Laboratory Exercise 11 – Report:

Interrupts & IO programming

Lê Văn Duẩn - 20194508

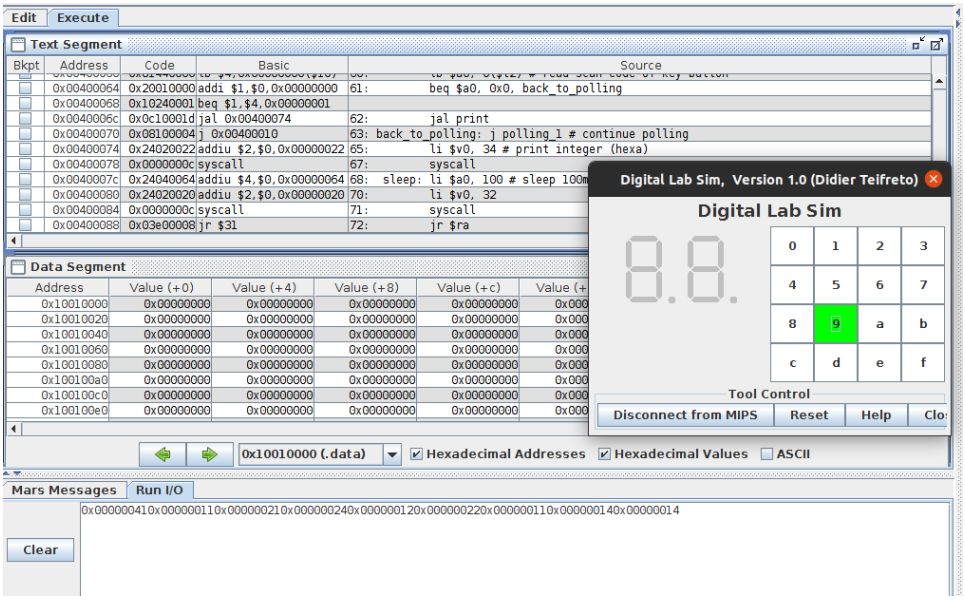
1. Assignment 1

- 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, 0x08 # check row 4 with key C, D, E, F
```

```
37
38  polling_1:li $t3, 0x1 # row 1
39          sb $t3, 0($t1 ) # must reassign expected row
40
41          lb $a0, 0($t2) # read scan code of key button
42
43          beq $a0, 0x0, polling_2
44          j print
45  polling_2:li $t3, 0x2 # row 2
46          sb $t3, 0($t1 ) # must reassign expected row
47
48          lb $a0, 0($t2) # read scan code of key button
49          beq $a0, 0x0, polling_3
50          j print
51   polling_3:li $t3, 0x4 # row 3
52          sb $t3, 0($t1 ) # must reassign expected row
53
54          lb $a0, 0($t2) # read scan code of key button
55          beq $a0, 0x0, polling_4
56          j print
57   polling_4:li $t3, 0x8 # row 4
58          sb $t3, 0($t1 ) # must reassign expected row
59
60          lb $a0, 0($t2) # read scan code of key button
61          j print
62  back_to_polling: j polling_1 # continue polling
63
64   print:
65          li $v0, 34 # print integer (hexa)
66
67          syscall
68   sleep: li $a0, 100 # sleep 100ms
69
70          li $v0, 32
71          syscall
72          j back_to_polling
```

- Kết quả chạy mô phỏng:



- Giải thích:

Các nhãn polling 1,2,3,4 tương ứng với việc quét các dòng trong bàn phím lab Sim bằng cách gán địa chỉ tương ứng từng dòng (ví dụ: row 1 có địa chỉ 0x1) vào thanh ghi t3.

Sau khi quét vào lưu giá trị nhập vào vào thanh ghi $a0, chúng ta kiểm tra nó có bằng 0x0 hay không vì 0x0 nghĩa là không ấn gì vào bàn phím trên dòng đang quét và chương trình sẽ nhảy đến polling tiếp theo để quét . Nếu khác 0x0 nghĩa là bạn đã ấn phím trên hàng này và in gía trị ra màn hình và quay trở lại quét từ đầu.

Trong ví dụ trên test case là 20194508 và ở lần sau số 8 tôi không ấn phím nào nên màn hình in ra gía trị 0x0 tương ứng. Kết quả chạy đúng như mong muốn.

2. Assignment 2

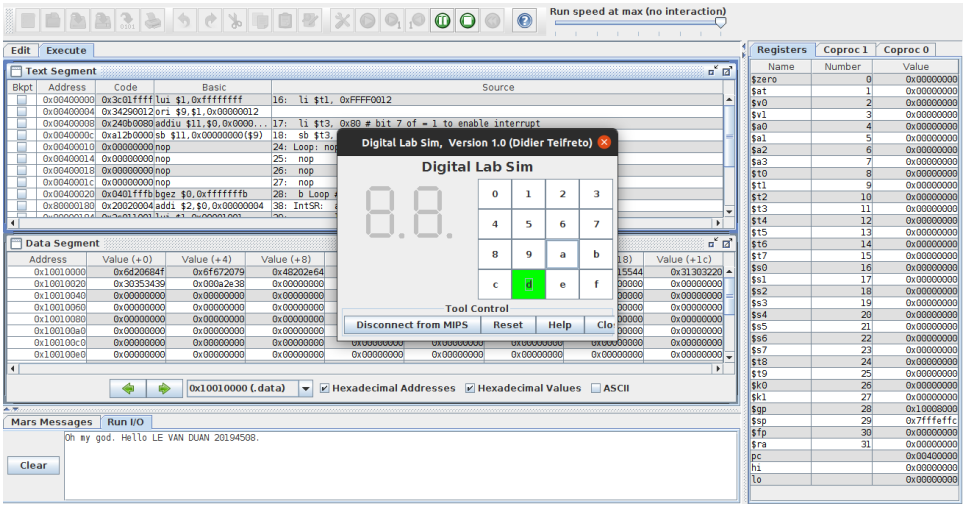
- Mã nguồn:

#Laboratory Exercise 11 Home Assignment 2

