

**BÁO CÁO THỰC HÀNH TUẦN 11**  
**Lương Thị Tâm – 20194663**

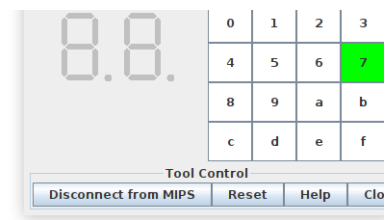
## Bài 1:

**\*Mã nguồn:**

```

22 .eqv IN_ADDRESS_HEX_A_KEYBOARD 0xFFFF0012
23 # receive row and column of the key pressed, 0 if not key pressed
24 # Eg. equal 0x11, means that key button 0 pressed.
25 # Eg. equal 0x28, means that key button D pressed.
26 .eqv OUT_ADDRESS_HEX_A_KEYBOARD 0xFFFF0014
27 .text
28 main: li $t1, IN_ADDRESS_HEX_A_KEYBOARD
29       li $t2, OUT_ADDRESS_HEX_A_KEYBOARD
30
31 polling: li $t3, 0x01 # check row 4 with key 0, 1, 2, 3
32         sb $t3, 0($t1) # must reassign expected row
33         lb $a0, 0($t2) # read scan code of key button
34         bne $a0, $0, print
35         li $t3, 0x02 # check row 4 with key 4, 5, 6, 7
36         sb $t3, 0($t1) # must reassign expected row
37         lb $a0, 0($t2) # read scan code of key button
38         bne $a0, $0, print
39         li $t3, 0x04 # check row 4 with key 8, 9, a, b
40         sb $t3, 0($t1) # must reassign expected row
41         lb $a0, 0($t2) # read scan code of key button
42         bne $a0, $0, print
43         li $t3, 0x08 # check row 4 with key c, d, e, f
44         sb $t3, 0($t1) # must reassign expected row
45         lb $a0, 0($t2) # read scan code of key button
46         bne $a0, $0, print
47     print: li $v0, 34 # print integer (hexa)
48           syscall
49     sleep: li $a0, 1000 # sleep 1000ms
50           li $v0, 32
51           syscall
52     back_to_polling: j polling # continue polling

```



**\*Kết quả chạy:**

[illegible]

## Bài 2:

### \*Mã nguồn:

```
1 .eqv IN_ADDRESS_HEXa_KEYBOARD 0xFFFF0012
2 .data
3 Message: .asciiz "Oh my god. Someone's presed a button.\n"
4 #-----
5 # MAIN Procedure
6 #-----
7 .text
8 main:
9 #-----
10 # Enable interrupts you expect
11 #-----
12 # Enable the interrupt of Keyboard matrix 4x4 of Digital Lab Sim
13 li $t1, IN_ADDRESS_HEXa_KEYBOARD
14 li $t3, 0x80 # bit 7 of = 1 to enable interrupt
15 sb $t3, 0($t1)
16 #-----
17 # No-end loop, main program, to demo the effective of interrupt
18 #-----
19 Loop: nop
20     nop
21     nop
22     b Loop # Wait for interrupt
23 end_main:
24 #-----
25 # GENERAL INTERRUPT SERVED ROUTINE for all interrupts
26 #-----
27 .ktext 0x80000180
28 #-----
29 # Processing
30 #-----
31 IntSR: addi $v0, $zero, 4 # show message
32     la $a0, Message
33     syscall
34 #-----
35 # Evaluate the return address of main routine
36 # epc <= epc + 4
37 #-----
38 next_pc:mfc0 $at, $14 # $at <= Coproc0.$14 = Coproc0.epc
39     addi $at, $at, 4 # $at = $at + 4 (next instruction)
40     mtc0 $at, $14 # Coproc0.$14 = Coproc0.epc <= $at
41 return: eret # Return from exception
```

### \*Kết quả chạy:

The screenshot shows the Digital Lab Sim environment. The main window displays the assembly code with addresses and comments. The 'Data Segment' window shows memory addresses and values. The 'Mars Messages' window shows the output of the program, which is 'Oh my god. Someone's presed a button.' repeated four times. The 'Digital Lab Sim' window shows a digital display with '8.8.' and a 4x4 keypad with a green button highlighted.

| Address    | Value (+0) | Value (+4) | Value (+14) | Value (+18) | Value (+1c) |
|------------|------------|------------|-------------|-------------|-------------|
| 0x10010000 | 0x6d20684f | 0x6f672079 | 0x65727020  | 0x20646573  | 0x75622061  |
| 0x10010020 | 0x6e6f7474 | 0x00000a2e | 0x00000000  | 0x00000000  | 0x00000000  |
| 0x10010040 | 0x00000000 | 0x00000000 | 0x00000000  | 0x00000000  | 0x00000000  |
| 0x10010060 | 0x00000000 | 0x00000000 | 0x00000000  | 0x00000000  | 0x00000000  |
| 0x10010080 | 0x00000000 | 0x00000000 | 0x00000000  | 0x00000000  | 0x00000000  |
| 0x100100a0 | 0x00000000 | 0x00000000 | 0x00000000  | 0x00000000  | 0x00000000  |
| 0x100100c0 | 0x00000000 | 0x00000000 | 0x00000000  | 0x00000000  | 0x00000000  |
| 0x100100e0 | 0x00000000 | 0x00000000 | 0x00000000  | 0x00000000  | 0x00000000  |
| 0x10010100 | 0x00000000 | 0x00000000 | 0x00000000  | 0x00000000  | 0x00000000  |
| 0x10010120 | 0x00000000 | 0x00000000 | 0x00000000  | 0x00000000  | 0x00000000  |
| 0x10010140 | 0x00000000 | 0x00000000 | 0x00000000  | 0x00000000  | 0x00000000  |
| 0x10010160 | 0x00000000 | 0x00000000 | 0x00000000  | 0x00000000  | 0x00000000  |
| 0x10010180 | 0x00000000 | 0x00000000 | 0x00000000  | 0x00000000  | 0x00000000  |

Mars Messages: Run I/O

Oh my god. Someone's presed a button.  
Oh my god. Someone's presed a button.  
Oh my god. Someone's presed a button.  
Oh my god. Someone's presed a button.  
-- program is finished running (dropped off bottom) --

Clear

### Bài 3:

#### \*Mã nguồn:

```
1 .eqv IN_ADDRESS_HEX_A_KEYBOARD 0xFFFF0012
2 .eqv OUT_ADDRESS_HEX_A_KEYBOARD 0xFFFF0014
3 .data
4 Message: .asciiz "Key scan code "
5 .text
6 main:
7     li $t1, IN_ADDRESS_HEX_A_KEYBOARD
8     li $t3, 0x80 # bit 7 = 1 to enable
9     sb $t3, 0($t1)
10    xor $s0, $s0, $s0 # count = $s0 = 0
11 Loop: addi $s0, $s0, 1 # count = count + 1
12 prn_seq: addi $v0, $zero, 1
13         add $a0, $s0, $zero # print auto sequence number
14         syscall
15 prn_eol: addi $v0, $zero, 11
16         li $a0, '\n' # print endofline
17         syscall
18 sleep: addi $v0, $zero, 32
19         li $a0, 300 # sleep 300 ms
20         syscall
21         nop # WARNING: nop is mandatory here.
22         b Loop # Loop
23 end_main:
24
25
26 .ktext 0x80000180
27 IntSR: addi $sp, $sp, 4 # Save $ra because we may change it later
28         sw $ra, 0($sp)
29         addi $sp, $sp, 4 # Save $at because we may change it later
30         sw $at, 0($sp)
31         addi $sp, $sp, 4 # Save $sp because we may change it later
32         sw $v0, 0($sp)
33         addi $sp, $sp, 4 # Save $a0 because we may change it later
34         sw $a0, 0($sp)
35         addi $sp, $sp, 4 # Save $t1 because we may change it later
36         sw $t1, 0($sp)
37         addi $sp, $sp, 4 # Save $t3 because we may change it later
38         sw $t3, 0($sp)
39
40 prn_msg: addi $v0, $zero, 4
```

```
41     la $a0, Message
42     syscall
43 get_cod: li $t2, IN_ADDRESS_HEX_A_KEYBOARD
44         li $t3, 0x81 # check row 4 and re-enable bit 7
45         sb $t3, 0($t2) # must reassign expected row
46         li $t1, OUT_ADDRESS_HEX_A_KEYBOARD
47         lb $a0, 0($t1)
48         bne $a0, $0, prn_cod
49         li $t3, 0x82 # check row 4 and re-enable bit 7
50         sb $t3, 0($t2) # must reassign expected row
51         lb $a0, 0($t1)
52         bne $a0, $0, prn_cod
53         li $t3, 0x84 # check row 4 and re-enable bit 7
54         sb $t3, 0($t2) # must reassign expected row
55         lb $a0, 0($t1)
56         bne $a0, $0, prn_cod
57         li $t3, 0x88 # check row 4 and re-enable bit 7
58         sb $t3, 0($t2) # must reassign expected row
59         lb $a0, 0($t1)
60 prn_cod: li $v0, 34
61         syscall
62         li $v0, 11
63         li $a0, '\n' # print endofline
64         syscall
65 next_pc: mfc0 $at, $14 # $at <= Coproc0.$14 = Coproc0.epc
66         addi $at, $at, 4 # $at = $at + 4 (next instruction)
67         mtc0 $at, $14 # Coproc0.$14 = Coproc0.epc <= $at
68 restore: lw $t3, 0($sp) # Restore the registers from stack
69         addi $sp, $sp, -4
70         lw $t1, 0($sp) # Restore the registers from stack
71         addi $sp, $sp, -4
72         lw $a0, 0($sp) # Restore the registers from stack
73         addi $sp, $sp, -4
74         lw $v0, 0($sp) # Restore the registers from stack
75         addi $sp, $sp, -4
76         lw $ra, 0($sp) # Restore the registers from stack
77         addi $sp, $sp, -4
78         lw $ra, 0($sp) # Restore the registers from stack
79         addi $sp, $sp, -4
80 return: eret # Return from exception
```

#### \*Kết quả chạy:

The screenshot displays the Mars MIPS simulator. The main assembly window shows the following code:

```

0x00400010: 0x02108026 xor $16,$16,$16      10: xor $s0,$s0,$s0 # count = $s0 = 0
0x00400014: 0x22100001 addi $16,$16,0x0000... 11: Loop: addi $s0,$s0,1 # count = count + 1
0x00400018: 0x20020001 addi $2,$0,0x00000001 12: prn_seq: addi $v0,$zero,1
0x0040001c: 0x02002020 add $4,$16,$0        13: add $a0,$s0,$zero # print auto sequence number
0x00400020: 0x0000000c syscall              14: syscall
0x00400024: 0x2002000b addi $2,$0,0x0000000b 15: prn_eol: addi $v0,$zero,11
0x00400028: 0x2404000a addiu $4,$0,0x0000000a 16: li $a0,'\n' # print endofline
0x0040002c: 0x0000000c syscall              17: syscall
0x00400030: 0x20020020 addi $2,$0,0x00000020 18: sleep: addi $v0,$zero,32
0x00400034: 0x2404012c addiu $4,$0,0x0000012c 19: li $a0,300 # sleep 300 ms
0x00400038: 0x0000000c syscall              20: syscall
0x0040003c: 0x00000000 nop                  21: nop # WARNING: nop is mandatory here
0x00400040: 0x0401fff4 bnez $0,0xfffffff4    22: b Loop # Loop

```

The 'Digital Lab Sim' window shows a digital display with '8.8.' and a 4x4 keypad with buttons 0-9, a, b, c, d, e, f. The 'Mars Messages' window shows a log with a 'Key scan code 0x00000012' message.

## Bài 4:

### \*Mã nguồn:

```

1 .eqv IN_ADDRESS_HEXa_KEYBOARD 0xffff0012
2 .eqv COUNTER 0xffff0013 # Time Counter
3 .eqv MASK_CAUSE_COUNTER 0x00000400 # Bit 10: Counter interrupt
4 .eqv MASK_CAUSE_KEYMATRIX 0x00000800 # Bit 11: Key matrix interrupt
5
6 .data
7 msg_keypress: .asciiz "Someone has pressed a key!\n"
8 msg_counter: .asciiz "Time interval!\n"
9 #~~~~~
10 # MAIN Procedure
11 #~~~~~
12 .text
13 main:
14 #-----
15 # Enable interrupts you expect
16 #-----
17 # Enable the interrupt of Keyboard matrix 4x4 of Digital Lab Sim
18 li $t1, IN_ADDRESS_HEXa_KEYBOARD
19 li $t3, 0x80 # bit 7 = 1 to enable
20 sb $t3, 0($t1)
21 # Enable the interrupt of TimeCounter of Digital Lab Sim
22 li $t1, COUNTER
23 sb $t1, 0($t1)
24
25 #-----
26 # Loop a print sequence numbers
27 #-----
28 Loop: nop
29      nop
30      nop
31 sleep: addi $v0,$zero,32 # BUG: must sleep to wait for Time Counter
32        li $a0,200 # sleep 300 ms
33        syscall
34        nop # WARNING: nop is mandatory here.
35        b Loop
36 end_main:
37 #~~~~~
38 # GENERAL INTERRUPT SERVED ROUTINE for all interrupts
39 #~~~~~
40 .ktext 0x80000180

```

```

41 IntSR: #-----
42 # Temporary disable interrupt
43 #-----
44 dis_int: li $t1, COUNTER # BUG: must disable with Time Counter
45         sb $zero, 0($t1)
46 # no need to disable keyboard matrix interrupt
47 #-----
48 # Processing
49 #-----
50 get_caus: mfc0 $t1, $13 # $t1 = Coproc0.cause
51 IsCount: li $t2, MASK_CAUSE_COUNTER # if Cause value confirm Counter..
52         and $at, $t1, $t2
53         beq $at, $t2, Counter_Intr
54 IsKeyMa: li $t2, MASK_CAUSE_KEYMATRIX # if Cause value confirm Key..
55         and $at, $t1, $t2
56         beq $at, $t2, Keymatrix_Intr
57 others: j end_process # other cases
58 Keymatrix_Intr: li $v0, 4 # Processing Key Matrix Interrupt
59               la $a0, msg_keypress
60               syscall
61               j end_process
62 Counter_Intr: li $v0, 4 # Processing Counter Interrupt
63               la $a0, msg_counter
64               syscall
65               j end_process
66 end_process:
67             mtc0 $zero, $13 # Must clear cause reg
68 en_int: #-----
69 # Re-enable interrupt
70 #-----
71         li $t1, COUNTER
72         sb $t1, 0($t1)
73 #-----
74 # Evaluate the return address of main routine
75 # epc <= epc + 4
76 #-----
77 next_pc: mfc0 $at, $14 # $at <= Coproc0.$14 = Coproc0.epc
78         addi $at, $at, 4 # $at = $at + 4 (next instruction)
79         mtc0 $at, $14 # Coproc0.$14 = Coproc0.epc <= $at
80 return: eret # Return from exception

```

## \*Kết quả chạy:

The screenshot displays the Digital Lab Sim environment, Version 1.0 (Didier ...). The main window is divided into several sections:

- Assembly Code Window:** Shows the MIPS assembly code being executed, with line numbers 41 through 80. The code includes comments in Vietnamese and assembly instructions like `li $t1, COUNTER`, `sb $zero, 0($t1)`, and `syscall`.
- Data Segment Window:** Displays memory addresses and their corresponding values. The values are mostly zeros, indicating a clean memory state.
- Simulation Window:** Shows a digital display with the value "8.8." and a numeric keypad. The keypad has buttons for digits 0-9, letters 'a' and 'b', and a green button labeled 'f'.
- Mars Messages Window:** Displays a series of "Time interval!" messages, indicating the simulation is running in real-time.

The simulation appears to be a simple keyboard input handler, where pressing a key triggers a message and updates the display.

## Bài 5:

### \*Mã nguồn:

```
1 .eqv KEY_CODE 0xFFFF0004 # ASCII code from keyboard, 1 byte
2 .eqv KEY_READY 0xFFFF0000 # =1 if has a new keycode ?
3 # Auto clear after lw
4 .eqv DISPLAY_CODE 0xFFFF000C # ASCII code to show, 1 byte
5 .eqv DISPLAY_READY 0xFFFF0008 # =1 if the display has already to do
6 # Auto clear after sw
7 .eqv MASK_CAUSE_KEYBOARD 0x00000034 # Keyboard Cause
8 .text
9     li $k0, KEY_CODE
10    li $k1, KEY_READY
11
12    li $s0, DISPLAY_CODE
13    li $s1, DISPLAY_READY
14 loop: nop
15 WaitForKey: lw $t1, 0($k1) # $t1 = [$k1] = KEY_READY
16             beq $t1, $zero, WaitForKey # if $t1 = 0 then Polling
17 MakeIntR:   teqi $t1, 1 # if $t1 = 1 then raise an Interrupt
18             j loop
19 #-----
20 # Interrupt subroutine
21 #-----
22 .ktext 0x80000180
23 get_caus: mfc0 $t1, $13 # $t1 = Coproc0.cause
24 IsCount: li $t2, MASK_CAUSE_KEYBOARD # if Cause value confirm Keyboard..
25          and $at, $t1, $t2
26          beq $at, $t2, Counter_Keyboard
27          j end_process
28 Counter_Keyboard:
29 ReadKey: lw $t0, 0($k0) # $t0 = [$k0] = KEY_CODE
30 WaitForDis: lw $t2, 0($s1) # $t2 = [$s1] = DISPLAY_READY
31            beq $t2, $zero, WaitForDis # if $t2 == 0 then Polling
32 Encrypt: addi $t0, $t0, 1 # change input key
33 ShowKey: sw $t0, 0($s0) # show key
34          nop
35 end_process:
36 next_pc: mfc0 $at, $14 # $at <= Coproc0.$14 = Coproc0.epc
37          addi $at, $at, 4 # $at = $at + 4 (next instruction)
38          mtc0 $at, $14 # Coproc0.$14 = Coproc0.epc <= $at
39 return: eret # Return from exception
```

### \*Kết quả chạy:

