**Báo cáo Thực hành KTMT buổi 12**

**Họ và tên:** Nguyễn Đức Phú

**MSSV:** 20215116

**Assignment 1:**

* Code:

.eqv IN\_ADDRESS\_HEXA\_KEYBOARD 0xFFFF0012

.eqv OUT\_ADDRESS\_HEXA\_KEYBOARD 0xFFFF0014

.text

main:

li $t1, IN\_ADDRESS\_HEXA\_KEYBOARD

li $t2, OUT\_ADDRESS\_HEXA\_KEYBOARD

**li $t3, 0x01 # check row 1 with key C, D,E, F**

**li $t4, 0x02 # check row 2 with key C, D,E, F**

**li $t5, 0x04 # check row 3 with key C, D,E, F**

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

li $t0, 0

polling:

beq $t0, 4, exit #exit after print 4 times

sb $t3, 0($t1 )

lb $a0, 0($t2)

bne $a0, $zero, print

sb $t4, 0($t1 )

lb $a0, 0($t2)

bne $a0, $zero, print

sb $t5, 0($t1 )

lb $a0, 0($t2)

bne $a0, $zero, print

sb $t6, 0($t1 )

lb $a0, 0($t2)

bne $a0, $zero, print

j continue

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

syscall

addi $v0,$zero,11

li $a0,'\n'

syscall

continue:

addi $t0, $t0, 1

sleep:

li $a0, 2500 # sleep 2500ms

li $v0, 32

syscall

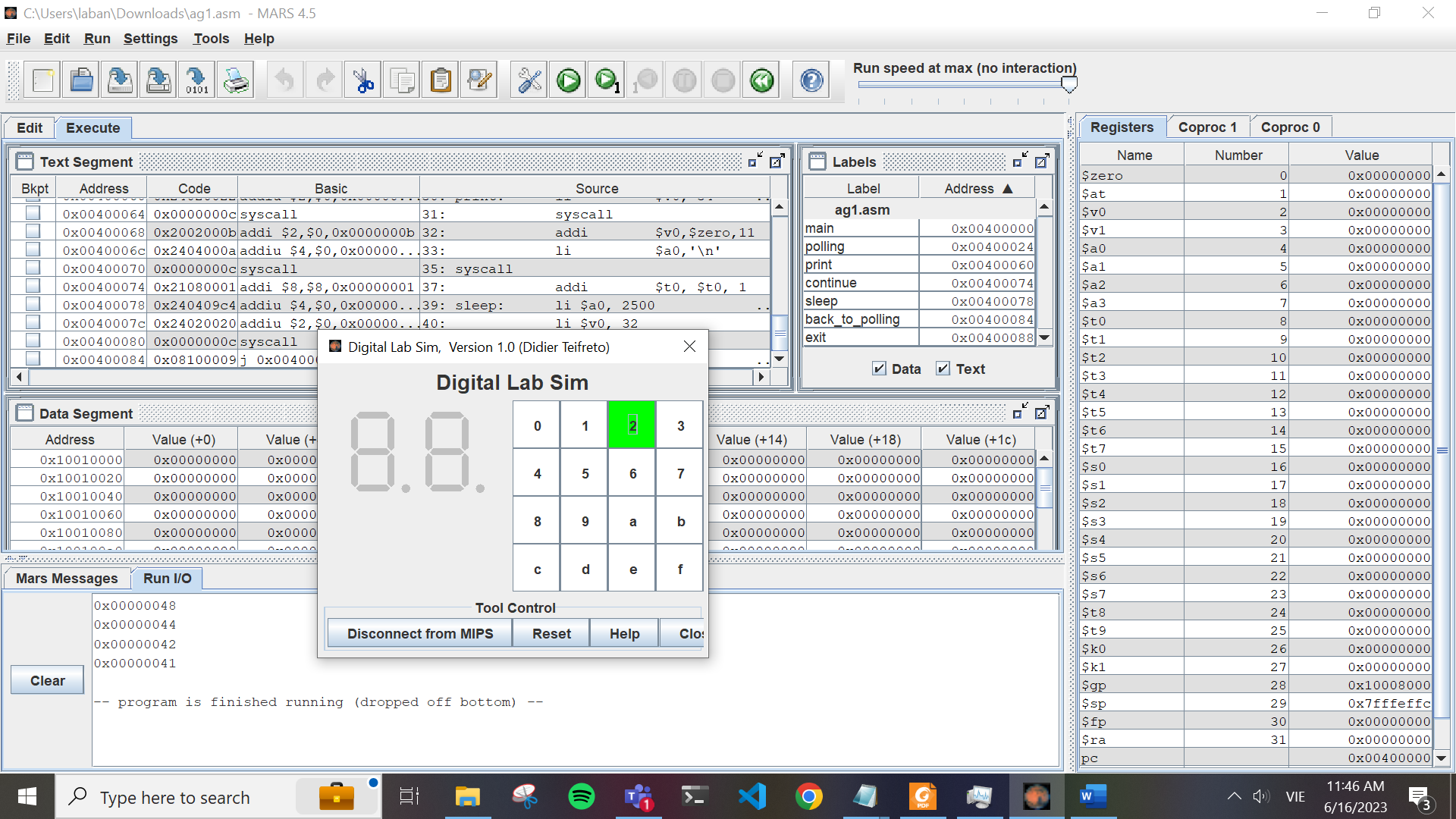
back\_to\_polling:

j polling # continue polling

exit:

* Kết quả chạy thử:

Nhấn lần lượt các phím e, a, 6,2



**Assignment 2:**

* Nhận xét hoạt động:
* Trước khi xảy ra ngắt:

A screenshot of a computer

Description automatically generated with low confidence

* Sau khi xảy ra ngắt:

Không có mô tả.

* Thanh ghi PC nhảy tới giá trị của của .ktext

A screenshot of a computer

Description automatically generated with low confidence

* Giá trị của thanh ghi 12 thay đổi báo hiệu xảy ra ngắt
* Thanh ghi 13 set giá trị 0x00000800 cho biết nguyên nhân là KeyMatrix
* Thanh ghi 14 chứa địa chỉ của lệnh vừa thực hiện xong

Giá trị này được tăng lên 4 bằng các câu lệnh trong chương trình con để sau đó trả về cho thanh ghi PC địa chỉ lệnh kế tiếp

A screenshot of a computer

Description automatically generated with low confidence

* Câu lệnh in ra Message được thực hiện tại chương trình con phục vụ ngắt

A screenshot of a computer

Description automatically generated

**Assignment 3:**

* Code:

.eqv IN\_ADRESS\_HEXA\_KEYBOARD 0xFFFF0012

.eqv OUT\_ADRESS\_HEXA\_KEYBOARD 0xFFFF0014

.data

Message: .asciiz "Key scan code "

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

# MAIN Procedure

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

.text

main:

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 $a0,1000 # sleep 1000 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

sw $ra,0($sp)

addi $sp,$sp,4 # Save $at

sw $at,0($sp)

addi $sp,$sp,4 # Save $sp

sw $v0,0($sp)

addi $sp,$sp,4 # Save $a0

sw $a0,0($sp)

addi $sp,$sp,4 # Save $t1

sw $t1,0($sp)

addi $sp,$sp,4 # Save $t3

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' # 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 from stack

addi $sp,$sp,-4

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

addi $sp,$sp,-4

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

addi $sp,$sp,-4

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

addi $sp,$sp,-4

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

addi $sp,$sp,-4

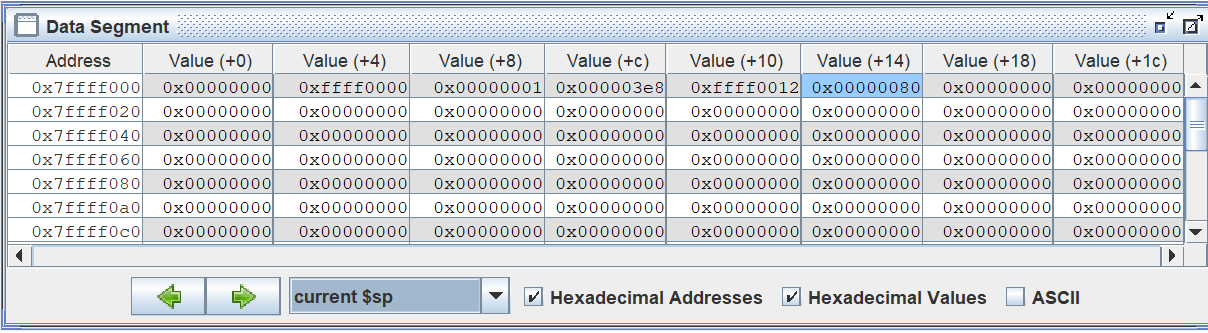
lw $ra, 0($sp) # Restore from stack

addi $sp,$sp,-4

return:

eret # Return from exception

* Kết quả:
* Khi nhấn nút tại Digital Lab Sim: ngắt xảy ra và có các cơ chế tương tự như đã trình bày ở Assignment 2
* Khi ngắt xảy ra giá trị các thanh ghi $ra $at $sp $a0 $t1 $t3 được lưu vào stack:



* Sau đó được restore lại khi kết thúc chương trình con ngắt
* Kết quả chạy:

A screenshot of a computer

Description automatically generated

**Assignment 4:**

* Với trường hợp ngắt do Time counter:

A screenshot of a computer

Description automatically generated

* Giá trị tại thanh ghi 13 là 0x00000400 cho biết nguyên nhân ngắt do time counter
* Giá trị này được and với *MASK\_CAUSE\_COUNTER* và trả về chính nó

Còn khi and với giá trị *MASK\_CAUSE\_KEYMATRIX* thì trả về 0x00000000

* Chương trình nhảy tới nhãn Counter\_Intr và in ra “Time interval!” tại cửa sổ Run I/O
* Sau đó giá trị tại $14 được tăng lên 4 qua các câu lệnh tại cuối chương trình con. Giá trị này được trả về cho thanh ghi PC và tiếp tục chương trình chính
* Với trường hợp ngắt do KeyMatrix:
* Các cơ chế ngắt xảy ra tương tự
* Khác so với trường hợp trên ở giá trị thanh ghi 13

$13 = 0x00000800 cho biết nguyên nhân ngắt do KeyMatrix

* Giá trị này được and với *MASK\_CAUSE\_KEYMATRIX* và trả về chính nó

Còn khi and với giá trị *MASK\_CAUSE\_COUNTER*  thì trả về 0x00000000

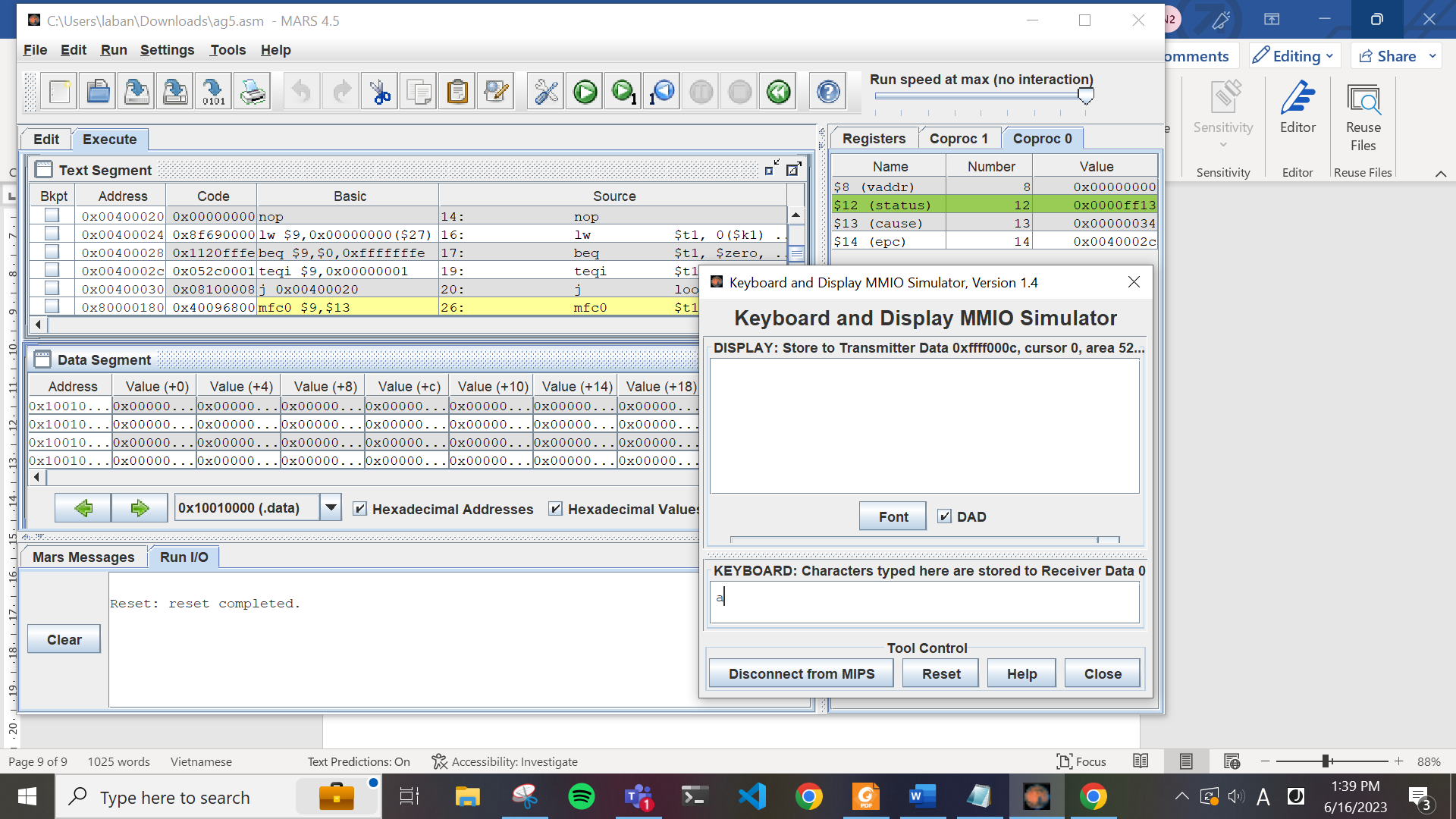
* Chương trình nhảy tới nhãn Keymatrix\_Intr và in ra “Someone has pressed a key” tại cửa sổ Run I/O

A screenshot of a computer

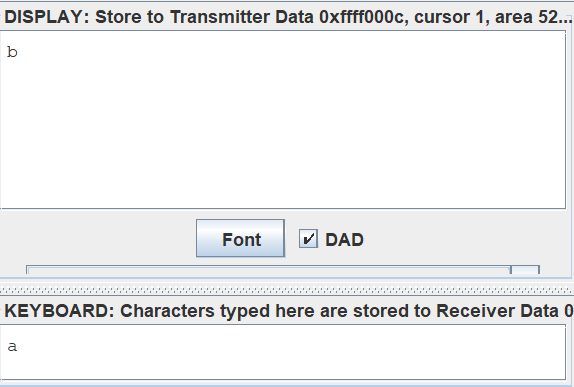
Description automatically generated

* Sau đó giá trị tại $14 được tăng lên 4 qua các câu lệnh tại cuối chương trình con. Giá trị này được trả về cho thanh ghi PC và tiếp tục chương trình chính

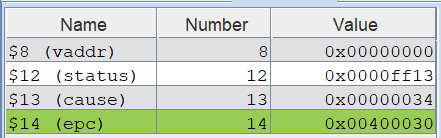
**Assignment 5:**



* Khi nhập ký tự bất kỳ tại Keyboard chương trình thoát khỏi vòng lặp WairForKey và sử dụng lệnh teqi tạo ra ngắt mềm
* Thanh ghi 12 thay đổi trạng thái báo hiệu ngắt
* Thanh ghi 13 thay đổi thành 0x00000034 cho biết nguyên nhân ngắt
* Tại chương trình con phục vụ ngắt: key được đọc vào và tăng giá trị mã ASCII thêm 1 rồi hiển thị ra Display



* Thanh ghi 14 được tăng lên 4 bằng các câu lệnh ở cuối chương trình con



* Trả về cho thanh ghi PC và tiếp tục chương trình chính

