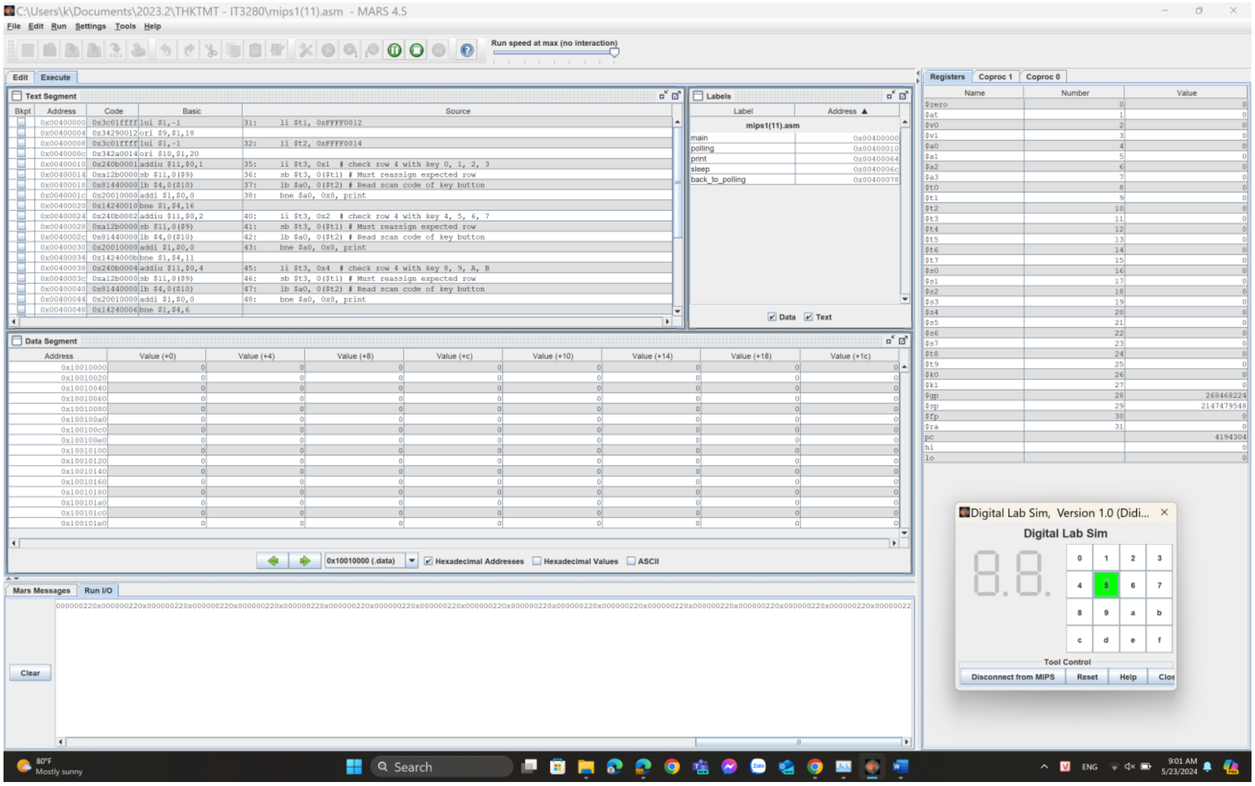
Laboratory 11

Họ tên: Vũ Quốc Bảo
MSSV: 20225694

Assignment 1:

```
#-----------------------------------------------------
#                col 0x1 col 0x2 col 0x4 col 0x8
#
# row 0x1       0       1       2       3
#               0x11    0x21    0x41    0x81
#
# row 0x2       4       5       6       7
#               0x12    0x22    0x42    0x82
#
# row 0x4       8       9       a       b
#               0x14    0x24    0x44    0x84
#
# row 0x8       c       d       e       f
#               0x18    0x28    0x48    0x88
#
#-----------------------------------------------------
# command row number of hexadecimal keyboard (bit 0 to 3)
# Eg. assign 0x1, to get key button 0,1,2,3
# assign 0x2, to get key button 4,5,6,7
# NOTE must reassign value for this address before reading,
# eventhough you only want to scan 1 row
.eqv IN_ADDRESS_HEXA_KEYBOARD 0xFFFF0012

# receive row and column of the key pressed, 0 if not key pressed
# Eg. equal 0x11, means that key button 0 pressed.
# Eg. equal 0x28, means that key button D pressed.
.eqv OUT_ADDRESS_HEXA_KEYBOARD 0xFFFF0014

.text
main:
    li $t1, IN_ADDRESS_HEXA_KEYBOARD
    li $t2, OUT_ADDRESS_HEXA_KEYBOARD

polling:
    li $t3, 0x1  # check row 4 with key 0, 1, 2, 3
    sb $t3, 0($t1) # Must reassign expected row
    lb $a0, 0($t2) # Read scan code of key button
    bne $a0, 0x0, print
```

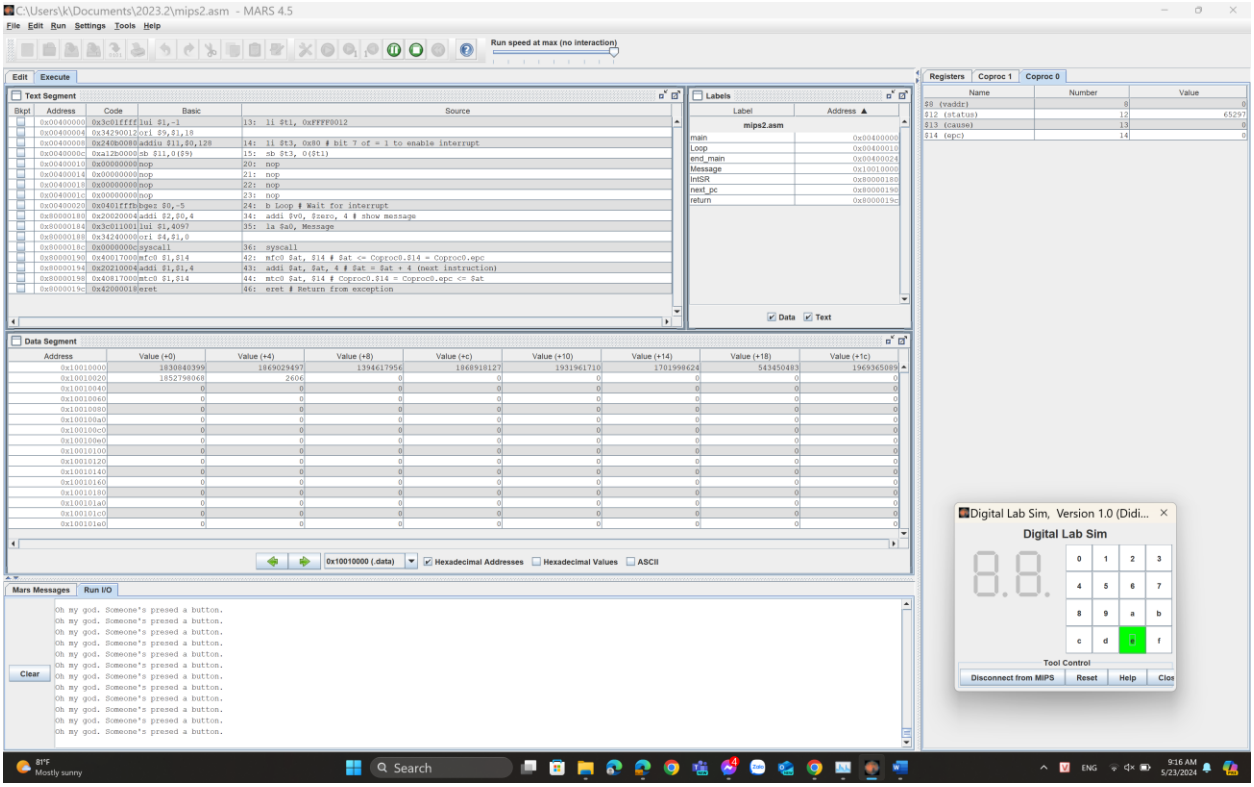
```
    li $t3, 0x2  # check row 4 with key 4, 5, 6, 7
    sb $t3, 0($t1) # Must reassign expected row
    lb $a0, 0($t2) # Read scan code of key button
    bne $a0, 0x0, print

    li $t3, 0x4  # check row 4 with key 8, 9, A, B
    sb $t3, 0($t1) # Must reassign expected row
    lb $a0, 0($t2) # Read scan code of key button
    bne $a0, 0x0, print

    li $t3, 0x8  # check row 4 with key C, D, E, F
    sb $t3, 0($t1) # Must reassign expected row
    lb $a0, 0($t2) # Read scan code of key button
    bne $a0, 0x0, print

    j polling # Continue polling

print:
    li $v0, 34 # Print integer (hexadecimal)
    syscall

sleep:
    li $a0, 100 # Sleep for 100ms
    li $v0, 32
    syscall

back_to_polling:
    j polling # Continue polling
```



<u>Assignment 2:</u>

```
.eqv IN_ADRESS_HEXA_KEYBOARD 0xFFFF0012
.data
Message: .asciiz "Oh my god. Someone's presed a button.\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 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
```

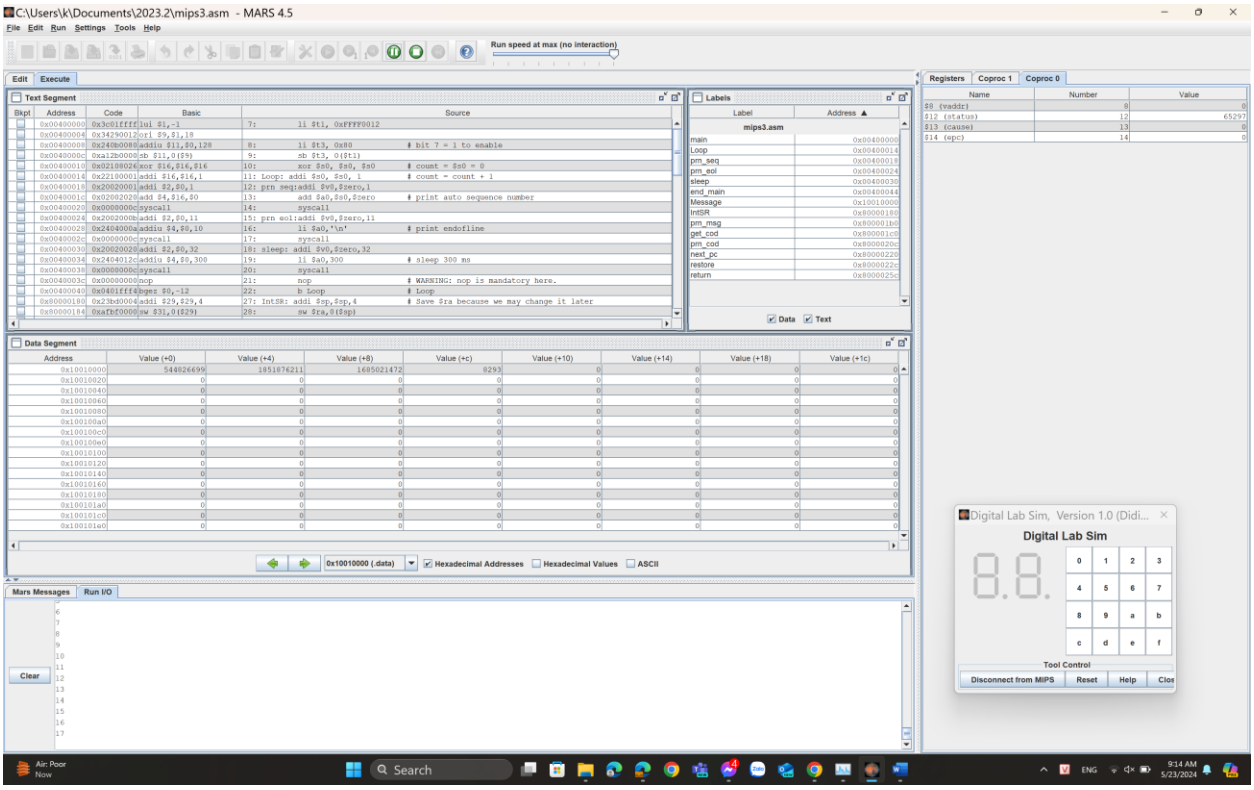


Assignment 3:

```
.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 $t2, IN_ADDRESS_HEXA_KEYBOARD
        li $t3, 0x81            # check row 4 and re-enable bit 7
        sb $t3, 0($t2)          # must reassign expected row
        li $t1, OUT_ADDRESS_HEXA_KEYBOARD
        lb $a0, 0($t1)
        bne     $a0, $0, prn_cod