```
.eqv IN_ADRESS_HEXA_KEYBOARD 0xFFFF0012
.data

Message: .asciiz "Oh my god. Hello LE VAN DUAN 20194508.\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
 #-----------------------------------------------------------
```

```
38 IntSR:   addi $v0, $zero, 4 # show message
39          la $a0, Message
40          syscall
41  #---------------------------------------------------------
42  # Evaluate the return address of main routine
43  # epc <= epc + 4
44  #---------------------------------------------------------
45
46 next_pc:mfc0 $at, $14 # $at <= Coproc0.$14 = Coproc0.epc
47          addi $at, $at, 4 # $at = $at + 4 (next instruction)
48          mtc0 $at, $14 # Coproc0.$14 = Coproc0.epc <= $at
49 return: eret # Return from exception
```

- Kết quả chạy:



- Giải thích:

Thanh ghi t3 lưu địa chỉ 0x80000180 mà ở đó khi bit thứ 7 bằng 1 chức interrupt được kích hoạt.

sử dụng chỉ thị .ktext để viết code ở địa chỉ 0x80000180 nói trên.

Sau khi kết thúc chương trình con, sử dụng lệnh eret để quay trở lại chương trình chính. Lệnh eret sẽ gán nội dung thanh ghi PC bằng giá trị trong thanh ghi $14 ($t6)

Kết quả khi ấn phím bất kì trên bàn phím thì chương trình in ra message.

3. Assignment 3

-Mã nguồn:

#Laboratory Exercise 11, Home Assignment 3

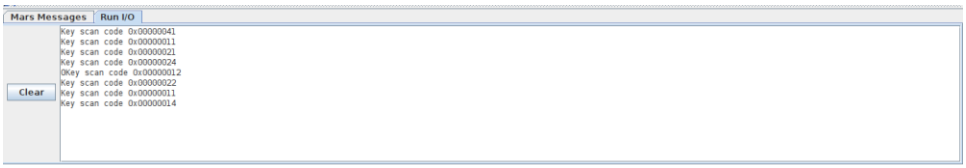
```
.eqv IN_ADRESS_HEXA_KEYBOARD 0xFFFF0012
.eqv OUT_ADRESS_HEXA_KEYBOARD 0xFFFF0014
.data

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

```

```
37
38  sleep: addi $v0,$zero,32
39   li $a0,300 # sleep 300 ms
40   syscall
41   nop # WARNING: nop is mandatory here.
42   b Loop # Loop
43  end_main:
44
45  #~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
46  # GENERAL INTERRUPT SERVED ROUTINE for all interrupts
47  #~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
48
49  .ktext 0x80000180
50
51   #-----------------------------------------------------
52   # SAVE the current REG FILE to stack
53   #-----------------------------------------------------
54
55  IntSR: addi $sp,$sp,4 # Save $ra because we may change it later
56   sw $ra,0($sp)
57   addi $sp,$sp,4 # Save $ra because we may change it later
58   sw $at,0($sp)
59   addi $sp,$sp,4 # Save $ra because we may change it later
60   sw $v0,0($sp)
61   addi $sp,$sp,4 # Save $a0, because we may change it later
62   sw $a0,0($sp)
63   addi $sp,$sp,4 # Save $t1, because we may change it later
64   sw $t1,0($sp)
65   addi $sp,$sp,4 # Save $t3, because we may change it later
66   sw $t3,0($sp)
67   #-----------------------------------------------------
68   # Processing
69   #-----------------------------------------------------
70
71  prn_msg:addi $v0, $zero, 4
72   la $a0, Message
73   syscall
```

```
 syscall

get_cod:li $t1, IN_ADRESS_HEXA_KEYBOARD

 interrupt_1: 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)
 beq $a0, 0x0, interrupt_2
 j prn_cod

interrupt_2: 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)
 beq $a0, 0x0, interrupt_3
  j prn_cod

 interrupt_3: 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)
 beq $a0, 0x0, interrupt_4
  j prn_cod

 interrupt_4: 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)
 beq $a0, 0x0, next_pc

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

return: eret # Return from exception
```

- Kết quả chạy mô phỏng:



- Giải thích:

Để kích enable interrup tương tự assignment 2 chúng ta gán giá trị 0x80000180 và viết code ở .ktext

Nhãn IntSR dùng stack với con trỏ stack là thanh ghi $sp để lưu tất cả các trạng thái có thể bị thay đổi vào stack bao gồm $ra, $at, $a0, $t1 và $t3

Nhãn restore để trả lại các giá trị trạng thái đã lưu bằng các pop từng phần tử ra từ stack theo thứ tự

Tại nhãn get_cod để có thể check tất cả các row của bàn phím lab sim ta cần thêm code để check các row còn lại tương tự như assignment 1 tuy nhiên khi gán địa chỉ cho row cần chuyển bit thứ 7 sang 1 để re-enable interrupt vì vậy địa chỉ gán vào $t3 sẽ thay đổi thành: row 1: 0x81 tương tự thì row2, 3, 4 sẽ là 0x82, 0x84 và 0x88.

Kết quả test case 20194508 được hiển thị ở console như hình và đúng với giá trị mong muốn.