BÁO CÁO THỰC HÀNH KIẾN TRÚC MÁY TÍNH TUẦN 10

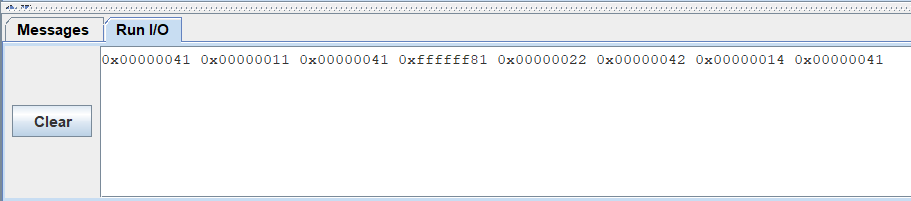
Họ và tên: Nguyễn Quý Đức MSSV: 20235682

Mã HP: IT3280 Mã lớp: 156788

1. Assignment

* Với mỗi dòng (row), nó:
  + Gửi mã dòng (t3) vào địa chỉ IN\_ADDRESS\_HEXA\_KEYBOARD.
  + Đọc mã phím từ địa chỉ OUT\_ADDRESS\_HEXA\_KEYBOARD.
  + Nếu có phím được nhấn (giá trị đọc ≠ 0):
    - In mã phím ra màn hình dưới dạng hexadecimal.
    - Ngủ 100ms để tránh đọc lặp nhanh.
  + Chuyển sang dòng tiếp theo (dịch trái 1 bit).
* Sau khi quét hết 4 dòng (0x01, 0x02, 0x04, 0x08), chương trình quay lại quét tiếp.

Kết quả khi bấm lần lượt từng nút 2-0-2-3-5-6-8-2:



.eqv IN\_ADDRESS\_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\_ADDRESS\_HEXA\_KEYBOARD 0xFFFF0014

.eqv newline 0xa

.text

main:

li t1, IN\_ADDRESS\_HEXA\_KEYBOARD

li t2, OUT\_ADDRESS\_HEXA\_KEYBOARD

polling:

li t3, 0x01 # start with row 1

row\_loop:

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

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

beqz a0, next\_row # if no key pressed, continue to next row

print:

li a7, 34 # print integer (hexa)

ecall

li a0, newline

li a7, 11

ecall

sleep:

li a0, 100 # sleep 100ms

li a7, 32

ecall

next\_row:

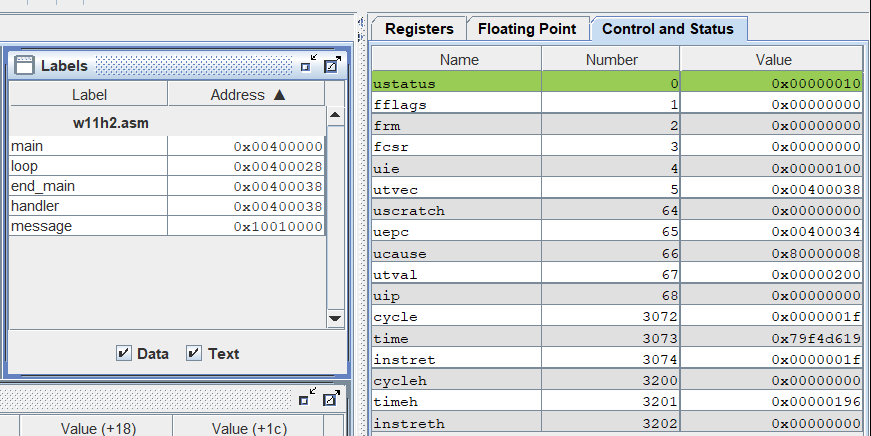
slli t3, t3, 1 # move to next row (shift left 1 bit)

li t4, 0x10 # maximum row mask

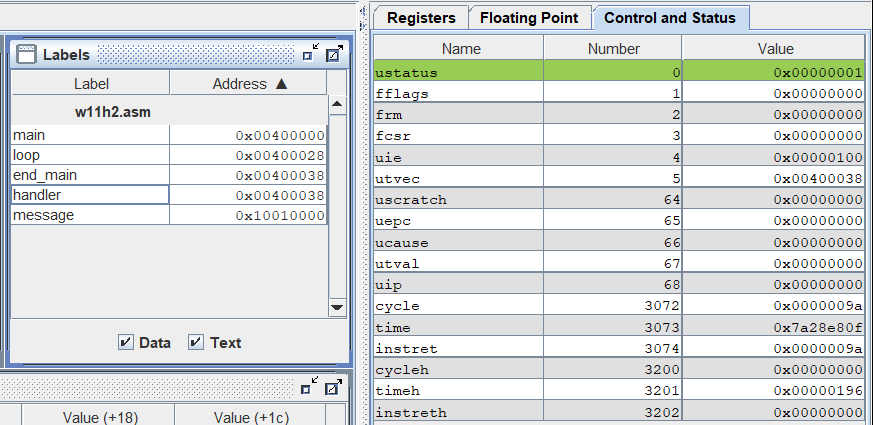
blt t3, t4, row\_loop # if not finished all rows, continue row\_loop

j polling # continue polling

1. Assignment 2

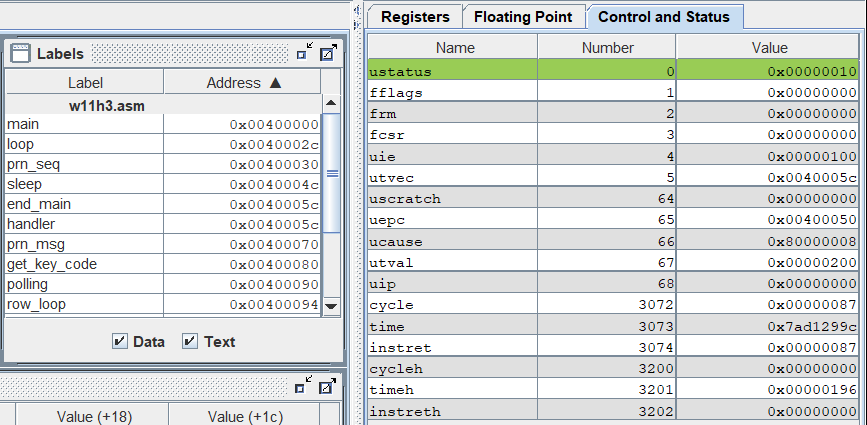


* Trong hình trên là trạng thái các thanh ghi khi xảy ra ngắt do bấm phím ở keypad:
  + ustatus = 0x10 nghĩa là: đang trong lúc xử lý ngắt, chương trình khóa tạm ngắt (0th bit UIE = 0), lưu lại trạng thái trước đó interrupt đã được bật (4th bit UPIE = 1).
  + uie = 0x00000100 nghĩa là: chỉ cho phép ngắt từ thiết bị ngoại vi (8th bit UEIE = 1 User External Interrupt Enable và các bit khác = 0)
  + utvec = 0x00400038 lưu địa chỉ của chương trình con interuption handler “*handler”*
  + uepc = 0x00400034 lưu giá trị thanh ghi pc khi xảy ra lệnh ngắt
  + ucause = 0x80000008 nghĩa là đang xảy ra interupt (31st bit = 1) với nguyên nhân là external interupt (bit[30:0] EXCCODE = 8)



* Trong hình trên là trạng thái chưa xảy ra interupt:
  + ustatus = 0x1 nghĩa là: chưa interupt, cho phép ngắt xảy ra (0th bit UIE = 1)
  + uie = 0x00000100
  + utvec = 0x00400038
  + uepc = 0x00000000 do chưa xảy ra ngắt nên không lưu địa chỉ xảy ra ngắt
  + ucause = 0x00000000 tương tự như trên, nguyên nhân ngắt = 0

