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

****

BÁO CÁO BÀI TẬP LỚN
HỌC PHẦN: THỰC HÀNH KIẾN TRÚC MÁY TÍNH

Nhóm: 01
Mã lớp học: 147789
Giảng viên: ThS. Lê Bá Vui

Danh sách sinh viên thực hiện:

STT	Họ và tên	Mã sinh viên	Đề tài số
01	Trịnh Hữu An	20225595	**06**
02	Phạm Đức Long	20225737	**03**

- Hà Nội, tháng 6 năm 2024 -

MỤC LỤC

Bài 6: Hàm cấp phát bộ nhớ malloc() 3
I, Đề bài 3
II, Phân tích cách làm, thuật toán 3
III, Mã nguồn 5
IV, Kết quả chạy mô phỏng 14
Bài 3: Kiểm tra tốc độ và độ chính xác khi gõ văn bản 18
I, Đề Bài 18
II, Phân tích cách làm, thuật toán 18
III, Mã nguồn 19
IV, Kết quả chạy mô phỏng 23

A, Trịnh Hữu An

Bài 6: Hàm cấp phát bộ nhớ malloc()

I, 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
    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.

II, Phân tích cách làm, thuật toán

Task 1: Cấp phát bộ nhớ động cho các biến trong chương trình

- Kiểm tra số phần tử và kích thước phần tử khi thêm vào, nếu số phần tử không lớn hơn không hoặc kích thước không phải 1 hoặc 4 thì báo lỗi. Nếu đúng, nhảy đến hàm malloc.

```
338 malloc:
339         la      $t9, Sys_TheTopOfFree
340         lw      $t8, 0($t9)             # Lay dia chi dau tien con trong
341 # 1:Khac phuc loi dam bao dia chi cua kieu word chia het cho 4
342
343         bne     $a2, 4, initialize      # Neu mang khoi tao co kieu Word, kiem tra dia chi dau co dam bao quy tac khong
344         andi    $t0, $t8, 0x03          # thuc hien andi voi so nhi phan 11(hay so thap phan 3)
345         beq     $t0, 0, initialize      # Neu khong co du, bo qua
346         addi    $t8, $t8, 4              #neu co du,lay dia chi dau cong voi 4
347         subu    $t8, $t8, $t0           #lay ket qua thu duoc tru di so du
348 initialize:
349 #TH KICH THUOC CAC PTHU #4-->chuyen sang phan tu ke tiep
350
351         sw      $t8, 0($a0)      # Cat dia chi do vao bien con tro
352         move    $v0, $t8 # Dong thoi la ket qua tra ve cua ham
353         mul     $t7, $a1,$a2     # Tinh kich thuoc cua mang can cap phat
354         add     $t6, $t8, $t7    # Tinh dia chi dau tien con trong
355         sw      $t6, 0($t9)      # Luu tro lai dia chi dau tien do vao bien Sys_TheTopOfFree
356         jr      $ra
357 #-----------------------------------------------
```

- Trong phần này, chúng ta kiểm tra xem địa chỉ đầu tiên của vùng nhớ cần cấp phát chia hết cho 4 không. Nếu không, chúng ta điều chỉnh nó để chia hết cho 4 bằng cách thêm hoặc trừ một số lượng nhỏ từ 1 đến 3. Điều này đảm bảo rằng địa chỉ bắt đầu của mảng kiểu word chắc chắn chia hết cho 4.

Task 2: Hiển thị giá trị của địa chỉ con trỏ, địa chỉ con trỏ trỏ đến

- Thực hiện lời gọi hệ thống với thanh ghi trả về $v0, load các biến, địa chỉ biến trỏ đến hay giá trị tại địa chỉ mà biến trỏ đến tại thanh ghi $a0 rồi thực hiện syscall

Task 3: Viết hàm thực hiện copy 2 con trỏ xâu ký tự

- Tương tự như trong ngôn ngữ lập trình C, ta sẽ lấy giá trị của địa chỉ xâu mà biến đang trỏ đến (ký tự đầu của xâu) sau đó, cộng 1 sau mỗi lần thực hiện vòng lặp ở cả xâu ký tự đã có và xâu ký tự thực hiện copy để copy xâu sang xâu mới.

Task 4: Viết hàm tính bộ lượng bộ nhớ đã cấp phát

- Thực hiện phép trừ giữa địa chỉ hiện tại của vùng nhớ trống (đỉnh của vùng nhớ tự do) và địa chỉ bắt đầu của vùng nhớ tự do. Kết quả của phép trừ này là tổng lượng bộ nhớ đã được cấp phát.

Task 5: Liên quan đến cấp phát mảng động 2 chiều

- Nhập số hàng và số cột, lưu ở thanh ghi a1 và a2
- Hàm malloc2, trước hết lưu lại giá trị của ra, số hàng, số cột để chạy được hàmcon 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.

Task 6: Get Array

```
227  #TH1:neu chon getArray
228  sub_case_1:
229          bne     $v0, 1, sub_case_2
230          li      $v0, 4
231          la      $a0, mess3.01
232          syscall
233          li      $v0, 5 #Nhap so hang i
234          syscall
235          move    $s0, $v0 #Luu so hang vai s0
236          li      $v0, 4
237          la      $a0, mess3.02
238          syscall
239          li      $v0, 5 #Nhap so cot j
240          syscall
241          move    $s1, $v0 #Luu vao s1
242
243          la      $t0, Array2Ptr
244          lw      $a0, 0($t0)
245          jal     getArray
246          move    $s2, $v0
247          li      $v0, 4
248          la      $a0, mess3.6
249          syscall
250          li      $v0, 1
251          move    $a0, $s2
252          syscall
253          j       sub_menu
```

- Nhập vào số hàng và số cột:

```
#------------------------8----------------------
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
#---------------------8----------------------
```

- 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.

Task 7: Set Array

```
227 sub_case_2:
228         bne     $v0, 2, sub_case_3
229         li      $v0, 4
230         la      $a0, mess3.01
231         syscall
232         li      $v0, 5
233         syscall
234         move    $s0, $v0
235         li      $v0, 4
236         la      $a0, mess3.02
237         syscall
238         li      $v0, 5
239         syscall
240         move    $s1, $v0
241         move    $s2, $v0
242         li      $v0, 4
243         la      $a0, mess0.3
244         syscall
245         li      $v0, 5
246         syscall
247         la      $t0, Array2Ptr
248         lw      $a0, 0($t0)
249         jal     setArray
250         j       sub_menu
```

```
382 setArray:
383         mul     $t0, $s0, $a2   # Vi tri cua phan tu = i * so cot + j
384         add     $t0, $t0, $s1
385         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
386         add     $t0, $t0, $a0   # Cong dia chi dau de ra dia chi phan tu
387         sw      $v0, 0($t0)     # Dat gia tri phan tu
388         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.

III, Mã nguồn

```
.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
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. Viet hàm thuc hien copy 2 con tro xâu kí tu?.
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. Giai phóng bo nho
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. Viet hàm tính toàn bo nho da cap 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		# Kiem tra xem dia chi dau tien co chia het cho 4 hay khong
	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		# Dieu chinh dia chi de chia het cho 4
initialize:
	sw	$t8, 0($a0)	# Ghi dia chi bat dau vao con tro tra ve
	move	$v0, $t8		# Tra ve dia chi bat dau cua vung nho cap phat
	mul	$t7, $a1,$a2	# Tinh kich thuoc cua mang can cap phat
	add	$t6, $t8, $t7	# Tinh dia chi duoi cung cua vung nho cap phat
	sw	$t6, 0($t9)	# Luu lai dia chi dau tien cua vung nho trong
	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
```

IV, Kết quả chạy mô phỏng

- Hàm lấy giá trị biến con trỏ và địa chỉ biến con trỏ:

```
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: 1
So phan tu: 3
So byte moi phan tu (1 hoac 4): 1
Nhap phan tu:
3
2
4
Gia tri cua con tro: 0x90000004
Dia chi cua con tro: 0x10010008
```

- Copy 2 con trỏ xâu kí tự:

```
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: 5

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

- Hàm cấp phát mảng 2 chiều:

```
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: 2

So cot: 3
Nhap phan tu:
1
2
3
4
5
6

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

- Set Array:

```
1. getArray[i][j]
2. setArray[i][j]
3. Thoat
Lua chon: 2
i = 2
j = 2
Nhap phan tu:
36
```

- Get Array

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

Gia tri cua phan tu: 36
```

- Tính lượng bộ nhớ đã cấp phát:

```
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: 1
So phan tu: 3
So byte moi phan tu (1 hoac 4): 4
Nhap phan tu:
4
3
2
Gia tri cua con tro: 0x90000004
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
6. Ket thuc chuong trinh
Lua chon: 5

Tong bo nho da cap phat: 12
```

- Giải phóng bộ nhớ:

```
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
```

B, Phạm Đức Long

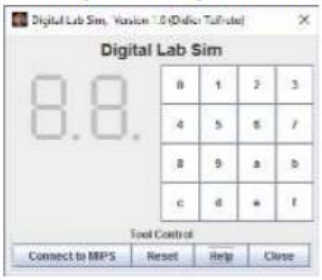
Bài 3: Kiểm tra tốc độ và độ chính xác khi gõ văn bản

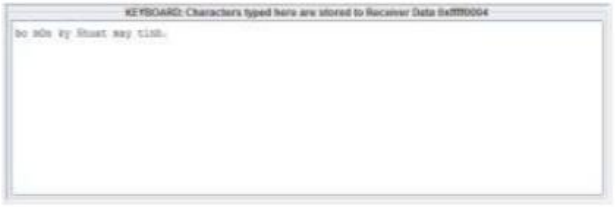
I, Đề Bài

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

Thực hiện chương trình đo tốc độ gõ bàn phím và hiển 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.
- Người dùng nhập các kí tự từ bàn phím. Ví dụ nhập *"bo m**O**n ky **5**huat may tinh"*. Chương trình cần phải đếm số kí tự đúng (trong ví dụ trên thì người dùng gõ sai chữ **O** và **5**) mà người dùng đã gõ và hiển thị lên các đèn led.
- Chương trình đồng thời cần tính được tốc độ gõ: thời gian hoàn thành và số từ trên một đơn vị thời gian.





II, Phân tích cách làm, thuật toán

- Đầu tiên ta khai báo các thư viện, các biến toàn cục sẽ được sử dụng trong bài làm
- Sau đó, tạo một vòng lặp vô hạn, kiểm tra giá trị $t1, tức là giá trị tại địa chỉ của KEY_READY. Nếu $t1 = 0 tức là người dùng chưa nhập từ bàn phím, ngược lại nếu = 1 thì người dùng đã nhập một ký tự nào đó từ bàn phím.
- Chương trình sẽ xảy ra ngắt trong hai trường hợp, là ngắt cứng thông qua bộ đếm Time Counter và ngắt mềm bằng lệnh teqi khi ta nhập vào một ký tự từ bàn phím.
- Khi xảy ra ngắt, con trỏ $pc sẽ nhảy đến vùng phục vụ ngắt .ktext. Trong .ktext, ta lấy ra giá trị bên trong thanh ghi Coproc0.cause($13) nhằm kiểm tra đây là loại ngắt nào (0x00000400 là ngắt do Counter, 0x00000034 là ngắt do nhận ký tự từ bàn phím)
- Xét trường hợp xảy ra ngắt do thực hiện nhập ký tự từ bàn phím:
    - Kiểm tra kí tự thứ **i** của string có trùng với kí tự thứ i mà ta nhập từ bàn phím hay không, nếu đúng thì cập nhật số ký tự đúng, nếu sai thì không cập nhật và tăng số kí tự đã đếm được
    - Kiểm tra nếu ký tự vừa nhập vào là ‘ ‘ mà kí tự trước đó khác ‘ ‘ thì tăng biến đếm số từ thêm 1
    - Sau đó tăng số ký tự nhập vào trong 1s lên 1, tạo một biến là thanh $s4 để lưu giá trị chữ cái này nhằm mục đích so sánh trong vòng lặp tiếp theo.
    - Cuối cùng tăng con trỏ string lên 1 để kiểm tra kí tự kế tiếp
- Xét trường hợp xảy ra ngắt cứng thực hiện bởi bộ đếm Counter
    - Chương trình kiểm tra xem số lần tạo lệnh ngắt của Timer đã đủ chưa (1s), nếu chưa đủ thì tăng biến đếm, còn nếu đã đủ thì hiển thị số kí tự đã gõ trong 1s lên DLS và khởi tạo lại biến đếm kí tự trong 1s, đồng thời tăng biến đếm thời gian hoàn thành nhập lên 1s.
    - Sau khi hoàn thành các câu lệnh trong vùng .ktext thì trở lại vòng lặp vô hạn ban đầu và thiết lập lại các thông số để đón nhận lần ngắt tiếp theo

- Ngoại lệ: Phím xóa (Backspace)
    - Chương trình đã tối ưu bằng cách khi xóa một ký tự thì chương trình vẫn sẽ trả về kết quả, nhưng gặp vấn đề ở chỗ kết quả được trả về lúc đúng lúc sai nếu chạy ở tốc độ tối đa, nhưng sẽ luôn đúng ở các tốc độ chậm hơn (30s/1 lệnh trở xuống kết quả nhận được vẫn đúng).
    - Điều này **có thể** được giải thích là do MIPS có cấu trúc tuần tự giả song song, do đó nếu ta để tốc độ chạy lệnh lên mức tối đa thì MIPS sẽ ưu tiên chạy những lệnh mà máy cho là ít thời gian xử lí hơn song song với các lệnh khác, nên đôi khi kết quả trả về nhận được sẽ là giá trị cũ (giá trị mà chưa được xử lí), dẫn đến kết quả nhận được hiển thị trên DLS có thể là sai.

III, Mã nguồn

```
.eqv SEVENSEG_LEFT 0xFFFF0011         # Dia chi cua den led 7 doan trai
.eqv SEVENSEG_RIGHT 0xFFFF0010        # Dia chi cua den led 7 doan phai
.eqv MASK_CAUSE_COUNTER 0x00000400       # Bit 10: Counter interrupt
.eqv COUNTER 0xFFFF0013                  # Time Counter
.eqv KEY_CODE   0xFFFF0004               # ASCII code from keyboard, 1 byte
.eqv KEY_READY  0xFFFF0000               # =1 if has a new keycode?
.data
mang_so:        .byte    63, 6,  91, 79, 102, 109 ,125, 7, 127, 111        # tu 0 den 9 (nhi phan)
string:  .asciiz "bach khoa ha noi"
message1:       .asciiz "Thoi gian hoan thanh: "
message2:       .asciiz "(s) \nToc do go trung binh: "
message3:       .asciiz " tu/phut\n"
Continue:       .asciiz "Tiep tuc nhap?"
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# MAIN Procsciiz ciiz edure
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
.text    # Cac bien toan cuc : k0, k1, s0, s1, s2, s3, s4,  , a1
MAIN:li         $k0,  KEY_CODE
         li         $k1,  KEY_READY
         # Enable the interrupt of TimeCounter of Digital Lab Sim
         li         $t1, COUNTER                  # Khoi tao bo dem timer
         sb         $t1, 0($t1)
         addi    $s0, $0, 0              # Dem so ky tu trong 1s
         addi    $s1, $0, 0              # Dem tong so ky tu dung
         addi    $s2, $0, 0              # Dem tong so tu nhap vao
         addi    $s3, $0, 0              # Dem so lan counter_intr
         addi    $s4, $0, 0              # Luu tru ky tu truoc do
         addi    $s5, $0, 0              # Dem thoi gian (giay)
         la         $a1, string
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# VONG LAP VO HAN DE DOI INTERRRUPT
loop:
         lw         $t1, 0($k1)              # $t1 = [$k1] = KEY_READY
         bne     $t1, $zero, make_Keyboard_Intr       # Tao interrupt khi nhan duoc ky tu tu
ban phim
         addi    $v0, $0, 32
         li         $a0, 5
         syscall
```

