**Thực hành kiến trúc máy tính**

**Assignment 1:**

1. Mã Nguồn

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

# col 0x1 col 0x2 col 0x4 col 0x8

#

# row 0x1 0 1 2 3

# 0x11 0x21 0x41 0x81

#

# row 0x2 4 5 6 7

# 0x12 0x22 0x42 0x82

#

# row 0x4 8 9 a b

# 0x14 0x24 0x44 0x84

#

# row 0x8 c d e f

# 0x18 0x28 0x48 0x88

#

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

# command row number of hexadecimal keyboard (bit 0 to 3)

# Eg. assign 0x1, to get key button 0,1,2,3

# assign 0x2, to get key button 4,5,6,7

# NOTE must reassign value for this address before reading,

# eventhough you only want to scan 1 row

.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 $t3, 0x1 # check row 1 with key 0,1,2,3

li $t4, 0x2 # check row 2 with key 4,5,6,7

li $t5, 0x4 # check row 3 with key 8,9,a,b

li $t6, 0x8 # check row 4 with key c,d,e,f

polling:

polling\_row1:

sb $t3, 0($t1 ) # must reassign expected row

lb $a0, 0($t2) # read scan code of key button

bnez $a0, print

polling\_row2:

sb $t4, 0($t1 ) # must reassign expected row

lb $a0, 0($t2) # read scan code of key button

bnez $a0, print

polling\_row3:

sb $t5, 0($t1 ) # must reassign expected row

lb $a0, 0($t2) # read scan code of key button

bnez $a0, print

polling\_row4:

sb $t6, 0($t1 ) # must reassign expected row

lb $a0, 0($t2) # read scan code of key button

bnez $a0, print

print: li $v0, 34 # print integer (hexa)

syscall

sleep: li $a0, 100 # sleep 100ms

li $v0, 32

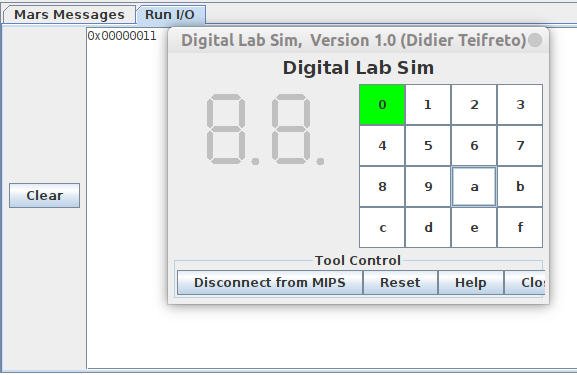
syscall

back\_to\_polling:

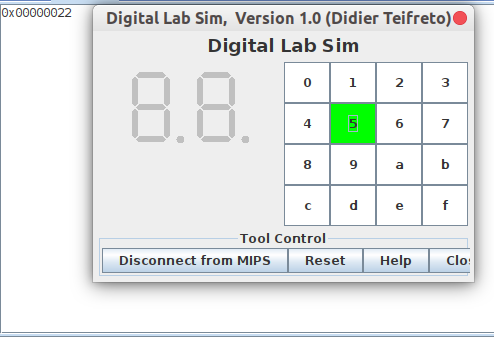
j polling # continue polling

1. **Kết quả**

**-Khi nhấn 1 phím hàng đầu :**



**- Khi nhấn 1 phím ở hàng 2 :**



**Assignment 2**

**1.Mã nguồn**

.eqv IN\_ADRESS\_HEXA\_KEYBOARD 0xFFFF0012

.data

Message: .asciiz "Oh my god. Someone's presed a button.\n"

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# 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 of = 1 to enable interrupt

sb $t3, 0($t1)

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

# No-end loop, main program, to demo the effective of interrupt

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

Loop: nop

nop

nop

nop

b Loop # Wait for interrupt

end\_main:

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# GENERAL INTERRUPT SERVED ROUTINE for all interrupts

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

.ktext 0x80000180

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

# Processing

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

IntSR: addi $v0, $zero, 4 # show message

la $a0, Message

syscall

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

# Evaluate the return address of main routine

# epc <= epc + 4

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

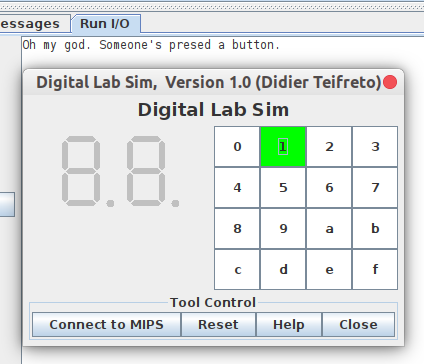
next\_pc:mfc0 $at, $14 # $at <= Coproc0.$14 = Coproc0.epc

addi $at, $at, 4 # $at = $at + 4 (next instruction)

mtc0 $at, $14 # Coproc0.$14 = Coproc0.epc <= $at

return: eret # Return from exception

**2.Kết quả**

****

**Assignment 3**

1. **Mã nguồn :**

.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 $at because we may change it later

sw $at,0($sp)

addi $sp,$sp,4 # Save $sp 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 $t6, 0x1

li $t3, 0x81 # check row 4 and re-enable bit 7

get\_cod:li $t1, IN\_ADRESS\_HEXA\_KEYBOARD

bgt $t3, 0x88, reset\_getcod # check row 4 and re-enable bit 7

sb $t3, 0($t1) # must reassign expected row

li $t1, OUT\_ADRESS\_HEXA\_KEYBOARD

lb $a0, 0($t1)

bnez $a0, prn\_cod

mul $t6, $t6, 2

add $t3, $t6, 0x80

j get\_cod

prn\_cod:li $v0,34

syscall

li $v0,11

li $a0,'\n' # print endofline

syscall

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

# 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 REG FILE from STACK

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

restore:lw $t3, 0($sp) # Restore the registers from stack

addi $sp,$sp,-4

lw $t1, 0($sp) # Restore the registers from stack

addi $sp,$sp,-4

lw $a0, 0($sp) # Restore the registers from stack

addi $sp,$sp,-4

lw $v0, 0($sp) # Restore the registers from stack

addi $sp,$sp,-4

lw $ra, 0($sp) # Restore the registers from stack

addi $sp,$sp,-4

lw $ra, 0($sp) # Restore the registers from stack

addi $sp,$sp,-4

return: eret # Return from exception

reset\_getcod:

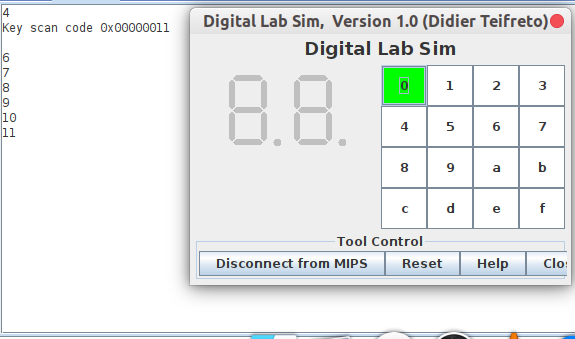
li $t3, 0x81

li $t6, 0x1

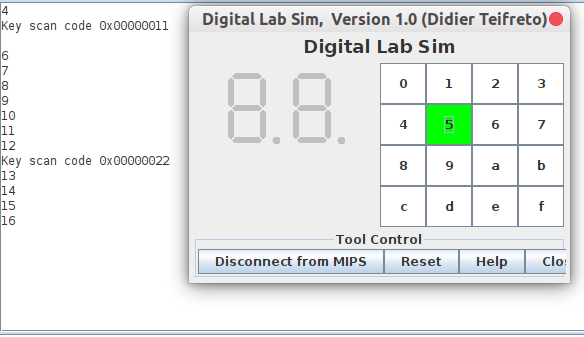
j get\_cod

1. **Kết quả :**

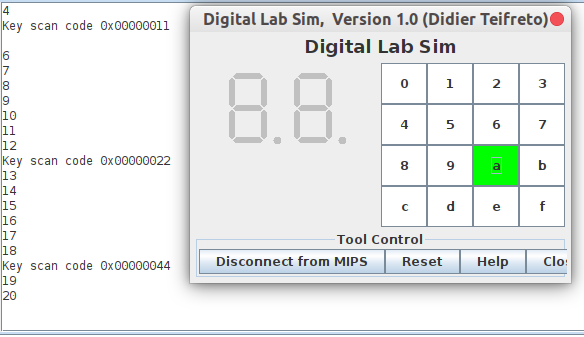
* **Khi nhấn nút ở hàng 1 :**



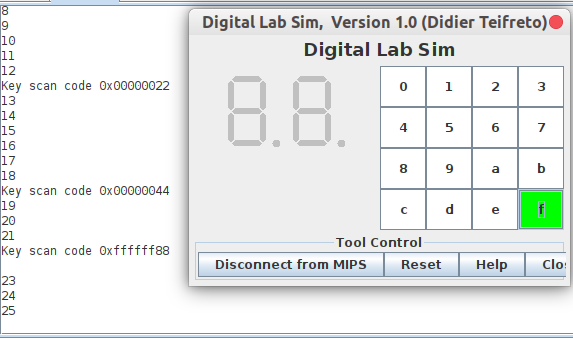
* **Khi nhấn nút ở hàng 2 :**



* **Khi nhấn nút ở hàng 3 :**



* **Khi nhấn nút ở hàng 4 :**



**Assignment 4:**

**1.Mã Nguồn**

.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 interrupt

.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 Sim

li $t1, IN\_ADRESS\_HEXA\_KEYBOARD

li $t3, 0x80 #

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 Counter

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 Counter..

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

# epc <= epc + 4

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

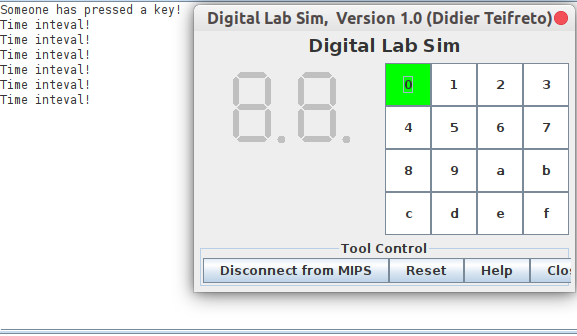
next\_pc:mfc0 $at, $14 # $at <= Coproc0.$14 = Coproc0.epc

addi $at, $at, 4 # $at = $at + 4 (next instruction)

mtc0 $at, $14 # Coproc0.$14 = Coproc0.epc <= $at

return: eret # Return from exception

**2.kết quả**

****

**Assignment 5**

1. **Mã nguồn**

.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 Keyboard..

and $at, $t1,$t2

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

1. Kết quả

Before you finish the laboratory exercise, think about the questions below:

**What is polling?**

Thăm dò là việc chủ động lấy trạng thái của của thiết bị bằng một chương trình như là hoạt động đồng bộ

**What are interrupts?**

Ngắt là việc chờ đợi tin hiệu của thiết bị, và thực hiện việc kiểm tra trang thái.

**What are interrupt routines?**

Là Exception: xảy ra khi có lỗi trong quá trình chạy, chẳng hạn tham chiếu bộ

nhớ không hợp lệ.

**What are the advantages of polling?**

Polling all row of matrix

**What are the advantages of using interrupts?**

- INTERRUPT & STACK

- INTERRUPT & STACK

- Ngắt mềm

**What are the differences between interrupts, exceptions and traps?**

- Exception: xảy ra khi có lỗi trong quá trình chạy, chẳng hạn tham chiếu bộ

nhớ không hợp lệ.

- Trap: xảy ra bởi cách lệnh kiểm tra

- Interrupt: do các thiết bị bên ngoài kích hoạt