Thực Hành Kiến Trúc Máy Tính - Lớp: 139365

Bài thực hành số 11

Họ và tên: Lê Thị Nhung MSSV: 20210662

Bài 1

```
.eqv IN_ADRESS_HEXA_KEYBOARD 0xFFFF0012
# receive row and column of the key pressed, 0 if not key pressed
# Eg. equal 0x11, means that key button 0 pressed.
# Eg. equal 0x28, means that key button D pressed.
.eqv OUT_ADRESS_HEXA_KEYBOARD 0xFFFF0014
.text
main:
		li $t1, IN_ADRESS_HEXA_KEYBOARD
		li $t2, OUT_ADRESS_HEXA_KEYBOARD
		li $t4,0x10
 set:
		li $t3, 0x01 # check row 4 with key C, D, E, F
polling:
		beq $t3,$t4,print
		 sb $t3, 0($t1 ) # must reassign expected row
		lb $a0, 0($t2) # read scan code of key button
		bne $a0,$0,print
		sll $t3,$t3,1
		j polling

 print:
		li $v0, 34 # print integer (hexa)
		syscall
 sleep:
		li $a0, 100 # sleep 100ms
		li $a0, 100 # sleep 100ms
		li $v0, 32
		syscall
		j set
back_to_polling:
		j polling # continue polling
```

Giải thích code:

Mục đích: in ra màn hình mã scan của số được bấm ở trên Digital Lab Sim qua ma trận nút key matrix:

	0x1	0x2	0x4	0x8
0x1	0	1	2	3
0x2	4	5	6	7
0x4	8	9	A	B
0x8	C	D	E	F

Các cột sẽ là chữ số hàng chục, các hàng sẽ là chữ số hàng đơn vị, chẳng hạn nếu ta bấm 0 màn hình sẽ hiện 0x00000011, còn bấm 6 sẽ hiện 0x00000048 (ngoại trừ F sẽ hiện ra 0xFFFFFF88).

-) Gán chỉ hàng để kiểm tra vào byte tại địa chỉ IN_ADRESS_HEXA_KEYBOARD (0xFFFF0012)

-) Đọc byte tại địa chỉ OUT_ADRESS_HEXA_KEYBOARD (0xFFFF0014), để phát hiện nút phím nào đã được nhấn.

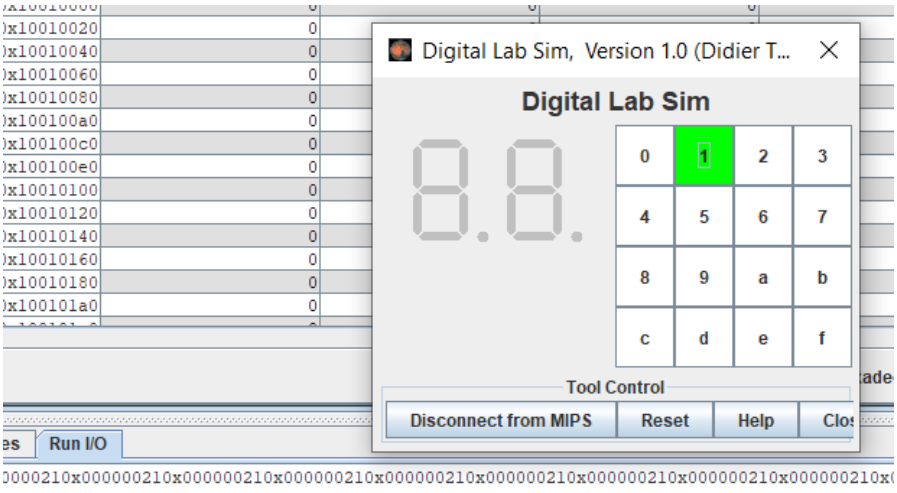
Đầu tiên ta sẽ có $t3 = 0x01 là giá trị hàng đầu tiên.