        li $t3, 0x82            # check row 4 and re-enable bit 7
        sb $t3, 0($t2)          # must reassign expected row
        lb $a0, 0($t1)
        bne $a0, $0, prn_cod
        li $t3, 0x84            # check row 4 and re-enable bit 7
        sb $t3, 0($t2)          # must reassign expected row
        lb $a0, 0($t1)
        bne $a0, $0, prn_cod
        li $t3, 0x88            # check row 4 and re-enable bit 7
        sb $t3, 0($t2)          # must reassign expected row
        lb $a0, 0($t1)
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
```

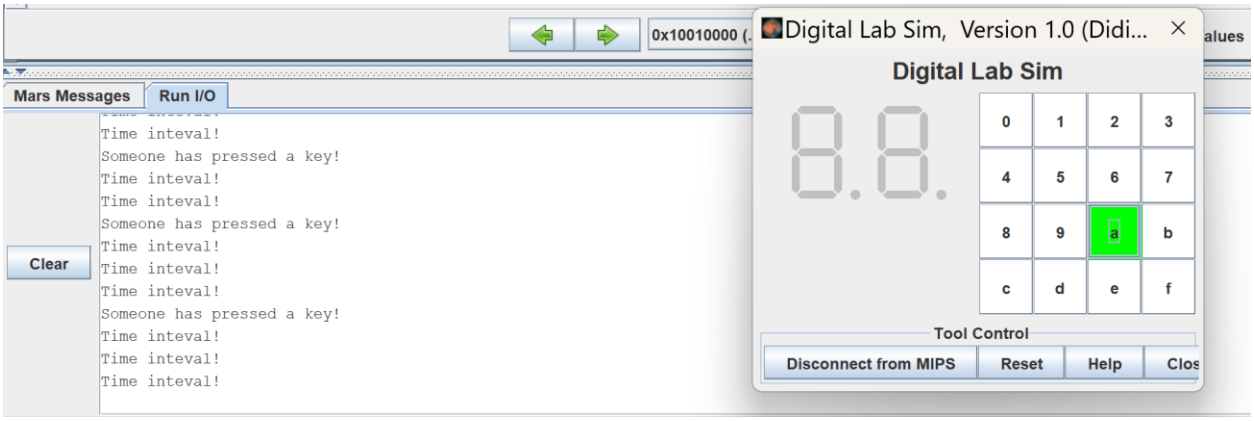


Assignment 4:

.eqv IN_ADRESS_HEXA_KEYBOARD 0xFFFF0012

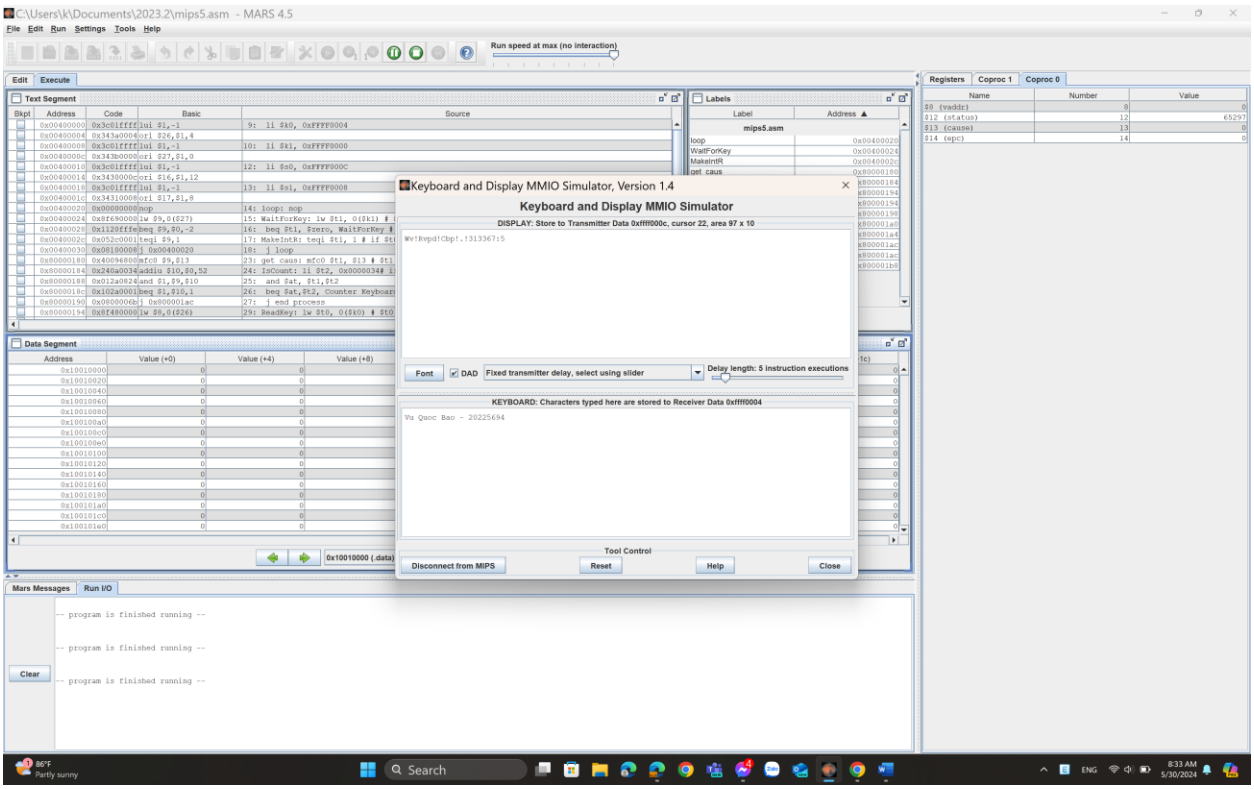
```
.eqv COUNTER 0xFFFF0013 # Time Counter
.eqv MASK_CAUSE_COUNTER 0x00000400 # Bit 10: Counter interrupt
.eqv MASK_CAUSE_KEYMATRIX 0x00000800 # Bit 11: Key matrix interrupt
.data
msg_keypress: .asciiz "Someone has pressed a key!\n"
msg_counter: .asciiz "Time inteval!\n"
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# 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
 li $t1, COUNTER
 sb $t1, 0($t1)

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



<u>Assignment 5:</u>

```
.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 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
```


C:\Users\k\Documents\2023.2\mips5.asm - MARS 4.5
File Edit Run Settings Tools Help
Run speed at max (no interaction)
Edit Execute
Text Segment
Bkpt Address Code Basic Source
Labels
Label Address
mips5.asm
loop 0x00400020
WaitForKey 0x00400024
MakeIntRt 0x00400028
get_caus 0x80000180
Registers Coproc 1 Coproc 0
Name Number Value
$9 (vaddr) 9 0
$12 (status) 12 65297
$13 (cause) 13 0
$14 (epc) 14 0
Keyboard and Display MMIO Simulator, Version 1.4
Keyboard and Display MMIO Simulator
DISPLAY: Store to Transmitter Data 0xffff000c, cursor 22, area 97 x 10
Font
DAD
Fixed transmitter delay, select using slider
Delay length: 5 instruction executions
KEYBOARD: Characters typed here are stored to Receiver Data 0xffff0004
Vu Quoc Bao - 20225694
Tool Control
Disconnect from MIPS
Reset
Help
Close
Data Segment
Address Value (+0) Value (+4) Value (+8)
0x10010000 (.data)
Mars Messages Run I/O
-- program is finished running --
Clear
86°F
Partly sunny
Search
ENG
8:33 AM
5/30/2024