1. Assignment 3



* Trong hình trên là trạng thái các thanh ghi khi xảy ra ngắt do bấm phím ở keypad:
  + ustatus = 0x10 nghĩa là: đang trong lúc xử lý ngắt, chương trình khóa tạm ngắt (bit 0 UIE = 0), lưu lại trạng thái trước đó interrupt đã được bật (bit 4 UPIE = 1)
  + uie = 0x00000100 nghĩa là: chỉ cho phép ngắt từ thiết bị ngoại vi (bit 8 UEIE = 1 User External Interrupt Enable và các bit khác = 0)
  + utvec = 0x00400070 lưu địa chỉ của chương trình con interruption handler "handler"
  + uepc = 0x00400050 lưu giá trị thanh ghi pc khi xảy ra lệnh ngắt
  + ucause = 0x80000008 nghĩa là đang xảy ra interrupt (bit 31 = 1), với nguyên nhân là external interrupt (EXCCODE[30:0] = 8)

.eqv IN\_ADDRESS\_HEXA\_KEYBOARD 0xFFFF0012

.eqv OUT\_ADDRESS\_HEXA\_KEYBOARD 0xFFFF0014

.data

message: .asciz "\nKey scan code: "

newline: .asciz "\n"

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

# MAIN Procedure

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

.text

main:

# Load the interrupt service routine address to the UTVEC register

la t0, handler

csrrs zero, utvec, t0

# Set the UEIE (User External Interrupt Enable) bit in UIE register

li t1, 0x100

csrrs zero, uie, t1 # uie - ueie bit (bit 8)

# Set the UIE (User Interrupt Enable) bit in USTATUS register

csrrsi zero, ustatus, 1 # ustatus - enable uie (bit 0)

# Enable the interrupt of keypad of Digital Lab Sim

li t1, IN\_ADDRESS\_HEXA\_KEYBOARD

li t3, 0x80 # bit 7 = 1 to enable interrupt

sb t3, 0(t1)

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

# Loop to print a sequence numbers

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

xor s0, s0, s0 # count = s0 = 0

loop:

addi s0, s0, 1 # count = count + 1

prn\_seq:

addi a7, zero, 1

add a0, s0, zero # Print auto sequence number

ecall

# Print EOL

addi a7, zero, 4

la a0, newline

ecall

sleep:

addi a7, zero, 32

li a0, 300 # Sleep 300 ms

ecall

j loop

end\_main:

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

# Interrupt service routine

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

handler:

# Saves the context

addi sp, sp, -16

sw a0, 0(sp)

sw a7, 4(sp)

sw t1, 8(sp)

sw t2, 12(sp)

# Handles the interrupt

prn\_msg:

addi a7, zero, 4

la a0, message

ecall

get\_key\_code:

li t1, IN\_ADDRESS\_HEXA\_KEYBOARD

li t2, OUT\_ADDRESS\_HEXA\_KEYBOARD

polling:

li t3, 0x01 # start with row 1

row\_loop:

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

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

beqz a0, next\_row # if no key pressed, continue to next row

print:

li a7, 34 # print integer (hexa)

ecall

li a0, 0xa

li a7, 11

ecall

next\_row:

slli t3, t3, 1 # move to next row (shift left 1 bit)

li t4, 0x10 # maximum row mask

blt t3, t4, row\_loop # if not finished all rows, continue row\_loop

# After polling, must re-enable keypad interrupt

re\_enable\_interrupt:

li t1, IN\_ADDRESS\_HEXA\_KEYBOARD

li t3, 0x80 # bit 7 = 1 to enable interrupt again

sb t3, 0(t1)

j restore

restore:

# Restores the context

lw t2, 12(sp)

lw t1, 8(sp)

lw a7, 4(sp)

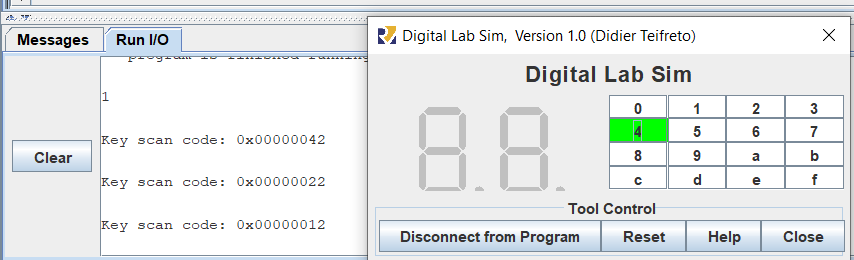
lw a0, 0(sp)

addi sp, sp, 16

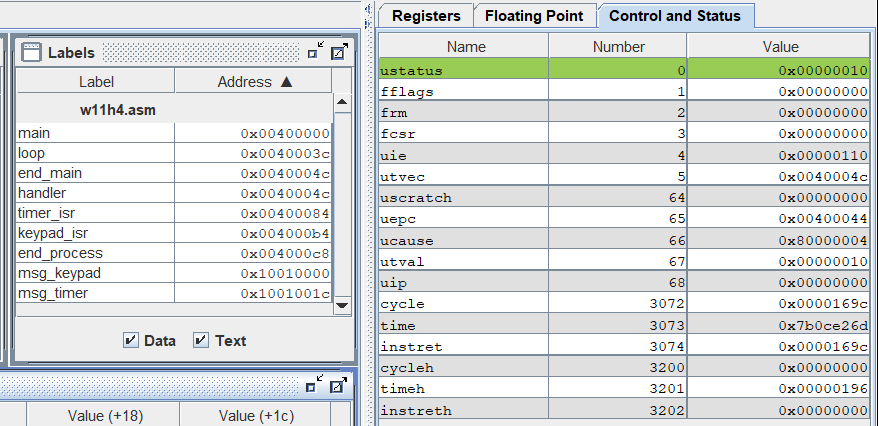
# Back to the main procedure

Uret

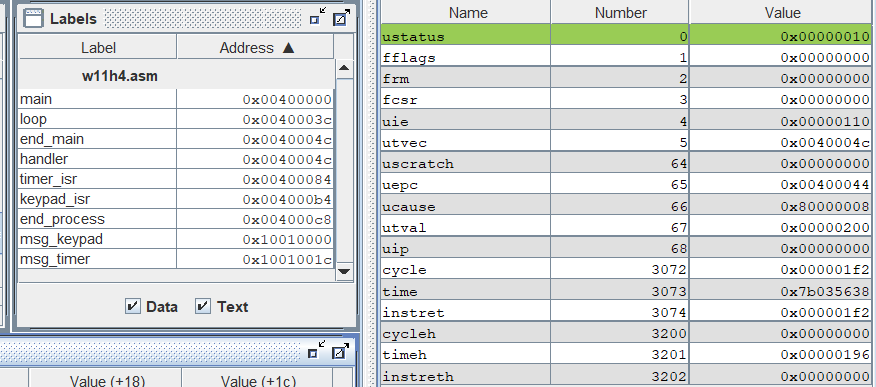
Console hiển thị khi bấm nút 6-5-4 như sau:



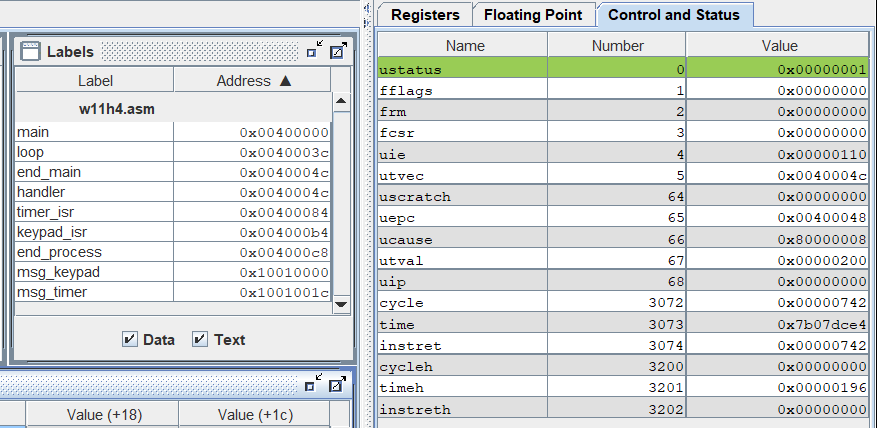
1. Assignment 4



* Trong hình trên là trạng thái các thanh ghi khi xảy ra ngắt do Timer đã đếm được 1000ms:
  + ustatus = 0x00000010 nghĩa là: đang trong lúc xử lý ngắt, chương trình tạm khóa global interrupt (bit 0 UIE = 0), nhưng lưu lại trạng thái trước đó interrupt đã bật (bit 4 UPIE = 1)
  + uie = 0x00000110 nghĩa là: chương trình cho phép cùng lúc 2 loại interrupt:
    - Bit 8 (UEIE) = 1 🡪External interrupt enable (cho phép ngắt từ thiết bị ngoại vi như Keypad)
    - Bit 4 (UTIE) = 1 🡪 Timer interrupt enable (cho phép ngắt từ Timer)
  + utvec = 0x0040004c lưu địa chỉ của chương trình con xử lý ngắt (handler), đúng theo label "handler" bắt đầu từ địa chỉ 0x0040004c
  + uepc = 0x00400044 lưu giá trị PC tại thời điểm xảy ra ngắt, nghĩa là lúc đó chương trình đang thực thi tại địa chỉ 0x00400044
  + ucause = 0x80000004 nghĩa là: đang có interrupt xảy ra (bit 31 = 1), với nguyên nhân là Timer interrupt (EXCCODE[30:0] = 4)

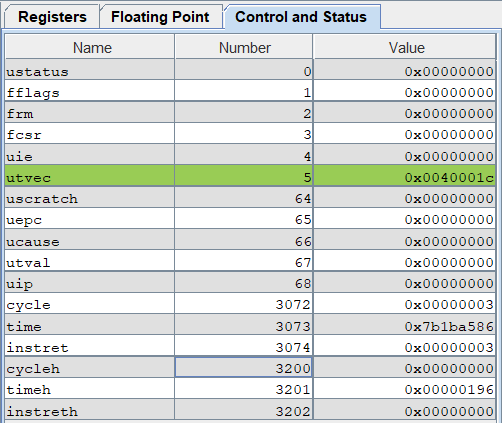


* Trong hình trên là trạng thái các thanh ghi khi xảy ra ngắt do bấm phím ở keypad:
  + ustatus = 0x00000010 nghĩa là: đang trong lúc xử lý ngắt, chương trình tạm khóa global interrupt (bit 0 UIE = 0), đồng thời lưu trạng thái trước đó interrupt đã bật (bit 4 UPIE = 1)
  + uie = 0x00000110 nghĩa là: chương trình cho phép cùng lúc 2 loại interrupt:
    - Bit 8 (UEIE) = 1 🡺 Cho phép External interrupt (thiết bị ngoại vi như Keypad)
    - Bit 4 (UTIE) = 1 🡺 Cho phép Timer interrupt
  + utvec = 0x0040004c lưu địa chỉ của chương trình con xử lý ngắt (handler), khớp với Label "handler" ở địa chỉ 0x0040004c
  + uepc = 0x00400044 lưu giá trị PC tại thời điểm xảy ra ngắt, tức là lúc đó chương trình chính đang thực thi tại địa chỉ 0x00400044
  + ucause = 0x80000008 nghĩa là: có interrupt xảy ra (bit 31 = 1), và nguyên nhân là External interrupt (EXCCODE[30:0] = 8) — tương ứng với ngắt do thiết bị ngoại vi Keypad



