**Lab Report  
Lab 11: Interrupts & IO programming**

|  |  |
| --- | --- |
| Student ID | 20235890 |
| Student Name | Nguyễn Đức Anh |

**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

    space: .asciz " "

.text

main:

    li t1, IN\_ADDRESS\_HEXA\_KEYBOARD

    li t2, OUT\_ADDRESS\_HEXA\_KEYBOARD

    li t4, 0x08 # declare the maximum row

    init\_row\_1:

        li t3, 0x01 # init the row 1

    polling:

        sb t3, 0(t1) # must reassign expected row

        lb a0, 0(t2) # read scan code of key button

        bne a0, zero, print # if a0 = 0, jump to print

    next\_row:

        slli t3, t3, 1 # shift left 1 bit to get next row

    check\_row:

        bgt t3, t4, init\_row\_1 # check if the current row is greater than maximum row -> jump back to row 1

    print:

        li a7, 34 # print integer (hexa)

        ecall

    print\_space:

        la a0, space # print thye space

        li a7, 4

        ecall

    sleep:

        li a0, 100 # sleep 100ms

        li a7, 32

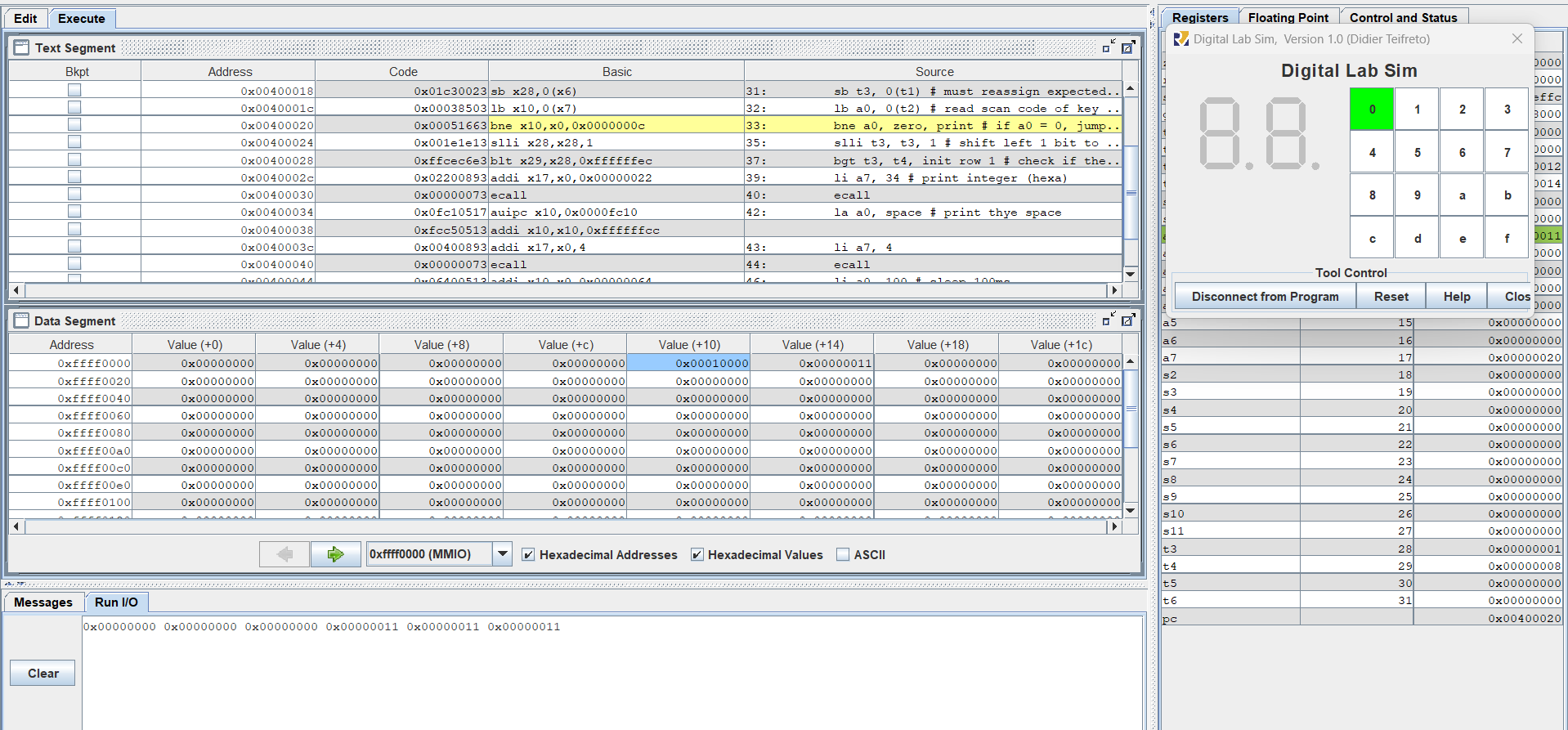
        ecall

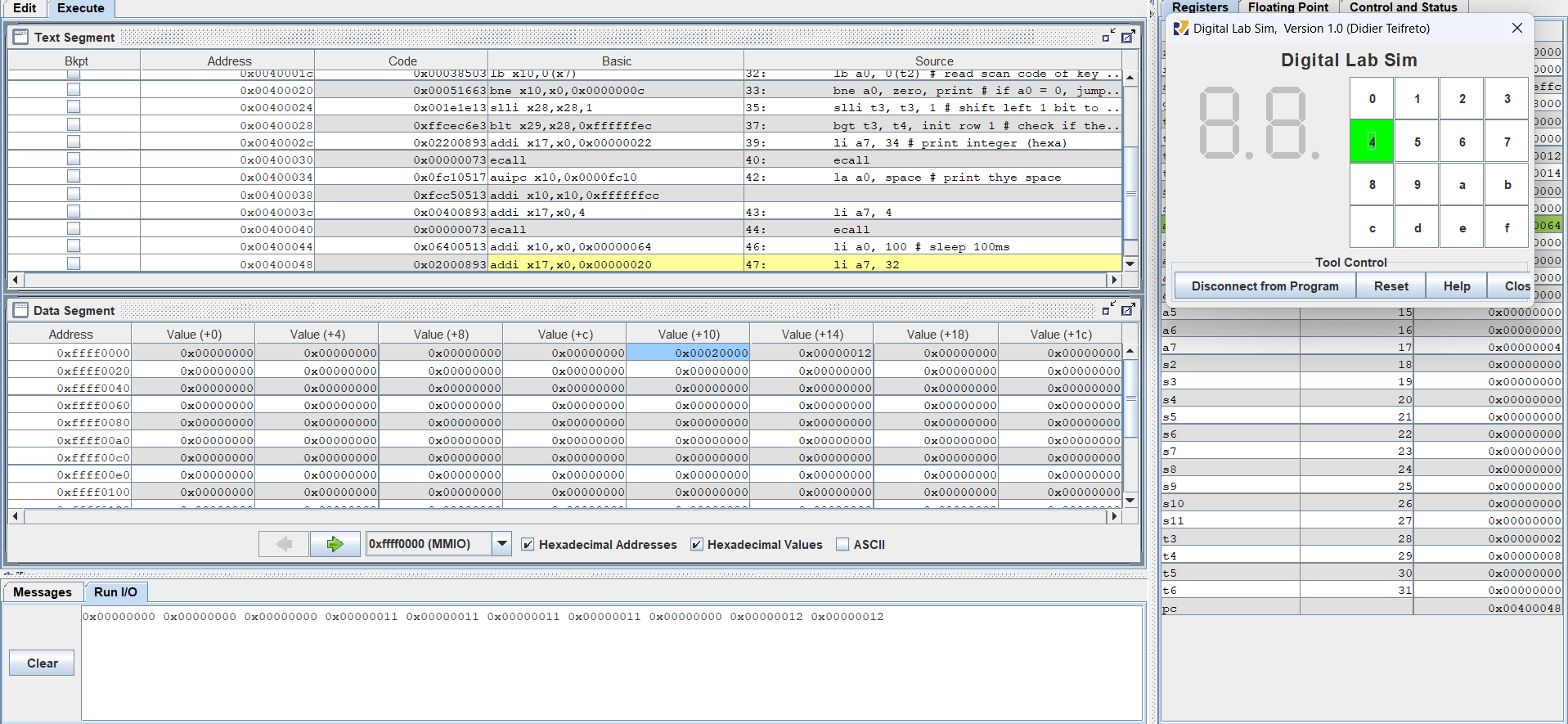
    back\_to\_polling:

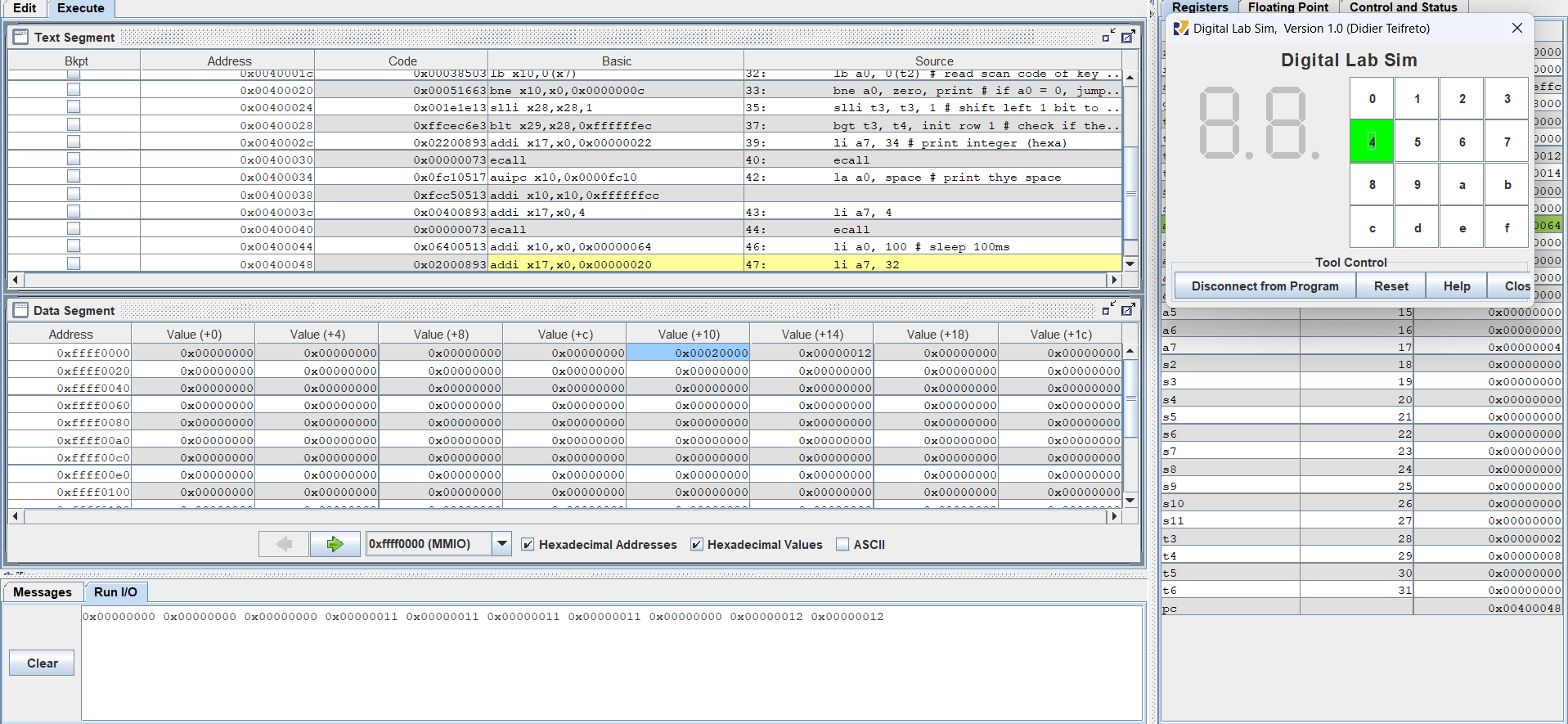
        j polling # continue polling

**Explaination:** The Algorithms is we shift left 1 bit to take the next value means next row and using branches condition to check the row not greater than 4 means 0x08.

**Result:**







A screenshot of a computer

Description automatically generated

**Assignment 2:**

**Code:**

.eqv IN\_ADDRESS\_HEXA\_KEYBOARD 0xFFFF0012

.eqv OUT\_ADDRESS\_HEXA\_KEYBOARD 0xFFFF0014

.data

    message: .asciz "Someone's presed a button.\n"

.text

main:

    # Load the interrupt service routine address to the UTVEC register

    la t0, handler

    csrrs zero, utvec, t0

    # Set the UEIE (User External Interrupt Enable) bit in UIE register

    li t1, 0x100

    csrrs zero, uie, t1 # uie - ueie bit (bit 8)

    # Set the UIE (User Interrupt Enable) bit in USTATUS register

    csrrsi zero, ustatus, 0x1 # ustatus - enable uie (bit 0)

    # Enable the interrupt of keypad of Digital Lab Sim

    li t1, IN\_ADDRESS\_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

        # Delay 10ms

        li a7, 32

        li a0, 10

        ecall

        nop

        j loop

end\_main:

# -----------------------------------------------------------------

# Interrupt service routine

# -----------------------------------------------------------------

handler:

    # ebreak # Can pause the execution to observe registers

    # Saves the context

    addi sp, sp, -8

    sw a0, 0(sp)

    sw a7, 4(sp)

    # Handles the interrupt

    # Shows message in Run I/O

    li a7, 4

    la a0, message

    ecall

    # Restores the context

    lw a7, 4(sp)

    lw a0, 0(sp)

    addi sp, sp, 8

    # Back to the main procedure

    uret

**Explaination:**

.eqv IN\_ADDRESS\_HEXA\_KEYBOARD 0xFFFF0012

.eqv OUT\_ADDRESS\_HEXA\_KEYBOARD 0xFFFF0014

.data

    message: .asciz "Someone's presed a button.\n"

* IN\_ADDRESS\_HEXA\_KEYBOARD and OUT\_ADDRESS\_HEXA\_KEYBOARD define specific memory-mapped addresses for input and output related to the keypad interrupt.
* “message” defines a string in memory to be displayed when an interrupt occurs.

1. Load the Handler Address:

main:

# Load the interrupt service routine address to the UTVEC register

    la t0, handler

    csrrs zero, utvec, t0

* la t0, handler loads the address of the interrupt service routine (handler) into t0.
* csrrs zero, utvec, t0 sets this address in the utvec register, which stores the user interrupt vector base address.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

1. Enable External User Interrupts:

# Set the UEIE (User External Interrupt Enable) bit in UIE register

    li t1, 0x100

    csrrs zero, uie, t1 # uie - ueie bit (bit 8)

* li t1, 0x100 sets the 8th bit (UEIE) of the uie register, enabling user-level external interrupts.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

1. Enable Global User Interrupts:

# Set the UIE (User Interrupt Enable) bit in USTATUS register

    csrrsi zero, ustatus, 0x1 # ustatus - enable uie (bit 0)

* csrrsi zero, ustatus, 0x1 sets the 0th bit (UIE) in the ustatus register, enabling the global interrupt system.

A screenshot of a computer

Description automatically generated

1. Enable Keypad Interrupt:

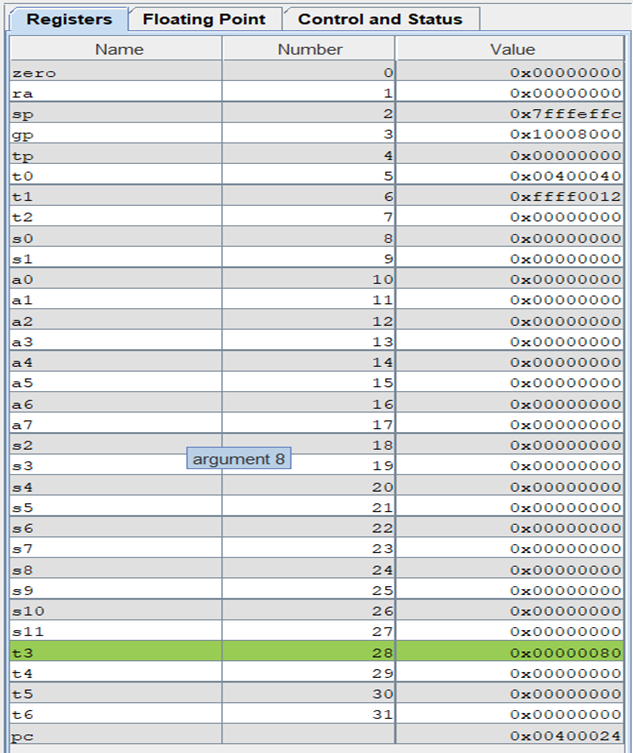
# Enable the interrupt of keypad of Digital Lab Sim

    li t1, IN\_ADDRESS\_HEXA\_KEYBOARD

    li t3, 0x80 # bit 7 = 1 to enable interrupt

    sb t3, 0(t1)

* The memory address IN\_ADDRESS\_HEXA\_KEYBOARD is loaded into t1.
* sb t3, 0(t1) stores 0x80 (bit 7 set) to enable keypad interrupts.



A screenshot of a computer

Description automatically generated

1. Main program loop:

loop:

        nop

        # Delay 10ms

        li a7, 32

        li a0, 10

        ecall

        nop

        j loop

end\_main:

* This loop keeps the program running indefinitely (nop instructions do nothing).
* li a7, 32 and li a0, 10 prepare for an environment call (ECALL) to delay for 10ms.
* ecall invokes the delay system call.

1. Interrupt Service Routine (ISR)

* Save the context:

# ebreak # Can pause the execution to observe registers

    # Saves the context

    addi sp, sp, -8

    sw a0, 0(sp)

    sw a7, 4(sp)

Saves a0 and a7 (used registers) onto the stack to preserve the program's state.

* Handle the Interrupt:

# Handles the interrupt

    # Shows message in Run I/O

    li a7, 4

    la a0, message

    ecall

li a7, 4 and la a0, message set up to display the string message.

ecall invokes a system call to output the message.

* Restore the Context:

# Restores the context

    lw a7, 4(sp)

    lw a0, 0(sp)

    addi sp, sp, 8

Restores a0 and a7 from the stack and adjusts the stack pointer.

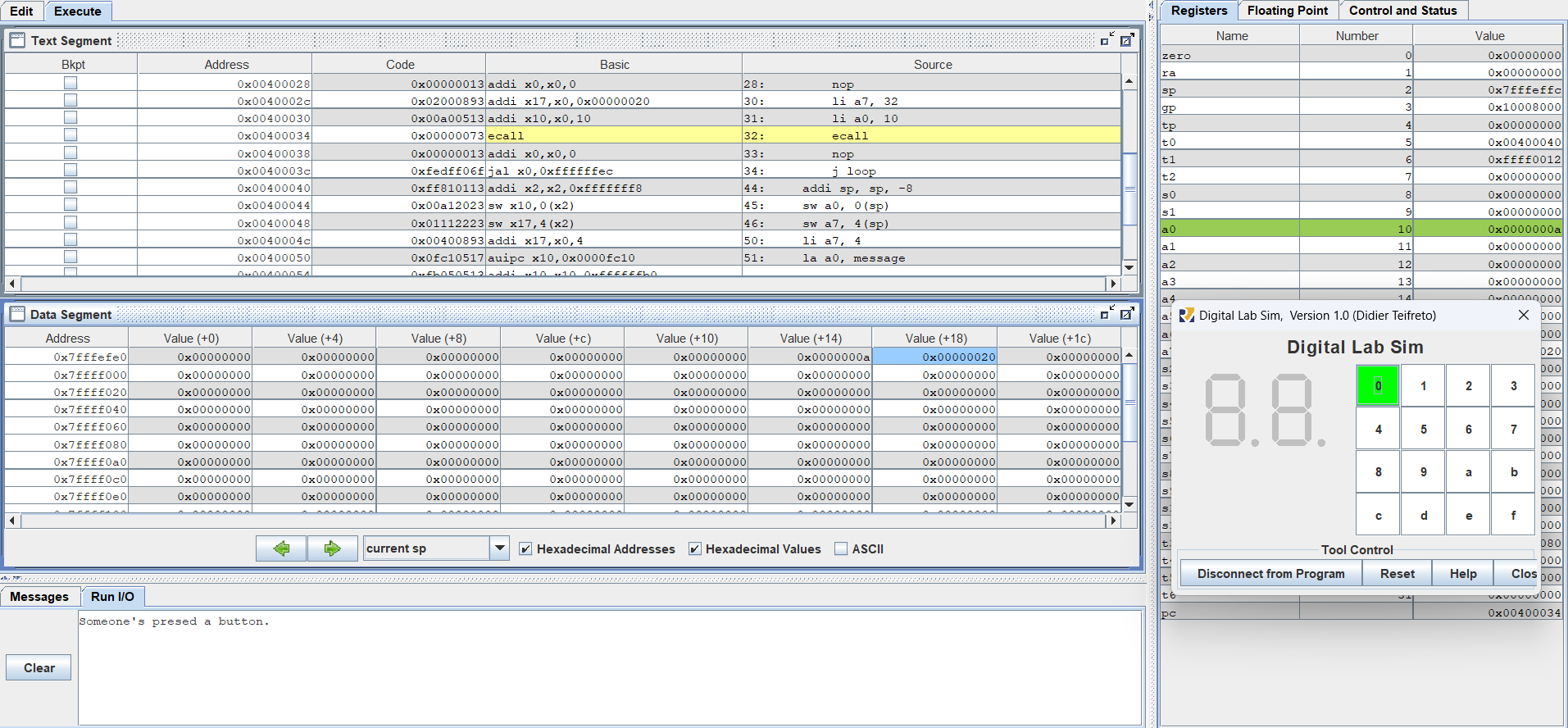
* Return from Interrupt

# Back to the main procedure

    Uret

uret resumes execution from the point where the interrupt occurred.

**Result:**



**Assignment 3:**

**Code:**

.eqv IN\_ADDRESS\_HEXA\_KEYBOARD 0xFFFF0012

.eqv OUT\_ADDRESS\_HEXA\_KEYBOARD 0xFFFF0014

.data

    message: .asciz "Key scan code: "

# -----------------------------------------------------------------

# MAIN Procedure

# -----------------------------------------------------------------

.text

main:

    # Load the interrupt service routine address to the UTVEC register

    la t0, handler

    csrrw zero, utvec, t0

    # Set the UEIE (User External Interrupt Enable) bit in UIE register

    li t1, 0x100

    csrrs zero, uie, t1  # Enable external interrupts in the user mode

    # Set the UIE (User Interrupt Enable) bit in USTATUS register

    csrrsi zero, ustatus, 0x1  # Enable interrupts globally

    # Enable the interrupt of keypad of Digital Lab Sim

    li t1, IN\_ADDRESS\_HEXA\_KEYBOARD

    li t3, 0x80  # Enable interrupt (bit 7)

    sb t3, 0(t1)

    # ---------------------------------------------------------

    # Loop to print a sequence of numbers

    # ---------------------------------------------------------

    xor s0, s0, s0  # count = s0 = 0

loop:

    addi s0, s0, 1  # Increment count

prn\_seq:

    addi a7, zero, 1

    add a0, s0, zero  # Print auto sequence number

    ecall

    addi a7, zero, 11

    li a0, '\n'  # Print EOL

    ecall

sleep:

    addi a7, zero, 32

    li a0, 300  # Sleep 300 ms

    ecall

    j loop  # Jump back to loop

end\_main:

# -----------------------------------------------------------------

# Interrupt Service Routine (Handler)

# -----------------------------------------------------------------

handler:

    # Save context

    addi sp, sp, -16

    sw a0, 0(sp)

    sw a7, 4(sp)

    sw t1, 8(sp)

    sw t2, 12(sp)

    # Print message

prn\_msg:

    addi a7, zero, 4

    la a0, message

    ecall

    # Scan all rows to detect pressed key

get\_key\_code:

    li t1, IN\_ADDRESS\_HEXA\_KEYBOARD

    li t2, 0x81

    sb t2, 0(t1)  # Enable row 1

    li t1, OUT\_ADDRESS\_HEXA\_KEYBOARD

    lb t3, 0(t1)

    bne t3, zero, prn\_key\_code  # If key pressed, go to print key

    li t1, IN\_ADDRESS\_HEXA\_KEYBOARD

    li t2, 0x82

    sb t2, 0(t1)  # Enable row 2

    li t1, OUT\_ADDRESS\_HEXA\_KEYBOARD

    lb t3, 0(t1)

    bne t3, zero, prn\_key\_code  # If key pressed, go to print key

    li t1, IN\_ADDRESS\_HEXA\_KEYBOARD

    li t2, 0x84

    sb t2, 0(t1)  # Enable row 3

    li t1, OUT\_ADDRESS\_HEXA\_KEYBOARD

    lb t3, 0(t1)

    bne t3, zero, prn\_key\_code  # If key pressed, go to print key

    li t1, IN\_ADDRESS\_HEXA\_KEYBOARD

    li t2, 0x88

    sb t2, 0(t1)  # Enable row 4

    li t1, OUT\_ADDRESS\_HEXA\_KEYBOARD

    lb t3, 0(t1)

    bne t3, zero, prn\_key\_code  # If key pressed, go to print key

    j end\_handler  # No key press, skip key printing

# Print the key code

prn\_key\_code:

    addi a7, zero, 34  # Print integer syscall

    add a0, t3, zero   # Key code to print

    ecall

    addi a7, zero, 11

    li a0, '\n'  # Print EOL

    ecall

# Restore context and return

end\_handler:

    lw t2, 12(sp)

    lw t1, 8(sp)

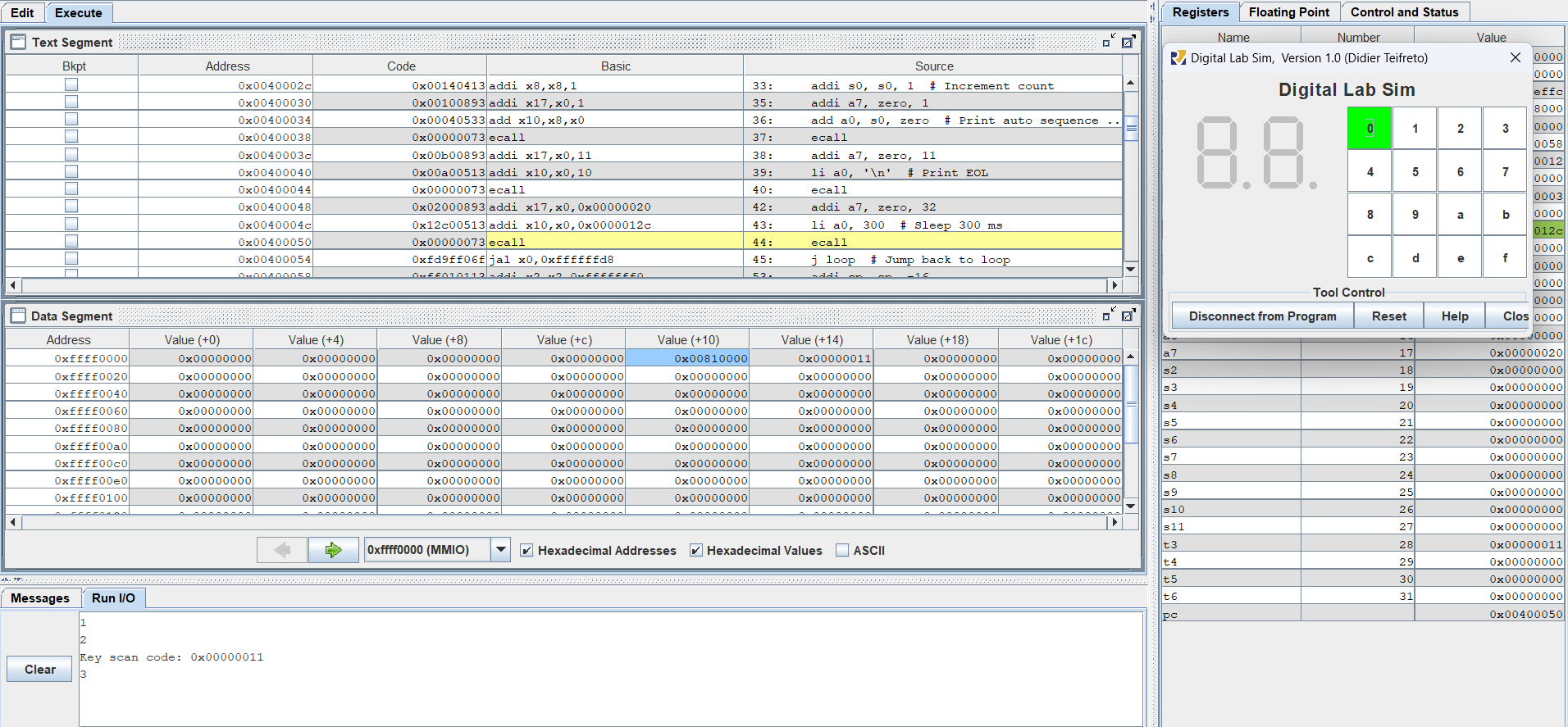
    lw a7, 4(sp)

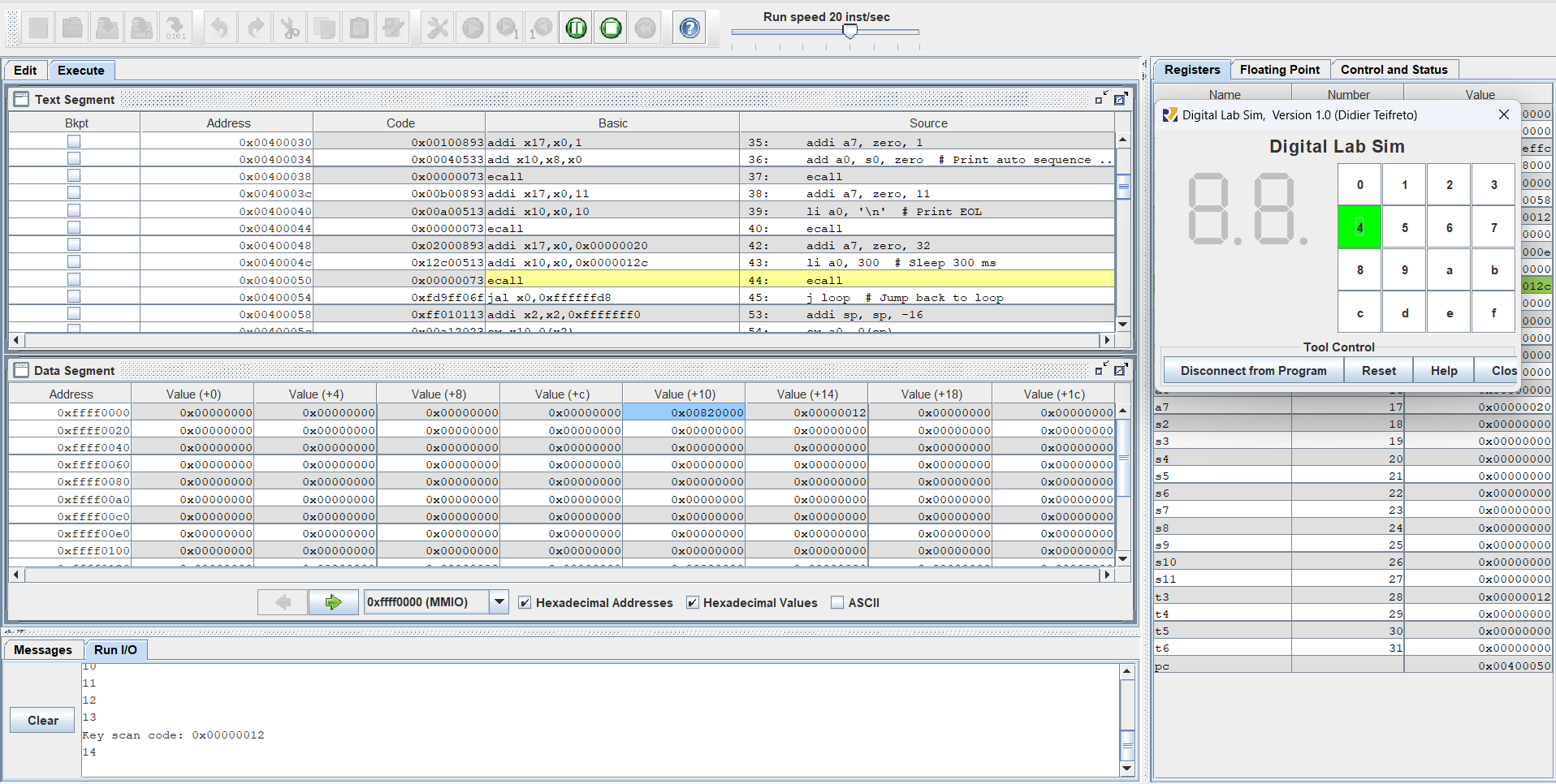
    lw a0, 0(sp)

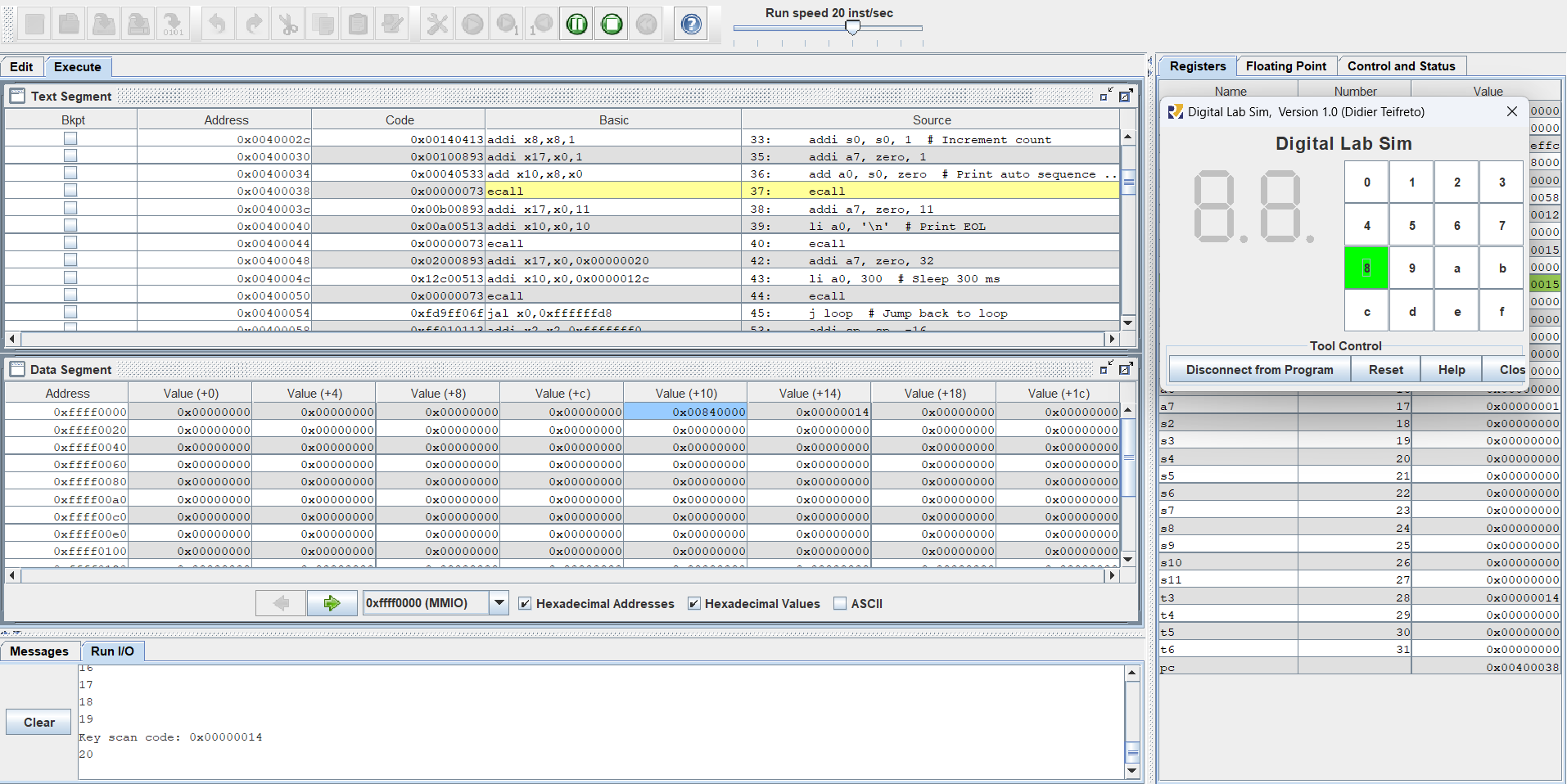
    addi sp, sp, 16

    uret  # Return from interrupt

**Result:**







A screenshot of a computer

Description automatically generated

**Assignment 4:**

**Code:**

.eqv IN\_ADDRESS\_HEXA\_KEYBOARD 0xFFFF0012

.eqv TIMER\_NOW 0xFFFF0018

.eqv TIMER\_CMP 0xFFFF0020

.eqv MASK\_CAUSE\_TIMER 4

.eqv MASK\_CAUSE\_KEYPAD 8

.data

    msg\_keypad: .asciz "Someone has pressed a key!\n"

    msg\_timer: .asciz "Time inteval!\n"

# -----------------------------------------------------------------

# MAIN Procedure

# -----------------------------------------------------------------

.text

main:

    # Load the interrupt service routine address to the UTVEC register

    la t0, handler

    csrrs zero, 5, t0

    # Set the UEIE (User External Interrupt Enable) bit in UIE register

    li t1, 0x100

    csrrs zero, 4, t1 # uie - ueie bit (bit 8) - external interrupt

    csrrsi zero, 4, 0x10 # uie - utie bit (bit 4) - timer interrupt

    # Set the UIE (User Interrupt Enable) bit in USTATUS register

    csrrsi zero, 0, 0x1 # ustatus - enable uie - global interrupt

# ---------------------------------------------------------

# Enable interrupts you expect

# ---------------------------------------------------------

    # Enable the interrupt of keypad of Digital Lab Sim

    li t1, IN\_ADDRESS\_HEXA\_KEYBOARD

    li t2, 0x80 # bit 7 of = 1 to enable interrupt

    sb t2, 0(t1)

    # Enable the timer interrupt

    li t1, TIMER\_CMP

    li t2, 1000

    sw t2, 0(t1)

# ---------------------------------------------------------

# No-end loop, main program, to demo the effective of interrupt

# ---------------------------------------------------------

    loop:

        nop

        li a7, 32

        li a0, 10

        ecall

        nop

        j loop

end\_main:

# -----------------------------------------------------------------

# Interrupt service routine

# -----------------------------------------------------------------

handler:

    # Saves the context

    addi sp, sp, -16

    sw a0, 0(sp)

    sw a1, 4(sp)

    sw a2, 8(sp)

    sw a7, 12(sp)

    # Handles the interrupt

    csrr a1, ucause

    li a2, 0x7FFFFFFF

    and a1, a1, a2 # Clear interrupt bit to get the value

    li a2, MASK\_CAUSE\_TIMER

    beq a1, a2, timer\_isr

    li a2, MASK\_CAUSE\_KEYPAD

    beq a1, a2, keypad\_isr

    j end\_process

    timer\_isr:

        li a7, 4

        la a0, msg\_timer

        ecall

        # Set cmp to time + 1000

        li a0, TIMER\_NOW

        lw a1, 0(a0)

        addi a1, a1, 1000

        li a0, TIMER\_CMP

        sw a1, 0(a0)

        j end\_process

    keypad\_isr:

        li a7, 4

        la a0, msg\_keypad

        ecall

        j end\_process

    end\_process:

        # Restores the context

        lw a7, 12(sp)

        lw a2, 8(sp)

        lw a1, 4(sp)

        lw a0, 0(sp)

        addi sp, sp, 16

        uret

**Explaination:**

.eqv IN\_ADDRESS\_HEXA\_KEYBOARD 0xFFFF0012

.eqv TIMER\_NOW 0xFFFF0018

.eqv TIMER\_CMP 0xFFFF0020

.eqv MASK\_CAUSE\_TIMER 4

.eqv MASK\_CAUSE\_KEYPAD 8

.data

    msg\_keypad: .asciz "Someone has pressed a key!\n"

    msg\_timer: .asciz "Time inteval!\n"

* IN\_ADDRESS\_HEXA\_KEYBOARD (0xFFFF0012): Address for the hexadecimal keyboard interrupt.
* TIMER\_NOW (0xFFFF0018): Address of the timer's current value.
* TIMER\_CMP (0xFFFF0020): Address to set the timer comparison value.
* MASK\_CAUSE\_TIMER (4): Mask to identify timer interrupt cause.
* MASK\_CAUSE\_KEYPAD (8): Mask to identify keypad interrupt cause.

Stores two message strings for interrupts:

* msg\_keypad: "Someone has pressed a key!\n"
* msg\_timer: "Time interval!\n"

**Main Procedure**

1. Setup for Interrupt Handling:

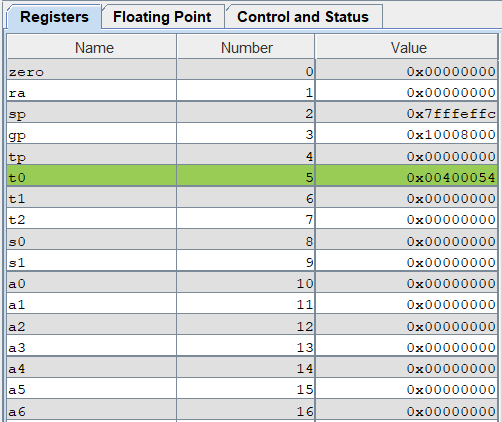
main:

# Load the interrupt service routine address to the UTVEC register

    la t0, handler

    csrrs zero, 5, t0

* la t0, handler: Load the address of the handler (interrupt service routine) into register t0.
* csrrs zero, 5, t0: Store the address in the utvec CSR (control and status register), which sets the interrupt handler.



A screenshot of a computer

Description automatically generated

1. Enable External and Timer Interrupts:

# Set the UEIE (User External Interrupt Enable) bit in UIE register

    li t1, 0x100

    csrrs zero, 4, t1 # uie - ueie bit (bit 8) - external interrupt

    csrrsi zero, 4, 0x10 # uie - utie bit (bit 4) - timer interrupt

* li t1, 0x100: Load 0x100 (bit 8) to enable external interrupts.
* csrrs zero, 4, t1: Set bit 8 in UIE register (User Interrupt Enable).
* csrrsi zero, 4, 0x10: Set bit 4 in UIE register to enable timer interrupts.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

1. Enable Global Interrupts:

# Set the UIE (User Interrupt Enable) bit in USTATUS register

    csrrsi zero, 0, 0x1 # ustatus - enable uie - global interrupt

* csrrsi zero, 0, 0x1: Enable global interrupts by setting the UIE bit in the ustatus register.

A screenshot of a computer

Description automatically generated

1. Enable Keypad and Timer Interrupts:

# Enable the interrupt of keypad of Digital Lab Sim

    li t1, IN\_ADDRESS\_HEXA\_KEYBOARD

    li t2, 0x80 # bit 7 of = 1 to enable interrupt

    sb t2, 0(t1)

* li t1, IN\_ADDRESS\_HEXA\_KEYBOARD: Load the address of the keypad interrupt control register.
* li t2, 0x80: Set bit 7 to enable keypad interrupts.
* sb t2, 0(t1): Store the value 0x80 in the memory location, enabling the keypad interrupt.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

# Enable the timer interrupt

    li t1, TIMER\_CMP

    li t2, 1000

    sw t2, 0(t1)

* li t1, TIMER\_CMP: Load the address of the timer comparison register.
* li t2, 1000: Set the timer comparison value to 1000.
* sw t2, 0(t1): Store the value 1000 to start the timer interrupt.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

1. Main Program loop:

loop:

        nop

        li a7, 32

        li a0, 10

        ecall

        nop

        j loop

end\_main:

* This loop keeps the program running indefinitely (nop instructions do nothing).
* li a7, 32 and li a0, 10 prepare for an environment call (ECALL) to delay for 10ms.
* ecall invokes the delay system call.

**Interrupt Service Routine**

1. Save the Context:

handler:

    # Saves the context

    addi sp, sp, -16

    sw a0, 0(sp)

    sw a1, 4(sp)

    sw a2, 8(sp)

    sw a7, 12(sp)

* Save the registers a0, a1, a2, and a7 onto the stack to preserve the context during the interrupt.

1. Handle Timer and Keypad Interrupts:

# Handles the interrupt

    csrr a1, ucause

    li a2, 0x7FFFFFFF

    and a1, a1, a2 # Clear interrupt bit to get the value

* csrr a1, ucause: Load the cause of the interrupt into a1.
* and a1, a1, a2: Mask out the interrupt bit to isolate the interrupt cause.

1. Timer Interrupt Handling:

li a2, MASK\_CAUSE\_TIMER

    beq a1, a2, timer\_isr

* Check if the interrupt cause matches the timer interrupt mask (4).
* If yes, branch to timer\_isr.

timer\_isr:

        li a7, 4

        la a0, msg\_timer

        ecall

        # Set cmp to time + 1000

        li a0, TIMER\_NOW

        lw a1, 0(a0)

        addi a1, a1, 1000

        li a0, TIMER\_CMP

        sw a1, 0(a0)

        j end\_process

* Prints the message "Time interval!" to indicate that a timer interrupt occurred.
* Schedules the next timer interrupt after a delay of 1000 time units by updating the TIMER\_CMP register.
* Returns to the main program after completing the interrupt service routine.

1. Keypad Interrupt Handling:

li a2, MASK\_CAUSE\_KEYPAD

    beq a1, a2, keypad\_isr

* Check if the interrupt cause matches the keypad interrupt mask (8).
* If yes, branch to keypad\_isr.

keypad\_isr:

        li a7, 4

        la a0, msg\_keypad

        ecall

        j end\_process

* Print the keypad message.

1. Restore Context and Return:

end\_process:

        # Restores the context

        lw a7, 12(sp)

        lw a2, 8(sp)

        lw a1, 4(sp)

        lw a0, 0(sp)

        addi sp, sp, 16

        uret

* Restore saved registers from the stack.
* uret: Return from the interrupt.

**Result:**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**