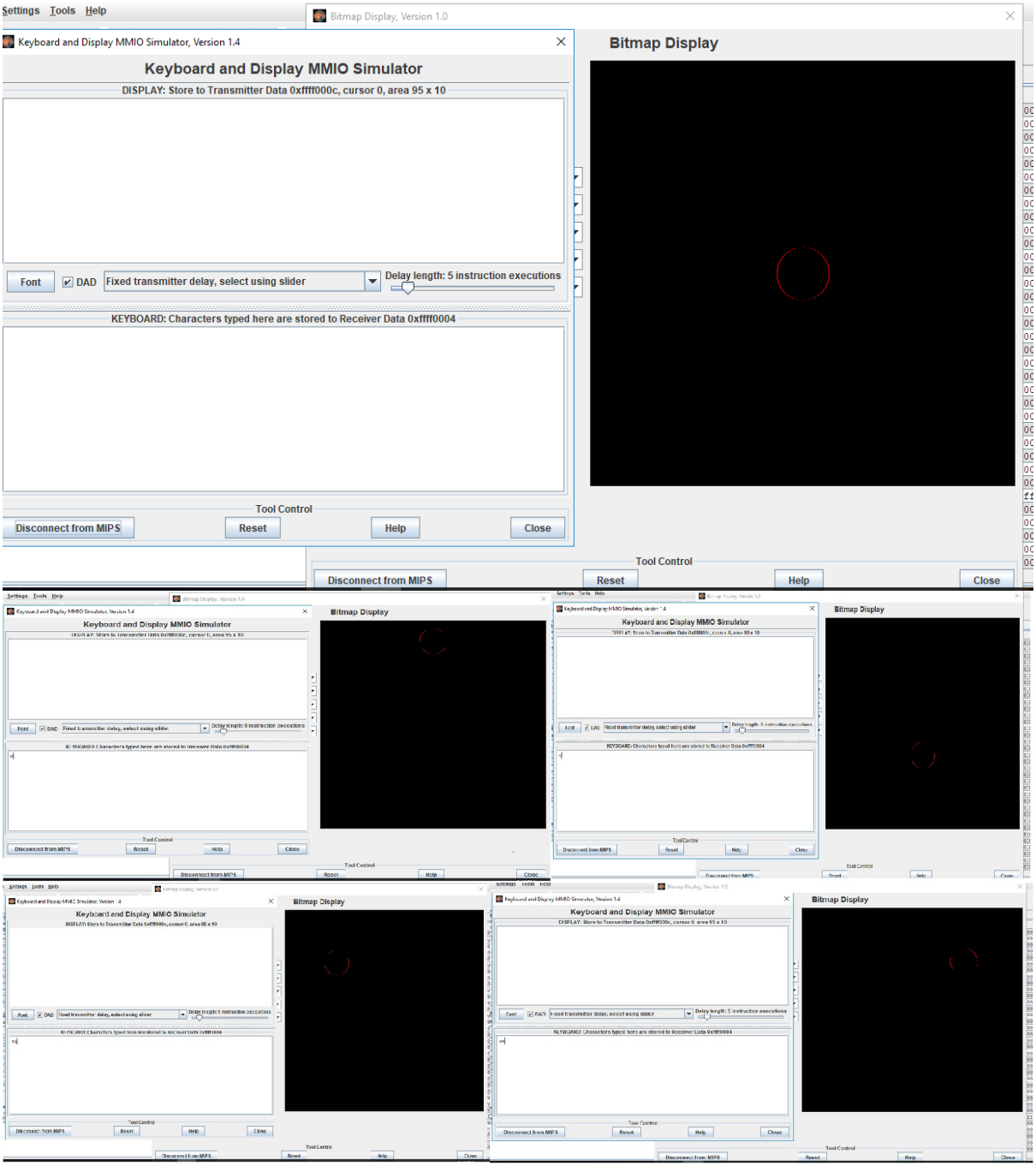
```
        jal square                  # square(R^2-(point y-Ry)^2)
        add $t1, $t1, $s0           # point x=square(R^2-(point y-Ry)^2)+Rx

        j phase1
endphase1:

addi $t3, $s1, 32                   # point to stop
phase2:
        beq $t2, $t3, endphase2             # if tmp x == x stop
        sll $t9, $t2, 9             # y*512
        add $t9, $t9, $t1           # y*512+x
        sll $t9, $t9, 2             # 4*(y*512+x)
        add $t9, $t9, $k0           # access to screen
        sw $t0, 0($t9)              # print pixel

        addi $t2, $t2, 1            # point y ++
        sub $t1, $t2, $s1           # point y-Ry
        mul $t1, $t1, $t1           # (point y-Ry)^2
        subi $t1, $t1, 1024         # (point y-Ry)^2-R^2
        sub $t1, $0, $t1            # R^2-(point y-Ry)^2
        jal square                  # square(R^2-(point y-Ry)^2)
        add $t1, $t1, $s0           # point x=square(R^2-(point y-Ry)^2)+Rx

        j phase2
endphase2:

phase3:
        beq $t2, $s1, endphase3             # if tmp y == y stop
        sll $t9, $t2, 9             # y*512
        add $t9, $t9, $t1           # y*512+x
        sll $t9, $t9, 2             # 4*(y*512+x)
        add $t9, $t9, $k0           # access to screen
        sw $t0, 0($t9)              # print pixel

        subi $t2, $t2, 1        # tmp y --
        sub $t1, $t2, $s1           # point y-Ry
        mul $t1, $t1, $t1           # (point y-Ry)^2
        subi $t1, $t1, 1024         # (point y-Ry)^2-R^2
        sub $t1, $0, $t1            # R^2-(point y-Ry)^2
        jal square                  # square(R^2-(point y-Ry)^2)
        sub $t1, $t1, $s0           # square(R^2-(point y-Ry)^2)-Rx
        sub $t1, $0, $t1            # point x=Rx-square(R^2-(point y-Ry)^2)

        j phase3
endphase3:


subi $t3, $s1, 32                   # point to stop
phase4:
```

```
                beq $t2, $t3, endphase4             # if point y == stop y => stop
                sll $t9, $t2, 9                 # y*512
                add $t9, $t9, $t1               # y*512+x
                sll $t9, $t9, 2                 # 4*(y*512+x)
                add $t9, $t9, $k0               # access to screen
                sw $t0, 0($t9)                  # print pixel
                subi $t2, $t2, 1        # tmp y --

                sub $t1, $t2, $s1               # point y-Ry
                mul $t1, $t1, $t1               # (point y-Ry)^2
                subi $t1, $t1, 1024             # (point y-Ry)^2-R^2
                sub $t1, $0, $t1                # R^2-(point y-Ry)^2
                jal square                      # square(R^2-(point y-Ry)^2)
                sub $t1, $t1, $s0               # square(R^2-(point y-Ry)^2)-Rx
                sub $t1, $0, $t1                # point x=Rx-square(R^2-(point y-Ry)^2)

                j phase4
