Họ và Tên: Hà Trung Chiến
MSSV: 20225794

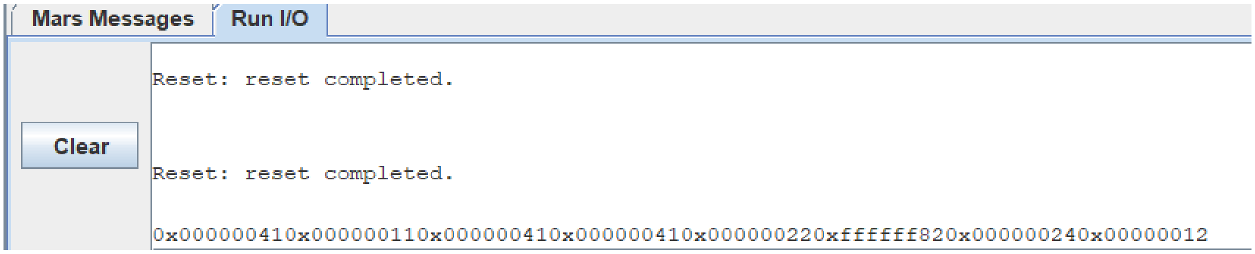
Báo cáo Lab 11

Assignment 1:

- *Mã nguồn:*

```
#-----------------------------------------------------
#          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,
.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
```

- *Kết quả chạy:*



```
Reset: reset completed.


Reset: reset completed.

0x000000410x000000110x000000410x000000410x000000220xffffff820x000000240x00000012
```

Input: MSSV: 20225794

- *Giải thích:*

Chương trình xác định phím được nhấn (bao gồm các phím từ 0 tới F)
Kết quả in ra khi nhập MSSV: 20225794 từ bàn phím

0x00000041 là số 2
0x00000011 là số 0
0x00000041 là số 2
0x00000022 là số 5
0xffffff82 là số 7
0x00000024 là số 9
0x00000012 là số 4
→ Kết quả đúng với lý thuyết

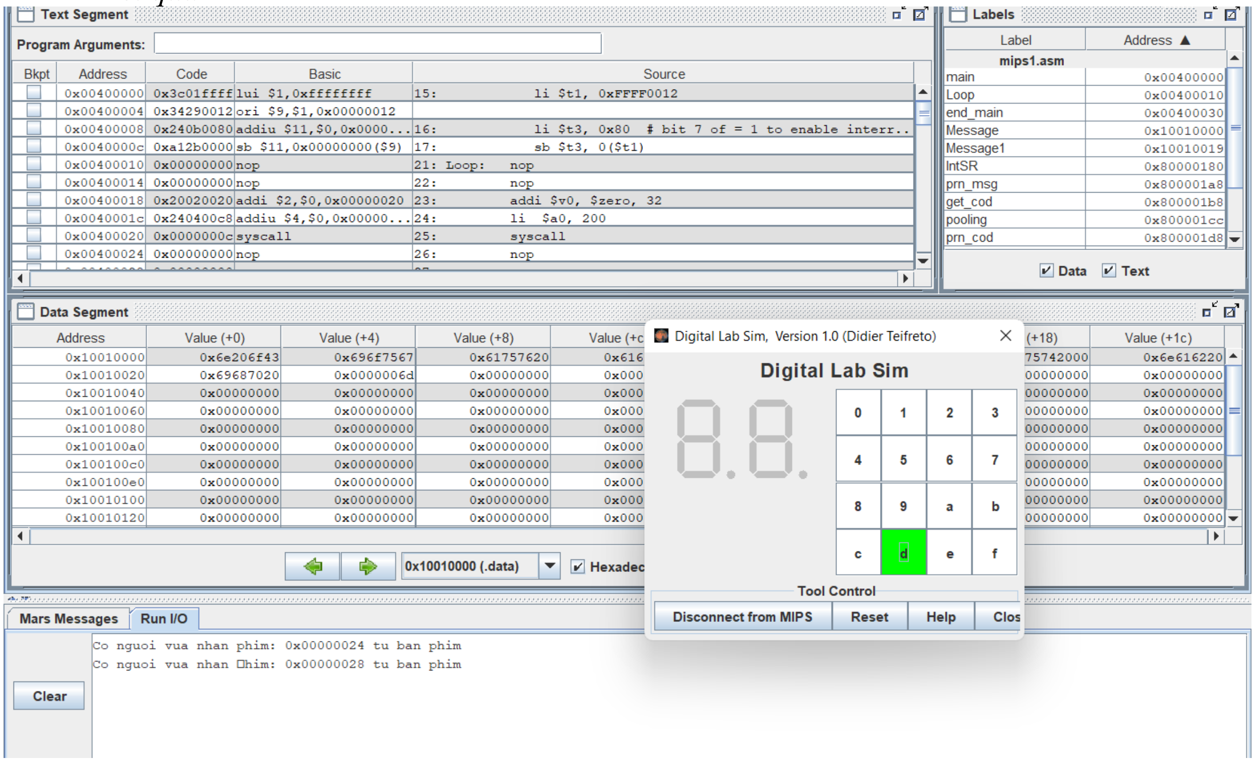
Assignment 2:

- *Mã nguồn:*

```
.eqv IN_ADDRESS_HEXA_KEYBOARD       0xFFFF0012
.eqv OUT_ADDRESS_HEXA_KEYBOARD      0xFFFF0014
.data
Message: .asciiz "Co nguoi vua nhan phim: "
Message1: .asciiz " tu ban phim"
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# 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
      addi $v0, $zero, 32
      li  $a0, 200
      syscall
      nop
      nop
      b  Loop           # Wait for interrupt
end_main:

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# GENERAL INTERRUPT SERVED ROUTINE for all interrupts
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
.ktext 0x80000180
      #---------------------------------------------------------
      # SAVE the current REG FILE to stack
```

```
        #-----------------------------------------------------
IntSR:  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_ADDRESS_HEXA_KEYBOARD
        li      $t2,   OUT_ADDRESS_HEXA_KEYBOARD
        li      $t3,   0x1       # check first row
pooling:
        sb      $t3,   0($t1)    # must reassign expected row
        lb      $a0,   0($t2)
        beqz    $a0,   back_to_pooling
prn_cod:li      $v0,34
        syscall
        #-----------------------------------------------------
        # Evaluate the return address of main routine
        # epc  <= epc + 4
        #-----------------------------------------------------
        li    $t3,   0x80      # bit 7 = 1 to enable
        sb    $t3,   0($t1)
        j next_pc
back_to_pooling:
        sll $t3, $t3, 1
        ble $t3, 0x8, pooling
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      $at, 0($sp)     # Restore the registers from stack
        addi    $sp,$sp,-4

return:
        addi    $v0, $zero,  4
        la      $a0, Message1
        syscall
        li      $v0,11
        li      $a0,'\n'        # print end of line
        syscall
        eret                    # Return from exception
```

- *Kết quả:*



Text Segment

Program Arguments:

Bkpt	Address	Code	Basic	Source
□	0x00400000	0x3c01ffff	lui $1,0xffffffff	15: li $t1, 0xFFFF0012
□	0x00400004	0x34290012	ori $9,$1,0x00000012	
□	0x00400008	0x240b0080	addiu $11,$0,0x0000...	16: li $t3, 0x80 # bit 7 of = 1 to enable interr..
□	0x0040000c	0xa12b0000	sb $11,0x00000000($9)	17: sb $t3, 0($t1)
□	0x00400010	0x00000000	nop	21: Loop: nop
□	0x00400014	0x00000000	nop	22: nop
□	0x00400018	0x20020020	addi $2,$0,0x00000020	23: addi $v0, $zero, 32
□	0x0040001c	0x240400c8	addiu $4,$0,0x00000...	24: li $a0, 200
□	0x00400020	0x0000000c	syscall	25: syscall
□	0x00400024	0x00000000	nop	26: nop

Labels

Label	Address ▲
mips1.asm	
main	0x00400000
Loop	0x00400010
end_main	0x00400030
Message	0x10010000
Message1	0x10010019
IntSR	0x80000180
prn_msg	0x800001a8
get_cod	0x800001b8
pooling	0x800001cc
prn_cod	0x800001d8

☑ Data ☑ Text

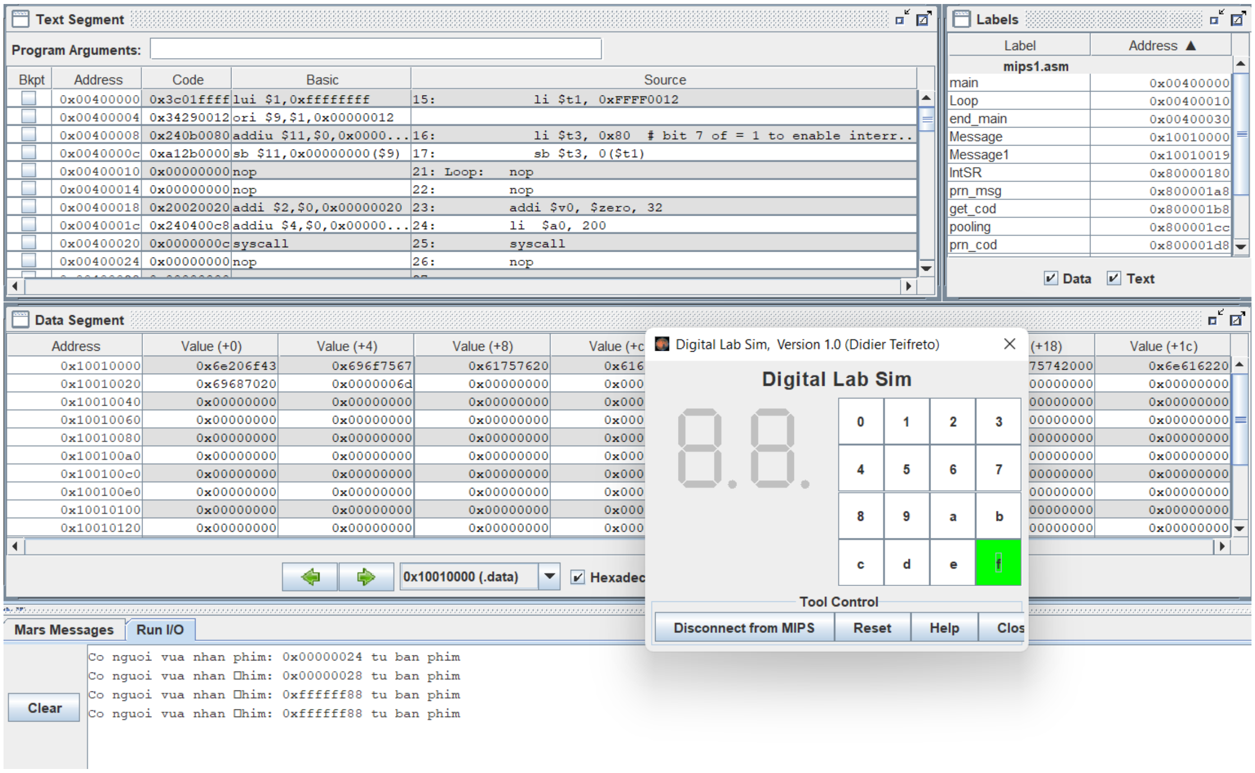
Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c	(+18)	Value (+1c)
0x10010000	0x6e206f43	0x696f7567	0x61757620	0x616	75742000	0x6e616220
0x10010020	0x69687020	0x0000006d	0x00000000	0x000	00000000	0x00000000
0x10010040	0x00000000	0x00000000	0x00000000	0x000	00000000	0x00000000
0x10010060	0x00000000	0x00000000	0x00000000	0x000	00000000	0x00000000
0x10010080	0x00000000	0x00000000	0x00000000	0x000	00000000	0x00000000
0x100100a0	0x00000000	0x00000000	0x00000000	0x000	00000000	0x00000000
0x100100c0	0x00000000	0x00000000	0x00000000	0x000	00000000	0x00000000
0x100100e0	0x00000000	0x00000000	0x00000000	0x000	00000000	0x00000000
0x10010100	0x00000000	0x00000000	0x00000000	0x000	00000000	0x00000000
0x10010120	0x00000000	0x00000000	0x00000000	0x000	00000000	0x00000000

0x10010000 (.data) ☑ Hexadec

Mars Messages | Run I/O

```
Co nguoi vua nhan phim: 0x00000024 tu ban phim
Co nguoi vua nhan □him: 0x00000028 tu ban phim
```



Digital Lab Sim, Version 1.0 (Didier Teifreto)

Bkpt	Address	Code	Basic	Source
□	0x00400000	0x3c01ffff	lui $1,0xffffffff	15: li $t1, 0xFFFF0012
□	0x00400004	0x34290012	ori $9,$1,0x00000012	
□	0x00400008	0x240b0080	addiu $11,$0,0x0000...	16: li $t3, 0x80 # bit 7 of = 1 to enable interr..
□	0x0040000c	0xa12b0000	sb $11,0x00000000($9)	17: sb $t3, 0($t1)
□	0x00400010	0x00000000	nop	21: Loop: nop
□	0x00400014	0x00000000	nop	22: nop
□	0x00400018	0x20020020	addi $2,$0,0x00000020	23: addi $v0, $zero, 32
□	0x0040001c	0x240400c8	addiu $4,$0,0x00000...	24: li $a0, 200
□	0x00400020	0x0000000c	syscall	25: syscall
□	0x00400024	0x00000000	nop	26: nop

Label	Address ▲
mips1.asm	
main	0x00400000
Loop	0x00400010
end_main	0x00400030
Message	0x10010000
Message1	0x10010019
IntSR	0x80000180
prn_msg	0x800001a8
get_cod	0x800001b8
pooling	0x800001cc
prn_cod	0x800001d8

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c	(+18)	Value (+1c)
0x10010000	0x6e206f43	0x696f7567	0x61757620	0x616	75742000	0x6e616220
0x10010020	0x69687020	0x0000006d	0x00000000	0x000	00000000	0x00000000
0x10010040	0x00000000	0x00000000	0x00000000	0x000	00000000	0x00000000
0x10010060	0x00000000	0x00000000	0x00000000	0x000	00000000	0x00000000
0x10010080	0x00000000	0x00000000	0x00000000	0x000	00000000	0x00000000
0x100100a0	0x00000000	0x00000000	0x00000000	0x000	00000000	0x00000000
0x100100c0	0x00000000	0x00000000	0x00000000	0x000	00000000	0x00000000
0x100100e0	0x00000000	0x00000000	0x00000000	0x000	00000000	0x00000000
0x10010100	0x00000000	0x00000000	0x00000000	0x000	00000000	0x00000000
0x10010120	0x00000000	0x00000000	0x00000000	0x000	00000000	0x00000000

```
Co nguoi vua nhan phim: 0x00000024 tu ban phim
Co nguoi vua nhan □him: 0x00000028 tu ban phim
Co nguoi vua nhan □him: 0xffffff88 tu ban phim
Co nguoi vua nhan □him: 0xffffff88 tu ban phim
```

- Khi ấn phím bất kì sẽ hiện ra phím họ thông báo về phím họ vừa nhấn

Assignment 3:

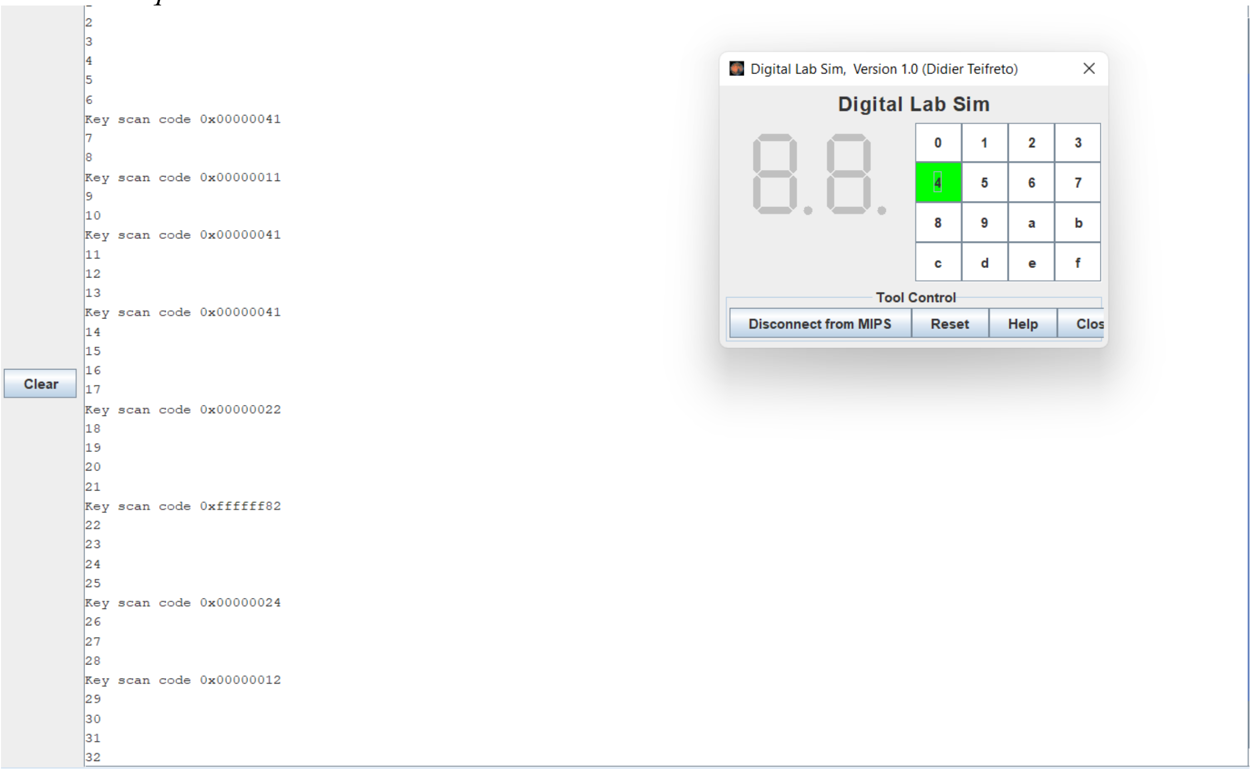
- *Mã nguồn:*

```
.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 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_ADRESS_HEXA_KEYBOARD
```

```
        li      $t3, 0x81       # check row 4 and re-enable bit 7
        sb      $t3, 0($t1)     # must reassign expected row
        li      $t1, OUT_ADRESS_HEXA_KEYBOARD
        lb      $a0, 0($t1)
        bne     $a0, $zero, prn_cod

        li      $t1, IN_ADRESS_HEXA_KEYBOARD
        li      $t3, 0x82       # check row 4 and re-enable bit 7
        sb      $t3, 0($t1)     # must reassign expected row
        li      $t1, OUT_ADRESS_HEXA_KEYBOARD
        lb      $a0, 0($t1)
        bne     $a0, $zero, prn_cod

        li      $t1, IN_ADRESS_HEXA_KEYBOARD
        li      $t3, 0x84       # check row 4 and re-enable bit 7
        sb      $t3, 0($t1)     # must reassign expected row
        li      $t1, OUT_ADRESS_HEXA_KEYBOARD
        lb      $a0, 0($t1)
        bne     $a0, $zero, prn_cod

        li      $t1, IN_ADRESS_HEXA_KEYBOARD
        li      $t3, 0x88       # check row 4 and re-enable bit 7
        sb      $t3, 0($t1)     # must reassign expected row
        li      $t1, OUT_ADRESS_HEXA_KEYBOARD
        lb      $a0, 0($t1)
        bne     $a0, $zero, 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
```

- *Kết quả:*



```
2
3
4
5
6
Key scan code 0x00000041
7
8
Key scan code 0x00000011
9
10
Key scan code 0x00000041
11
12
13
Key scan code 0x00000041
14
15
16
17
Key scan code 0x00000022
18
19
20
21
Key scan code 0xffffff82
22
23
24
25
Key scan code 0x00000024
26
27
28
Key scan code 0x00000012
29
30
31
32
```

- *Giải thích:*

Kết quả in ra khi nhập MSSV: 20225794 từ bàn phím
0x00000041 là số 2
0x00000011 là số 0
0x00000041 là số 2
0x00000022 là số 5
0xffffff82 là số 7
0x00000024 là số 9
0x00000012 là số 4
→ Kết quả đúng với lý thuyết