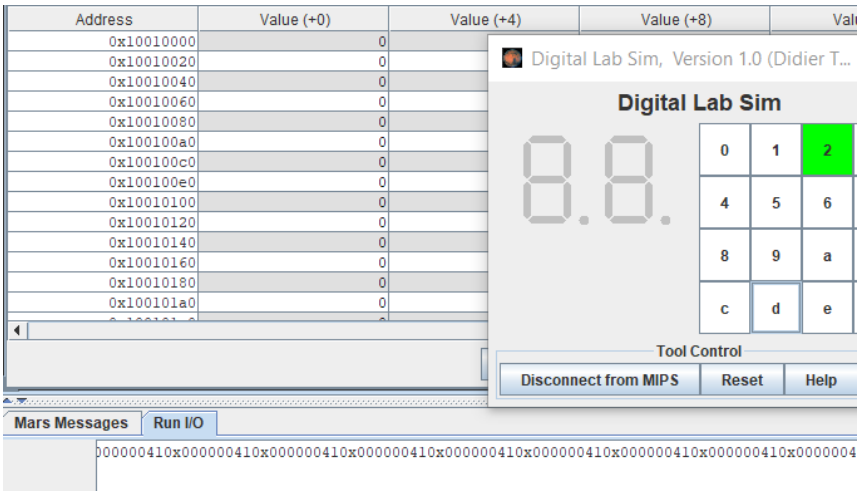
Polling: nếu

Để có thể đọc hết cả 4 hàng, để kiểm tra điều kiện: Sau khi lưu hàng $t0 vào trong IN_ADDRESS_HEXA_KEYBOARD, ta kiểm tra xem số ở trong địa chỉ OUT_ADDRESS_HEXA_KEYBOARD (được gán vào $a0) có bằng 0 không, nếu có tức là số ta vừa bấm không ở hàng đó, khi đó ta sẽ nhân đôi giá trị ở $t0 lên và lặp lại vòng lặp check. Nếu $t0 = 16 thì tức là ta chưa bấm gì cả và ta sẽ dừng vòng lặp rồi nhảy đến print, còn nếu $a0 khác 0 ở bất kỳ hàng nào ta cũng dừng vòng lặp lại và nhảy đến print

-) Hàm print sẽ giúp ta in ra mã scan của số ta vừa nhập ra màn hình. Hàm sleep sẽ làm cho chương trình ngủ 100ms.

Kết quả chạy được :


Digital Lab Sim, Version 1.0 (Didier T...
Digital Lab Sim
0 1 2 3
4 5 6 7
8 9 a b
c d e f
Tool Control
Disconnect from MIPS
Reset
Help
Run I/O


Address
Value (+0)
Value (+4)
Value (+8)
Digital Lab Sim, Version 1.0 (Didier T...
Digital Lab Sim
0 1 2
4 5 6
8 9 a
c d e
Tool Control
Disconnect from MIPS
Reset
Help
Mars Messages
Run I/O

Bài 2 :

```
.eqv IN_ADRESS_HEXA_KEYBOARD 0xFFFF0012
.data
Message: .asciiz "Oh my god. Someone's presed a button.\n"
.text
main:
                li $t1, IN_ADRESS_HEXA_KEYBOARD
               li $t3, 0x80 # bit 7 of = 1 to enable interrupt
               sb $t3, 0($t1)
Loop:          nop
               nop
               nop
               nop
               b Loop # Wait for interrupt
end_main:
.ktext 0x80000180
IntSR:
               addi $v0, $zero, 4 # show message
               la $a0, Message
               syscall
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
```

Giải thích code:

Mục đích : Chương trình trên minh họa cho cách chương trình ngắt khi ta bấm một phím trên Ma trận phím.

-) Ta khởi tạo giá trị IN_ADDRESS = 0xFFFF0012 và Message để thông báo mỗi khi ta bấm một phím trên ma trận phím và chương trình thực hiện ngắt.

-) Thanh ghi $t1= IN_ADDRESS và $t3 lưu địa chỉ 0x80(khi bit thứ 7 bằng 1 thì chức năng interrupt được kích hoạt) , sau đó gán giá trị $t3 vào địa chỉ ở thanh ghi $t1.

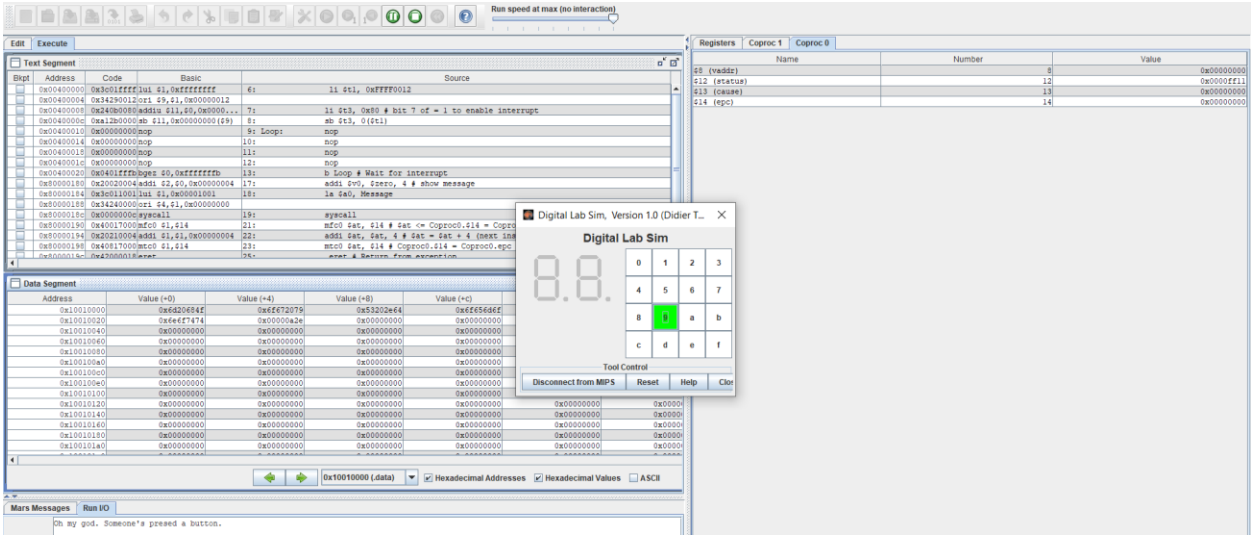
-)Loop: b Loop để quay trở lại vòng lặp. Vòng lặp cứ tiếp tục cho tới khi có tín hiệu ngắt xảy ra, khi đó chương trình nhảy đến chương trình con phục vụ ngắt ở k.text.

-) k.text gán bằng địa chỉ cố định 0x80000180

-) Chương trình con phục vụ ngắt : in ra màn hình message khi có tín hiệu ngắt đc ấn từ bàn phím dig. Sau đó gán $at bằng giá trị ở thanh ghi $14 của bộ đồng xử lý C0- đây sẽ là địa chỉ của lệnh mà ngắt xảy ra, ta cần tăng địa chỉa để trỏ tới địa chri tiếp theo (+4 vào $at rồi gán lại vào $14).

Sau khi kết thúc chương trình con, sử dụng lệnh eret để quay trở lại chương trình chính.

Kết quả chạy được :