        endphase4:
        lw $ra, 0($sp)
        jr $ra
endprint:

square:
        mtc1 $t1, $f0                   # move content to float register
        cvt.d.w $f0, $f0                # convert to float
        sqrt.d $f0, $f0                 # square root
        cvt.w.d $f0, $f0                # convert content to integer
        mfc1 $t1, $f0                   # move content float after casting to integer register
        jr $ra
endsquare:

moving:
        ReadKey:
                lw $s3, 0($a2)                  # load keycode from keyboard
                beq $s3, 119 ,UP                # w => up
                beq $s3, 115, DOWN              # s => down
                beq $s3, 97, LEFT               # a => left
                beq $s3, 100, RIGHT             # d => right
                beq $s3, 122, SPEEDUP                   # z => speed up
                beq $s3, 120, SPEEDDOWN                 # x => speed down
                beq $s3, 101, exit              # e => exit
                j ReadKey
        EndReadKey:

        UP:
                addi $s3, $0, 119               # set to up
                sw $s3, 0($a2)
                blt $s1, 32, DOWN               # rebound to down if meet edge
                li $t0, BLACK                   # print again circle with color of screen
```

```
        jal print
        li $a0, 0                       # load time
        li $v0, 32                      # delay
        syscall
        li $t0, RED                     # load color RED
        sub $s1, $s1, $s4               # update new location
        jal print                       # print new circle
        j ReadKey
ENDUP:

DOWN:
        addi $s3, $0, 115               # set to down
        sw $s3, 0($a2)
        bgt $s1, 480, UP                # rebound to up if meet edge
        li $t0, BLACK                   # print again circle with color of screen
        jal print
        li $a0, 0                       # load time
        li $v0, 32                      # delay
        syscall
        li $t0, RED                     # load color RED
        add $s1, $s1, $s4               # update new location
        jal print                       # print new circle
        j ReadKey
ENDDOWN:

LEFT:
        addi $s3, $0, 97                # set to left
        sw $s3, 0($a2)
        blt $s0, 32, RIGHT              # rebound to right if meet edge
        li $t0, BLACK                   # print again circle with color of screen
        jal print
        li $a0, 0                       # load time
        li $v0, 32                      # delay
        syscall
        li $t0, RED                     # load color RED
        sub $s0, $s0, $s4               # update new location
        jal print                       # print new circle
        j ReadKey
ENDLEFT:

RIGHT:
        addi $s3, $0, 100               # set to right
        sw $s3, 0($a2)
        bgt $s0, 480, LEFT              # rebound to left if meet edge
        li $t0, BLACK                   # print again circle with color of screen
        jal print
        li $a0, 0                       # load time
        li $v0, 32                      # delay
        syscall
```

```
            li $t0, RED                     # load color RED
            add $s0, $s0, $s4               # update new location
            jal print                       # print new circle
            j ReadKey
        ENDRIGHT:

        SPEEDUP:
            sw $0, 0($a2)
            sll $s4, $s4, 1                 # speed up x2
            bgt $s4, 8, UPDATEUP                    # speed max=8
            j ReadKey
        ENDSPEEDUP:

        UPDATEUP:
            addi $s4, $0, 8                 # set time delay=0
            j ReadKey
        ENDUPDATEUP:

        SPEEDDOWN:
            sw $0, 0($a2)
            srl $s4, $s4, 1                 # speed down x2
            blt $s4, 1, UPDATEDOWN                  # speed min=1
            j ReadKey
        ENDSPEEDDOWN:

        UPDATEDOWN:
            addi $s4, $0, 1             # set time delay=10
            j ReadKey
        ENDUPDATEDOWN:
endmoving:
```

Kết quả chạy mô phỏng:



Bài 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 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
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.

Bài làm

Phân tích cách làm:
1) Ta sửa lỗi bằng cách sửa hàm Malloc. Ta lấy địa chỉ từ còn trống từ Sys_TheTopOfFree, nếu thấy địa chỉ hiện tại không chia hết cho 4, ta tăng thêm 1 đơn vị. Đến khi địa chỉ chia hết cho 4, ta sử dụng địa chỉ đo cho biến word. Những địa chỉ này ta coi như cấp phát nhưng không sử dụng.

```
malloc:
        la      $t9, Sys_TheTopOfFree   #
        lw      $t8, 0($t9)     #Lay dia chi dau tien con trong
        #1) Word phai duoc cap phat o dia chi chia het cho 4
        #Dia chi kieu word chia het 4
        #while( diachi % 4 != 0)
        # diachi ++ => cap phat cho bien char them dia chi
checkWord:
        bne     $a2,4,endCheckWord
        nop
While:  div     $t8,$a2
        mfhi    $t0
        beqz    $t0,endCheckWord
        nop
        addi    $t8,$t8,1       # Dia chi + 1
        j       While
endCheckWord:
        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, $al,$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
```

2) Câu này ta đơn giản lấy giá trị với câu lệnh lw
3) Ta sẽ lấy được địa chỉ biến con trỏ tùy vào hàm được gọi.

```
#2) Viết hàm lấy giá trị của biến con trỏ.
#-----------------------------------------
# Ham lay gia gia tri cua bien con tro
# @param [in/out] $a0 Chua dia chi cua bien con tro
# @return $v0 gia tri cua bien con tro
#-----------------------------------------
PtrValue:       lw $v0,0($a0)
                jr  $ra
#3) Viết hàm lấy địa chỉ biến con trỏ.
