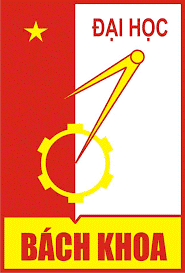
**TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI**

**VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG**



Final Project

Computer Architecture

**GVHD: Lê Bá Vui**

**Nhóm: 03**

**Thành Viên: Phạm Ngọc Bảo Anh - 20176682**

**Nguyễn Khắc Thắng - 20176869**

**Mã Lớp: 113834**

MỤC LỤC

[LỜI MỞ ĐẦU 3](#_Toc43682632)

[I. Bài 3: 4](#_Toc43682634)

[1. Đề bài 4](#_Toc43682635)

[2. Phân tích đề bài 4](#_Toc43682636)

[3. Phân tích cách thực hiện 4](#_Toc43682637)

[o Chương trình chính được chia làm 3 phần: 4](#_Toc43682638)

[o Cách chạy chương trình: 4](#_Toc43682639)

[4. Mã nguồn 4](#_Toc43682640)

[5. Hình ảnh kết quả mô phỏng 9](#_Toc43682641)

[II. Bài 6: 10](#_Toc43682642)

[1. Đề bài 10](#_Toc43682643)

[2. Phân tích đề bài 10](#_Toc43682644)

[3. Phân tích cách làm 11](#_Toc43682645)

[4. Mã nguồn 11](#_Toc43682646)

[5. Hình ảnh kết quả mô phỏng 21](#_Toc43682647)

[o Menu lựa chọn: 21](#_Toc43682648)

[o Cấp phát bộ nhớ cho biến con trỏ CharPtr 21](#_Toc43682649)

[o Cấp phát bộ nhớ cho biến con trỏ BytePtr 22](#_Toc43682650)

[o Cấp phát bộ nhớ cho biến con trỏ WordPtr 22](#_Toc43682651)

[o Lấy giá trị của các biến con trỏ 22](#_Toc43682652)

[o Lấy địa chỉ của các biến con trỏ 22](#_Toc43682653)

[o Copy xâu 22](#_Toc43682654)

[o Tính toán lượng bộ nhớ đã cấp phát 23](#_Toc43682655)

[o Cấp phát bộ nhớ cho mảng 2 chiều 23](#_Toc43682656)

[o Thiết lập giá trị các phần tử cho mảng 2 chiều 23](#_Toc43682657)

[o Lấy giá trị của các phần tử mảng 2 chiều 23](#_Toc43682658)

# **LỜI MỞ ĐẦU**

Nhóm gồm 2 thành viên Phạm Ngọc Bảo Anh và Nguyễn Khắc Thắng

Bản báo cáo khái quát quá trình thực hiện 2 bài tập lớn là bài 3 và bài

8 gồm các nội dung chính:

❖ Đề bài.

❖ Phân tích đề bài.

❖ Cách làm.

❖ Mã nguồn.

❖ Hình ảnh kết quả.

Bản báo cáo sẽ không tránh khỏi những sai sót. Nhóm rất mong nhận

được ý kiến góp ý của thầy giáo và các bạn.

Chúng em chân thành cảm ơn.

BÁO CÁO MINI PROJECT MÔN THỰC HÀNH KIẾN TRÚC MÁY TÍNH

## Nhóm 3: Phạm Ngọc Bảo Anh - Nguyễn Khắc Thắng

# **Bài 3:**

*Kiểm tra tốc độ và độ chính xác khi gõ văn bản.*

1. Đề bài

Chương trình sau sẽ đo tốc độ gõ bàn phím và hiện thị kết quả bằng 2 đèn led 7 đoạn.  
Nguyên tắc:

* + Cho một đoạn văn bản mẫu, cố định sẵn trong mã nguồn. Ví dụ “bo mon ky thuat may tinh”
  + Sử dụng bộ định thời Timer (trong bộ giả lập Digital Lab Sim) để tạo ra khoảng thời gian để đo. Đây là thời gian giữa 2 lần ngắt, chu kì ngắt.
  + Trong khoảng thời gian đó, người dùng nhập các kí tự từ bán phím. Ví dụ “bo mOn ky 5huat may tinh”. Chương trình cần phải đếm số kí tự đúng(trong ví dụ trên thi người dùng gõ sai chữ O và 5) mà người dùng đã gõ và hiển thị lên các đèn led.

1. Phân tích đề bài
   * Input: Cho sẵn một đoạn mã nguồn mẫu.
   * Trong một khoảng thời gian cố định, người dùng nhập kí tự từ bàn phím.
   * Output: Hiển thị ra số lượng kí tự gõ đúng ra 2 đèn led 7 đoạn.
2. Phân tích cách thực hiện

* Chương trình chính được chia làm 3 phần:
  + Nhập chuỗi từ MMIO Simulator --> lưu vào buffer
  + Xử lý với số lượng chu kỳ ngắt
  + Tính số lượng ký tự đúng --> hiển thị kết quả ra 2 đèn led 7 đoạn
* Cách chạy chương trình:
  + Connect Digital Lab Sim và Keyboard and Display MMIO Simulator với MIPS
  + Nhập ký tự từ bàn phím --> nhấn enter để kết thúc
  + Xem tốc độ nhập ở console và số ký tự đúng ở 2 đèn led 7 đoạn (Digital Lab Sim)
* Cách làm:
  + Sử dụng một mảng để lưu mã nguồn mẫu (mảng string), một mảng lưu các kí tự gõ vào (mảng buffer)
  + Khi nhập: dùng một biến để đếm số lượng các kí tự đã nhập vào. Nếu gặp phím xóa thì giảm biến này đi. Khi ấn enter thì kết thúc việc gõ.
  + So sánh 2 mảng string và mảng buffer. Dùng một biến để đếm số kí tự đúng.
  + Mã C:

int count = 0;

for(int i = 0; i < strlen(string); i++){

if(string[i] == buffer[i])

count++;