Bài 3:

```
.eqv IN_ADRESS_HEXA_KEYBOARD 0xFFFF0012
.eqv OUT_ADRESS_HEXA_KEYBOARD 0xFFFF0014
.data
Message: .asciiz "Key scan code "
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# MAIN Procedure
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
.text
main:
 #-------------------------------------------------------------
 # Enable interrupts you expect
 #-------------------------------------------------------------
 # Enable the interrupt of Keyboard matrix 4x4 of Digital Lab Sim
 li $t1, IN_ADRESS_HEXA_KEYBOARD
 li $t3, 0x80 # bit 7 = 1 to enable
 sb $t3, 0($t1)
#-------------------------------------------------------------
 # Loop an print sequence numbers
 #-------------------------------------------------------------
 xor $s0, $s0, $s0 # count = $s0 = 0
Loop: addi $s0, $s0, 1 # count = count + 1
prn_seq:addi $v0,$zero,1
 add $a0,$s0,$zero # print auto sequence number
 syscall
prn_eol:addi $v0,$zero,11
```

```
 li $a0,'\n' # print endofline
 syscall
sleep: addi $v0,$zero,32
 li $a0,300 # sleep 300 ms
 syscall
 nop # WARNING: nop is mandatory here.
 b Loop # Loop
end_main:
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# GENERAL INTERRUPT SERVED ROUTINE for all interrupts
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
.ktext 0x80000180
 #---------------------------------------------------------
 # SAVE the current REG FILE to stack
 #---------------------------------------------------------
IntSR: addi $sp,$sp,4 # Save $ra because we may change it later
 sw $ra,0($sp)
 addi $sp,$sp,4 # Save $ra because we may change it later
 sw $at,0($sp)
 addi $sp,$sp,4 # Save $ra because we may change it later
 sw $v0,0($sp)
 addi $sp,$sp,4 # Save $a0, because we may change it later
 sw $a0,0($sp)
 addi $sp,$sp,4 # Save $t1, because we may change it later
```

```
 sw $t1,0($sp)
 addi $sp,$sp,4 # Save $t3, because we may change it later
 sw $t3,0($sp)
 #---------------------------------------------------------
 # Processing
 #---------------------------------------------------------
prn_msg:addi $v0, $zero, 4
 la $a0, Message
 syscall
 li $t3,0x01
get_cod:li $t1, IN_ADRESS_HEXA_KEYBOARD
 ori $t4, $t3, 0x80
 sb $t4, 0($t1) # must reassign expected row
 li $t1, OUT_ADRESS_HEXA_KEYBOARD
 lb $a0, 0($t1)
 bne $a0,$0,prn_cod
 sll $t3,$t3,1
 j get_cod
prn_cod:li $v0,34
 syscall
 li $v0,11
 li $a0,'\n' # print endofline
 syscall
 #---------------------------------------------------------
```

```
73   #-----------------------------------------------------
74   # Evaluate the return address of main routine
75   # epc <= epc + 4
76   #-----------------------------------------------------
77  next_pc:mfc0 $at, $14 # $at <= Coproc0.$14 = Coproc0.epc
78   addi $at, $at, 4 # $at = $at + 4 (next instruction)
79   mtc0 $at, $14 # Coproc0.$14 = Coproc0.epc <= $at
80   #-----------------------------------------------------
81   # RESTORE the REG FILE from STACK
82   #-----------------------------------------------------
83  restore:lw $t3, 0($sp) # Restore the registers from stack
84   addi $sp,$sp,-4
85   lw $t1, 0($sp) # Restore the registers from stack
86   addi $sp,$sp,-4
87   lw $a0, 0($sp) # Restore the registers from stack
88   addi $sp,$sp,-4
89   lw $v0, 0($sp) # Restore the registers from stack
90   addi $sp,$sp,-4
91   lw $ra, 0($sp) # Restore the registers from stack
92   addi $sp,$sp,-4
93  return: eret # Return from exception
```

Giải thích code:

Mục đích: Chương trình trên sẽ liên tục đếm ra màn hình và in ra màn hình các số mà ta bấm trên Ma trận phím sử dụng ngắt.

Chương trình có 2 phần chính là .text để chạy bình thường còn .ktext sẽ là phần xử lý khi gặp ngắt (ngắt ở đây là khi ta bấm 1 số trên Ma trận phím)

-) Ban đầu, ta khởi tạo cho IN_ADDRESS là 0xFFFF0012 và OUT_ADDRESS là 0xFFFF0014 và gán IN_ADDRESS vào thanh ghi $t1.

-) Thanh ghi $t1= IN_ADDRESS và $t3 lưu địa chỉ 0x80(khi bit thứ 7 bằng 1 thì chức năng interrupt được kích hoạt) , sau đó gán giá trị $t3 vào địa chỉ ở thanh ghi $t1. Xor : thiết lập giá trị của $s0(biến đếm) thành 0.

-) Loop: Tăng giá trị của $s0 lên 1->tạo ra một chuỗi số tự động.

-) prn_seq: in số thứ tự tự động.

-) k.text gán bằng địa chỉ cố định 0x80000180

-) IntSR:Tăng giá trị của con trỏ ngăn xếp lên 4 để lưu thanh ghi $ra, $a0,$t1,$t3 vào địa chỉ ngăn xếp->lưu trữ các thanh ghi hiện tại vào ngăn xếp

-) get_cod:

+)Lưu địa chỉ của thanh ghi đầu vào vào thanh ghi $t1 rồi OR logic giữa giá trị trong $t3 và 0x80, kết quả được lưu vào $t4. Sau đó lưu giá trị trong $t4 vào địa chỉ ghi ở thanh ghi $t1.

+)Lưu địa chỉ của thanh ghi đầu ra vào thanh ghi $t1 rôi đọc giá trị từ địa chỉ được chỉ định bởi $t1 và lưu vào $a0. So sánh giá trị trong $a0 với 0,nếu khác nhau, nhảy tới nhãn prn_cod để hiển thị mã quét của nút bàn phím bấm vào.

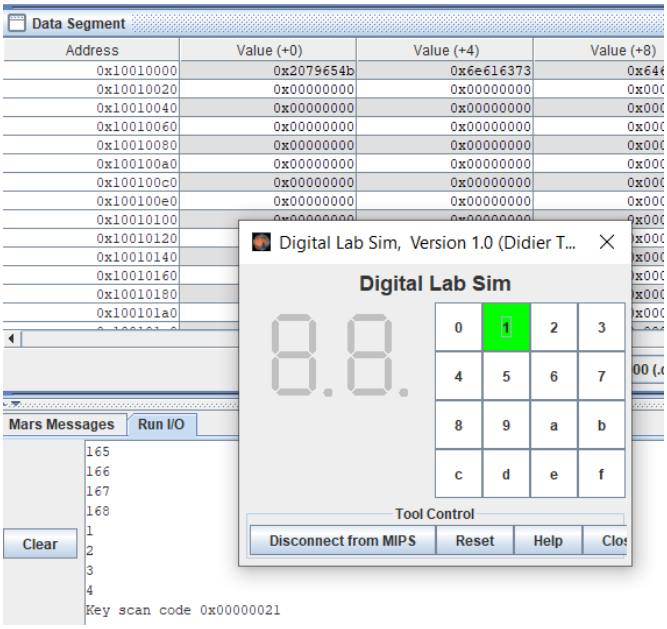
+)sll: Dịch trái giá trị trong $t3 một vị trí ->chuẩn bị cho lần lặp tiếp theo.

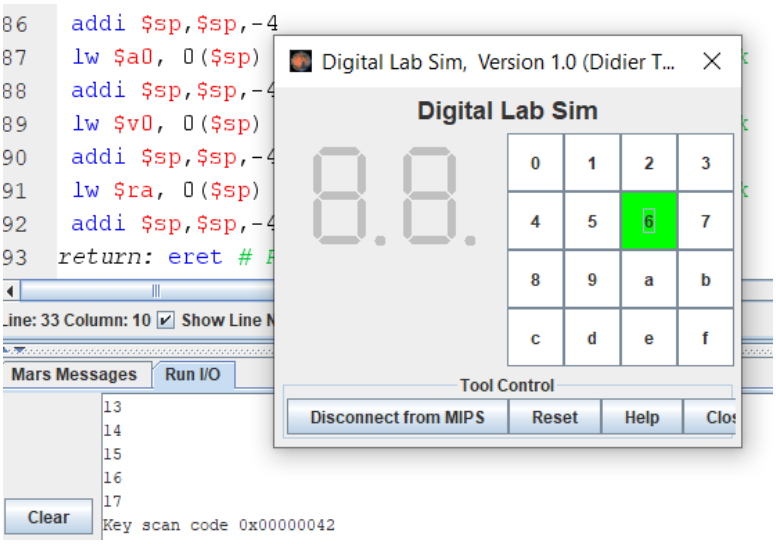
-)next_pc: mfc0 $at, $14: Sao chép giá trị của thanh ghi Coprocessor 0, $14 (epc), vào thanh ghi $at; addi $at, $at, 4: thực hiện lệnh gán epc = epc + 4 ->chuyển tới lệnh tiếp theo; mtc0 $at, $14: Sao chép giá trị từ thanh ghi $at vào thanh ghi Coprocessor 0, $14 (epc).

-)restore: có công dụng khôi phục lại các thanh ghi từ ngắn xếp.

Sau khi kết thúc chương trình con, sử dụng lệnh eret để quay trở lại chương trình chính.

Kết quả chạy được :





Bài 4:

```
.eqv IN_ADRESS_HEXA_KEYBOARD 0xFFFF0012
.eqv COUNTER 0xFFFF0013 # Time Counter
.eqv MASK_CAUSE_COUNTER 0x00000400 # Bit 10: Counter interrupt
.eqv MASK_CAUSE_KEYMATRIX 0x00000800 # Bit 11: Key matrix
.data
msg_keypress: .asciiz "Someone has pressed a key!\n"
msg_counter: .asciiz "Time inteval!\n"
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# MAIN Procedure
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
.text
main:
 #-------------------------------------------------------
 # Enable interrupts you expect
 #-------------------------------------------------------
 # Enable the interrupt of Keyboard matrix 4x4 of Digital Lab
 li $t1, IN_ADRESS_HEXA_KEYBOARD
 li $t3, 0x80 # bit 7 = 1 to enable
 sb $t3, 0($t1)
# Enable the interrupt of TimeCounter of Digital Lab Sim
 li $t1, COUNTER
 sb $t1, 0($t1)

 #-------------------------------------------------------
 # Loop an print sequence numbers
```

```
Loop: nop
 nop
 nop
sleep: addi $v0,$zero,32 # BUG: must sleep to wait for Time
 li $a0,200 # sleep 300 ms
 syscall
 nop # WARNING: nop is mandatory here.
 b Loop
end_main:
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# GENERAL INTERRUPT SERVED ROUTINE for all interrupts
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
.ktext 0x80000180
IntSR: #-----------------------------------------------------------
 # Temporary disable interrupt
 #-----------------------------------------------------------
dis_int:li $t1, COUNTER # BUG: must disable with Time Counter
 sb $zero, 0($t1)
 # no need to disable keyboard matrix interrupt
 #-----------------------------------------------------------
 # Processing
 #-----------------------------------------------------------
get_caus:mfc0 $t1, $13 # $t1 = Coproc0.cause
```

```
IsCount:li $t2, MASK_CAUSE_COUNTER# if Cause value confirm
 and $at, $t1,$t2
 beq $at,$t2, Counter_Intr
IsKeyMa:li $t2, MASK_CAUSE_KEYMATRIX # if Cause value confirm Key..
 and $at, $t1,$t2
 beq $at,$t2, Keymatrix_Intr
others: j end_process # other cases
Keymatrix_Intr: li $v0, 4 # Processing Key Matrix Interrupt
 la $a0, msg_keypress
 syscall
 j end_process
Counter_Intr: li $v0, 4 # Processing Counter Interrupt
 la $a0, msg_counter
 syscall
 j end_process
end_process:
 mtc0 $zero, $13 # Must clear cause reg
en_int: #-----------------------------------------------------------
 # Re-enable interrupt
 #-----------------------------------------------------------
 li $t1, COUNTER
 sb $t1, 0($t1)
 #-----------------------------------------------------------
 # Evaluate the return address of main routine
```

```
74   # epc <= epc + 4
75   #---------------------------------------------------------
76  next_pc:mfc0 $at, $14 # $at <= Coproc0.$14 = Coproc0.epc
77   addi $at, $at, 4 # $at = $at + 4 (next instruction)
78   mtc0 $at, $14 # Coproc0.$14 = Coproc0.epc <= $at
79   return: eret # Return from exception
```

Giải thích code:

-) Chương trình cho phép ngắt đồng thời bằng 2 cách: từ bàn phím Dig Lab Sim và bộ đếm thời gian của Lab Sim. Nếu là ngắt do quá tgian thì in ra dòng "time interval" , ngắt do ma trận phím thì nó in ra "someone pressed a button"

-) Tại Coproc0, thanh ghi 13 lưu giá trị để phân biệt kiểu ngắt: 0x400 -> ngắt timer, 0x800 -> ngắt từ bàn phím

-) Đầu tiên, khởi tạo 2 giá trị địa chỉ cho IN_ADDRESS = 0xFFFF0012 và COUNTER = 0xFFFF0013, lần lượt là địa chỉ cho bộ ma trận phím và bộ đếm thời gian. msg_keypress: thông báo ngắt bởi ma trận phím và msg_counter: thông báo ngắt bởi bộ đếm thời gian.

-) Gán $t1 = IN_ADDRESS, $t3 = 0x80 (bit 7 là 1) để cho phép chương trình có thể ngắt bằng ma trận phím và lưu giá trị $t3 vào trong địa chỉ ghi thanh ghi $t1, gán lại $t1 = COUNTER và gán giá trị ở $t1 vào chính địa chỉ được lưu ở $t1 đó để cho phép ngắt bằng bộ đếm thời gian.

-) Loop: b Loop quay lại vòng lặp để đợi cho ngắt xảy ra. Khi hết thời gian đếm hoặc người dùng bấm một phím nào đó trong ma trận phím thì chương trình thực hiện ngắt và ta nhảy đến chương trình con phục vụ ngắt ở .ktext ở địa chỉ 0x80000180.

-) Ở chương trình con, trước tiên ta sẽ tạm tắt không cho phép ngắt bằng bộ đếm thời gian để tránh xảy ra lỗi bằng cách gán 0 vào trong địa chỉ ở $t1 vừa rồi.

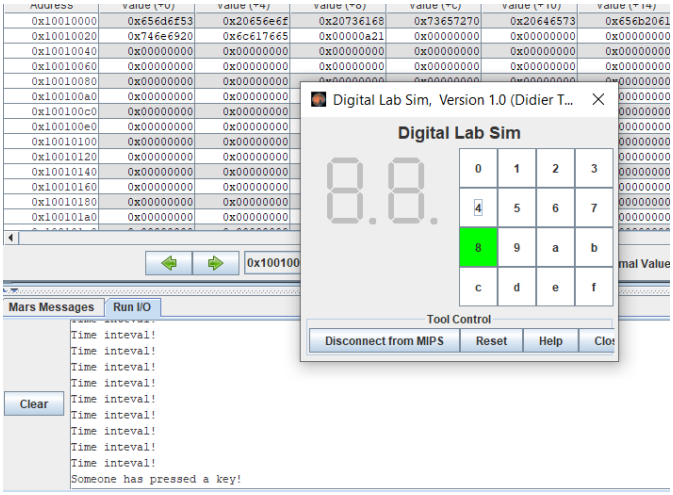
-)Sau đó ta trích xuất giá trị ở thanh ghi $13 của bộ đồng xử lý C0 để kiểm tra nguyên nhân ngắt, để làm điều đó ta gán cho $t2 = MASK_CAUSE_COUNTER, sau đó xem $t1 có bằng $t2 không, nếu có thì đúng là ngắt do bộ đếm thời gian, khi đó ta sẽ nhảy để Counter_Intr để in ra màn hình thông báo ngắt do hết thời gian bộ đếm. Nếu không ta gán $t2 = MASK_CAUSE_KEYMATRIX, sau đó ta lại so sánh $t1 với $t2, nếu bằng tức là ngắt do ma trận phím, khi đó ta sẽ nhảy đến Keymatrix_Intr để in ra màn hình thông báo ngắt do ma trận phím. Nếu không phải

cả 2 thì tức là ngắt do một yếu tố khác, khi đó ta sẽ ở others để nhảy đến end_process và thoát ra khỏi hàm con.

-)Sau khi thực hiện in thông báo ta sẽ xóa dữ liệu thanh ghi $13 đi bằng cách lưu giá trị 0 vào đó. Sau đó ta lưu lại $t1 vào địa chỉ ở $t1 để bật lại cho phép ngắt bằng thời gian.

-) Sau đó ta sẽ quay trở lại chương trình chính bằng cách trích xuất giá trị ở thanh $14 của bộ đồng xử lý C0, cộng thêm 4 rồi lưu lại vào, sau đó dùng lệnh eret để quay lại chương trình chính.

Kết quả chạy :



Bài 5:

```
.eqv KEY_CODE 0xFFFF0004 # ASCII code from keyboard, 1 byte
.eqv KEY_READY 0xFFFF0000 # =1 if has a new keycode ?
 # Auto clear after lw
.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
.eqv MASK_CAUSE_KEYBOARD 0x0000034 # Keyboard Cause
.text
 li $k0, KEY_CODE
 li $k1, KEY_READY
 li $s0, DISPLAY_CODE
 li $s1, DISPLAY_READY
loop: nop
WaitForKey: lw $t1, 0($k1) # $t1 = [$k1] = KEY_READY
 beq $t1, $zero, WaitForKey # if $t1 == 0 then Polling
MakeIntR: teqi $t1, 1 # if $t0 = 1 then raise an Interrupt
 j loop
#---------------------------------------------------------------
# Interrupt subroutine
#---------------------------------------------------------------
.ktext 0x80000180
get_caus: mfc0 $t1, $13 # $t1 = Coproc0.cause
IsCount: li $t2, MASK_CAUSE_KEYBOARD# if Cause value confirm
 and $at, $t1,$t2
 beq $at,$t2, Counter_Keyboard
```

```
 beq $at,$t2, Counter_Keyboard
 j end_process
Counter_Keyboard:
ReadKey: lw $t0, 0($k0) # $t0 = [$k0] = KEY_CODE
WaitForDis: lw $t2, 0($s1) # $t2 = [$s1] = DISPLAY_READY
 beq $t2, $zero, WaitForDis # if $t2 == 0 then Polling
Encrypt: addi $t0, $t0, 1 # change input key
ShowKey: sw $t0, 0($s0) # show key
 nop
end_process:
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

```

Giải thích code:

Mục đích: Chương trình trên minh họa cách chương trình ngắt với bàn phím. Chương trình trên sẽ nhận ký tự ta nhập vào, sau đó in ra màn hình kí tự có mã ASCII lớn hơn 1.

-) Khởi tạo các giá trị địa chỉ cho:

+) KEY_CODE 0xFFFF0004: Địa chỉ thanh ghi lưu trữ mã ASCII của phím từ bàn phím, kích thước 1 byte.

+)KEY_READY 0xFFFF0000: Địa chỉ thanh ghi cho biết có mã ASCII mới từ bàn phím hay không, bằng 1 nếu có, kích thước 1 byte.

+)DISPLAY_CODE 0xFFFF000C: Địa chỉ thanh ghi lưu trữ mã ASCII để hiển thị trên màn hình, kích thước 1 byte.

+) DISPLAY_READY 0xFFFF0008: Địa chỉ thanh ghi cho biết liệu màn hình đã sẵn sàng để hiển thị hay chưa, có giá trị bằng 1 nếu đã sẵn sàng, kích thước 1 byte.

+)MASK_CAUSE_KEYBOARD 0x0000034 : để xác định nguyên nhân ngắt từ bàn phím.

-)Đầu tiên lần lượt gán các giá trị của địa chỉ thanh ghi lưu trữ trên lần lượt vào các thanh ghi $k0, $k1, $s0, $s1.

-)WaitForKey: đọc giá trị từ địa chỉ được lưu trong thanh ghi $k1 vào $t1. Nếu giá trị trong thanh ghi $t1 bằng 0, tức là không có mã ASCII mới, thì quay lại WaitForKey để tiếp tục kiểm tra

-)MakeIntR : so sánh giá trị trong thanh ghi $t1 với 1. Nếu bằng nhau, tức là có mã ASCII mới từ bàn phím, thì gây ra một ngắt. Nếu không thì quay lại loop để tiếp tục kiểm tra.

-)get_caus: đọc giá trị từ thanh ghi Coproc0.cause vào thanh ghi $t1 để xác định nguyên nhân của ngắt.

-)IsCount: kiểm tra xem nguyên nhân của ngắt có phải là từ bàn phím hay không bằng cách so sánh giá trị trong thanh ghi $t1 với MASK_CAUSE_KEYBOARD.

-)ReadKey: đọc giá trị từ địa chỉ được lưu trong thanh ghi $k0 vào thanh ghi $t0 để lưu trữ mã ASCII của phím từ bàn phím .

-)WaitForDis: Đọc giá trị từ địa chỉ được lưu trong thanh ghi $s1 vào thanh ghi $t2 để kiểm tra xem màn hình đã sẵn sàng để hiển thị hay chưa.Nếu giá trị trong thanh ghi $t2 bằng 0, tức là màn hình chưa sẵn sàng, thì quay lại WaitForDis để tiếp tục kiểm tra.

-)Encrypt : Tăng mã ascii của KEY_CODE thêm 1 đơn vị.

-)ShowKey: Ghi giá trị trong thanh ghi $t0 vào địa chỉ thanh ghi $s0 để hiển thị mã ASCII của phím trên màn hình.

Sau đó ta sẽ quay trở lại chương trình chính bằng cách trích xuất giá trị ở thanh $14 của bộ đồng xử lý C0, cộng thêm 4 rồi lưu lại vào, sau đó dùng lệnh eret để quay lại chương trình chính.

Kết quả chạy được :