**Lớp: 147799 – Học phần: Thực Hành Kiến Trúc Máy Tính**

**Họ và tên: Lương Văn Khanh - MSSV: 20225728**

**Asg1:**

* Code:

.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

.data

el: .asciiz "\n"

.text

main:

li $t1, IN\_ADRESS\_HEXA\_KEYBOARD

li $t2, OUT\_ADRESS\_HEXA\_KEYBOARD

li $t3, 0x01 # check row 4 with key 0,1,2,3

li $t4, 0x02 # check row 4 with key 4,5,6,7

li $t5, 0x04 # check row 4 with key 8,9,A,B

li $t6, 0x08 # check row 4 with key C, D,E, F

li $t0, 0

polling:

beq $t0, 20, exit

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

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

bne $a0, $zero, print

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

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

bne $a0, $zero, print

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

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

bne $a0, $zero, print

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

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

bne $a0, $zero, print

j con

print:

li $v0, 34 # print integer (hexa)

syscall

la $a0, el

li $v0, 4

syscall

con:

addi $t0, $t0, 1

sleep:

li $a0, 2000 # sleep 2s

li $v0, 32

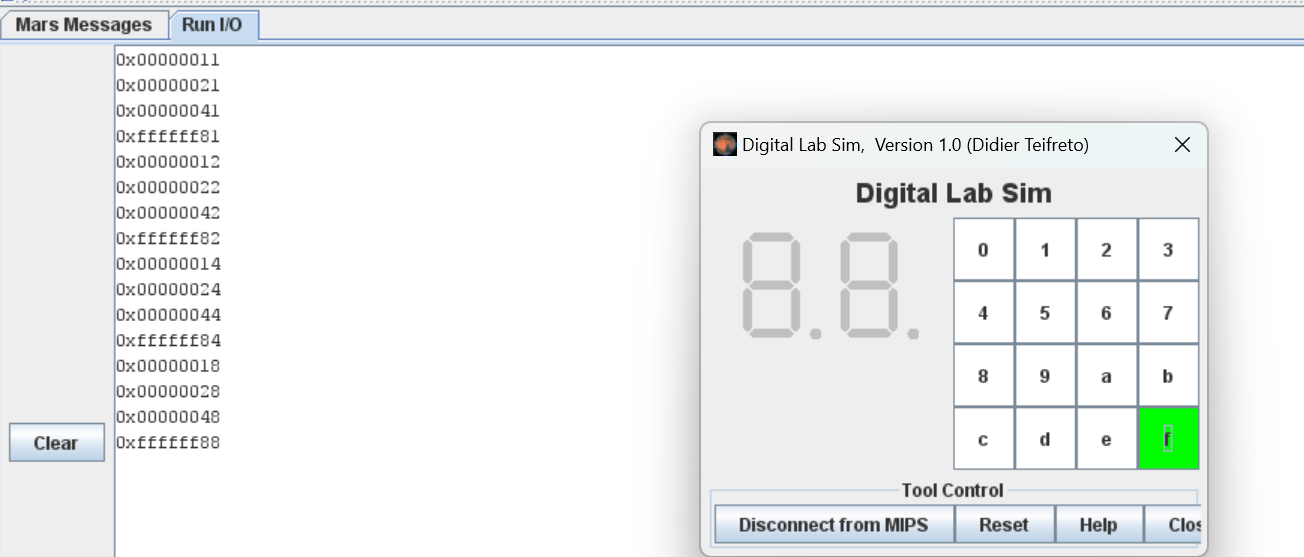
syscall

back\_to\_polling:

j polling # continue polling

exit:

* Kết quả chạy : lần lượt ấn phím từ 0-F thì kết quả là



Asg2:

* Code:

.eqv IN\_ADDRESS\_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\_ADDRESS\_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

addi $v0, $zero, 32

li $a0, 200

syscall

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:

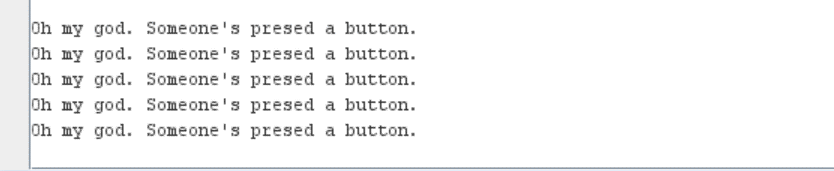
eret # Return from exception

* Giải thích code : Chương trình trên bao gồm hai phần chính: phần chương trình chính và phần xử lý ngoại lệ (interrupt).

+ Phần chương trình chính: Đầu tiên, địa chỉ của bộ điều khiển bàn phím 4x4 được lưu trữ trong thanh ghi $t1 và giá trị 0x80 được lưu trữ trong thanh ghi $t3. Giá trị này được sử dụng để kích hoạt ngắt bàn phím.Tiếp theo, một vòng lặp vô hạn được tạo ra để đợi cho ngắt bàn phím xảy ra.

+Phần xử lý ngoại lệ (interrupt):Địa chỉ bắt đầu của phần xử lý ngoại lệ (interrupt) là 0x80000180.Khi có ngắt bàn phím xảy ra, lệnh addi được sử dụng để sao chép giá trị 4 vào thanh ghi $v0 để hiển thị tin nhắn.Sau đó, địa chỉ của chuỗi "Oh my god. Someone's presed a button." được sao chép vào thanh ghi $a0 sử dụng lệnh la.Cuối cùng, lệnh syscall được sử dụng để hiển thị chuỗi thông qua console.Tiếp theo, địa chỉ của lệnh tiếp theo trong chương trình chính được tính bằng cách tăng giá trị thanh ghi $at (điều này đại diện cho bộ đếm lệnh) thêm 4 và lưu vào lại thanh ghi $14 (đại diện cho thanh ghi epc).Lệnh eret được sử dụng để quay về chương trình chính.

* Kết quả chạy :



Asg3:

* Code:

.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'

syscall

sleep:

addi $v0,$zero,32

li $a0,300 # sleep 0,3 s

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

get\_cod:

li $t1, IN\_ADRESS\_HEXA\_KEYBOARD

li $t3, 0x81 # check row 1 and re-enable bit 7

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

li $t1, OUT\_ADRESS\_HEXA\_KEYBOARD

lb $a0, 0($t1)

bne $a0, $zero, prn\_cod

li $t1, IN\_ADRESS\_HEXA\_KEYBOARD

li $t3, 0x82 # check row 2 and re-enable bit 7

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

li $t1, OUT\_ADRESS\_HEXA\_KEYBOARD

lb $a0, 0($t1)

bne $a0, $zero, prn\_cod

li $t1, IN\_ADRESS\_HEXA\_KEYBOARD

li $t3, 0x84 # check row 3 and re-enable bit 7

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

li $t1, OUT\_ADRESS\_HEXA\_KEYBOARD

lb $a0, 0($t1)

bne $a0, $zero, prn\_cod

li $t1, IN\_ADRESS\_HEXA\_KEYBOARD

li $t3, 0x88 # 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)

bne $a0, $zero, prn\_cod

prn\_cod:

li $v0,34

syscall

li $v0,11

li $a0,'\n'

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

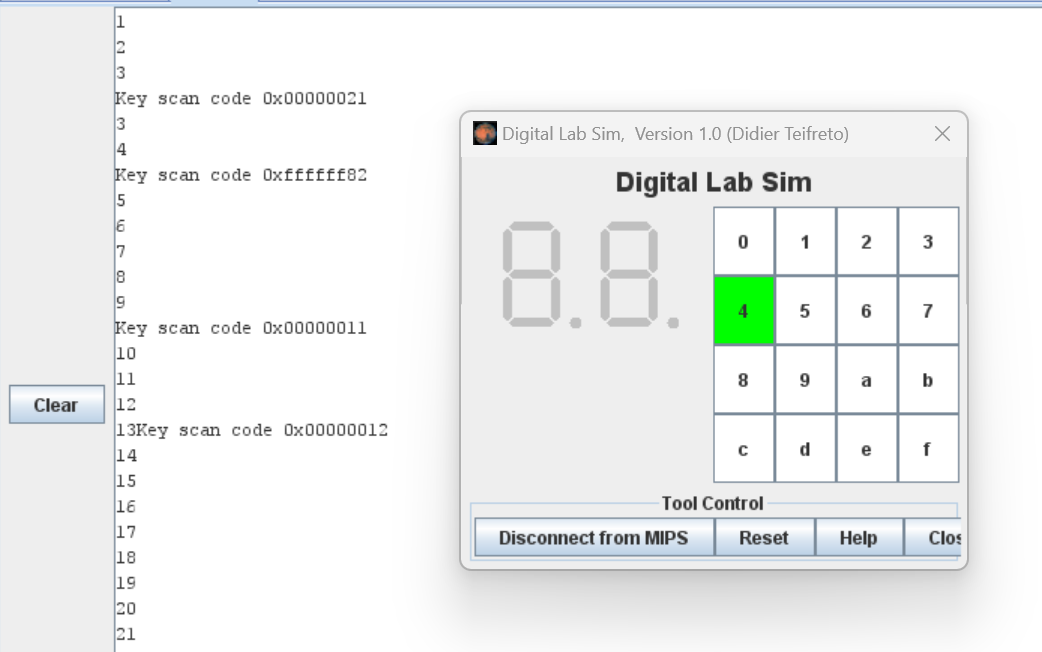
lw $ra, 0($sp) # Restore the registers from stack

addi $sp,$sp,-4

return:

eret # Return from exception

* Kết quả chạy : Nhập phím 1,7,0,4 thì kết quả thu được là



Asg4:

* Code:

.eqv IN\_ADRESS\_HEXA\_KEYBOARD 0xFFFF0012

.eqv OUT\_ADRESS\_HEXA\_KEYBOARD 0xFFFF0014

.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 # 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 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

# read key & print

addi $t3, $zero, 0x11

for:

bgt $t3, 0x88, next\_pc #if value = 0 -> out

li $t1, IN\_ADRESS\_HEXA\_KEYBOARD

sb $t3, 0($t1)

li $t1, OUT\_ADRESS\_HEXA\_KEYBOARD

lb $a0, 0($t1)

bnez $a0, end\_for

sll $t3, $t3, 1

j for

end\_for:

li $v0, 34

syscall

li $v0, 11

li $a0, '\n'

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

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

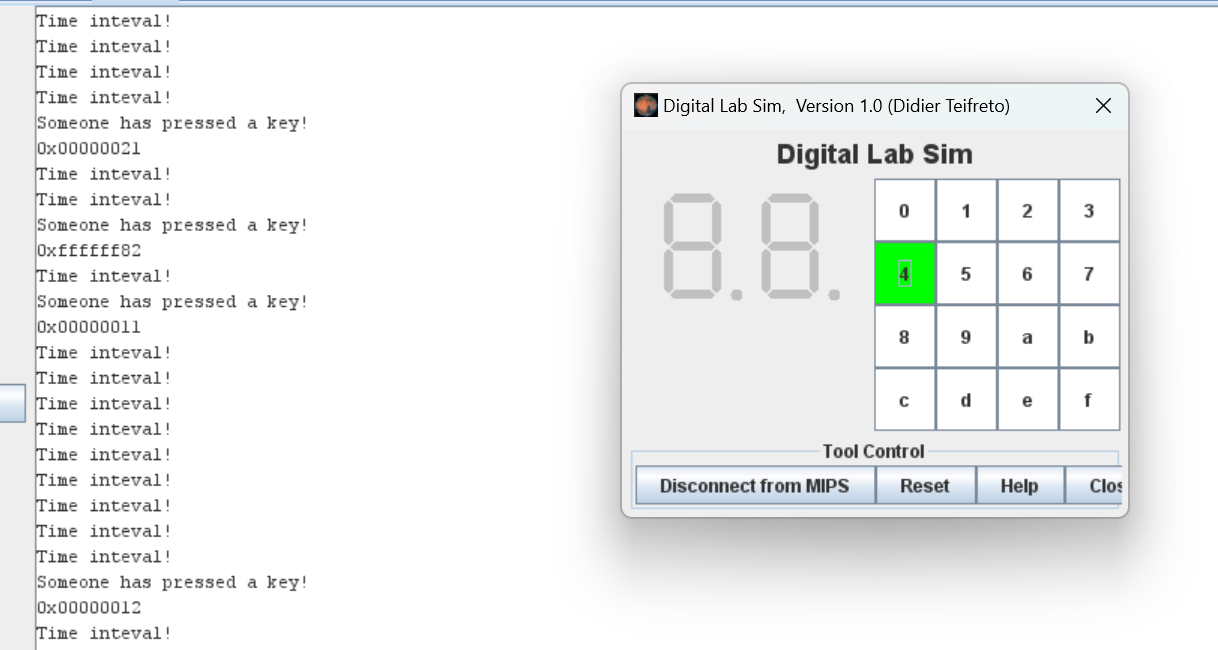
next\_pc:mfc0 $at, $14 # $at <= Coproc0.$14 = Coproc0.epc

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

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

return: eret # Return from exception

* Kết quả chạy : Khi bấm các phím 1,7,0,4 thì kết quả sẽ hiện ra các giá trị hexa của các số đó



Asg5:

* Code:

.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 $t1 = 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

* Kết quả chạy : Nhập chuỗi ah Do Gia Huy k66 dz