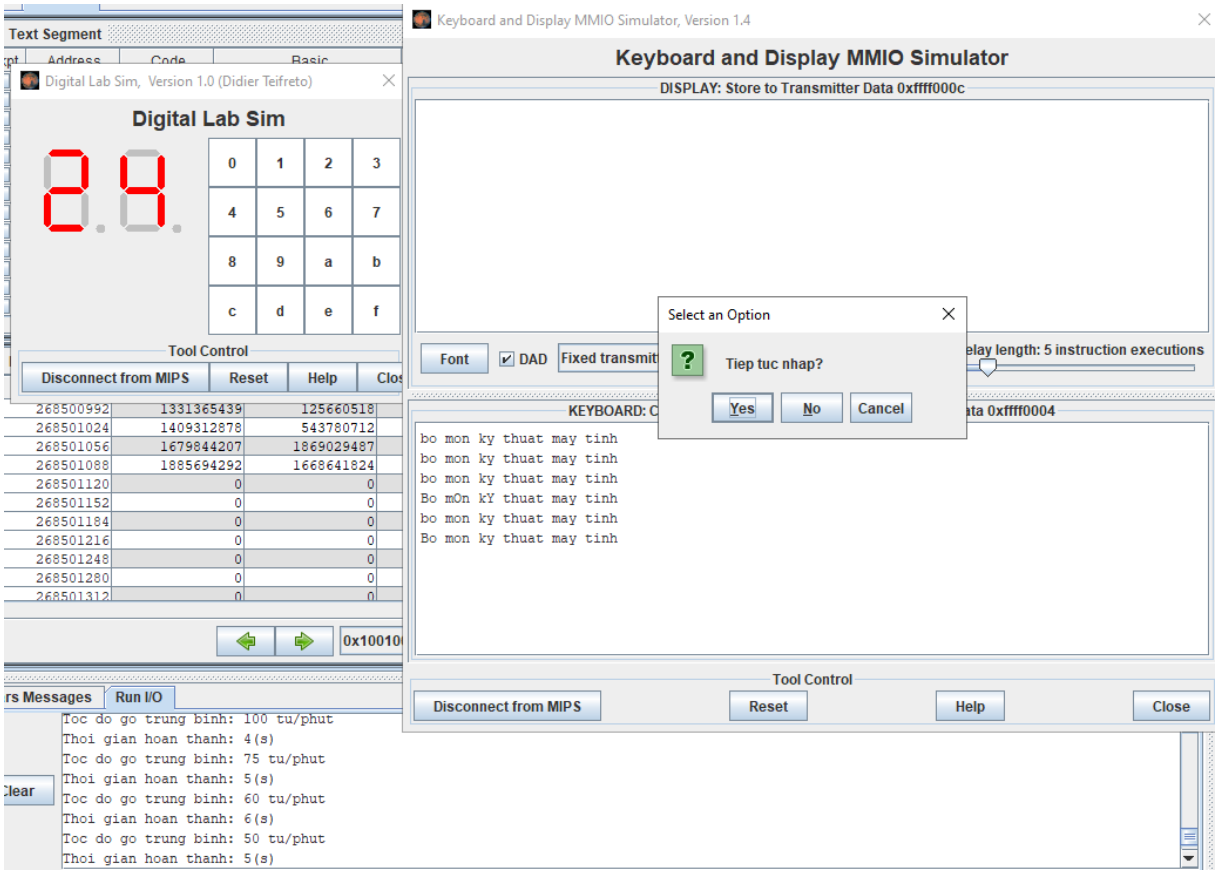
```
        b       loop                    # So lenh trong 1 vong lap = 6 => cu lap 5 lan thi
tao 1 counter interrupt
        nop
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
make_Keyboard_Intr:
        teqi    $t1, 1                  # $t1 = 1 => interrupt
        b       loop                    # Quay lai vong lap de cho doi su kien interrupt
tiep theo
        nop

end_Main:

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# PHAN PHUC VU NGAT
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
.ktext 0x80000180

dis_int:
        li      $t1, COUNTER            # BUG: must disable with Time Counter
        sb      $zero, 0($t1)
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# LAY GIA TRI CUA THANH GHI C0.cause DE KIEM TRA LOAI INTERRUPT
get_Caus:
        mfc0    $t1, $13                # $t1 = Coproc0.cause
isCount:
        li      $t2, MASK_CAUSE_COUNTER         # if Cause value confirm
Counter..
        and     $at, $t1, $t2
        bne     $at, $t2, keyboard_Intr
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# NGAT DO BO DEM COUNTER
counter_Intr:
        blt     $s3, 40, continue       # Neu so lap ngat do counter = 40 : da du 1s ->
khoi tao lai $s3, chieu toc do go ra DLS, tang bien dem thoi gian len 1
        jal     hien_thi
        addi    $s3, $0, 0              # Khoi tao lai $s3
        addi    $s5, $s5, 1             # Tang bien dem thoi gian(s)
        j       en_int
        nop
continue:
        addi    $s3, $s3, 1             # Neu chua du 1s thi tang bien dem so lan ngat
        j       en_int
        nop
keyboard_Intr:
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# NGAT DO BAN PHIM
kiem_tra_ky_tu:                         # Kiem tra ky tu nhap vao
        lb      $t0, 0($a1)             # Lay ki tu thu i trong mang da cho
        lb      $t1, 0($k0)             # Lay ki tu nhap vao tu ban phim
        beq     $t1, $0, en_int         # Loi
        beq     $t1, '\n', check_enter  # Ki tu la '\n', tien hanh kiem tra
```

```
        beq     $t1, 8, kiem_tra_backspace    # Neu ki tu nhap vao la backspace thi phai kiem
tra lai
        bne     $t0, $t1, kiem_tra_dau_cach  # Neu ki tu nhap vao va ki tu thu i trong mang da
cho giong nhau -> dem so ki tu dung
        nop
        addi    $s1, $s1, 1                   # Tang bien dem so ky tu dung
        b       kiem_tra_dau_cach
kiem_tra_backspace:
        beq     $s0, $s1, check1
        addi    $s1, $s1, 1
check1:
        bne     $s4, ' ', check2          # Neu ky tu truoc do la ' ' thi phai giam bien dem so tu
da nhap di 1
        addi    $s4, $zero, 0                 # Thay doi $s4 de tranh bi nham
        sub     $s2, $s2, 1
check2:
        sub     $a1, $a1, 1
        sub     $s0, $s0, 1                   # Giam so ky tu trong 1s
        sub     $s1, $s1, 1                   # Giam so ky tu dung
        b       loop
kiem_tra_dau_cach:                            # Kiem tra ki tu nhap vao co phai la ' ' hay ko
        bne     $t1, ' ', end_Process         # Neu ky tu nhap vao == ' ' && ky tu truoc do !=
' ' thi dem so tu da nhap
        nop
        beq     $s4, ' ', end_Process
        nop
        addi    $s2, $s2, 1                   # Tang bien dem so tu da nhap
end_Process:
        addi    $s0, $s0, 1                   # Tang so ky tu trong 1s len 1
        addi    $s4, $t1, 0                   # Cap nhat lai ky tu truoc do
        addi    $a1, $a1, 1                   # Tang con tro len 1 <=> string+i
        j       en_int
check_enter:
        beq     $s4, ' ', end_Program         # Neu ky tu truoc do khong phai la dau ' ' thi phai
cong so tu dem duoc them 1
        addi    $s2, $s2, 1
        b       end_Program
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# CHIEU RA MAN HINH DIGITAL LAB SIM GIA TRI CUA $s0
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
hien_thi:
        addi    $sp, $sp, -4
        sw      $ra, ($sp)
        addi    $t0, $0, 10
        div     $s0, $t0
        mflo    $v1                           # Lay so hang chuc
        mfhi    $v0                           # Lay so hang don vi
        la      $a0, mang_so
        add     $a0, $a0, $v1
        lb      $a0, 0($a0)                   # Set value for segments
```

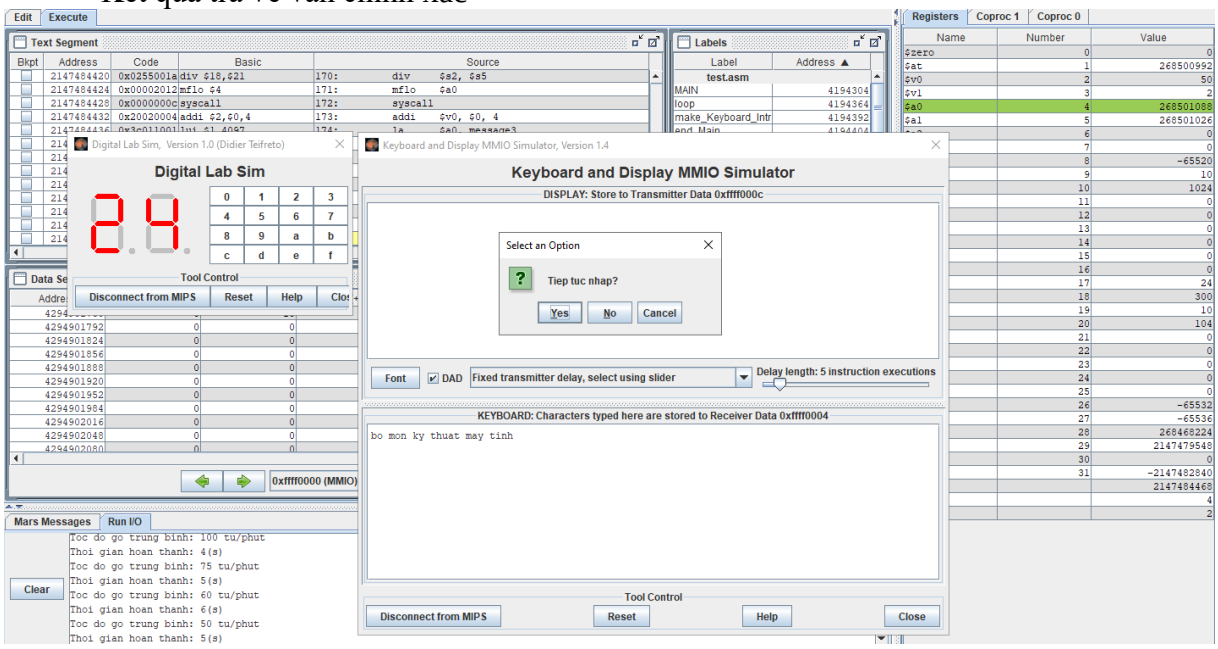
```
        jal     SHOW_7SEG_LEFT                  # Hien thi

        la      $a0, mang_so
        add     $a0, $a0, $v0
        lb      $a0, 0($a0)             # Set value for segments
        jal     SHOW_7SEG_RIGHT                 # Hien thi

        addi    $s0, $0, 0              # Sau khi chieu ra man hinh thi khoi tao lai bien
dem
        lw      $ra, ($sp)
        addi    $sp, $sp, 4
        jr      $ra
SHOW_7SEG_LEFT:
        li      $t0, SEVENSEG_LEFT              # Assign port's address
        sb      $a0, 0($t0)             # Assign new value
        jr      $ra
SHOW_7SEG_RIGHT:
        li      $t0, SEVENSEG_RIGHT             # Assign port's address
        sb      $a0, 0($t0)             # Assign new value
        jr      $ra
        nop
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# KET THUC CHUONG TRINH VA HIEN THI SO KY TU DUNG
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
end_Program:
        # Hien thi thoi gian hoan thanh (giay) va toc do go trung binh (tu/phut)
        addi    $v0, $0, 4
        la      $a0, message1
        syscall

        # Thoi gian hoan thanh
        addi    $v0, $0, 1
        addi    $a0, $s5, 0
        syscall

        # Toc do go trung binh = 60 * (so tu da nhap) / (thoi gian hoan thanh)
        addi    $v0, $0, 4
        la      $a0, message2
        syscall
        addi    $v0, $0, 1
        addi    $a0, $0, 60
        mult    $s2, $a0
        mflo    $s2
        div     $s2, $s5
        mflo    $a0
        syscall
        addi    $v0, $0, 4
        la      $a0, message3
        syscall
        addi    $s0, $s1, 0
        jal     hien_thi                # hien thi so tu dung ra led 7 doan
CONTINUE:
```

```
        # tiep tuc?
        li $v0, 50
        la $a0, Continue
        syscall
        beq $a0, 0, MAIN
        li $v0, 10
        syscall

# ---------------------- Ket thuc xu ly ngat ------------------------------------
en_int:
        li      $t1, COUNTER
        sb      $t1, 0($t1)
        mtc0    $zero, $13                  # Must clear cause reg
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
return:
        eret                                # Return from exception


#-----------------------------length check-----------------------------------------
#length:
        # kiem tra do dai String
#       add     $t9, $t9, 1             # $t9 = length
#       lb      $s0, 0($a1)
#       addi    $a1, $a1, 1
#       bne     $s0, 0, length          # kiem tra neu ky tu do la '\0' thi ket thuc kiem tra
```

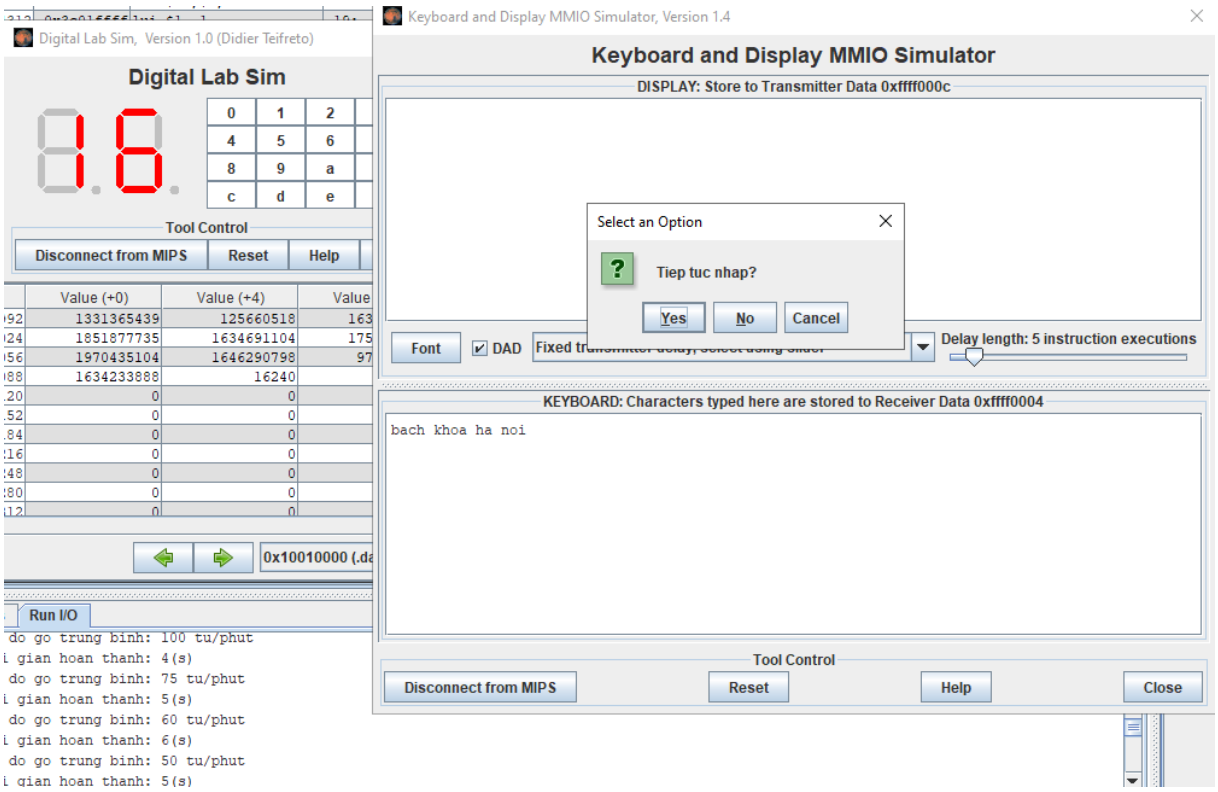
IV, Kết quả chạy mô phỏng



- Trường hợp nhập:
  - abcdef_ (backspace)
  - (đã xóa hết)
  - bo mon ky thuat may tinh
- Kết quả trả về vẫn chính xác



- Thử thay đổi chuỗi đầu vào string = ‘bach khoa ha noi’


Digital Lab Sim, Version 1.0 (Didier Teifreto)
Digital Lab Sim
0
1
2
4
5
6
8
9
a
c
d
e
Tool Control
Disconnect from MIPS
Reset
Help
Value (+0)
Value (+4)
Value
1331365439
125660518
1851877735
1634691104
1970435104
1646290798
1634233888
16240
0x10010000 (.da
Run I/O
do go trung binh: 100 tu/phut
i gian hoan thanh: 4(s)
do go trung binh: 75 tu/phut
i gian hoan thanh: 5(s)
do go trung binh: 60 tu/phut
i gian hoan thanh: 6(s)
do go trung binh: 50 tu/phut
i gian hoan thanh: 5(s)
Keyboard and Display MMIO Simulator, Version 1.4
Keyboard and Display MMIO Simulator
DISPLAY: Store to Transmitter Data 0xffff000c
Select an Option
Tiep tuc nhap?
Yes
No
Cancel
Font
DAD
Delay length: 5 instruction executions
KEYBOARD: Characters typed here are stored to Receiver Data 0xffff0004
bach khoa ha noi
Tool Control
Disconnect from MIPS
Reset
Help
Close