Nguyễn Thế Đức 20194515

Báo cáo bài tập tuần 11

Assignment 1

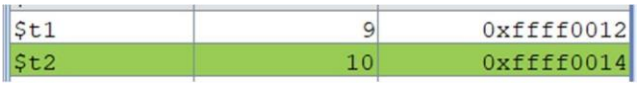
Chương trình:

```
#-----------------------------------------------------------------------------
# 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

polling:
li $t3, 0x01 # check row 4 with key 0,1,2,3
sb $t3, 0($t1 ) # must reassign expected row
lb $a0, 0($t2) # read scan code of key button
li $v0, 34 # print integer (hexa)
syscall
li $a0, 100 # sleep 100ms
li $v0, 32
syscall

li $t3, 0x02 # check row 4 with key 4,5,6,7
```

```
sb $t3, 0($t1 ) # must reassign expected row
lb $a0, 0($t2) # read scan code of key button
li $v0, 34 # print integer (hexa)
syscall
li $a0, 100 # sleep 100ms
li $v0, 32
syscall

li $t3, 0x04 # check row 4 with key 8,9,a,b
sb $t3, 0($t1 ) # must reassign expected row
lb $a0, 0($t2) # read scan code of key button
li $v0, 34 # print integer (hexa)
syscall
li $a0, 100 # sleep 100ms
li $v0, 32
syscall

li $t3, 0x08 # check row 4 with key C, D, E, F
sb $t3, 0($t1 ) # must reassign expected row
lb $a0, 0($t2) # read scan code of key button
li $v0, 34 # print integer (hexa)
syscall
li $a0, 100 # sleep 100ms
li $v0, 32
syscall

back_to_polling: j polling # continue polling
```

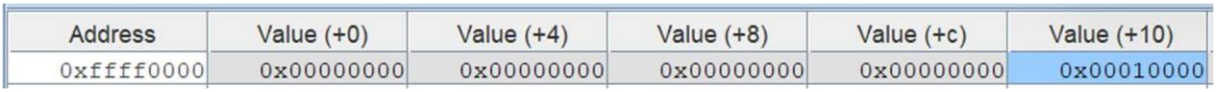
Giải thích:

Đầu tiên, gắn địa chỉ $t1, $t2 là địa chỉ input và output của dữ liệu được nhập:



$t1	9	0xffff0012
$t2	10	0xffff0014

Tiếp theo, gắn $t3 = 0x1 để quét hàng đầu tiên (0, 1, 2, 3) và lưu giá trị này vào IN_ADDRESS.



Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)
0xffff0000	0x00000000	0x00000000	0x00000000	0x00000000	0x00010000

Khi nhấn 2, OUT_ADDRESS nhận giá trị 0x41:

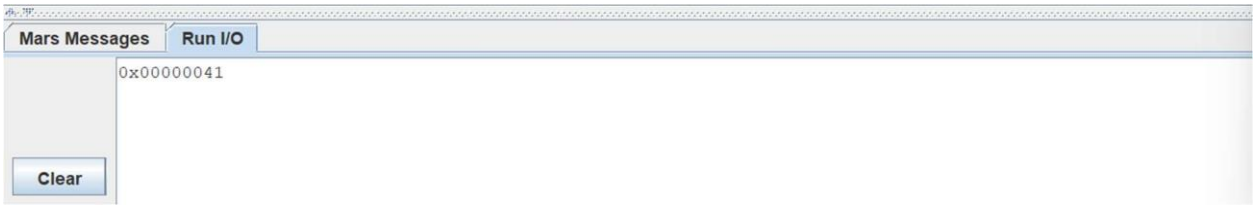
Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+
0xffff0000	0x00000000	0x00000000	0x00000000	0x00000000	0x00010000	0x00000041	0x00000000	0x0000

Giá trị này sau đó được load vào a0:

$a0	4	0x00000041

Tiếp theo dùng lệnh syscall để in ra màn hình:



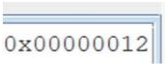
Sau đó, lần lượt quét các hàng tiếp theo.

Tiếp theo, ta nhấn 0, tương tự như trên, khi $t3 được gán giá trị 0x01, ta sẽ nhận được giá trị 0x11 tương ứng với số 0:

Tiếp theo, ta ấn 1 và vòng lặp tương tự, ta thu được 0x21:

Khi ấn 9, lúc này có sự thay đổi. Khi $t3 được gán giá trị 0x04 để quét hàng thứ 3, ta sẽ thu được 0x24 tương ứng với 9:

Với số 4, giá trị được nhận khi $t3 mang giá trị $0x02. Khi này, giá trị nhận được là 0x12:



Tương tự, với lần lượt các số 5, 5, 0:

Assignment 2

Chương trình:

```
.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
nop
#-------------------------------------------------------------------------------------------------------------------------
# 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
nop
```

Giải thích:

Đầu tiên, ta gán $t1 địa chỉ của IN_ADDRESS, lưu 0x80 vào địa chỉ này để bắt được lỗi.

Sau đó ta vào vòng lặp vô hạn để đợi interrupts.

Thanh ghi $pc lúc này:

pc		0x00400020

Khi ta nhấn nút, chương trình tự động nhảy đến .ktext:

pc		0x80000180

Đồng thời địa chỉ của câu lệnh vừa thực hiện được lưu lại:

$8 (vaddr)	8	0x00000000
$12 (status)	12	0x0000ff13
$13 (cause)	13	0x00000800
$14 (epc)	14	0x00400020

Thông báo lỗi được in ra:

```
Oh my god. Someone's presed a button.
```

Lệnh mfc0 load địa chỉ của thanh ghi trước đó vào $at:

$at	1	0x00400020

Sau đó tăng giá trị này lên 4 và gọi lệnh mtc0 để lưu lại giá trị này vào thanh ghi epc

Gọi lệnh eret để tiếp tục chương trình. Lúc này đã thực hiện xong hàm main nên kết thúc chương trình:

```
Oh my god. Someone's presed a button.

-- program is finished running (dropped off bottom) --
```

Assignment 3

Chương trình:

```
.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 LabSim
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 $ra because we may change it later
sw $at,0($sp)
addi $sp,$sp,4 # Save $ra 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, 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)
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
```

Giải thích:

Đầu tiên, ta gán $t1 địa chỉ của IN_ADDRESS, lưu 0x80 vào địa chỉ này để bắt được lỗi.

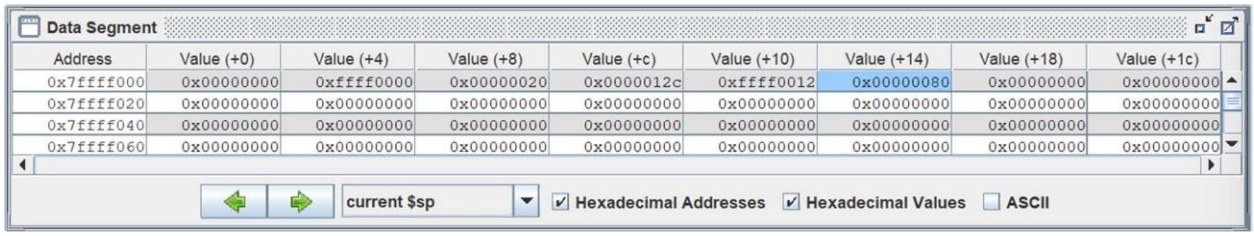
Sau đó ta gán biến chạy = 0

Vào vòng lặp, ta thực hiện tăng biến chạy lên 1 và in giá trị biến chạy ra:

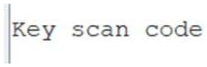
Khi ta ấn 1 số, ví dụ 5, chương trình tự động nhảy đến .ktext như bài 2, thanh ghi epc vẫn lưu giá trị địa chỉ câu lệnh vừa thực hiện:

Name	Number	Value
$8 (vaddr)	8	0x00000000
$12 (status)	12	0x0000ff13
$13 (cause)	13	0x00000800
$14 (epc)	14	0x00400040

Sau đó khai báo stack và lưu các giá trị cần thiết:



Sau đó in ra message:



Gán $t1 địa chỉ của IN_ADDRESS, $t3 0x88 để check hàng thứ 4 và bật bit 7 để bắt được lỗi. Vì 5 không thuộc hàng 4 nên giá trị sau khi được load vào $a0 là 0:

$a0	4	0x00000000

Sau đó in giá trị này ra màn hình:

```
Key scan code 0x00000000
```

Tiếp theo load giá trị epec vào $at, cộng thêm 4 và lưu ngược lại để trỏ đến câu lệnh tiếp tương tự bài 2.

Trước khi gọi eret để tiếp tục chương trình, ta pop các giá trị trong stack đã được lưu trước đó

Vì đã kết thúc hàm main nên chương trình kết thúc:

```
1
Key scan code 0x00000000

-- program is finished running (dropped off bottom) --
```

Tương tự như trên, ta chạy chương trình khi nhấn d. Ở bước in key code, ta nhận được giá trị 0x28 tại a0 và in ra màn hình:

```
1
Key scan code 0x00000028
```