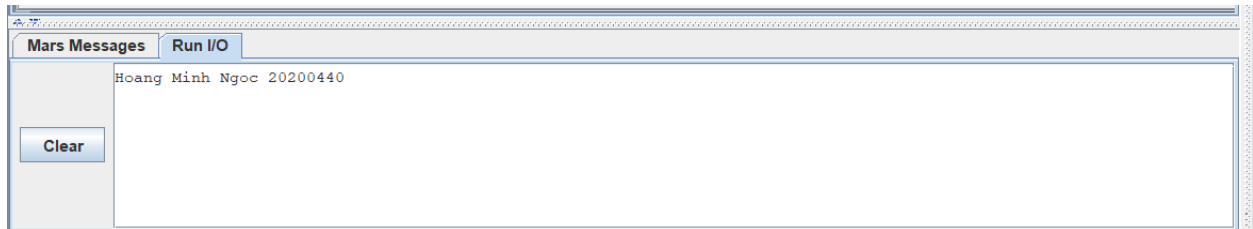


# BÁO CÁO TUẦN 13

## Bài 2 tuần 11

### Run



### Code

FA\BTKTMT\week13\Bai2week11 - MARS 4.5

File Edit Run Settings Tools Help

0101

1 `sb $t3, 0($t1)`  
 2 `#-----`  
 3 `.eqv IN_ADDRESS_HEXa_KEYBOARD 0xFFFF0012`  
 4 `.data`  
 5 `Message: .asciiz "Hoang Minh Ngoc 20200440\n"`  
 6 `#-----`  
 7 `# MAIN Procedure`  
 8 `#-----`  
 9 `.text`  
 10 `main:`  
 11 `#-----`  
 12 `# Enable interrupts you expect`  
 13 `#-----`  
 14 `# Enable the interrupt of Keyboard matrix 4x4 of Digital Lab Sim`  
 15 `li $t1, IN_ADDRESS_HEXa_KEYBOARD`  
 16 `li $t3, 0x80 # bit 7 of = 1 to enable interrupt`  
 17 `sb $t3, 0($t1)`  
 18 `#-----`  
 19 `# No-end loop, main program, to demo the effective of interrupt`  
 20 `#-----`  
 21 `Loop: nop`  
 22 `nop`  
 23 `nop`  
 24 `nop`  
 25 `b Loop # Wait for interrupt`  
 26 `end_main:`  
 27 `#-----`  
 28 `# GENERAL INTERRUPT SERVED ROUTINE for all interrupts`  
 29 `#-----`  
 30 `.ktext 0x80000180`  
 31 `#-----`  
 32 `# Processing`  
 33 `#-----`  
 34 `IntSR: addi $v0, $zero, 4 # show message`  
 35 `la $a0, Message`  
 36 `syscall`  
 37 `#-----`  
 38 `# Evaluate the return address of main routine`  
 39 `# epc <= epc + 4`  
 40 `#-----`  
 41 `next_pc:mfc0 $at, $14 # $at <= Coproc0.$14 = Coproc0.epc`  
 42 `addi $at, $at, 4 # $at = $at + 4 (next instruction)`  
 43 `mtc0 $at, $14 # Coproc0.$14 = Coproc0.epc <= $at`  
 44 `return: eret # Return from exception`  
 45

Line: 25 Column: 30 ☒ Show Line Numbers

Mars Messages Run I/O

Hoang Minh Ngoc 20200440

Clear

Coproc 0

Coproc 1

Registers

Name	N...	Value
\$8...	8	0x000...
\$1...	12	0x000...
\$1...	13	0x000...
\$1...	14	0x000...

3:33 CH  
01/07/2022

Dòng số 5 lưu chuỗi vào trong Message

Dòng 15-17 là kích hoạt theo dõi sự thay đổi trên các nút mặc định

Dòng 21-25 là vòng lặp vô hạn

Dòng 30 nhảy tới địa chỉ 0x80000180 và viết code ở đây

Dòng 34-36 in Message “Hoang Minh Ngoc 20200440”

Dòng 41-43 Lưu địa chỉ lệnh kế vào thanh ghi \$14

Dòng 44 gán nội dung thay ghi \$14 vào thanh ghi PC

### Code trong MIPS

```
sb $t3, 0($t1)
```

```
#-----
```

```
.eqv IN_ADDRESS_HEXKEYBOARD 0xFFFF0012
```

```
.data
```

```
Message: .asciiz "Hoang Minh Ngoc 20200440\n"
```

```
#~~~~~  
~~~~~
```

```
# MAIN Procedure
```

```
#~~~~~  
~~~~~
```

```
.text
```

```
main:
```

```
#-----
```

```
# Enable interrupts you expect
```

```
#-----
```

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

```
li $t1, IN_ADDRESS_HEXKEYBOARD
```

```
li $t3, 0x80 # bit 7 of = 1 to enable interrupt
```

```
sb $t3, 0($t1)
```

```
#-----
```

```
# No-end loop, main program, to demo the effective of interrupt
```

```
#-----
```

```
Loop: nop
```

```
nop
```

```
nop
```

```
nop
```

```
b Loop # Wait for interrupt
```

```
end_main:
```

```
#~~~~~
```

```
~~~~~
```

```
# GENERAL INTERRUPT SERVED ROUTINE for all interrupts
```

```
#~~~~~
```

```
~~~~~
```

```
.ktext 0x80000180
```

```
#-----
```

```
# Processing
```

```
#-----
```

```
IntSR: addi $v0, $zero, 4 # show message
```

```
la $a0, Message
```

```
syscall
```

```
#-----
```

```
# Evaluate the return address of main routine
```

```
# epc <= epc + 4
```

```
#-----
```

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

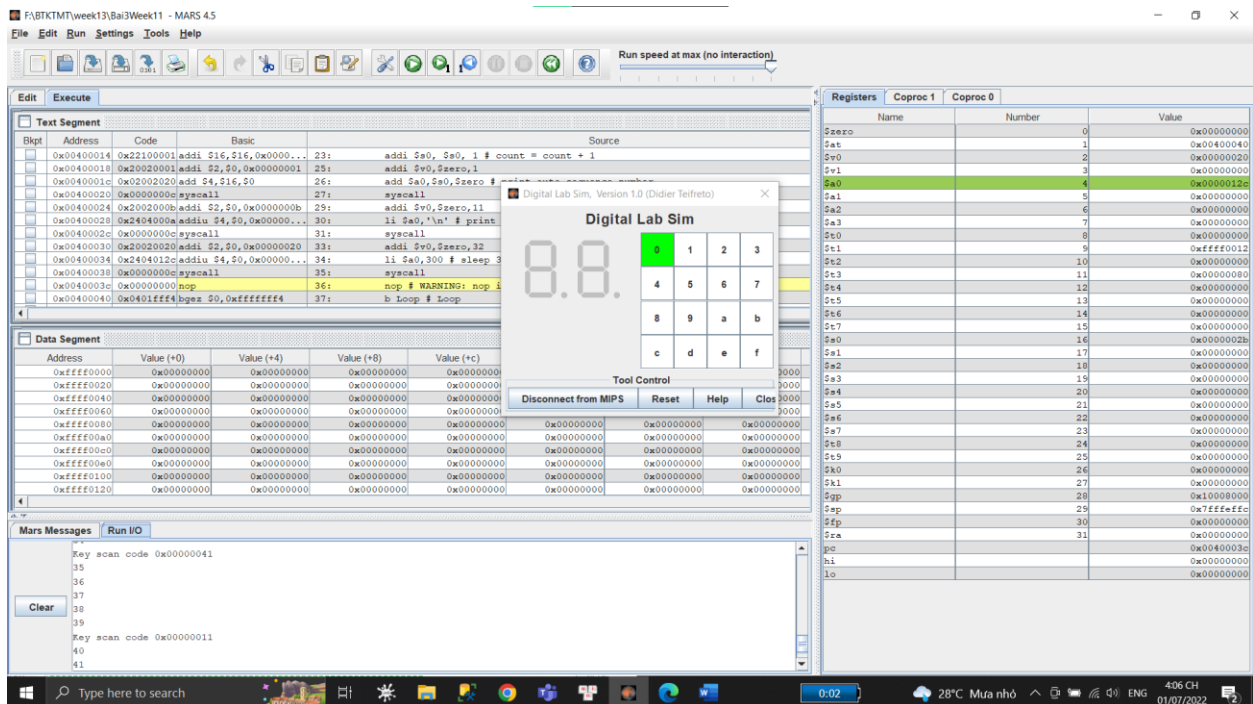
addi \$at, \$at, 4 # \$at = \$at + 4 (next instruction)

$$\text{mtc0 } \$at, \$14 \# \text{ Coproc0.} \$14 = \text{Coproc0.epc} \leq \$at$$

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

## Bài 3

## Run



```

Bai2week11  Bai3Week11*
1  .eqv IN_ADDRESS_HEX_KEYBOARD 0xFFFF0012
2  .eqv OUT_ADDRESS_HEX_KEYBOARD 0xFFFF0014
3  .data
4  Message: .asciiz "Key scan code "
5  #-----
6  # MAIN Procedure
7  #-----
8  .text
9  main:
10 #-----
11 # Enable interrupts you expect
12 #-----
13 # Enable the interrupt of Keyboard matrix 4x4 of Digital Lab Sim
14 li $t1, IN_ADDRESS_HEX_KEYBOARD
15 li $t3, 0x80 # bit 7 = 1 to enable
16 sb $t3, 0($t1)
17 #-----
18 # Loop an print sequence numbers
19 #-----
20 xor $s0, $s0, $s0 # count = $s0 = 0
21 Loop:
22 addi $s0, $s0, 1 # count = count + 1
23 prn_seq:
24 addi $v0, $zero, 1
25 add $a0, $s0, $zero # print auto sequence number
26 syscall
27 prn_eol:
28 addi $v0, $zero, 11
29 li $a0, '\n' # print endofline
30 syscall

```

Line: 15 Column: 38 ☒ Show Line Numbers

```

Bai2week11  Bai3Week11*
28 addi $v0, $zero, 11
29 li $a0, '\n' # print endofline
30 syscall
31 sleep:
32 addi $v0, $zero, 32
33 li $a0, 300 # sleep 300 ms
34 syscall
35 nop # WARNING: nop is mandatory here.
36 b Loop # Loop
37 end_main:
38 #-----
39 # GENERAL INTERRUPT SERVED ROUTINE for all interrupts
40 #-----
41 .ktext 0x80000180
42 #-----
43 # SAVE the current REG FILE to stack
44 #-----
45 IntSR:
46 addi $sp, $sp, 4 # Save $ra because we may change it later
47 sw $ra, 0($sp)
48 addi $sp, $sp, 4 # Save $at because we may change it later
49 sw $at, 0($sp)
50 addi $sp, $sp, 4 # Save $sp because we may change it later
51 sw $v0, 0($sp)
52 addi $sp, $sp, 4 # Save $a0 because we may change it later
53 sw $a0, 0($sp)
54 addi $sp, $sp, 4 # Save $t1 because we may change it later
55 sw $t1, 0($sp)
56 addi $sp, $sp, 4 # Save $t3 because we may change it later
57 sw $t3, 0($sp)

```

Line: 15 Column: 38 ☒ Show Line Numbers

```

Bai2week11  Bai3Week11*
52      addi $sp,$sp,4 # Save $a0 because we may change it later
53      sw $a0,0($sp)
54      addi $sp,$sp,4 # Save $t1 because we may change it later
55      sw $t1,0($sp)
56      addi $sp,$sp,4 # Save $t3 because we may change it later
57      sw $t3,0($sp)
58      #-----
59      # Processing
60      #-----
61      prn_msg:
62          addi $v0, $zero, 4
63          la $a0, Message
64          syscall
65      get_cod:
66          li $t1, IN_ADDRESS_HEXa_KEYBOARD
67          li $t3, 0x81 # check row 4 and re-enable bit 7
68          sb $t3, 0($t1) # must reassign expected row
69          li $t1, OUT_ADDRESS_HEXa_KEYBOARD
70          lb $a0, 0($t1)
71          bne $a0, 0x0, prn_cod
72
73          li $t1, IN_ADDRESS_HEXa_KEYBOARD
74          li $t3, 0x82 # check row 4 and re-enable bit 7
75          sb $t3, 0($t1) # must reassign expected row
76          li $t1, OUT_ADDRESS_HEXa_KEYBOARD
77          lb $a0, 0($t1)
78          bne $a0, 0x0, prn_cod
79
80          li $t1, IN_ADDRESS_HEXa_KEYBOARD
81          li $t3, 0x84 # check row 4 and re-enable bit 7

