Thực hành kiến trúc máy tính

Báo cáo thực hành

Bài 11. Lập trình xử lý ngắt

|  |  |
| --- | --- |
| Họ Tên | Lê Thành An |
| MSSV | 20235631 |

**ASSIGNMENT 4**

ĐOẠN MÃ :

.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

sw t2, 0(t1)

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

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

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

loop:

nop

nop

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

Kết quả:

Ảnh có chứa văn bản, ảnh chụp màn hình, biểu đồ, Song song

Nội dung do AI tạo ra có thể không chính xác.

Thanh ghi:

| **Thời điểm** | **utvec** | **uie** | **ustatus** | **PC (giả định)** | **Ghi chú** |
| --- | --- | --- | --- | --- | --- |
| Trước cấu hình ngắt | Không xác định | 0 | 0 | main | Bắt đầu chương trình |
| Sau csrrs utvec, t0 | handler addr | 0 | 0 | main+4 | Gán địa chỉ trình xử lý ngắt |
| Sau csrrs uie, t1 | Không đổi | 0x100 | 0 | main+12 | Mở ngắt ngoài người dùng (bit 8) |
| Sau csrrsi uie, 0x10 | Không đổi | 0x110 | 0 | main+16 | Mở ngắt timer (bit 4) |
| Sau csrrsi ustatus, 1 | Không đổi | 0x110 | 0x1 | main+20 | Cho phép tiếp nhận ngắt (Global interrupt enable) |
| Trước khi có ngắt | Không đổi | 0x110 | 0x1 | loop+12 (vị trí j loop) | Đang thực hiện vòng lặp vô tận |
| Khi có ngắt xảy ra | Không đổi | 0x110 | 0x0 | handler | PC nhảy đến handler, ustatus.UIE tự động bị xóa |
| Trong handler (ucause) | Không đổi | 0x110 | 0x0 | handler+20 | ucause chứa loại ngắt (0x80000004 hoặc 0x80000008) |
| Sau uret | Không đổi | 0x110 | 0x1 | loop+12 | PC quay lại tiếp tục thực thi, ustatus.UIE được khôi phục |

**ASSIGNMENT 5**

ĐOẠN MÃ :

.data

message: .asciz "Exception occurred.\n"

.text

main:

try:

la t0, catch

csrrw zero, utvec, t0

# Set utvec to the handler address

csrrsi zero, ustatus, 1

# Set interrupt enable bit in ustatus

lw zero, 0

# Trigger trap for Load access fault

finally:

li a7, 10

# Exit the program

ecall

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

uret

Kết quả:

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, hàng

Nội dung do AI tạo ra có thể không chính xác.

Thanh ghi:

| **Thời điểm** | **utvec** | **ustatus** | **uepc** | **ucause** | **PC (giả định)** | **Ghi chú** |
| --- | --- | --- | --- | --- | --- | --- |
| Trước cấu hình exception | Không xác định | 0 | Không xác định | Không xác định | main | Bắt đầu chương trình |
| Sau csrrw utvec, t0 | catch addr | 0 | Không xác định | Không xác định | try+4 | Gán địa chỉ trình xử lý exception |
| Sau csrrsi ustatus, 1 | Không đổi | 0x1 | Không xác định | Không xác định | try+8 | Cho phép exception (UIE = 1) |
| Trước lệnh gây exception | Không đổi | 0x1 | Không xác định | Không xác định | try+12 | Trước lệnh lw zero, 0 |
| Khi exception xảy ra | Không đổi | 0x0 | try+12 | 0x5 | catch | PC nhảy đến catch, ustatus.UIE tự động bị xóa, uepc lưu địa chỉ lệnh lỗi, ucause = 5 (load access fault) |
| Sau ecall hiển thị thông báo | Không đổi | 0x0 | try+12 | 0x5 | catch+12 | Đã hiển thị thông báo exception |
| Sau csrrw uepc, t0 | Không đổi | 0x0 | finally addr | 0x5 | catch+16 | Gán địa chỉ của finally vào uepc |
| Sau uret | Không đổi | 0x1 | Không đổi | 0x5 | finally | PC nhảy đến finally, ustatus.UIE được khôi phục |

**ASSIGNMENT 6**

ĐOẠN MÃ :

.data

overflow\_msg:

.asciz "Error: Integer overflow occurred!\n"

.text

.globl main

main:

# --- Khởi tạo ngăn xếp cho main và lưu ra ---

addi sp, sp, -16

sw ra, 12(sp)

# --- Thiết lập ISR và bật ngắt mềm ---

la t0, overflow\_isr

csrrw zero, utvec, t0 # utvec <- &overflow\_isr

csrrsi zero, ustatus, 1 # UIE = 1

csrrsi zero, uie, 1 # USIE = 1

# --- Khởi tạo số và cộng ---

li s1, 0x7FFFFFFF

li s2, 1

add s3, s1, s2

# INT\_MAX

# --- Kiểm tra overflow ---

xor t0, s1, s2

srli t0, t0, 31

bne t0, zero, no\_overflow

xor t0, s1, s3

srli t0, t0, 31

beq t0, zero, no\_overflow

# --- Overflow → kích soft-irq ---

csrrsi zero, uip, 1

nop

wait\_irq:

j wait\_irq

no\_overflow:

# In kết quả bình thường

mv a0, s3

li a7, 1

ecall

li a7, 11

li a0, '\n'

ecall

# Restore và exit

lw ra, 12(sp)

addi sp, sp, 16

li a7, 10

ecall

# ==========================

# ISR: overflow\_isr

# ==========================

overflow\_isr:

# --- Sao lưu ngữ cảnh (alignment 16) ---

addi sp, sp, -16

sw ra, 12(sp)

sw t0, 8(sp)

sw t1, 4(sp)

# --- Lấy ucause vào t0 ---

csrrc t0, ucause, zero

# t0 = ucause

# --- Kiểm tra interrupt vs exception ---

li t1, 0x80000000

and t1, t0, t1

beq t1, zero, end\_isr # nếu không phải interrupt, bỏ qua

# --- Lấy mã cause (lower 4 bits) ---

li t1, 0xF

and t0, t0, t1

bne t0, zero, end\_isr # nếu không phải soft-irq (code=0), bỏ qua

# --- Clear USIP để tránh lặp lại IRQ ---

csrrci zero, uip, 1

# --- In thông báo lỗi ---

la a0, overflow\_msg

li a7, 4

ecall

# --- Exit ---

li a7, 10

ecall

end\_isr:

# --- Khôi phục ngữ cảnh ---

lw t1, 4(sp)

lw t0, 8(sp)

lw ra, 12(sp)

addi sp, sp, 16

uret

Kết quả:

Ảnh có chứa ảnh chụp màn hình, văn bản, phần mềm, hàng

Nội dung do AI tạo ra có thể không chính xác.

| **Thời điểm** | **utvec** | **ustatus** | **uie** | **uip** | **ucause** | **uepc** | **PC (giả định)** | **Ghi chú** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Bắt đầu chương trình | Không xác định | 0 | 0 | 0 | Không xác định | Không xác định | main | Bắt đầu chương trình |
| Sau csrrw utvec, t0 | overflow\_isr addr | 0 | 0 | 0 | Không xác định | Không xác định | main+8 | Gán địa chỉ ISR |
| Sau csrrsi ustatus, 1 | Không đổi | 0x1 | 0 | 0 | Không xác định | Không xác định | main+12 | Enable global interrupt (UIE = 1) |
| Sau csrrsi uie, 1 | Không đổi | 0x1 | 0x1 | 0 | Không xác định | Không xác định | main+16 | Enable software interrupt (USIE = 1) |
| Sau khi tính toán và phát hiện overflow | Không đổi | 0x1 | 0x1 | 0 | Không xác định | Không xác định | main+40 | Đã phát hiện tràn số khi cộng 0x7FFFFFFF + 1 |
| Sau csrrsi uip, 1 | Không đổi | 0x1 | 0x1 | 0x1 | Không xác định | Không xác định | main+44 | Kích hoạt software interrupt (USIP = 1) |
| Trước khi nhảy vào ISR | Không đổi | 0x1 | 0x1 | 0x1 | Không xác định | Không xác định | wait\_irq | Đang chờ trong vòng lặp |
| Khi ngắt được xử lý | Không đổi | 0x0 | 0x1 | 0x1 | 0x80000000 | wait\_irq addr | overflow\_isr | PC nhảy đến overflow\_isr, ustatus.UIE = 0, ucause = 0x80000000 (software interrupt) |
| Sau csrrc ucause, zero | Không đổi | 0x0 | 0x1 | 0x1 | 0x80000000 | wait\_irq addr | overflow\_isr+16 | Đọc giá trị ucause |
| Sau csrrci uip, 1 | Không đổi | 0x0 | 0x1 | 0x0 | 0x80000000 | wait\_irq addr | overflow\_isr+32 | Xóa bit USIP để tránh ngắt lặp lại |
| Khi kết thúc chương trình | Không đổi | 0x0 | 0x1 | 0x0 | 0x80000000 | wait\_irq addr | (Kết thúc) | Chương trình kết thúc với ecall (a7 = 10) |

**ASSIGNMENT 7**

ĐOẠN MÃ :

# RISC-V Program: Counter with Timer and Keypad

# Displays a counter (00-99) on two 7-segment displays

# Controls:

# Button 0: Count up mode

# Button 1: Count down mode

# Button 4: Decrease cycle (increase speed)

# Button 5: Increase cycle (decrease speed)

# Memory-mapped I/O addresses

.eqv IN\_ADDRESS\_HEXA\_KEYBOARD 0xFFFF0012 # Input from hexadecimal keyboard

.eqv OUT\_ADDRESS\_HEXA\_KEYBOARD 0xFFFF0014 # Output from hexadecimal keyboard

.eqv TIMER\_NOW 0xFFFF0018 # Current time

.eqv TIMER\_CMP 0xFFFF0020 # Time for next interrupt

.eqv MASK\_CAUSE\_TIMER 4 # Timer interrupt cause code

.eqv MASK\_CAUSE\_KEYPAD 8 # Keyboard interrupt cause code

.eqv SEVENSEG\_LEFT 0xFFFF0011 # Left 7-segment display

.eqv SEVENSEG\_RIGHT 0xFFFF0010 # Right 7-segment display

.data

# Configuration and state variables

count: .word 0 # Current counter value (0-99)

count\_mode: .word 0 # 0: count up, 1: count down

cycle\_time: .word 1000 # Initial cycle time

# 7-segment display codes for digits 0-9

seg\_codes: .byte 0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07, 0x7F, 0x6F

# Debug messages

msg\_keypad: .asciz "Keypad button pressed: "

msg\_key\_code: .asciz "Key scan code: 0x"

msg\_timer: .asciz "Timer event! Count: "

msg\_mode\_up: .asciz "Mode changed: Count Up\n"

msg\_mode\_down: .asciz "Mode changed: Count Down\n"

msg\_cycle\_inc: .asciz "Cycle time increased to: "

msg\_cycle\_dec: .asciz "Cycle time decreased to: "

newline: .asciz "\n"

.text

main:

# Set up the interrupt handler

la t0, handler

csrrs zero, utvec, t0

# Enable external and timer interrupts

li t1, 0x100 # External interrupt enable (bit 8)

csrrs zero, uie, t1

csrrsi zero, uie, 0x10 # Timer interrupt enable (bit 4)

# Enable global interrupts

csrrsi zero, ustatus, 1

# Enable keypad interrupt

li t1, IN\_ADDRESS\_HEXA\_KEYBOARD

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

sb t2, 0(t1)

# Set initial timer comparison value

li t1, TIMER\_NOW

lw t2, 0(t1) # Get current time

la t3, cycle\_time

lw t3, 0(t3)

add t2, t2, t3 # Add cycle time

li t1, TIMER\_CMP

sw t2, 0(t1) # Set comparison value

# Initialize display

jal update\_display

# Main loop - do nothing, interrupts handle everything

loop:

nop

j loop

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

# Interrupt Service Routine

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

handler:

# Save context

addi sp, sp, -28

sw ra, 0(sp)

sw a0, 4(sp)

sw a1, 8(sp)

sw a2, 12(sp)

sw t0, 16(sp)

sw t1, 20(sp)

sw a7, 24(sp)

# Get interrupt cause

csrr a1, ucause

li a2, 0x7FFFFFFF

and a1, a1, a2 # Clear MSB to get cause value

# Check interrupt type

li a2, MASK\_CAUSE\_TIMER

beq a1, a2, timer\_isr

li a2, MASK\_CAUSE\_KEYPAD

beq a1, a2, keypad\_isr

j end\_handler

timer\_isr:

# Handle timer interrupt - update counter

la t1, count\_mode

lw t0, 0(t1)

bnez t0, count\_down # If mode is 1, count down

count\_up:

la t1, count

lw t0, 0(t1)

addi t0, t0, 1 # Increment counter

li t1, 100

rem t0, t0, t1 # Keep in range 0-99

la t1, count

sw t0, 0(t1)

j timer\_continue

count\_down:

la t1, count

lw t0, 0(t1)

addi t0, t0, -1 # Decrement counter

bltz t0, wrap\_to\_99 # If negative, wrap to 99

la t1, count

sw t0, 0(t1)

j timer\_continue

wrap\_to\_99:

li t0, 99

la t1, count

sw t0, 0(t1)

timer\_continue:

# Update display

jal update\_display

# Set next timer interrupt

li a0, TIMER\_NOW

lw a1, 0(a0) # Get current time

la t1, cycle\_time

lw t0, 0(t1)

add a1, a1, t0 # Add cycle time

li a0, TIMER\_CMP

sw a1, 0(a0) # Set new comparison value

j end\_handler

keypad\_isr:

# Get key scan code by scanning each row

# First row

li t0, IN\_ADDRESS\_HEXA\_KEYBOARD

li t2, 0x81 # Check row 1 (0x1) and re-enable interrupt (0x80)

sb t2, 0(t0)

li t0, OUT\_ADDRESS\_HEXA\_KEYBOARD

lb t1, 0(t0)

bnez t1, process\_key

# Second row

li t0, IN\_ADDRESS\_HEXA\_KEYBOARD

li t2, 0x82 # Check row 2 (0x2) and re-enable interrupt (0x80)

sb t2, 0(t0)

li t0, OUT\_ADDRESS\_HEXA\_KEYBOARD

lb t1, 0(t0)

bnez t1, process\_key

# Third row

li t0, IN\_ADDRESS\_HEXA\_KEYBOARD

li t2, 0x84 # Check row 3 (0x4) and re-enable interrupt (0x80)

sb t2, 0(t0)

li t0, OUT\_ADDRESS\_HEXA\_KEYBOARD

lb t1, 0(t0)

bnez t1, process\_key

# Fourth row

li t0, IN\_ADDRESS\_HEXA\_KEYBOARD

li t2, 0x88 # Check row 4 (0x8) and re-enable interrupt (0x80)

sb t2, 0(t0)

li t0, OUT\_ADDRESS\_HEXA\_KEYBOARD

lb t1, 0(t0)

beqz t1, end\_handler # No key pressed, exit

process\_key:

# Print key scan code

li a7, 4

la a0, msg\_key\_code

ecall

mv a0, t1

li a7, 34 # Print in hex

ecall

li a7, 4

la a0, newline

ecall

# Map scan code to button number

# Row 1: 0x11(0), 0x21(1), 0x41(2), 0x81(3)

# Row 2: 0x12(4), 0x22(5), 0x42(6), 0x82(7)

# Row 3: 0x14(8), 0x24(9), 0x44(10), 0x84(11)

# Row 4: 0x18(12), 0x28(13), 0x48(14), 0x88(15)

# Check row 1 keys

li t0, 0x11

beq t1, t0, key\_0

li t0, 0x21

beq t1, t0, key\_1

li t0, 0x41

beq t1, t0, end\_handler # Key 2 - not used

li t0, 0x81

beq t1, t0, end\_handler # Key 3 - not used

# Check row 2 keys

li t0, 0x12

beq t1, t0, key\_4

li t0, 0x22

beq t1, t0, key\_5

li t0, 0x42

beq t1, t0, end\_handler # Key 6 - not used

li t0, 0x82

beq t1, t0, end\_handler # Key 7 - not used

# Other keys not used in this program

j end\_handler

key\_0: # Set count up mode

li t0, 0

la t1, count\_mode

sw t0, 0(t1)

# Debug message

li a7, 4

la a0, msg\_mode\_up

ecall

j end\_handler

key\_1: # Set count down mode

li t0, 1

la t1, count\_mode

sw t0, 0(t1)

# Debug message

li a7, 4

la a0, msg\_mode\_down

ecall

j end\_handler

key\_4: # Decrease cycle time (minimum 100)

la t1, cycle\_time

lw t0, 0(t1)

addi t0, t0, -100 # Decrease by 100

li t2, 100

blt t0, t2, set\_min\_cycle

sw t0, 0(t1)

# Debug message

li a7, 4

la a0, msg\_cycle\_dec

ecall

li a7, 1

mv a0, t0

ecall

li a7, 4

la a0, newline

ecall

j end\_handler

set\_min\_cycle:

li t0, 100

la t1, cycle\_time

sw t0, 0(t1)

# Debug message

li a7, 4

la a0, msg\_cycle\_dec

ecall

li a7, 1

li a0, 100

ecall

li a7, 4

la a0, newline

ecall

j end\_handler

key\_5: # Increase cycle time (maximum 3000)

la t1, cycle\_time

lw t0, 0(t1)

addi t0, t0, 100 # Increase by 100

li t2, 3000

bgt t0, t2, set\_max\_cycle

sw t0, 0(t1)

# Debug message

li a7, 4

la a0, msg\_cycle\_inc

ecall

li a7, 1

mv a0, t0

ecall

li a7, 4

la a0, newline

ecall

j end\_handler

set\_max\_cycle:

li t0, 3000

la t1, cycle\_time

sw t0, 0(t1)

# Debug message

li a7, 4

la a0, msg\_cycle\_inc

ecall

li a7, 1

li a0, 3000

ecall

li a7, 4

la a0, newline

ecall

end\_handler:

# Re-enable keyboard interrupt

li t0, IN\_ADDRESS\_HEXA\_KEYBOARD

li t1, 0x80 # Set bit 7 to enable interrupt

sb t1, 0(t0)

# Restore context

lw ra, 0(sp)

lw a0, 4(sp)

lw a1, 8(sp)

lw a2, 12(sp)

lw t0, 16(sp)

lw t1, 20(sp)

lw a7, 24(sp)

addi sp, sp, 28

uret

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

# Function: update\_display

# Updates both 7-segment displays with the current counter value

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

update\_display:

# Save context

addi sp, sp, -16

sw ra, 0(sp)

sw t0, 4(sp)

sw t1, 8(sp)

sw t2, 12(sp)

# Get counter value

la t1, count

lw t0, 0(t1)

# Calculate tens digit

li t1, 10

div t2, t0, t1 # t2 = tens digit

# Get segment code for tens digit

la t1, seg\_codes

add t1, t1, t2

lb a0, 0(t1)

# Display tens digit on left display

jal SHOW\_7SEG\_LEFT

# Calculate ones digit

la t1, count

lw t0, 0(t1)

li t1, 10

rem t2, t0, t1 # t2 = ones digit

# Get segment code for ones digit

la t1, seg\_codes

add t1, t1, t2

lb a0, 0(t1)

# Display ones digit on right display

jal SHOW\_7SEG\_RIGHT

# Restore context

lw ra, 0(sp)

lw t0, 4(sp)

lw t1, 8(sp)

lw t2, 12(sp)

addi sp, sp, 16

jr ra

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

# Function: SHOW\_7SEG\_LEFT

# Displays a value on the left 7-segment display

# param[in] a0: value to show

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

SHOW\_7SEG\_LEFT:

li t0, SEVENSEG\_LEFT

sb a0, 0(t0)

jr ra

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

# Function: SHOW\_7SEG\_RIGHT

# Displays a value on the right 7-segment display

# param[in] a0: value to show

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

SHOW\_7SEG\_RIGHT:

li t0, SEVENSEG\_RIGHT

sb a0, 0(t0)

jr ra

Kết quả:

Ảnh có chứa văn bản, ảnh chụp màn hình, màn hình, số

Nội dung do AI tạo ra có thể không chính xác.