

Họ và Tên: Hà Trung Chiến
MSSV: 20225794

LAB 11(2)

Assignment 4:

Mã nguồn:

Assignment 4

.eqv IN_ADDRESS_HEX_A_KEYBOARD 0xFFFF0012

.eqv OUT_ADDRESS_HEX_A_KEYBOARD 0xFFFF0014

.eqv COUNTER 0xFFFF0013 # Time Counter

.eqv MASK_CAUSE_COUNTER 0x00000400 # Bit 10: Counter interrupt

.eqv MASK_CAUSE_KEYMATRIX 0x00000800 # Bit 11: Key matrix interrupt

.data

msg_keypress: .ascii "Someone has pressed a key: "

msg_counter: .ascii "Time interval "

msg_counter_endl: .ascii "\n"

~~~~~

MAIN Procedure

~~~~~

.text

li \$s7, 1

main:

#-----

Enable interrupts you expect

#-----

Enable the interrupt of Keyboard matrix 4x4 of Digital Lab Sim

li \$t1, IN_ADDRESS_HEX_A_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 number

#-----

Loop:

nop

nop

nop

sleep:

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

li \$a0,200 # 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
    li $t1, IN_ADRESS_HEX_A_KEYBOARD
    li $t2, OUT_ADRESS_HEX_A_KEYBOARD
inter_1:
    li $t3, 0x81 # check row 1 with key 0, 1, 2, 4
    sb $t3, 0($t1) # must reassign expected row
    jal inter
inter_2:
    li $t3, 0x82 # check row 2 with key 4, 5, 6, 7
    sb $t3, 0($t1) # must reassign expected row
    jal inter
inter_3:
    li $t3, 0x84 # check row 3 with key 8, 9, A, B
    sb $t3, 0($t1) # must reassign expected row
    jal inter
inter_4:
    li $t3, 0x88 # check row 4 with key C, D, E, F

```

```

        sb $t3, 0($t1) # must reassign expected row
        jal inter
after_inter_4:
        beq $a0, 0x0, prn_cod
        j next_pc
inter:
        lb $a0, 0($t2) # read scan code of key button
        bne $a0, 0x0, prn_cod
        jr $ra
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
        li $v0, 1
        add $a0, $s7, $zero
        syscall
        add $s7, $s7, 1
        li $v0, 4 # Processing Counter Interrupt
        la $a0, msg_counter_endl
        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:

The screenshot displays the Digital Lab Sim environment. The main window shows MIPS assembly code for `mips1.asm`. The code includes instructions for system calls, register manipulation, and interrupt handling. A floating window titled "Digital Lab Sim" is overlaid on the code, featuring a digital display showing "8.8." and a 4x4 keypad with digits 0-9 and letters a-f. Below the keypad is a "Tool Control" section with buttons for "Disconnect from MIPS", "Reset", "Help", and "Close". At the bottom of the interface, the "Mars Messages" and "Run I/O" tabs are visible, with the "Run I/O" tab showing a list of time intervals from 1 to 5. A "Clear" button is located to the left of this list.

```
106      syscall
107      li $v0, 1
108      add $a0, $s7, $zero
109      syscall
110      add $s7, $s7, 1
111      li $v0, 4 # Processing Counter Interrupt
112      la $a0, msg_counter_endl
113      syscall
114      j end_process
115 end_process:
116      mtc0 $zero, $13 # Must clear cause reg
117 en_int: #-----
118      # Re-enable interrupt
119      #-----
120      li $t1, COUNTER
121      sb $t1, 0($t1)
122      #-----
123      # Evaluate the return address of main routine
124      # epc <= epc + 4
125      #-----
126 next_pc:
127      mfc0 $at, $14 # $at <= Coproc0.$14 = Coproc0.epc
128      addi $at, $at, 4 # $at = $at + 4 (next instruction)
129      mtc0 $at, $14 # Coproc0.$14 = Coproc0.epc <= $at
130 return:
131      eret # Return from exception
132
```

Line: 114 Column: 12 ☒ Show Line Numbers

Mars Messages Run I/O

Time interval 1
Time interval 2
Time interval 3
Time interval 4
Time interval 5

Clear

The screenshot displays the Digital Lab Sim environment, which includes a MIPS assembly editor, a message log, and a simulated keypad interface.

MIPS Assembly Code (mips1.asm):

```

106      syscall
107      li $v0, 1
108      add $a0, $s7, $zero
109      syscall
110      add $s7, $s7, 1
111      li $v0, 4 # Processing Counter Interrupt
112      la $a0, msg_counter_endl
113      syscall
114      j end_process
115 end_process:
116      mtc0 $zero, $13 # Must clear cause reg
117 en_int: #-----
118 # Re-enable interrupt
119 #-----
120      li $t1, COUNTER
121      sb $t1, 0($t1)
122 #-----
123 # Evaluate the return address of main routine
124 # epc <= epc + 4
125 #-----
126 next_pc:
127      mfc0 $at, $14 # $at <= Coproc0.$14 = Coproc0.epc
128      addi $at, $at, 4 # $at = $at + 4 (next instruction)
129      mtc0 $at, $14 # Coproc0.$14 = Coproc0.epc <= $at
130 return:
131      eret # Return from exception
132

```

Mars Messages:

Time interval 4
Time interval 5
Time interval 6
Time interval 7
Time interval 8
Someone has pressed a key: 0x00000024
Time interval 9

Digital Lab Sim (Version 1.0):

The keypad simulation shows the following state:

0	1	2	3
4	5	6	7
8	9	a	b
c	d	e	f

The display shows "8.8.". The keypad has a green highlight on the "9" key.

Mars Messages:

Someone has pressed a key: 0xffffffff88
Time interval 21
Time interval 22
Time interval 23
Time interval 24

- *Giải thích:*

- Khi đang thực hiện vòng lặp mà có tín hiệu nhấn từ ma trận Lab Sim hoặc Khoảng thời gian lập tới giới hạn thì Chương trình sẽ thực hiện ngắt.

- Đầu tiên chương trình thực hiện so sánh và tìm ra nguyên nhân ngắt, nếu là vượt quá thời gian lặp thì sẽ in ra message còn nếu là tín hiệu nhấn từ Lab Sim thì sẽ in ra ký tự được nhấn (hệ cơ số 16).

Assignment 5:

Mã nguồn:

Assignment 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 0x0000034 # 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 \$t1 = 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

ShowKey:

```
sw $t0, 0($s0) # show key
```

```
nop
```

end_process:

next_pc:

```
mfc0 $at, $14 # $at <= Coproc0.$14 = Coproc0.epc
```

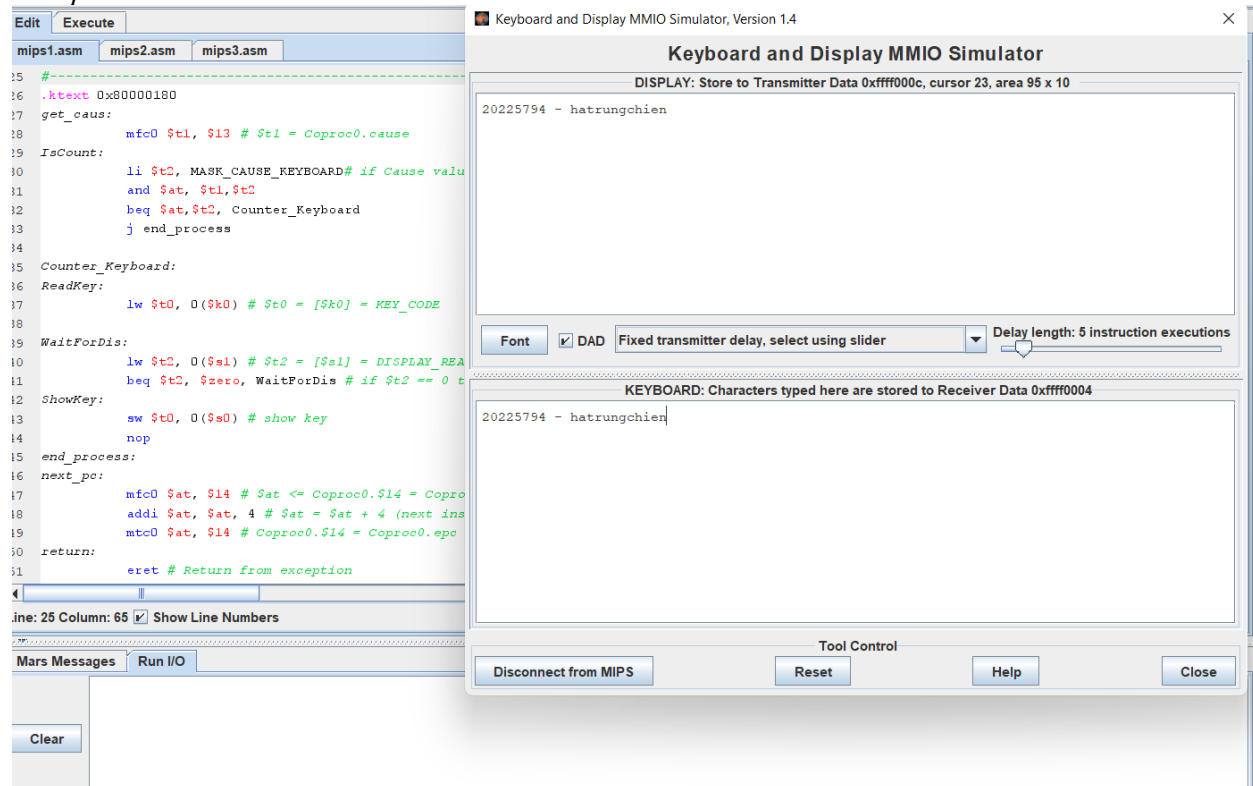
```
addi $at, $at, 4 # $at = $at + 4 (next instruction)
```

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

return:

```
eret # Return from exception
```

Kết quả:



- Giải thích:

Ngắt mềm được thực thi khi có một phím bất kỳ được nhấn.

- Dòng lệnh 17 -> 21 thực hiện điều này

- Khi mà có một phím được nhập vào từ KEYBOARD thì biến t1 sẽ bằng 1 (KEY_READY đã sẵn sàng)

- Mà câu lệnh dòng 21 “teqi \$t1, 1”: tức là khi biến t1 = 1 thì ngắt mềm sẽ được thực thi