}

1. Mã nguồn

# MMIO Simulator

.eqv KEY\_CODE 0xFFFF0004

.eqv KEY\_READY 0xFFFF0000

.eqv DISPLAY\_CODE 0xFFFF000C # ASCII code to show, 1 byte

.eqv DISPLAY\_READY 0xFFFF0008 # =1 if the display has already to do

# Auto clear after sw

# Led 7 doan

.eqv SEVENSEG\_RIGHT 0xFFFF0010 # Dia chi cua den led 7 doan phai

.eqv SEVENSEG\_LEFT 0xFFFF0011 # Dia chi cua den led 7 doan trai

.eqv LIMIT\_TIME 100000 # Gioi han thoi gian 100000 chu ki ngat

.data

string: .asciiz "hello nhom 3 nguyen khac thang"

buffer: .space 100

count\_true: .asciiz "\nso ky tu dung la: "

speed: .asciiz "\ntoc do: "

don\_vi: .asciiz " time/char"

.text

li $t0, 1

sb $t0, 0xFFFF0013($zero) # Set bit tai 0xFFFF0013 khac 0 de kich hoat ngat

# Sau 30 cau lenh se ngat 1 lan

li $t0, 0 # Bien dem i = 0

li $s0, 0 # $s0 Dem so chu ky ngat

li $t8, 0 # Bien dieu kien dem so chu ky

# $t8 = 0: Chua dem, $t8 = 1: Bat dau dem

in\_string\_mau:

WaitForDis:

lw $t2, DISPLAY\_READY # Vong lap cho display san sang

beq $t2, $zero, WaitForDis #

nop

lb $t1, string($t0) # Doc string[i]

beq $t1, '\0', doc\_ban\_phim # if string[i] == NULL then break

sw $t1, DISPLAY\_CODE # Hien thi string[i]

add $t0, $t0, 1 # i++

j in\_string\_mau # Lap lai vong lap

doc\_ban\_phim:

li $t0, 0 # Bien dem i = 0

li $s0, 0 # $s0 Dem so chu ky ngat

li $t8, 0 # Bien dieu kien dem so chu ky

# $t8 = 0: Chua dem, $t8 = 1: Bat dau dem

WaitForKey:

lw $k1, KEY\_READY # Vong lap cho ban phim san sang

beqz $k1, WaitForKey #

nop

li $t8, 1 # Dat $t8 = 1, Bat dau dem cac chu ky ngat

lw $k0, KEY\_CODE # Doc ky tu tu ban phim

beq $k0, 8, backspace # Kiem tra nut xoa

beq $k0, 10, exit # Ket thuc go neu nguoi dung nhan enter

sb $k0, buffer($t0) # Luu lai phim vua go

add $t0, $t0, 1 # i++

j WaitForKey

backspace:

beqz $t0, WaitForKey # if i == 0 {lap lai vong lap}

sb $zero, buffer($t0) # else{

# buffer[i] = NULL;

add $t0, $t0, -1 # i--;}

sb $zero, buffer($t0) # Ky tu vua xoa cung gan bang null

j WaitForKey # Lap lai vong lap

exit:

li $v0, 4 #

la $a0, buffer #

syscall # In xau thu duoc tu ban phim

li $v0, 4 #

la $a0, speed #

syscall # In xau speed

mtc1 $t0, $f1 # Convert $t0(integer) -> $f1(float): so luong phim da nhap

mtc1 $s0, $f2 # Convert $s0(integer) -> $f2(float): so luong time (so chu ky)

div.s $f12, $f2, $f1 # Toc do trung binh (so chu ky ngat/1 ky tu)

li $v0, 2

syscall

li $v0, 4 #

la $a0, don\_vi #

syscall # In message don\_vi

li $s4, 0 # dung: $s4 dem so ky tu go dung

li $t0, 0 # i: $t0 bien dem

string\_cmp:

lb $t1, string($t0) # $t1 = string[i]

lb $t2, buffer($t0) # $t2 = buffer[i]

beqz $t1, mess # if(string[i] == NULL) break;

bne $t1, $t2, next\_char # If (String[i] == buffer[i])

add $s4, $s4, 1 # dung++;

next\_char:

add $t0, $t0, 1 # i++;

j string\_cmp # Lap lai vong lap

mess:

li $v0, 4 #

la $a0, count\_true #

syscall # In message count\_true

move $a0, $s4 #

li $v0, 1 # In so ky tu go dung ra man hinh console

syscall #

li $t4, 10

div $s4, $t4 # Chia so ki tu go dung cho 10

mflo $t0 # Lay so thuong ( Hang chuc)

jal SET\_DATA\_FOR\_7SEG #

move $a1, $a0 # Dat du lieu hang chuc cho led

mfhi $t0 # Lay so du ( Hang don vi)

jal SET\_DATA\_FOR\_7SEG # Dat du lieu hang don vi cho led

jal SHOW\_7SEG\_RIGHT # Hien thi hang don vi led phai

jal SHOW\_7SEG\_LEFT # Hien thi hang chuc led trai

li $v0, 10 #

syscall # Goi thu tuc ket thuc chuong trinh

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

# Function SHOW\_7SEG\_RIGHT : turn on/off the 7seg

# @param [in] $a0 value to shown

# remark $t0 changed

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

SHOW\_7SEG\_RIGHT:

sb $a0, SEVENSEG\_RIGHT # assign new value

jr $ra

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

# Function SHOW\_7SEG\_LEFT : turn on/off the 7seg

# @param [in] $a1 value to shown

# remark $t0 changed

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

SHOW\_7SEG\_LEFT:

sb $a1, SEVENSEG\_LEFT # assign new value

jr $ra

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

# Function SET\_DATA\_FOR\_7SEG : Chuyen du lieu he 10 sang kieu ma hoa LED

# @param [in] $t0 gia tri he 10

# @return $a0 Ma hoa tung vung hien thi den led

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

SET\_DATA\_FOR\_7SEG:

beq $t0, 0, \_\_0

beq $t0, 1, \_\_1

beq $t0, 2, \_\_2

beq $t0, 3, \_\_3

beq $t0, 4, \_\_4

beq $t0, 5, \_\_5

beq $t0, 6, \_\_6

beq $t0, 7, \_\_7

beq $t0, 8, \_\_8

beq $t0, 9, \_\_9

nop

\_\_0: li $a0, 0x3f

j END\_\_F

\_\_1: li $a0, 0x06

j END\_\_F

\_\_2: li $a0, 0x5B

j END\_\_F

\_\_3: li $a0, 0x4f

li $s2, 3

j END\_\_F

\_\_4: li $a0, 0x66

j END\_\_F

\_\_5: li $a0, 0x6D

j END\_\_F

\_\_6: li $a0, 0x7d

j END\_\_F

\_\_7: li $a0, 0x07

j END\_\_F

\_\_8: li $a0, 0x7f

j END\_\_F

\_\_9: li $a0, 0x6f

j END\_\_F

END\_\_F:

jr $ra

#+++++++++++++++++++++++++++++++++++++++++

#

# XU LY NGAT CHUONG TRINH

#

#+++++++++++++++++++++++++++++++++++++++++

.ktext 0x80000180

IntSR:

move $t9, $at # Luu lai gia tri thanh ghi $at

mfc0 $v0, $13 # Kiem tra ma nguyen nhan ngat

bne $v0, 1024, exit # Ma ngat 1024, bo qua ma ngat do Counter cua Digit Lab Sim

# Ma ngat khac, la Loi => Ket thuc chuong trinh

add $s0, $s0, $t8 # Tang bien dem so chu ky ngat

# Bien dem $s0 chi tang khi $t8 == 1 (Khi bat dau go)

sge $v0, $s0, LIMIT\_TIME # Thoat neu dat so chu ky ngat toi da

bnez $v0, exit #

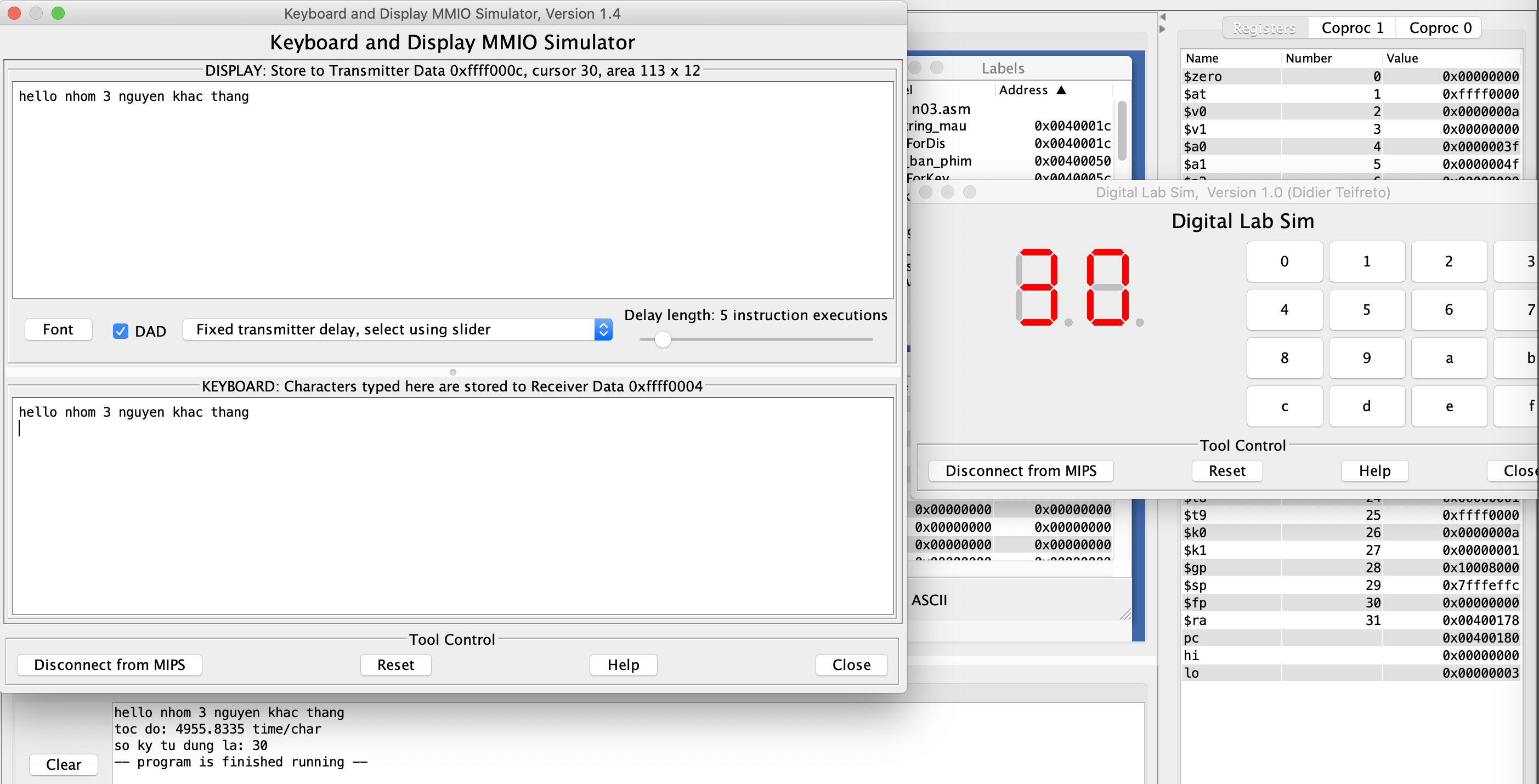
nop

move $at, $t9 # Khoi phuc thanh ghi $at

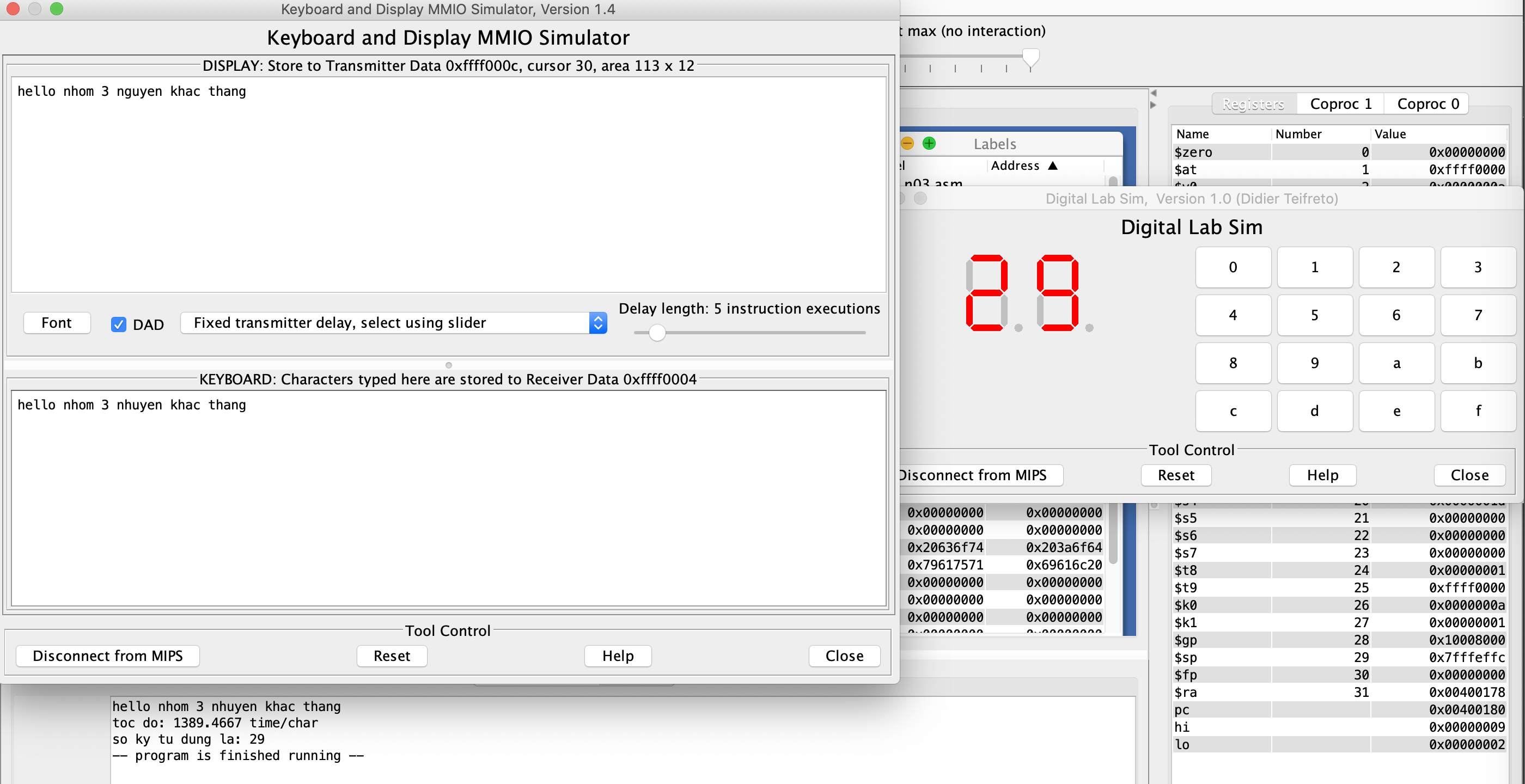
return: eret # Quay lai vi tri ngat

1. Hình ảnh kết quả mô phỏng

* Trường hợp 1: Chuỗi nhập vào đúng toàn bộ



* Trường hợp 2: Chuỗi nhập vào là “hello nhom 3 nhuyen khac thang” (bị sai chữ g 🡪 chữ h)



# **Bài 6:**

*Hàm cấp phát bộ nhớ Malloc()*

1. Đề bài

Chương trình cho bên dưới là hàm malloc(), kèm theo đó là ví dụ minh họa, được viết bằng hợp ngữ MIPS, để cấp phát bộ nhớ cho một biến con trỏ nào đó. Hãy đọc chương trình và hiểu rõ nguyên tắc cấp phát bộ nhớ động.

Trên cơ sở đó, hãy hoàn thiện chương trình như sau. Lưu ý, ngoài viết các hàm đó, cần viết thêm một số ví dụ minh họa để thấy việc sử dụng hàm đó như thế nào.