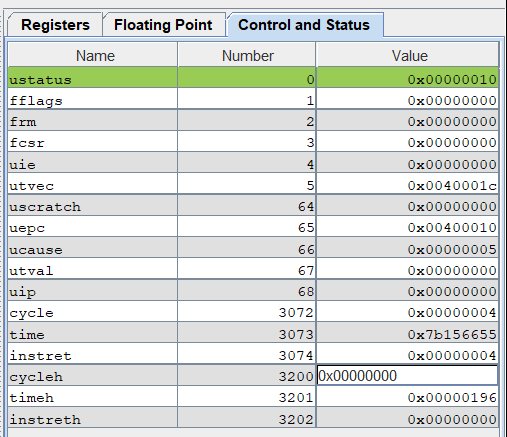
* Trong hình trên là trạng thái các thanh ghi khi bình thường, không bị interupt:
  + ustatus = 0x1 nghĩa là: chưa interupt, cho phép ngắt xảy ra (0th bit UIE = 1)
  + uie = 0x00000110 nghĩa là: chương trình cho phép cùng lúc 2 loại interrupt:
    - Bit 8 (UEIE) = 1 🡺 Cho phép External interrupt (thiết bị ngoại vi như Keypad)
    - Bit 4 (UTIE) = 1 🡺 Cho phép Timer interrupt

1. Assignment 5



Sau lệnh csrrsi zero, ustatus, 1:

* ustatus = 0x00000000 nghĩa là không bật global interruption
* uie = 0x00000000 nghĩa là: chương trình không bật cho phép bất kỳ interrupt nào lúc này (không bật UEIE, UTIE,…)
* utvec = 0x0040011c lưu địa chỉ của chương trình con xử lý exception (handler hoặc tương tự), đang đặt tại địa chỉ 0x0040011c
* uepc = 0x00000000 do chưa xảy ra exeption
* ucause = 0x00000000 do chưa xảy ra exeption



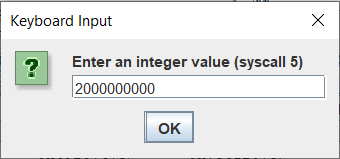
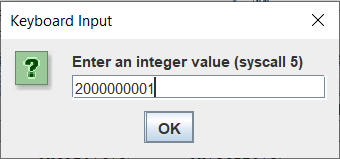
Sau lệnh lw zero, 0

* ustatus = 0x00000010 nghĩa là: chương trình đang trong quá trình xử lý trap/exception, global interrupt bị tạm khóa (bit 0 UIE = 0), nhưng lưu trạng thái trước đó interrupt đã bật (bit 4 UPIE = 1)
* uie = 0x00000000 nghĩa là: chương trình không bật cho phép bất kỳ interrupt nào lúc này (không bật UEIE, UTIE, )
* utvec = 0x0040011c lưu địa chỉ của chương trình con xử lý exception (handler hoặc tương tự), đang đặt tại địa chỉ 0x0040011c
* uepc = 0x00400100 lưu địa chỉ PC khi xảy ra exception, tức là chương trình đã bị trap khi đang thực thi lệnh tại địa chỉ 0x00400100
* ucause = 0x00000005 nghĩa là:
  + Bit 31 = 0 🡪 Đây là exception (không phải interrupt)
  + EXCCODE[30:0] = 5 🡪 Load Access Fault: lỗi truy cập khi thực hiện lệnh load bộ nhớ
* utval = 0x00000010 chứa địa chỉ bộ nhớ gây ra lỗi (ví dụ truy cập vào địa chỉ 0x10 gây lỗi)

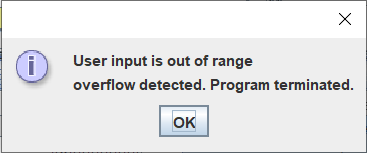
1. Assignment 6

Chương trình tự phát hiện overflow → set bit USIP → CPU thấy có pending software interrupt → nhảy vào handler để xử lý exception.

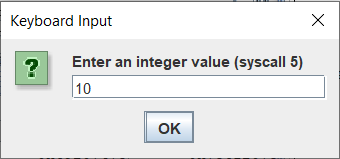
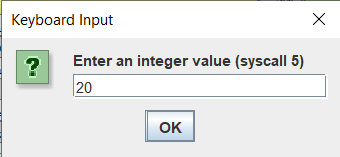
**- Khi người dùng nhập 2 số nguyên có tổng lớn hơn INT32\_MAX = 2147483647**

Sẽ có thông báo tràn số:



**- Khi người dùng nhập 2 số mà tổng không bị tràn:**

Sẽ in ra thông báo tổng 2 số:

