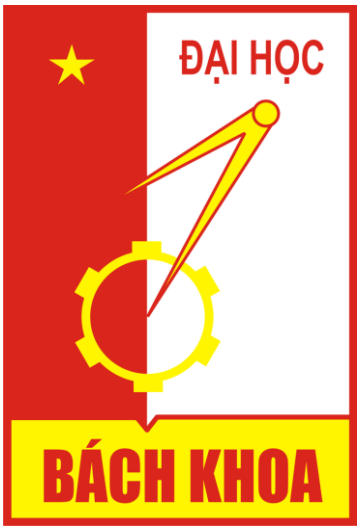
ĐẠI HỌC BÁCH KHOA HÀ NỘI

TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG



BÁO CÁO CUỐI KỲ MÔN

Thực hành Kiến trúc máy tính

*BÀI TẬP SỐ **4+6***

Giảng viên hướng dẫn: .. Lê Bá Vui

Sinh viên:..... Trần Quang Huy - 20194589

..... Ngô Hoàng Vũ - 20194721

Nhóm: 11

Lớp: 130939

Mục lục

PHỤ LỤC HÌNH ẢNH

I. BÀI TẬP 4

1. YÊU CẦU: POSTSCRIPT CNC MARSBOT

Máy gia công cơ khí chính xác CNC Marsbot được dùng để cắt tấm kim loại theo các đường nét được qui định trước. CNC Marsbot có một lưỡi cắt dịch chuyển trên tấm kim loại, với giả định rằng:

- Nếu lưỡi cắt dịch chuyển nhưng không cắt tấm kim loại, tức là Marsbot di chuyển nhưng không để lại vết (Track)
- Nếu lưỡi cắt dịch chuyển và cắt tấm kim loại, tức là Marsbot di chuyển và có để lại vết.

Để điều khiển Marsbot cắt đúng như hình dạng mong muốn, người ta nạp vào Marsbot một mảng cấu trúc gồm 3 phần tử:

<Góc chuyển động>, <Cắt/Không cắt>, <Thời gian>

Trong đó:

- <Góc chuyển động> là góc của hàm HEADING của Marsbot
- <Cắt/Không cắt> thiết lập lưu vết/không lưu vết
- <Thời gian> là thời gian duy trì quá trình vận hành hiện tại

Hãy lập trình để CNC Marsbot có thể:

- Thực hiện cắt kim loại như đã mô tả
- Nội dung postscript được lưu trữ cố định bên trong mã nguồn
- Mã nguồn chứa 3 postscript và người dùng sử dụng 3 phím 0, 4, 8 trên bàn phím Key Matrix để chọn postscript nào sẽ được gia công.
- Một postscript chứa chữ DCE cần gia công. Hai script còn lại sinh viên tự đề xuất (tối thiểu 10 đường cắt)

2. PHÂN TÍCH CÁCH LÀM, THUẬT TOÁN

- *B1*: Tạo vòng lặp vô hạn để chờ lệnh từ người dùng.
- *B2*: Interrupt từ bàn phím Hexa sẽ nhận lệnh người dùng.
- *B3*: Lần lượt đọc các giá trị trong postscript và tiến hành gia công.
- *B4*: Quay lại chờ lệnh tiếp theo sau khi gia công hoàn thành.

3. Ý NGHĨA CÁC HÀM TRONG MÃ NGUỒN

3.1. ROTATE

- *Ý nghĩa*: quay Marsbot theo hướng có số độ lưu trong thanh ghi $a0
- *Đầu vào*: thanh ghi $a0

- Lưu giá trị trong thanh ghi $a0 vào địa chỉ HEADING (0xffff8010) để Marsbot chuyển hướng.

3.2. SET_TRACK

- *Ý nghĩa*: thay đổi trạng thái lưu vết của Marsbot.
- *Đầu vào*: thanh ghi $a0
- Lưu giá trị trong thanh ghi $a0 vào địa chỉ LEAVETRACK (0xffff8020) để thay đổi trạng thái lưu vết của Marsbot.

3.3. GO, STOP

- *Ý nghĩa*: điều khiển Marsbot bắt đầu chuyển động (GO) hoặc dừng lại (STOP)
- Load 1 vào địa chỉ MOVING (0xffff8050) nếu muốn để lại vết và load 0 nếu muốn kết thúc vết.

3.4. draw_padding

- *Ý nghĩa*: di chuyển Marsbot ra khỏi cạnh màn hình để tránh trường hợp vẽ bị mất nét do syscall sleep không thực sự thời gian thực.

3.5. main

- *Ý nghĩa*: hàm xử lý chính của chương trình kích hoạt interrupt từ bàn phím Hexa và chứa một vòng lặp vô hạn để chờ đến khi có postscript cần gia công trong thanh ghi $s0 sẽ tiến hành gia công.

3.6. start_cut_metal

- *Ý nghĩa*: hàm gia công một postscript. Đầu tiên hàm tạm dừng cơ chế bắt interrupt từ bàn phím Hexa, sau đó đọc từng bộ ba số từ postscript và tiến hành đổi hướng marsbot và xác định xem có lưu vết hay không trong một khoảng thời gian tương ứng. Sau khi đọc đến giá trị -1 (âm một) từ postscript sẽ cho Marsbot dừng lại, và quay lại hàm main.

3.7. Các thao tác trong phần xử lí interrupt

- Lần lượt quét các hàng của Digital Lab Sim để xem phím nào được bấm. Nếu phím bấm hợp lệ sẽ gán postscript cần gia công tương ứng cho thanh ghi $s0.

4. MÃ NGUỒN

Lưu trong file n04_g11_TranQuangHuy.asm đi kèm.

```
# Programmer: Trần Quang Huy                                    #
# Description: This program simulate a CNC Marsbot to cut metal.    #
# Bugs:                                                          #
```

```
#################################################################################################
######


#.....................................................................................

# Hexadecimal keyboard

.eqv IN_ADRESS_HEXA_KEYBOARD 0xFFFF0012

.eqv OUT_ADRESS_HEXA_KEYBOARD 0xFFFF0014

#.....................................................................................

# Marsbot

.eqv HEADING 0xffff8010  # Integer: An angle between 0 and 359

                                                # 0 : North (up)

                                                # 90: East (right)

                                                # 180: South (down)

                                                # 270: West (left)

.eqv MOVING 0xffff8050                  # Boolean whether or not to move

.eqv LEAVETRACK 0xffff8020        # Boolean whether or not to leave a track

.eqv WHEREX 0xffff8030   # Integer: Current x-location of MarsBot

.eqv WHEREY 0xffff8040   # Integer: Current y-location of MarsBot

#.....................................................................................

# Key value

   .eqv KEY_0 0x11

   .eqv KEY_4 0x12

   .eqv KEY_8 0x14


#=====================================================

#=====================================================
```

```
.data
  DCE_script:
  .word  180, 1, 5000, 70, 1, 1500, 35, 1, 1500, 0, 1, 1500, 325, 1, 1500, 290,    1, 1500,     # D
            90, 0, 5000,                                                                          # sleeping
            255, 1, 1500, 195, 1, 2186, 165, 1, 2186, 105, 1, 1500,                               # C
            90, 0, 1000,                                                                          # sleeping
            90, 1, 2000, 270, 0, 2000, 0, 1, 2500, 90, 1, 2000, 270, 0, 2000, 0, 1,               # E
            2500, 90, 1, 2000,
            90, 0, 5000, -1                                                                       # sleeping

HUY_script:
.word  180, 1, 5000, 0, 0, 2500, 90, 1, 2500, 180, 1, 2500, 0, 0, 2500, 0 1, 2500               # H
            90, 0, 1000,                                                                          # sleeping
            180, 1, 4300, 140, 1, 990, 90, 1, 1300, 40, 1, 990,                                   # U
            0, 1, 4500,
            90, 0, 1000,                                                                          # sleeping
            135, 1, 2828, 180, 1 3000, 0, 0, 3000, 45, 1, 2828,                                   # Y
            90, 0, 5000, -1                                                                       # sleeping

  BUMA_script:
.word 180, 1, 5000, 80, 1, 1500, 35, 1, 750, 0, 1, 800, 325, 1, 750,                              # B
          280, 1, 750, 270, 1, 750, 80, 1, 1500, 35, 1, 750, 0, 1, 800,
          325, 1, 750, 295, 1, 750, 270, 1, 750,
            90, 0, 3000,                                                                          # sleeping
            180, 1, 4300, 140, 1, 990, 90, 1, 1300, 40, 1, 990,                                   # U
```

```
        0, 1, 4500,

        90, 0, 1000,                                                    # sleeping
        180, 1, 5000, 0, 0, 5000, 150, 1, 2500, 30, 1, 2500,           # M
        180, 1, 5000,
        90, 0, 1000,                                                    # sleeping
        15, 1, 5200, 165, 1, 5200, 0, 0, 2300, 270, 1, 3000,           # A
        0, 0, 2700, 90, 0, 7500, -1                                     # sleeping
padding_script: .word 90, 0, 1000, 180, 0, 1000, -1


#=====================================================

#=====================================================
```

```
.text
draw_padding:
  la $s0, padding_script
  j start_cut_metal
main:
  #.....................................................................................
  # Enable the interrupt of Keyboard matrix 4x4 of Digital Lab Sim
  li $t0, IN_ADRESS_HEXA_KEYBOARD
  li $t1, 0x80          # bit 7 = 1 to enable
  sb $t1, 0($t0)
  #.....................................................................................
  # Infinite loop that wait for interrupt
  loop:
    nop
    bnez $s0, start_cut_metal
    nop
    b loop
    nop
    b loop

  start_cut_metal:
  li $t1, 0x00                  # disable interrupt while cutting
  sb $t1, 0($t0)
  jal GO

  read_script:
```

```
    lw $a0, 0($s0)     # load direction, -1 if script end.
    beq $a0, -1, stop
    jal ROTATE                  # change direction of Marsbot

    lw $a0, 4($s0)
    jal SET_TRACK

  sleep:
    li $v0, 32                  # keep running by sleeping
    lw $a0, 8($s0)
    syscall

    li $a0, 0
    jal SET_TRACK# auto untrack everytime to keep path

    addi $s0, $s0, 12
    j read_script

  stop:
    jal STOP                    # stop Marsbot
    li $s0, 0           # clear current script
    j main              # wait another draw command

#-------------------------------------------------------------------------------------------
# SET_TRACK procedure, to start drawing line
# param[in]           $a0, A boolean whether leavetrack or not
```

```
#------------------------------------------------------------------------------------------
SET_TRACK:
  li $at, LEAVETRACK       # change LEAVETRACK port
  sb $a0, 0($at)
  nop
  jr $ra
  nop
#------------------------------------------------------------------------------------------
# ROTATE procedure, to rotate the robot
# param[in]           $a0, An angle between 0 and 359
#------------------------------------------------------------------------------------------
ROTATE:
  li $at, HEADING  # change HEADING port
  sw $a0, 0($at)       # to rotate robot
  nop
  jr $ra
  nop
#------------------------------------------------------------------------------------------
# GO procedure, to start running
# param[in]           none
#------------------------------------------------------------------------------------------
GO: li $at, MOVING           # change MOVING port
 addi $k0, $zero, 1  # to logic 1,
 sb $k0, 0($at)        # to start running
 nop
 jr $ra
```

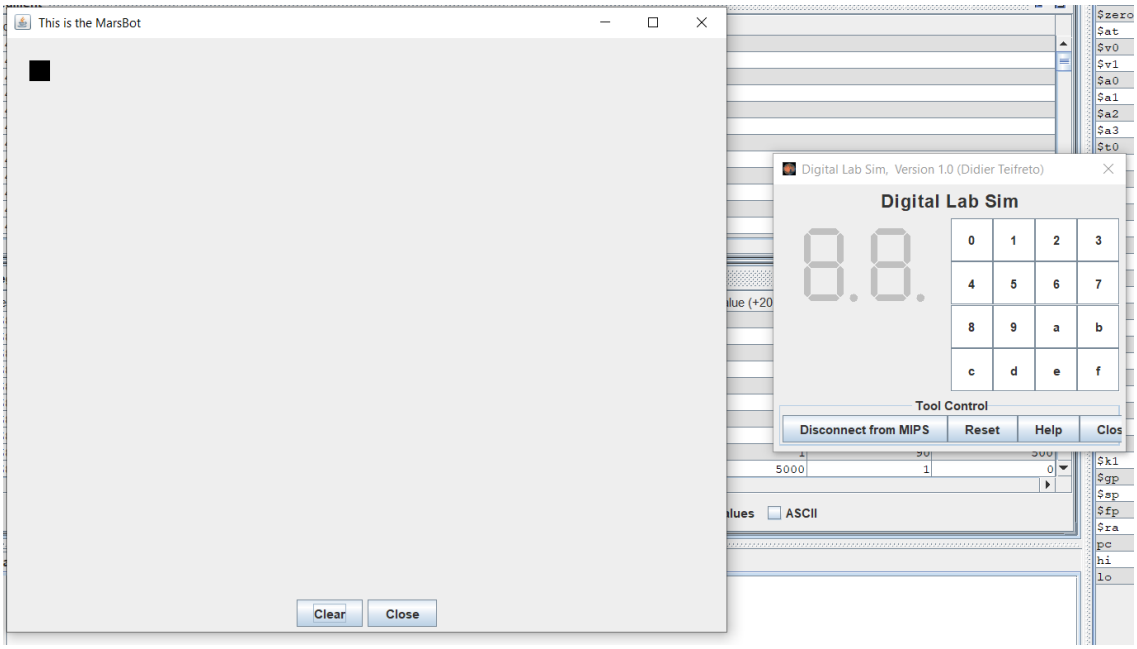
```
 nop
#-----------------------------------------------------------------------------------------
# STOP procedure, to stop running
# param[in]            none
#-----------------------------------------------------------------------------------------
STOP: li $at, MOVING # change MOVING port to 0
 sb $zero, 0($at) # to stop
 nop
 jr $ra
 nop
#=====================================================
#=====================================================
.ktext 0x80000180
#.....................................................................................
# Backup
  addi $sp, $sp, 4
  sw $t1, 0($sp)
  addi $sp, $sp, 4
  sw $t2, 0($sp)
  addi $sp, $sp, 4
  sw $t3, 0($sp)
  addi $sp, $sp, 4
  sw $a0, 0($sp)
#.....................................................................................
# Processing
get_number:
```

```
  li $t1, IN_ADRESS_HEXA_KEYBOARD
  li $t2, OUT_ADRESS_HEXA_KEYBOARD
scan_row1:                          # Scan number 0, 1, 2, 3
  li $t3, 0x81
  sb $t3, 0($t1)
  lbu $a0, 0($t2)
  bnez $a0, check_number
scan_row2:                          # Scan number 4, 5, 6, 7
  li $t3, 0x82
  sb $t3, 0($t1)
  lbu $a0, 0($t2)
  bnez $a0, check_number
scan_row3:                          # Scan number 8, 9 + letter A, B
  li $t3, 0x84
  sb $t3, 0($t1)
  lbu $a0, 0($t2)
  bnez $a0, check_number

check_number:
  beq $a0, KEY_0, case_0
  beq $a0, KEY_4, case_4
  beq $a0, KEY_8, case_8
  j next_pc

case_0:
  la $s0, DCE_script
```

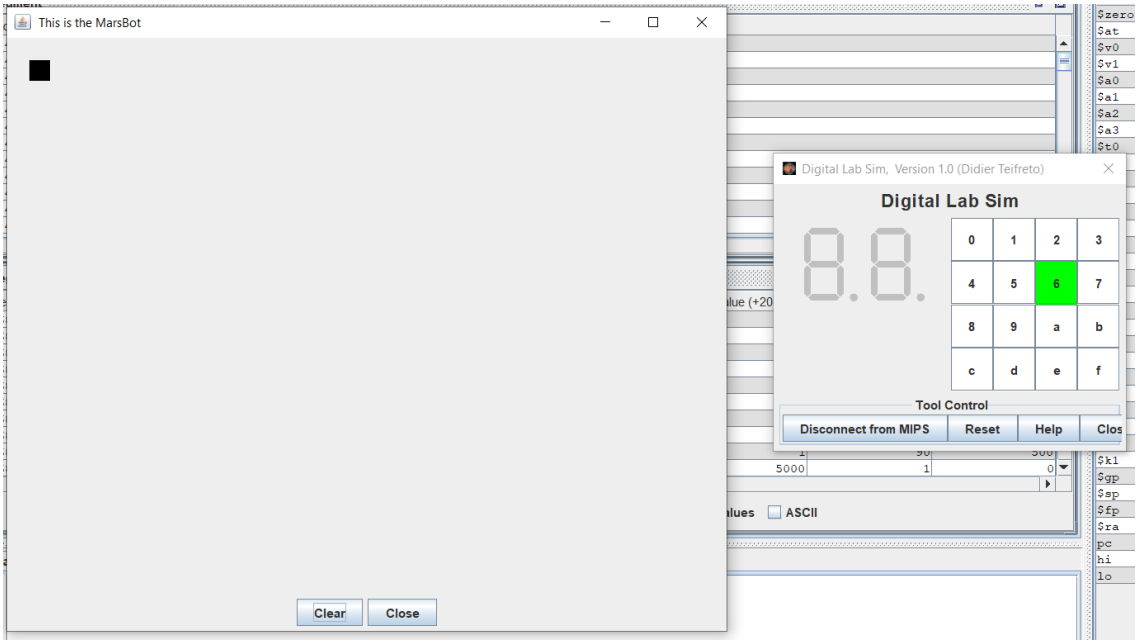
```
  j next_pc


case_4:
  la $s0, HUY_script
  j next_pc


case_8:
  la $s0, BUMA_script
  j next_pc


#.....
# Evaluate the return address of main routine epc <- epc + 4
next_pc:
  mfc0 $at, $14       # $at <= Coproc0.$14 = Coproc0.epc
  addi $at, $at, 4    # $at = $at + 4 (next instruction)
  mtc0 $at, $14       # Coproc0.$14 = Coproc0.epc <= $at
#.....
# Restore the registers from stack
  lw $a0, 0($sp)
  addi $sp, $sp, -4
  lw $t3, 0($sp)
  addi $sp, $sp, -4
  lw $t2, 0($sp)
  addi $sp, $sp, -4
  lw $t1, 0($sp)
  addi $sp, $sp, -4
```

return: eret # Return from exception

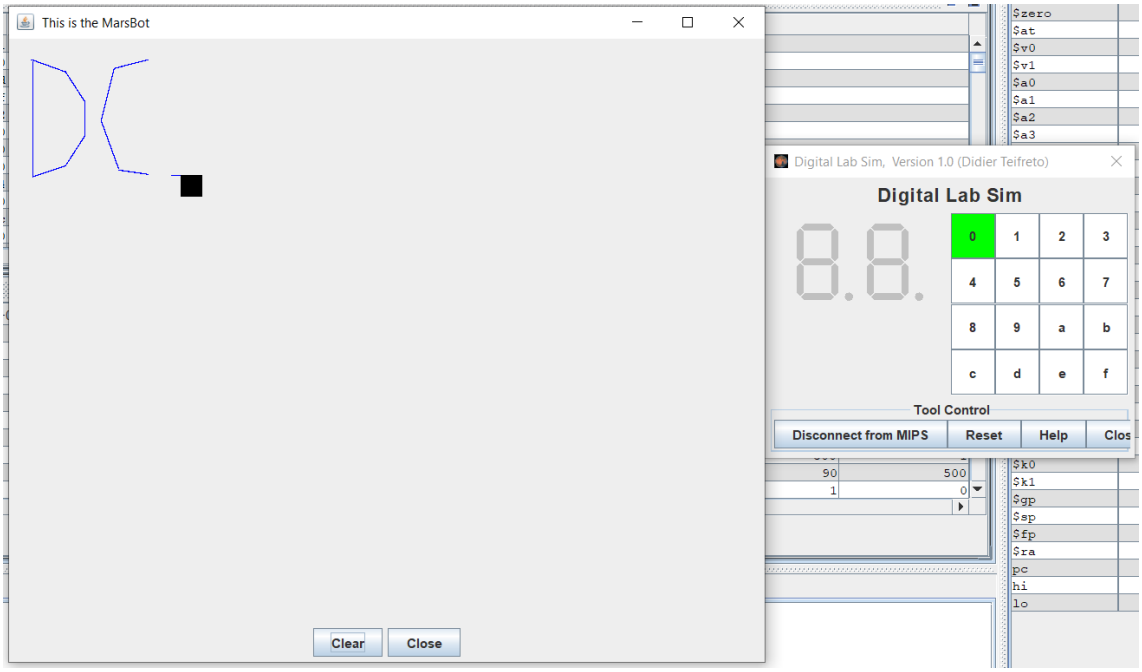
5. HÌNH ẢNH MÔ PHỎNG



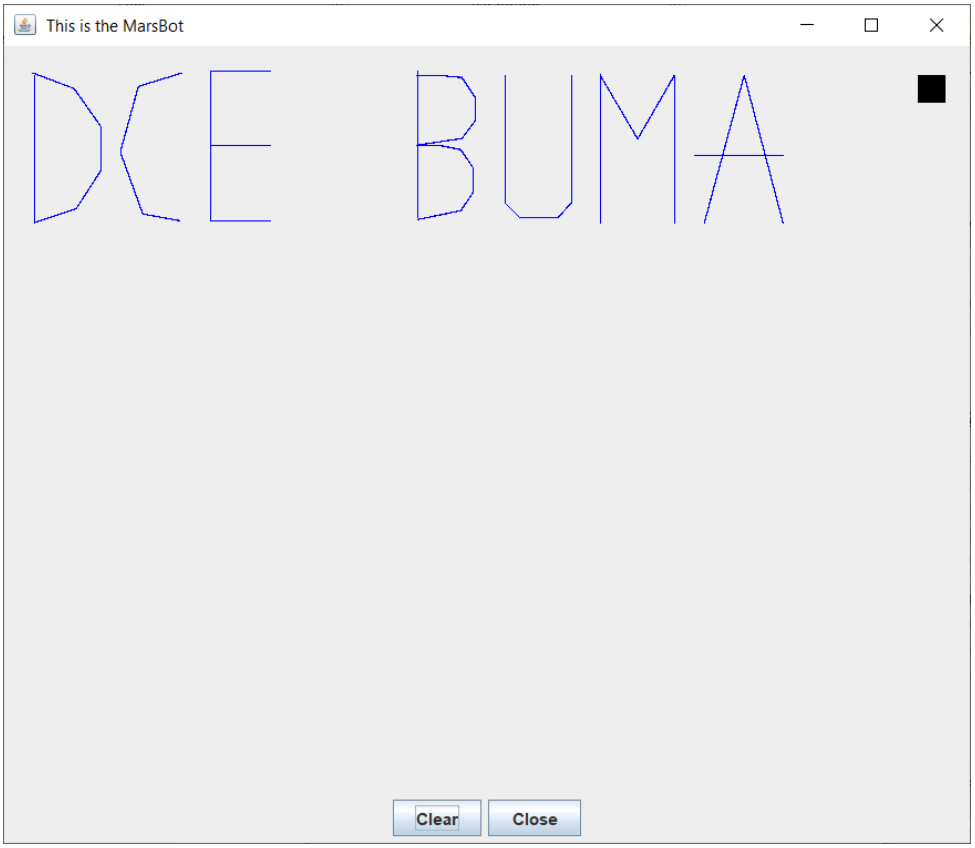
Hình I.1. Marsbot dừng chờ lệnh từ người dùng sau khi in padding



Hình I.2. Chương trình không làm gì vì người dùng lựa chọn số không hợp lệ



Hình I.3. Đang tiến hành gia công postscript có chữ DCE khi chọn số 0



Hình I.4. Hoàn thành gia công hai postscript (0 và 8)

II. BÀI TẬP 6

1. YÊU CẦU: 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 viết2hàmgetArray[i][j] và setArray[i][j] đểlấy/thiết lập giá trị cho phần tử ở dòng I cột j của mảng.

2. PHÂN TÍCH CÁCH LÀM, THUẬT TOÁN

- Tạo 1 vòng lặp menu để đợi yêu cầu của người dùng
- Đề bài có 8 yêu cầu nhưng menu chỉ thực hiện 6 option. Yêu cầu 1 không cần nằm trong menu. Yêu cầu 4 đã được viết hàm nhưng không nằm trong option của menu.
- Sau khi thực hiện xong option của menu, tiếp tục vòng lặp và đợi yêu cầu tiếp theo
- Gỉa sử có 1 dữ liệu string đã được nhập vào được lưu trong biến Input kiểu ascii "Name: Ngo Hoang Vu-MSSV: 20194721"

3. Ý NGHĨA CÁC HÀM TRONG MÃ NGUỒN

3.1. SysInitMem

- Ý nghĩa: Khỏi tạo vùng nhớ cho việc cấp phát động.
- Sys_TheTopOfFree: con trỏ mang giá trị là địa chỉ đầu tiênĩa ở vùng còn trống.
- Sys_MyFreeSpace: con trỏ có địa chỉ đầu tiên của vùng nhớ cấp phát.

3.2. Malloc

- Ý nghĩa: cấp phát 1 phần bộ nhớ cho 1 biến
- Đầu vào: $a0 – mang địa chỉ của biến. $a1 – mang giá trị số phần tử của mảng. $a2 – mang giá trị số byte của mỗi phần tử
- Đầu ra: $v0 - địa chỉ phần bộ nhớ đã được cấp phát cho biến
- Mỗi lần cấp phát, bộ nhớ cần cấp phát số byte = $a1 * $a2. Để thỏa mãn cấp phát vùng nhớ theo word (số byte chia hết cho 4) hàm sẽ cấp phát một số byte = số nhỏ nhất chia hết cho 4 lớn hơn số byte đã tính vừa nãy.

3.3. Strcpy

- Ý nghĩa: copy 2 xâu ký tự
- Đầu vào: $a1 - địa chỉ nguồn, $a2 - địa chỉ đích.
- Đọc từng byte bắt đầu từ địa chỉ nguồn lưu vào địa chỉ đích. Vòng lặp kết thúc khi đọc đến giá trị kết thúc (/0)

3.4. Opt1

- Đầu vào: $a0 - địa chỉ biến con trỏ. $t0 - giá trị offset được nhập bởi người dùng
- Đầu ra: $s0 - lưu giá trị byte lấy từ vùng nhớ được cấp phát cho biến con trỏ.
- Đọc giá trị byte được lưu với địa chỉ mà biến con trỏ chỉ đến và giá trị offset.

3.5. Opt2

- Đầu vào: $a0 - lưu địa chỉ biến con trỏ
- Đầu ra: $s1 - địa chỉ vùng nhớ được cấp phát cho biến
- Từ địa chỉ vùng nhớ được lưu tại địa chỉ biến con trỏ, trả về lưu tại $s1

3.6. Opt3

- Ý nghĩa: Giải phóng toàn bộ vùng nhớ đã được cấp phát cho các biến con trỏ.
- Từ địa chỉ đầu tiên trong vùng nhớ cấp phát Sys_MyFreeSpace đến trước khi chạm địa chỉ đầu tiên trong vùng còn trống(lưu trong Sys_TheTopOfFree), gán cho các vùng nhớ giá trị 0.

3.7. Opt4bb

- Ý nghĩa: Đếm số byte vùng nhớ đã cấp phát cho các biến con trỏ.
- Tương tự Opt3, tuy nhiên không gán cho các vùng nhớ mà trong mỗi vòng lặp lại cộng biến count lên 4 byte.

3.8. Malloc2

- Ý nghĩa: Cấp phát bộ nhớ cho mảng 2 chiều kiểu word

- Đầu vào : $a0 - địa chỉ của biến con trỏ trỏ đến mảng mảng 2 chiều. $a1 - số dòng. $a2 – số cột.
- Cả mảng 2 chiều sẽ sử dụng tổng cộng $a1 * $a2 vùng nhớ, mỗi vùng nhớ dùng 4 byte.

3.9. Opt5

- Lấy giá trị lưu tại Array2[i][j] với i,j theo yêu cầu của người dùng.
- Đầu vào: $a0 - địa chỉ vùng nhớ cấp cho mảng 2 chiều. $a1- dòng thứ i. $a2 - dòng thứ j.
- Đầu ra: $s2 - lưu giá trị word lấy từ vùng nhớ của phần tử Array2[i][j]
- Từ i và j tính ra địa chỉ vùng nhớ lưu giá trị phần tử Array2[i][j] = địa chỉ mảng Array2 + (i * tổng số cột + j) *4

3.10. Opt6

- Thiết lập giá trị lưu tạo Array[i][j] với i, j theo yêu cầu của người dùng.
- Đầu vào: như Opt5
- Như Opt5, có thể tính được vùng nhớ lưu giá trị phần tử -> gán giá trị theo yêu cầu vào vùng nhớ này

4. MÃ NGUỒN

Lưu trong file n06_g11_NgoHoangVu.asm đính kèm

```
.data
Menu:           .asciiz         "\n----------\n1/ Lay gia tri bien con tro\n2/ Lay dia chi bien con tro\n3/ Giai
phong bo nho da cap phat cho bien con tro\n4/ Tinh toan bo luong bo nho da cap phat\n5/ GetArray[i][j]\n6/
SetArray[i][j]\n7/ Thoat\n----------\nChon:_____?\n"
String:         .word   0       #Bien con tro, tro toi kieu ascii
Array2:         .word   0       #Bien con tro, tro toi mang 2 chieu kieu word
Success1:       .asciiz "\nOption "
Success2:       .asciiz         " thuc hien thanh cong!"
Opt3Message1:   .asciiz "\nLuong bo nho da cap phat: "
Opt3Message2:   .asciiz " byte!"
Input:          .asciiz "Name: Ngo Hoang Vu-MSSV: 20194721"     #Bien chua xau ky tu gia su duoc nhap tu
ban phim
.kdata
Sys_TheTopOfFree:       .word   1
Sys_MyFreeSpace:
.text
Main:
jal     SysInitMem

la      $a0, String
addi    $a1, $zero, 50      #50 phan tu
addi    $a2, $zero, 1       #moi phan tu 1 byte
jal     malloc              #Cap bo nho cho bien String

la      $a1,Input
lw      $a2,String
```

```
jal     strcpy          #Copy xau da nhap vao Bien da duoc cap phat bo nho

la      $a0, Array2
addi    $a1, $zero, 5   #5 dong
addi    $a2, $zero, 6   #6 cot
jal     malloc2         #Cap bo nho cho mang 2 chieu

la      $a0,Array2
li      $a3,1402
li      $a1,2
li      $a2,1
jal     SetArray2

la      $a0,Array2
li      $a3,10
li      $a1,0
li      $a2,0
jal     SetArray2

la      $a0,Array2
li      $a3,2001
li      $a1,4
li      $a2,3
jal     SetArray2

LoopMenu:
li      $v0, 4          #print Menu
la      $a0, Menu
syscall

li      $v0, 5          #doc lua chon Menu
syscall

beq     $v0, 1, Opt1
beq     $v0, 2, Opt2
beq     $v0, 3, Opt3
beq     $v0, 4, Opt4
beq     $v0, 5, Opt5
beq     $v0, 6, Opt6
beq     $v0, 7, EndMain
ContLoopMenu:
j       LoopMenu
EndMain:
li      $v0, 10
syscall

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

div     $t7, $t7, 4             #Thay doi kich thuoc de chia het cho 4
mfhi    $t0
beq     $t0, 0, NoChange #Neu size %4 == 0 -> branch
addi    $t7, $t7, 1
mul     $t7, $t7, 4             #else size bằng số nhỏ nhất chia hết cho 4 > $t7
j       EndGetArraySize
NoChange:
mul     $t7,$t7,4               #Khong thay doi
EndGetArraySize:

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
#--------------------
strcpy:
add     $t0,$zero,$zero       #$t0=i=0
L1:
add     $t1,$t0,$a1          #$t1 = $t0 + $a1 = i + y[0]
#    = dia chi cua y[i]
lb      $t2,0($t1)           #$t2 = gia tri tai $t1 = y[i]
add     $t3,$t0,$a2          #$t3 = $t0 + $a0 = i + x[0]
#    = adia chi x[i]
sb      $t2,0($t3)           #x[i]= $t2 = y[i]
beq     $t2,$zero,end_of_strcpy #if y[i]==0, exit
nop
addi    $t0,$t0,1           #$t0=$t0 + 1 <-> i=i+1
j       L1                  #char tiep theo
nop
end_of_strcpy:
jr      $ra
#--------------------
Opt1:
la      $a0, String
lw      $a1,0($a0)
li      $t0,0                   #Offset = 0
add     $a1,$a1,$t0             #Lay gia tri tai offset = $t0
lbu     $s0,0($a1)              #Tra ve gia tri byte

#li     $v0, 4                  #print String
#add    $a0, $a1,$zero
#syscall

li      $v0, 4                  #print Succes
la      $a0, Success1
syscall
```

```
li          $v0, 1
li          $a0, 1
syscall

li          $v0, 4
la          $a0, Success2
syscall

j           ContLoopMenu
#--------------------
Opt2:
la          $a0, String
lw          $s1,0($a0)              #Tra ve dia chi

li          $v0, 4                  #print Succes
la          $a0, Success1
syscall

li          $v0, 1
li          $a0, 2
syscall

li          $v0, 4
la          $a0, Success2
syscall

j           ContLoopMenu
#--------------------
Opt3:
la          $a0,Sys_TheTopOfFree
lw          $a1,0($a0)              #Lay dia chi dich -> $a1
la          $a0,Sys_MyFreeSpace     #Lay dia chi nguon
add         $a2,$a0,$zero           #$a2 = dia chi nguon
LoopFree:
beq         $a2,$a1,EndLoopFree     #$a2 == dia chi dich -> ket thuc giai phong
sw          $zero,0($a2)            #Giai phong du lieu o word hien tai
addi        $a2,$a2,4               #$a2-> dia chi word tiep theo
j           LoopFree
EndLoopFree:
la          $a0,Sys_MyFreeSpace
sw          $a0,Sys_TheTopOfFree    #Sys_TopOfFree chua dia chi dau tien cua vung nho con trong

li          $v0, 4                  #print Succes
la          $a0, Success1
syscall

li          $v0, 1
li          $a0, 3
syscall

li          $v0, 4
la          $a0, Success2
```

```
syscall

j	ContLoopMenu
#--------------------
Opt4:
la	$a0,Sys_TheTopOfFree
lw	$a1,0($a0)
la	$a0,Sys_MyFreeSpace
add	$a2,$a0,$zero
add	$t0,$zero,$zero
LoopCount:
beq	$a2,$a1,EndLoopCount
addi	$a2,$a2,4
addi	$t0,$t0,4
j	LoopCount
EndLoopCount:
li	$v0, 4			#print Message
la	$a0, Opt3Message1
syscall

li	$v0, 1
add	$a0,$t0,$zero
syscall

li	$v0, 4
la	$a0, Opt3Message2
syscall

li	$v0, 4			#print Succes
la	$a0, Success1
syscall

li	$v0, 1
li	$a0, 4
syscall

li	$v0, 4
la	$a0, Success2
syscall

j	ContLoopMenu
#--------------------
malloc2:
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
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
mul	$t7, $t7, 4		#Nhan so phan tu mang voi 4 byte trong 1 word
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
add	$s6,$a1,$zero		#Luu lai so han va so cot
add	$s7,$a2,$zero
```

```
jr      $ra
#--------------------
GetArray2:
lw      $t0,0($a0)              #Dia chi cua vung nho cua Array2
mul     $t1,$a1,$s7             #$t1 = i *so cot
add     $t1,$t1,$a2             #$t1 = i * so cot + j = index cua mang 2 chieu
mul     $t1,$t1,4
add     $t0,$t0,$t1             #$t0 = Array2[0][0] + offsdet
lw      $s2,0($t0)              #lay gia tri tu trong mang luu trong $s2
jr      $ra
#--------------------
Opt5:
la      $a0,Array2
li      $a1,2
li      $a2,1
jal     GetArray2

li      $v0, 4                  #print Succes
la      $a0, Success1
syscall

li      $v0, 1
li      $a0, 5
syscall

li      $v0, 4
la      $a0, Success2
syscall

j       ContLoopMenu
#--------------------
SetArray2:
lw      $t0,0($a0)              #Dia chi cua vung nho cua Array2
mul     $t1,$a1,$s7             #$t1 = i *so cot
add     $t1,$t1,$a2             #$t1 = i * so cot + j = index cua mang 2 chieu
mul     $t1,$t1,4
add     $t0,$t0,$t1             #$t0 = Array2[0][0] + offsdet
sw      $a3,0($t0)              #luu gia tri vao vung nho cua phan tu trong mang
jr      $ra
#--------------------
Opt6:
la      $a0,Array2
li      $a3,194721
li      $a1,2
li      $a2,3
jal     SetArray2

li      $v0, 4                  #print Succes
la      $a0, Success1
syscall

li      $v0, 1
li      $a0, 6
```

```
syscall

li      $v0, 4
la      $a0, Success2
syscall

j       ContLoopMenu
```