1. Việc cấp phát bộ nhớ kiểu word/mảng word có 1 lỗi, đó là chưa bảo đảm qui tắc địa chỉ của kiểu word phải chia hết cho 4. Hãy khắc phục lỗi này.
2. Viết hàm lấy giá trị Word /Byte của biến con trỏ (tương tự như \*CharPtr, \*BytePtr, \*WordPtr).
3. Viết hàm lấy địa chỉ biến con trỏ (tương tự như &CharPtr, &BytePtr, \*WordPtr).
4. Viết hàm thực hiện copy 2 con trỏ xâu kí tự (Xem ví dụ về CharPtr).
5. Viết hàm tính toàn bộ lượng bộ nhớ đã cấp phát cho các biến động.
6. Hãy viết hàm Malloc2 để cấp phát cho mảng 2 chiều kiểu .word với tham số vào gồm:
   1. Địa chỉ đầu của mảng
   2. Số dòng
   3. Số cột
7. Tiếp theo câu 6, hãy viết 2 hàm GetArray[i][j] và SetArray[i][j] để lấy/thiết lập giá trị cho phần tử ở dòng I cột j của mảng.
8. Phân tích đề bài
   * Hiển thị ra màn hình menu các yêu cầu của đề bài.
   * Hiểu rõ nguyên tắc cấp phát bộ nhớ động.
   * Viết các hàm cấp phát bộ nhớ động (malloc) cho các kiểu biến con trỏ.
9. Phân tích cách làm
   * Sử dụng 1 thanh ghi để lưu địa chỉ của các biến con trỏ, nếu không phải kiểu word, sẽ ép giá trị thanh ghi đó tăng lên số chia hết cho 4
   * Đối với từng hàm riêng biệt cần chú ý lượng bộ nhớ cần cấp phát cho mỗi kiểu biến con trỏ là khác nhau.
   * Sử dụng cách thao tác stack của asembly để thao tác với mảng theo yêu cầuđề bài.  (trong hàm Malloc2).
   * Để thực hiện copy xâu ký tự ta sẽ copy từng ký tự từ xâu mà người dùng nhập vào kết hợp với việc cấp phát bộ nhớ cho biến con trỏ kiểu Char.
   * Để cấp phát bộ nhớ cho mảng hai chiều ta sẽ đưa về dạng cấp phát bộ nhớ cho mảng một chiều và sử dụng hàm malloc.
10. 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 mang kieu Word

ArrayPtr: .word 0 # Bien con tro, tro toi mang hai chieu

CharPtr1: .word 0 # Bien con tro, dung trong yeu cau copy xau

CharPtr2: .word 0 # Bien con tro, dung trong yeu cau copy xau

Newline: .asciiz "\n" # Ky tu xuong dong

row: .word 1

col: .word 1

menu: .asciiz "\n1. Malloc CharPtr.\n2. Malloc BytePtr.\n3. Malloc WordPtr.\n4. Tra ve gia tri cua cac bien con tro.\n5. Tra ve dia chi cua cac bien con tro.\n6. Copy 2 con tro xau ki tu.\n7. Tinh toan luong bo nho da cap phat cho cac bien dong (malloc).\n8. Malloc2 (Mang hai chieu kieu .word).\n9. Set Array[i][j].\n10. Get Array[i][j].\nThoat neu khac 1-10"

char\_str: .asciiz "\nNhap so phan tu cua mang kieu Char: "

byte\_str: .asciiz "\nNhap so phan tu cua mang kieu Byte: "

word\_str: .asciiz "\nNhap so phan tu cua mang kieu Word: "

copy\_str: .asciiz "\nXau da duoc copy: "

nb\_row: .asciiz "\nNhap so hang cua mang: "

nb\_col: .asciiz "\nNhap so cot cua mang: "

input\_row: .asciiz "\nNhap i (so thu tu cua dong): "

input\_col: .asciiz "\nNhap j (so thu tu cua cot): "

input\_val: .asciiz "\nNhap gia tri gan cho phan tu cua mang: "

output\_val: .asciiz "\nGia tri tra ve: "

address\_str: .asciiz "\nDia chi cua bien con tro CharPtr | BytePtr | WordPtr | ArrayPtr la: "

value\_str: .asciiz "\nGia tri cua bien con tro CharPtr | BytePtr | WordPtr | ArrayPtr la: "

malloc\_str: .asciiz "\nBo nho da cap phat: "

bytes\_str: .asciiz " bytes"

input\_str: .asciiz "\nNhap vao mot xau ky tu: "

malloc\_success: .asciiz "\nCap phat bo nho thanh cong. Mang bat dau tai dia chi: "

mal\_error: .asciiz "\nError: So hang hoac so cot phai nho hon 1000"

bound\_error: .asciiz "\nError: Ngoai vung bo nho cho phep cua mang"

null\_error: .asciiz "\nError: Chua khoi tao mang"

overflow\_error: .asciiz "\nError: Gia tri input qua lon (> 2000)"

zero\_error: .asciiz "\nError: Gia tri input phai khac 0"

string\_copy: .space 100 # Xau copy

.kdata

# Luu gia tri la dia chi dau tien cua vung nho con trong

Sys\_TheTopOfFree: .word 1

# Vung khong gian tu do, dung de cap phat bo nho cho cac bien con tro

Sys\_MyFreeSpace:

.text

# Khoi tao vung nho cap phat dong

jal SysInitMem

main:

print\_menu:

la $a0, menu

jal integer\_input # get integer input value from user

move $s0, $a0 # switch option

beq $s0, 1, option1

beq $s0, 2, option2

beq $s0, 3, option3

beq $s0, 4, option4

beq $s0, 5, option5

beq $s0, 6, option6

beq $s0, 7, option7

beq $s0, 8, option8

beq $s0, 9, option9

beq $s0, 10, option10

j end

option1: # Malloc Char

la $a0, char\_str

jal integer\_input

jal check\_input # kiem tra gia tri input (0 < input < 2000)