#@return tra ve dia chi cua tung bien con tro
addressCharPtr:la       $v0,CharPtr
               jr       $ra
addressBytePtr:la       $v0,BytePtr
               jr       $ra
addressWordPtr:la       $v0,WordPtr
               jr       $ra
```

4) Truyền tham số: địa chỉ biến con trỏ gốc, số phần tử đã cấp phát, địa chỉ con trỏ cần copy(chưa cấp phát). Ta sẽ cấp phát cho con trỏ copy rồi thực hiện sao chép từng ký tự một từ con trỏ gốc vào con trỏ copy.

```
#4) Viết hàm thực hiện copy 2 con trỏ xâu kí tự.
#-----------------------------------------
# Ham lay gia gia tri cua bien con tro
# @param [in] $a0 Chua dia chi cua bien con tro xau^ goc'
# @param [in] $a1 Chua so phan tu cua con tro xau^ goc'
# @param [in/out] $a2 Chua dia chi con tro xau copy, khi ket thuc chua dia chi copy chuoi moi
# @return $v0 gia tri cua bien con tro
#-----------------------------------------
CopyPtr:
        sw      $ra,-4($sp)     #store the return address
        sw      $a0,-8($sp)     #store $a0 value
        sw      $a1,-12($sp)    #store $a1 value
        sw      $a2,-16($sp)    #store $a2 value
        addi    $sp,$sp,-16     #allocate space for $ra,$a0,$a1,$a2

        add     $a0,$0, $a2     # cap phat bo nho cho con tro copy
        add     $a1, $zero, $a1 # So phan tu cap phat cho string copy
        addi    $a2, $zero, 1   # so byte cho phan tu
        jal     malloc

        lw      $a2,0($sp)      #restore $a2
        lw      $a1,4($sp)      #restore $a1
        lw      $a0,8($sp)      #restore $a0
        lw      $ra,12($sp)     #restore $ra
        addi    $sp,$sp,16      #restore stack pointer

        lw      $t0,0($a0)      #lay dia chi da cap phat cua xau goc
        lw      $t1,0($a2)      #lay dia chi da cap phat cua xau copy

        li      $t2,0           #i = 0
LoopCopy:       slt     $t9,$t2,$a1     #i < so phan tu cua xau
                beqz    $t9,end_LoopCopy
                nop

                add     $t9,$t0,$t2     # $t9 = dia chi xau goc + i
                add     $t7,$t1,$t2     # $t7 = dia chi xau copy +i
                lb      $t3,0($t9)      #  t7[i] = t9[i]
                sb      $t3,0($t7)      #
                addi    $t2,$t2,1       # i ++
                j       LoopCopy
                nop
end_LoopCopy:
        jr      $ra
```

5) Hàm giải phóng con trỏ. Ta lấy Sys_MyFreeSpace đến địa chỉ Sys_TheTopOfFree + 3, tức là địa chỉ đầu tiên có thể cấp phát địa chỉ bộ nhớ động, cập nhật từng giá trị của nó là null. Đồng thời gán con trỏ là null và đặt lại giá trị cho Sys_TopOfFree.

```
#5) Viết hàm giải phóng bộ nhớ đã cấp phát cho các biến con trỏ.
FreeMem:
        la      $t9, Sys_TheTopOfFree   #Lay con tro chua dia chi dau tien con trong
        lw      $t7,0($t9)      #gia tri dia chi dau tien con trong
        addi    $t6,$t9,3       #mut' dia chi cuoi cung cua Sys_TheTopOfFre
        addi    $t7,$t7,-1      #lui 1 byte den dia chi cuoi cung duoc cap phat
CleanLoop:      slt     $t8,$t6,$t7 # chua den vi tri con trong ban dau duoc cap phat (kdata +4) thi tiep tuc
                beqz    $t8,End_CleanLoop
                nop
                sb      $0,0($t7)
                addi    $t7,$t7,-1 # lui 1 byte
                j       CleanLoop
End_CleanLoop:
                la      $t8,Sys_MyFreeSpace # dat lai gia tri dia chi con trong co the cap phat
                sw      $t8, 0($t9)
                la      $t8,CharPtr #empty CharPtr
                sw      $0,0($t8)
                la      $t8,BytePtr #empty BytePtr
                sw      $0,0($t8)
                la      $t8,WordPtr #empty WordPtr
                sw      $0,0($t8)
                jr      $ra
```

6) Ta tính bằng công thức lấy Sys_myFreeSpace - (Sys_TheTopOfFree +4). Thêm 4 là do MIPS cấp phát liên tục là Sys_TheTopOfFree chiếm 4 byte và lưu giá trị địa chỉ còn trống để cấp phát. Ta tính cả địa chỉ tự do đã cấp phát.