```

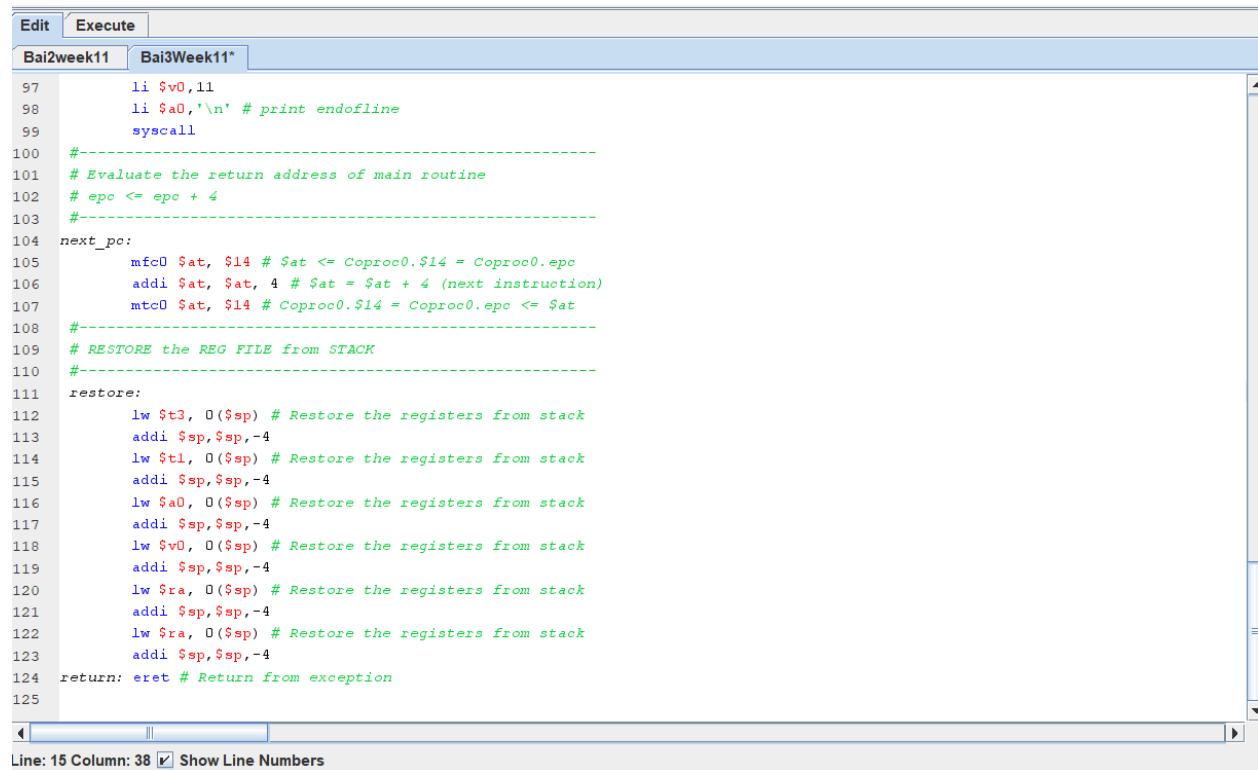
Line: 15 Column: 38 ☒ Show Line Numbers

```

Bai2week11  Bai3Week11*
70          lb $a0, 0($t1)
71          bne $a0, 0x0, prn_cod
72
73          li $t1, IN_ADDRESS_HEXa_KEYBOARD
74          li $t3, 0x82 # check row 4 and re-enable bit 7
75          sb $t3, 0($t1) # must reassign expected row
76          li $t1, OUT_ADDRESS_HEXa_KEYBOARD
77          lb $a0, 0($t1)
78          bne $a0, 0x0, prn_cod
79
80          li $t1, IN_ADDRESS_HEXa_KEYBOARD
81          li $t3, 0x84 # check row 4 and re-enable bit 7
82          sb $t3, 0($t1) # must reassign expected row
83          li $t1, OUT_ADDRESS_HEXa_KEYBOARD
84          lb $a0, 0($t1)
85          bne $a0, 0x0, prn_cod
86
87          li $t1, IN_ADDRESS_HEXa_KEYBOARD
88          li $t3, 0x88 # check row 4 and re-enable bit 7
89          sb $t3, 0($t1) # must reassign expected row
90          li $t1, OUT_ADDRESS_HEXa_KEYBOARD
91          lb $a0, 0($t1)
92          bne $a0, 0x0, prn_cod
93
94      prn_cod:
95          li $v0, 34
96          syscall
97          li $v0, 11
98          li $a0, '\n' # print endofline
99          syscall

```

Line: 15 Column: 38 ☒ Show Line Numbers



```
111  next_pc:
112      mfc0 $at, $14 # $at <= Coproc0.$14 = Coproc0.epc
113      addi $at, $at, 4 # $at = $at + 4 (next instruction)
114      mtc0 $at, $14 # Coproc0.$14 = Coproc0.epc <= $at
115
116      #-----
117      # RESTORE the REG FILE from STACK
118      #-----
119      restore:
120          lw $t3, 0($sp) # Restore the registers from stack
121          addi $sp, $sp, -4
122          lw $t1, 0($sp) # Restore the registers from stack
123          addi $sp, $sp, -4
124          lw $a0, 0($sp) # Restore the registers from stack
125          addi $sp, $sp, -4
126          lw $v0, 0($sp) # Restore the registers from stack
127          addi $sp, $sp, -4
128          lw $ra, 0($sp) # Restore the registers from stack
129          addi $sp, $sp, -4
130          lw $ra, 0($sp) # Restore the registers from stack
131          addi $sp, $sp, -4
132
133      return: eret # Return from exception
134
```

Line: 15 Column: 38 ☒ Show Line Numbers

## Giải thích

Dòng 14->16: bật chức năng theo dõi toàn bộ phím

20-26: tăng giá trị vòng lặp lên 1 và in ra ngoài màn hình

27-30 in ra khoảng cách giữa các giá trị

31-36 sleep giữa các vòng lặp

41 nhảy đến 0x80000180

45-57 : lưu \$ra,\$at,\$v0,\$a0,\$t1,\$t3 lưu vào stack

61-64 in ra message “ Key scan code “

66-71 kiểm tra xem phím có được ở dòng 1 hay không, nếu như có thì nhảy tới prn\_cod

73-78 kiểm tra xem phím có được ở dòng 2 hay không, nếu như có thì nhảy tới prn\_cod

80-85 kiểm tra xem phím có được ở dòng 3 hay không, nếu như có thì nhảy tới prn\_cod



87-92 kiểm tra xem phím có được ở dòng 1 hay không, nếu như có thì nhảy tới prn\_cod

94-99 in ra mã của phím vừa được nhấn và kí tự xuống dòng \n

104-107 lưu địa chỉ của lệnh kế tiếp vào thanh ghi \$14

111-123 lấy các giá trị đã lưu trong stack

124 gán nội dung của thanh ghi \$14 vào thanh ghi PC

### Code trong MIPS

```
.eqv IN_ADDRESS_HEXKEYBOARD 0xFFFF0012
```

```
.eqv OUT_ADDRESS_HEXKEYBOARD 0xFFFF0014
```

```
.data
```

```
Message: .asciiz "Key scan code "
```

```
#~~~~~  
~~~~~
```

```
# MAIN Procedure
```

```
#~~~~~  
~~~~~
```

```
.text
```

```
main:
```

```
#-----
```

```
# Enable interrupts you expect
```

```
#-----
```

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

```
li $t1, IN_ADDRESS_HEXKEYBOARD
```

```
li $t3, 0x80 # bit 7 = 1 to enable
```

```
sb $t3, 0($t1)
```

```
#-----
```

# Loop an print sequence numbers

#-----

xor \$s0, \$s0, \$s0 # count = \$s0 = 0

Loop:

addi \$s0, \$s0, 1 # count = count + 1

prn\_seq:

addi \$v0,\$zero,1

add \$a0,\$s0,\$zero # print auto sequence

number

syscall

prn\_eol:

addi \$v0,\$zero,11

li \$a0,'\n' # print endofline

syscall

sleep:

addi \$v0,\$zero,32

li \$a0,300 # sleep 300 ms

syscall

nop # WARNING: nop is mandatory here.

b Loop # Loop

end\_main:

#~~~~~  
~~~~~

# GENERAL INTERRUPT SERVED ROUTINE for all interrupts

#~~~~~  
~~~~~

.ktext 0x80000180

#-----

# SAVE the current REG FILE to stack

#-----

IntSR:

change it later                      addi \$sp,\$sp,4 # Save \$ra because we may

sw \$ra,0(\$sp)

change it later                      addi \$sp,\$sp,4 # Save \$at because we may

sw \$at,0(\$sp)

change it later                      addi \$sp,\$sp,4 # Save \$sp because we may

sw \$v0,0(\$sp)

change it later                      addi \$sp,\$sp,4 # Save \$a0 because we may

sw \$a0,0(\$sp)

change it later                      addi \$sp,\$sp,4 # Save \$t1 because we may

sw \$t1,0(\$sp)

change it later                      addi \$sp,\$sp,4 # Save \$t3 because we may

sw \$t3,0(\$sp)

#-----

# Processing

#-----

prn\_msg:

```
addi $v0, $zero, 4
```

```
la $a0, Message
```

```
syscall
```

get\_cod:

```
li $t1, IN_ADDRESS_HEX_KEYBOARD
```

```
li $t3, 0x81 # check row 4 and re-enable bit 7
```

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

```
li $t1,
```

OUT\_ADDRESS\_HEX\_KEYBOARD

```
lb $a0, 0($t1)
```

```
bne $a0, 0x0, prn_cod
```

```
li $t1, IN_ADDRESS_HEX_KEYBOARD
```

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

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

```
li $t1,
```

OUT\_ADDRESS\_HEX\_KEYBOARD

```
lb $a0, 0($t1)
```

```
bne $a0, 0x0, prn_cod
```

```
li $t1, IN_ADDRESS_HEX_KEYBOARD
```

```
li $t3, 0x84 # check row 4 and re-enable bit 7
```

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

```
li $t1,
```

OUT\_ADDRESS\_HEX\_KEYBOARD

```
lb $a0, 0($t1)
```

```
                                bne $a0, 0x0, prn_cod

                                li $t1, IN_ADDRESS_HEXA_KEYBOARD
                                li $t3, 0x88 # check row 4 and re-enable bit 7
                                sb $t3, 0($t1) # must reassign expected row
                                li $t1,
OUT_ADDRESS_HEXA_KEYBOARD
                                lb $a0, 0($t1)
                                bne $a0, 0x0, prn_cod

prn_cod:

                                li $v0, 34
                                syscall
                                li $v0, 11
                                li $a0, '\n' # print endofline
                                syscall

#-----
# 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
```



# Bài 4

Code

```

Bai4Week11
1  .eqv IN_ADRESS_HEXa_KEYBOARD 0xFFFF0012
2  .eqv OUT_ADRESS_HEXa_KEYBOARD 0xffff0014
3  .eqv COUNTER 0xFFFF0013 # Time Counter
4  .eqv MASK_CAUSE_COUNTER 0x00000400 # Bit 10: Counter interrupt
5  .eqv MASK_CAUSE_KEYMATRIX 0x00000800 # Bit 11: Key matrix interrupt
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_ADRESS_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 add $s5, $0, 0
23 li $t1, COUNTER
24 sb $t1, 0($t1)
25
26 #-----
27 # Loop a print sequence numbers
28 #-----
29 Loop: nop
30 nop
31 nop
32 sleep: addi $v0,$zero,32 # BUG: must sleep to wait for Time Counter
33 li $a0,200 # sleep 300 ms
34 syscall
35 nop # WARNING: nop is mandatory here.
36 b Loop
37 end_main:
38 #-----
39 # GENERAL INTERRUPT SERVED ROUTINE for all interrupts
40 #-----
41 .ktext 0x80000180
42 IntSR: #-----
43 # Temporary disable interrupt
44 #-----
45 dis_int: li $t1, COUNTER # BUG: must disable with Time Counter

```

Line: 5 Column: 68 ☒ Show Line Numbers



```

Bai4Week11
43  # Temporary disable interrupt
44  #-----
45  dis_int: li $t1, COUNTER # BUG: must disable with Time Counter
46  sb $zero, 0($t1)
47  # no need to disable keyboard matrix interrupt
48  #-----
49  # Processing
50  #-----
51  get_caus: mfc0 $t1, $13 # $t1 = Coproc0.cause
52  IsCount: li $t2, MASK_CAUSE_COUNTER # if Cause value confirm Counter..
53  and $at, $t1, $t2
54  beq $at, $t2, Counter_Intr
55  IsKeyMa: li $t2, MASK_CAUSE_KEYMATRIX # if Cause value confirm Key..
56  and $at, $t1, $t2
57  beq $at, $t2, Keymatrix_Intr
58  others: j end_process # other cases
59  Keymatrix_Intr: li $v0, 4 # Processing Key Matrix Interrupt
60  la $a0, msg_keypress
61  syscall
62
63  li $t1, IN_ADDRESS_HEX_A_KEYBOARD
64  li $t2, OUT_ADDRESS_HEX_A_KEYBOARD
65
66  li $t3, 0x81 #check row 1 and re-enable bit 7
67  jal check
68
69  li $t3, 0x82 #check row 2 and re-enable bit 7
70  jal check
71
72  li $t3, 0x84 #check row 3 and re-enable bit 7
73  jal check
74
75  li $t3, 0x88 #check row 4 and re-enable bit 7
76  check:
77  sb $t3, 0($t1) #must reassign expected row
78  lb $a0, 0($t2)
79  bne $a0, 0x0, prn_cod
80  jr $ra
81  prn_cod: li $v0, 34
82  syscall
83  li $v0, 11
84  li $a0, '\n' #print endofline
85  syscall
86  j end_process
87  Counter_Intr: li $v0, 4 # Processing Counter Interrupt

```

```

Bai4Week11
71
72     li $t3, 0x84 #check row 3 and re-enable bit 7
73     jal check
74
75     li $t3, 0x88 #check row 4 and re-enable bit 7
76 check:
77     sb $t3, 0($t1) #must reassign expected row
78     lb $a0, 0($t2)
79     bne $a0, 0x0, prn_cod
80     jr $ra
81 prn_cod: li $v0, 34
82     syscall
83     li $v0, 11
84     li $a0, '\n' #print endofline
85     syscall
86     j end_process
87 Counter_Intr: li $v0, 4 # Processing Counter Interrupt
88     la $a0, msg_counter
89     syscall
90
91     addi $s5, $s5, 1 #count = count + 1
92     addi $v0, $0, 1
93     add $a0, $s5, $0 #print auto sequence number
94     syscall
95
96     li $v0, 11
97     li $a0, '\n' #print endofline
98     syscall
99     j end_process
100 end_process:
101     mtc0 $zero, $13 # Must clear cause reg
102 en_int: #-----
103     # Re-enable interrupt
104     #-----
105     li $t1, COUNTER
106     sb $t1, 0($t1)
107     #-----
108     # Evaluate the return address of main routine
109     # epc <= epc + 4
110     #-----
111 next_pc: mfc0 $at, $14 # $at <= Coproc0.$14 = Coproc0.epc
112     addi $at, $at, 4 # $at = $at + 4 (next instruction)
113     mtc0 $at, $14 # Coproc0.$14 = Coproc0.epc <= $at
114 return: eret # Return from exception
115
Line: 5 Column: 68 ☒ Show Line Numbers

```

run

F:\BTKTMT\week13\Bai4Week11 - MARS 4.5

File Edit Run Settings Tools Help

Run speed at max (no interaction)

Registers Coproc 1 Coproc 0

Name	Number	Value
\$zero	0	0x00000000
\$at	1	0x0040002c
\$v0	2	0x00000020
\$v1	3	0x00000000
\$a0	4	0x000000c8
\$a1	5	0x00000000
\$a2	6	0x00000000
\$a3	7	0x00000000
\$t0	8	0x00000000
\$t1	9	0xffff0013
\$t2	10	0x00000400
\$t3	11	0x00000000

Text Segment

Bkpt	Address	Code	Basic	Source
	0x0040000c	0x12b0000	sb \$11,0x00000000(\$9)	20: sb \$t3, 0(\$t1)
	0x00400010	0x20150000	addi \$21,\$0,0x00000000	22: add \$s5, \$0, 0
	0x00400014	0x3c01ffff	lui \$1,0xffffffff	23: li \$t1, 0xFFFF0013
	0x00400018	0x34290013	ori \$9,\$1,0x00000013	
	0x0040001c	0x1290000	sb \$9,0x00000000(\$9)	24: sb \$t1, 0(\$t1)
	0x00400020	0x00000000	nop	29: Loop: nop
	0x00400024	0x00000000	nop	30: nop
	0x00400028	0x00000000	nop	31: nop
	0x0040002c	0x20020020	addi \$2,\$0,0x00000020	32: sleep: addi \$v0,\$zero,32 # BUG: mus...
	0x00400030	0x240400c8	addiu \$4,\$0,0x000000c8	33: li \$a0,200 # sleep 300 ms
	0x00400034	0x00000000	syscall	34: syscall
	0x00400038	0x00000000	nop	35: nop # WARNING: nop is mandato

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)
0x10010000	0x656d6f53	0x20656e6f	0x20736168	0x73657270	0x20646573	0x656b2061	0x000a2179
0x10010020	0x746e6920	0x6c617665	0x00000a21	0x00000000	0x00000000	0x00000000	0x00000000
0x10010040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010100	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010120	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

Digital Lab Sim, Version 1.0 (Didier Teifreto)

Digital Lab Sim

8.8.

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

Tool Control

Disconnect from MIPS Reset Help Close

Mars Messages Run I/O

0x00000011  
Time interval!  
11

Time interval!  
12  
Someone has pressed a key!  
0x00000012  
Time interval!  
13

Time interval!  
14  
Someone has pressed a key!  
0x00000012  
Time interval!  
15

Time interval!  
16  
Someone has pressed a key!  
0x00000011

Clear

\$fp	30	0x00000000
\$ra	31	0x800001d4
pc		0x00400038
hi		0x00000000
lo		0x00000000

## Giải thích code

18-20 : theo dõi toàn bộ phím nhập vào

22-24 : khởi tạo biến đếm và interrupt cho bộ đếm thời gian

29-36: vòng lặp vô hạn có slêp cho time counter

45-46: dừng bộ đếm thời gian

51-58: dùng mask kiểm tra xem nguyên nhân gây interrupt

59-86: kiểm tra xem vị trí phím được nhấn và in ra mã hex của nó

87-99: tăng biến đếm thêm 1 và in ra

100-101: xóa hết nguyên nhân đã gây ra interrupt

105-105: khởi động lại bộ đếm thời gian

111-113: lưu địa chỉ lệnh kế tiếp vào thanh ghi \$14

Dòng 114: gán nội dung của thanh ghi \$14 vào thanh ghi pc

## Code trong MIPS

```
.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!\n"
```

```
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\_HEXa\_KEYBOARD

li \$t3, 0x80 # bit 7 = 1 to enable

sb \$t3, 0(\$t1)

# Enable the interrupt of TimeCounter of Digital Lab Sim

add \$s5, \$0, 0

li \$t1, COUNTER

sb \$t1, 0(\$t1)

#-----

# Loop a print sequence numbers

#-----

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, Keypress\_Intr

others: j end\_process # other cases

Keypress\_Intr: li \$v0, 4 # Processing Key Matrix Interrupt

la \$a0, msg\_keypress

syscall

li \$t1, IN\_ADRESS\_HEXА\_KEYBOARD

li \$t2, OUT\_ADRESS\_HEXА\_KEYBOARD

li \$t3, 0x81 #check row 1 and re-enable bit 7

jal check

li \$t3, 0x82 #check row 2 and re-enable bit 7

jal check

li \$t3, 0x84 #check row 3 and re-enable bit 7

jal check

li \$t3, 0x88 #check row 4 and re-enable bit 7

check:

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

lb \$a0, 0(\$t2)

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
```

```
addi $s5, $s5, 1 #count = count + 1
addi $v0, $0, 1
add $a0, $s5, $0 #print auto sequence number
syscall
```

```
li $v0, 11
li $a0, '\n' #print endofline
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
```

## Bài 5

## Code

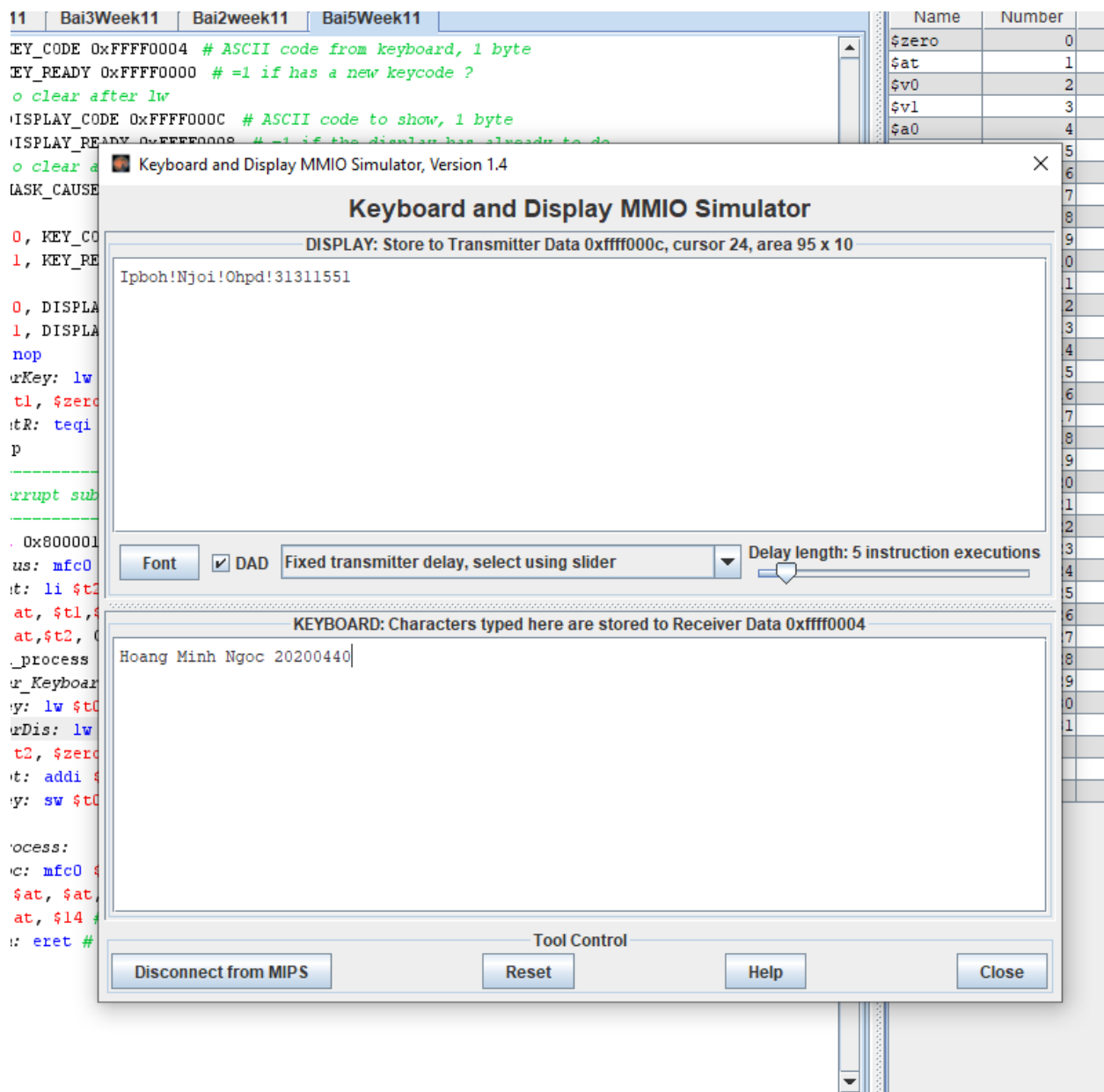
```

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: 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
40

```

Line: 30 Column: 58 ☒ Show Line Numbers

## Run



## Giải thích

14-18: vòng lặp vô hạn nếu như t1 mà bằng 1 thì sẽ ngắt thoát

23-27: kiểm tra nguyên nhân tạo interrupt có phải keyboard không

Dòng 29 lấy code kí tự được nhấn

30-31: vòng lặp cho tới khi nào mà bảng hiển thị đã sẵn sàng nhận

32: mã hóa kí tự

33: hiển thị ký tự mã hóa ra màn hình

36-38: lưu địa chỉ lệnh kế tiếp vào thanh ghi \$14

Dòng 29: gán nội dung thanh ghi \$14 vào thanh ghi pc

### Code trong MIPS

```
.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
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
```