

Báo cáo thực hành Kiến trúc máy tính – IT3280

Họ và tên: Thân Cát Ngọc Lan

MSSV: 20225646

Assignment 4:

*Code:

```
.eqv IN_ADRESS_HEXА_KEYBOARD 0xFFFF0012
.eqv OUT_ADRESS_HEXА_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: .asciiz "Someone has pressed a key is "
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) #set one bit of this byte to enable Counter interrupt
```

```

#-----

# Loop an print sequence numbers

#-----

Loop: nop

    nop

    nop

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

    li $a0,200 # sleep 200 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 $t3, 0x01
get_cod:li $t1, IN_ADRESS_HEXА_KEYBOARD
        ori $t4, $t3, 0x80 # check row 4 and re-enable bit 7
        sb $t4, 0($t1) # must reassign expected row
        li $t1, OUT_ADRESS_HEXА_KEYBOARD
        lb $a0, 0($t1)
        bne $a0, $0, prn_cod
        #addi $t3, $t3, 0xffff ff80
        sll $t3, $t3,1
        addi $t3, $t3, 0x80
        j get_cod

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)
```

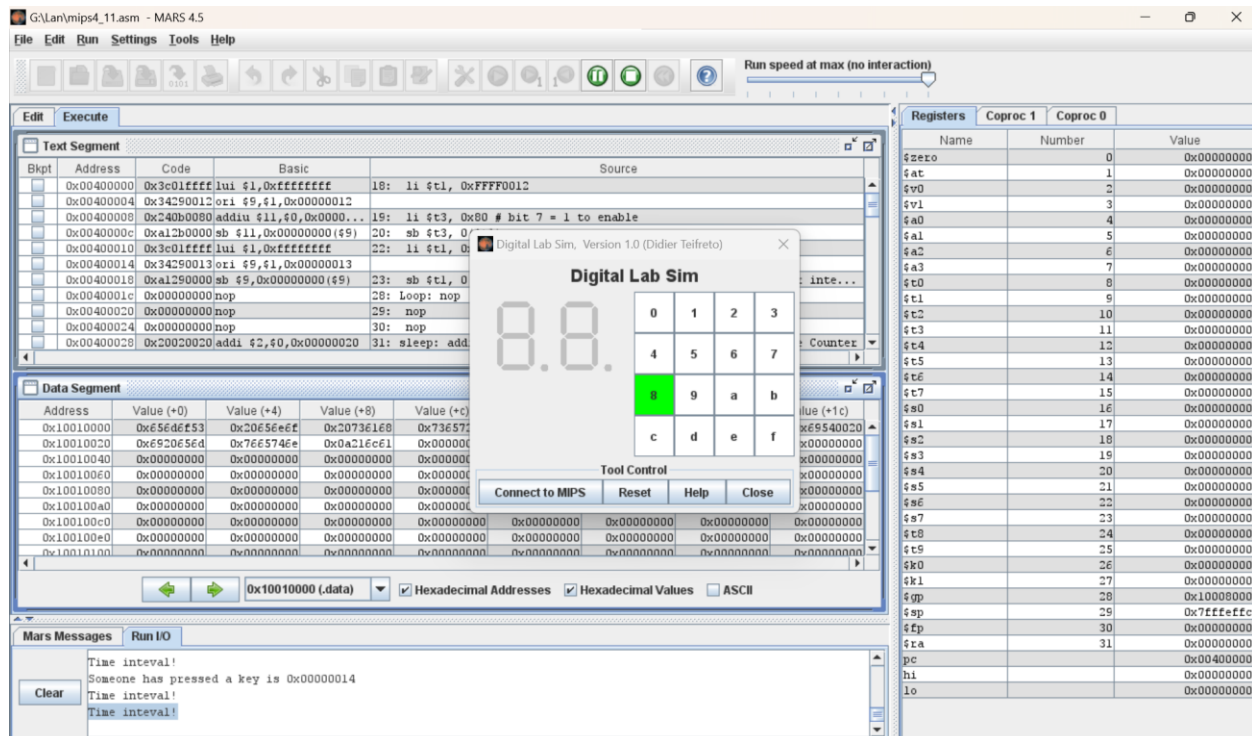
```
mtc0 $at, $14 # Coproc0.$14 = Coproc0.epc <= $at
```

```
return: eret # Return from exception
```

*Kết quả:

The screenshot shows the Mars MIPS simulator interface. The main window displays assembly code with columns for Address, Code, Basic, and Source. A digital display in the center shows the value 8.8. The right panel shows the Registers table with various registers and their values. The bottom panel shows the Mars Messages window with a log of events.

Name	Number	Value
\$zero	0	0x00000000
\$at	1	0x00000000
\$v0	2	0x00000000
\$v1	3	0x00000000
\$a0	4	0x00000000
\$a1	5	0x00000000
\$a2	6	0x00000000
\$a3	7	0x00000000
\$t0	8	0x00000000
\$t1	9	0x00000000
\$t2	10	0x00000000
\$t3	11	0x00000000
\$t4	12	0x00000000
\$t5	13	0x00000000
\$t6	14	0x00000000
\$t7	15	0x00000000
\$t8	16	0x00000000
\$t9	17	0x00000000
\$s0	18	0x00000000
\$s1	19	0x00000000
\$s2	20	0x00000000
\$s3	21	0x00000000
\$s4	22	0x00000000
\$s5	23	0x00000000
\$s6	24	0x00000000
\$s7	25	0x00000000
\$s8	26	0x00000000
\$s9	27	0x00000000
\$k0	28	0x00000000
\$k1	29	0x00000000
\$gp	30	0x00000000
\$sp	31	0x00000000
\$ra	32	0x00000000
\$pc	33	0x00000000
\$hi	34	0x00000000
\$lo	35	0x00000000



*Giải thích:

- chương trình trên xử lý các ngắt từ một ma trận bàn phím và một bộ đếm thời gian, in các thông báo tương ứng.

- Mã trên khởi tạo các ngắt cho bàn phím và bộ đếm.
- Nó nhập một vòng lặp nơi nó chờ đợi các ngắt.
- Kiểm tra nguyên nhân của ngắt và xử lý nó tương ứng (hoặc in một thông báo về một phím đã được nhấn hoặc một khoảng thời gian).
- Sau khi xử lý ngắt, kích hoạt lại các ngắt và quay lại vòng lặp chính.

Assignment 5:

*Code:

```
.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 $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
IsKeyboard:
    li $t2, MASK_CAUSE_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

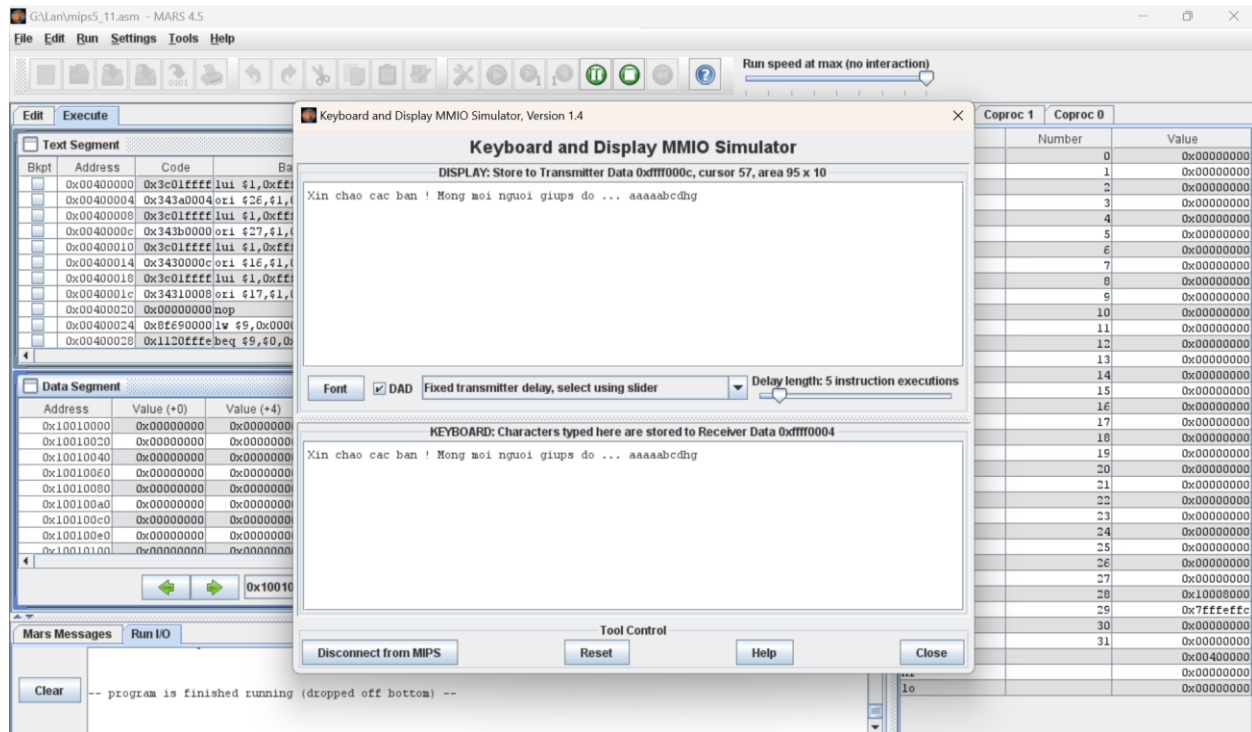
        addi $at, $at, 4 # $at = $at + 4 (next instruction)

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

return: eret # Return from exception

```

*Kết quả:



*Giải thích:

Chương trình trên đọc mã từ bàn phím và hiển thị chúng lên màn hình. Khi một phím được nhấn, chương trình tạo ra 1 ngắt để xử lí.

- Khi có phím được bấm thì Key_ready = 1, lệnh teqi nhảy vào chương trình ngắt.
- Kiểm tra nguyên nhân ngắt có phải là key board hay không. Nếu phải thì in ra màn hình.