```
                jr      $ra
#6) Viết hàm tính toàn bộ lượng bộ nhớ đã cấp phát.
#Dia chi da cap phat bao gom dia chi tu do cap phat cho byte.
#Cach tinh : myFreeSpace - (TheTopOfFree + 4)
# @param: Khong co
AllocatedMem:   la      $t9, Sys_TheTopOfFree ##Lay con tro chua dia chi con trong
                lw      $t8,0($t9)      #Lay dia chi con trong
                sub     $a0,$t8,$t9     #gia tri dia chi dau tien con trong - dia chi TheTopOfFree
                addi    $a0,$a0,-4      #So phan tu duoc cap phat
                li      $v0,1           #In ra so Phan tu da duoc cap phat
                syscall
                jr      $ra
```

7) Ta cấp phát như malloc nhưng với malloc 2 ta phái lấy số hàng.số cột.4 và kiểm tra kiểu word phải ở địa chỉ chia hết cho 4.

```
#7) Hãy viết hàm malloc2 để cấp phát cho mảng 2 chiều kiểu .word với tham số vào gồm:
#@param Địa chỉ con tro 2 chieu          $a0
#@param Số dòng                          $a1
#@param Số cột                           $a2
malloc2:
        la      $t9, Sys_TheTopOfFree   #Lay con tro chua dia chi con trong
        lw      $t8, 0($t9)             #Lay dia chi con trong
        li      $t7,4                   #So byte word
#Kiem tra xem dia chi co chia het cho 4 hay khong
WhileWord:
        div     $t8,$t7 #Neu dia chi con trong khong chia het cho 4 thi cap phat them
        mfhi    $t0     #load so du
        beqz    $t0,endWhileWord#so du = 0 thi ket thuc while
        nop
        addi    $t8,$t8,1       # Dia chi + 1
        j       WhileWord
endWhileWord:
        sw      $t8,0($a0)      #cat dia chi cua mang vao con tro chua
        mul     $t7, $a1,$a2    #Tinh kich thuoc cua mang can cap phat
        mul     $t7, $t7,4      #Bo nho can cap phat= Kich thuoc mang * 4
        add     $t6, $t8, $t7   #Gia tri dia chi trong moi = Gia tri dia chi trong cu + Bo nho can cap phat
        sw      $t6, 0($t9)     #Luu tro lai dia chi dau tien do vao con tro Sys_TheTopOfFree
        jr      $ra
```

8) Ta coi mảng 2 chiều như 1 mảng một chiều được nối vào nhau.
Với hàm getArray, và setArray tương đối giống nhau. Chỉ khác setArray ta có gán tham số, còn getArray ta trả về giá trị.

```
#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
#
#@param[in] Địa chỉ đầu của mảng        $a0
#@param[in] Số dòng                     $a1
#@param[in] Số cột                      $a2
#@param[in] Số cột max                  $t1
#Gia tri gan                            $a3
setArray:
        mul     $t2,$t1,4       #t2 = Khoang cach nhay giua cac dong
        mul     $a1,$a1,$t2     #a1 = Dia chi dong can den
        mul     $t3,$a2,4       #offset Dia chi cot can den so voi hang
        add     $a1,$a1,$t3     #Dia chi a[i][j] tuong doi voi dia chi mang
        add     $t4,$a0,$a1     #$t4 = Dia chi a[i][j] tuyet doi
        sw      $a3,0($t4)
        jr      $ra
#
#@param[in] Địa chỉ đầu của mảng        $a0
#@param[in] Số dòng                     $a1
#@param[in] Số cột                      $a2
#@param[in] Số cột max                  $t1
#return gia tri                         $v0
getArray:
        mul     $t2,$t1,4       #Khoang cach nhay giua cac dong
        mul     $a1,$a1,$t2     #Dia chi dong can den
        mul     $t3,$a2,4       #Dia chi cot can den
        add     $a1,$a1,$t3     #Dia chi a[i][j] tuong doi so voi goc
        add     $t4,$a0,$a1     #Dia chi a[i][j] tuong doi voi dia chi mang
        lw      $v0,0($t4)      #$t4 = Dia chi a[i][j] tuyet doi
        jr      $ra
```

Mã nguồn:

```
.data
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 kieu Word
CharCopy: .word 0 #Copy bien con tro
Array2D: .word 0 #Bien con tro 2 chieu
Message1: .asciiz "\nNhap du lieu test ham 2 con tro:"
Message2: .asciiz "\nDu lieu a[i][j]:"
Message3: .asciiz "\nTong du lieu: "
#CharSize
#byteSize
#wordSize
.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
#-----------------------
# Cap phat cho bien con tro, gom 3 phan tu,moi phan tu 1 byte
#-----------------------
la      $a0, CharPtr
addi    $a1, $zero, 3
addi    $a2, $zero, 1
jal     malloc

#--------------------------------------
#Ham copy xau
#--------------------------------------
li $v0,4
la $a0,Message1
syscall
jal     NhapChuoi     #tao gia tri kiem thu
la      $a0,CharPtr
li      $a1,3
la      $a2,CharCopy
jal     CopyPtr

#--------------------------------------
#Cap phat bo nho cho mang 2 chieu
#--------------------------------------
# allocated a array a[2][3] ( mang 2 hang 3 cot )
la      $a0,Array2D   #dia chi dau vao
addi    $a1,$0,2      #dong max
addi    $a2,$0,3      #cot max
jal     malloc2
#set a[1][2] = 5
la      $a0,Array2D
lw      $a0,0($a0)
addi    $a1,$0,1 #dong
addi    $a2,$0,2 #cot
addi    $t1,$0,3 #cot max
```

```
addi    $a3,$0,5 #gia tri gan cho mang
jal     setArray
#set a[1][1] = 4
la      $a0,Array2D
lw      $a0,0($a0)
addi    $a1,$0,1 #dong
addi    $a2,$0,1 #cot
addi    $t1,$0,3 #cot max
addi    $a3,$0,4 #gia tri gan cho mang
jal     setArray
#message get
li $v0,4
la $a0,Message2 #"\nDu lieu a[i][j]:"
syscall
#get a[1][1] =4
la      $a0,Array2D
lw      $a0,0($a0)
addi    $a1,$0,1 #dong
addi    $a2,$0,1 #cot
addi    $t1,$0,3 #cot max

jal     getArray
add     $a0,$0,$v0
addi    $v0,$0,1
syscall
#set a[1][1] =3
la      $a0,Array2D
lw      $a0,0($a0)
addi    $a1,$0,1 #dong
addi    $a2,$0,1 #cot
addi    $t1,$0,3 #cot max
addi    $a3,$0,3 #gia tri gan cho mang
jal     setArray

#set a[1][0] =2
la      $a0,Array2D
lw      $a0,0($a0)
addi    $a1,$0,1 #dong
addi    $a2,$0,0 #cot
addi    $t1,$0,3 #cot max
addi    $a3,$0,2 #gia tri gan cho mang
jal     setArray

#message get
li $v0,4
la $a0,Message2  #"\nDu lieu a[i][j]:"
syscall
#get a[1][1] =3
la      $a0,Array2D
lw      $a0,0($a0)
addi    $a1,$0,1 #dong
addi    $a2,$0,1 #cot
addi    $t1,$0,3 #cot max
jal     getArray
add     $a0,$0,$v0
addi    $v0,$0,1
syscall
#in ra gia tri cua mang 2 chieu
```

```
#-------------------------------------
#Ham tinh tong bo nho
#-------------------------------------
li $v0,4
la $a0,Message3
syscall

jal AllocatedMem
#-------------------------------------
#Ham Clean bo nho
#-------------------------------------
jal FreeMem


lock:   j lock
nop
#-----------------------------------------
# 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       #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
#-----------------------------------------
# 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
        #1) Word phai duoc cap phat o dia chi chia het cho 4
        #Dia chi kieu word chia het 4
        #while( diachi % 4 != 0)
        # diachi ++ => cap phat cho bien char them dia chi
checkWord:
        bne     $a2,4,endCheckWord
        nop
While:  div     $t8,$a2
        mfhi    $t0
        beqz    $t0,endCheckWord
        nop
        addi    $t8,$t8,1       # Dia chi + 1
        j       While
endCheckWord:
        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 dia chi dau tien do vao bien Sys_TheTopOfFree
        jr      $ra
```

```
#2) Viết hàm lấy giá trị của biến con trỏ.
#-----------------------------------------
# Ham lay gia gia tri cua bien con tro
# @param [in/out] $a0 Chua dia chi cua bien con tro
# @return $v0 gia tri cua bien con tro
#-----------------------------------------
PtrValue:       lw $v0,0($a0)
                jr  $ra
#3) Viết hàm lấy địa chỉ biến con trỏ.
#@return tra ve dia chi cua tung bien con tro
addressCharPtr:la      $v0,CharPtr
               jr  $ra
addressBytePtr:la      $v0,BytePtr
               jr      $ra
addressWordPtr:la      $v0,WordPtr
               jr      $ra
#4) Viết hàm thực hiện copy 2 con trỏ xâu kí tự.
#-----------------------------------------
# Ham lay gia gia tri cua bien con tro
# @param [in] $a0 Chua dia chi cua bien con tro xau^ goc'
# @param [in] $a1 Chua so phan tu cua con tro xau^ goc'
# @param [in/out] $a2 Chua dia chi con tro xau copy, khi ket thuc chua dia chi copy chuoi moi
# @return $v0 gia tri cua bien con tro
#-----------------------------------------
CopyPtr:
        sw      $ra,-4($sp)     #store the return address
        sw      $a0,-8($sp)     #store $a0 value
        sw      $a1,-12($sp)    #store $a1 value
        sw      $a2,-16($sp)    #store $a2 value
        addi    $sp,$sp,-16     #allocate space for $ra,$a0,$a1,$a2

        add     $a0,$0, $a2     # cap phat bo nho cho con tro copy
        add     $a1, $zero, $a1 # So phan tu cap phat cho string copy
        addi    $a2, $zero, 1   # so byte cho phan tu
        jal     malloc

        lw      $a2,0($sp)      #restore $a2
        lw      $a1,4($sp)      #restore $a1
        lw      $a0,8($sp)      #restore $a0
        lw      $ra,12($sp)     #restore $ra
        addi    $sp,$sp,16      #restore stack pointer

        lw      $t0,0($a0)      #lay dia chi da cap phat cua xau goc
        lw      $t1,0($a2)      #lay dia chi da cap phat cua xau copy

        li      $t2,0           #i = 0
LoopCopy:       slt     $t9,$t2,$a1     #i < so phan tu cua xau
                beqz    $t9,end_LoopCopy
                nop
                add     $t9,$t0,$t2     # $t9 = dia chi xau goc + i
                add     $t7,$t1,$t2     # $t7 = dia chi xau copy +i
                lb      $t3,0($t9)      #  t7[i] = t9[i]
                sb      $t3,0($t7)      #
                addi    $t2,$t2,1       # i ++
                j       LoopCopy
                nop
end_LoopCopy:
```

```
        jr      $ra
#
NhapChuoi:
la      $a0,CharPtr
lw      $t0,0($a0) # load dia chi dau
addi    $t1,$t0,3 # dia chi dich
loop:
slt     $t3,$t0,$t1# Neu chua den dia chi dich thi tiep tuc read character
beqz    $t3,endNhap
nop
li      $v0,12 #read chracter
syscall
nop
beq     $v0,10,endNhap
nop
sb      $v0,0($t0)
addi    $t0,$t0,1
j       loop
nop
endNhap:
jr      $ra


#5) Viết hàm giải phóng bộ nhớ đã cấp phát cho các biến con trỏ.
FreeMem:
        la      $t9, Sys_TheTopOfFree       #Lay con tro chua dia chi dau tien con trong
        lw      $t7,0($t9)      #gia tri dia chi dau tien con trong
        addi    $t6,$t9,3       #mut' dia chi cuoi cung cua Sys_TheTopOfFre
        addi    $t7,$t7,-1      #lui 1 byte den dia chi cuoi cung duoc cap phat
CleanLoop:      slt     $t8,$t6,$t7 # chua den vi tri con trong ban dau duoc cap phat (kdata +4) thi
tiep tuc
                beqz    $t8,End_CleanLoop
                nop
                sb      $0,0($t7)
                addi    $t7,$t7,-1 # lui 1 byte
                j       CleanLoop
End_CleanLoop:
                la      $t8,Sys_MyFreeSpace # dat lai gia tri dia chi con trong co the cap phat
                sw      $t8, 0($t9)
                la      $t8,CharPtr #empty CharPtr
                sw      $0,0($t8)
                la      $t8,BytePtr #empty BytePtr
                sw      $0,0($t8)
                la      $t8,WordPtr #empty WordPtr
                sw      $0,0($t8)
                jr      $ra
#6) Viết hàm tính toàn bộ lượng bộ nhớ đã cấp phát.
#Dia chi da cap phat bao gom dia chi tu do cap phat cho byte.
#Cach tinh : myFreeSpace - (TheTopOfFree + 4)
# @param: Khong co
AllocatedMem:           la      $t9, Sys_TheTopOfFree ##Lay con tro chua dia chi con trong
                lw      $t8,0($t9)      #Lay dia chi con trong
                sub     $a0,$t8,$t9     #gia tri dia chi dau tien con trong - dia chi TheTopOfFree
                addi    $a0,$a0,-4      #So phan tu duoc cap phat
                li      $v0,1           #In ra so Phan tu da duoc cap phat
                syscall
                jr      $ra
#7) Hãy viết hàm malloc2 để cấp phát cho mảng 2 chiều kiểu .word với tham số vào gồm:
```

```
#@param Địa chỉ con tro 2 chieu    $a0
#@param Số dòng                   $a1
#@param Số cột                    $a2
malloc2:
        la      $t9, Sys_TheTopOfFree       #Lay con tro chua dia chi con trong
        lw      $t8, 0($t9)             #Lay dia chi con trong
        li      $t7,4                   #So byte word
#Kiem tra xem dia chi co chia het cho 4 hay khong
WhileWord:
        div     $t8,$t7         #Neu dia chi con trong khong chia het cho 4 thi cap phat them
        mfhi    $t0     #load so du
        beqz    $t0,endWhileWord#so du = 0 thi ket thuc while
        nop
        addi    $t8,$t8,1       # Dia chi + 1
        j       WhileWord
endWhileWord:
        sw      $t8,0($a0)      #cat dia chi cua mang vao con tro chua
        mul     $t7, $a1,$a2    #Tinh kich thuoc cua mang can cap phat
        mul     $t7, $t7,4      #Bo nho can cap phat= Kich thuoc mang * 4
        add     $t6, $t8, $t7   #Gia tri dia chi trong moi = Gia tri dia chi trong cu + Bo nho can cap
phat
        sw      $t6, 0($t9)     #Luu tro lai dia chi dau tien do vao con tro Sys_TheTopOfFree
        jr      $ra
#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
#
#@param[in] Địa chỉ đầu của mảng                 $a0
#@param[in] Số dòng                      $a1
#@param[in] Số cột               $a2
#@param[in] Số cột max                   $t1
#Gia tri gan                    $a3
setArray:
        mul     $t2,$t1,4       #t2 = Khoang cach nhay giua cac dong
        mul     $a1,$a1,$t2     #a1 = Dia chi dong can den
        mul     $t3,$a2,4       #offset Dia chi cot can den so voi hang
        add     $a1,$a1,$t3     #Dia chi a[i][j] tuong doi voi dia chi mang
        add     $t4,$a0,$a1     #$t4 = Dia chi a[i][j] tuyet doi
        sw      $a3,0($t4)
        jr      $ra
#
#@param[in] Địa chỉ đầu của mảng                 $a0
#@param[in] Số dòng                      $a1
#@param[in] Số cột               $a2
#@param[in] Số cột max                   $t1
#return gia tri                 $v0
getArray:
        mul     $t2,$t1,4       #Khoang cach nhay giua cac dong
        mul     $a1,$a1,$t2     #Dia chi dong can den
        mul     $t3,$a2,4       #Dia chi cot can den
        add     $a1,$a1,$t3     #Dia chi a[i][j] tuong doi so voi goc
        add     $t4,$a0,$a1     #Dia chi a[i][j] tuong doi voi dia chi mang
        lw      $v0,0($t4)      #$t4 = Dia chi a[i][j] tuyet doi
        jr      $ra
```

Kết quả chạy: Các ô nhớ đã được clear và chạy cuối cùng.