BÁO CÁO THỨC HÀNH KIẾN TRÚC MÁY TÍNH TUẦN 11.5

Họ và tên: Phan Khánh Vũ

MSSV: 20235880

1. **Assignment 4:**

- **Nhập chương trình:**

```
# 0
.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:
   la      t0, handler
   csrrs   zero, utvec, t0
```

```
    li      t1, 0x100
    csrrs   zero, uie, t1     # uie - ueie bit (bit 8) - external interrupt
    csrrsi  zero, uie, 0x10   # uie - utie bit (bit 4) - timer interrupt

    csrrsi  zero, ustatus, 1    # 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
# 2
   sw      t2, 0(t1)

    # ------------------------------------------------------
    # No-end loop, main program, to demo the effective of interrupt
    # ------------------------------------------------------
# 10
```

```
loop:
   nop
   nop
   nop
   j   loop
end_main:


# ---------------------------------------------------------------
# Interrupt service routine
# ---------------------------------------------------------------
# 3
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
  # 4
    j       end_process
  # 5
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)
    # 6
    j       end_process
  # 7
keypad_isr:
    li      a7, 4
    la      a0, msg_keypad
```

```
    ecall
# 8
    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
# 9
    uret
```

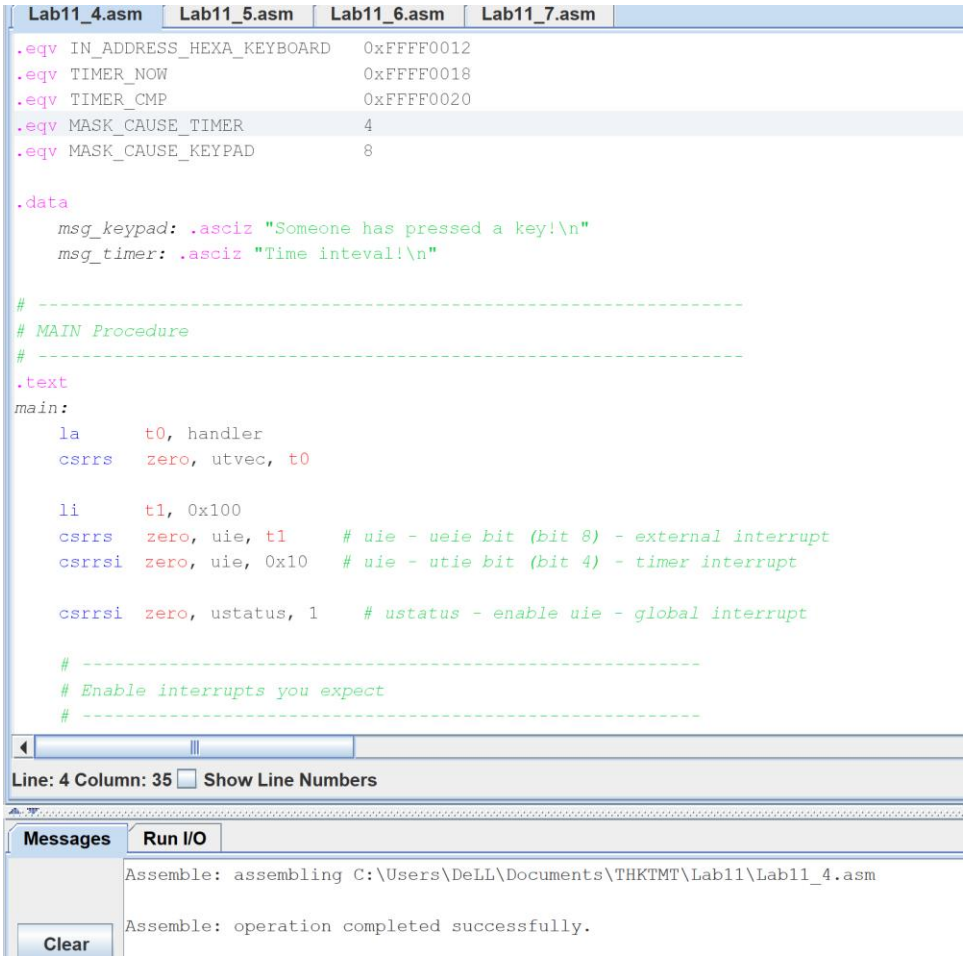
- **Các bước hoạt động của chương trình:**

- Trước tiên, chương trình sẽ khởi tạo các hằng số cần thiết để sử dụng.
- Tiếp theo, khởi tạo chuỗi nhằm hiển thị thông báo dữ liệu ra màn hình.
- Khi bắt đầu chương trình, các ngắt timer và ngắt bàn phím sẽ được thiết lập, sau đó kích hoạt hệ thống ngắt. Khi có ngắt xảy ra, chương trình sẽ kiểm tra để xác định đó là ngắt từ timer hay từ bàn phím, rồi xử lý tương ứng và hiển thị thông báo ra màn hình. Sau khi xử lý xong, ngắt sẽ được khôi phục và chương trình tiếp tục hoạt động.
- Vòng lặp được sử dụng để đảm bảo chương trình không bị treo trong quá trình chạy.

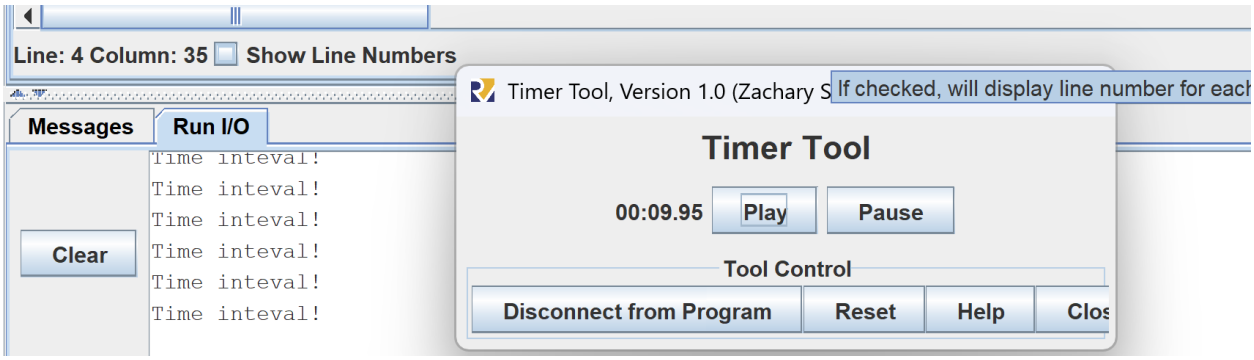
- **Giá trị của các thanh ghi:**

No	STT	utvec	ustatus	uie	ucause	sp	pc	Chú thích
1	0	0x0000	0x0000	0x0000	0x0000	0x7fffeffc	0x400000	Bắt đầu chương trình
2	1	0x0000	0x0000	0x0000	0x0000	0x7fffeffc	0x400000	Khởi tạo hằng số và chuỗi
3	2	0x40004c	0x0001	0x0110	0x0000	0x7fffeffc	0x40003c	Khởi tạo ngắt và bật ngắt
4	3	0x40004c	0x0001	0x0110	0x80000004	0x7fffefec	0x40004c	Phát hiện ngắt
5	5	0x40004c	0x0010	0x0110	0x80000004	0x7fffefdc	0x400084	Lưu dữ liệu và kiểm tra loại ngắt
6	6	0x40004c	0x0010	0x0110	0x80000004	0x7fffefdc	0x400084	Xử lý ngắt timer
7	7	0x40004c	0x0001	0x0110	0x80000004	0x7fffeffc	0x400048	Bật lại ngắt thoát xử lý ngắt
8	8	0x40004c	0x0001	0x0110	0x80000004	0x7fffeffc	0x400048	Quay lại vòng lặp

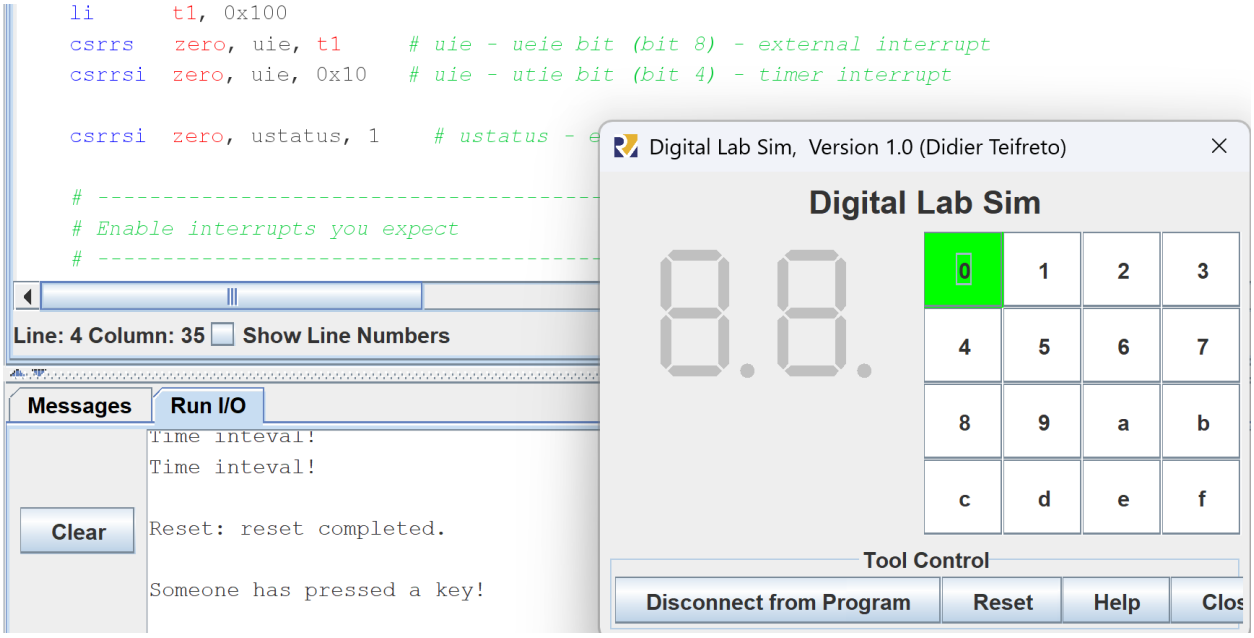
- **Chạy chương trình:**



- **Kết quả chạy chương trình:**
- Xử lý khi có ngắt Timer:



- Xử lý ngắt khi có phím bấm:



Kết quả đúng kỳ vọng

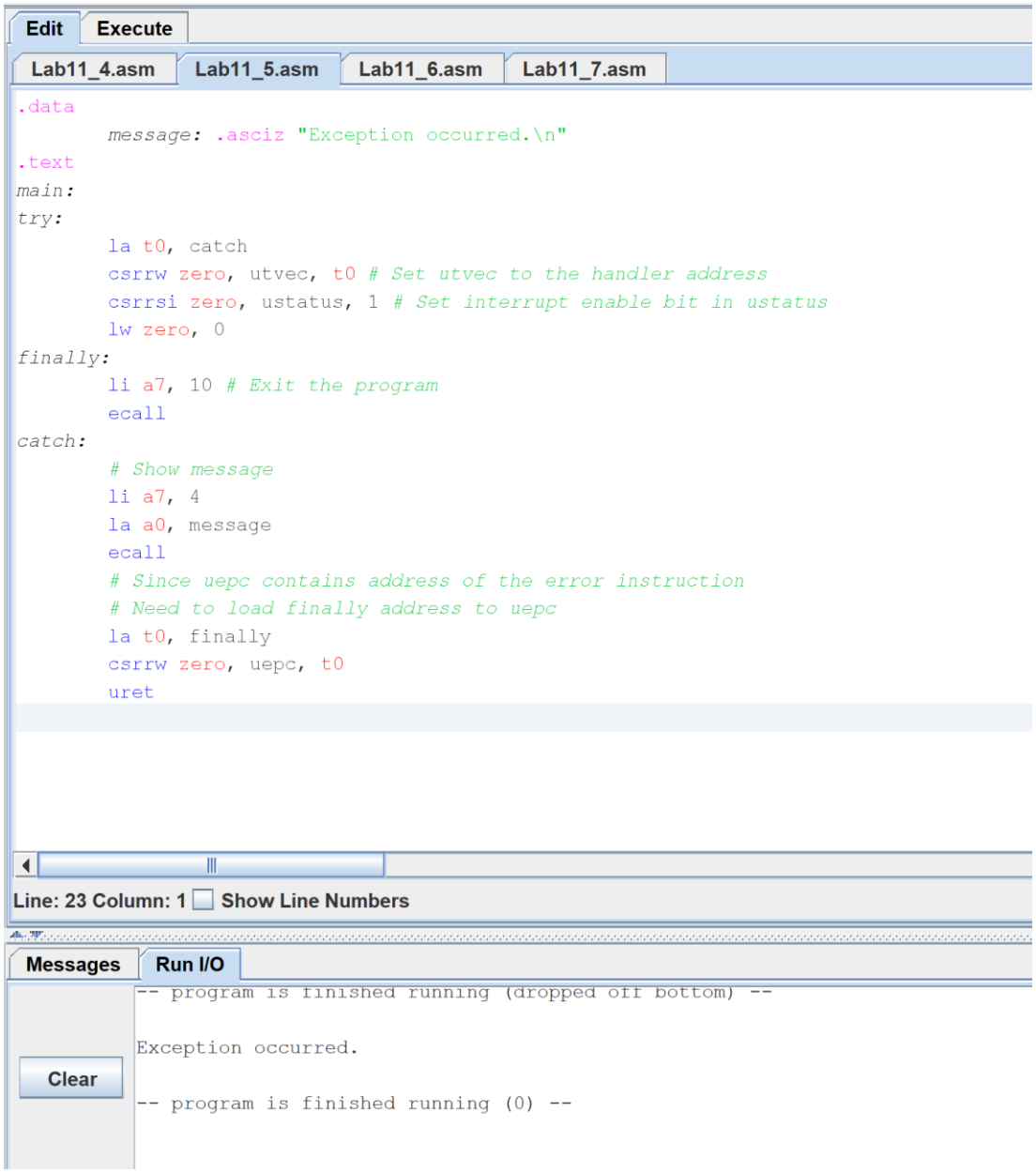
2. **Assignment 5:**
- **Nhập chương trình:**

```
# 0
.data
        message: .asciz "Exception occurred.\n"
```

```
.text
main:
try:
        la t0, catch
        csrrw zero, utvec, t0 # Set utvec to the handler address
# 1
        csrrsi zero, ustatus, 1 # Set interrupt enable bit in ustatus
        lw zero, 0
finally:
        li a7, 10 # Exit the program
# 2
ecall
# 3
catch:
        # Show message
        li a7, 4
        la a0, message
        ecall
        # Since uepc contains address of the error instruction
        # Need to load finally address to uepc
        la t0, finally
        csrrw zero, uepc, t0
# 4
uret
```

- **Các bước thực hiện của chương trình:**

- Khởi tạo chuỗi ký tự dùng để hiển thị kết quả lên màn hình.
- Khi chương trình bắt đầu, thực hiện thiết lập và kích hoạt các ngắt. Nếu xảy ra lỗi, chương trình sẽ hiển thị thông báo lỗi ra màn hình, sau đó chuẩn bị quay về phần finally, và thực thi finally để kết thúc chương trình.
- **Chạy chương trình:**



- **Giá trị các thanh ghi:**

No	STT	ustatus	Utvec	uepc	Pc	Chú thích
1	0	0x00	0x00	0x00	0x400000	Bắt đầu chương trình
2	1	0x01	0x40001c	0x00	0x400010	Khởi tạo ngắt
3	3	0x10	0x40001c	0x400010	0x40001c	Phát hiện trường hợp ngoại lệ
4	4	0x01	0x40001c	0x400014	0x400014	Xử lý ngoại lệ và quay lại chương trình
5	2	0x01	0x40001c	0x400014	0x40001c	Kết thúc chương trình

3. Assignment 6:

- **Nhập chương trình:**

```
.data
        overflow_msg: .asciz "overflow detected. Program terminated.\n"
        overflow_header: .asciz "User input is out of range\n"
        sum_msg: .asciz "Sum of the 2 numbers:\n"
.text
main:
        # init
        li a7, 5
        ecall
        mv s1, a0

        li a7, 5
        ecall
        mv s2, a0
```

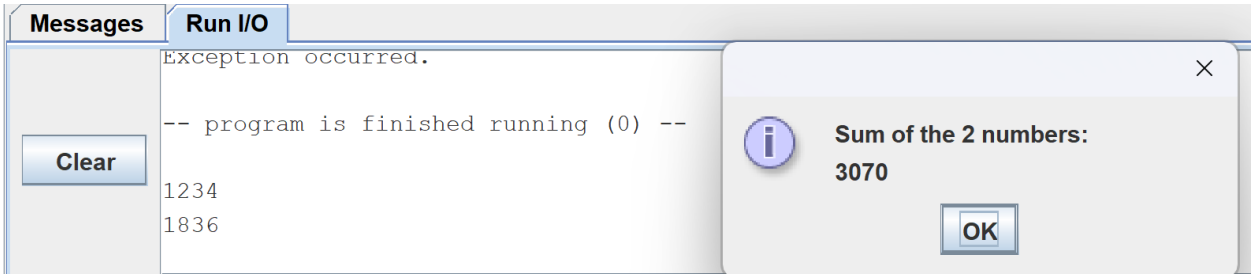
```
        li t0, 0            # no overflow initially

        # Setup interrupt handler
        la t1, handler
        csrrw zero, utvec, t1
        csrrsi zero, ustatus, 1   # enable global interrupt
        csrrsi zero, uie, 0x1     # enable software interrupt

        # Add two numbers
        add s3, s1, s2

        # Check overflow
        xor t1, s1, s2
        blt t1, zero, print       # different signs ? no overflow
        blt s1, zero, neg
        bge s3, s1, print         # normal addition ? no overflow

        j overflow

neg:
        bge s1, s3, print         # normal subtraction ? no overflow

overflow:
        li t0, 1                  # overflow detected
```

```
        # trigger interrupt
        csrr t1, uip
        ori t1, t1, 0x1          # set USIP (bit 0)
        csrrw zero, uip, t1

print:
        mv a1, s3
        la a0, sum_msg
        li a7, 56
        ecall
        li a7, 10               # print program
        ecall

# ------------------------------------------------------
# Interrupt service routine
# ------------------------------------------------------
handler:
        addi sp, sp, -8
        sw ra, 0(sp)
        sw a0, 4(sp)



        la a0, overflow_header
```

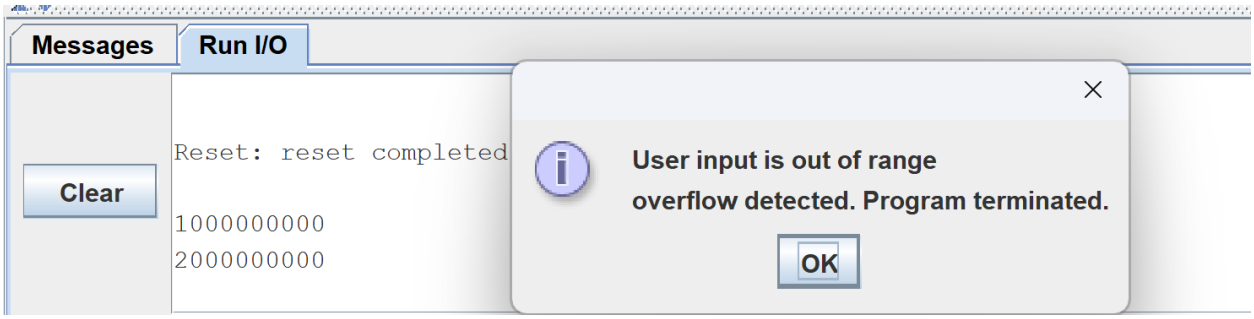
```
la a1, overflow_msg
li a7, 59
ecall

li a7, 10
ecall

lw a0, 4(sp)
lw ra, 0(sp)
addi sp, sp, 8

uret
```

- **Các bước hoạt động của chương trình:**
- Trước tiên, khởi tạo các chuỗi ký tự để sử dụng trong việc hiển thị kết quả lên màn hình.
- Khi chương trình bắt đầu, người dùng nhập vào hai số nguyên từ bàn phím.
- Sau đó, chương trình thiết lập và kích hoạt hệ thống ngắt, đồng thời thực hiện phép cộng hai số và kiểm tra xem có xảy ra tràn số hay không.
- Nếu xảy ra tràn số, chương trình sẽ đặt giá trị USIP bằng 1 để kích hoạt ngắt mềm.
- Tiếp theo, xử lý ngắt mềm, hiển thị kết quả ra màn hình, rồi bật lại ngắt.
- Trong trường hợp không có tràn số, kết quả sẽ được in trực tiếp lên màn hình.

- **Kết quả của chương trình:**
  + Với kết quả tổng của 2 số không tràn số



+ Với kết quả tổng của 2 số bị tràn



Kết quả đúng kỳ vọng

4. Assignment EX (Bài tập bổ xung):

- **Nhập chương trình:**

```
.eqv   IN_ADDRESS_HEXA_KEYBOARD   0xFFFF0012
.eqv    OUT_ADDRESS_HEXA_KEYBOARD  0xFFFF0014
.eqv   TIMER_NOW              0xFFFF0018
.eqv   TIMER_CMP              0xFFFF0020
.eqv   SEVENSEG_LEFT          0xFFFF0011
.eqv   SEVENSEG_RIGHT         0xFFFF0010
.eqv   MASK_CAUSE_TIMER       0x00000004
.eqv   MASK_CAUSE_KEYPAD      0x00000008
.eqv    newline               0xa
.data
      count:        .word 0       # Giá trị đếm hiện tại (0-99)
```

```
    direction:      .word 1         # 1 = tăng, -1 = giảm
    period:         .word 1000      # Chu kỳ mặc định (1000ms)
    digit_patterns: .byte 0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07, 0x7F, 0x6F  # Mẫu các số 0-9

.text
main:
    # Thiết lập trình xử lý ngắt
    la      t0, handler
    csrrs   zero, utvec, t0

    # Kích hoạt ngắt (ngoại vi và timer)
    li      t1, 0x100
    csrrs   zero, uie, t1     # Kích hoạt ngắt ngoại vi
    csrrsi  zero, uie, 0x10   # Kích hoạt ngắt timer

    csrrsi  zero, ustatus, 1  # Kích hoạt ngắt toàn cục

    # Kích hoạt ngắt bàn phím
    li      t1, IN_ADDRESS_HEXA_KEYBOARD
    li      t2, 0x80          # Kích hoạt ngắt
    sb      t2, 0(t1)

    # Thiết lập giá trị so sánh timer ban đầu
    li      t1, TIMER_NOW
    lw      t2, 0(t1)
    lw      t3, period
```

```
    add     t2, t2, t3
    li      t1, TIMER_CMP
    sw      t2, 0(t1)

    # Hiển thị ban đầu
    jal     update_display

    # Vòng lặp chính - chờ ngắt
loop:
    nop
    j       loop
end_main:

# ----------------------------------------------------------------
# Trình xử lý ngắt
# ----------------------------------------------------------------
handler:
    # Lưu ngữ cảnh
    addi    sp, sp, -16
    sw      a0, 0(sp)
    sw      a1, 4(sp)
    sw      a2, 8(sp)
    sw      a7, 12(sp)

    # Kiểm tra nguyên nhân ngắt
    csrr    a1, ucause
```

```
    li      a2, 0x7FFFFFFF
    and     a1, a1, a2      # Xóa bit ngắt

    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:
    # Cập nhật bộ đếm theo hướng
    lw      a0, count
    lw      a1, direction
    add     a0, a0, a1

    # Xử lý vòng lặp (99->00 hoặc 00->99)
    li      a2, 100
    bge     a0, a2, wrap_around_high
    bltz    a0, wrap_around_low
    j       store_count

wrap_around_high:
    li      a0, 0
    j       store_count

wrap_around_low:
```

```
    li      a0, 99

store_count:
    sw      a0, count, t0   # Lưu giá trị mới vào count

    # Cập nhật hiển thị
    jal     update_display   # Gọi hàm cập nhật hiển thị

    # Đặt lại timer cho khoảng tiếp theo
    li      a0, TIMER_NOW
    lw      a1, 0(a0)
    lw      a2, period
    add     a1, a1, a2
    li      a0, TIMER_CMP
    sw      a1, 0(a0)
    j       end_process

keypad_isr:
    # Quét bàn phím từng hàng để kiểm tra phím nào được nhấn
    li      t1, IN_ADDRESS_HEXA_KEYBOARD
    li      t2, OUT_ADDRESS_HEXA_KEYBOARD
    li      t3, 0x01              # Bắt đầu từ hàng 1 (00000001)
    li      a1, 0                 # a1 sẽ lưu mã phím nếu có

row_loop:
    sb      t3, 0(t1)             # Gửi tín hiệu chọn hàng bàn phím
```

```
    lb      a0, 0(t2)              # Đọc mã phím (scan code)
    beq     a0, zero, next_row     # Nếu không có phím, chuyển hàng tiếp theo
    add     a1, a0, zero           # Lưu mã phím vào a1
    j       check_key              # Nhảy đến xử lý mã phím

next_row:
    slli    t3, t3, 1              # Chuyển sang hàng tiếp theo (dịch trái 1 bit)
    li      t4, 0x10               # Giới hạn là hàng 4 (10000)
    blt     t3, t4, row_loop       # Nếu chưa hết hàng, lặp lại

    # Không phát hiện phím nào, bật lại ngắt keypad và kết thúc
    jal     enable_keypad_interrupt
    j       end_process

check_key:
    # Kiểm tra mã phím để xác định chức năng
    li      a2, 0x11               # Phím 0
    beq     a1, a2, set_increment
    li      a2, 0x21               # Phím 1
    beq     a1, a2, set_decrement
    li      a2, 0x12               # Phím 4
    beq     a1, a2, decrease_period
    li      a2, 0x22               # Phím 5
    beq     a1, a2, increase_period

    # Phím không hợp lệ, vẫn bật lại ngắt keypad
```

```
        jal     enable_keypad_interrupt
        j       end_process

set_increment:
        li      a0, 1
        la      t4, direction
        sw      a0, 0(t4)               # Gán hướng tăng
        jal     enable_keypad_interrupt
        j       end_process

set_decrement:
        li      a0, -1
        la      t4, direction
        sw      a0, 0(t4)               # Gán hướng giảm
        jal     enable_keypad_interrupt
        j       end_process

decrease_period:
        la      t4, period
        lw      t5, 0(t4)
        li      t6, 500                 # Giảm từng bước 100
        sub     t5, t5, t6
        li      a3, 500                 # Giới hạn tối thiểu
        bge     t5, a3, save_decrease
        li      t5, 500                 # Nếu nhỏ hơn 100 thì giữ ở 100
save_decrease:
```

```
        sw      t5, 0(t4)
        jal     enable_keypad_interrupt
        j       end_process

increase_period:
        la      t4, period
        lw      t5, 0(t4)
        li      t6, 100             # Tăng từng bước 100
        add     t5, t5, t6
        li      a3, 2000            # Giới hạn tối đa
        ble     t5, a3, save_increase
        li      t5, 2000            # Nếu lớn hơn 2000 thì giữ ở 2000
save_increase:
        sw      t5, 0(t4)
        jal     enable_keypad_interrupt
        j       end_process




end_process:
        # Khôi phục ngữ cảnh
        lw      a7, 12(sp)
        lw      a2, 8(sp)
        lw      a1, 4(sp)
        lw      a0, 0(sp)
        addi    sp, sp, 16
```

```
        uret
# ---------------------------------------------------------
# Bật lại ngắt bàn phím sau mỗi lần xử lý để nhận phím mới
# ---------------------------------------------------------
enable_keypad_interrupt:
        li      t1, IN_ADDRESS_HEXA_KEYBOARD
        li      t2, 0x80                # Bit kích hoạt ngắt
        sb      t2, 0(t1)
        ret
# ---------------------------------------------------------------
# Cập nhật hiển thị LED 7 đoạn
# ---------------------------------------------------------------
update_display:
        addi    sp, sp, -12
        sw      ra, 0(sp)
        sw      a0, 4(sp)
        sw      a1, 8(sp)

        lw      a0, count

        # Lấy chữ số hàng đơn vị
        li      a1, 10
        rem     a1, a0, a1      # a1 = count % 10
        la      t0, digit_patterns
        add     t0, t0, a1
        lb      a1, 0(t0)       # Lấy mẫu cho chữ số phải
```

```
    li      t0, SEVENSEG_RIGHT
    sb      a1, 0(t0)       # Hiển thị chữ số phải

    # Lấy chữ số hàng chục
    li      a1, 10
    div     a1, a0, a1      # a1 = count / 10
    la      t0, digit_patterns
    add     t0, t0, a1
    lb      a1, 0(t0)       # Lấy mẫu cho chữ số trái
    li      t0, SEVENSEG_LEFT
    sb      a1, 0(t0)       # Hiển thị chữ số trái

    lw      a1, 8(sp)
    lw      a0, 4(sp)
    lw      ra, 0(sp)
    addi    sp, sp, 12
    jr      ra
```

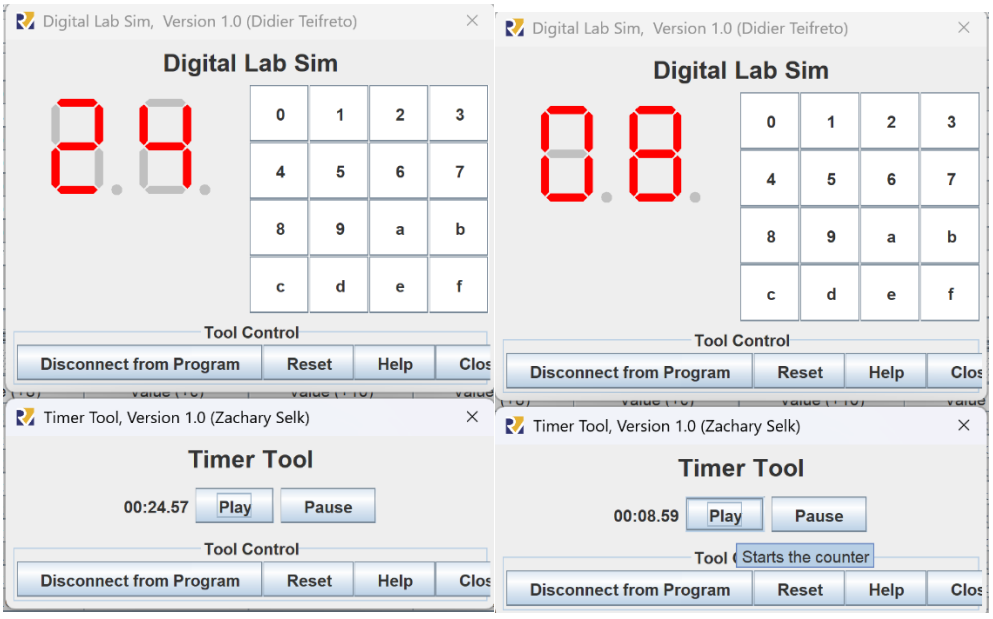
- **Các bước thực hiện của chương trình**:

- Khởi tạo các hằng số cần thiết và chuẩn bị sẵn các biến kiểu word chứa giá trị sẽ sử dụng trong quá trình xử lý.
- Khi chương trình bắt đầu, tiến hành thiết lập ngắt cho timer và keyboard, sau đó kích hoạt các ngắt.
- Tiếp theo, thiết lập giá trị ban đầu cho timer và hiển thị số 00 lên màn hình.
- Khi xảy ra ngắt, chương trình sẽ lưu trạng thái các thanh ghi vào stack, sau đó kiểm tra nguyên nhân gây ra ngắt:
    - Nếu là ngắt từ timer:
      Kiểm tra hướng đếm hiện tại là tăng hay giảm. Nếu bộ đếm vượt giới

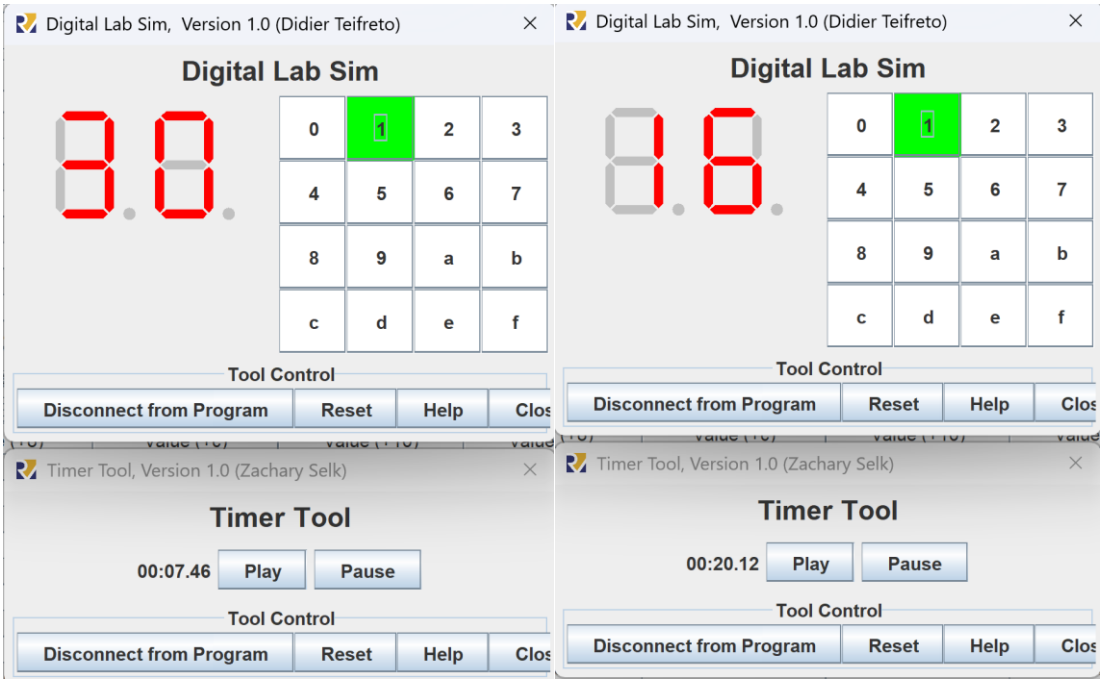
hạn, sẽ thực hiện quay vòng (từ 99 về 0 hoặc từ 0 về 99). Cập nhật giá trị mới của biến đếm và hiển thị lên màn hình. Sau đó, tăng khoảng ngắt tiếp theo thông qua điều chỉnh chu kỳ timer, thiết lập lại ngắt và quay về chương trình chính.

- Nếu là ngắt từ bàn phím:
  Thực hiện quét hàng – cột để xác định phím nào được nhấn. Dựa trên mã phím, xác định chức năng tương ứng:
  - Phím 0: chuyển bộ đếm sang chế độ tăng dần
  - Phím 1: chuyển bộ đếm sang chế độ giảm dần
  - Phím 4: giảm chu kỳ timer
  - Phím 5: tăng chu kỳ timer
    Với phím 0 và 1, chương trình đặt lại hướng đếm, bật lại ngắt và quay lại chương trình chính.
    Với phím 4 và 5, chương trình điều chỉnh biến khoảng chu kỳ, sau đó quay lại chương trình, khôi phục ngắt và tiếp tục chờ nhận phím mới.

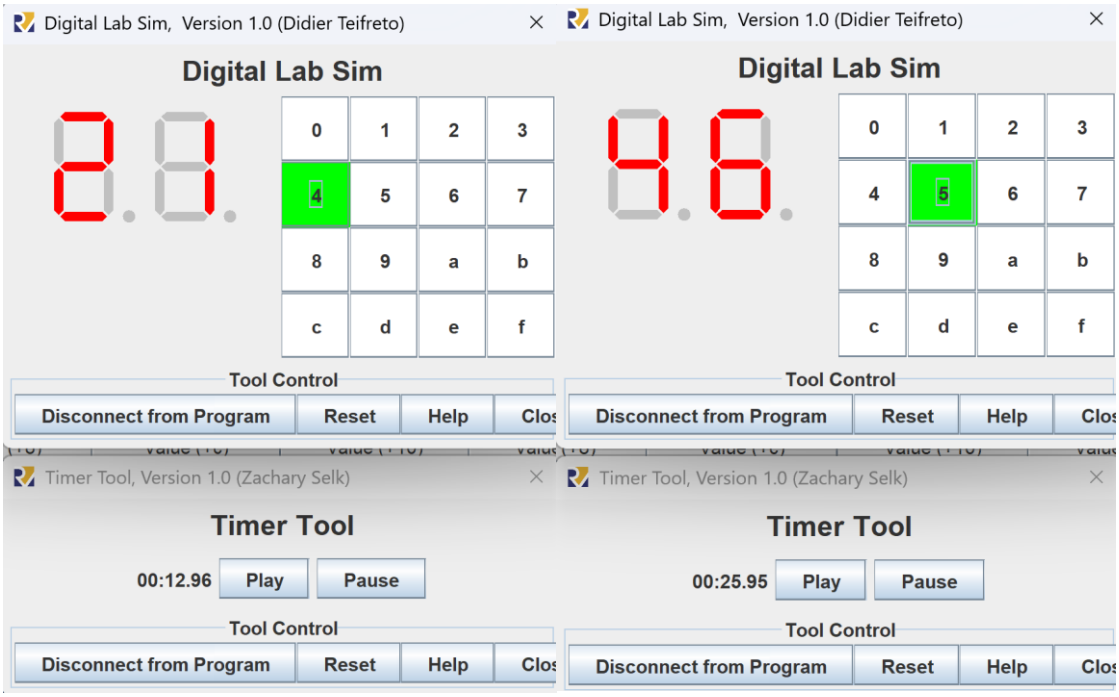
- **Kết quả của chương trình:**
  + Với mode 0 tăng:



+ Với mode 1 giảm:



+ Với mode 4, 5 tăng giảm chu kì:



Kết quả đúng kỳ vọng