Báo cáo thực hành Kiến trúc máy tính – IT3280

Họ và tên: Thân Cát Ngọc Lan

MSSV: 20225646

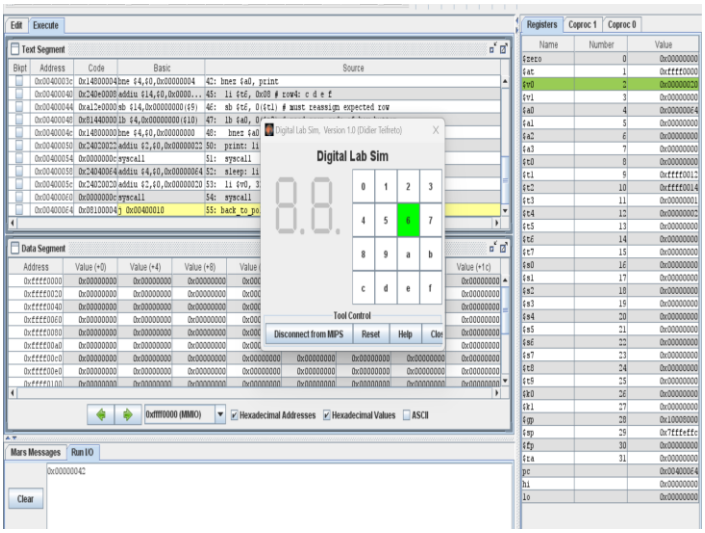
Assignment 1:

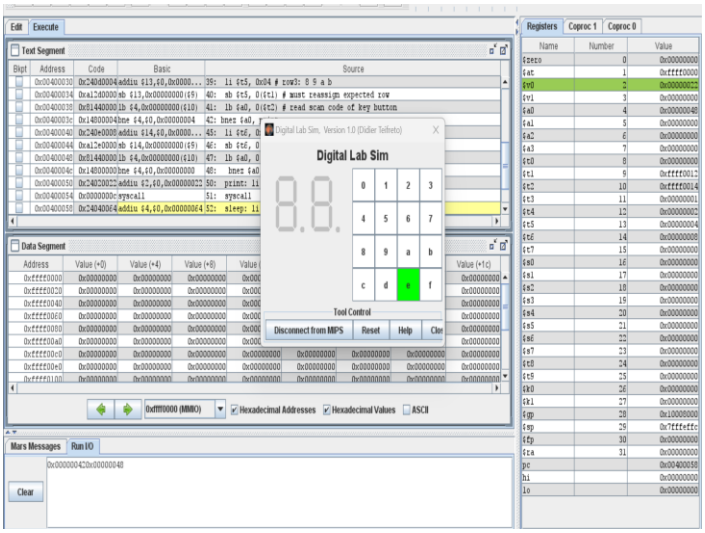
*Code:

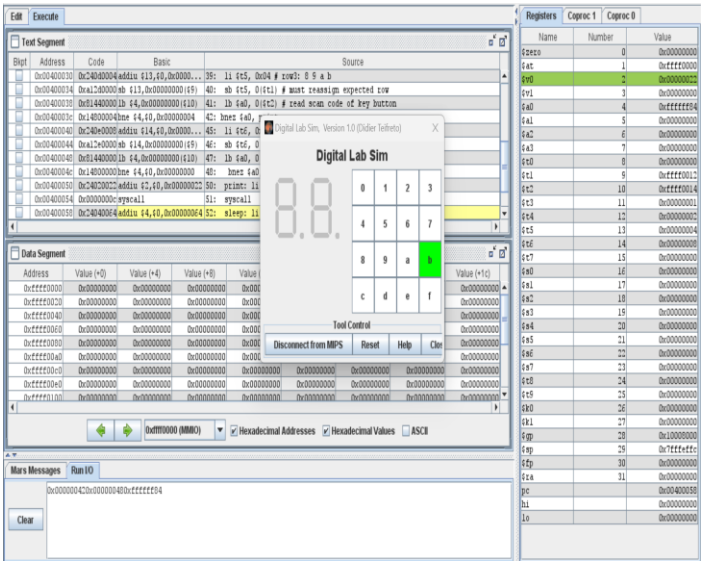
```
#BAI1
#---------------------------------------------------
# col 0x1 col 0x2 col 0x4 col 0x8
#
# row 0x1 0     1    2    3
#        0x11 0x21 0x41 0x81
#
# row 0x2 4    5    6   7
#       0x12 0x22 0x42 0x82
#
# row 0x4 8    9    a   b
#       0x14 0x24 0x44 0x84
#
# row 0x8 c    d    e    f
#       0x18 0x28 0x48 0x88
#
#---------------------------------------------------
.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
```

```
polling:
row1:
 li $t3, 0x01 # row1: 0 1 2 3
 sb $t3, 0($t1) # must reassign expected row
lb $a0, 0($t2) # read scan code of key button
 bnez $a0, print

row2:
 li $t4, 0x02 # row2: 4 5 6 7
 sb $t4, 0($t1) # must reassign expected row
 lb $a0, 0($t2) # read scan code of key button
 bnez $a0, print

 row3:
 li $t5, 0x04 # row3: 8 9 a b
 sb $t5, 0($t1) # must reassign expected row
 lb $a0, 0($t2) # read scan code of key button
bnez $a0, print

 row4:
 li $t6, 0x08 # row4: c d e f
 sb $t6, 0($t1) # must reassign expected row
 lb $a0, 0($t2) # read scan code of key button
  bnez $a0, print

 print: li $v0, 34 # print integer (hexa)
 syscall
 sleep: li $a0, 100 # sleep 100ms
 li $v0, 32
 syscall
back_to_polling: j polling # continue polling
```

*Kết quả:







*Giải thích:

Chương trình trên in số phím được nhấn (tính theo số thứ tự cột và hàng trên bàn phím theo hệ hexa)

Ví dụ: 0x00000042 tương đương với phím ở cột 3 (0100) hàng 2 (0010) là phím "6"

Kết quả chạy đúng theo ví dụ trên.

Thực thi: Gán lần lượt các giá trị 0x01,0x02,0x04,0x08 vào $t1. Sau đó kiểm tra $t2 xem hàng nào được bấm. Nếu $t2!=0 thì in ra.

Assignment 2:

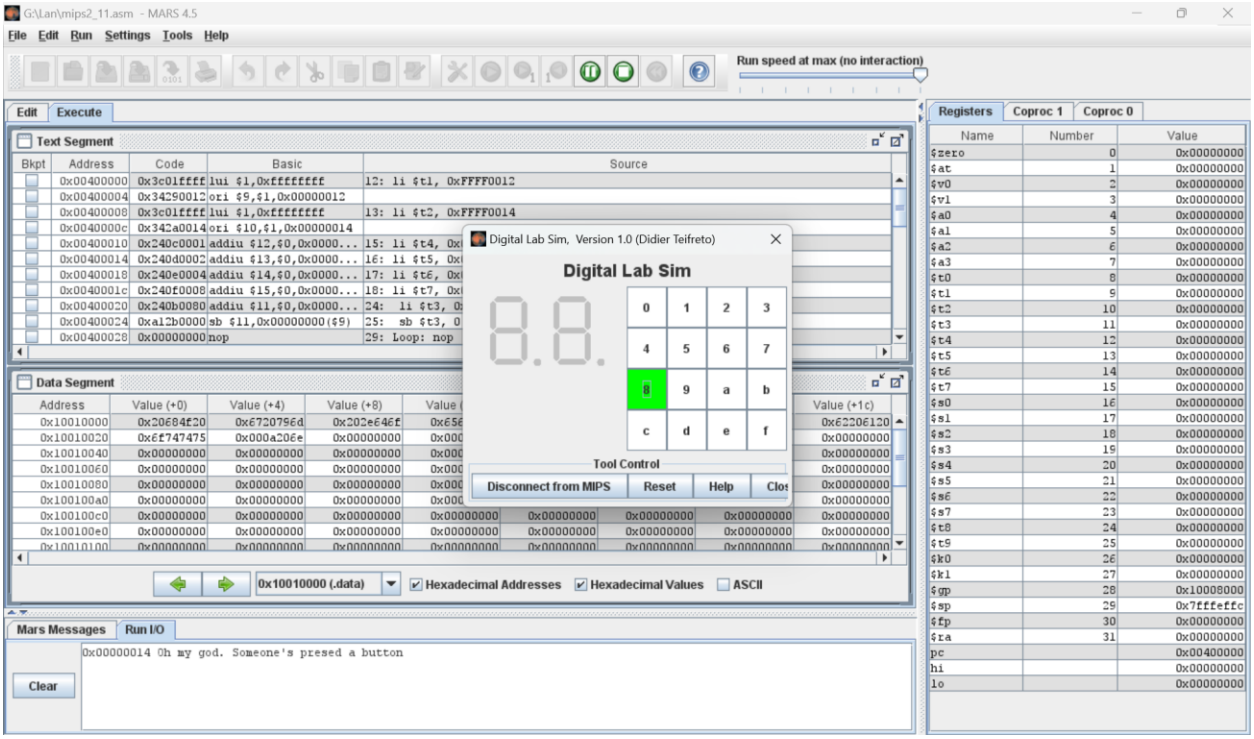
*Code:

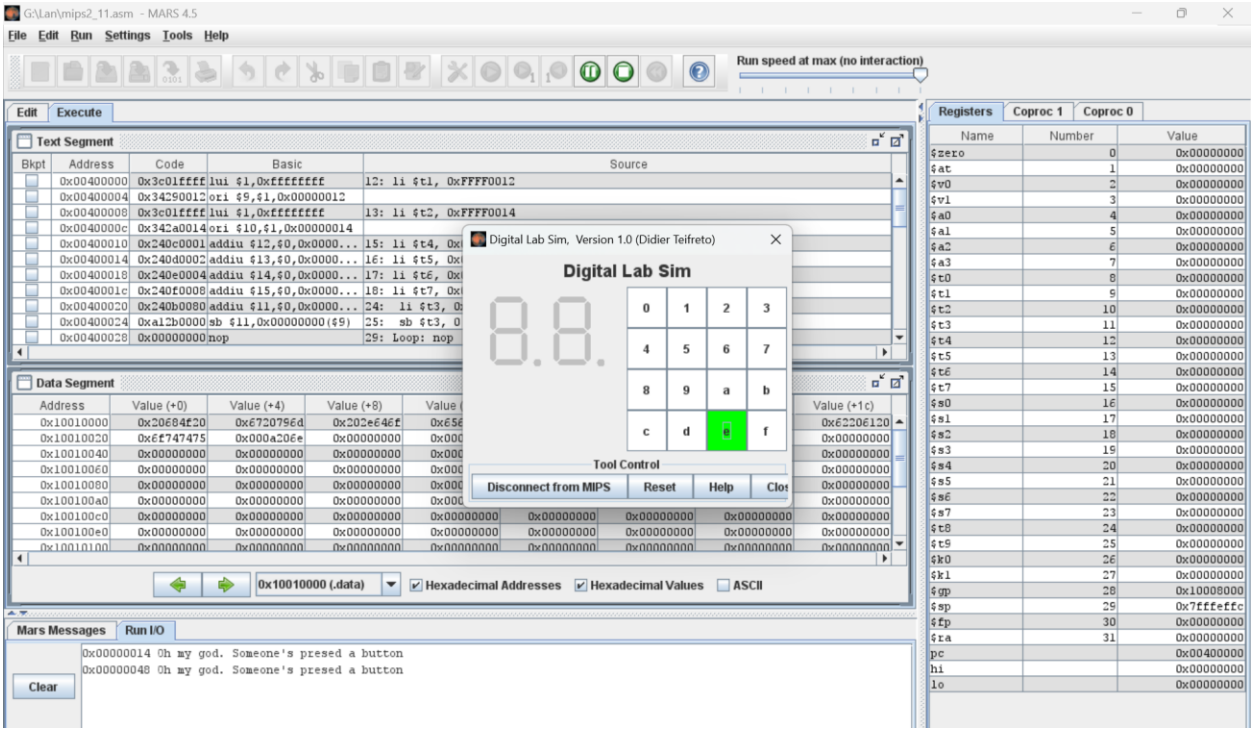
```
#bai2
.eqv IN_ADRESS_HEXA_KEYBOARD 0xFFFF0012
.eqv OUT_ADRESS_HEXA_KEYBOARD 0xFFFF0014
.data
Message: .asciiz " Oh my god. Someone's presed a button \n"
Message1: .asciiz "\n"

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# MAIN Procedure
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
.text
main:
li $t1, IN_ADRESS_HEXA_KEYBOARD
li $t2, OUT_ADRESS_HEXA_KEYBOARD

li $t4, 0x01 # row1: 0 1 2 3
li $t5, 0x02 # row2: 4 5 6 7
li $t6, 0x04 # row3: 8 9 a b
li $t7, 0x08 # row4: c d e f
```

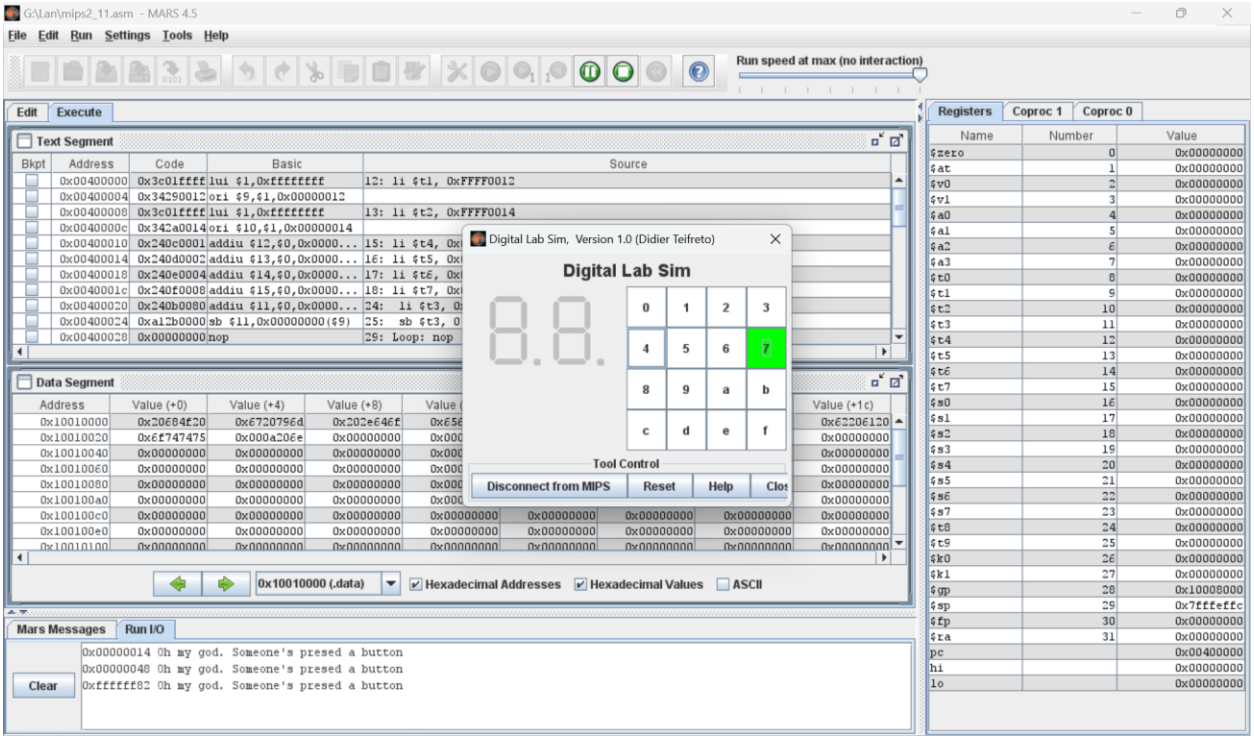
```
 #-----------------------------------------------------------
 # Enable interrupts you expect
 #-----------------------------------------------------------
 # Enable the interrupt of Keyboard matrix 4x4 of Digital Lab Sim
 li $t3, 0x80 # bit 7 of = 1 to enable interrupt
 sb $t3, 0($t1)
 #-----------------------------------------------------------
 # 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
 #-----------------------------------------------------------
 polling:
 sb $t4, 0($t1) # must reassign expected row
 lb $a0, 0($t2) # read scan code of key button
 bnez $a0, print

 sb $t5, 0($t1) # must reassign expected row
 lb $a0, 0($t2) # read scan code of key button
 bnez $a0, print
```

```
 sb $t6, 0($t1) # must reassign expected row
 lb $a0, 0($t2) # read scan code of key button
 bnez $a0, print

 sb $t7, 0($t1) # must reassign expected row
 lb $a0, 0($t2) # read scan code of key button
 bnez $a0, print

print: li $v0, 34 # print integer (hexa)
 syscall
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
 li $t3, 0x80 # bit 7 of = 1 to enable interrupt
 sb $t3, 0($t1)
return: eret # Return from exception
```

*Kết quả:


G:\Lan\mips2_11.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
0x00400000 0x3c01ffff lui $1,0xffffffff 12: li $t1, 0xFFFF0012
0x00400004 0x34290012 ori $9,$1,0x00000012
0x00400008 0x3c01ffff lui $1,0xffffffff 13: li $t2, 0xFFFF0014
0x0040000c 0x342a0014 ori $10,$1,0x00000014
0x00400010 0x240c0001 addiu $12,$0,0x0000... 15: li $t4, 0x
0x00400014 0x240d0002 addiu $13,$0,0x0000... 16: li $t5, 0x
0x00400018 0x240e0004 addiu $14,$0,0x0000... 17: li $t6, 0x
0x0040001c 0x240f0008 addiu $15,$0,0x0000... 18: li $t7, 0x
0x00400020 0x240b0080 addiu $11,$0,0x0000... 24: li $t3, 0x
0x00400024 0xa12b0000 sb $11,0x00000000($9) 25: sb $t3, 0
0x00400028 0x00000000 nop 29: Loop: nop
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 Clos
Data Segment
Address Value (+0) Value (+4) Value (+8) Value (+1c)
0x10010000 0x20684f20 0x6720796d 0x202e64ef 0x62206120
0x10010020 0x6f747475 0x000a206e 0x00000000 0x00000000
0x10010040 0x00000000 0x00000000 0x00000000 0x00000000
0x100100e0 0x00000000 0x00000000 0x00000000 0x00000000
0x10010080 0x00000000 0x00000000 0x00000000 0x00000000
0x100100a0 0x00000000 0x00000000 0x00000000 0x00000000
0x100100c0 0x00000000 0x00000000 0x00000000 0x00000000
0x100100e0 0x00000000 0x00000000 0x00000000 0x00000000
0x10010100 0x00000000 0x00000000 0x00000000 0x00000000
0x10010000 (.data) Hexadecimal Addresses Hexadecimal Values ASCII
Mars Messages Run I/O
0x00000014 0h my god. Someone's presed a button
Clear
Registers Coproc 1 Coproc 0
Name Number Value
$zero 0 0x00000000
$at 1 0x00000000
$v0 2 0x00000000
$v1 3 0x00000000
$a0 4 0x00000000
$a1 5 0x00000000
$a2 6 0x00000000
$a3 7 0x00000000
$t0 8 0x00000000
$t1 9 0x00000000
$t2 10 0x00000000
$t3 11 0x00000000
$t4 12 0x00000000
$t5 13 0x00000000
$t6 14 0x00000000
$t7 15 0x00000000
$s0 16 0x00000000
$s1 17 0x00000000
$s2 18 0x00000000
$s3 19 0x00000000
$s4 20 0x00000000
$s5 21 0x00000000
$s6 22 0x00000000
$s7 23 0x00000000
$t8 24 0x00000000
$t9 25 0x00000000
$k0 26 0x00000000
$k1 27 0x00000000
$gp 28 0x10008000
$sp 29 0x7fffeffc
$fp 30 0x00000000
$ra 31 0x00000000
pc 0x00400000
hi 0x00000000
lo 0x00000000


G:\Lan\mips2_11.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
0x00400000 0x3c01ffff lui $1,0xffffffff 12: li $t1, 0xFFFF0012
0x00400004 0x34290012 ori $9,$1,0x00000012
0x00400008 0x3c01ffff lui $1,0xffffffff 13: li $t2, 0xFFFF0014
0x0040000c 0x342a0014 ori $10,$1,0x00000014
0x00400010 0x240c0001 addiu $12,$0,0x0000... 15: li $t4, 0x
0x00400014 0x240d0002 addiu $13,$0,0x0000... 16: li $t5, 0x
0x00400018 0x240e0004 addiu $14,$0,0x0000... 17: li $t6, 0x
0x0040001c 0x240f0008 addiu $15,$0,0x0000... 18: li $t7, 0x
0x00400020 0x240b0080 addiu $11,$0,0x0000... 24: li $t3, 0x
0x00400024 0xa12b0000 sb $11,0x00000000($9) 25: sb $t3, 0
0x00400028 0x00000000 nop 29: Loop: nop
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 Clos
Data Segment
Address Value (+0) Value (+4) Value (+8) Value (+1c)
0x10010000 0x20684f20 0x6720796d 0x202e64ef 0x62206120
0x10010020 0x6f747475 0x000a206e 0x00000000 0x00000000
0x10010040 0x00000000 0x00000000 0x00000000 0x00000000
0x100100e0 0x00000000 0x00000000 0x00000000 0x00000000
0x10010080 0x00000000 0x00000000 0x00000000 0x00000000
0x100100a0 0x00000000 0x00000000 0x00000000 0x00000000
0x100100c0 0x00000000 0x00000000 0x00000000 0x00000000
0x100100e0 0x00000000 0x00000000 0x00000000 0x00000000
0x10010100 0x00000000 0x00000000 0x00000000 0x00000000
0x10010000 (.data) Hexadecimal Addresses Hexadecimal Values ASCII
Mars Messages Run I/O
0x00000014 0h my god. Someone's presed a button
0x00000048 0h my god. Someone's presed a button
Clear
Registers Coproc 1 Coproc 0
Name Number Value
$zero 0 0x00000000
$at 1 0x00000000
$v0 2 0x00000000
$v1 3 0x00000000
$a0 4 0x00000000
$a1 5 0x00000000
$a2 6 0x00000000
$a3 7 0x00000000
$t0 8 0x00000000
$t1 9 0x00000000
$t2 10 0x00000000
$t3 11 0x00000000
$t4 12 0x00000000
$t5 13 0x00000000
$t6 14 0x00000000
$t7 15 0x00000000
$s0 16 0x00000000
$s1 17 0x00000000
$s2 18 0x00000000
$s3 19 0x00000000
$s4 20 0x00000000
$s5 21 0x00000000
$s6 22 0x00000000
$s7 23 0x00000000
$t8 24 0x00000000
$t9 25 0x00000000
$k0 26 0x00000000
$k1 27 0x00000000
$gp 28 0x10008000
$sp 29 0x7fffeffc
$fp 30 0x00000000
$ra 31 0x00000000
pc 0x00400000
hi 0x00000000
lo 0x00000000



*Giải thích:

Tương tự bài 1. Nhưng có xuất hiện chương trình con phục vụ ngắt được lưu trên địa chỉ cố định là 0x80000180.Và sử dụng chỉ thị .ktext để viết chương trình trên địa chỉ trên.

Chương trình trên chờ 1 ngắt từ bàn phím và in ra phím đó + thông báo.

Assignment 3:

*Code:

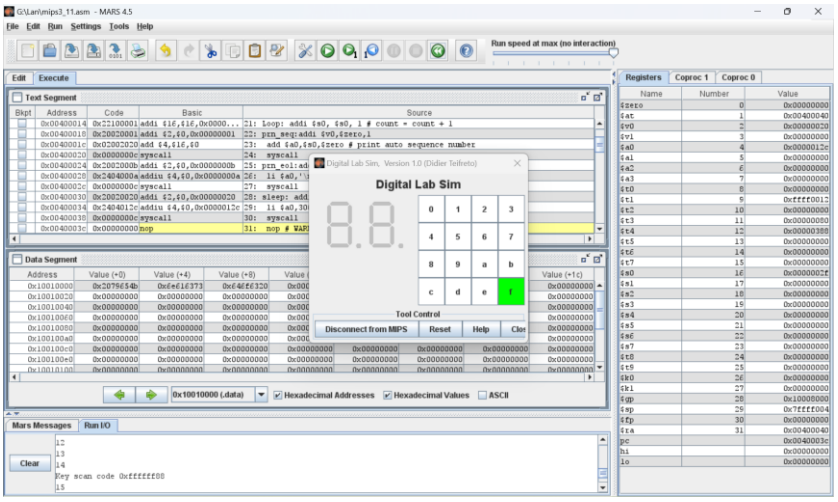
```
.eqv IN_ADRESS_HEXA_KEYBOARD 0xFFFF0012

.eqv OUT_ADRESS_HEXA_KEYBOARD 0xFFFF0014

.data

Message: .asciiz "Key scan code "

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# MAIN Procedure

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

.text

main:
```

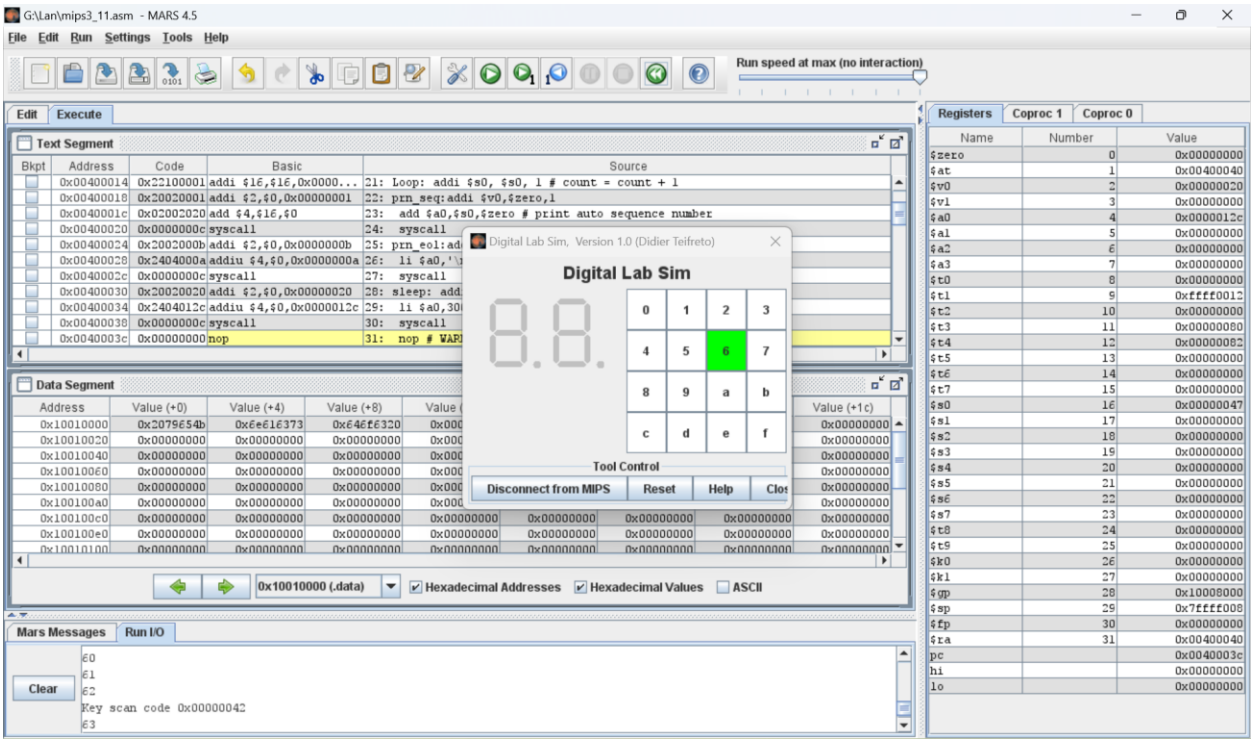
```
 #-----------------------------------------------------------
 # Enable interrupts you expect
 #-----------------------------------------------------------
 # Enable the interrupt of Keyboard matrix 4x4 of Digital Lab Sim
 li $t1, IN_ADRESS_HEXA_KEYBOARD
 li $t3, 0x80 # bit 7 = 1 to enable
 sb $t3, 0($t1)
 #-----------------------------------------------------------
 # Loop 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 $ra because we may change it later
 sw $at,0($sp)
 addi $sp,$sp,4 # Save $ra 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

 li $t3, 0x01
get_cod:li $t1, IN_ADRESS_HEXA_KEYBOARD
 ori $t4, $t3, 0x80 # check row 4 and re-enable bit 7
 sb $t4, 0($t1) # must reassign expected row
 li $t1, OUT_ADRESS_HEXA_KEYBOARD
 lb $a0, 0($t1)
 bne $a0, $0, prn_cod
#addi $t3, $t3, 0xffff ff80
 sll $t3, $t3,1
 addi $t3, $t3, 0x80
 j get_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
return: eret # Return from exception
```

*Kết quả:





*Giải thích:

Tương tự bài 2. Chương trình trên chỉ in ra số thứ tự và cho phép 1 ngắt từ bàn phím. Khi ngắt xảy ra sẽ in ra lệnh quét và phím được bấm.