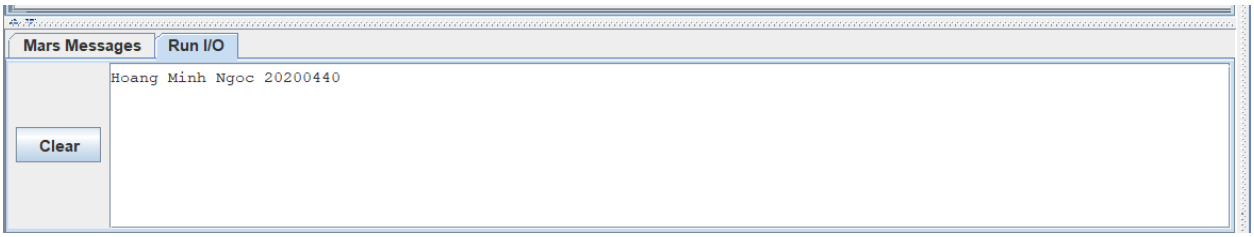
BÁO CÁO TUẦN 13

Bài 2 tuần 11

Run



Code

F:\BTKTMT\week13\Bai2week11 - MARS 4.5

File Edit Run Settings Tools Help

Edit Execute

Bai2week11

```
sb $t3, 0($t1)
 #-----------------------------------------------------
 .eqv IN_ADDRESS_HEXA_KEYBOARD 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_HEXA_KEYBOARD
 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)
 mtc0 $at, $14 # Coproc0.$14 = Coproc0.epc <= $at
return: eret # Return from exception
```

Line: 25 Column: 30 Show Line Numbers

Coproc 0 | Coproc 1 | Registers

Name	N...	Value
$8...	8	0x00...
$1...	12	0x00...
$1...	13	0x00...
$1...	14	0x00...

Mars Messages | Run I/O

```
Hoang Minh Ngoc 20200440
```

Clear

Type here to search ENG 3:33 CH 01/07/2022

Dòng số 5 lưu chuỗi vào trong Messsage

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_HEXA_KEYBOARD 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_HEXA_KEYBOARD
 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

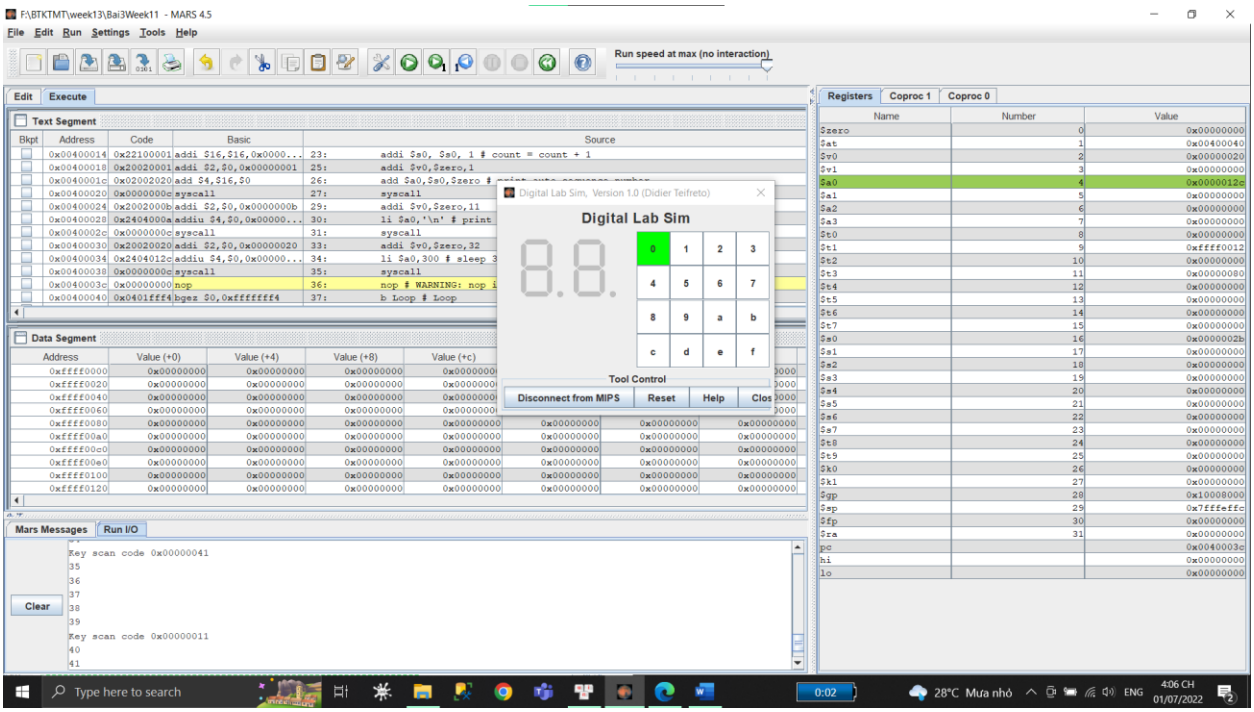
addi $at, $at, 4 # $at = $at + 4 (next instruction)

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

return: eret # Return from exception

Bài 3

Run



Bai2week11 | Bai3Week11*

```
.eqv IN_ADDRESS_HEXA_KEYBOARD 0xFFFF0012
.eqv OUT_ADDRESS_HEXA_KEYBOARD 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_HEXA_KEYBOARD
        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
```

Line: 15 Column: 38 Show Line Numbers

Bai2week11 | Bai3Week11*

```
        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:
        addi $sp,$sp,4 # Save $ra because we may change it later
        sw $ra,0($sp)
        addi $sp,$sp,4 # Save $at because we may change it later
        sw $at,0($sp)
        addi $sp,$sp,4 # Save $sp because we may change it later
        sw $v0,0($sp)
        addi $sp,$sp,4 # Save $a0 because we may change it later
        sw $a0,0($sp)
        addi $sp,$sp,4 # Save $t1 because we may change it later
        sw $t1,0($sp)
        addi $sp,$sp,4 # Save $t3 because we may change it later
        sw $t3,0($sp)
```

Line: 15 Column: 38 Show Line Numbers

Bai2week11 | Bai3Week11*

```
        addi $sp,$sp,4 # Save $a0 because we may change it later
        sw $a0,0($sp)
        addi $sp,$sp,4 # Save $t1 because we may change it later
        sw $t1,0($sp)
        addi $sp,$sp,4 # Save $t3 because we may change it later
        sw $t3,0($sp)
#-------------------------------------------------------
# Processing
#-------------------------------------------------------
prn_msg:
        addi $v0, $zero, 4
        la $a0, Message
        syscall
get_cod:
        li $t1, IN_ADDRESS_HEXA_KEYBOARD
        li $t3, 0x81 # 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

        li $t1, IN_ADDRESS_HEXA_KEYBOARD
        li $t3, 0x82 # 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

        li $t1, IN_ADDRESS_HEXA_KEYBOARD
        li $t3, 0x84 # check row 4 and re-enable bit 7
```

Line: 15 Column: 38 Show Line Numbers

Bai2week11 | Bai3Week11*

```
        lb $a0, 0($t1)
        bne $a0, 0x0, prn_cod

        li $t1, IN_ADDRESS_HEXA_KEYBOARD
        li $t3, 0x82 # 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

        li $t1, IN_ADDRESS_HEXA_KEYBOARD
        li $t3, 0x84 # 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

        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
```

Line: 15 Column: 38 Show Line Numbers

Edit Execute

Bai2week11 Bai3Week11*

```
97          li $v0,11
98          li $a0,'\n' # print endofline
99          syscall
100   #-----------------------------------------------------
101   # Evaluate the return address of main routine
102   # epc <= epc + 4
103   #-----------------------------------------------------
104 next_pc:
105         mfc0 $at, $14 # $at <= Coproc0.$14 = Coproc0.epc
106         addi $at, $at, 4 # $at = $at + 4 (next instruction)
107         mtc0 $at, $14 # Coproc0.$14 = Coproc0.epc <= $at
108   #-----------------------------------------------------
109   # RESTORE the REG FILE from STACK
110   #-----------------------------------------------------
111   restore:
112         lw $t3, 0($sp) # Restore the registers from stack
113         addi $sp,$sp,-4
114         lw $t1, 0($sp) # Restore the registers from stack
115         addi $sp,$sp,-4
116         lw $a0, 0($sp) # Restore the registers from stack
117         addi $sp,$sp,-4
118         lw $v0, 0($sp) # Restore the registers from stack
119         addi $sp,$sp,-4
120         lw $ra, 0($sp) # Restore the registers from stack
121         addi $sp,$sp,-4
122         lw $ra, 0($sp) # Restore the registers from stack
123         addi $sp,$sp,-4
124 return: eret # Return from exception
125
```

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_HEXA_KEYBOARD 0xFFFF0012

.eqv OUT_ADDRESS_HEXA_KEYBOARD 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_HEXA_KEYBOARD

                                        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:
                                        addi $sp,$sp,4 # Save $ra because we may
change it later
                                        sw $ra,0($sp)
                                        addi $sp,$sp,4 # Save $at because we may
change it later
                                        sw $at,0($sp)
                                        addi $sp,$sp,4 # Save $sp because we may
change it later
                                        sw $v0,0($sp)
                                        addi $sp,$sp,4 # Save $a0 because we may
change it later
                                        sw $a0,0($sp)
                                        addi $sp,$sp,4 # Save $t1 because we may
change it later
                                        sw $t1,0($sp)
                                        addi $sp,$sp,4 # Save $t3 because we may
change it later
                                        sw $t3,0($sp)
 #-----------------------------------------------------
 # Processing
 #-----------------------------------------------------
prn_msg:
```

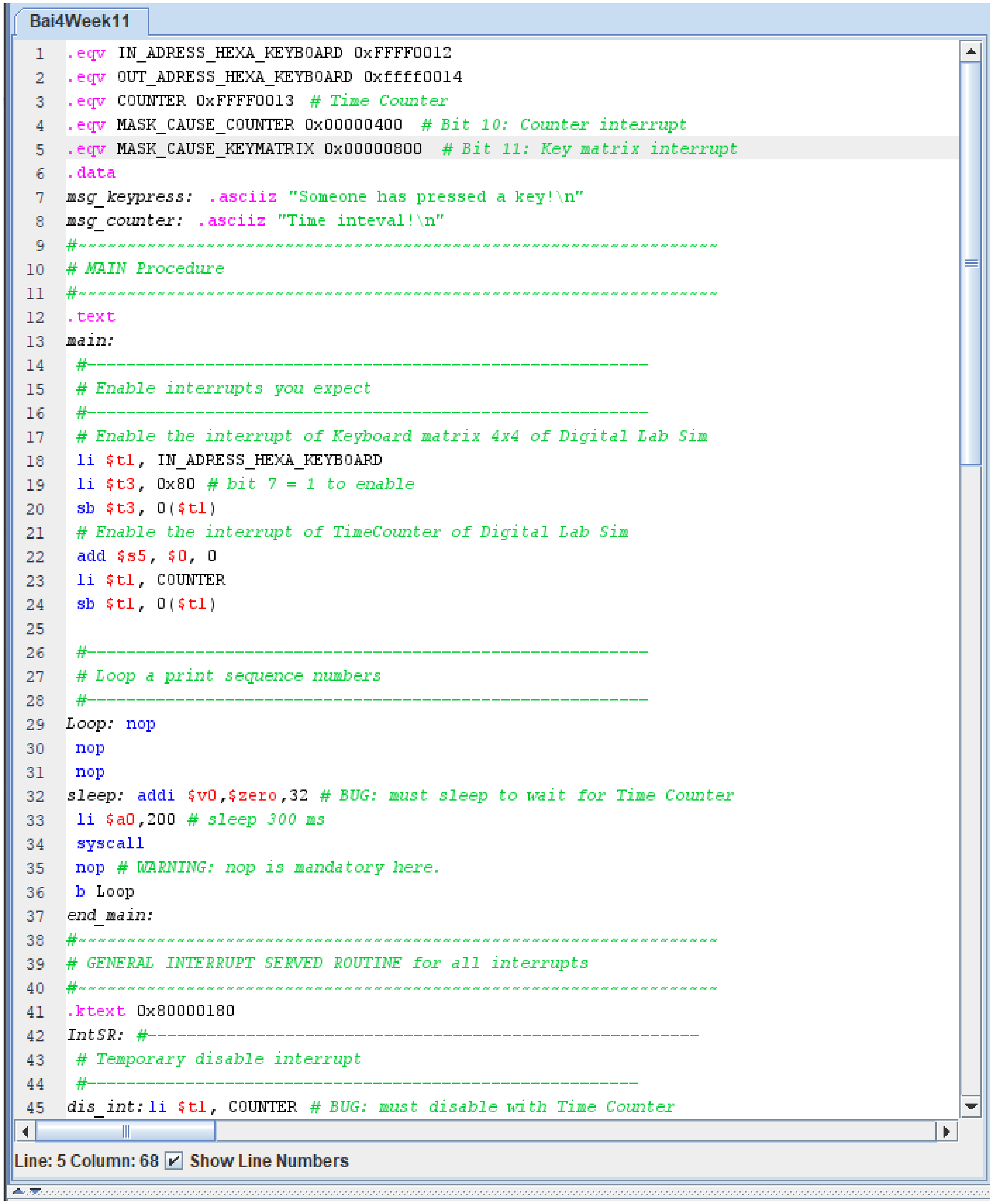
```
                    addi $v0, $zero, 4
                    la $a0, Message
                    syscall
get_cod:
                    li $t1, IN_ADDRESS_HEXA_KEYBOARD
                    li $t3, 0x81 # check row 4 and re-enable bit 7
                    sb $t3, 0($t1) # must reassign expected row
                    li $t1,
OUT_ADDRESS_HEXA_KEYBOARD
                    lb $a0, 0($t1)
                    bne $a0, 0x0, prn_cod

                    li $t1, IN_ADDRESS_HEXA_KEYBOARD
                    li $t3, 0x82 # 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

                    li $t1, IN_ADDRESS_HEXA_KEYBOARD
                    li $t3, 0x84 # 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

                    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
```

```
#-------------------------------------------------------
# RESTORE the REG FILE from STACK
#-------------------------------------------------------
restore:
                    lw $t3, 0($sp) # Restore the registers from stack
                    addi $sp,$sp,-4
                    lw $t1, 0($sp) # Restore the registers from stack
                    addi $sp,$sp,-4
                    lw $a0, 0($sp) # Restore the registers from stack
                    addi $sp,$sp,-4
                    lw $v0, 0($sp) # Restore the registers from stack
                    addi $sp,$sp,-4
                    lw $ra, 0($sp) # Restore the registers from stack
                    addi $sp,$sp,-4
                    lw $ra, 0($sp) # Restore the registers from stack
                    addi $sp,$sp,-4
return: eret # Return from exception
```

Bài 4

Code



```
.eqv IN_ADRESS_HEXA_KEYBOARD 0xFFFF0012
.eqv OUT_ADRESS_HEXA_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
```

Bai4Week11

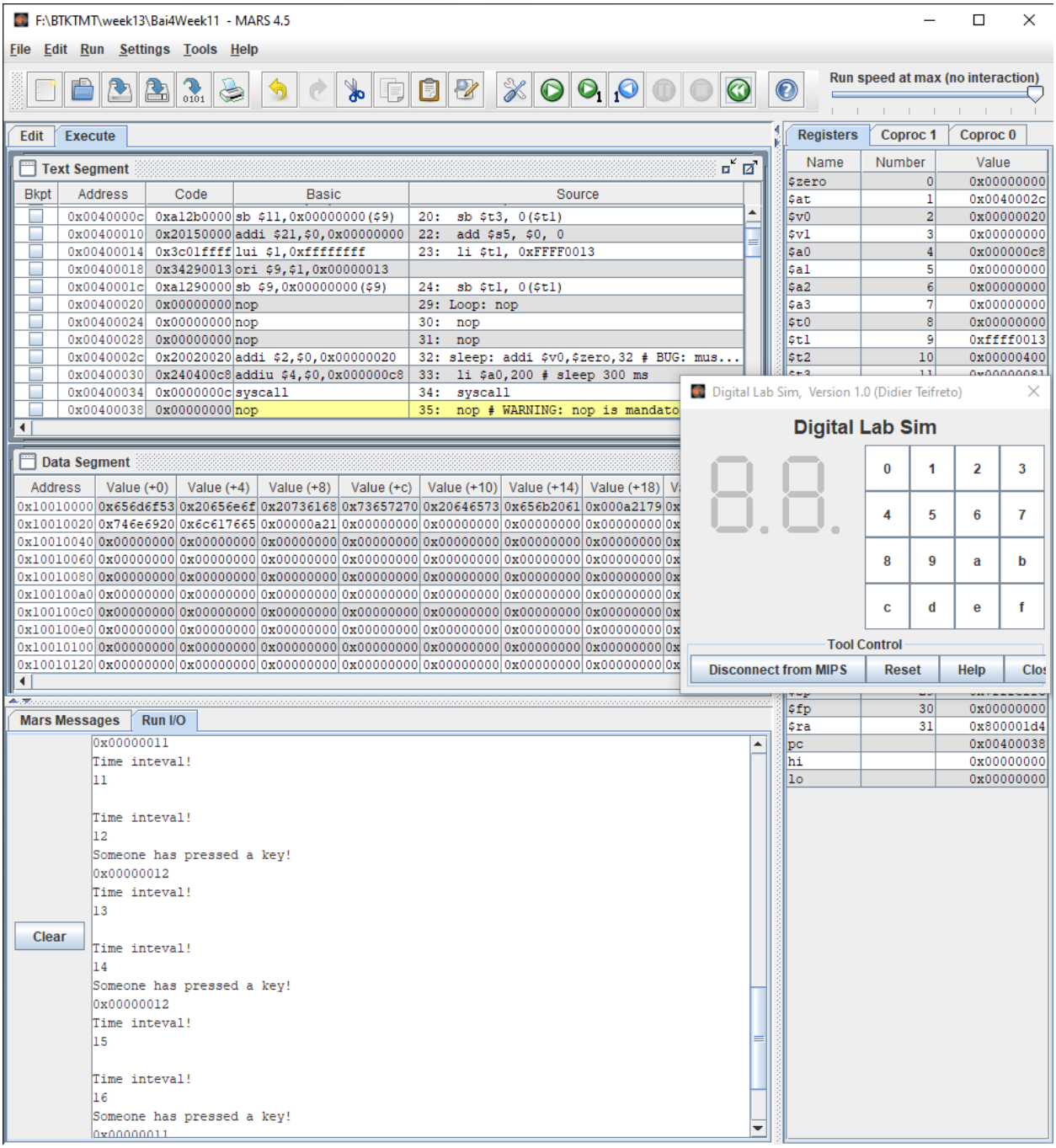
```
 # Temporary disable interrupt
 #-----------------------------------------------------
dis_int: li $t1, COUNTER # BUG: must disable with Time Counter
 sb $zero, 0($t1)
 # no need to disable keyboard matrix interrupt
 #-----------------------------------------------------
 # Processing
 #-----------------------------------------------------
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_HEXA_KEYBOARD
 li $t2, OUT_ADRESS_HEXA_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
```

Bai4Week11

```
 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
```

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)
Edit Execute
Text Segment
Bkpt Address Code Basic Source
0x0040000c 0xa12b0000 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 0xa1290000 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 0x0000000c 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
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
$fp 30 0x00000000
$ra 31 0x800001d4
pc 0x00400038
hi 0x00000000
lo 0x00000000
Digital Lab Sim, Version 1.0 (Didier Teifreto)
Digital Lab Sim
0 1 2 3
4 5 6 7
8 9 a b
c d e f
Tool Control
Disconnect from MIPS Reset Help
Mars Messages Run I/O
0x00000011
Time inteval!
11
Time inteval!
12
Someone has pressed a key!
0x00000012
Time inteval!
13
Time inteval!
14
Someone has pressed a key!
0x00000012
Time inteval!
15
Time inteval!
16
Someone has pressed a key!
Clear

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_HEXA_KEYBOARD 0xFFFF0012
.eqv OUT_ADRESS_HEXA_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, 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_HEXA_KEYBOARD
 li $t2, OUT_ADRESS_HEXA_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

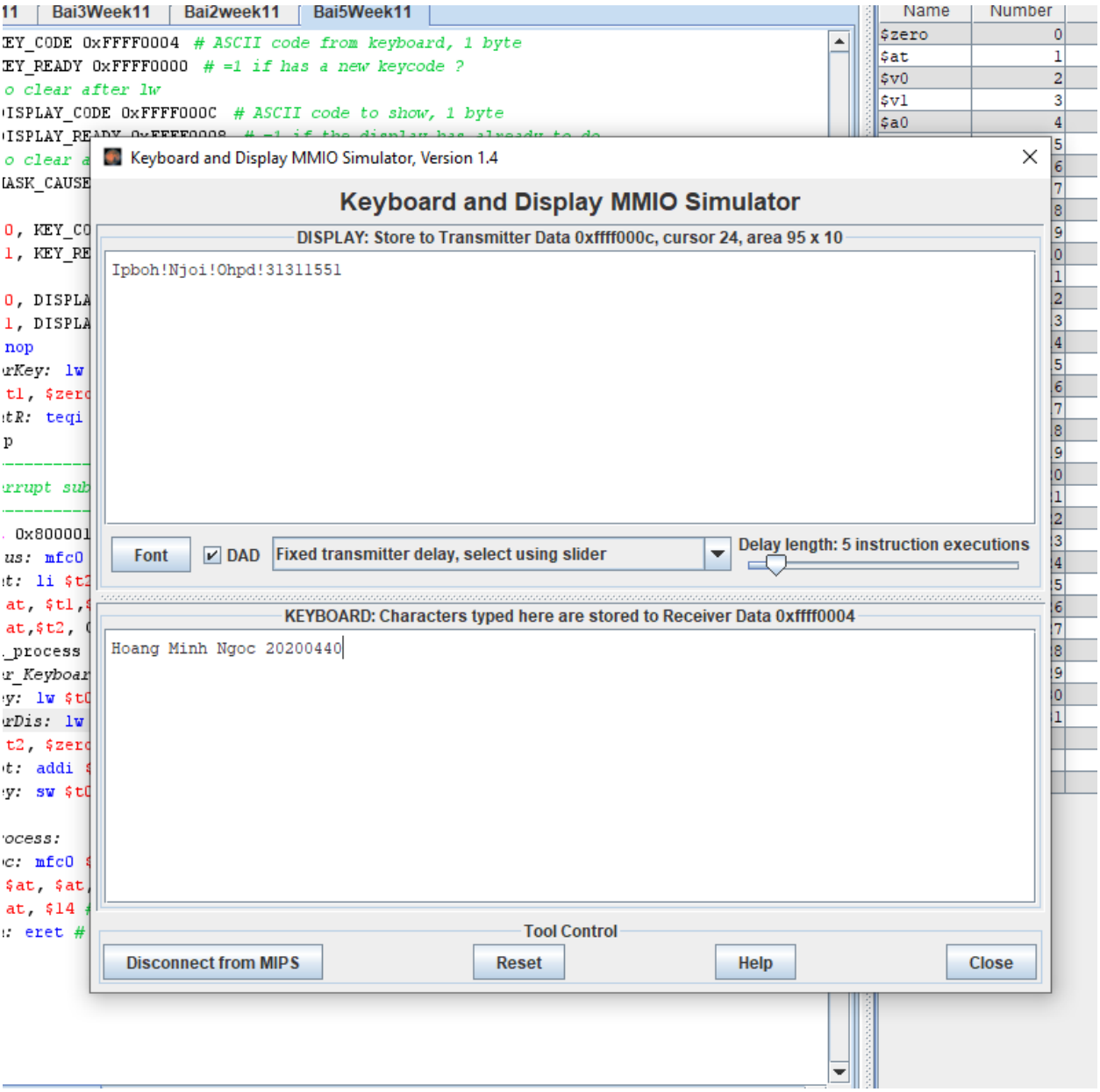
Code

Bai4Week11 | Bai3Week11 | Bai2week11 | Bai5Week11

```
.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
```

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
```