move $a1, $a0 # Luu input (so phan tu cua mang) vao $a1

la $a0, CharPtr # Luu dia chi cua CharPtr vao $a0

li $a2, 1 # Kich thuoc Char = 1 byte

jal malloc # Cap phat bo nho

move $s0, $v0 # Luu gia tri tra ve cua ham malloc vao $s0

la $a0, malloc\_success # Thong bao cap phat thanh cong

li $v0, 4 # print string service

syscall

move $a0, $s0 # Chuyen gia tri tu $s0 vao $a0

li $v0, 34 # print integer in hexadecimal service

syscall # in ra gia tri integer cua $a0

j main

option2: # Malloc Byte

la $a0, byte\_str

jal integer\_input

jal check\_input

move $a1, $a0 # Luu input (so phan tu cua mang) vao $a1

la $a0, BytePtr # Luu dia chi cua BytePtr vao $a0

li $a2, 1 # Kich thuoc Byte = 1 byte

jal malloc # Cap phat bo nho

move $s0, $v0 # Luu gia tri tra ve cua ham malloc vao $s0

la $a0, malloc\_success # Thong bao cap phat thanh cong

li $v0, 4 # print string service

syscall

move $a0, $s0 # Chuyen gia tri tu $s0 vao $a0

li $v0, 34 # print integer in hexadecimal service

syscall

j main

option3: # Malloc Word

la $a0, word\_str

jal integer\_input

jal check\_input

move $a1, $a0 # Luu input (so phan tu cua mang) vao $a1

la $a0, WordPtr # Luu dia chi cua WordPtr vao $a0

li $a2, 4 # Kich thuoc Word = 4 bytes

jal malloc # Cap phat bo nho

move $s0, $v0 # Luu gia tri tra ve cua ham malloc vao $s0

la $a0, malloc\_success # Thong bao cap phat thanh cong

li $v0, 4 # print string service

syscall

move $a0, $s0 # Chuyen gia tri tu $s0 vao $a0

li $v0, 34 # print integer in hexadecimal service

syscall

j main

option4:

la $a0, value\_str

li $v0, 4 # print string service

syscall

li $a0, 0

jal Ptr\_val # Lay gia tri cua CharPtr

jal print\_value

li $a0, 1

jal Ptr\_val # Lay gia tri cua BytePtr

jal print\_value

li $a0, 2

jal Ptr\_val # Lay gia tri cua WordPtr

jal print\_value

li $a0, 3

jal Ptr\_val # Lay gia tri cua ArrayPtr

jal print\_value

j main

option5:

la $a0, address\_str

li $v0, 4 # print string service

syscall

li $a0, 0 # Lay dia chi cua CharPtr

jal Ptr\_addr

jal print\_value

li $a0, 1 # Lay dia chi cua BytePtr

jal Ptr\_addr

jal print\_value

li $a0, 2 # Lay dia chi cua WordPtr

jal Ptr\_addr

jal print\_value

li $a0, 3 # Lay dia chi cua ArrayPtr

jal Ptr\_addr

jal print\_value

j main

option6:

input\_string:

li $v0, 54 # InputDialogString

la $a0, input\_str

la $a1, string\_copy # Dia chi luu string dung de copy

li $a2, 100 # So ki tu toi da co the doc duoc = 100

syscall

la $a1, string\_copy # Load lai 1 lan

la $s1, CharPtr1 # Load dia chi cua CharPtr1

sw $a1, 0($s1) # Luu string vua nhap vao CharPtr1

copy:

la $a0, CharPtr2 # Load dia chi cua CharPtr2

la $t9, Sys\_TheTopOfFree

lw $t8, 0($t9) # Lay dia chi dau tien con trong

sw $t8, 0($a0) # Cat dia chi do vao bien con tro CharPtr2

lw $t4, 0($t9) # Dem so luong ki tu trong string

lw $t1, 0($s1) # Load gia tri con tro CharPtr1

lw $t2, 0($a0) # Load gia tri con tro CharPtr2

copy\_loop:

lb $t3, ($t1) # Load 1 ki tu (tren cung) tai $t1 vao $t3

sb $t3, ($t2) # Luu 1 ki tu cua $t3 vao o nho tai dia chi $t2

addi $t4, $t4, 1 # $t4 : dem so luong ki tu string

addi $t1, $t1, 1 # Chuyen sang dia chi ki tu tiep theo cua CharPtr1

addi $t2, $t2, 1 # Chuyen sang dia chi ki tu tiep theo cua CharPtr2

beq $t3, '\0', exit\_copy # Check null => end string

j copy\_loop

exit\_copy:

la $a0, copy\_str

li $v0, 4 # print string service

syscall

sw $t4,($a0) # Luu so byte(s) dung de luu string

la $a2, CharPtr2 # Load dia chi CharPtr2 vao $a2

lw $a0, ($a2) # Luu xau da copy tu $a0 vao CharPtr2

li $v0, 4 # In ra gia tri CharPtr2

syscall

la $a0, Newline

syscall

j main

option7: # Tinh luong bo nho da cap phat

la $a0, malloc\_str

li $v0, 4 # print string service

syscall

jal MemoryCount # tinh luong bo nho da cap phat va luu vao $v0

move $a0, $v0

li $v0, 1 # print integer

syscall

la $a0, bytes\_str

li $v0, 4 # print string service

syscall

j main

option8: # Cap phat bo nho cho mang 2 chieu Malloc2

la $a0, nb\_row

jal integer\_input # Nhap vao so hang

move $s0, $a0

la $a0, nb\_col

jal integer\_input # Nhap vao so cot

move $a1, $s0 # malloc2 input\_row parameter

move $a2, $a0 # malloc2 input\_col parameter

la $a0, ArrayPtr

jal Malloc2 # Cap phat bo nho

move $s0, $v0 # save return value of malloc

la $a0, malloc\_success

li $v0, 4

syscall

move $a0, $s0

li $v0, 34

syscall # In ra gia tri mang vua nhap

j main

option9: # Set[i][j]

la $a0, ArrayPtr

lw $s7, 0($a0)

beqz $s7, nullptr # if \*ArrayPtr==0 error null pointer

la $a0, input\_row

jal integer\_input # get row

move $s0, $a0

la $a0, input\_col

jal integer\_input # get col

move $s1, $a0

la $a0, input\_val

jal integer\_input # get val

move $a3, $a0

move $a1, $s0

move $a2, $s1

move $a0, $s7

jal SetArray

j main

option10: # Get[i][j]

la $a0, ArrayPtr

lw $s1, 0($a0)

beqz $s1, nullptr # if \*ArrayPtr == 0 return error null pointer

la $a0, input\_row

jal integer\_input # get row

move $s0, $a0 # $s0 = so hang

la $a0, input\_col

jal integer\_input # get col

move $a2, $a0 # $a2 = so cot

move $a1, $s0 # $a1 = so hang

move $a0, $s1 # $a0 = gia tri thanh ghi

jal GetArray

move $s0, $v0 # save return value of GetArray

la $a0, output\_val

li $v0, 4

syscall

move $a0, $s0

li $v0, 34

syscall

j main

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

# 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

bne $a2, 4, skip # Neu khong phai kieu Word thi nhay sang skip

addi $t8, $t8, 3

andi $t8, $t8, 0xfffffffc # gia tri luu tai $t8 luon la 1 so chia het cho 4

skip:

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

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

# Ham cap phat bo nho dong cho mang 2 chieu

# Idea: Dua ve cap phat bo nho cho mang 1 chieu co ROW \* COL phan tu, su dung lai ham malloc

# @param [in/out] $a0 Chua dia chi cua bien con tro can cap phat

# Khi ham ket thuc, dia chi vung nho duoc cap phat se luu tru vao bien con tro

# @param [in] $a1 so hang

# @param [in] $a2 so cot

# @return $v0 Dia chi vung nho duoc cap phat

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

Malloc2:

addiu $sp, $sp, -4 # them 1 phan tu vao stack

sw $ra, 4($sp) # push $ra

bgt $a1, 1000, mal\_err # kiem tra loi so luong

bgt $a2, 1000, mal\_err # phan tu hang (cot) qua lon

la $s0, row

sw $a1, 0($s0) # luu so hang vao row

sw $a2, 4($s0) # luu so cot vao col

mul $a1, $a1, $a2 # tra ve so phan tu cua Array

li $a2, 4 # kich thuoc kieu Word = 4 bytes

jal malloc

lw $ra, 4($sp)

addiu $sp, $sp, 4 # pop $ra

jr $ra

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

# gan gia tri cua phan tu trong mang hai chieu

# @param [in] $a0 Chua dia chi bat dau mang

# @param [in] $a1 hang (i) # @param [in] $a2 cot (j)

# @param [in] $a3 gia tri gan

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

SetArray:

la $s0, row # $s0 = dia chi so hang

lw $s1, 0($s0) # $s1 so hang

lw $s2, 4($s0) # $s2 so cot

bge $a1, $s1, bound\_err # Neu so cot vuot qua gioi han => error

bge $a2, $s2, bound\_err # Neu so hang vuot qua gioi han => error

mul $s0, $s2, $a1

addu $s0, $s0, $a2 # $s0 = i\*col +j

sll $s0, $s0, 2

addu $s0, $s0, $a0 # $s0 = \*array + (i\*col +j)\*4

sw $a3, 0($s0)

jr $ra

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

# lay gia tri cua trong mang

# @param [in] $a0 Chua dia chi bat dau mang

# @param [in] $a1 hang (i)

# @param [in] $a2 cot (j)

# @return $v0 gia tri tai hang a1 cot a2 trong mang

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

GetArray:

la $s0, row # $s0 = dia chi so hang

lw $s1, 0($s0) # $s1 so hang

lw $s2, 4($s0) # $s2 so cot

bge $a1, $s1, bound\_err # Neu so cot vuot qua gioi han => error

bge $a2, $s2, bound\_err # Neu so hang vuot qua gioi han => error

mul $s0, $s2, $a1

addu $s0, $s0, $a2 # $s0= i\*col +j

sll $s0, $s0, 2

addu $s0, $s0, $a0 # $s0 = \*array + (i\*col +j)\*4

lw $v0, 0($s0)

jr $ra

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

# Ham lay gia tri cua cac bien con tro

# @param [in] $a0 {0: char ; 1: byte ; 2: word ; 3: array}

# @return $v0 gia tri bien con tro

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

Ptr\_val:

la $t0, CharPtr # Luu dia chi bien con tro CharPtr vao $t0

sll $t1, $a0, 2 # CharPtr, BytePtr, WordPtr nam lien tiep nhau

add $t0, $t0, $t1 # $t0 luu dia chi cua CharPtr/BytePtr/WordPtr/ArrayPtr

lw $v0, 0($t0) # lay gia tri luu tai bien con tro va luu vao $v0 (gia tri tra ve)

jr $ra

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

# Ham lay dia chi cua cac bien con tro

# @param [in] $a0 {0: char ; 1: byte ; 2: word ; 3: array}

# @return $v0 dia chi bien con tro

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

Ptr\_addr:

la $t0, CharPtr # Luu dia chi bien con tro CharPtr vao $t0

sll $t1, $a0, 2 # CharPtr, BytePtr, WordPtr nam lien tiep nhau

add $v0, $t0, $t1 # $v0 luu dia chi cua CharPtr/BytePtr/WordPtr/ArrayPtr

jr $ra

print\_value: # in ra gia tri $v0

move $a0, $v0

li $v0, 34 # print integer in hexadecimal service

syscall

li $a0, ';'

li $v0, 11 # print character service

syscall

jr $ra

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

# Tinh tong luong bo nho da cap phat

# @param: none

# @return $v0 chua luong bo nho da cap phat

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

MemoryCount:

la $t9, Sys\_TheTopOfFree

lw $t9, 0($t9) # $t9 = Gia tri tai dia chi con trong dau tien

la $t8, Sys\_MyFreeSpace # Sys\_MyFreeSpace luon co dinh la thanh ghi ngay sau Sys\_TheTopOfFree

sub $v0, $t9, $t8 # Tra ve gia tri $v0 = luong bo nho da cap phat

jr $ra

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

# Wrapper for syscall 51 (InputDialogInt)

# repeat if status value !=0

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

integer\_input:

move $t9, $a0

li $v0, 51

syscall

beq $a1, 0, doneIn

beq $a1, -2, end

move $a0, $t9

j integer\_input

doneIn:

jr $ra

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

# kiem tra gia tri nhap vao >0 va <2000

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

check\_input:

bge $a0, 2000, too\_big

beqz $a0, zero\_err

jr $ra

too\_big:

la $a0, overflow\_error

j error

zero\_err:

la $a0, zero\_error

j error

mal\_err: # In ra thong bao loi so luong malloc

la $a0, mal\_error

j error

bound\_err: # In ra thong bao loi chi so vuot ngoai pham vi

la $a0, bound\_error

j error

nullptr: # In ra thong bao loi con tro rong ( null)

la $a0, null\_error

j error

error:

li $v0, 4 # In ra thong bao loi

syscall

j main

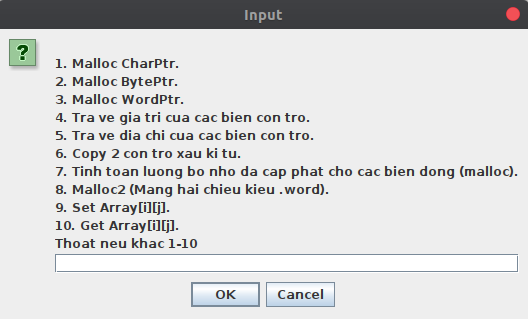
end:

li $v0, 10 # Terminate

syscall

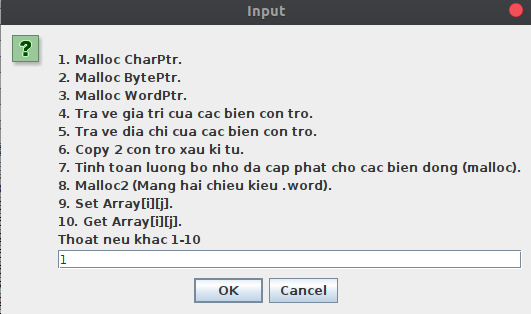
1. Hình ảnh kết quả mô phỏng

* Menu lựa chọn:

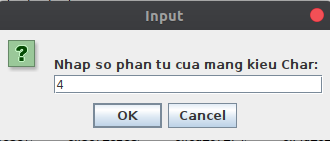


* Cấp phát bộ nhớ cho biến con trỏ CharPtr

Lựa chọn 1:



Cấp phát bộ nhớ cho 4 phần tử:



Kết quả:



* Cấp phát bộ nhớ cho biến con trỏ BytePtr

Lựa chọn 2 🡪 Cấp phát bộ nhớ cho 4 phần tử

Kết quả:



* Cấp phát bộ nhớ cho biến con trỏ WordPtr

Lựa chọn 3 🡪 Cấp phát bộ nhớ cho 1 phần tử

Kết quả:



* Lấy giá trị của các biến con trỏ

Lựa chọn 4 🡪 Kết quả:



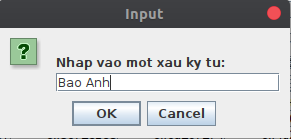
* Lấy địa chỉ của các biến con trỏ

Lựa chọn 5 🡪 kết quả:



* Copy xâu

Lựa chọn 6 🡪 Nhập xâu đầu vào: Bao Anh



Kết quả:



* Tính toán lượng bộ nhớ đã cấp phát

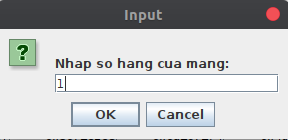
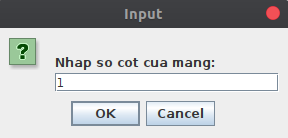
Lựa chọn 7 🡪 Kết quả: Do ta cấp phát 4 bytes cho 4 phần tử Char, 4 bytes cho 4 phần tử Byte và 4 bytes cho 1 phần tử Word nên tổng cộng bộ nhớ được cấp phát là 12 bytes.



* Cấp phát bộ nhớ cho mảng 2 chiều

Lựa chọn 8

Nhập số hàng là 1 Nhập số cột là 1

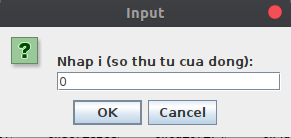
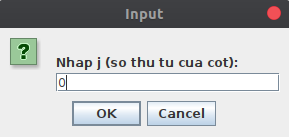
Kết quả:



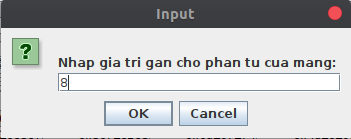
* Thiết lập giá trị các phần tử cho mảng 2 chiều

Lựa chọn 9: Do ở lựa chọn 8 ta chỉ cấp phát mảng 2 chiều kích thước 1x1 nên ta sẽ chỉ cần nhập vào giá trị cho phần tử Array[0][0].

Nhập số thứ tự của dòng (i) Nhập số thứ tự của cột (j)

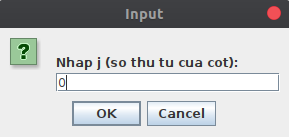
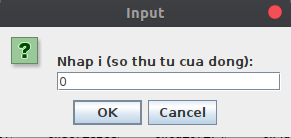
 

Nhập giá trị cho phần tử Array[0][0] bằng 8:



* Lấy giá trị của các phần tử mảng 2 chiều

Lựa chọn 10:



Kết quả:

