

BÁO CÁO THỰC HÀNH TUẦN 11

Lương Thị Tâm – 20194663

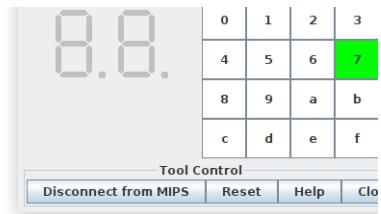
Bài 1:

*Mã nguồn:

```

22 .eqv IN_ADDRESS_HEXA_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_HEXA_KEYBOARD 0xFFFF0014
27 .text
28 main: li $t1, IN_ADDRESS_HEXA_KEYBOARD
29     li $t2, OUT_ADDRESS_HEXA_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:

Digital Lab Sim, Version 1.0 (Didier ...)

Digital Lab Sim

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

Tool Control
Disconnect from MIPS Reset Help Close

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)
0x10010000	0x00000000	0x00000000	0x00000000	0x000000
0x10010020	0x00000000	0x00000000	0x00000000	0x000000
0x10010040	0x00000000	0x00000000	0x00000000	0x000000
0x10010060	0x00000000	0x00000000	0x00000000	0x000000
0x10010080	0x00000000	0x00000000	0x00000000	0x000000
0x100100a0	0x00000000	0x00000000	0x00000000	0x000000
0x100100c0	0x00000000	0x00000000	0x00000000	0x000000
0x100100e0	0x00000000	0x00000000	0x00000000	0x000000
0x10010100	0x00000000	0x00000000	0x00000000	0x000000
0x10010120	0x00000000	0x00000000	0x00000000	0x000000
0x10010140	0x00000000	0x00000000	0x00000000	0x000000
0x10010160	0x00000000	0x00000000	0x00000000	0x000000

Mars Messages Run I/O

Tool Control
Disconnect from MIPS Reset Help Close

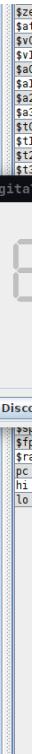
Bài 2:

*Mã nguồn:

```

1 .eqv IN_ADDRESS_HEXA_KEYBOARD 0xFFFF0012
2 .data
3 .ascii "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 software interface running the provided MIPS assembly code. The assembly window displays the source code with line numbers and addresses. The Registers window shows the state of registers like \$v0, \$a0, \$t1, etc. The Stack window shows the stack contents. The Data Segment window shows memory dump at address 0x00000000. The digital display shows the value 8.8. The Mars Messages window shows repeated output of "Oh my god. Someone's presed a button." and the message "- program is finished running (dropped off bottom) --".

Bkpt	Address	Code	Basic	Source
	0x00400000	0x3c01fffflui \$1,0xffffffff	13:	li \$t1, 0xFFFF0012
	0x00400004	0x34290012ori \$9,\$1,0x00000012	14:	li \$t3, 0x80 # bit 7 of = 1 to enable interrupt
	0x00400008	0x24000080addiu \$11,\$0,0x0000... 0x12b00000s \$11,0x00000000(\$9)	15:	sb \$t3, 0(\$t1)
	0x00400010	0x00000000nop	19:	Loop: nop
	0x00400014	0x00000000nop	20:	nop
	0x00400018	0x00000000nop	21:	nop
	0x0040001c	0x401ffffbgez \$0,0xffffffffc 0x20020004addi \$2,\$0,0x00000004	22:	b Loop # Wait for interrupt
	0x80000180	0x3c010001lui \$1,0x00001001	31:	IntSR: addi \$v0, \$zero, 4 # show message
	0x80000184	0x3c010001ori \$4,\$1,0x00000000	32:	la \$a0, Message
	0x80000188	0x34240000syscall	33:	syscall
	0x80000190	0x40017000mfc0 \$1,\$14	38:	next_pc:mfc0 \$at, \$14 # \$at <= Coproc0.\$14 = Coproc0.epc
	0x80000194	0x20210004addi \$1,\$1,0x00000004	39:	addi \$at, \$at, 4 # \$at = \$at + 4 (next instruction)
	0x80000198	0x40817000mtc0 \$1,\$14	40:	mtc0 \$at, \$14 # Coproc0.\$14 = Coproc0.epc <= \$at
	0x8000019c	0x42000018eret	41:	return: eret # Return from exception

Label	Address
mips2.asm	0x00400000
main	0x00400010
Loop	0x00400020
end_main	0x00100090
Message	0x10000000
IntSR	0x80000180
next_pc	0x80000190
return	0x8000019c

Digital Lab Sim, Version 1.0 (Didier ...)

Digital Lab Sim

Tool Control

Mars Messages Run I/O

Clear

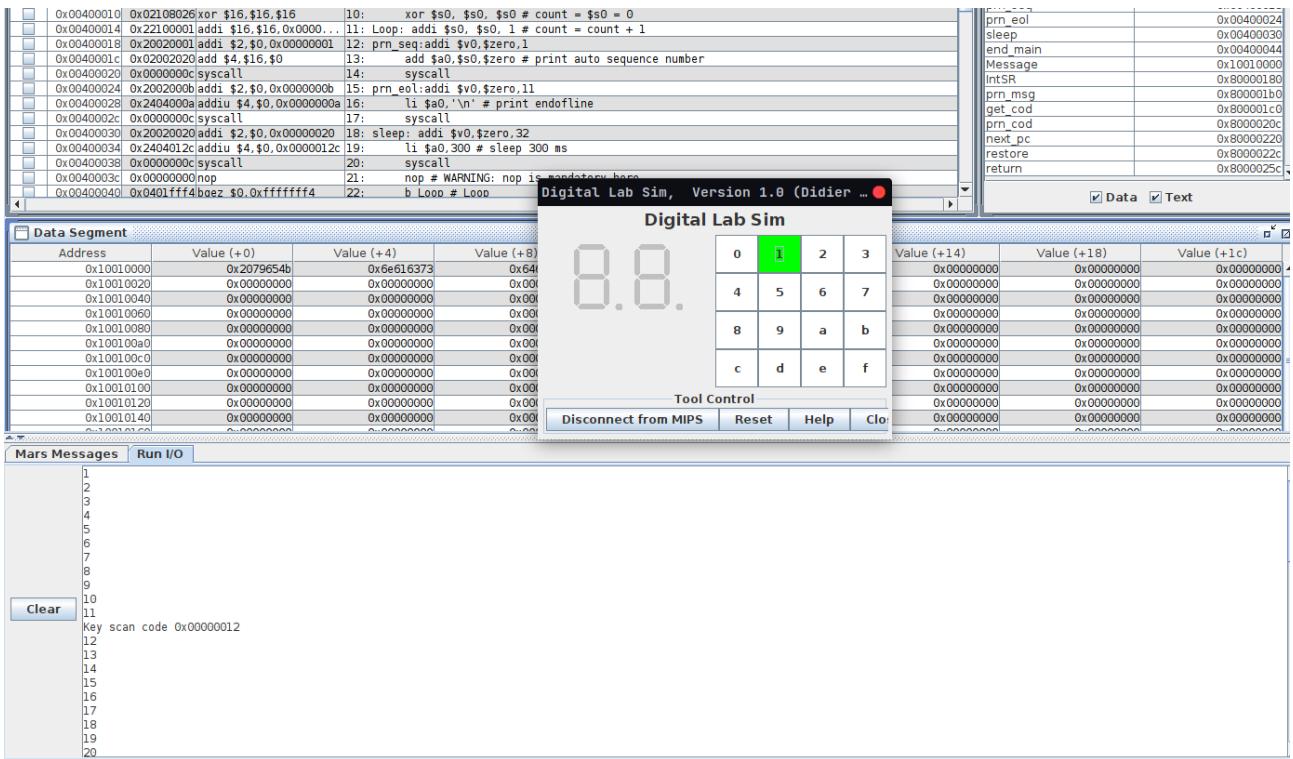
Bài 3:

*Mã nguồn:

```
1 .eqv IN_ADDRESS_HEXA_KEYBOARD 0xFFFF0012
2 .eqv OUT_ADDRESS_HEXA_KEYBOARD 0xFFFF0014
3 .data
4 Message: .asciiz "Key scan code "
5 .text
6 main:
7     li $t1, IN_ADDRESS_HEXA_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_HEXA_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_HEXA_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:



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 inteval!\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, Keypad_Intr
57 others: j end_process # other cases
58 Keypad_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 shows the Mars 4.5 debugger interface with several windows open:

- Registers Window:** Shows the CPU registers (Bkpt, Address, Code, Basic, Source) for the MIPS4 assembly code. The stack pointer (\$sp) is at 0x00000184.
- Labels Window:** Shows the labels and addresses of the assembly code, including main, Loop, sleep, end_main, msg_keypress, msg_counter, IntSR, dis_int, get_caus, IsCount, IsKeyMa, others, Keypad_Intr, Counter_Intr, and end_process.
- Data Segment Window:** Shows memory dump areas for Address, Value (+0), Value (+4), Value (+8), and Value (+18). It includes a digital display showing the value 8.8.
- Mars Messages Window:** Displays a log of messages, including frequent "Time interval!" entries and a "Someone has pressed a key!" message.
- Digital Lab Sim Window:** A simulation window titled "Digital Lab Sim, Version 1.0 (Did ...)" showing a digital display and a truth table for a 4-bit counter.

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 0x0000034 # 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: teui $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:

