

Bài 4

```
.eqv IN_ADRESS_HEXА_KEYBOARD 0xFFFF0012
.eqv COUNTER 0xFFFF0013 # Time Counter
.eqv OUT_ADRESS_HEXА_KEYBOARD 0xFFFF0014
.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 code: "
```

```
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_HEXА_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 and print sequence numbers

#-----

Loop: nop

    nop

    nop

sleep: addi $v0,$zero,32 # BUG: must sleep to wait for Time Counter

    li $a0,400    # 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

    get_cod:


    li   $t1, IN_ADRESS_HEXА_KEYBOARD

    li   $t3, 0x82  # check row 4 and re-enable bit 7

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

    li   $t1, OUT_ADRESS_HEXА_KEYBOARD

    lb   $a0, 0($t1)

    bne  $a0, $zero, prn_cod


    li   $t1, IN_ADRESS_HEXА_KEYBOARD

```

```

li    $t3, 0x84    # 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

```

```

        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

```

```

li    $t1, IN_ADRESS_HEXa_KEYBOARD
li    $t3, 0x81    # 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
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)

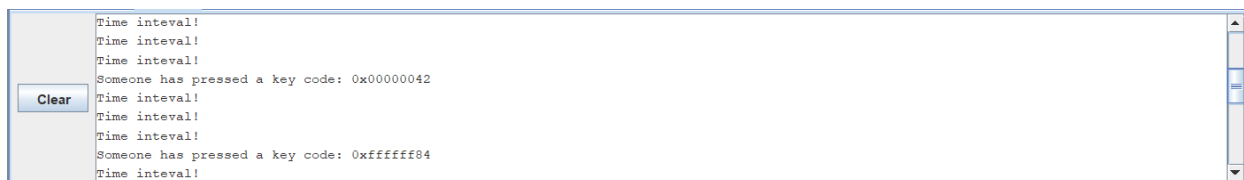
    mtc0 $at, $14        # Coproc0.$14 = Coproc0.epc <= $at


return: eret              # Return from exception

```

kết quả khi chạy chương trình

Bấm các nút với 6 và b thì kết quả thu được



Khi bấm vào digital lab sim và chạy thì nếu không bấm nút nào sau 1 khoảng thời gian qui định sẽ hiện chữ time interval hoặc nếu có bấm 1 nút nào đó thì sẽ hiện dòng someone has pressed a key code : và sau đó đọc code của nút đã bấm

Bài 5

```
.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 0x00000034 # 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 $t0 = 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

The screenshot displays the 'Keyboard and Display MMIO Simulator, Version 1.4' interface. It features a 'Text Segment' table with assembly instructions, a 'Data Segment' table with memory values, and a central display area for keyboard input and output.

Bkpt	Address	Code	Source	Basic
	0x8000018c	0x102a000132:	beq Sat, St2, Counter Keyboard	
	0x80000190	0x080000cb33:	j end process	
	0x80000194	0x8f48000036: ReadKey:	lw St0, 0(\$k0) # St0 = (\$k0)	
	0x80000198	0x0a2a000039: WaitForDis:	lw St2, 0(\$s1) # St2 = (\$s1)	
	0x8000019c	0x1140fffe39:	beq St2, Seers, WaitForDis # if St2 == 0	
	0x800001a0	0x2108000141: Encrypt:	addi St0, St0, 1 # change input	
	0x800001a4	0xae08000043: ShowKey:	sw St0, 0(\$s0) # show key	
	0x800001a8	0x0000000044:	nop	
	0x800001ac	0x4001700047: next pc:	mfc0 Sat, \$14 # Sat <= Coproc0.\$14	
	0x800001b0	0x2021000448:	addi Sat, Sat, 4 # Sat = Sat + 4 (next)	
	0x800001b4	0x4081700049:	mtc0 Sat, \$14 # Coproc0.\$14 = Coproc0.\$14	
	0x800001b8	0x4200001850: return:	eret # Return from exception	

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)
0xffff0000	0x00000000	0x00000073	0x00000000	0x00000074
0xffff0020	0x00000000	0x00000000	0x00000000	0x00000000
0xffff0040	0x00000000	0x00000000	0x00000000	0x00000000
0xffff0060	0x00000000	0x00000000	0x00000000	0x00000000
0xffff0080	0x00000000	0x00000000	0x00000000	0x00000000
0xffff00a0	0x00000000	0x00000000	0x00000000	0x00000000
0xffff00c0	0x00000000	0x00000000	0x00000000	0x00000000
0xffff00e0	0x00000000	0x00000000	0x00000000	0x00000000
0xffff0100	0x00000000	0x00000000	0x00000000	0x00000000
0xffff0120	0x00000000	0x00000000	0x00000000	0x00000000

The central display area shows 'DISPLAY: Store to Transmitter Data 0xffff000c' and 'KEYBOARD: Characters typed here are stored to Receiver Data 0xffff0004'. The keyboard input field contains the character 's'.

Sau khi chạy chương trình thì chương trình nhận được interrupt từ bàn phím được hiện ở \$13 là 0x34, kí tự được nhập và lưu ở địa chỉ 0xffff4004 mã 0x73 là kí tự 's' sau đó ký tự được encrypt lưu ở địa chỉ 0xffff400c mã 0x74 là kí tự 't' sau đó \$14 trả về địa chỉ trước khi interrupt và tiếp tục chạy chương trình