BÁO CÁO THỰC HÀNH
THỰC HÀNH KIẾN TRÚC MÁY TÍNH
Tuần 11:

Họ và tên	Phạm Quốc Cường
Mã số sinh viên	20225604

Assignment 1:

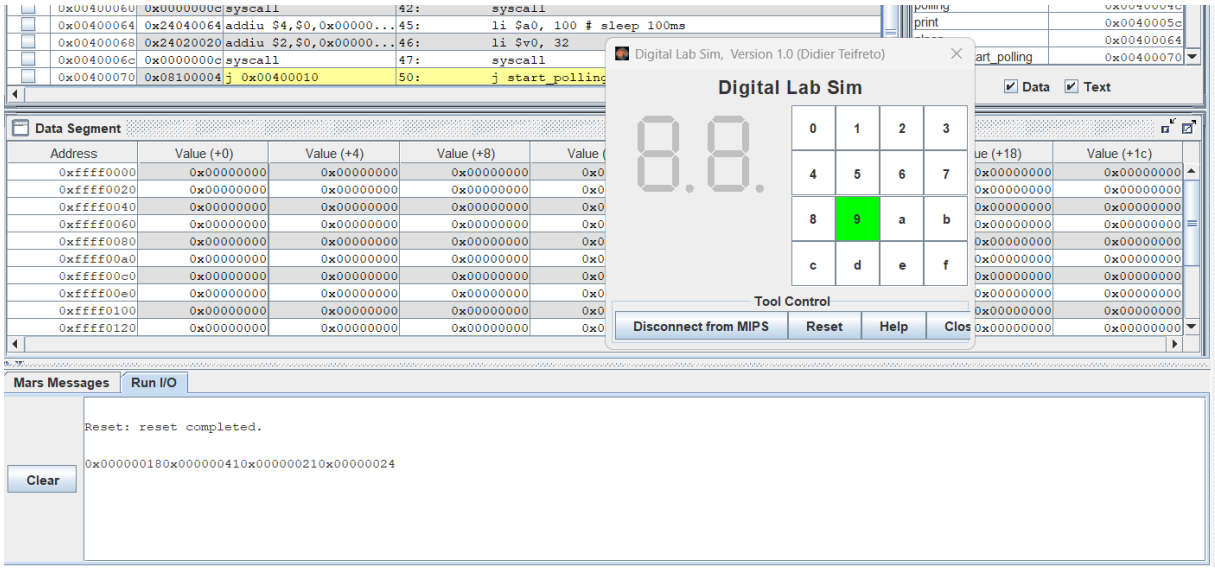
Code:

```
.eqv IN_ADRESS_HEXA_KEYBOARD 0xFFFF0012
.eqv OUT_ADRESS_HEXA_KEYBOARD 0xFFFF0014
.text
main:
li $t1, IN_ADRESS_HEXA_KEYBOARD
        li $t2, OUT_ADRESS_HEXA_KEYBOARD

start_polling_1:
  li $t3, 0x01 # check row 1 with key 0, 1, 2, 4
  sb $t3, 0($t1) # must reassign expected row
  jal polling

start_polling_2:
li $t3, 0x02 # check row 2 with key 4, 5, 6, 7
sb $t3, 0($t1) # must reassign expected row
jal polling

start_polling_3:
li $t3, 0x04 # check row 3 with key 8, 9, A, B
sb $t3, 0($t1) # must reassign expected row
jal polling

start_polling_4:
        li $t3, 0x08 # check row 4 with key C, D, E, F
        sb $t3, 0($t1) # must reassign expected row
```

```
        jal polling
check_after_polling_4:
beq $a0, 0x0, print
        j start_polling_1
polling:
        lb $a0, 0($t2) # read scan code of key button
        bne $a0, 0x0, print
        jr $ra
print:
li $v0, 34 # print integer (hexa)
        syscall
sleep:
li $a0, 100 # sleep 100ms
        li $v0, 32
        syscall
back_to_start_polling:
j start_polling_1     # back to check row 1
```

Result:

Kết quả khi nhập lần lượt: **c-2-1-9**



Explain:

Các nhãn start_polling 1,2,3,4 quét các dòng trong bàn phím bằng cách gán địa chỉ tương ứng từng dòng vào thanh ghi t3.

Assignment 2:

Code:

```
.eqv IN_ADRESS_HEXA_KEYBOARD 0xFFFF0012
.eqv OUT_ADRESS_HEXA_KEYBOARD 0xFFFF0014

.data
Message: .asciiz "Oh my god. Cuong Pham pressed a button. Key: "

.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 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:

.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)
```

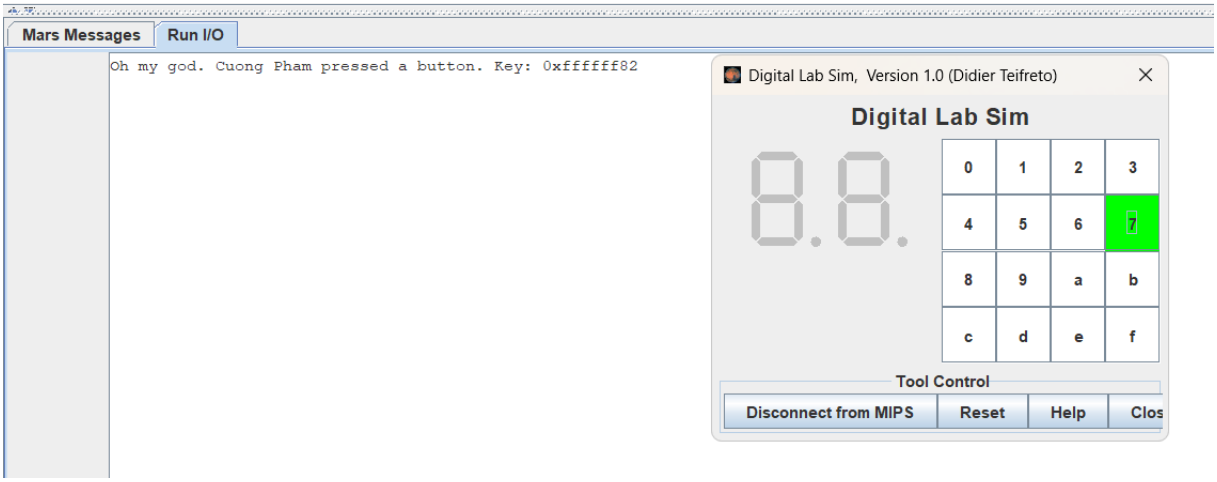
```
    addi $sp, $sp, 4 # Save $at because we may change it later
    sw $at, 0($sp)
    addi $sp, $sp, 4 # Save $v0 because we may change it later
    sw $v0, 0($sp)
    addi $sp, $sp, 4 # Save $a0 because we may change it later
    sw $a0, 0($sp)
    addi $sp, $sp, 4 # Save $t1 because we may change 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_ADRESS_HEXA_KEYBOARD
    li $t2, OUT_ADRESS_HEXA_KEYBOARD

start_interrupt_1:
    li $t3, 0x81 # check row 1 with key 0, 1, 2, 3
    sb $t3, 0($t1) # must reassign expected row
    jal interrupt

start_interrupt_2:
    li $t3, 0x82 # check row 2 with key 4, 5, 6, 7
    sb $t3, 0($t1) # must reassign expected row
    jal interrupt
```

```
start_interrupt_3:
    li $t3, 0x84 # check row 3 with key 8, 9, A, B
    sb $t3, 0($t1) # must reassign expected row
    jal interrupt

start_interrupt_4:
    li $t3, 0x88 # check row 4 with key C, D, E, F
    sb $t3, 0($t1) # must reassign expected row
    jal interrupt

check_after_interrupt_4:
    beq $a0, 0x0, prn_cod
    j next_pc

interrupt:
    lb $a0, 0($t2) # read scan code of key button
    bne $a0, 0x0, prn_cod
    jr $ra

prn_cod:
    li $v0, 34
    syscall
    li $v0, 11
    li $a0, '\n' # print end of line
    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

return:
    eret # Return from exception
```

Result:

Ấn phím bất kỳ trên bàn phím thì màn hình in ra Message và giá trị của phím vừa bấm.



Explain:

$t3 lưu địa chỉ 0x80000180 mà ở đó khi bit thứ 7 bằng 1 chức interrupt được kích hoạt.

Sử dụng chỉ thị .ktext để viết code ở địa chỉ 0x80000180 nói trên.

Sau khi kết thúc chương trình con, sử dụng lệnh eret để quay trở lại chương trình chính. Lệnh eret sẽ gán nội dung thanh ghi PC bằng giá trị trong thanh ghi $14 ($t6)

Assignment 3:

Code:

```
.eqv IN_ADRESS_HEXA_KEYBOARD 0xFFFF0012
.eqv OUT_ADRESS_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_ADRESS_HEXA_KEYBOARD
    li $t3, 0x80 # bit 7 = 1 to enable
    sb $t3, 0($t1)

    #-----------------------------------------------------------
    # Loop and 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 end of line
    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 $v0 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_ADRESS_HEXA_KEYBOARD
    li $t2, OUT_ADRESS_HEXA_KEYBOARD

start_interrupt_1:
    li $t3, 0x81 # check row 1 with key 0, 1, 2, 3
    sb $t3, 0($t1) # must reassign expected row
    jal interrupt

start_interrupt_2:
    li $t3, 0x82 # check row 2 with key 4, 5, 6, 7
    sb $t3, 0($t1) # must reassign expected row
    jal interrupt

start_interrupt_3:
    li $t3, 0x84 # check row 3 with key 8, 9, A, B
    sb $t3, 0($t1) # must reassign expected row
```

```
    jal interrupt

start_interrupt_4:
    li $t3, 0x88 # check row 4 with key C, D, E, F
    sb $t3, 0($t1) # must reassign expected row
    jal interrupt

check_after_interrupt_4:
    beq $a0, 0x0, prn_cod
    j next_pc

interrupt:
    lb $a0, 0($t2) # read scan code of key button
    bne $a0, 0x0, prn_cod
    jr $ra

prn_cod:
    li $v0, 34
    syscall
    li $v0, 11
    li $a0, '\n' # print end of line
    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

return:
    eret # Return from exception
```

Result: