Report Lab 11

Nguyễn Khánh Nam

20225749

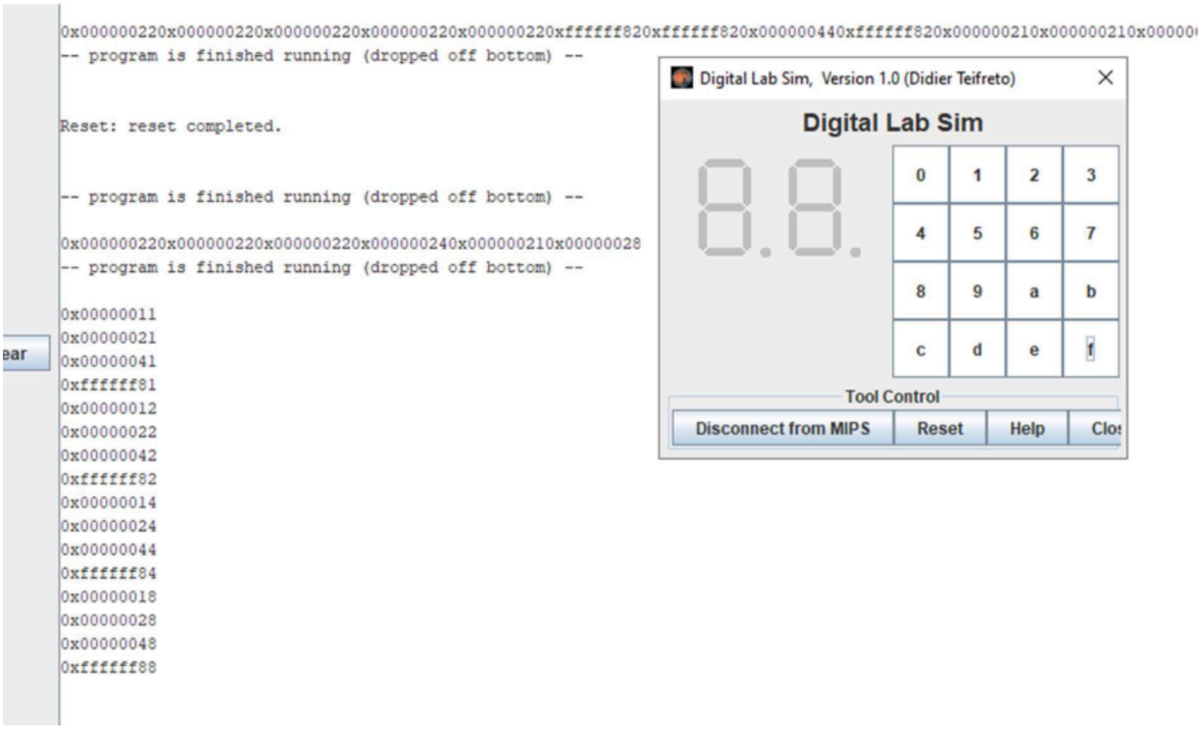
Assignment 1

Code:

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

```
.data
        n: .asciiz "\n"

.text
main:
        li     $t1,       IN_ADDRESS_HEXA_KEYBOARD
        li     $t2,       OUT_ADDRESS_HEXA_KEYBOARD
        li     $t3,       0x01   # check row 4 with key 0, 1, 2, 3
      li     $t4,       0x02   # check row 4 with key 4, 5, 6, 7
        li     $t5,       0x04   # check row 4 with key 8, 9, A, B
        li     $t6,       0x08   # check row 4 with key C, D, E, F
      li $t0, 0

polling:
        beq $t0, 100, exit

        sb      $t3,       0($t1)                 # must reassign expected row
        lb      $a0,       0($t2)                 # read scan code of key button
        bne $a0, $0, print

        sb      $t4,       0($t1)                 # must reassign expected row
        lb      $a0,       0($t2)                 # read scan code of key button
        bne $a0, $0, print

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

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

        j continue


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

   la $a0, n
   li $v0, 4
   syscall

continue:
        add $t0, $t0, 1

sleep:
        li $a0, 5000    #5s nhận 1 giá trị nhập vào
        li $v0, 32
        syscall


back_to_polling:
    j       polling                         # continue polling

exit:
```

Result:



```
0x000000220x000000220x000000220x000000220x000000220xffffff820xffffff820x000000440xffffff820x000000210x000000210x00000
-- program is finished running (dropped off bottom) --


Reset: reset completed.


-- program is finished running (dropped off bottom) --

0x000000220x000000220x000000220x000000240x000000210x00000028
-- program is finished running (dropped off bottom) --

0x00000011
0x00000021
0x00000041
0xffffff81
0x00000012
0x00000022
0x00000042
0xffffff82
0x00000014
0x00000024
0x00000044
0xffffff84
0x00000018
0x00000028
0x00000048
0xffffff88
```

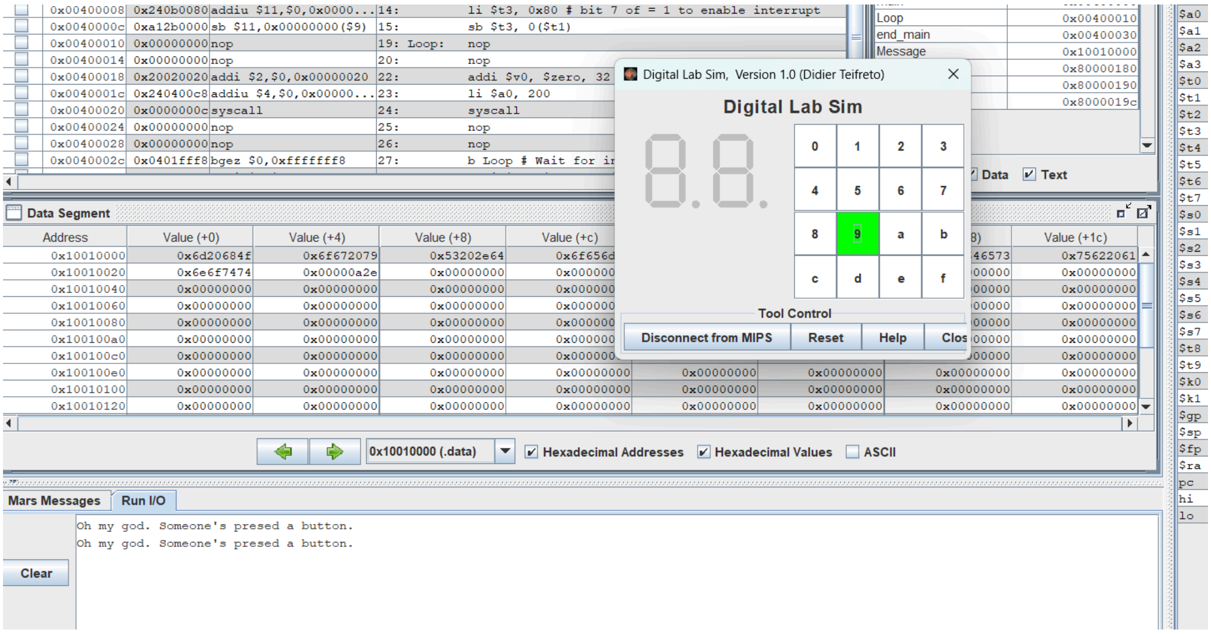
Assignment 2

Code:

```
.eqv IN_ADDRESS_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_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
#--------------------------------------------------------
# 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
```

Result:



```
0x00400008 0x240b0080 addiu $11,$0,0x0000... 14:          li $t3, 0x80 # bit 7 of = 1 to enable interrupt
0x0040000c 0xa12b0000 sb $11,0x00000000($9)   15:          sb $t3, 0($t1)
0x00400010 0x00000000 nop                     19: Loop:    nop
0x00400014 0x00000000 nop                     20:          nop
0x00400018 0x20020020 addi $2,$0,0x00000020   22:          addi $v0, $zero, 32
0x0040001c 0x240400c8 addiu $4,$0,0x00000...  23:          li $a0, 200
0x00400020 0x0000000c syscall                 24:          syscall
0x00400024 0x00000000 nop                     25:          nop
0x00400028 0x00000000 nop                     26:          nop
0x0040002c 0x0401fff8 bgez $0,0xfffffff8      27:          b Loop # Wait for ir
```

Loop	0x00400010
end_main	0x00400030
Message	0x10010000
	0x80000180
	0x80000190
	0x8000019c

☑ Data ☑ Text

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)			8)	Value (+1c)
0x10010000	0x6d20684f	0x6f672079	0x53202e64	0x6f656d			46573	0x75622061
0x10010020	0x6e6f7474	0x00000a2e	0x00000000	0x000000			00000	0x00000000
0x10010040	0x00000000	0x00000000	0x00000000	0x000000			00000	0x00000000
0x10010060	0x00000000	0x00000000	0x00000000	0x000000			00000	0x00000000
0x10010080	0x00000000	0x00000000	0x00000000	0x000000			00000	0x00000000
0x100100a0	0x00000000	0x00000000	0x00000000	0x000000			00000	0x00000000
0x100100c0	0x00000000	0x00000000	0x00000000	0x000000			00000	0x00000000
0x100100e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010100	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010120	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

0x10010000 (.data) ☑ Hexadecimal Addresses ☑ Hexadecimal Values □ ASCII

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

```
Oh my god. Someone's presed a button.
Oh my god. Someone's presed a button.
```

Assignment 3

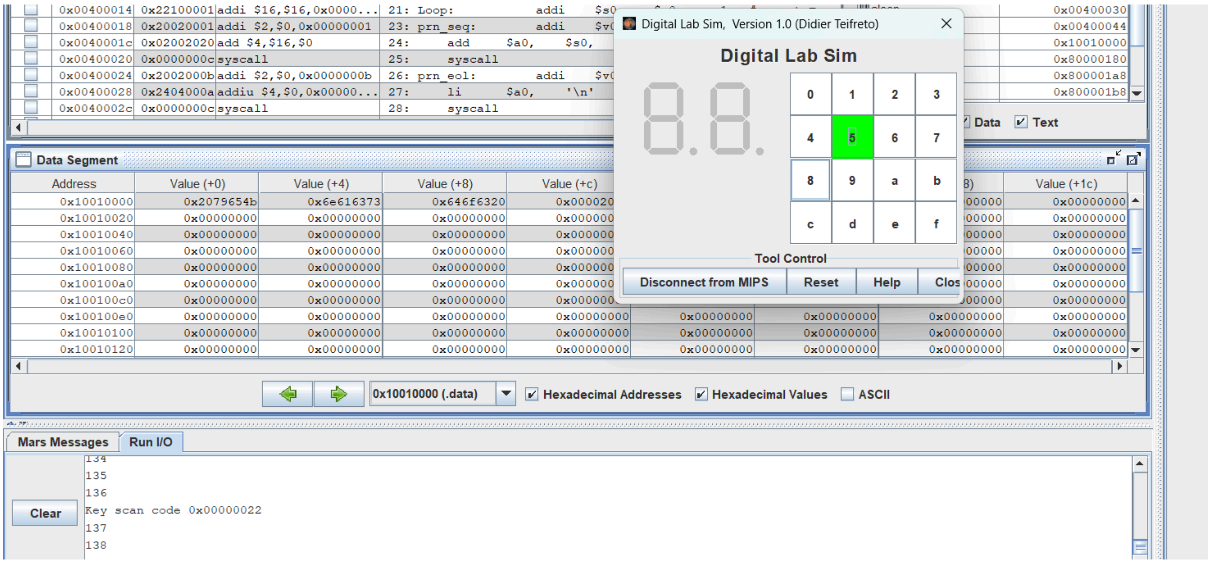
Code:

```
        .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 $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, $0, prn_cod #check

               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, $0, prn_cod #check

               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, $0, prn_cod #check

                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, $0, prn_cod #check



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      $at,    0($sp)          # Restore the registers from stack
   addi    $sp,    $sp,                     -4
return:         eret            # Return from exception
```

Result:



```
0x00400014 0x22100001 addi $16,$16,0x0000...  21: Loop:        addi    $s0
0x00400018 0x20020001 addi $2,$0,0x00000001   23: prn_seq:     addi    $v0
0x0040001c 0x02002020 add $4,$16,$0           24:     add     $a0,    $s0,
0x00400020 0x0000000c syscall                 25:     syscall
0x00400024 0x2002000b addi $2,$0,0x0000000b   26: prn_eol:     addi    $v0
0x00400028 0x2404000a addiu $4,$0,0x00000...  27:     li      $a0,    '\n'
0x0040002c 0x0000000c syscall                 28:     syscall
```

```
0x00400030
0x00400044
0x10010000
0x80000180
0x800001a8
0x800001b8
```

☑ Data ☑ Text

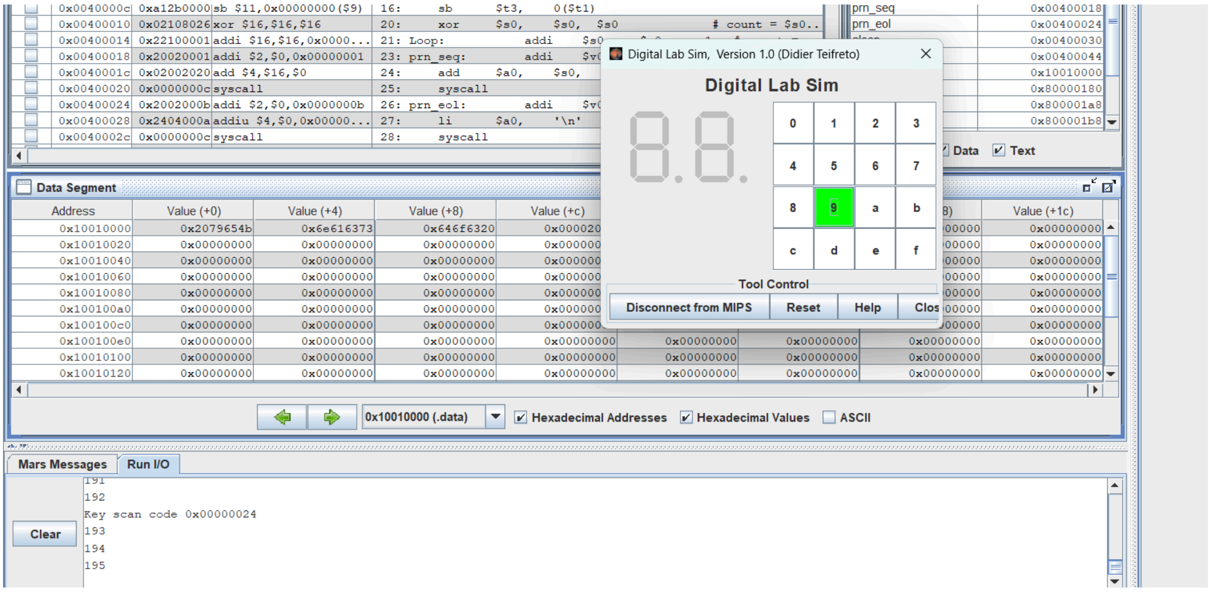
Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+1c)
0x10010000	0x2079654b	0x6e616373	0x646f6320	0x000020	0x00000000
0x10010020	0x00000000	0x00000000	0x00000000	0x000000	0x00000000
0x10010040	0x00000000	0x00000000	0x00000000	0x000000	0x00000000
0x10010060	0x00000000	0x00000000	0x00000000	0x000000	0x00000000
0x10010080	0x00000000	0x00000000	0x00000000	0x000000	0x00000000
0x100100a0	0x00000000	0x00000000	0x00000000	0x000000	0x00000000
0x100100c0	0x00000000	0x00000000	0x00000000	0x000000	0x00000000
0x100100e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010100	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010120	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

0x10010000 (.data) ☑ Hexadecimal Addresses ☑ Hexadecimal Values □ ASCII

Mars Messages | Run I/O

```
134
135
136
Key scan code 0x00000022
137
138
```



```
0x0040000c 0xa12b0000 sb $11,0x00000000($9)   16:     sb      $t3,    0($t1)
0x00400010 0x02108026 xor $16,$16,$16         20:     xor     $s0,    $s0,  $s0        # count = $s0..
0x00400014 0x22100001 addi $16,$16,0x0000...  21: Loop:        addi    $s0
0x00400018 0x20020001 addi $2,$0,0x00000001   23: prn_seq:     addi    $v0
0x0040001c 0x02002020 add $4,$16,$0           24:     add     $a0,    $s0,
0x00400020 0x0000000c syscall                 25:     syscall
0x00400024 0x2002000b addi $2,$0,0x0000000b   26: prn_eol:     addi    $v0
0x00400028 0x2404000a addiu $4,$0,0x00000...  27:     li      $a0,    '\n'
0x0040002c 0x0000000c syscall                 28:     syscall
```

```
prn_seq    0x00400018
prn_eol    0x00400024
           0x00400030
           0x00400044
           0x10010000
           0x80000180
           0x800001a8
           0x800001b8
```

☑ Data ☑ Text

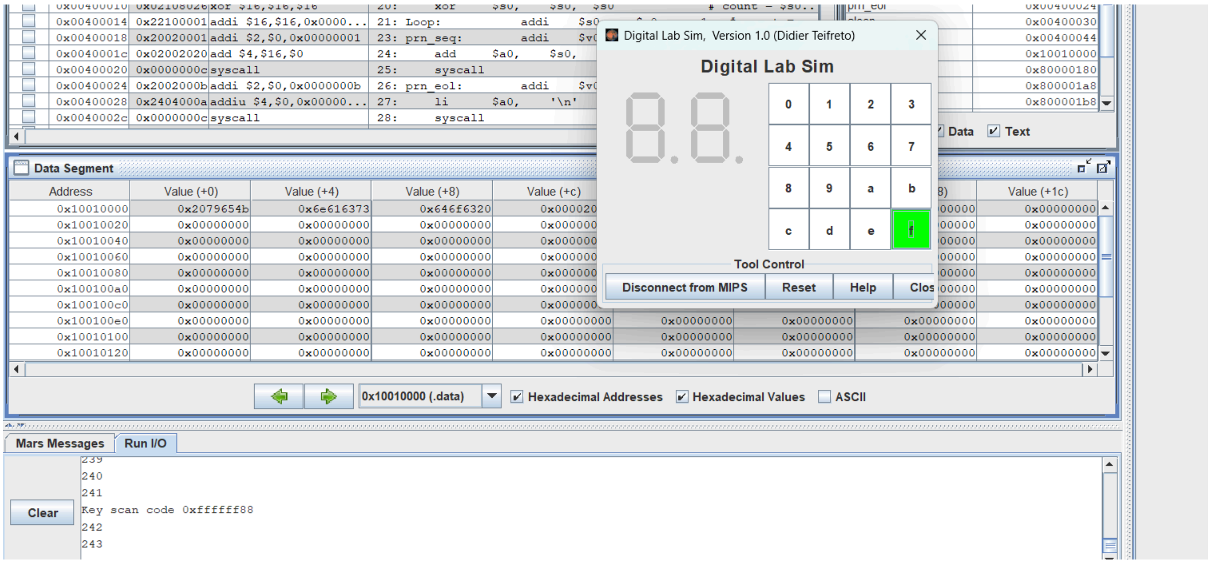
Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+1c)
0x10010000	0x2079654b	0x6e616373	0x646f6320	0x000020	0x00000000
0x10010020	0x00000000	0x00000000	0x00000000	0x000000	0x00000000
0x10010040	0x00000000	0x00000000	0x00000000	0x000000	0x00000000
0x10010060	0x00000000	0x00000000	0x00000000	0x000000	0x00000000
0x10010080	0x00000000	0x00000000	0x00000000	0x000000	0x00000000
0x100100a0	0x00000000	0x00000000	0x00000000	0x000000	0x00000000
0x100100c0	0x00000000	0x00000000	0x00000000	0x000000	0x00000000
0x100100e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010100	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010120	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

0x10010000 (.data) ☑ Hexadecimal Addresses ☑ Hexadecimal Values □ ASCII

Mars Messages | Run I/O

```
191
192
Key scan code 0x00000024
193
194
195
```



```
0x00400010 0x02108026 xor $16,$16,$16         20:     xor     $s0,    $s0,  $s0        # count = $s0..
0x00400014 0x22100001 addi $16,$16,0x0000...  21: Loop:        addi    $s0
0x00400018 0x20020001 addi $2,$0,0x00000001   23: prn_seq:     addi    $v0
0x0040001c 0x02002020 add $4,$16,$0           24:     add     $a0,    $s0,
0x00400020 0x0000000c syscall                 25:     syscall
0x00400024 0x2002000b addi $2,$0,0x0000000b   26: prn_eol:     addi    $v0
0x00400028 0x2404000a addiu $4,$0,0x00000...  27:     li      $a0,    '\n'
0x0040002c 0x0000000c syscall                 28:     syscall
```

```
prn_eol    0x00400024
           0x00400030
           0x00400044
           0x10010000
           0x80000180
           0x800001a8
           0x800001b8
```

☑ Data ☑ Text

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+1c)
0x10010000	0x2079654b	0x6e616373	0x646f6320	0x000020	0x00000000
0x10010020	0x00000000	0x00000000	0x00000000	0x000000	0x00000000
0x10010040	0x00000000	0x00000000	0x00000000	0x000000	0x00000000
0x10010060	0x00000000	0x00000000	0x00000000	0x000000	0x00000000
0x10010080	0x00000000	0x00000000	0x00000000	0x000000	0x00000000
0x100100a0	0x00000000	0x00000000	0x00000000	0x000000	0x00000000
0x100100c0	0x00000000	0x00000000	0x00000000	0x000000	0x00000000
0x100100e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010100	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010120	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

0x10010000 (.data) ☑ Hexadecimal Addresses ☑ Hexadecimal Values □ ASCII

Mars Messages | Run I/O

```
239
240
241
Key scan code 0xffffff88
242
243
```

Assignment 4
Code:

Result: